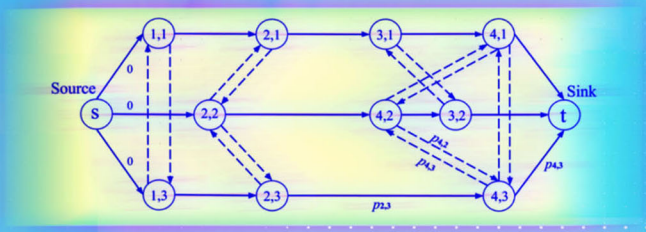
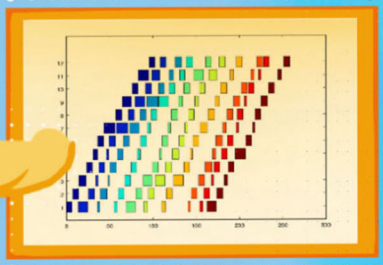
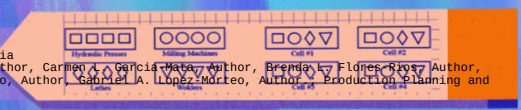




PRODUCTION PLANNING AND SCHEDULING FOR LOT PROCESSING

Larysa Burtseva
Rainier Romero
Brenda L. Flores-Rios
Eddy M. Delgado-Arana
Gabriel A. Lopez-Morteo
Frank Werner
Carmen L. Garcia-Mata
Victor Yaurima
Felix F. Gonzalez-Navarro



Production Planning and Scheduling for Lot Processing

Production Planning and Scheduling for Lot Processing

By

Larysa Burtseva,
Frank Werner,
Rainier Romero,
Carmen L. Garcia-Mata,
Brenda L. Flores-Rios,
Victor Yaurima,
Eddy M. Delgado-Arana,
Felix F. Gonzalez-Navarro
and Gabriel A. Lopez-Morteo

Cambridge
Scholars
Publishing

Production Planning and Scheduling for Lot Processing

By Larysa Burtseva, Frank Werner, Rainier Romero,
Carmen L. Garcia-Mata, Brenda L. Flores-Rios, Victor Yaurima,
Eddy M. Delgado-Arana, Felix F. Gonzalez-Navarro
and Gabriel A. Lopez-Morteo

This book first published 2022

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-5275-8502-6
ISBN (13): 978-1-5275-8502-7

To the Memory of

Anatoliy Panishev

Scientist, Teacher, Person

TABLE OF CONTENTS

List of Figures

List of Tables

PREFACE

Advanced plants with High Mix and High Volume of the production have a common characteristic: the manufacturing process is organized by lots and batches to reach a maximal productivity maintaining a high flexibility to react on challenges provoked by world market changes and unexpected complications of the business, such as, e.g., effects of the consequences of a pandemic. Manufacturing with lot processing has two evident planning aspects: organization of the production and the modeling of the plant operations.

The organization of the production involves the production philosophy and culture, integrating a variety of paradigms and concepts. The goal of modeling the plant operations is the optimization of the planning and scheduling. Both these features are discussed in the book.

The planning systems have been changed since the creation of the Toyota production system, which gave rise to new production paradigms, such as Just-In-Time and Single Minute Exchange of Die, which reduced drastically the inventory and setup times, respectively. In parallel, the methodologies considering the Group Technology, lot processing and batching were introduced into scheduling methods, both in practice and the approaches.

Scheduling theory is a rather new discipline. It started in the 1950s, and since then it attracted the attention of planners due to the high applicability of the results and of researchers due to the high computational complexity of the problems. This theory has been mainly written in the recent years. It is not yet completed, and the models considering production lots issues might be currently one of the most active research directions.

New perspectives appear by the challenges of the practice and the actual state of science in general. Therefore, the main idea of the book is to highlight the main topics of scheduling theory, which are related to lot processing, starting with its effect on the planning system, and making the principal accent on the optimization of the scheduling process in the shop floor.

This book is an intention to present a general landscape of the related theory in its actual state, remembering the pioneering works, giving historical sketches, and then describing the principal results and showing some directions for future research. The book does not include all

algorithms and methods developed, but gives a general state-of-the-art and references to investigate every mentioned topic more in detail.

The book consists of ten chapters.

Chapter 1 introduces production planning systems. The general model of the planning in a plant and its components are explained. A classification of production planning problems into strategic, tactical and operational planning levels as well as principal planning challenges are described, and an introduction into lean manufacturing, which has a wide application in manufacturing plants with lot processing, is given.

Chapters 2-4 describe general topics of planning and scheduling, but with an accent to special features when the production is realized in lots and batches of identical items.

Chapter 2 is dedicated to the inventory management in plants with serial manufacturing. Various related concepts are introduced. The principal components of the inventory are described: raw material, WIP and finished goods. A formula to calculate the average inventory level is given together with an example. The Push/Pull policies, which are used to control the inventory level, are explained. A big part of this chapter is dedicated to Material Requirements Planning, which is a computer-based production planning and inventory control system for the production and scheduling in the plant. A historical sketch of its development is given, also some features and extensions. Some aspects of the Japanese Toyota production system are described, namely, the “zero concept”, CONWIP, the paradigms Just-In-Time and Just-In-Sequence. The kanban method is described in detail. The next concept is the Optimized Production Technology, and then the Drum-Buffer-Rope method, which can be used for its implementation, is described. An analysis of the available methods of inventory optimization finishes the chapter.

In Chapter 3, the standard notations and the basic scheduling models, which are used in the book, are described. A survey of problems and solutions, which consider lot processing, is given for the principal machine environments and shop conditions.

In Chapter 4, machine setup times are considered. The reduction of setup times is the main purpose of processing the items in batches and lots. The basic structure and a classification of setup times are explained. Then it is shown how lot scheduling problems are formulated for different environment configurations and different kinds of setups. The problems with sequence-dependent setup times are considered in most detail. This kind of setup is typical for modern industries, where effect of setups is substantial. New directions in this research area are described, such as machine/resource-dependent setup times: time-dependent setup times, past-

sequence-dependent (p-s-d) setup times, including learning/deteriorating effects. Specially, there were considered family/batch setup times. Then a systematic classification of changeovers and setup/changeover reduction schemes are described. Finally, the SMED/OTED method is explained.

Chapter 5 takes the most attention in the book, due to numerous methods and approaches. The chapter begins with explaining the history, the first authors and the philosophy of the Group Technology, which is interesting by itself. This theory started in between 1920 and 1930 and looks complete at the moment, nevertheless, one can note a continuous interest of researchers in this methodology due to its big potential. First, the part family concept is introduced which is followed by grouping methods. The famous Opitz classification system is explained in detail. There is also given a resume of the most famous classification systems used, including the developers and accessible references for a consultation. Then the Burbidge production flow analysis is explained. The next basic approach of Group Technology is the cluster analysis. It is used for the formation of the production cells, which causes frequently difficulties in the plant. Various methodologies are illustrated by examples. This chapter is finished with a description of the cellular manufacturing approach and an explanation of the cell formation problem.

Chapter 6 is dedicated to batch scheduling. The difference and similarity between batch and lot terms is explained at the beginning. Then batching is studied by an example of semiconductor manufacturing, where batching appears with different features, and the burn-in operation gives rise to basic batching concepts, such as batch machine. Possible batching models for different environments are described. Computational approaches to form the cell architecture finish the chapter.

Lot streaming and lot sizing are approaches, which are directly dedicated to the modeling of the problems that consider lot processing. In Chapter 7, the concepts and problems of lot streaming and job splitting are explained, and the effect of these phenomena on the scheduling models is shown. Lot splitting problems require a special system of notations. It is described in this chapter, and an analysis of the corresponding models is given for different environments.

In Chapter 8, the background of the lot sizing problem is explained. It started in 1913 after the paper “How Many Parts to Make at Once” published by Eng. Ford Whitman Harris, where the famous Harris' square-root formula for the order quantity was introduced. It inspired a family of lot sizing models, which is not complete yet.

Rescheduling is a hot topic in modern scheduling theory due to the required high computational effort in connection with modeling and special

theories, which appeared recently. Chapter 9 describes these theories and possible approaches to the problem modeling.

Chapter 10 concludes the book. It is dedicated to three practical problems on different aspects of lot processing, which were solved by the book authors for big corporations of the region.

All chapters are finished by some concluding remarks, which contain a brief analysis of the state-of-the-art, references to available surveys, and some recommendations for future researches.

The group of authors jointly worked on research issues and participated in postgraduate and editorial projects in the lot scheduling area, referring to affiliations in Mexico: the Universidad Autónoma de Baja California (UABC), Universidad Politécnica de Baja California (UPBC), Tecnológico Nacional de México-Instituto Tecnológico de Chihuahua (TecNM/ITCH), Universidad Estatal de Sonora (UES), Skyworks Solutions, Inc., and the Faculty of Mathematics of the Otto-von-Guericke University Magdeburg (OVGU) in Germany.

The authors are grateful to the administration of the Instituto de Ingeniería of the UABC for the support of our research projects, to many people from the corporations COTO and Skyworks, who dedicated their time and interest to reach a result following our recommendations, and to the UABC students Paola Velazquez, Jaqueline Contreras, Laura Bravo, and Luis Martinez, who helped in the design of this book.

The book contains a wide collection of theories, methods and approaches, which are developed at the date, and it is directed to production planners for practical use. This book can be used by beginners and for a deep study because it considers not only algorithmic aspects but also the problem philosophy and history. It can also be useful for students of all university levels, which are interested in studying this subject. We hope that it will also find the interest of researchers in this area due to the wide analysis of the state-of-the-art of the studied subjects, which is systematized according to the book structure.

Authors

CHAPTER ONE

INTRODUCTION

In big manufacturing plants with lot processing, planning and scheduling activities play nowadays a key role to provide an efficient functioning of these complex organisms from a machine to an entire business operation. In this area, one can meet plenty of new and traditional theories, models and paradigms. The researcher and planner communities are interested in identifying the actual state-of-the-art to maintain an advanced production corresponding to the challenges of a modern competition level. In the following, some relevant concepts and their place in an entire planning system of a modern plant are presented and studied as production programming subjects. A general view onto such a system is given at the beginning. Scheduling is considered as an extension of the planning activities. This area belongs to the hot topics of modern science. Therefore, various concepts and notations still require commonly accepted definitions.

1.1 Planning and scheduling for manufacturing plants with lot processing

In advanced production systems, such as semiconductor and electronic components industries, as well as at diverse assembly lines, where the manufactured products have the characteristics of *high volume and high mixture* (HV/HM) of nomenclatures, the product components are processed by *lots* (pallets, containers, boxes) of many identical items. Another immanent characteristic of such systems is a *multi-stage production process* with parallel machines or workstations at each stage. Typically, a High Tech (HT) company with characteristics HV/HM has dozens and even hundreds of machines of different years and brands at several stages. The lot processing and the use of *batch machines* for parallel working are typical for modern manufacturing systems with mass production. The complexity of the production process organization is characterized by a *high frequency of changes* in the nomenclature of the ready products and components. This consideration together with the occurrence of unexpected events and

priority changes, require a *flexibility* of the planning and a high level of its automation to support the decision-making.

In such an environment, the operation by lots has an effect on various aspects of planning and scheduling, such as the system of production and inventory control, which reflects the *philosophy of planning* as a leading mode of the manufacturing organization.

Processing by lots also implies other typical features. The lot processing machines require a *setup* service, which is necessary to handle the next lot. It includes an adjustment and preparation of the recourses. When adjacent lots have a big technological difference (size, shape, row material, machine programming, etc.), a setup takes a considerable time while similar products may have a minimal setup time, which can be insignificant, even negligible. The majority of scheduling problems, which involve intermediate setups, have a high computational complexity and represent an interest for the researchers since the 1960s until now, especially when lot processing is considered. The setup duration has a direct connection with *lot batching*, which is a usual practice in the plants. The lots of the same or similar products are coupled for processing to eliminate or reduce the setup times used for the adjustment of the machines.

A theoretical and practical interest on this issue can be met in those models, where *lot splitting* is permitted, i.e., one lot is allowed to be split into sublots to be processed in parallel on a batch machine or on various parallel machines, to accelerate the lot output time. In many models, the *sublot size* is not evident and must be optimized. When a sublot requires a considerable setup time on a machine, the complexity of the problem increases. Merging of sublots is another important aspect of lot processing. The scheduling theory offers two principal concepts to describe and solve problems that involve the treatment of lots: *lot streaming* and *job batching*. These commonly appear together in problems, where the jobs are allowed to be split.

A typical situation in many plants is *reworking* and *rescheduling* of those items, which did not pass the testing stage. This fact and the existence of scrap, which usually is not considered in theoretical studies or is taken as a constant percentage of the plant output, is still an open research area.

The models, which contemplate lot processing, have differences in comparison with traditional job scheduling. The job notation depends on the problem assumptions and requires an explicit interpretation. One lot, one work order or the whole customer's demand may be treated as a job. The typical scheduling notations, e.g. those introduced by Pinedo (2008, 56), are insufficient for a formal description of such problems and therefore, they must be revised and updated according to the lot processing assumptions.

The planning problem in the plant includes the management and control of inventory, in particular the so-called Work-In-Process (WIP). Sometimes, the term work in progress is used with the same abbreviation. The last one describes the costs of unfinished goods that remain in the manufacturing process while work-in-process refers to materials that are turned into goods within a short period. The terms *Work-In-Progress* and *Work-In-Process* are used interchangeably referring to products midway through the plant or assembly lines. [1]

From the one side, an excessive WIP requires an additional shop floor space to be stocked up. This delays also the output of finished goods and elevates the production costs. Nevertheless, some quantity of WIP is necessary to balance the capacities of the stages and to equilibrate unexpected situations. The size of the WIP and the intermediate buffers between the production stages require also the attention of the planner.

The planning and scheduling activities in the companies with lot processing are subjects of intensive research due to the diversity of the problems and the high cost of a decision taken. Actually, one can observe a growing interest in the industries to the results of theoretical researches and the stimulation of such researches in the companies. Big companies employ specialists in solving the planning problems and dedicate a considerable attention to the preparation of proper planners in high-level education schools.

The enumerated subjects and relevant problems are discussed below in detail. This book pretends to analyze the available literature with the goal to extract the basic topics and approaches used for the production management in manufacturing plants with lot processing. In this way, it tries to reduce the gap between theory and practical necessities.

1.2 Production chain

The production process represents the transformation of the raw materials into the final products, usually through a series of steps producing and consuming intermediate products and components. Raw materials, intermediate products and finished goods are often inventoried, allowing the production and consumption in different sizes and production stages. Each transformation requires several input components and products and has one or more outputs. The raw materials are acquired by suppliers, and the final products are sold to external customers. Sometimes, intermediate products

[1] Investopedia.com: Work in Progress vs. Work in Process: What's the Difference? Accessed 09.06.2020.

are also sold (spare parts, etc.). This general definition of the production as a transformation process is shown in Fig. 1-1. The material inventory is drawn by triangles, the transformation activities are denoted by circles, and the flow of the materials through the process is indicated by arrows as the inputs or outputs during the transformation).

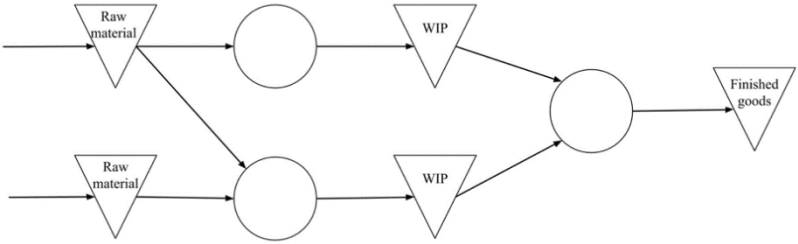


Fig. 1-1. Production process.

1.3 Production planning challenges

Production planning in plants is a process, by which the manufacturing departments organize the machinery resources over time in order to optimize their use and thus to achieve the highest possible productivity. Although the planning is a common problem for any company, it has not been systematically solved given the large number of variables that affect the decisions, which must be made. Therefore, the formalization of these activities is very difficult.

The production planning includes the management of the resources, which are necessary for the transformation of the raw material into the final products, in order to satisfy the customers in a most efficient or economical way. In other words, the decisions made during the planning are typically made for seeking a best balance between the financial objectives and the customer service objectives. The financial objectives are usually represented by the production costs for the machines, materials, labor, start-up costs, overhead, insurance, inventory costs, opportunity costs of the capital invested in the shares, etc. The objectives of the customer service are represented by the ability to offer the right product in the requested quantity, on the date and at the place promised.

When unexpected events occur, such as a quality problem, a process out of control, a change of priority, bad weather, road traffic, among many others, the exact fulfillment of the plan is prevented and its adjustment is required. Despite unexpected events and setbacks, which are usually out of

control and surely always appear, the plan must be done in detail. Therefore, the main challenge of a planner is not only carrying out the planning itself. He must also be able to react to unexpected events without losing the control over the production process and follow with continuous planning. The main challenge of the production planning is the possibility to discover the weak points in the plant.

There exist conflicting goals when choosing the best way to arrange the jobs:

- If a good use of the resources is sought, the completion time becomes worse; the cost of stocks and delays is increased;
- If the delivery time of the products is minimized, the current stock is less, but the use of resources is worse.

The final objective of a detailed production planning is to take decisions about the sequence of the jobs that each resource of the company processes in a smallest possible planning horizon. This activity is referred to as production programming, or scheduling, as it is shown in Fig. 1-2.

In addition, planning has other objectives:

- On-time delivery;
- Minimizing the completion time;
- Minimizing the production cost;
- Minimizing the WIP;
- Maximizing the overall effectiveness of the equipment;
- Minimizing the lead time.

Machine scheduling is a central component and the heart of the production planning responsibility. It is a base for shop loading, the management of the supply chain and the row material requirements, demand forecasting, project planning, etc. Therefore, the scheduling decisions take the main attention in the planning optimization. The main goal of production scheduling is the possibility of discovering the *bottleneck*s or the *capacity constraints* in the plant processes. Scheduling is considered a source of improvement projects, which try to eliminate the restrictions that hinder the search for the best job sequence.

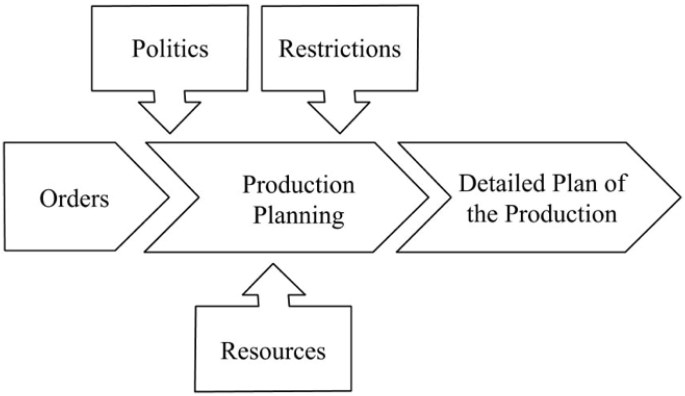


Fig. 1-2. General model of the planning.

A production schedule defines when the customer orders will be completed. Sometimes, it is necessary to fix a delivery date to the customer when a demand is fulfilled. In some cases, a too late or too optimistic date spoils the customer relationship or leads to penalties.

1.4 General scheme of the production planning

The production planning problems are classified into strategic, tactical, and operational planning levels.

Strategic problems deal with the management of changes in the production process and the acquisition of the resources needed to produce in the *long term*. This includes, for example, a combination of products, a perspective plan development, as well as the location, the supply chain design, and the investment decisions. The objective pursued in the solution of these strategic problems is to maintain an advantage and a competitive capacity in order to keep a growing rhythm. For this, it is necessary to propose long-term decisions using consolidated volumes of demand.

Tactical planning problems analyze the use of the resources in the *medium term*. The solutions are based on the information from the consolidated volumes. It consists, for example, in making decisions about the flow of materials, the size of the inventory, the capacity for use, the maintenance planning by the operational management. The usual objective at this stage is to improve the cost efficiency and the customer satisfaction.

Manufacturing Planning and Control (MPC) systems have been developed to address these complex planning environments, and they

integrate these multi-term (multi-level) planning problems into a management system. In these systems, the medium-term production planning is the decision on how to use the capacity and aggregated levels of the inventory to meet the projected medium-term demand during approximately one year.

The *Master Production Schedule* (MPS) represents a detailed short-term plan of the manufacturing of the final products in order to meet an expected demand and an aggregate customer order, taking into account the use of the capacity and the global inventory level obtained at the processing stage.

An MPS is a document that answers the following questions in detail:

- *What* products will be produced?
- *In what quantities* will they be produced?
- *When* will they be produced?

In an MPS, the time is usually expressed in weeks and corresponds to the duration of the production cycle. With the *Material Requirements Planning* (MRP, also referred to as MRP-I) system, short-term plans are established for all components (intermediate products and raw materials) of the final products considered in the MPS phase and in the database of the product structure, forming the *Bill of Materials* (BOM). A BOM represents a list of all materials, subassemblies, and other components needed to make all product nomenclature of the plant. It also contains the inventory data.

An MPS has an important function regardless of whether the MRP system is used or not. It accomplishes a coordinating function between manufacturing, marketing, finance, and sometimes engineering. Master scheduling is a decision-making process that can be considered as both a threat and an opportunity.

Then the plant control systems (for the component manufacturing) and the supplier monitoring systems (for the purchase of the components) are developed in the MRP-I phase for the *very short-term* execution activities of the plans. The time in this last stage takes usually a few days.

The MPS and MPR-I determine the weekly commitments of the delivery of each order to the customer, but not the day or a sequence, in which these orders will be processed in the plant facilities. The definition of the priorities of the items to be processed follows some optimization criteria, such as the cost, the time to change tools, or the importance of the customer. An MRP-1 diagram is presented in Fig. 1-3.

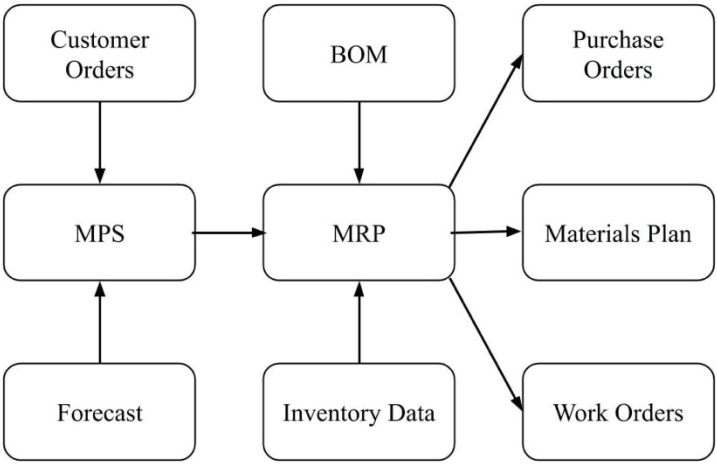


Fig. 1-3. Diagram of an MRP-I system.

The *Manufacturing Resource Planning* system, referred to as MRP-II, incorporates the original MRP-I system, defined by Orlicky in 1975, and follows the principles of *Hierarchical Production Planning* (HPP) defined by Hax and Meal in 1973 (Ptak and Smith 2016, 356). Other aspects of the MPR system are described in Section 2.4. Fig. 1-4 explains how tactical and operational planning problems are integrated into a classic MRP-II system. Other well-known production planning concepts and systems are adapted to this general scheme.

1.5 Lean manufacturing

Under the conditions of high competitiveness, a modern planning system tends to follow the lean manufacturing philosophy. *Lean manufacturing* is known as a production practice that considers the expenditure of resources for any goal different from the creation of a value for the end customer to be wasteful and thus, a target for elimination (Motwani 2003, 340–41; Ulutas 2011, 1194; Ptak and Smith 2016, 70). Practically, it is a set of tools that assist in the identification and steady elimination of waste (*muda*). Table 1-1 summarizes the typical seven prominent waste groups in the lean context. The main tools of a manufacturing program are a setup reduction system, such as the Single Minute Exchange of Die (SMED) approach, a visual control and fast intervention strategy, called the Total Productivity Maintenance (TPM) system, 5S, 5W+2H, Six Sigma.

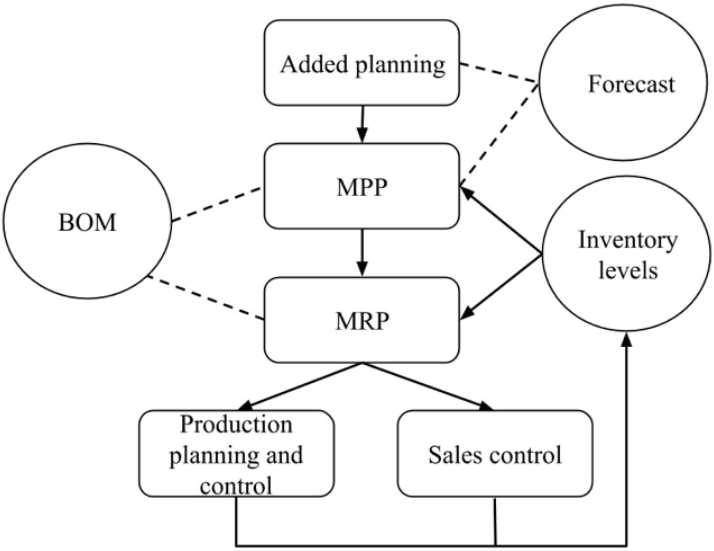


Fig. 1-4. Diagram of an MRP-II system.

In this context, 5S means five Japanize words, *Seiton* - Systematize, *Seiri* - Sort, *Seiso* - Sweep, *Seiketsu* – Standardize, and *Shitsuke* – Self-Discipline, the five concepts, which act as a base of *Kaizen* – a continuous improvement culture, including personal, familiar, and social contexts.

The 5W+2H methodology does not give a solution, but allows defining the problem and facilitates the focus on its causes. It includes five W-questions and two H-questions:

1. *What*? - A brief description of the problem that is being presented, a maximum of two lines.
2. *When*? - When does the problem appear? At what time of the day and/or the process in question?
3. *Where*? - Where can the problems be observed (Line/Machine/Place)? In which part/place of the product/process did the problem appear?
4. *Who*? - Who does it happen to? Is the problem related to people's abilities?
5. *Why*? - Why does the problem happen?

6. *How*? - How does the problem differ from the normal (optimal) state? Is the trend random in which the problem appears or does it follow a pattern?
7. *How Much*? - How many times does this problem occur in a day? In a week? In a month? How much money is involved?

The loan manufacturing tools focus on certain aspects and areas of the manufacturing process to eliminate the waste and improve the quality while the production time and cost are reduced. The waste reduction philosophy considers the changeover time as a non-value-added activity.

The reader can consult details of the concepts discussed above in the books by Anthony (1965); Vollmann et al. (2005); Jacobs et al. (2011); Ptak and Smith (2016), among others.

No.	Waste	Description
1	Overproduction	Producing items earlier or in greater quantities than needed by the customer generates other waste, such as overstaffing, storage, and transportation costs due to the inventory excess. Inventory is represented by a physical inventory or a queue of information.
2	Waiting	Workers merely serving as watch persons for an automated machine, or having to stand around waiting for the next processing step, tool, supply, part, etc., or just plain having no work because of no stock, lot processing delays, equipment downtime, and capacity bottlenecks.
3	Transportation	Moving the WIP from a place to another place in a process, even if it is only a short distance, having to move materials, parts, finished goods into or out of storage or between processes.
4	Over Processing	Taking unneeded steps to process the parts. Inefficiently processing due to a poor tool and product design, causing unnecessary motion and producing defects. Waste is when providing higher quality products than it is necessary.
5	Excess Inventory	Excessive raw material, WIP, and finished goods causing longer lead times, obsolescence, damaged goods, transportation/storage costs, and delay. Extra inventory such as production imbalances, late deliveries from suppliers, defects, equipment downtime, and long setup times.
6	Unnecessary Motion	Any motion made by employees during the course of their work other than adding value to the part.
7	Defects	Production or correction of defective parts. Repair, rework, scrap, replacement production, and inspection mean wasteful of handling, time and efforts.

Table 1-1. The seven waste groups in lean context. Adapted from Mistry and Desai (2015, 35–36).

CHAPTER TWO

INVENTORY MANAGEMENT IN PLANTS WITH SERIAL PRODUCTION

Inventory is evil.
(*Cuatrecasas-Arbós et al. 2015, 951*)

A stock or a store of different goods that are held for a purpose or use is defined as an *inventory*. The inventory management is of key importance in serial manufactures. An inventory may be located close to the place of immediate use, or it may be held in a distant warehouse or in a distribution center for future employment. An excessive inventory requires an additional floor shop space, freezes the capital, provokes blocking between the machines, and consecutively, increases the production cost. On the other hand, a lack of inventory causes starvation effects on the machines, resulting in a loss of productive capacities. The goal of industrials and researchers is to meet a convenient production system paradigm to maintain a balanced inventory.

A general framework of inventory management comprises the following four steps:

1. Formulate a mathematical model describing the behavior of the inventory system;
2. Seek for an optimal inventory policy with respect to the selected model;
3. Employ a computerized information processing system to maintain a record of the current inventory levels;
4. Using this record of current inventory levels, apply an optimal policy to signal when and how much to replenish the inventory.

There are three basic inventory systems known, namely the MRP, the Just-In-Time (JIT) discipline and the Optimized Production Technology (OPT). Usually, these systems are based on push or pull policies. In this section, some aspects related to the inventory management are discussed.

2.1 Inventory components

A plant with lot processing has always a quantity of different kinds of goods in a store to ensure a continuous operation due to different reasons. The main ones are:

- A certain quantity of a product stored is desirable and necessary when a company has the characteristics of a dynamic environment, where a customer demand can rise and fall quickly;
- A reserve of row material should be maintained against unexpected changes;
- Some supply of finished products is necessary in order to ensure or accelerate the fulfillment of a demand;
- The production process causes by itself intermediate parts accumulated in buffers between the machines.

There exist many reasons to maintain a sufficiently large inventory, like buying materials to take advantage of distributer discounts or ordering more in advance of an impending price increase, etc. On the other side, an excessive inventory affects the production efficiency, increases the production costs, and occupied areas, among other production indexes.

Two subsystems of the inventory can be distinguished:

- Directly involved in the production process;
- Support of different functions of the company.

An inventory subsystem in the production process includes traditionally three components:

1. Raw material;
2. WIP;
3. Finished goods.

Commonly, the raw material is a subject of labor, the WIP includes the unfinished parts throughout the manufacturing system, while manufactured products, which are ready for sale and delivery to a costumer, are named *finished goods*. In the literature, WIP is sometimes referred to as work-in-progress (see Section 1.1).

According to the function, the inventory within a production system can also be categorized as follows:

- *Maintenance, Repair and Operation supplies* (MRO) are the items, which are used to support the production process and its infrastructure, for example, oils, lubricants, coolants, janitorial supplies, uniforms, gloves, packing material, tools, among many others, up to office supplies, such as pens and pencils;
- *Transit inventory* results from the need to transport the items or material from one location to another one (pipeline, trucks);
- *Buffer inventory* is a safety stock, which is sometimes used to protect against the uncertainties of supplies and demands, as well as unpredictable events such as poor delivery reliability or low quality of a supplier's products;
- *Anticipation inventory* is used when a company purchases and holds an inventory, which is in excess to their current need, in anticipation of a possible future event;
- *Decoupling inventory* serves to absorb a shock of the system against production irregularities, for example, breakdowns or maintenance of the machines;
- *Cycle inventory* or *lot size inventory* reflects the relationship between the ordered products and the setup costs. When large quantities are ordered/produced, the inventory holding costs are increased, but the ordering/setup costs are decreased. Conversely, when the lot sizes decrease, the inventory holding/carrying costs decrease, but the ordering/setup cost increases since more orders/setups are required to meet a demand. When the two costs are equal (holding/carrying costs and ordering/setup costs), the total cost is minimized.

All these components are mutually dependent (Fig. 2-1).

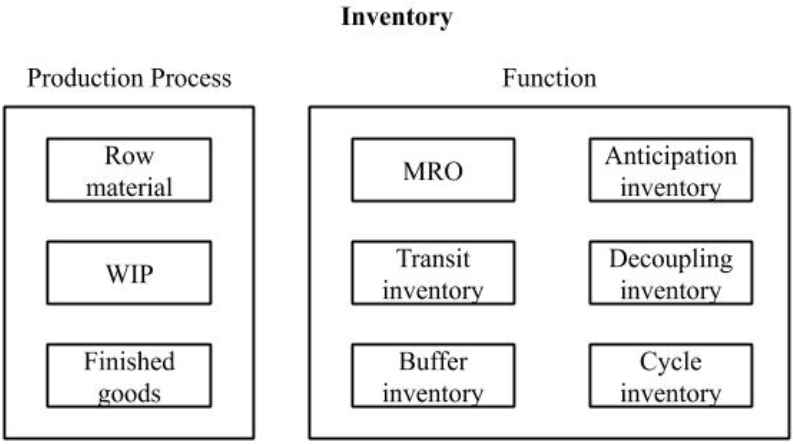


Fig. 2-1. Components of an inventory system.

2.2 Calculation of inventory average

In some manufacturing environments, such as assembly lines, an inventory is continuously generated because of the insufficient efficiency of planning and scheduling. It can also be generated due to the stochastic nature of the production process. Thus, the presence of WIP is rather inevitable and in a certain sense, it is even desirable. Too much WIP overloads the storage, holds the plant floor space, freezes the material resources, prolongs the product time from the start of the elaboration until it is ready to leave the facility, increases the production costs, decreases the production efficiency, etc. On the other side, too low WIP constrains the flexibility of the resources when the outflow is blocked due to the absence of working subjects. Conway et al. (1988, 229) observed that in serial production lines, the role of the WIP is to give some operational independence for each stage of a production system and thus, the WIP should be presented in every factory that has a production line. In addition, the WIP between two serial workstations increases the capacity by reducing the frequency and severity of blocking and machine starvation. However, to be competitive, the manufacturing facilities need to maintain a low ratio of WIP to throughput. *Inventory management*, or inventory control, is an attempt to balance the inventory storage and the production requirements with the goal to minimize the costs resulting from obtaining and holding the inventory. It is mainly concentrated on the optimization of the WIP amount. As a metric,

the WIP cost per time unit of a job may be used. It is usually taken as a constant throughout the manufacturing process.

The *inventory level* is the current level of a product that a company has in stock. *Inventory days*, also called cover days, stock cover, inventory days, or sales days, are calculated as the average number of days for which the goods remain in the inventory after being sold:

Inventory days = Average inventory × 365/ *Sales revenue*.

As a measure of short-term sales potential, a number above the industry norm indicates problems with sale forecast. On the contrary, a number below the norm indicates a loss of sales due to the company's inability to fulfill the demand.

An average inventory can be calculated using *Little's Law*. This law was formulated in 1961. It provides a fundamental relationship between three key parameters in a queuing (or waiting line/service) system:

1. Average number of items in the system;
2. Average waiting time (or flow time) for an item in the system;
3. Average arrival rate of an item to the system.

Little's Law says that, under steady state conditions, the average number of items in a queuing system is equal to the average rate, at which the items arrive, multiplied by the average time that an item spends in the system (Little and Graves 2008, 82):

$$L = \lambda W,$$

where

L - average number of items in the queuing system (WIP);
W - average waiting time of an item in the system;
λ - average number of items arriving per time unit.

An important feature of Little's Law is that by knowing, perhaps via a direct measurement, two of the three parameters, the third one can be calculated. This is an extremely useful property since the measurement of all three parameters may be difficult in certain applications.

Little's Law is applicable to many environments including manufacturing and service industries as well as everyday decision-making by individuals. Thus, if the average number of wafers arriving per day in a semiconductor factory is $\lambda = 1000$, the average number of wafers in the queuing system is

$L = 45000$ (WIP), the average waiting time in the system for one wafer is as follows:

$$W = L/\lambda = 45000/1000 = 45 \text{ days}.$$

Anupindi, Chopra, and Deshmukh (2012, 55–56) proposed an average inventory for a given throughput as a theoretical metric, assuming that no WIP item had to wait in a buffer:

$$\textit{Theoretical Inventory} = \textit{Throughput} \times \textit{Theoretical Flow Time}.$$

In the above equation, the *Theoretical Flow Time* equals the sum of all activity times (without waiting time) required to process one unit. Therefore, the WIP will be equal to the *Theoretical Inventory* whenever the actual process flow time is equal to the *Theoretical Flow Time*. This would obviously be an ideal situation, where the inflow, processing, and outflow rates are all equal at any point of time, i.e., it represents the minimal inventory needed for the goods to flow through the system without waiting. Here, the difference between the flow time and the lead time should be emphasized. The *flow time* is the time between the job release and its completion. The flow time is typically random. The *lead time* is a constant used for planning purposes. The *service level* is defined as the fraction of jobs whose flow time is no greater than their lead time.

2.3 Push/pull policies

The inventory control, implemented in the industry, can be of pull or push type. It can also be a hybrid system, which combines both effects (Spearman, Woodruff, and Hopp 1990; Hopp and Roof 1998; Olaitan, Yu, and Alfnes 2017). A *push system* means 'make all we can just in case'. It is associated with the production approximation, anticipated use, large lots, high inventories, waste, and management by fire extinguishing, poor communication. Push systems are popular in the form of the MRP approach and were successful in many factories, reducing inventory levels and improving customer service. On the contrary, a *pull system* means 'make what's needed when we need it'. It does not schedule the start of jobs but instead, authorizes production. This system is categorized by the production precision, actual consumption, small lots, low inventories, waste reduction, management by sight, and better communication.

In a *push* system, the inventory must be determined in advance and goods produced must be forecasted. While lean methodology is a *pull*

system, in which nothing is produced or purchased without evidence of actual demand.

The basic difference that exists between push and pull systems is the mode of material flow. In a *push* system, a material is always pushed as *far as possible* and *as soon as possible*. In contrast, in a *pull* system, material flows on an *as-needed* and not on an *as-available* basis. This distinction is critical for an understanding of the modeling requirements of production systems.

Hopp and Roof (1998, 897) and many other authors consider that pull systems have some advantages over push systems. They noted that a push system *schedules* the releases, while a pull system *authorizes* them. As a result, a push system controls the release rate (and hence the throughput) and observes the WIP, while a pull system controls the WIP and observes the throughput. The authors indicated the following advantages of pull over push:

1. *Observability*. The WIP is directly observable, while the capacity (with respect to which the release rate must be set) is not;
2. *Efficiency*. A pull system can achieve the same throughput rate as a push system with a smaller average of the WIP level;
3. *Variability*. Flow times are less variable in a pull system than in a push system, because a pull system regulates the fluctuation of the WIP level, while a push system does not;
4. *Robustness*. Pull systems are less sensitive to errors in the WIP level than push systems.

However, Bonney et al. (1999) showed that both systems have similar results in the performance. Another position tends to a wide consensus that pull controls are better in minimizing the system inventory, while push controls are better for maximizing the throughput (Gaury, Kleijnen, and Pierreval 2001). Andersson et al. (2010, 1520) highlighted that in a push environment, the work is scheduled on the base of demand and therefore, the inventory levels have direct consequences of the order releases and quantities, while they are ideally fixed and pre-calculated in a pull environment, where the work is authorized through the status of the system.

The main disadvantage of a push system is its vulnerability when sales become variable. In this scenario, the forecasts become inaccurate and cause either a shortage of inventory or an excess of inventory that requires storage.

2.4 MRP system

2.4.1 Features

Material requirements planning (MRP) is a *computer-based* production planning and inventory control system for the production and scheduling in the plant, i.e., it includes two approaches, a concept and a software. According to the APICS Dictionary[2], MRP is defined as

> 'a set of techniques that uses bill of material data, inventory data, and the master production schedule to calculate requirements for materials. It makes recommendations to release replenishment orders for material. Further, because it is time-phased, it makes recommendations to reschedule open orders when due dates and need dates are not in phase. Time-phased MRP begins with the items listed on the MPS and determines (1) the quantity of all components and materials required to fabricate those items and (2) the date that the components and material are required. Time-phased MRP is accomplished by exploding the bill of material, adjusting for inventory quantities on hand or on order, and offsetting the net requirements by the appropriate lead times.'

The MRP processes the information from the MPS, BOM, and inventory requirements, while the output information includes the work and purchase orders as well as the required material quantities together with their release times in the manufacturing process.

With MRP, a triune problem is solved:

1. Nomenclature of raw materials and components;
2. Quantity of each component;
3. Time when everyone is needed.

The MRP synchronizes the flow of materials, components, and parts. It converts the MPS into work orders with the goal to keep adequate inventory levels, which assure that the required materials will be available when needed. The MRP is a universal system, which is applicable mostly in manufacturing and fabrication industries, because it is applied to the analysis of all company activities in terms of the customer demands and the management of resources via its own logic and data processing. On the other hand, the MRP is not useful for job shops or for continuous processes that are tightly linked.

[2] https://www.academia.edu/31291212/APICS_Dictionary_13th_ed

The MRP system is considered a *push* type system, where the type and amount of all purchased products must be determined *in advance* for the plant, and goods produced must be *forecasted* and then pushed to the consumers. This contrasts with the *lean methodology*, which is characteristic for a *pull* system, and in which nothing is made or purchased without evidence of an actual demand.

In this manner, *the primary objective* of the MRP is to ensure the availability of materials and components when they are needed in the production process to keep the continuity in the manufacturing. While MRP is designed to ensure an adequate inventory at the required times, a company can hold a bigger inventory than it is necessary, thereby driving up the inventory costs. Therefore, *another goal* is the development of an effective inventory management and the scheduling optimization.

2.4.2 Developers and extensions of the MRP

The MRP system is really a method of production management. The IBM engineer Joseph Orlicky proposed this system in 1964 after studying the Toyota Production System, which was a model of the *lean production* methodology. Orlicky's ideas became popular after the publication of his book *Material Requirements Planning: The New Way of Life in Production and Inventory Management* (Orlicky 1975) and changed the world of manufacturing forever. The reader can meet plenty of interesting information about this theory and a comprehensive discussion in the paper by Wilson (2016).

Oliver Wight, a management expert and co-worker of Orlicky, developed an extension of MRP called MRP2 (or MPR-II) in 1983. This system enriched the planning process by including other resources of the company, such as financials and added processes for the product design, capacity planning, cost management, shop floor control, sales, and operations, among many others.

In 1990, the analyst firm Gartner introduced the term *Enterprise Resource Planning* (ERP) to denote a still more expanded and generalized version of MRP2, which includes other major functions of a business, such as accounting, human resources, and supply chain management, all in a centralized database.

The Gartner Glossary[3] defined ERP as

[3] https://www.gartner.com/en/information-technology/glossary/enterprise-resource-planning-erp

'the ability to deliver an integrated suite of business applications. ERP tools share a common process and data model, covering broad and deep operational end-to-end processes, such as those found in finance, HR, distribution, manufacturing, service and the supply chain'.

Both MRP and MRP2 are considered direct predecessors of ERP, which consists of three primary successive steps:

1. Take the inventory of the materials and components at hand;
2. Identify which additional ones are needed;
3. Schedule the production or purchase.

ERP could be quickly expanded to other industries, including services, banking and retail, which did not need an MRP component. However, MRP is still an important part of the ERP software used by manufacturers.

After the death of Orlicky in 1986, the second edition of his MPR was written and adjusted by the practitioner George Possl, a coworker and follower of Orlicky. This advanced version of MRP, called DDMRP, was released in 1994. Orlicky's system was updated with a *demand-driven* planning process that used the actual sale orders, rather than the typical MRP method of a sales forecast, to calculate the material requirements. This variant of MRP is a *pull* approach, which is sometimes considered as a controversial one and a violation of important principles established by Orlicky.

In 2011, Carol Ptak and Chad Smith revised again Orlicky's seminal work. Their book was released as the third edition of MPR. Nevertheless, it had essential deviations from the author's version. Later, Ptak and Smith proposed the following editions in 2016 and 2019 with a new focus on DDMRP. This version helps in the achievement of a better matching of supply and demand, which in turn reduces the product costs and increases revenues as customer demand is fully met and no revenue opportunities are lost from missed shipping dates or inventory shortfalls. Version 2 of DDMRP was based upon the connection between the creation, protection and acceleration of the flow of relevant materials and information to drive returns on the asset performance (Ptak and Smith 2016, Ch 1).

2.5 JIT paradigm

2.5.1 Features

The JIT production model emerged from the pull system, which is known as the production of the necessary items in the necessary quantities at the

necessary time. Taiichi Ohno developed this approach between 1948 and 1975. It was called Japanese *Toyota Production System* (TPS). TPS was based upon very low levels of waste. In general, the stored quantity of goods exceeds the immediate necessities of a company, for example, within the next few hours or days. JIT adopted a 'zero concept', which means the achievement of the goals of zero defects, zero queues, zero inventories, zero breakdown, and so on. It ensures the supply of the right parts in the right quantity, at the right place and at the right time. As a result, any product had the desired quality and quantity, and it was manufactured in the requested period. Since then, many worldwide companies have adopted *lean manufacturing* in order to increase the productivity, to reduce the lead time, the inventory costs, as well as to improve the product quality (Motwani 2003, 349). The old system of material acquisition, the relationships with the supply chain and the consumer companies were changed to new revolutionary concepts. At the plant level, the conventional method of a push production system linked with MRP was changed to a pull-type JIT production system to meet the global competition, where the WIP is managed and controlled more accurately than in a push production system (Sendil Kumar and Panneerselvam 2007, 393).

The control structure can be described through the relationship between two hypothetical nodes (buffers), which are defined as follows (see Fig. 2-2):

- *A puller* is the node to which the material is pulled. It can be considered as a *topological downstream* of the source node;
- *A source* is the node from which the material is pulled. It can be considered as a *topological upstream* of the puller node.

A control pulse activates the pulling of material from its source node to a puller node.

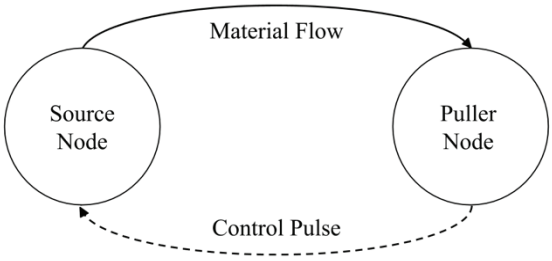


Fig. 2-2. JIT node relationships. Adapted from Mejabi and Wasserman (1992, 143).

The control structure is made up of the two following basic functions:

- *Generation of a pull pulse*. This occurs when the inventory level at a buffer drops below some predetermined level;
- *Physical pull of material from a source to a puller*. This implies a termination of the residence of the material at the source buffer and the establishment of residence at the puller buffer.

According to Finch and Cox (1986, 331), JIT is composed of eight linked areas requiring a management and continuous improvement:

- A focused factory;
- Reduced setup times;
- Group technology (GT);
- Total preventive maintenance;
- Cross-trained employees;
- Uniform workload;
- JIT delivery of purchased parts and materials;
- Kanban method of production control.

2.5.2 Kanban method

Taiichi Ohno also proposed the *kanban*[4] concept. It plays a significant role in a JIT production system and represents a signaling system to manage and synchronize the processes at the upstream and downstream of a multi-stage production scheduling. It is also used as an inventory control system. Physically, *kanban* is usually a simple plastic card containing complete information required for the production and the assembly of a product at each stage and the details of its completion path. It is also a method of labeling the production lots to obtain a better control over raw materials, WIP, and finished goods. Each step of the production process must deliver its output to the following step neither to delay the start of production at this step nor to create excessive WIP. A work cannot be started at a workstation unless a kanban indicates that this work is required by the next downstream workstation.

There exist different variants of a kanban. An example of a low-tech kanban is shown in Fig. 2-3. A visual kanban can be seen in Fig. 2-4.

[4] The word '*kanban*' is of Japanese origin; it means a card, a board or a post-it.

Fig. 2-3. A simple kanban[5].

Fig. 2-4. A visual kanban.

Two types of kanban cards are used:

- A withdrawal kanban is attached to waiting lots of WIP. It authorizes moves between workstations;
- A production kanban is attached to lots being processed at a workstation and serves as a work order.

A production may only be initiated when the upstream process receives the card. It makes sure that every process is producing the right part, in the right quantity, and at the right time.

2.5.3 Operating the kanban method

A plant having two workstations is used to illustrate how the kanban method operates (Fig. 2-5). The first workstation (Storage area A) is a transfer line consisting of several processing operations. At the second workstation (Storage area B), a number of assembly operations are carried out. A production order is divided into small lots of equal size, which are then processed individually, and a kanban is attached to every order. A lot flows through the shop as follows:

1. A kanban lot of unprocessed parts is withdrawn from the store, processed, and then stored temporarily as WIP;

[5] https://www.pdcahome.com/metodo-kanban/

2. When called for, it is transferred to the assembly workstation, where it is again stored as WIP;
3. It is subsequently withdrawn; the assembly operations are carried out;
4. The lot of finished goods is then placed into the storage of finished goods.

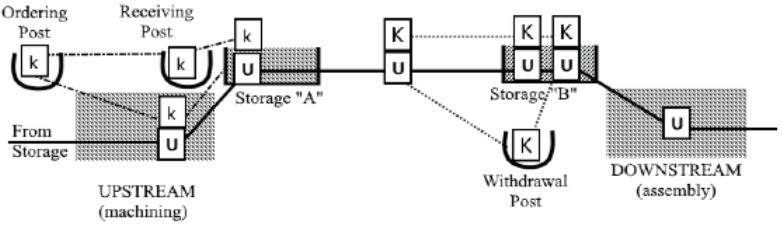


Fig. 2-5. A two-card kanban system in a plant with two workstations. The dotted lines show the flow of the withdrawal kanban. The dashed lines show the movements of kanban cards. **K** is a withdrawal kanban; k is a production kanban; **U** is a production unit. Adapted from Price, Gravel, and Nsakanda (1994, 2).

The kanban cards control the workflow in the following manner. In order to withdraw a waiting lot from a storage point at the head of the workstation, the operator must take a free production kanban from a kanban post at the exit of the workstation. If there is no free kanban, he must wait. If there is a free kanban, he withdraws a waiting lot from the storage point, detaches the withdrawal kanban, attaches the free production kanban, and sends the recently freed withdrawal kanban back to the storage point at the exit of the previous workstation, where it authorizes the forward movement of another lot.

When the processing of the lot is finished, the operator places the kanban into the WIP storage at the exit of the workstation. A production kanban remains attached to this lot until it is called downstream to the next workstation by a free withdrawal kanban. At this time, the production kanban is freed and can be used to restart the processing cycle at the workstation.

The operator at the assembly workstation takes the lot from the storage area B, removes the withdrawal-attached kanban, and places it into the withdrawal kanban post. This unattached card is returned to storage area A, authorizing the transfer of another lot from A to B. Each lot waiting in storage area A is accompanied by a production kanban, which will be replaced by an unattached withdrawal kanban before the lot moves to

storage area B. When detached, the cards go to a kanban receiving post and are then send the production ordering kanban post at the head of the machining workstation to authorize the processing of another lot.

The kanban cards authorize the production at the machining workstation only when it is required to supply the assembly workstation, thus limiting the stored inventory. Since the number of the kanbans is limited, the processing workstation stops the production when no cards are present and resumes the production when a card is sent back from the assembly workstation. In this manner, the kanban cards control the material flow to minimize the involved inventory. The number of kanban cards in circulation fixes the absolute maximum of processing lots.

A kanban is not an immanent element required for a JIT or pull system to operate. It is observed that the use of kanbans is, in essence, just a tool for the physical realization of the control.

2.5.4 Number of kanbans

An appropriate number of kanbans at a workstation, which are needed to circulate between downstream and upstream, is as follows [6]:

$$N = T/C = [D \times LT\,(1+\alpha)]/\,C,$$

where:

- N - number of kanbans;
- T - total required inventory;
- D - hourly/daily/weekly demand;
- LT - lead time (processing time + waiting time for the kanban, in hours/days/weeks);
- α - safety factor;
- C - container capacity (items held per container).

For example, let a downstream process use 200 items/hour (on average). The lead time is 12 hours. The container capacity is 144 items. The variation in the lead time or demand, called the safety factor, is 15%. Therefore, the needed number of kanban cards is:

$$N = 20 \times 12 \times (1 + 0.15)/144 = 20.$$

[6] LeanLab. 2017. “Lean Manufacturing.” How to Calculate the Number of Kanban. 2017. http://www.leanlab.name/how-to-calculate-the-number-of-kanban

Later, Spearman, Woodruff, and Hopp (1990) described a pull alternative to kanban, named CONWIP (CONstant Work-In-Process), where the WIP is not constrained for every operation or machine but the quantity of WIP in the total production flow is constrained. CONWIP regulates the release of the items at a global level into the system but does not specify their processing sequence. This concept contributes to operating a global pull control with a push control at the local level (Hopp and Roof 1998, 687). It is also used to model low-level logistic problems (Ni and Werner 2017). As a result, such mechanisms are able to combine the low inventory benefits of the pull control with the high throughput benefits of the push control.

A critical review of the JIT literature and an analysis of the revealed trends in JIT/kanban systems has been given by (Sendil Kumar and Panneerselvam 2007).

2.5.5 Just-In-Sequence discipline

In recent years, the JIT paradigm has been advanced into a discipline called *Just-In-Sequence* (JIS). JIS is mainly an inventory strategy that is originated from the production logistics. It represents an advanced synchronization between delivering the manufactured components and the downstream assembly. With JIS, the components and the parts arrive at a production line right at the time moment when they are scheduled to be assembled. The sequencing allows the companies to eliminate supply buffers as soon as the necessary quantity in the component part buffers is reduced to a minimum. It is an extreme technology, which permits a company to work practically without any inventory. Currently, JIT/JIS is mainly implemented in industries with high automation level, such as automotive, semiconductor, and digital devices industries, see SIEMENS products[7].

The JIS supply is widely used in multi-OEM (*Original Equipment Manufacturer*) environments, where a company produces parts and equipment to be used as merchandise for other manufacturers. Such a merchandise is referred to as value-added reseller (VAR). A VAR adds a value to the original item by augmenting or incorporating features or services. The VAR works closely with the OEM, which often customizes designs based on the needs and specifications of the VAR company.

One of the basic examples of the OEM functioning is the relationship between an auto manufacturer and a producer of auto-parts. Parts such as exhaust systems or brake cylinders are manufactured by a wide variety of

[7] Opcenter Execution Discrete
https://www.plm.automation.siemens.com/global/en/products/manufacturing-operations-center/simatic-it-unified-architecture-discrete-manufacturing.html

OEMs. The OEM parts are then sold to an auto-manufacturer, which assembles the item into a car. The completed car is then marketed to auto dealers to be sold to individual consumers. An OEM may also refer to a company that buys products and then incorporates or rebrands them into a new product under its own name. It is typically used in the computer industry. For example, Microsoft supplies its Windows software to Dell Technologies, which incorporates it into their personal computers and sells a complete PC system directly to the public. In the traditional sense of the term, Microsoft is the OEM and Dell the VAR. However, for consumers, Dell is usually referred to as the OEM.

2.6 OPT system

2.6.1 Features

The OPT system is a production improvement concept, which was developed in the 1980s by Eliyahu M. Goldratt as a software package. Later, this concept became an entire production control philosophy based on *bottleneck management* and *finite-capacity scheduling* (Goldratt 2012). It is now better known as *Theory of Constraints* (TOC). The *objective of an OPT* is to schedule the production so that the production output is maximized. The *benefit* claimed for an OPT is dedicated to schedule finite resources in order to achieve a maximum factory effectiveness, focusing in the following points:

- Address the key problem of bottlenecks;
- Improve the profitability by simultaneously increasing the throughput;
- Reduce the inventory and operating expenses.

The constraints can be *internal* or *external* to the system. The *types of internal constraints* are:

- *Equipment* – the limits of the system ability to produce more salable goods/services;
- *People* - lack of skilled people limits the system;
- *Policy* - a written or unwritten policy prevents the system from making more.

If the capacity throughput of a constraint is elevated to the point, where it is no longer a limiting factor of the system, this means 'break' the

constraint. In this case, the limiting factor is some other part of the system or may be an external constraint.

The method is based on the concept that there are two fundamental manufacturing phenomena:

- *Dependent events* - all processes rely on the completion of the preceding operations;
- *Statistical fluctuations* – the processing times fluctuate around the average.

These phenomena have an influence on the capacity of a plant, which becomes unbalanced and therefore, the bottlenecks are inevitable. The key distinctive feature of the OPT is its ability to identify and isolate the bottleneck operations. Then the OPT focuses on these bottlenecks to determine the production plans and the schedules for the entire shop. This simple idea leads to a better utilization of the manufacturing resources, resulting in raising the throughput while reducing the inventory and the operating costs. Consequently, a smooth and continuous flow of the WIP is achieved.

The following nine features characterize an OPT system:

- Balance flow but not capacity;
- The utilization level of any system part, which is not a bottleneck, depends on other constraints of the system, not the worker potential;
- The utilization and the activation of a resource are not synonymous;
- An hour lost at a bottleneck is an hour lost for the total system;
- An hour saved at a non-bottleneck is just a mirage;
- Bottlenecks govern both throughput and inventories;
- The transfer batch may not be equal to the process batch;
- The process batch should be variable, not fixed;
- Schedules should be established by looking at all constraints simultaneously.

Lead times are the results of the schedule and cannot be predetermined. It is found that it would be the best for small/medium size assembly companies to embark on using OPT. It can improve the assembly throughput, inventories, and operating costs before deciding to implement lean manufacturing or lean Six Sigma.

2.6.2 Plant types according to the TOC

There are four primary types of plants defined in the TOC. These four plant types can be visualized drawing the flow of material from the bottom of a page to the top. Specifying the general flow of the materials through a system, some hints, where one must look for typical problems, may be provided. This type of procedure is known as *VATI analysis*[8] as it uses the bottom-up shapes of the letters *V*, *A*, *T*, and *I* for determining the general flow of parts and products from raw materials to finished products (logical product structure).

V-plant: The general flow of material is *one-to-many*, such as a plant that uses one or a few raw materials and makes a number of final products, which flow through divergent points in their routings. Classic examples are meat rendering plants or steel manufacturing. The primary problem in *V*-plants is 'robbing', where one operation (A) immediately after a diverging point 'steals' material meant for another operation (B). Once the material has been processed by A, it cannot come back and be run through B without significant rework.

A-plant: Converging points dominate this logical structure. The general flow of material is *many-to-one*, such as many sub-assemblies converge to a final assembly. The primary problem in an *A*-plant consists in synchronizing the converging lines so that each line supplies the final assembly point at the right time.

T-plant: The general flow represents one line (*I*-plant) or multiple lines, which then split into many assemblies, subassemblies, and parts (*many-to-many*). Most manufactured parts are used in multiple assemblies, and nearly all assemblies use multiple parts. Customized devices, such as computers, are good examples of such an environment. *T*-plants suffer from both synchronization problems of *A*-plants (parts are not all available for an assembly). The robbing problems of *V*-plants (one assembly steals parts that could have been used in another one).

I-plant: This logical structure is the simplest one of production flows. The primary work is done in a straight sequence of events (*one-to-one*). Material flows in a sequence, such as in an assembly line. Once the general flow is determined, the system control points (gating operations, convergent points, divergent points, constraints, and shipping points) can be identified and managed. The constraint is the slowest operation.

These four flow types may be combined in many ways in large facilities, for example, 'an *A*-plant feeding a *V*-plant'.

[8] https://www.academia.edu/31291212/APICS_Dictionary_13th_ed

2.6.3 Drum-buffer-rope method

The OPT method of scheduling recommends to launch the material onto the shop floor only at the rate at which it is consumed by the bottleneck. Furthermore, a time buffer of work protects the production in the bottleneck operations. For example, a work is scheduled for the third day. It arrives on day one and creates a buffer of two days. This buffer is used as a protection against a disruption in the operations. This implies that within the manufacturing operations and operations management, *the solution seeks to **pull** materials through the system, rather than **push** them into the system*. The primary methodology used is the *Drum-Buffer-Rope* (DBR) method (see Fig. 2-6). Mainly, the DBR is a method of synchronizing the production to the constraint while minimizing the inventory.

The method is based on a fundamental assumption that within any plant, there is one or a limited number of scarce resources, which control the overall output of that plant. This is the '*Drum*', which sets the pace for all other resources. The Drum is a constraint. The speed, at which the constraint runs, sets the 'beat' for the process, and determines the total throughput. The bottlenecks, which beat out the pace like a Drum for the whole factory, should be kept fully scheduled and working at all times. The bottlenecks must be protected against any interruption caused by the breakdowns, quality restrictions, setup times, labor concerns or any other variation. This protection is achieved by the creation of time buffers. All other operations become then synchronized with the bottleneck operation, and some work is pulled through as if it was on a rope.

The '*Buffer*' is the level of the inventory needed to maintain a consistent production. It represents the amount of time, in which WIP must arrive in advance of being used to ensure a steady operation of the protected resource.

Typically, there are two buffer types:

1. *Constraint Buffer* - a buffer immediately before the constraint that protects this constraint;
2. *Customer Buffer* - a buffer at the very end of the process that protects the shipping schedule.

The '*Rope*' is a signal generated by the constraint indicating that some amount of the inventory has been consumed.

In order to maximize the output of the system, the planning and execution activities are focused on exploiting and protecting the Drum against disruption using 'time buffers'. Another aspect is synchronizing or subordinating all other resources and decisions to the function of the Drum through a mechanism that is akin to a Rope.

An example of the use of this method in the manufacturing environment is given below.

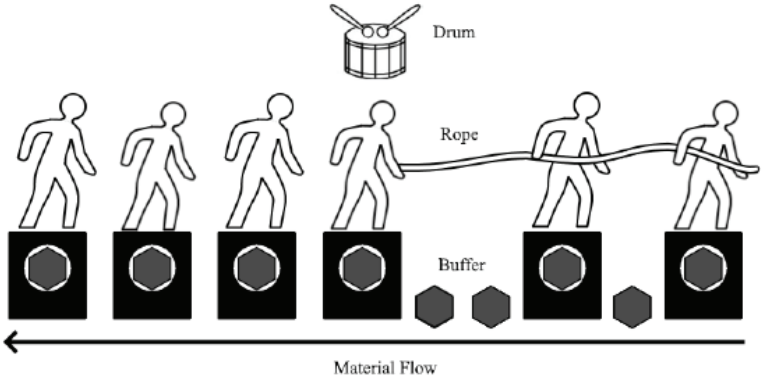


Fig. 2-6. The Drum-Buffer-Rope concept.

2.6.4 Scheduling of a production line with the DBR method

An example of applying the DBR method in production planning to a manufacturing system that represents a line of machines in series (such as a flow shop), is presented in Fig. 2-7. The system consists of four successive operations, and there are no assemblies or divergences in the process. For simplicity, it is assumed that there is only one raw material and one customer.

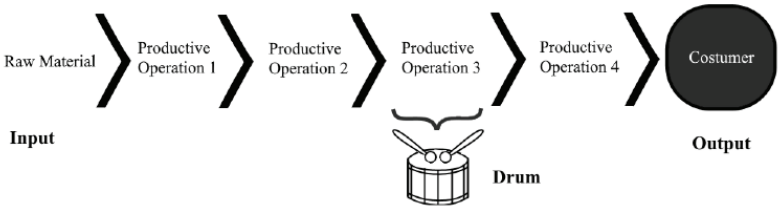


Fig. 2-7. Representation of a production system as a series of machines.

The method contains the steps described below.

Step 1. Identify the system constraint

The first step is to identify the Drum. The Drum is usually a resource (or a work center) with the largest workload in the plant (Table 2-1). It is fundamental to mention that in the DBR method, the most important step is not the input or output but the production process itself.

The slowest of the resources is identified as the constraint. When the production system is studied (Table 2-1, Fig. 2-8), it is determined, which resource can stop the production. In this case, it is the operation 3 (the red resource). The other three resources, which are the productive operations 1, 2 and 4 (the yellow, green, and blue operations, respectively), have an excess in the capacity.

Step 2. Exploit the system constraint

It is assumed that resource 3, which represents the constraint, has two pieces of material in process ready to be treated on day 1. Resource 3 takes 6 hours to process the first piece and the next 6 hours to process the second piece. This sets the pace of the constraint, which is 1 piece every 6 hours.

The resources prior to the constraint have an excess capacity and should supply the constraint with the material before the moment when it runs out of parts without any problem. If the raw material supplier stops delivering for some reason, resource 3 is able to maintain the manufacturing for 6 hours after finishing the first piece because it has material. However, after the next 6 hours, it will stop due to lack of material. The previous resources (1 and 2) would not have enough material to supply although they have sufficient capacity (Fig. 2-9).

Production operations	**Day**	**Part**	**Time** (*hs*)
1	1	1	2
2	1	1	1
3	**1**	**1**	**6**
4	1	1	1
1	1	1	2
2	1	1	1

Table 2-1. Production schedule per day.

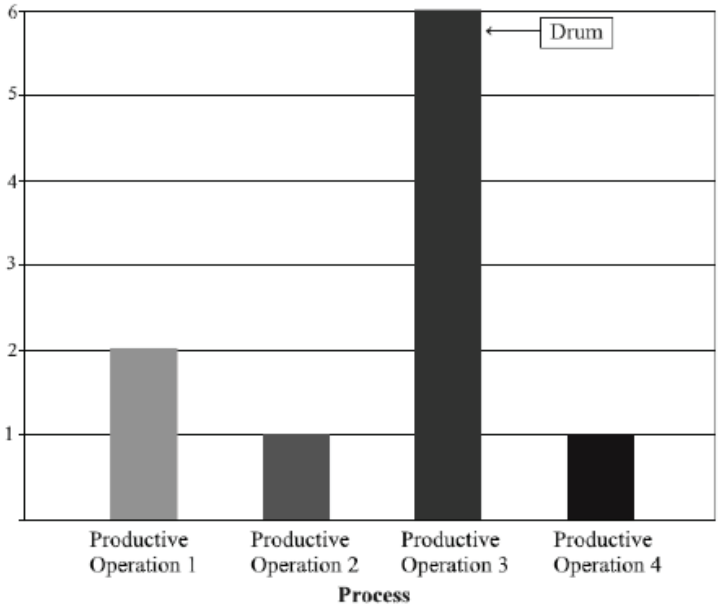


Fig. 2-8. Drum chart.

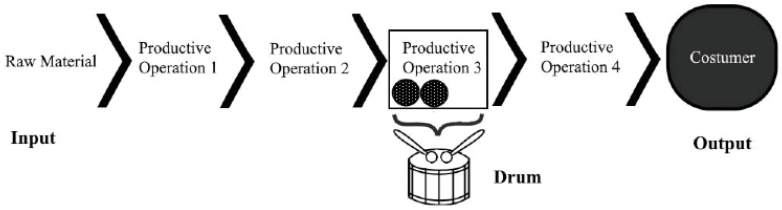


Fig. 2-9. Production system with two products in line.

Step 3. Establish the Buffer

To reduce the negative impact on the constraint, a solution is to put a buffer, or a time buffer, in front of the constraint (Fig. 2-10).

As it is already established, the rhythm is one piece every 6 hours. Then the question is: What time must the constraint be protected? Let us suppose that we came up with placing four pieces at the start, which means in this case, 24 hours of protection with a buffer. Thus, regardless of the previous processes, the constraint can work 24 hours without an interruption.

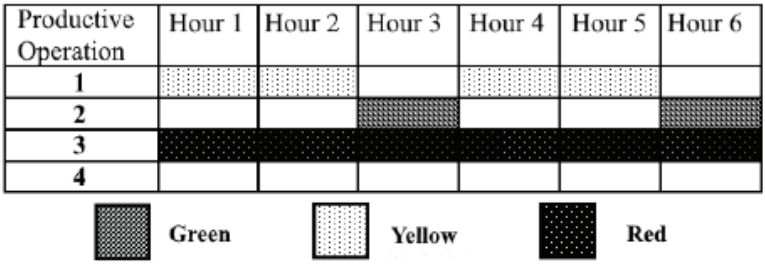


Fig. 2-10. Establishing the buffer.

Step 4. Synchronizer the Drum – Subordination

The buffer and the processing time of the pieces at the resources prior to the bottleneck (identified as the constraint) set the size of the Rope that is thrown to the first point of the production line. After programming the Drum, the material release and shipment are connected to it, using the buffer offset. The material is released with the same speed as the Drum consumes that material. The orders are shipped with the production rate, which is identified by the Drum, and every 6 hours one piece will be completed within the production line.

Step 5. The DBR programming algorithm

In this step, it is assumed that the work to be carried out through the bottleneck (system constraint) must be programmed and protected by the creation of a time buffer, also called the *sending buffer*. The determination of the buffer size is a complex procedure, since there is no formula for its calculation. Its value depends on numerous variables, such as the processing time or the failures in the process tools.

Following the example, on day 1, resource 3 is working first 6 hours until the raw material supplier fails. Nevertheless, at the level of the constraint, there is no information about the fault, because it has the material and continues manufacturing. On day 2, the following situation may occur. The supplier of the raw material still cannot supply but the system works until the fourth piece is consumed.

If then the supplier failed to react, a system of traffic lights of the buffer, in green, yellow, and red colors is used, helping to keep the process safe (Fig. 2-11). This system ensures that the working process is programmed well, i.e., conflicts are solved, and potentially late jobs (red lot) are identified. In this way, the behavior of the buffer will be as follows.

If a green lot is always presented in the buffer, this means that the buffer is too long. In the normal case, the buffer moves between a green lot and a yellow lot. When the material receives the yellow color, the cause must be sought and the planner must act quickly. When the red lot enters into the buffer, it is a warning sign, and an action has to be taken to bring quickly material before the constraint stops.

Once the buffer is placed with sufficient material, it is ensured that the line can continue to operate under normal conditions. When the constraint takes a piece, this is a warning that there is a lack of material and more lots must be supplied (Fig. 2-11).

Thus, according to this method, it is more convenient to have material in excess at a critical point to ensure that the constraint has always material to work instead of having inventory in all processes. The buffer monitoring displays whether the size of the lot should be increased or reduced.

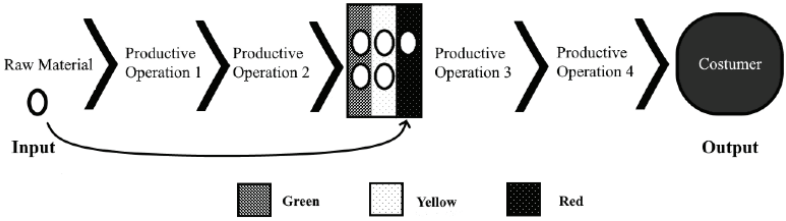


Fig. 2-11. Production system with DBR scheduling.

2.7 A comparison between JIT, MRP and OPT paradigms

A detailed comparison between JIT, MRP, and OPT inventory control systems is presented in the literature (Golhar and Stamm 1991, 659). The authors noted that some researchers argue that JIT, MRP and OPT are mutually exclusive. In a repetitive manufacturing environment with a moderate product variety, kanban (an inventory control method in JIT) is found to be effective because it drastically reduces the inventory, simplifies planning and control. On the other hand, in a job shop environment with large product variety, an MRP improves the customer service and moderately reduces the inventory. However, in a complex production environment, an OPT system is preferred to MRP or JIT. Meanwhile, another opinion is that both kanban and OPT are more productive systems than MRP. The authors suggest that the selection of an ideal inventory control system is specific for every situation.

According to the simulation and mathematical studies on the comparison of inventory systems, the manufacturing environment is most crucial in reducing the inventory. In particular, a levelled production, reduced setup times and lot sizes are most effective in reducing the inventory and improving the customer service. With kanban, the JIT philosophy focuses on simplifying the production process and finding ways to reduce setup times and lot sizes. When compared to MRP and OPT, kanban provides an appropriate manufacturing environment for an effective control of the inventory. Another critical factor in a successful implementation of an inventory system is human involvement. The kanban system was found to be advantageous because it is relatively simple and easy to implement. On the other hand, MRP has many human involvement problems. Employees are excluded from decision-making process. They fail to see how their work contributes to a reduction of the inventory. OPT requires managers to make changes in procedures and work methods prior to their implementation. These changes increase the employee involvement. As a result, OPT has fewer employee problems in control of the inventory than MRP. The papers, which are dedicated to this subject, focus on contrasting kanban, MRP, and OPT. However, some studies argue that kanban and MRP systems are complementary. According to the opinion of various researchers, the strength of MRP consists in long-term planning and scheduling, while kanban is better at daily operations providing a visible control of the production and reducing the inventory. Thus, the integration of MRP and kanban would allow a manufacturer to improve the productivity and customer service level.

2.8 Methods of inventory optimization

The inventory control and optimization attract the attention of researches due to the interest of the manufacturers in a reduction of the production costs, as well as the computational complexity of the problem. Nevertheless, one can note a lack of systematic studies. Following the idea that for lean manufacturing 'inventory is evil' (Cuatrecasas-Arbós et al. 2015, 951), the papers are focused on the problem of the WIP control and keeping it at a low level. The proposed methods may be roughly classified into the following categories:

1. A visual control of the location and quantity of the inventory combined with polices and heuristics (Maimon, Dar-El, and Carmon 1993, 173–80; Papadopoulos and Vidalis 2001, 188–95; Duwayri et al. 2006, 719–23; S. Kim and Uzsoy 2009, 1926–29; Yang and

Posner 2010, 7–12; Cuatrecasas-Arbós et al. 2015, 959–61; Olaitan, Yu, and Alfnes 2017, 279–80).

2. Simulation methods were used for the study of simple generic serial systems (Conway et al. 1988, 232–40), or designing more complex models such as a Virtual Production Line (VPL) (Qiu 2005, 168–70), a Production-tracking system (G. Zhang, Zhang, and Tian 2011, 1331–33), Discrete-event simulation (Xanthopoulos and Koulouriotis 2014, 1508–13), and Monte Carlo simulation (M. Li et al. 2016, 118–20).
3. Optimization methods such as linear and nonlinear programming (S. X. Bai and Gershwin 1990, 558–62; J. S. Kim and Leachman 1994, 155), artificial neural networks (Y.-H. Lin, Shie, and Tsai 2009, 3422–25), and fuzzy logic (Tsourveloudis 2010, 649–50).
4. Stochastic methods such as Markovian processes (Vidalis, Papadopoulos, and Heavey 2005, 826–28; Sivakumar and Arivarignan 2009, 48), system nervousness measurements (Kilic and Tarim 2011, 288–89), and Bayesian modeling (L. Chen and Plambeck 2008, 239–40).
5. Integrated multi-objective methods such as inventory and production planning (Aghezzaf and Landeghem 2002, 4324–32), schedule evaluation (Lopez de Haro, Gershwin, and Rosenfield 2009, 186–89), robust design of manufacturing systems (Sharda and Banerjee 2013, 317–19), and a Markov decision process combined with dynamic programming (DP) (Ni and Werner 2017, 98–100).

The more frequently used methods to control the WIP are simple heuristics and known dispatching rules.

Maimon, Dar-El, and Carmon (1993, 180–83) used the average WIP inventory level as a performance measure to design setup saving schemes for a printed circuit board (PCB) assembly line, which is characterized by long-time setups. When the data took a value over the average WIP for the PCB line of type *i*, the average WIP of a group and the occupation time of the machines were calculated by Grouped Set-Up (GSU) and Sequence-Dependent Scheduling (SDS) methods. The analysis was illustrated using real data from a typical production line. In the paper by Papadopoulos and Vidalis (2001), the objective was to find an optimal buffer allocation (OBA) that minimizes the average WIP inventory subject to a minimum required throughput. A heuristic algorithm, which reduces the search space by over 50% compared to an enumeration, was proposed to find an OBA. Duwayri et al. (2006, 720–22) used the WIP inventory as a measure to evaluate the performance of the proposed setup scheduling heuristic for the bottleneck

workstation in a semiconductor wafer fabrication with a reentrant production flow network, which is a typical modern semiconductor manufacturing system. The variable WIP levels are calculated by means of a workload index of a loop *i*, which is known to be directly correlated with the inventory levels.

Kim and Uzsoy (2009) formulated the single work-center problem as a shortest path problem. Nonlinear clearing functions were used to relate the expected output of a production resource in the planning period to the expected WIP inventory level. Clearing functions permitted an explicit modeling of the WIP levels to ensure that the demand was met while the WIP levels remained within the desired levels. On this basis, two greedy constructive heuristics and a Lagrangian heuristic were proposed for the multi-stage environment. Yang and Posner (2010) highlighted the importance of the knowledge over the location and the size of the WIP in a production system. The problem of minimizing the total WIP cost associated with the value, which was added at each stage during the production process, was modeled for a generalized flow shop in this paper. This problem is NP-hard even for two machines. Based on a value-added model, the authors showed how the unit time WIP cost increased as a job passed through the various stages in the production process. The Shortest Processing Time (SPT) first rule and several simple and intuitive heuristics were presented and empirically evaluated. Cuatrecasas-Arbós et al. (2015) found and quantified the relationships between the WIP inventory, lead time (LT) manufacturing, and the operational variables they depend on. Such relationships provide guidelines and performance indicators in the process management. For a discrete deterministic serial batch process, the authors developed equations to analyze how the WIP and LT depend on the parameters, such as the performance time (of each workstation) and the batch size. Those relationships were extended to processes with different lots and multiple lots. In the paper by Olaitan, Yu, and Alfnes (2017), a WIP control approach was developed and applied in order to manage the inventory by a reduction of the throughput time and its variations under the CONWIP, and the push production control mechanisms in a High Product Mix manufacturing environment. This approach sets a threshold inventory level per product for each of the manufacturing stages and uses the deviation from the respective thresholds to prioritize the production. It can be useful for a JIS policy.

One of the earlier works dealing with the simulation of an inventory control is due to (Conway et al. 1988). Some simple generic serial systems were simulated under conditions, where a storage was provided between the processes to avoid an interference due to the lack of synchronization. This

paper investigated the behavior of the lines buffered in this way. It also explored the distribution and the quantity of the WIP inventory accumulated. One of the conclusions of the authors was that the position as well as the capacity of the buffers were important. Qiu (2005) presented a potentially practical solution for a manageable and well distributed WIP control system by addressing several issues, such as the real-time performance, the scalability, and the reconfigurability. A VPL was also proposed as a method to maintain a certain WIP level. Zhang, Zhang, and Tian (2011) proposed a radio frequency identification (RFID)-enabled real-time production tracking system for the PCB assembly industry. The RFID technology enables an automatic data collection and processing. Various methods of production information processing, such as data filtering and data selection algorithms, were proposed to extract the production tracking information. This information represents, for example, the production progress, WIP inventory, and real-time data by a filtering and selection.

In the paper by Xanthopoulos and Koulouriotis (2014), a discrete-event simulation was developed for the multi-objective optimization control of inventory systems in a multi-stage serial manufacturing. These systems are supposed to be controlled by kanban, base stock, CONWIP, and CONWIP/kanban hybrid mechanisms. Four simulation cases were examined. Some optimal and near-optimal parameters for the control policies were obtained by integrating the proposed simulation models into a multi-objective evolutionary algorithm in order to minimize simultaneously the mean WIP and the mean number of backordered demands. The non-dominated sets were compared in terms of several metrics by means of the Pareto fronts. A metamodeling-based Monte Carlo simulation approach was developed by Li et al. (2016) for responsive production planning to capture accurately the dynamic stochastic behavior of a manufacturing system and to allow a real-time evaluation of several performance metrics of the release plan. The evaluation capability was then embedded into a multi-objective optimization framework to search for near-optimal release plans. The system outputs, namely the WIP levels and job completions, are non-stationary bivariate time series. These outputs interact with the time series representing the customer demand. The results are the fulfillment/nonfulfillment of that demand and the holding cost of both the WIP and the finished goods inventory. The proposed method was applied to solve a production planning problem in a system for semiconductor wafer fabrication.

Various methods of mathematical optimization were used to solve the inventory problem: linear and nonlinear programming (S. X. Bai and Gershwin 1990; J. S. Kim and Leachman 1994), artificial neural networks

(Y.-H. Lin, Shie, and Tsai 2009), and fuzzy logic (Tsourveloudis 2010).

Bai and Gershwin (1990) considered a scheduling problem with WIP inventory in manufacturing systems for the following cases:

1. Single-part-type systems;
2. Multiple-part-type systems;
3. Reentrant systems.

A real-time feedback control algorithm was developed. The WIP inventory was allocated dynamically according to the demand and the machine parameters. A convex function was used to specify the feedback control strategy for the production. It penalized extreme components. The feedback control law with a quadratic function was also applied to verify the optimal production flow rate. The starvation and blockage fractions of a machine were defined using the hedging points, which correspond to the desirable operating states of the system. Kim and Leachman (1994) tried to optimize the WIP using linear programming. The introduced model was developed using the general framework of dynamic production theory. A simulation was performed. Lin, Shie, and Tsai (2009) used an artificial neural network combined with a sequential quadratic programming (SQP) method. The objective was to identify the optimal WIP level and to maximize the throughput rate in a wafer fabrication process. The efficiency of the method finding the optimal WIP level was substantially improved. The mean cycle time and the standard deviation of the cycle time were minimized simultaneously. Tsourveloudis (2010) tested evolutionary tuned fuzzy controllers under variable demand conditions. A significant reduction of the WIP in all production lines and networks was achieved.

In some papers, various useful stochastic methods were proposed for the inventory control. The more frequent ones used Markov chain variations.

Vidalis, Papadopoulos, and Heavey (2005) investigated how to maximize the throughput/minimum of the average WIP when the total service time and the total number of service phases among the stations are fixed. These are the 'work-load' and 'phase-load' allocation problems, respectively. The evaluation of the throughput involved the generation of the transition matrix of the underlying finite state with a continuous time Markov chain. This method in conjunction with the Gauss-Seidel method were used to speed up the convergence in order to solve the resulting system of linear equations and to obtain then the stationary distribution of the Markov chain. In the papers by Sivakumar and Arivarignan (2009), Kilic and Tarim (2011), stochastic systems were investigated. Each of them had an incontrollable throughput process of demand arrivals. In the paper by

Sivakumar and Arivarignan (2009), an inventory system was modeled by means of perishable commodities, in which the arrivals of regular and negative customers were described as an independent Markovian Arrival Process (MAP). The optimal cost rate of the demands was given in this paper. In the paper by Kilic and Tarim (2011), a stochastic inventory system was considered. A demand was represented by a discrete random variable. The authors analyzed the system nervousness resulting from the pure demand uncertainty. The effects of the setup frequency, the demand variability, and various demand patterns on the cost and the instability performances were characterized in terms of (R,S) and (s,S) replenishment inventory policies. Another studied problem was dedicated to find an optimal inventory level for an assembly process. Chen and Plambeck (2008) used Bayesian modeling for a dynamic inventory management with a learning effect. The directions used were:

1. The general demand distributions were discrete ones;
2. The perishability assumption was relaxed;
3. The Bayesian inventory management was extended to consider the effect of learning for the probability that a customer will accept a substitute product.

The use of RFID for a continuous observation of the product flow was suggested. The authors considered the obtained results as most insightful for the inventory and the capacity management of an innovative product like the Toyota Prius Hybrid.

Some recent publications displayed multi-objective approaches to reach advanced results for complex problems.

In the paper by Aghezzaf and Landeghem (2002), an integrated system was considered. It is dedicated to inventory and production planning in a two-stage hybrid production environment with a process production system at the first stage and a job shop production system with batching at the second stage. The stages are separated by an intermediate warehouse, which should be optimized to provide a trade-off between the cost for carrying the inventory of the semi-finished products, the minimum batch size requirement at the first stage, and the required service level at the second stage. An integrated model for planning the production was developed. A model and an objective function for each problem were given. The solution framework included original and known algorithms. Branch-and-bound and linear programming codes together with the CPLEX mixed integer solver were used. Lopez de Haro, Gershwin, and Rosenfield (2009) examined the opposite case, namely a lack of WIP. The authors provided a new approach

to validate the feasibility of schedules in a multiple-step mixed model for unstable manufacturing environments with different changeover times. The approach was based on a new type of a visual representation of the schedules and an estimation of the probability of starvation. Sharda and Banerjee (2013) used multi-objective genetic algorithms, Petri nets, and Bayesian model averaging (BMA) for a robust design of a manufacturing system by an evaluation of multiple decision variables, such as the makespan, WIP, and number of machines. Different forms of manufacturing uncertainties, such as uncertainties in the processing times and the product demand, were considered. The integrated modeling framework coupled with Bayesian methods for the uncertainty representation provided a single tool to design, analyze and simulate the candidate models. Ni and Werner (2017) discussed relationships between the WIP and material handling tools (MHT) in a discrete manufacturing system, particularly in a semiconductor factory. The MHT are responsible for the transitions of the lots between the stations. Any improvement in the MHT has a great potential for reducing the inventory, minimizing the cycle time for the production and enhancing the deliverable orders. The CONWIP and Little's law methodologies were used. A Markov decision process (MDP) was applied to model the MHT problems. A dynamic programming-based algorithm was developed to determine a solution for the MDP model. The control of the WIP level in the whole line within certain lower and upper limits was reached using a special reward function.

2.9 Conclusions

In HT companies with lot processing and capital-intensive products, like semiconductor manufacturing, digital device and automotive industries, to mention a few of them, the management and control of the inventory is of key importance. The reduction of the inventory implies a reduction of the inversions into the infrastructure and the capital involved. Different models for the production control were proposed in the literature and practice to obtain a maximal efficiency of the production. They can be divided into push/pull approaches, where push systems are those where production jobs are scheduled while in pull systems, the start of a job is triggered by the completion of another one (Spearman, Woodruff, and Hopp 1990). The use of kanban tools facilitates the production control, reduces the production time and the WIP in HV/HM companies (Sendil Kumar and Panneerselvam 2007). The adoption of a kanban system improved the efficiency and flexibility of manufacturing.

The attention of researchers is actually focused on pull/kanban/CONWIP and hybrid systems, which resulted in a new paradigm, named JIS. The last one requires a deep systematic investigation of the associated shop and supply chain problems.

There are various reviews, which are dedicated to different aspects of production system paradigms and WIP management, such as in the papers by Golhar and Stamm (1991); Price, Gravel, and Nsakanda (1994); Hum and Lee (1998); Sendil Kumar and Panneerselvam (2007). Inventory management has an inherent connection with planning and logistic problems in the plants. The most recent survey about industrial aspects and the dedicated literature for combined inventory management and routing problems can be met in the paper by Andersson et al. (2010). There exists a comprehensible open access book by Ptak and Smit (2016), which is dedicated to the DDMPR.

CHAPTER THREE

MODELING THE SCHEDULING PROBLEMS

Optimization of a schedule plays a vital role in modern manufacturing, planning and control systems.
(*Shakhlevich, Sotskov, and Werner 2000, 343*)

In big manufacturing plants with lot processing, the scheduling problems are usually difficult to model. The main reasons for this are:

- Large technological routes;
- Complex environments with dozens or hundreds of machines of different brands and production years to make the same operation;
- Frequent changes in the market restrictions;
- Reentrance of items, among others.

Manufacturing plants with lot processing display a variety of shop environments, both traditional ones and recently appeared. The seminal work for modeling scheduling problems was due to Johnson (1954). There was given a simple rule to meet an optimal schedule for a number of jobs on two successive machines. In this chapter, traditional and forthcoming models are highlighted as special models due to their utility for those problems, which are the subjects of this book – planning and scheduling for massive manufacturing of mixed products. A variety of the models considered in the literature as well as some associated questions are briefly discussed below.

3.1 Description of the main problems

Mathematical modeling requires a formal description of a problem. For such purposes, a three-field notation $\alpha|\beta|\gamma$, named *Graham's triplet* (Graham et al. 1979, 288) is commonly used in the scheduling theory. This triplet denotes a scheduling problem in terms of jobs and machines. For a general description of this triplet, the reader is referred to Pinedo (2008, 13–19); Błażewicz et al. (2007, 57–60). Some advanced variants can be found in the

literature (Ruiz and Vázquez-Rodríguez 2010, 57–60; Allahverdi 2015, 347). The triplet describes any scheduling problem in a convenient format. It is also sufficiently flexible and adaptable to define problems with new features. Below, the main values of the three parameters are described.

3.1.1 Notations for indexes, variables and sets

There are no strict rules to use the notations in the modeling of a deterministic scheduling problem. Nevertheless, in the recent literature, standardization elements appeared in the selection of the notations. Some basic notations used in this book are listed below. These notations are typical in the literature. Nevertheless, an author is free to use a proper notation system.

Indexes:
- j - job, $j = 1,\ldots, n$;
- i - machine, $i = 1,\ldots, m$;
- k - stage, $k = 1,\ldots, K$;
- f - family, $f = 1,\ldots, F$.

Parameters:
- p_{ij} - processing time of job j on machine i;
- d_j - due date of job j;
- r_j - release time of job j into the system;
- w_j - weight (importance) of job j.

Sets:
- J - jobs;
- M - machines.

3.1.2 Machine environment (α field)

In the triplet, the α field specifies two characteristics of a problem: the shop type and the machine environment. The main possible values for the machine environment are listed in Table 3-1.

If a process contains only one operation and is realized on one machine, either on a unique or a parallel one, the description of the machine setting is directly included into the α field. If a job has to be processed on more than one machine, the *shop type* (processing order of the product), such as a flow shop, a job shop, an open shop, or others, this must be indicated by the number of machines (unless the number of machines is variable). Thus, Fm, $J4$ and $O3$ denote an m-machine flow shop, a 4-machine job shop, and a 3-machine open shop, respectively. If the number m of machines is variable

or it is a part of the input, the parameter m is skipped in the notation, for example, F or P only.

The principal shop types are listed in Table 3-2.

Notation	Description
∅	*Single machine*: all jobs are processed on the same machine.
Pm	*m identical parallel machines*: a job may be processed on any of *m* machines.
Qm	*m uniform parallel machines*: the time a job is processed on a machine depends on the machine speed.
Rm	*m unrelated parallel machines*: the time a job spends on a machine depends on the machine speed and on the job.
Dm	*m dedicated machines*: each machine is specialized for the execution of certain jobs, so the jobs to be processed are known in advance for each machine.

Table 3-1. Machine environments.

Notation	Description
1	*Single machine*
Fm	*m*-machine *flow shop* (FS): all jobs have a unidirectional flow on the machines.
Jm	*m*-machine *job shop* (JS): each job has a specified processing order through the machines.
Om	*m*-machine *open shop* (OS): there are no restrictions on the processing sequence of the jobs on the machines..
Mm	*m*-machine *mixed shop*: for some jobs, the sequence of machines is given while for other jobs not.
FFk	*k*-stage *flexible flow shop* (FFS): it is a generalization of a flow shop, when the production is realized at *k* stages, and at least one of them has an identical parallel machine environment.
HFk	*k*-stage *hybrid flow shop* (HFS): it is a generalization of an FFS, when one or more of the *k* stages have a non-identical parallel machine environment or a shop.
AFk	*k*-stage *assembly flow shop* (AFS): it is a generalization of a flow shop, when the first stages are parallel shops dedicated to the production of the components, and the last stage is performed on an assembly machine.
FJk	*k*-stage *flexible job shop* (FJS): it is a generalization of a job shop, when one or more stages have a parallel machine environment.
FOk	*k*-stage *flexible open shop* (FOS): it is a generalization of an open shop, when one or more stages have a parallel machine environment.

Table 3-2. Shop types (α field).

For generalized multi-stage shops, such as FFS, HFS, AFS, FJS, and FOS, both the shop type and the machine environment at each stage can be described in detail. The α field is composed of four parameters, denoted by α_1, α_2, α_3, and α_4, which are sequenced in the order: $\alpha_1\alpha_2$ ($\alpha_3\alpha_4^{(1)}$, $\alpha_3\alpha_4^{(2)}$, …, $\alpha_3\alpha_4^{(\alpha 2)}$). Here, the value of α_1 indicates the shop type, and α_2 denotes the number of stages. Then each stage k can be described explicitly by the parameters $\alpha_3\alpha_4^{(k)}$. In such a case, for each stage k, the parameters α_3 and α_4 together describe the machine environment in the shop at stage k. More specifically, the type of the machines is written in the α_3 position, while α_4 specifies the number of machines at the stage.

If there are several consecutive stages with the same machine environments the parameters α_3 and α_4 can be grouped as follows: $((\alpha_3\alpha_4)^{(i)})_{i=s}^{l}$, where s and l are the indexes of the first, and the last consecutive stages, respectively. The parameter $\alpha_3 \in \{\varnothing, P, Q, R, D\}$ is according to Table 3.1. The value $\alpha_3 = \varnothing$ implies that the α_3 position is omitted in the single machine case (Błażewicz et al. 2007, 68; Ruiz and Vázquez-Rodríguez 2010, 2).

In the paper by Potts and Kovalyov (2000, 230), the notations $\tilde{P}$, $\tilde{Q}$, $\tilde{R}$, $\tilde{F}$, $\tilde{J}$, $\tilde{O}$ are used for the corresponding environments with the characteristic of batch processing. No other specific symbols have been reported in the literature for the α field.

The following examples illustrate the use of the settings described above.

$FF2(Pm^{(2)})$ and $FF2(1^{(1)}, Pm^{(2)})$ are two notations, which denote a two-stage FFS with a single machine at the first stage, and m identical machines in parallel at the second stage.

Problem $HF4(1, (Om_i^{(i)})_{i=2}^{3}, R2)$ refers to a four-stage HFS with a single machine at the first stage, and m_2 and m_3 uniform parallel machines at the second and third stages, and two unrelated parallel machines at the fourth stage, respectively.

Problem $AF2(Jm^{(1)}, 1)$ describes a two-stage AFS with an m-machine job shop at the first stage and one assembly machine at the second stage.

Problem $FJ5(P3^{(2)})$ denotes a five-stage FJS with two identical parallel machines at the third stage.

3.1.3 Shop conditions (β field)

The β field provides the shop properties as well as other processing conditions and important details, which characterize the problem. Multiple entries may be enumerated. This field may also be empty if there are no special conditions.

The most common model properties, which are frequently associated with a problem with lot scheduling, are listed in Table 3-3.

Notation	Description
$\varnothing$	No special properties.
batch(b)	Batch processing: a machine is able to process up to b jobs continuously without any setup.
Brkdwn	A machine may not be continuously available due to a breakdown or shutdown.
block	A blocking may occur when the buffers between two successive machines have limited capacities.
$b_k = m_k$	There are m_k batch processing machines at stage k.
D_j	A soft due date for the completion of a job j is given. If a job is completed later than its due date, a penalty is imposed.
$\bar{d}_j$	A strict due date of the completion of a job j is given, referred to as a *deadline.*
fmls	Job families: The n jobs belong to F different job families. The jobs from the same family are processed on a machine one after another without any setup in between.
lot	Lot processing.
M_{jk}	Machine eligibility restrictions: Not all machines at stage k are capable to process job j.
mop	There are multiple orders per job (MOJ).
nwp	No-wait condition: the jobs cannot wait between their operations.
prmp	Preemptions of jobs are allowed.
R	Removal time: a machine becomes free only after the setup of the job has been removed.
rcrc	Recirculation: A job may visit a machine or a work center more than once. Reentrance operations are allowed.
split	Splitting of jobs: A job/lot can be split into several parts/sublots so that their operations may overlap. In the case of a shop with lot processing, this situation is referred to as lot streaming.
s_{si}	Sequence-independent setup times: The setup time on a machine depends only on the job to be processed.
s_{sd}	Sequence-dependent setup times (SDST): The setup time to process the next job on a machine depends on the previously processed job.
sc_{sd}	Sequence-dependent setup cost (SDSC): The setup cost to process the next job on a machine depends on the previously processed job.
s_{psd}	Past-sequence-dependent setup times (p-s-d): The setup time to process the next job on a machine is proportional to the length of the already scheduled jobs.
$s_{sd,gf}$	Sequence-dependent family setup times. The setup time on a machine to process a job/batch, which belongs to family g, depends on the previous job/batch family f processed on this machine.
$sc_{sd,gf}$	Sequence-dependent family setup costs. The setup cost for a machine to process a job/batch, which belongs to family g, depends on the previous job/batch family f processed on this machine.
w_j	Weight or importance of job j: it is the priority factor of the jobs in the system.

Table 3-3. Shop properties for problems with lot processing (β field).

3.1.4 Performance measures (γ field)

The γ field establishes the objective function to be minimized. An objective function is defined using the following parameters associated with a job j in a schedule (Table 3-4).

Notation	Description
C_j	Completion time
$F_j = C_j - r_j$	Flow time
$L_j = C_j - d_j$	Lateness
$T_j = \max\{C_j - d_j, 0\}$	Tardiness
$E_j = \max\{d_j - C_j, 0\}$	Earliness

Table 3-4. Schedule performance measures (γ field).

The most frequently used objective function to be minimized in a scheduling problem is the completion time when the last job leaves the system, referred to as the *makespan* or $C_{\max}$. The most common objective functions used in scheduling problems, which are associated with lot processing, are listed in Table 3-5.

Some illustrative examples of scheduling problems described by using Graham's triplet are given below.

The notation $FF6||C_{\max}$ describes an FFS with six stages and the $C_{\max}$ criterion. This shop has no special properties.

Problem $FJk|r_j, s_{sd}, rcrc|\ \Sigma w_j T_j$ refers to an FJS with k stages (work centers). The jobs have different release dates of the jobs. There appear sequence-dependent setup times, and recirculation (reentrance) is allowed, i.e., a job may visit a work center more than once. The objective is to minimize total weighted tardiness. Such a problem typically arises in a semiconductor facility (Pinedo 2008, 20).

Problem $AF2(Jm^{(1)}, D2)|$ batch, $w_j, D_j \mid T_{\max}$ describes a two-stage AFS with an m-machine job shop at the first stage and two dedicated assembly machines at the second stage. The processing is organized in batches. Every job has a priority factor w_j and a deadline D_j. The goal is to minimize maximum tardiness.

Some overviews on the complexity of scheduling problems can be met in the literature (Lenstra, Rinnooy Kan, and Brucker 1977; Potts and Van Wassenhove 1992).

Notation	Description
$C_{max} = \max\{C_j\}$	Schedule length (makespan)
$E_{max} = \max\{E_j\}$	Maximum earliness
$L_{max} = \max\{L_j\}$	Maximum lateness
$T_{max} = \max\{T_j\}$	Maximum tardiness
$D_{max} = \max\{D_j\}$	Maximum delivery time
$C = \sum_{j=1}^{n} C_j$	Total completion time
$F = \sum_{j=1}^{n} F_j$	Total flow time
$E = \sum_{j=1}^{n} E_j$	Total earliness
$L = \sum_{j=1}^{n} L_j$	Total lateness
$T = \sum_{j=1}^{n} T_j$	Total tardiness
$W = \sum_{j=1}^{n} W_j$	Total waiting time
$U = \sum_{j=1}^{n} U_j$	Number of tardy jobs, $U_j = 1$ if $C_j > d_j$ and 0 otherwise.
$C_w = \sum_{j=1}^{n} w_j C_j$	Total weighted completion time
$F_w = \sum_{j=1}^{n} w_j F_j$	Total weighted flow time
$E_w = \sum_{j=1}^{n} w_j E_j$	Total weighted earliness
$U_w = \sum_{j=1}^{n} w_j U_j$	Weighed number of tardy jobs, $U_j = 1$ if $C_j > d_j$ and 0 otherwise.
$T_w = \sum_{j=1}^{n} w_j T_j$	Total weighted tardiness
$W_w = \sum_{j=1}^{n} w_j W_j$	Total weighted waiting time
$\bar{T} = \frac{1}{n}\sum_{j=1}^{n} T_j$	Mean tardiness
$\bar{D} = \frac{1}{n}\sum_{j=1}^{n} D_j$	Mean delivery time
$\bar{F}_w = \sum_{j=1}^{n} w_j F_j / \sum_{j=1}^{n} w_j$	Mean weighted flow time
$\bar{E}_w = \sum_{j=1}^{n} w_j E_j / \sum_{j=1}^{n} w_j$	Mean weighted earliness
$\bar{L}_w = \sum_{j=1}^{n} w_j L_j / \sum_{j=1}^{n} w_j$	Mean weighted lateness
$\bar{T}_w = \sum_{j=1}^{n} w_j T_j / \sum_{j=1}^{n} w_j$	Mean weighted tardiness
$\bar{D}_w = \sum_{j=1}^{n} w_j D_j / \sum_{j=1}^{n} w_j$	Mean weighted delivery time

Table 3-5. Objective functions for scheduling problems.

3.1.5 Multi-objective scheduling

In a scheduling problem, two or more different criteria can be considered together. In such a case, the criterion can be accomplished in a number of ways.

A multi-objective optimization problem can be defined as the optimization of a vector of decision variables formed by a set of objective functions, which are usually in conflict with each other. Such a problem can be formulated as follows:

$\{f_1(X), f_2(X), \dots, f_M(X)\} \rightarrow \min$,

s.t.

$X \in D$,

where

X - vector of decision variables;

$f_j(X)$- objective function j, j =1,…, M;

D - set of feasible solutions.

A solution X^1 is said to be *dominated* by a solution X^2 if for all j = 1, 2, …, M, $f_j(X^1) \geq f_j(X^2)$ and $f_j(X^1) > f_j(X^2)$ for at least one solution. A solution X^1 is *Pareto-optimal* if there is no $X' \in D$ that dominates X. The set of all Pareto-optimal solutions is called the *Pareto-optimal set* or *Pareto-optimal front*.

In a multi-objective scheduling problem, it is very problematic to obtain a unique optimal solution. Usually, a set of equally good optimal solutions, which correspond to a Pareto-optimal set, is obtained. Moving the solutions from one point to another one within the Pareto set, one objective is improved while others become deteriorated, because they are usually contradictory. Normally, a specific non-dominant solution cannot be chosen, since all solutions are equally competitive, and none of them can dominate the others. Therefore, other additional information is required. A successful policy to solve a multi-criteria scheduling problem permits to reduce the number of objectives and focuses the optimization process on a few crucial ones. There are several available methods to solve multi-objective scheduling problems. One of them is the weighted-average of the objectives, see Husseinzadeh Kashan, Karimi, and Jolai (2010).

For a bi-criteria problem, there are two principal approaches:

1. The two criteria are assumed to be prioritized as *primary* and *secondary*. The objective is to find the best schedule for the

secondary criterion among all alternative optimal schedules for the primary criterion. The less important criterion is minimized subject to the restriction that the most important criterion is minimized. The triplet $1||\gamma_2/\gamma_1$, where γ_1 is the primary criterion, and γ_2 is the secondary criterion, denotes this problem;

2. A weighting function is used to combine the two criteria. A decision maker assigns different values of a weight (importance) to the criteria γ_1 and γ_2. A scheduling problem with the two criteria γ_1 and γ_2, and a given weighting function f is denoted by $1||f(\gamma_1, \gamma_2)$. For scheduling problems, it makes sense to restrict f to a linear combination of various regular criteria, for example, $f(\gamma_1, \gamma_2) = \lambda_1 \gamma_1 + \lambda_2 \gamma_2$, where λ_1 and λ_2 are non-negative integers.

The majority of the researches, which reported about multi-criteria scheduling, has been concerned with bi-criteria single-machine scheduling problems due to the complexity of multi-objective optimization (L. L. Liu, Ng, and Cheng 2009).

3.2 Single-stage production systems

In the modern literature, one can find an enormous variety of models for production systems, which require new approaches and new descriptions of the environment. The basic models are intensively studied, and new problems are inspired by the trend to be closer to real production, which usually is difficult and complex to model. Nevertheless, even new models are based on a few traditional machine environments, which are discussed below.

3.2.1 Single machine

Single machine problems have been intensively studied in the previous decades. Błażewicz et al. (2007, 73) noted that the problem $1|prec|C_{\max}$, where all jobs are assumed non-preemptible, ordered by some precedence relation, and available at time $t = 0$, is a trivial scheduling problem with the schedule length $C_{\max} = \sum_{j=1}^{n} p_j$. The maximum lateness problem $1||L_{\max}$ can be solved by Jackson's rule: schedule the jobs in order of non-increasing delivery times (Jackson 1955; L. A. Hall and Shmoys 1992, 22) . The total weighted completion time problem $1||\Sigma w_j C_j$ can be solved by the Weighted Shortest Processing Time (WSPT) first rule (Smith 1956, 62–63), and Moore's algorithm (Moore 1968, 102–4) solves the number of tardy jobs problem $1||\Sigma U_j$.

Moreover, the problems $1|r_j|C_{\max}$ and $1||L_{\max}$ are equivalent (Błażewicz et al. 2007, 73–74). However, the majority of the classical non-preemptive single machine problems, such as the weighted number late jobs $1||\Sigma w_j U_j$ and the total (weighted) tardiness problems $1||\Sigma w_j T_j$ are NP-hard (Lenstra, Rinnooy Kan, and Brucker 1977, 356–57).

The actual interest is focused mainly on complex problems of multi-agent scheduling (L. Tang et al. 2017; Shisheng Li et al. 2018; Yin et al. 2018), multi-criteria scheduling (L. L. Liu, Ng, and Cheng 2009), and the simultaneous consideration of various factors such as:

- Learning effects (Biskup 1999; J.-B. Wang 2007b; 2008; J.-B. Wang and Li 2011; Pei, Liu, Pardalos, Migdalas, et al. 2017; Fan et al. 2018; Pei et al. 2019);
- Batch/lot sizing (Hazır and Kedad-Sidhoum 2014; Nobil, Nobil, and Cárdenas-Barrón 2017);
- Machine maintenance and supply chain uncertainties (W.-J. Chen 2009);
- Complex or costly setups (J.-F. Chen 2009; Giglio 2015; W. Huang, Wu, and Liu 2018; Kress, Barketau, and Pesch 2018);
- Batch machines with non-identical capacities for different jobs (W. Huang, Wu, and Liu 2018);
- Multiple orders per job (Mason and Chen 2010; Sobeyko and Mönch 2015);
- Time-dependent, resource-dependent, and controllable processing times (Su and Lien 2009; J.-B. Wang and Li 2011; Mor and Mosheiov 2014; Giglio 2015; Pei, Liu, Pardalos, Fan, et al. 2017; Pei, Liu, Pardalos, Migdalas, et al. 2017);
- Dynamically arriving jobs (Y.-C. Choi 2016; Pei et al. 2016);
- Energy consumption (Y.-C. Choi 2016);
- Deteriorating jobs (J.-B. Wang 2007b; Pei, Liu, Pardalos, Fan et al. 2017; Pei, Liu, Pardalos, Migdalas et al. 2017; L. Tang et al. 2017; Fan et al. 2018; Chunetg, Gupta, and Qiu 2019; Kong et al. 2019);
- Unbounded batches (L. L. Liu, Ng, and Cheng 2010), etc.

In big plants with lot processing, a single machine environment may be associated with a bottleneck problem, for example, the programming of semiconductor wafer probing facilities may be formulated as a single machine problem (Bang and Kim 2011, 671).

3.2.2 Parallel machines

The problems with a single-stage environment and parallel machines may be found in small manufacturing plants as well as in big factories representing a part of a production process. There are four possibilities for the set of parallel machines. They may be:

- Identical (P*m*);
- Uniform (Q*m*);
- Unrelated (R*m*);
- Dedicated (D*m*).

It is a typical situation for a big company, when the parallel machines are first identical, then after some time, a part of the old machines is substituted, new machines are added due to the extension of the production. The later installed machines are usually not identical with the old ones because they are more recently produced. They are of other brands, or the manufacturing conditions in the plant have been changed. Therefore, the parallel machines, which make more or less the same operation, have different speeds and other technological characteristics. *Uniform* machines have proper speeds, which do not depend on the job under processing. In contrast, the speeds of *unrelated* machines depend on both the machine and the job. *Dedicated* machines are also frequent in big plants. They may represent different platforms, where everyone performs some specific operations for a part of the products. Another example was met at the reed switch production, where some parallel machines at the last stage are dedicated to make different shapes of knifes (Section 10.3).

Similar to single machine models, parallel machine environments have also been intensively investigated in previous years. The total completion time problem $Pm||\Sigma C_j$ was solved by Conway, Maxwell, and Miller in 1967 (T. C. E. Cheng and Sin 1990, 287) using the SPT rule, which requires O(n log n) steps. The parallel machine makespan $Pm||C_{\max}$ and the total weighted completion time $Pm||\Sigma w_j C_j$ problems on identical machines are found to be NP-hard. Therefore, these more general problems have also a high computational complexity.

In this class of models, one can meet a number of researches considering lot processing. Chen et al. (2002) proposed an optimal short-term scheduling of multi-product single-stage batch plants with parallel lines. Chen and Powell (2003) proposed an exact branch-and-bound algorithm for scheduling multiple families with the objective to minimize total weighted completion time of the jobs and the same problem with the objective to

minimize the weighted number of tardy jobs. Geismar, Dawande, and Sriskandarajah (2004) studied the problem of sequencing operations in buffer-less robotic cell flow shops with parallel machines focusing on the cells that produce identical parts. The objective was to find a cyclic sequence of robot moves that maximizes the steady state throughput. The papers by Chen, Ye, and Zhang (2006) and Chen (2009; 2015) were concerned with the problem of scheduling independent jobs on unrelated parallel machines with different classes of setup times and due date constraints with the makespan and total weighted completion time criteria. This problem is NP-hard. Two polynomial schemes were proposed. Pfund et al. (2008) developed a greedy approach for the NP-hard problem of scheduling jobs on parallel machines with setup times and ready times. Fu et al. (2010) proposed online scheduling on two parallel machines with batching and limited restarts to minimize the makespan. The authors provided a best online algorithm with a competitive ratio $(\sqrt{3}+1)/2 \approx 1.366$ and showed that no online algorithm can have a competitive ratio less than 1.298. Lushchakova and Strusevich (2010) addressed a scheduling problem with two parallel machines and incompatible jobs. The authors modeled the machine environment as parallel and dedicated machines, which means that for each machine, the jobs to be processed are known in advance. Wang, Wang, and Zheng (2016) proposed a hybrid estimation of a distribution algorithm with iterated greedy search (EDA-IG) for a similar case. Actually, multi-criteria problems with a combination of different constraints are also considered in the literature (I.-L. Wang, Wang, and Chen 2013; Fakher, Nourelfath, and Gendreau 2017; Junqueira and Morabito 2017).

The reader can meet a state-of-the-art review on parallel machine scheduling research in the paper by Cheng and Sin (1990).

3.3 Multi-stage production systems

In the scheduling theory, there are three principal configurations, called *shops*: flow shop (FS), job shop (JS), and open shop (OS). A shop describes an integrated configuration of machine resources and the order of their use to model complex machine environments. These shops are considered below more in detail.

3.3.1 Flow shop

A flow shop, also called a conveyor, is a model of an environment with a unidirectional flow of the jobs over the m machines, which are arranged in

series. Every job starts its processing on an initial machine, proceeds through several intermediate machines, and concludes on a final machine. Usually, this process starts with product parts or a raw material, which leave the system as finished goods. Such a structure is typical for assembly lines. In some flow shop models, a job may skip a particular machine, but the flow is conserved to be unidirectional. An example is given in Fig. 3-1. For instance, although every job must be processed from machine 1 to machine m, some jobs may skip a machine. For example, a job may go from machine 1 to machine 3 skipping machine 2, and then it goes to machine 4.

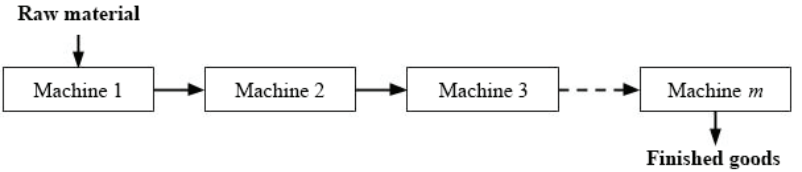


Fig. 3-1. An m-machine flow shop.

Permutation schedules, in which the jobs are processed in the same order on all machines and a machine is not kept idle if a job is ready for processing, are usually used as solutions of the flow shop problems. Therefore, one of the possible $n!$ permutations of the jobs determines an optimal sequence giving a unique sequence, in which they are processed on the machines. If a non-permutation schedule is considered, i.e., a schedule that does not require the same job sequence on all machines, a search among $(n!)^m$ possible sequences would be required to obtain an optimal solution, what is extremely time consuming.

Problem $F2||C_{\max}$ is widely known in the scheduling literature as *the Johnson problem*. It has been solved by Johnson with a simple O(n log n) algorithm (Johnson 1954, 64–65). In contrast, problem $F3||C_{\max}$ is already NP-hard. This means that its exact solution in polynomial time is unlikely. Gupta (1986) proved that problem $Fm|ST_{sd}|C_{\max}$ is NP-hard. Gupta and Darrow (1986) proved that problem $F2|ST_{sd}|C_{\max}$ is also NP-hard and that for this problem, there does not necessarily exist a permutation schedule, which minimizes $C_{\max}$. Four efficient approximate algorithms were proposed to find approximate schedules for the problem: a branch-and-bound method, two Johnson's rule-based heuristics, and a neighborhood search technique.

A usual policy for lot processing is the First-In, First-Out (FIFO) policy. According to this rule, the processing order is given as a permutation of the jobs, i.e., the processing order of the jobs is identical on all machines, and

reordering the jobs on the machines is not permitted. The quality of a schedule can be improved if the lots are allowed to be divided. In this case, the resulting solution is not a permutation. Such solutions may also be efficient in other cases, for example, in the case of complex setups or arrivals of jobs with a higher priority.

Flow shops are frequent models in industries with lot processing. Recent publications consider shop characteristics such as, for example, complex setups of the machines, lot splitting, no-wait restrictions, limited buffers or transport operations between the machines, etc., which increase the complexity of the problem, making the models closer to real production conditions. For instance, lot sizing and scheduling of two-machine flow shops with machine setup and cleaning activities were considered in the food industry (Stefansdottir, Grunow, and Akkerman 2017). The minimization of the waiting time variation in a generalized two-machine flow shop with waiting time constraints and skipping jobs in the wafer fabrication was considered in the paper by Yu, Kim, and Lee (2017). The problem $F2|\overline{d_i}, skip|\sum|w_i - \overline{w}|$ was reported to be NP-hard despite a simple machine environment. Wang et al. (2012) dealt with a flow shop scheduling problem in a semiconductor manufacturing process. The problem seeks to minimize the number of tardy jobs and the makespan when sequence-dependent setup times, release times, due dates, and tool constraints are given. The paper by Rossit et al. (2016) addressed a non-permutation *m*-machine flow shop scheduling problem with lot streaming. A mathematical model was proposed. It was noted that the majority of the recent publications in the flow shop area were related to lot streaming and setup considerations.

An efficient heuristic, called the *NEH algorithm*, was proposed by Nawaz, Enscore, and Ham (1983) to solve the flow shop sequencing problem. It is a constructive algorithm, which shows a better performance compared to other heuristics, especially for large instances in both static and dynamic sequencing environments. The NEH heuristic is a curtailed-enumerative algorithm, which is based on the premise that jobs with larger total processing time should be given a higher priority than jobs with a smaller total processing time.

The steps of the algorithm are as follows.

Step 1. For each job *j* calculate

$P_j = \sum_{i=1}^{m} p_{j,i}$,

where $p_{j,i}$ is the processing time of job *j* on machine *i*, $j = 1,\ldots,n$, $i = 1,\ldots,m$.

Step 2. Arrange the jobs in descending order of P_j.

Step 3. Select the two jobs from the first and second positions of the list generated in Step 2; find the best sequence for these two jobs by calculating the makespan for the two possible sequences. Do not change the relative positions of these two jobs with respect to each other in the remaining steps of the algorithm. Set $j = 3$.

Step 4. Select the job in the position j of the list generated in Step 2; find the best sequence by placing it at all possible j positions in the partial sequence found in the previous step without changing the relative positions to each other of the already assigned jobs. The number of enumerations at this step is equal to j.

Step 5. If $n = j$, STOP, otherwise set $j = j+1$. Go to Step 4.

The number of enumerations in the algorithm is

$$\frac{n(n+1)}{2} - 1,$$

where n complete sequences are generated and the remaining ones are partial sequences.

In the recent literature, one can meet modifications of the NEH algorithm. It is also used to receive an initial solution for a multi-objective flow shop problem (MFSP) (Chakravarthy and Rajendran 1999) and to verify the quality of metaheuristics (Dong, Huang, and Chen 2008; Eren and Güner 2008).

A review of the flow shop scheduling research with setup times was presented by Cheng, Gupta, and Wang (2000). A review and evaluation of multi-objective algorithms for the flow shop scheduling problem was given by Minella, Ruiz, and Ciavotta (2008). A more recent review on the flow shop scheduling problem was presented by Sun et al. (2011). This work was dedicated to multi-objective optimization algorithms.

3.3.2 Job shop

A job shop represents the following scheduling environment. There are a set of n jobs and a set of m machines with the constraint that each machine can handle at most one job at a time and each job has a specified processing order through the machines. This means that for any two jobs, the order, in which they pass the machines, may be distinct. This is the difference between a job shop and a flow shop, where in the latter problem the processing order on the machines is identical for all n jobs so that a flow of jobs is formed. A usual objective of a job shop problem is to minimize the makespan.

A machine i, $i = 1,\ldots, m$, may be *initial*, *intermediate*, or *terminal* for a job j. A job j in a job shop can arrive at a machine i to execute the first operation or a part of the WIP after its processing on the previous machine. A job j can leave the shop after processing on machine j or being kept as a WIP waiting for processing on the next machine (Fig. 3-2).

Applegate and Cook (1991, 149) noted that the job shop problem is not only NP-hard, but it also has the well-earned reputation of being one of the most computationally stubborn combinatorial problems. This intractability is one of the reasons of the attention of the researchers.

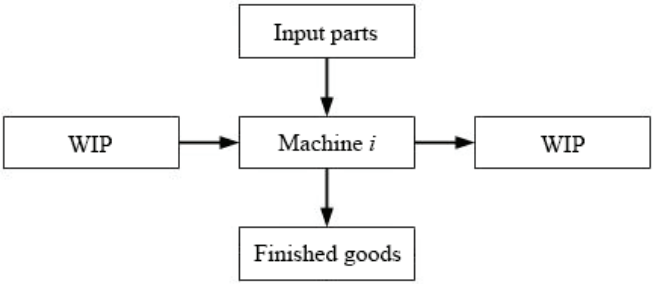


Fig. 3-2 Processing of a job on machine i: machine i might be an initial, an intermediate or a final one.

Problem $J2||C_{\max}$ without a repetition of machines for the jobs is one of the few job shop scheduling problems, for which a polynomial time algorithm was found. The solution was obtained by a reduction of this problem to problem $F2||C_{\max}$ (Pinedo 2008, 180): Let $J_{1,2}$ denote the set of jobs that have to be processed first on machine 1, and $J_{2,1}$ the set of jobs that have to be processed first on machine 2. The solution was obtained by applying the SPT rule to sequence the set $J_{1,2}$ and the Largest Processing Time (LPT) first rule to sequence the set $J_{2,1}$.

It is convenient to represent the problem $Jm||C_{\max}$ without recirculation by a disjunctive graph $G = (V, A, E)$. In this notation, V denotes the set of vertices (i, j), which correspond to the processing of job j on machine i, A is the set of *conjunctive* arcs (solid lines), which represent the routes of the jobs over the machines, and E is the set of *disjunctive* arcs (dashed lines) that go in both directions. The arcs of the disjunctive graph E are formed by the operations, which belong to two different jobs but have to be processed on the same machine. In graph theory, a graph, in which any two vertices are connected to each other, is called *a clique*. The disjunctive arcs of the graph E form m cliques of double arcs, one clique for each machine. In

addition, the graph has two dummy vertices: a source *s* and a sink *t*. In a feasible schedule, one disjunctive arc is selected from each pair in such a way that the resulting directed graph is acyclic.

In the obtained acyclic graph, a longest path from *s* to *t* determines the makespan of a feasible schedule. Then the problem of minimizing the makespan is reduced to finding a selection of one arc of each pair of disjunctive arcs so that the resulting graph is acyclic and the longest path from *s* to *t* becomes minimal. An illustrative example is given in Fig. 3-3.

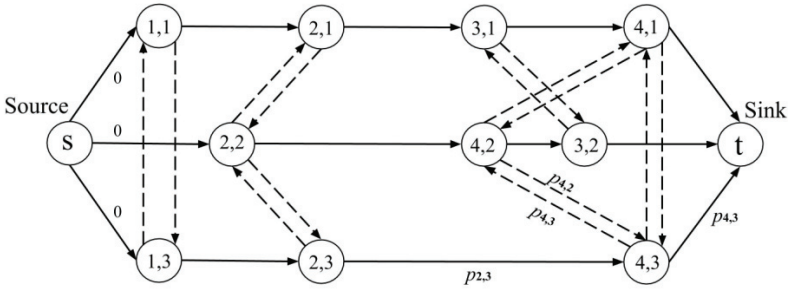


Fig. 3-3. Directed graph for a four-machine job shop to process three jobs with given sequences of the machines: q_1 = (1,2,3,4), q_2 = (2,4,3), q_3 = (1,2,4), respectively; p_{ij} is the processing time of job *j* on machine *i*.

Garey, Johnson, and Sethi (1976, 127–28) noted that the job shop schedule length minimization problem is among the hardest combinatorial optimization problems, which is NP-hard even for two machines. Heuristic procedures are widely used because of its inherent intractability. The *Shifting Bottleneck* (SB) heuristic is an efficient method to treat the $C_{\max}$ and $L_{\max}$ objectives in a job shop. It was proposed by Adams, Balas, and Zawack (1988). It is an iterative approximation method. It sequences the machines one by one, selecting each time the machine identified as a bottleneck among the machines that are not yet sequenced. Every time when a new machine is sequenced, all previously established sequences are locally reoptimized. The reoptimization is performed by freeing up and resequencing each machine in turn with the sequences on the other machines held fixed. Both the bottleneck identification and the local reoptimization procedures are based on repeatedly solving certain one-machine scheduling problems.

With the elimination of some drawbacks, Dauzere-Péres and Lasserre (1993) modified the procedure. Later, Wenqi and Aihua (2004) proposed an improved shifting bottleneck procedure (ISB) for the job shop scheduling

problem with the objective to minimize the makespan. A refined version, which combined the ISB with the strategy of backtracking, was also presented. An integration of lot sizing and scheduling decisions in a job shop was considered in the paper by Kumar et al. (2013, 1634), where the shifting bottleneck technique was used.

Due to the complexity of the model, there are a few works published recently about this subject in the lot processing context. A tutorial survey of job shop scheduling problems was presented by Cheng et al. (1996). A review and a classification of job shop scheduling approaches was given by Parveen and Ullah (2010). The most recent review on job shop scheduling with setup times was offered by Sharma and Jain (2015).

3.3.3 Open shop

In an *m*-machine open shop, each of *n* jobs has to be processed on each of the *m* machines. There are no restrictions on the processing sequence of a job on the machines. Some of the machines may be skipped. In such cases, the corresponding processing times are zero. The route of a job is described by the sequence of machines to visit.

Şen and Benil (1998, 136) noted an interesting and important characteristic of the open shop model. Namely, the one-to-one correspondence between 1) an open shop problem with *m* machines, *n* jobs, and processing times p_{ij} of job *j* on machine *i,* and 2) an open shop problem with *n* machines, *m* jobs, and the processing times p_{ij} of job *i* on machine *j*. In the second case, the machines are considered as jobs and the jobs as machines. The Gantt chart given in Fig. 3-4 illustrates this relationship. There are a two-machine, three-job open shop and the corresponding three-machine, two-job open shop. One can note that the completion time of job *j* on machine *i* in the first case is equal to the completion time of job *i* on machine *j* in the second case. It may also be seen that the route of job *j* in the first case is equivalent to the sequence of jobs on machine *j* in the second case, and vice versa. Optimizing the makespan in the first case is equivalent to optimizing the makespan in the second case. This duality of open shops can be used to simplify the solution of the routing problem.

The open shop scheduling problem was introduced to model several real-world applications that did not quite fit under the flow shop model (Gonzalez and Sahni 1976; Gonzalez 2004). A linear time O(n) algorithm was proposed to solve the problem $O2||C_{\max}$. A polynomial time algorithm was also developed for the problem $Om|prmp|C_{\max}$, where $m > 2$. The authors also showed that if $m > 2$, the problem $Om||C_{\max}$ with non-preemptive schedules is NP-hard. For $m = 2$, the open shop problems, which

with minimize maximum lateness $O2||L_{\max}$ and total completion time $O2||\Sigma C_j$, were reported to be NP-hard (Achugbue and Chin 1982).

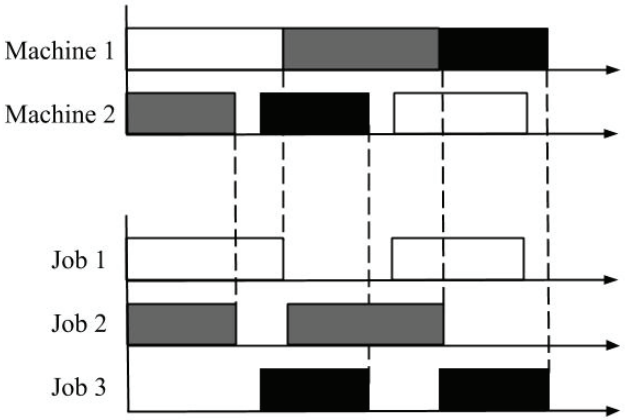


Fig. 3-4. Routing symmetry in an open shop problem.

Pinedo (2008, 218–20) proposed two rules to solve the open shop problem: the longest alternate processing time first (LAPT) rule and the longest total remaining processing on other machines first (LTRPOM) rule. The LAPT is an exact algorithm for the problem $O2||C_{\max}$. The LTRPOM rule is a generalization of the LAPT rule for the case of an open shop with more than two machines. According to the *LTRPOM rule*, the processing required on the machine currently available does not affect the priority level of a job. The LTRPOM rule is a constructive heuristic for the problem $Om||C_{\max}$ with $m \geq 3$. However, this heuristic does not always result in an optimal schedule because the problem $Om||C_{\max}$ is NP-hard for $m \geq 3$. Pinedo noted that the LAPT rule for $O2||C_{\max}$ is one of the few polynomial time algorithms for non-preemptive open shop problems.

Strusevich (1999) considered the problem of scheduling n jobs in a two-machine open shop to minimize the makespan, denoted as $O2|\tau_i|C\max$, where inter-stage transportation times are involved and preemptions are not allowed. This problem was known to be NP-hard. The author presented an O(n log n) algorithm with a worst-case performance ratio of 3/2. Glass, Gupta, and Potts (1994, 380) used a network model to analyze three-machine flow shop problems with lot streaming. Then they extended the results to $O3$ problems and $J2$ problems. The authors employed the algorithm given by Gonzalez and Sahni (1976, 666–67) to solve m-machine problems with two equal-sized sublots.

If all $p_{i,j}$ values are 0 or 1, a *unit-processing time* open shop problem is obtained. The problem $Om|p_{ij} \in \{0,1\}|C_{\max}$ corresponds to a bipartite graph edge coloring problem.

In a *bipartite* graph, the set of vertices is partitioned into two sets, say A and B, such that every edge connects a vertex in A to a vertex in B. The *bipartite graph edge coloring problem* consists of assigning a color to each edge in the graph in such a way that no two adjacent edges have the same color and the total number of different colors utilized is as small as possible. An edge from a vertex in the set A to a vertex in the set B represents a job operation with unit processing time. Each color represents a time unit. The coloring rules guarantee that an edge coloring for the graph corresponds to a feasible open shop schedule with unit processing times. The makespan or maximum completion time of a job corresponds to the number of different colors used to color the bipartite graph (Gonzalez 2004, 2).

Potts and Van Wassenhove (1992) gave a survey about solution algorithms and the complexity of problems, which integrated batching and lot sizing into scheduling. The authors noted that at this time, no results were available for the maximum completion time in a two-machine open shop. Since that time, some studies were reported. Liaw, Cheng, and Chen (2005) considered the NP-hard problem $O2|nwt|C_{\max}$. An exact algorithm, based on a branch-and-bound scheme, was developed to solve optimally medium-size problems. Hall et al. (2005) studied the same problem but with lot streaming, i.e., problem $O2|nwt,split|C_{\max}$. The authors developed a dynamic programming algorithm, in which the sublot sizes are consistent, i.e., they remain the same over all machines. Soleimani and Majid (2012) included machine maintenance activities into open shop problems. A genetic algorithm was used. Naderi and Zandieh (2014) investigated the no-wait open shop problem and proposed a series of models and algorithms to solve the problem $Om|nwt|C_{\max}$. A survey of the existing results for open shop problems can be found in various papers (Potts and Van Wassenhove 1992; R. Cheng, Gen, and Tsujimura 1996; Anand and Panneerselvam 2015).

3.4 Flexible production systems

3.4.1 Flexibility of a multi-stage system

Flexible shops represent multi-stage generalizations of flow shop, job shop, and open shop models, in which additional machines are used in parallel. The relationships between the different machine environments and the possible shops is shown in Fig. 3-5. It is assumed that all multi-stage systems are flexible if at least one stage has two machines in parallel. However, the notion 'flexibility' requires a clarification.

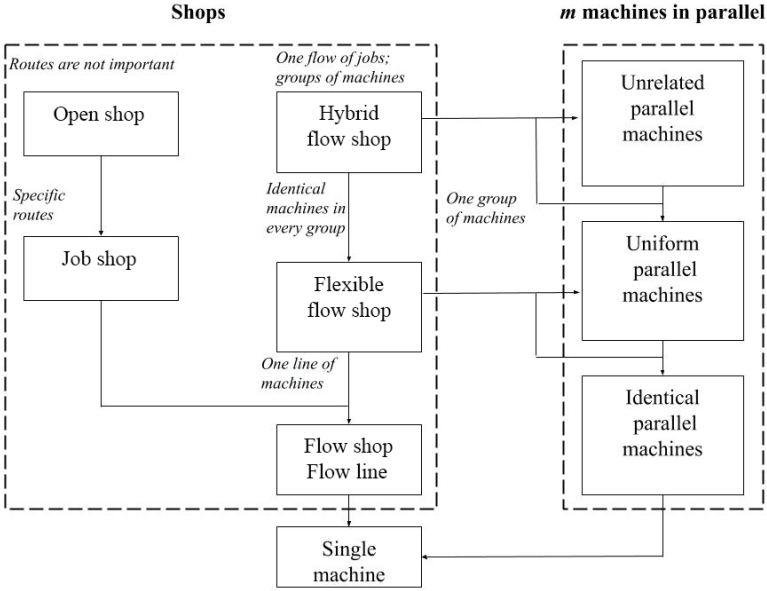


Fig. 3-5. Relationships between machine environments and shops.

A production system must possess several characteristics to be classified as a flexible one. Automated manufacturing systems display flexibility in multiple and intertwined ways, which pertain to the equipment, processes, products, production volumes, etc. It is important to know what the difference between a flexible production system and a traditional one is, what exactly the concept of flexibility means, and what the definition of specific production planning models for flexible production systems

justifies. Among the more important properties, the following ones can be mentioned (Crama 1997; Vairaktarakis 2004, 5-4-5–20):

1. *Machine flexibility* is the ability of the machines to perform various types of operations without requiring a prohibitive effort in switching from one operation to another;
2. *Material handling flexibility* is the ability of the material handling system to move different parts efficiently for a proper positioning and processing through the manufacturing facility;
3. *Operation flexibility* is its ability to be realized in different ways;
4. *Processing flexibility* refers to the ability to process a job on any parallel machine at a stage, this also means that jobs may skip stages, or there is a set of product types, which can be produced by the system without major setups;
5. *Routing flexibility* is the ability of a manufacturing system to produce a product by alternate routes through the system.

A production model with a set of machines in parallel has to comply at least with one of these concepts to be classified as a flexible shop. *Hybridization* occurs when some products require special manufacturing conditions, for example, special qualities and capacities of the machines at the same stage, the assignment of some jobs to certain machines, or other special requirements.

3.4.2 FFS and HFS environments

The flexible flow shop (FFS) model is a generalization of the classic flow shop. An FFS allows multiple machines in parallel at one stage to increase the general capacity of the plant, to balance the capacities of the stages, or to eliminate or reduce the impact of bottlenecks. An FFS can also be seen as a combination of two basic environments: a flow shop and a parallel machine model. These parallel machines can be identical. However, they have normally different capacities.

An FFS is a very common model in industries, which have the same technological route for all products as the sequence of stages. This shop type has important applications in flexible manufacturing systems (FMS), such as electronics and furniture industries as well as chemical, textile, metallurgical, semiconductors, and PCB manufacturing plants. It is also a frequent environment in pharmaceutical, petroleum, food and automobile productions, steel production systems, etc. (L. Tang et al. 2002; Jinn et al.

2002; Maharaja and Sivakumar 2006; Yaurima, Burtseva, and Tchernykh 2009).

The key decisions in an FFS problem are:

1. The allocation of the jobs to the parallel machines at each stage;
2. Sequencing the jobs at all stages of the shop.

An example of an FFS with Graham's triplet is given in Fig. 3-6.

A generalization of an FFS is a hybrid flow shop (HFS), in which at least one stage has uniform or unrelated machines in parallel, or an FFS was hybridized with a special condition, as it was noted in the previous section.

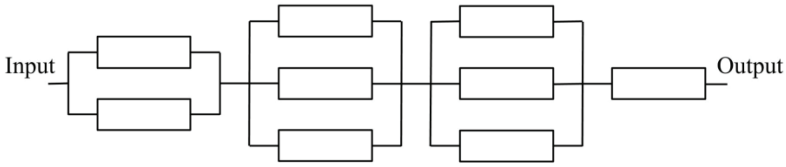


Fig. 3-6. A graphical illustration of a problem $FF4|P2^{(1)}, P3^{(2,3)},1^{(4)}|C_{max}$, where the machine environment is a four-stage FFS with two identical parallel machines at the first stage, three identical machines at the second and third stages, and one machine at the fourth stage.

3.4.3 FFS vs. HFS

In the literature, the exact difference between an FFS and an HFS has not been described. Nevertheless, it exists. Uetake et al. (1995, 395) defined the HFS as a system, which is characterized by a mixture of different types of production processes and the material flow in one direction through these processes. In such a manufacturing environment, inappropriate lot sizing and sequencing might result in the requirements for a high setup frequency and the production of excessive WIP. It implies the necessity to design procedures for lot sizing and sequencing in order to improve the manufacturing performance metrics such as the makespan and the WIP level.

Although the HFS has been studied since the 1970s, the researchers still put attention to this model. Some new designs were proposed during recent years. This fact probably implicates mistakes and confusions in the terminology and notations. A variety of known models may be interpreted as an HFS or its special case. Nevertheless, one should take into account the following considerations:

Some authors (Pinedo 2008, 15; Jungwattanakit et al. 2009, 359) do not use the notion of an HFS and describe the more complex configurations as an FFS with non-identical parallel machines at least at one stage. A variety of authors do not differ between the terms FFS and HFS referring to these models as a *flexible (hybrid) flow shop* (Allahverdi et al. 2008, 988); or they use the HFS term for an identical parallel machine environment (Naderi et al. 2009; Xie and Wang 2005; Luo et al. 2009).

A *flexible flow line* (FFL) (Kochhar and Morris 1987, 299; Mahdieh, Bijari, and Clark 2011, 107) and a *flow shop with multiple processors* (FSMP or MPFS) (H.-T. Lin and Liao 2003, 133) are usually equivalent to an FFS. Zandieh, Ghomi, and Husseini (2006, 112) noted that an HFS model is known commonly as an FFL because the flow of jobs in such a system is unidirectional.

A *hybrid flexible flow shop* or *flexible hybrid flow line* (HFFL) is equivalent to an HFS, where some jobs might skip certain stages (*processing flexibility* across the production stages) (Ruiz and Vázquez-Rodríguez 2010, 1; Allahverdi et al. 2008, 1007).

A parallel flow shop (PFS) environment is composed of a number of parallel FSs (Varela et al. 2017). A *parallel hybrid flow shop* (PHFS) represents an HFS decomposed into smaller HFS sub-designs operating in parallel. More specifically, a PHFS is composed of a number of independent sub-designs, each of which is an HFS with a unidirectional routing (*routing flexibility*) (Vairaktarakis 2004, 5-21-5–30).

3.4.4 Modeling FFS and HFS

An FFS restricted to two processing stages, even in the simplest case when one stage contains two identical machines and the second one contains only a single machine, is already NP-hard (J. N. D. Gupta 1988). The total number of possible solutions for an HFS is equal to $n!(\Pi_{i=1}^{k} m_i)^n$, while the number of possible solutions in a regular flow shop scheduling problem is $n!$ This is the reason for the intensive studies of this model.

Hoogeveen, Lenstra, and Veltman (1996) showed that preemptive scheduling in a two-stage FFS with at least two identical parallel machines at one of the stages and the goal of minimizing the makespan, i.e., either $FF2(P2,1)|\text{prmp}|C_{\max}$ or $FF2(1, P2)|\text{prmp}|C_{\max}$, is NP-hard. Mohammadi and Jafari (2011) investigated the problem of integrating lot sizing, loading, and scheduling in capacitated FFSs with sequence-dependent setups. The machines are identical at every stage. Nevertheless, the problems considering lot processing have a high computational complexity. A mathematical model and a mixed integer programming (MIP) algorithm

were proposed. Luo et al. (2009) studied a C_{max} minimization problem for a metalworking company. The process was characterized by a two-stage hybrid batching FS with multiple identical parallel machines at the first stage, while the second stage had only one machine. Setup times, which depend on the preceding job, blocking, and machine availability constraints were considered. Machines were not continuously available due to preventive maintenance and machine breakdowns. A genetic algorithm was proposed. The paper by Bang and Kim (2011) is focused on a semiconductor wafer probing facility, in which wafer lots with distinct ready times are processed on a series of workstations, each with identical parallel machines, i.e., it is an FFS. A heuristic algorithm for the problem was developed with the objective of minimizing total tardiness of the orders. The algorithm employs a bottleneck-focused scheduling method, in which a schedule at the bottleneck workstation is developed first. Then schedules for the other workstations are generated using the schedule at the bottleneck station. The prospective tardiness of the lots was considered as well as sequence-dependent setup times between different types of wafer lots. A rolling horizon method was presented for the implementation of the scheduling method for a dynamic situation. The suggested methods were evaluated by a series of computational experiments. The results showed that the proposed methods worked better than existing heuristic methods.

In the paper by X. Wang and Tang (2009), an HFS problem with identical parallel machines at each stage. Finite intermediate buffers was converted into one without WIP buffers but with blocking. A tabu search heuristic was proposed. A comprehensive mathematical model for the HFS lot streaming problem with unrelated parallel machines at each stage was given by Defersha (2011).

Lot splitting procedures for a makespan reduction in an HFS with batch production were proposed (Azzi et al. 2012; Nejati et al. 2014; M. Cheng, Sarin, and Singh 2016; B. Zhang et al. 2017). The lot streaming problem was also considered (Lalitha, Mohan, and Pillai 2017). The HFS consisted of one machine at each of the first (k-1) stages and m machines at stage k. A mixed integer linear programming (MILP) model was proposed and solved using the LINGO solver. An HFS problem with sequence-dependent family setup times and uncertain due dates was studied by Ebrahimi, Fatemi Ghomi, and Karimi (2014) with the minimization of the makespan and total tardiness criteria. A non-dominated sorting genetic algorithm (NSGAII) and a multi-objective genetic algorithm (MOGA) were proposed for a heuristic solution. In the paper by J. T. Lin and Chen (2015), a problem in a semiconductor back-end assembly facility was modeled as an HFS to perform a simulation. The objective was to achieve a feasible minimal flow

time by determining an optimal assignment of the production line and machine type at each stage for each order. A simulation optimization approach was adopted due to the complex and stochastic nature of the problem. The approach included a simulation model for a performance evaluation, an optimization strategy with the application of a genetic algorithm, and an acceleration technique via an optimal computing budget allocation. An analysis of the different levels of demand, product mix, and lot sizing were performed to reveal the advantage of simulation. An HFS batching and scheduling problem with sequence-dependent setup times and a bi-criteria objective were considered in the paper by Shahvari and Logendran (2016). The objective was to minimize simultaneously the weighted sum of the total weighted completion time and total weighted tardiness. Two algorithms incorporated a tabu search into the framework of path relinking to exploit the information on good solutions. Then these solutions were compared with a population-based algorithm. A method was implemented to find an initial solution and to trigger the search into the solution space. In order to reflect the real industrial requirements, there were considered dynamic machine availability times, dynamic job release times, machine eligibility, and machine capability for processing the jobs, desired lower bounds on the batch sizes, and job skipping. Varela et al. (2017) proposed two alternative configurations in a two-stage product-oriented manufacturing system, exploring the HFS and the PFS environments.

Lin and Liao (2003) considered a two-stage HFS with the characteristics of sequence-dependent setup times at stage 1, dedicated machines at stage 2, and two due dates taken from a label sticker manufacturing company (Fig. 3-7). Stage 1 consists of a single high-speed machine (called *calender*). It glues the surface material and liner together. Stage 2 has two types of cutting machines, each one consisting of two identical machines. The machines of one type slit the label stickers into specified width and winds on the rolls, and the machines of another type cut the label stickers into sheets, which are conform to the required size (i.e., unit length and width), referring to the two types of cutting machines CM1 and CM2, respectively. Depending on the requirement of the customer orders, each job is processed on either a CM1 or a CM2 machine at stage 2. When the calendar is changed over from jobs of one class to jobs of a new class, a setup time is required for the changeover task. This setup time depends on both the previous and the current classes of jobs. At stage 2, the setup time is sequence-independent and relatively short and hence, it is included into the processing time. In addition, the transfer time is negligible because the distance between the machines at the two stages is short. The objective is to schedule a one-day mix of label stickers across the shop to minimize weighted maximal

tardiness. A heuristic was proposed to find a near-optimal schedule for this problem.

Surveys on FFS and HFS scheduling problems as well as solution algorithms can be found in the literature (Linn and Zhang 1999; H. Wang 2005; Ruiz and Vázquez-Rodríguez 2010). A comparison of scheduling algorithms for FFS problems with unrelated parallel machines, setup times, and dual criteria was given by Jungwattanakit et al. (2009). Ribas, Leisten, and Framiñan (2010) presented a systematic review and a classification of HFS scheduling problems from the perspectives of a production system, and a solution procedure. Although the FFS problem has attracted the interest of the scientific community, as it is reflected in the contributions of the literature that cover this particular problem, the problem of lot processing in this type of shops has not yet gained enough attention.

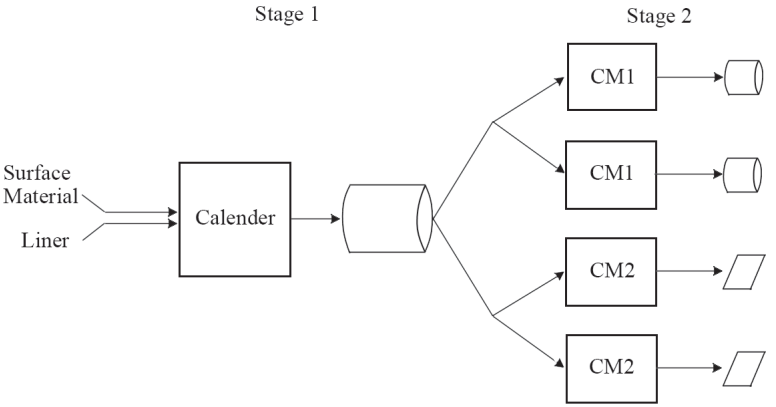


Fig. 3-7. Workflow of the label sticker production system with dedicated machines. Adapted from Lin and Liao (2003, 135).

3.4.5 FJS and FOS environments

It was evident that job shop and open shop environments, which are two basic scheduling models, will be generalized to their flexible versions, in which machine capacities are balanced. The flexible job shop (FJS) scheduling problem is an extension of the classical job shop scheduling problem by allowing an operation to be assigned to one of a set of eligible machines during the scheduling process. An analogous definition can be given for the flexible open shop (FOS) scheduling problem.

Similar to the classical job shop variant, an FJS scheduling problem includes two major sub-problems as follows (Yousefi Yegane, Khanlarzade, and Nakhai Kamalabadi 2017, 261):

1. The allocation sub-problem that deals with dedicating any operation to a machine among the possible machines;
2. The scheduling sub-problem, in which the sequence of the operations on the machines is determined.

There are a few works dedicated to these subjects. Some considered problems and developed approaches are focused on the following models.

Defersha and Chen (2012) addressed lot streaming in an FJS consisting of *m* machines, where the machines with common functionalities were grouped into a department. The problem was to determine the size of each sublot, to assign the operation of each sublot to one of the eligible machines and to determine the sequence and starting times of the assigned operations on each machine. The objective was to minimize the makespan of the schedule. An island-model parallel genetic algorithm was proposed. Gen, Zhang, and Lin (2015) studied some practical manufacturing scheduling problems in hard disc device (HDD) and thin-film transistor-liquid crystal display (TFT- LCD) industries. The authors discussed how to design hybrid genetic algorithms and multi-objective hybrid genetic algorithms (Mo-HGA) to solve the practical multi-objective manufacturing scheduling problems expanded by a multi-objective FJS model, operation sequencing and assignment of the resources. The production models were considered in detail. Bożek and Werner (2018) dealt with FJS scheduling including lot streaming and sublot size optimization. A set of operations is associated with each job, and a non-empty set of machines is assigned to each operation. Each set includes the machines that can process the sublots of this operation. Transportation times between the machines and machine setup times are considered. A two-stage optimization procedure was proposed. First, the makespan was minimized with the smallest sublots defined for the problem instance; this permitted to transfer each sublot separately to the next operation of a job. In the second stage, the sizes of the sublots were maximized without increasing the obtained makespan value. Two objectives were defined for the second stage: 1) the maximization of the sum of the sublot sizes of all operations, 2) the maximization of the number of operations, which do not need to be split at all. MILP, constraint programming and graph-based models were implemented for the solution of the problem. In the paper by Yousefi Yegane, Khanlarzade, and Nakhai Kamalabadi (2017, 261), sublots with overlapping operations were

introduced. It was a new assumption that plays an essential role in the reduction of the completion time for the lot streaming problem in an FJS environment. The considered problems were solved by means of a memetic algorithm with both permitted and not permitted lot streaming. Then the obtained solution was improved by using the critical path heuristic. Mönch and Drießel (2005) considered complex job shops that can be decomposed into a set of work areas. Each work area consists of several groups of parallel machines, which are located at the same area of the manufacturing system. In the wafer fabrication plants, the photolithography area, the diffusion area, and the etching area may serve as examples for similar work areas. The authors denoted this type of job shops as *decomposable FJSs*.

The first work, which addressed an FOS, was given by Lawler, Luby, and Vazirani (1982). The considered environment was referred to as an *open shop with parallel machines*. The authors summarized that open shops with single-operation machines of equal speed are scheduled with essentially no more difficulty than an ordinary open shop. Open shops with multiple-operation machines of equal speed are scheduled with the aid of a sequence of network flow computations. This generalized open shop problem with parallel machines of arbitrary speeds can be solved by a linear programming algorithm, mainly in the same way as an optimal preemptive schedule can be found for unrelated parallel machines.

Although the generalized open shop problem has a long history, there are still a few related publications, in which there are more than two operations and lot processing is considered. A recent paper by Bai, Zhang, and Zhang (2016) should be mentioned here. The authors presented a short but in-depth review on FOS scheduling and then studied both static and dynamic versions of the multi-stage FOS scheduling problem, where each stage has a set of identical machines. The goal was to minimize the makespan. The asymptotic optimality of the general dense scheduling (GDS) algorithm was proven by the boundedness hypothesis. For large-scale problems, GDS-based heuristic algorithms were presented to accelerate the convergence. For moderate-size problems, a differential evolution algorithm was employed to obtain high-quality solutions. To the best of our knowledge, a short state-of-the-art review on FOS scheduling, which was also presented in this paper, is the most recent and in-depth analysis of the FOS problem.

3.5 Mixed shop scheduling

In the scheduling theory, any multi-stage environment is described using a set of jobs to be sequentially allocated to a set of machines and a job processing discipline. The obtained configurations fall basically into three groups, called *shops*: flow shop, job shop, and open shop. Recall that in a flow shop, all jobs maintain the same machine order following the same route through the machines. In a job shop, the machine order is given for each job and different jobs may have different routes. In an open shop, each job is processed on each machine exactly once and the order of the operations is not fixed. The production and machine environments in the production companies have complex interconnected structures, which are difficult to design as 'pure' shops, but it is more realistic to use a combination of more simple shops. Recently, a *mixed shop* (MS), which is a further step in the development of scheduling theory, was introduced. In this environment, the routes are given for some jobs as in a flow shop or a job shop and the routes of the others are as in an open shop. Therefore, a mixed shop is a combination of three problem: a flow shop, a job shop, and an open shop. The mixed shop is an NP-hard problem. Such types of a shop are typical in semiconductor manufacturing industries.

An example of such a mixed shop is given below. Table 3-6 contains the input data for 3 jobs to be processed in a mixed shop. For every job, the data are given as pairs of numbers (m, p), where m denotes the number of machine and p denotes the processing time units. In Fig- 3-8, the Gantt chart represents a schedule with $C_{\max} = 9$. It is evident that it is an optimal schedule.

Job (Shop)	Operation		
	1	2	3
J1 (JS)	(1,3)	(2,2)	(3,4)
J2 (JS)	(2,2)	(3,3)	(1,3)
J3 (OS)	Any order		
	(3,2)	(1,2)	(2,3)

Table 3-6. An MS combined of a JS and an OS.

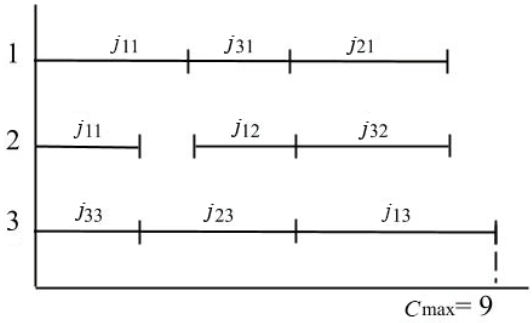


Fig- 3-8. The Gantt chart of an optimal schedule with $C_{max} = 9$.

The MS model was introduced first by Masuda, Ishii, and Nishida (1985). The authors considered a two-machine shop scheduling problem with two disjoint job subsets *F* and *O*, where *F* is the set of flow shop-type jobs, while *O* is the set of open shop-type jobs. The objective was to minimize C_{max}. This model has interesting special cases. For the case when *O* is empty, this problem is reduced to an F2 problem, which can be solved by Johnson's optimal algorithm. In the case when *F* is empty, the problem is reduced to an *O*2 scheduling problem, for which there exists an optimal algorithm developed by Gonzalez and Sahni (1976, 666–67), while for the case, in which both *F* and *O* are not empty, the situation is complicated. However, the authors gave an optimal heuristic algorithm for this nontrivial case. Strusevich (1999) introduced a problem, which was denoted as *Two-machine super-shop scheduling problem*. In this problem, the flow shop, the job shop, the open shop, and the mixed shop scheduling problems were the special cases. A polynomial time algorithm to find both preemptive and non-preemptive optimal schedules was proposed. Nguyen and Bao (2016) dealt with a mixed shop, which combines a job shop with an open shop processing order. The general case with a set of *m* machines was considered with the goal of minimizing C_{max}. A genetic algorithm was proposed to determine a solution.

The only survey was given by Shakhlevich, Sotskov, and Werner (2000). The authors discussed previous results on the computational complexity of mixed shop scheduling problems. The main attention was devoted to establish the boundary between polynomially solvable and NP-hard problems.

3.6 Assembly production systems

Since the 1980s, it has become clear that real manufacturing systems do not conform to the classical canons of scheduling theory. The machinery and production flows in advanced companies with mass production and frequent changes of the nomenclature have complex structures and connections, while the cost of the planning decisions is extremely high. As a solution, there were introduced special types of production systems, which represent hybrid shop structures that combine well-known environments, such as a job shop and an open shop with an assembly flow line. The obtained structures were defined as *assembly flow shop* (AFS) and *assembly open shop* (AOS), respectively. In such systems, the fabrication areas are arranged as a job shop or an open shop, which produce the component parts, and the fabricated parts are fed into an assembly line arranged as a flow shop for the final assembly operations. This type of production order is common among the manufacturing companies with multi-plant facilities with various fabrication plants that are connected to a final assembly plant, which is the final one in the production chain. Examples of such systems are typical for automotive, semiconductor, and electronic devices industries, to mention a few. Some considered models and problems are referred to below.

An *assembly job shop* (AJS) combines a job shop, in which the fabrication of component parts is realized according to the rule: every job has a predetermined route over a set of machines, with a flow shop to assemble these parts. A generic illustration is given in Fig. 3-9.

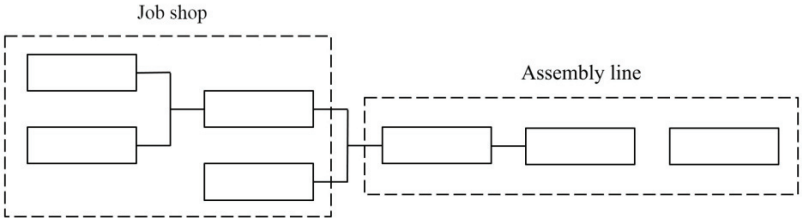


Fig. 3-9. A generic AJS.

The first study found was due to C.-Y. Lee, Cheng, and Lin (1993). The authors defined the studied system as a three-machine assembly-type flow shop (3MAF) (Fig. 3-10 a). In the considered problem, two parallel machines M_a and M_b prepare components of every job and when being ready, they are assembled on machine M_2. The authors applied a modified

Johnson's rule to find the optimal value of C_{max} and proved that the general version of this problem is NP-hard.

Potts et al. (1995) considered a generalization of the same environment and referred to this problem as a *two-stage assembly problem*, in which there are n jobs to be processed. At the first stage, each machine M_i, $i = 1,\ldots, m$, $m \geq 2$, processes a component of a job (Fig. 3-10 b). These machines work independently of each other. At the second stage, the assembly machine M_a assemblies the m prepared components of each job. The objective is to complete all jobs as soon as possible. First, the parts are produced independently, frequently in different company plants, and then a final product is assembled and packed at an assembly line or a workstation. Potts et al. proved that this problem is NP-hard even if $m = 2$. The authors modeled this problem as a generalization of a flow shop. They also showed that for the two-stage assembly scheduling problem, the search for an optimal solution might be restricted to the class of permutation schedules. The special case of $Am||C_{max}$ with $m = 2$ represents an $F2||C_{max}$ problem, which is optimally solved by the Johnson algorithm (Johnson 1954, 64–65).

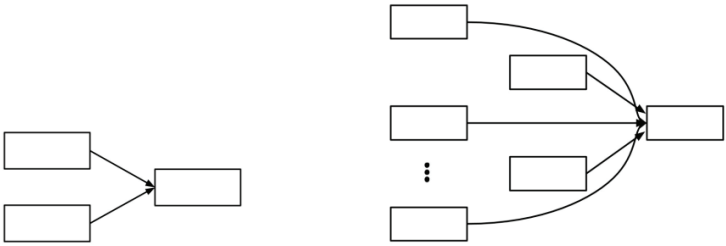


Fig. 3-10. Two-stage assembly problem: a) Model of Lee, Cheng, and Lin (1993, 618); b) Model of Potts et al. (1995, 348–49).

Lin, Cheng, and Chou (2007, 146) generalized the model of Lee, Cheng, and Lin (1993). Machines one and two are arranged in parallel for producing component parts individually, and machine three is an assembly line arranged as the second stage of a flow shop for processing the component parts in batches. Whenever a batch is formed on the second-stage machine, a constant setup time is required. The authors proved the NP-hardness of the C_{max} problem for the case when all jobs have the same processing time at the second-stage machine, and they solved optimally a special case. Wong and Ngan (2013, 1393) applied a lot streaming technique to an AJS

and examined the impact of lot streaming on the minimization of the makespan.

In recent years, the model given by Potts et al. (1995) is usually referred to as a two-stage assembly flow shop (AFS or TSAFS). Wang, Wang, and Cheng (2016, 1–3) introduced a transportation operation into the AFS scheduling problem, in which the product components are first produced and assembled, and then the complete final products are delivered to a customer in batches. The considered problem was NP-hard. Therefore, two fast heuristics were proposed, namely, an SPT-based heuristic and an LPT-based heuristic. A new hybrid metaheuristic (HGA-OVNS) was presented in order to minimize the weighted sum of the average arrival time at the customer and total delivery cost. This metaheuristic was obtained by the hybridization of an evolutionary genetic algorithm with variable neighborhood search (VNS), which is an effective local search procedure to generate the offspring individuals and an opposition-based learning (OBL) to establish some novel opposite neighborhood structures.

Framinan and Perez-Gonzalez (2017) looked for the minimization of total completion time in an AFS. The authors first analyzed the existing constructive heuristics for the problem. On the base of these ideas, they developed a constructive heuristic that outperforms the existing constructive heuristics for the problem, providing solutions almost in real-time. A variable local search algorithm was also proposed for the cases in which extremely high-quality solutions are required. Jung, Woo, and Kim (2017) considered the TSAFS scheduling problem for processing products with dynamic component-sizes, where a machine is able to process different components and therefore, a setup time is required. To solve the problem, an MLP model was derived and three genetic algorithms were proposed due to the intractability of finding an optimal solution for large-sized problems. Komaki et al. (2017) incorporated the transportation stage, realized by an automated guided vehicle (AGV), between m parallel machines and the one-machine assembly stage. The obtained model is a three-stage AFS in the $AF3(m,1,1)||C_{\max}$ problem. An improved discrete version of a cuckoo optimization algorithm (COA), which is a bio-inspired metaheuristic, was proposed. Some new adjustments such as clustering, egg laying, and immigration of the cuckoos, based on a discrete representation scheme were incorporated. In addition, for this problem, a lower bound and some dispatching rules were proposed.

The earliest review considering AJS systems can be found in the paper by Price, Gravel, and Nsakanda (1994). It was dedicated to optimization models of kanban-based production systems covering serial production lines, bottleneck workstations, and AJS production. A recent review on the

two-stage assembly scheduling problem with minimizing total tardiness and setup times was given by Allahverdi, Aydilek, and Aydilek (2016, 77–78).

3.7 Reentrant environments

Reentrant production systems, where a job cyclically visits the same machines or stages, are frequent in modern industries. So, in semiconductor manufacturing process, each wafer re-visits the same machines for multiple processing steps to produce a needed layer on each circuit (Sobeyko and Mönch 2015, 709–10). Another reason is the reparation of parts after testing. Indeed, the raw material used is of high cost due to the presence of precious metals. Some parts after testing are considered and dispatched for rework at several operations for a possible recovering. Examples of such manufacturing systems can also be met in thin film lines and LCD panels.

Chen, Pan, and Lin (2008) defined an RFS as follows. There are n jobs to be processed on m machines in the shop, and every job must be processed on the machines in the order M_1, M_2, ..., M_m, M_1, M_2, ..., M_m, ..., and M_1, M_2,..., M_m.

The production schedule of such a reentrant system is extremely difficult to optimize. Specifically, in recent years, the production paradigm changed from HV/HM to Low Volume/High Mix (LV/HM). This change implicates difficulties in the machine use because product mixes have become increased, while production lots have become much smaller. The reentrant problem has a higher computational complexity than traditional job scheduling because it has a large scope such as deadlock avoidance, capacity distribution, inventory control, and system design, apart from traditional job scheduling. Kaihara, Kurose, and Fujii (2012, 467–69) proposed an optimized scheduling methodology for such FFSs. Fig. 3-11 shows an example of a reentrant FFS.

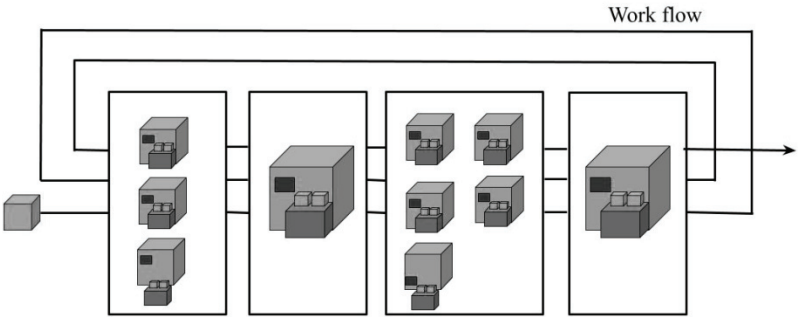


Fig. 3-11. An example of a reentrant FFS. Adapted from Kaihara, Kurose, and Fujii (2012, 468).

Danping and Lee (2011, 2223) classified reentrant manufacturing problems into two big sub-categories:

- General scheduling problems, which are generally applicable, not only in any particular field;
- Shop floor scheduling problems, which arise in the material handling in an industrial workshop.

The last ones were then subdivided into reentrant job shop scheduling problems (RJSP) and reentrant flow shop scheduling problems (RFSP). Later the reentrant FFS problem (RFFSP) was introduced (Kaihara, Kurose, and Fujii 2012, 467–68).

Danping and Lee (2011, 2223–29) classified reentrant scheduling problems into three types on the base of the objective function:

- *Type* 1 is the general problem version, which considers the timing aspect such as minimizing the makespan, the (weighted) flow time or the buffer time.
- *Type* 2 tries to increase the facility utilization, so the performance metrics might be the mean throughput, the WIP inventory, the machine idle rate.
- *Type* 3 is closer to the service level and is shown in terms of minimizing the (weighted) tardiness or minimizing the queue length of each buffer.

Among all these objectives, the majority of the papers consider a single objective or two objectives, while some have even multiple objectives.

The solution methods can be divided into: mathematical methods, heuristics, and metaheuristics. In the mathematical methods, there were found a mean value analysis, Petri nets, branch-and-bound, integer programming, etc., see the paper by Sobeyko and Mönch (2015). The heuristics include dispatching rules, constructive and improvement heuristics.

The following dispatching or scheduling policies have been suggested by various authors (Narahari and Khan 1996, 38–39; H. Zhang, Jiang, and Guo 2009; Danping and Lee 2011, 2232–37):

- Earliest due date (EDD) first;
- Least slack (LS) first;
- First come, first served (FCFS)/ FIFO;
- First buffer, first served (FBFS)/Last buffer, first served (LBFS);
- SPT/ WSPT/Truncated shortest processing time (TSPT);
- LPT/Weighted longest processing time first (WLPT);
- Critical ratio (CR);
- Least setup cost (LSC);
- Uniform, fixed WIP;
- Workload regulating;
- Dynamic bottleneck dispatching (DBD);
- Starvation avoidance.

Fixed WIP is a popular policy. This means that a new job is released into the system only when a finished job emerges from the system and thus, the number of jobs in the system (WIP) is always fixed. When such a policy is used, the stability of the system is guaranteed. Chen, Gärling, and Kitamura (2004, 1) first classified workstations into *dynamic bottlenecks* and *non-dynamic bottlenecks* and applied different dispatching rules for different workstations. Mönch and Drießel (2005) considered *distributed versions* of a modified *shifting bottleneck* heuristic for complex job shops appearing in the manufacturing of integrated circuits (IC) on silicon wafers. The considered job shop environment contains parallel batching machines, machines with sequence-dependent setup times, and reentrant process flows. The facilities for the semiconductor wafer fabrication are typical examples of manufacturing systems with these characteristics. The used performance measure is Total Weighted Tardiness (TWT). A two-layer hierarchical approach was suggested in order to decompose the overall scheduling problem. Zhang, Jiang, and Guo (2009, 114) designed a Dynamic Bottleneck Dispatching (DBD) policy, which integrates simulation and the Response Surface Methodology (RSM) to provide a very quick and rather good solution. The paper by Che et al. (2012) dealt with a no-wait

reentrant flow shop with multiple robots for material handling, which are widely used in the electronic industry, such as PCB and semiconductor manufacturing. This paper addressed cyclic scheduling due to large lot sizes and the simplicity of implementation. A polynomial algorithm was proposed to find the minimum number of robots for all feasible cycle times.

An online scheduling approach for two reentrant FMSs with due dates and sequence-dependent setup times was proposed by Pinxten et al. (2017). Online scheduling of operations is essential to optimize the productivity of an FMS, where the manufacturing process requests the arrival by a flight. An FMS processes the parts according to a particular flow across the processing stations, where products pass twice and there is a limited buffer between the processing stations. The authors applied a metaheuristic that simultaneously explores several alternatives considering the trade-offs between the used metrics. The metaheuristics used were a genetic algorithm, tabu search, simulated annealing, stochastic diffusion search, and Lagrangian decomposition. The resulting algorithm was much faster than those presented in the state-of-the-art and produced good results with respect to the makespan value.

Chen, Pan, and Lin (2008, 571–74) used a genetic algorithm to move from a local optimum to a near optimal solution for RFS scheduling problems. Kaihara, Kurose, and Fijii (2012, 467–69) applied the well-known cooperative scheduling method, Lagrangian decomposition, and the coordination method, which was initially developed for a parallel machine scheduling problem, within an actual large-scale model of a reentrant FFS in semiconductor manufacturing. This method showed a good performance. Bard et al. (2013, 7967–69) proposed a multi-stage approach, first applying a reactive greedy randomized search procedure (GRASP) to develop a schedule for the current lots waiting for processing and then to schedule additional passes and changeovers using a similar procedure. Chamnanlor et al. (2013, 309–13) used a hybrid genetic algorithm for solving an RFS scheduling with time windows. Gen, Zhang, and Lin (2015, 32–36) proposed a multi-objective hybrid genetic algorithm for production scheduling, where a reentrant flow was considered.

A systematic review of the research methodologies for the reentrant scheduling problem was given by Danping and Lee (2011). A summary of the different problems reviewed was also presented. Allahverdi (2015) referred to some works dealing with reentrant environments in the third comprehensive survey on scheduling problems with setup times/costs. For a complete list of the related models and references, one can refer to Chen et al. (2015, 1198–1201) (in Chinese).

3.8 Variable processing times

Some new concepts were introduced in the literature within the recent two decades in order to make the scheduling models closer to real-life problems. New approaches appeared also to model the main input variable, namely the processing time.

3.8.1 Scheduling jobs with deteriorating and learning effects

In classical scheduling, it is assumed that the job processing times are fixed and known for the entire process. However, empirical studies show that in many production environments, the job processing time depends on different factors such as the position in the sequence, i.e., a job, which is processed later, consumes a distinct time than the same job when being processed earlier. For example, the temperature of an ingot, while waiting to enter into a rolling machine, drops below a certain level, requiring the ingot to be reheated before rolling. The time and effort required to control a fire increase if there is a delay in the start of the fire fighting. It is similar to a common situation when a more extensive medical treatment might be necessary as a patient's health condition worsens. Scheduling models with such settings are known as scheduling *deteriorating jobs*. Many similar models with *time-dependent processing times* have been developed from a variety of such perspectives.

Gupta and Gupta (1988, 387) introduced a scheduling model, in which the processing time of a task was a polynomial function of its starting time. In their paper, single facility scheduling is considered. The job processing time depends on its starting (or waiting) time in the sequence. Later, Browne and Yechiali (1990, 495–96) presented a single processor stochastic scheduling model, where the jobs deteriorate if they have to wait for a service, causing a growth of their processing times during waiting (at job-specific rates). The authors considered a class of non-preemptive processing strategies and found policies to minimize the variance of the expected makespan for different deterioration schemes and to minimize the variance of the makespan. The authors indicated the control of some queuing and communication systems as areas, where such a model is used.

There is another reason why the processing times can vary, even if the jobs are processed under the same or almost the same conditions. For example, a worker has to assemble a large number of similar products. The processing time to assemble one product depends for the worker on his knowledge, skills, organization of the working place, and other aspects. With a repetition of the same activities, the worker learns how to make his

job more efficiently. Consequently, he becomes better skilled, his knowledge is increased, and the working place is better organized. As a result, the job processing time decreases for the worker to realize his activities. This phenomenon is known in the literature as *learning effect*. The learning effect was first noted in the aircraft industry by Wright (1936, 124). Then the concepts of a time-dependent processing time and a *deteriorating/learning effect* were introduced at the beginning for the single machine problem, see the paper by Wu, Shiau, and Lee, (2008). Only a few exceptions deal with parallel machines (Ji and Cheng 2008) and two-machine flow shop models (L. Tang and Liu 2009). The objective was usually to minimize the traditional performance measures such as the makespan, flow time, maximum lateness, and the number of tardy tasks. The time-dependent function used to model the processing time of a task was typically a linear or piecewise linear function of the starting time of the job in a schedule.

Biskup (1999, 173–74) and Cheng and Wang (2000, 274) introduced the *learning effect* into scheduling problems. Learning effect means that the skills of the workers are continuously improving by processing one job after the other, for example, the ability to perform setups faster, to deal with the operations of the machines and software or to handle raw materials, components or similar operations of the jobs at a greater pace. Biskup assumed that the processing time of a job is a log-linear learning curve, i.e., if a job J_j is scheduled at the position r in a sequence, its actual processing time is $p_{jr} = p_j r^a$, where p^j is the normal processing time of job J_j and $a \geq 0$ is a constant learning effect. The author proved that single machine scheduling problems with minimizing the sum of the job flow times and the total deviations of the job completion times from a common due date are polynomially solvable. Cheng and Wang approximated this effect by a volume-dependent function. The objective was to minimize maximum lateness. The authors showed that the single machine problem is NP-hard and identified two polynomially solvable special cases.

Koulamas and Kyparisis (2007, 403–4) considered a single machine scheduling problem with general learning functions. The learning phenomenon was expressed as a function of the sum of the processing times of the already processed jobs. The author showed that the SPT-sequence is an optimal sequence for the objectives of minimizing the makespan and total completion time. The authors also considered two-machine flow shop scheduling with general learning functions.

Kuo and Yang (2007, 22) considered single machine scheduling with a *job-independent (job-dependent) learning effect* and *past-sequence-dependent setup times* (p-s-d). Later, Wang (2008, 586) extended the result

of Kuo and Yang to the single machine scheduling problem with time-dependent learning effect and p-s-d setup times. He defined the *time-dependent learning effect* as the processing time of a job that is calculated by a function of the total normal processing time of the already processed jobs.

Wu, Shiau, and Lee (2008) explored the job deterioration concept in the context of GT. The deterioration phenomenon was considered in a framework used for minimizing the makespan and total completion time in single machine group scheduling problems. It was shown that these problems remain polynomially solvable when deterioration is considered.

Cheng, Ding, and Lin (2004, 2–3) proposed an adaptation of the classical scheduling notations (Graham et al. 1979, 288–90) to a formal statement for the basic linear model.

A task system consisting of n independent tasks is denoted by TS = $(\{T_i\};\{a_i\};\{b_i\};\{r_i\};\{d_i\})$. Each task T_i is associated with a normal processing time $a_i \geq 0$, processing rate b_i, release time r_i, and due date d_i. The actual processing time of T_i depends on its start time s_i, which is given by $p_i = a_i \pm b_i s_i$. The processing time functions proposed by other authors for different applications are as follows:

$p_i = a_i + b_i s_i$, which can be interpreted as extra efforts (Mosheiov 1994, 654–55);
$p_i = a_i - b_i s_i$ (K. I.-J. Ho, Leung, and Wei 1993, 315);
$p_i = b_i s_i$, $a_i = 0$ (Mosheiov 1994, 655);
$a_i = a$; $b_i = b$; $s_i = s$ (T. C. E. Cheng et al. 2003; J.-B. Wang 2007a, 2047).

Other precise adapting functions supplemented by an analysis of the solvability of the corresponding problems can be found in the literature (T. C. E. Cheng, Wu, and Lee 2008; J.-B. Wang 2007a; 2007b).

Tang and Liu (2009) considered a two-machine flow shop in which a single machine is followed by a batching machine, and there are *deteriorating jobs* to be processed on the single machine. The authors derived optimal polynomial algorithms to minimize the makespan, total completion time, and maximum lateness, respectively.

An extension, entitled *general position-dependent* and *time-dependent* learning effects, was considered in the paper by Wang and Li (2011, 1390). This means that the actual processing time of a job is not only a function of the total normal processing times of the jobs, which were already processed, but also a function of the scheduled position of a job.

Inderfurth et al. (2006, 1598) combined rework processes with *limited deterioration*. At the rework stage, some defective items belonging to the same batch must be reworked. Each reworked item has the required good quality. While waiting for rework, defective items deteriorate. A deterioration time limit is given. A defective item, which is decided not to be reworked or cannot be reworked because its waiting time will exceed the deterioration time limit, is disposed of immediately after its work operation completes. The deterioration results in an increase in time and cost for performing the rework processes. It is supposed that the percentage of defective items is the same in each batch, and they are evenly distributed in each batch. This problem is typical for the semiconductor industry. The authors proposed a polynomial dynamic programming algorithm for its solution.

Wang (2007b, 398–99), then Cheng, Wu, and Lee (2008, 974) and later Lee (2014, 137) considered *both learning and deteriorating effects* for the single machine problem. The phenomena of a learning effect and deteriorating jobs can co-exist in many real-life situations. Wang interpreted the deteriorating effect as forgetting of the recently obtained abilities by workers due to a big product variety, short production runs, and frequent product changes. A model was considered, where the job processing times were defined by functions of the starting times and positions in the sequence. The SPT and EDD rules were proposed for the solution of some problems.

Cheng, Wu, and Lee (2008, 974, 977–78) considered some scheduling problems with deteriorating jobs and learning effects in processes when, for example, a silicon-based raw material is first heated up in an oven until it became a lump of malleable dough. Then a craftsman cuts pieces from this dough and shapes them according to different designs into different glass craft products. The initial time for heating up the raw material to the threshold temperature, at which it can be shaped, is long. Therefore, the first piece has a long processing time. This time includes both the heating time (the deterioration effect) and the shaping time (the normal processing time). The second piece requires a shorter time to re-heat the dough to the threshold temperature (a smaller deterioration effect). Similarly, a later piece, which is cut from the dough, needs a shorter heating time to reach the threshold temperature. On the other hand, the pieces that are shaped later require shorter shaping times because the craftsman's productivity improves as a result of learning.

Lee (2014, 135–37) noted that scheduling with learning effect and deteriorating jobs becomes more popular and gives a detailed up-to-date revision of the related literature. In his paper, a model was proposed, where

deteriorating jobs, the learning effect, and setup times were simultaneously present. The actual job processing time was a general function of the processing times of the jobs already processed and the scheduled position of this job. Optimal schedules for some single machine problems were provided.

Pei, Liu, Pardalos, Fan, et al. (2017) studied a single machine scheduling model with deteriorating jobs, where multiple job types and sequence-dependent setup times were considered simultaneously. Later, Pei, Liu, Pardalos, Migdalas, et al. (2017) added a learning effect to the problem assumptions. Both the *learning effect and the allocated resource* determine the processing time when the total resource for all jobs is limited. Deteriorating jobs were processed on a serial batching machine in an aluminum manufacturing factory. It was found that the workers and machines could increase the processing efficiency by repeating the production operations. Then, Pei et al. (2018) considered simultaneously serial batching, a learning effect, resource-dependent processing times, and setup operations. That is, the effect of learning was also observed during the aluminum ingots processing. Finally, Pei et al. (2019) introduced serial batching scheduling problems with *position-based learning effect*, where the actual job processing time was a function of its position. Structural properties were derived for problems with minimizing the makespan, the number of tardy jobs, and maximum earliness. Various polynomially solvable cases were detected for each category.

3.8.2 Controllable processing times

In scheduling problems with *controllable processing times*, the job processing times are not given as constants as in classical scheduling but they can be controlled (i.e., compressed) by allocating additional resources, such as additional money, overtime, energy, fuel, catalysts, subcontracting, or additional manpower, to the job operations. This concept has been considered starting from the pioneering works by Vickson (1980a; 1980b). In the last thirty years, several works have employed such a class of decision variables to improve the performance of a system.

A general definition of scheduling problems with controllable job processing times can be stated as follows (Shabtay and Steiner 2007, 1644):

There are n independent jobs $J = \{1,2,..,\ n\}$ to be processed on m machines, $M \in \{M_1, M_2, .., M_m\}$, and O_{ij} is the operation of job j on machine i for $i = 1,\ldots, m$, $j = 1,\ldots, n$. The machines are arranged in a specific technological configuration, which can be a single machine ($m = 1$), machines in parallel, machines in a flow shop, job shop, or open shop. The

release time r_j of job j is the time at which the job arrives to the system and becomes ready for processing. The processing time p_{ij} of job j on machine i is a non-increasing function of the amount u_{ij} of the resource allocated to process operation O_{ij}. In the single machine case, the machine index is omitted so that, for example, p_j is the processing time of job j on the single machine. The resource may be used either in continuous or discrete quantities. In the first case, the processing time of a job is determined by the amount of a divisible resource (for example, gas and electricity) allocated to it and therefore, it can vary continuously. On the other hand, a discrete type of resource is indivisible (for example, manpower and supporting equipment). Therefore, the processing time of a job has only a finite number of possible durations.

Controllable or variable processing times and setups were jointly considered in some job scheduling models proposed in the last two decades. So, Cheng and Kovalyov (1995) considered a single machine batch scheduling problem with the objective to determine a feasible schedule (with respect to the deadlines), which minimizes the amount of the resource used to decrease the processing times.

In this class of problems, the assumption of *resource-dependent processing times* is also considered. It appears commonly when jobs are processed in batches, in situations when the job processing times can be compressed by the allocation of a continuously divisible resource. One of the pioneering works in this area was due to Cheng, Janiak, and Kovalyov (2001, 177–78). The used compressing function was:

$$p_j = b_j - e_j y, j = 1,\ldots,n,$$

where

$b_j > 0$,
$e_j > 0$,
$0 \le y \le y_{\max} \le \min \{(b_j - b_{\min})/ e_j\}$,
$b_{\min} > 0$ - a technological restriction, which defines the minimum job processing time.

Shakhlevich and Strusevich (2006) considered a single machine environment in the context of a supply chain. The authors investigated the cases when the *job processing speed/times* and/or *job release speed/dates* are controllable. The possible changes in the controllable parameters were either individual or done by controlling the relevant processing or release rate. The objective was to minimize the sum of the makespan plus the cost

for changing the parameters. A number of polynomial time algorithms was provided, and a sufficiently complete classification of complexity was given.

Mor and Mosheiov (2014) introduced a new class of models for single machine scheduling, which combined two phenomena, namely batch scheduling and controllable processing times of jobs. A linear compression cost for controlling the job processing times was assumed. Two problems were solved: 1) minimizing total flow time plus the compression cost, and 2) minimizing the flow time subject to an upper bound on the maximum batch compression. In both cases, the solution for the relaxed version consisted of a decreasing arithmetic sequence of the batch sizes, for which closed form solutions were obtained.

In the single machine family scheduling problem with *controllable processing times*, considered by Giglio (2015), it was possible to reduce the processing time of a job at the price of the payment of an extra cost. The problem was characterized by *costly* sequence-dependent setups and generalized due dates. The machine was assumed unreliable. Perturbations, such as breakdowns (or, in general, machine unavailability) and slowdowns, may affect the nominal behavior of the system. The problem was to minimize the sum of total weighted tardiness plus the total weighted consumption cost of a continuous resource plus the total setup cost. It was formulated as an optimal control problem in which a solution consisted of optimal control strategies that were functions of the system state.

Pei et al. (2018) combined simultaneously *resource-dependent processing times* with serial batching, a learning effect, and setup operations. A hybrid algorithm, which combined a gravitational search algorithm (GSA), and tabu search, was developed to solve the general case.

The most recent survey of scheduling problems with controllable processing times was presented by Shabtay and Steiner (2007). Comprehensive reviews on different models and problems concerning these subjects were given in the literature (Alidaee and Womer 1999; T. C. E. Cheng, Ding, and Lin 2004; Biskup 2008).

3.9 Conclusions

The above description of shop models is really a retrospective review on the development of the scheduling theory. One can observe a progress in modeling the problems, a diversification of the methodologies and a refinement of the terminology. Numerous new models and approaches of flexible and dynamic nature, which are now more realistic, fill the gap

between theory and practice, which is a permanent subject of discussions in the scheduling literature.

Actually, the focus of the researches shifts from the development of methodologies for usual shops to a modeling and an analysis of complex manufacturing systems, particularly to the solution of scheduling problems, which also include mixed shops, reentrance of jobs and rescheduling. Reentrance of jobs is a new hot subject in the theory. Many publications appeared recently on this subject. A new survey on reentrant models can be suggested.

There are many recent publications, which are dedicated to scheduling problems with time-dependent processing times. It is a class of machine scheduling problems, in which the processing time of a job depends on its starting time in a schedule. An up-to-date review was given by Cheng, Ding and Lin (2004). Problems with controllable processing times, in which the job processing times can be controlled (i.e., compressed) by allocating additional resources, can also be included into this class (Mor and Mosheiov 2014).

Modern manufacturing systems deal with long routes of the component parts through the facilities, which have hundreds of machines. It is very problematic to obtain a precise model of such a production process. On the other side, even if the production flow is complex, this may be described as a combination of more simple models. Solutions can be found faster and in better quality than for a complex system. The development of mixed structures, which combine known shops and approaches in the modeling process, is characteristic for the current state-of-the-art in the scheduling area. The study of short flexible variants of assembly lines, job shops, and open shops with a few machines can give good practical results.

A systematic review of the literature on basic models of the flow shop scheduling/sequencing research in the period 1952-1994 was given by Reisman, Kumar, and Motwani (1997). A survey of the scheduling research addressing position-dependent processing times can be found in the paper by Bachman and Janiak (2004). Reviews of publications dealing with scheduling problems in semiconductor manufacturing and different kinds of solution methods can be met in the literature (Geiger, Uzsoy, and Aytuğ 2006; I.-L. Wang, Wang, and Chen 2013).

CHAPTER FOUR

MACHINE SETUP TIMES

Think of the productivity improvement that could be attained if a setup operation requiring three hours could be reduced to three minutes!
(*Shingo 1985, XIX*)

In real manufacturing systems, a large number of product variants are usually processed in the same flow, and adjustment operations are required at every changeover. As it has been defined by Shingo (1985, 6), the creator of the SMED system, the *setup operation* is the preparation or post adjustment that is performed once before and once after each lot is processed. The time required to shift from one product to another one on a given machine is defined in the scheduling theory as *setup time* or additional production cost. The scheduling problems, which consider setup times, have a high computational complexity. Pinedo (2008, 597) presented a proof of the NP-hardness for the single machine case with setup consideration. Nevertheless, this subject has received a high attention of industrial planners and scientific researchers due to the high influence on the production effectivity. The setup time is one of the vital parameters used in any manufacturing industry for every machine or workstation. An explicit treatment of setup times in most applications is required and represents a special interest because it is a significant factor for production scheduling in many cases. Setup activities may take up a time ranging from some milliseconds until various hours and can easily consume more than 20% of the available machine capacity when they are not well handled. In this chapter, the structure and types of setup activities are discussed for a deep understanding of this concept as well as various theoretical and practical approaches to solve scheduling problems with setup consideration.

4.1 Basic structure of a setup time

The time that a job spends on a machine includes three principal phases: *setup*, *processing*, and *removal*. In a discrete production, typical setup/removal activities for a processing machine are machine adjustment

and feeder preparation to process the next job, dismantling after processing the previous job, machine calibration, inspection of accessories or tools, cleaning of the machines and adjacent areas, and so on. As a machine becomes free only after the setups of all jobs have been removed, the removal times are either zero or positive. Therefore, the removal times can also be included into the calculation of the makespan.

Setup activities may depend on the type of the job, the type of the machine, or both. In any case, the time between producing the last product of a series and producing the first product of a new series, which meets all quality requirements, has always been considered as *waste* or added cost (see Section 1.5).

The importance of the setup reduction is evident from a simple equation for calculating the time required for manufacturing a series of parts and assembly of the components (Kušar et al. 2010, 833):

$$p = p_s + n \cdot p_1,$$

where

- p - time required for manufacturing parts and assembling components (hs/series);
- p_s - machine setup time or the assembly workplace setup time (hs/series);
- n - number of units within a series (pieces/series);
- p_1 - manufacturing/assembly time per unit (hs/piece).

There are two ways to reduce the total downtime due to setups:

- Reduce the setup frequency;
- Reduce the time that is needed to perform the setup.

Reducing the setup frequency is less preferable compared to reducing the setup time. The main setup reduction benefits are shorter lead times, higher productivity, increased capacity, greater flexibility and fewer defects.

The period between finishing the last good product from the previous production on the machine and the first production good coming out from the following production order is defined in the literature as the *changeover time* (Gest et al. 1995, 205).

The terms *setup* and *changeover* are sometimes used interchangeably, however, this use is incorrect.

A changeover can be divided into three 'ups':

1. Clean-up – cleaning space, products, materials or machine components from the line (dust, smalt, etc.);
2. Set-up – covers the adjustment process of the equipment;
3. Run-up or Start-up - is the time spent for fine tuning the equipment after it has been restarted. It is generally caused by the variability of the clean-up and setup, as well as by the variability in the product or its components.

In a strict sense, a setup is only one component of a changeover. In some cases, the setup operations may affect the configuration of the production lines or the manufacturing cells at the plant floor. With this viewpoint, the setup operations have some of the following characteristics (Andrés et al. 2005, 276):

- There are machine precedence relationships between and within every zone. The precedence relations are the same for every product. It only changes their duration and the presence or absence of the operations;
- Setup operations may require *removing machines* from the lines (Fig. 4-1a);
- Some setup operations consist in *adding machines* to the conveyors or cells (Fig. 4-1b);
- There may occur setup operations, which consist in *changing the distribution* of the machines along the lines, without adding or removing machines. This operation can be considered as a combination of the previous two cases (Fig. 4-1c).

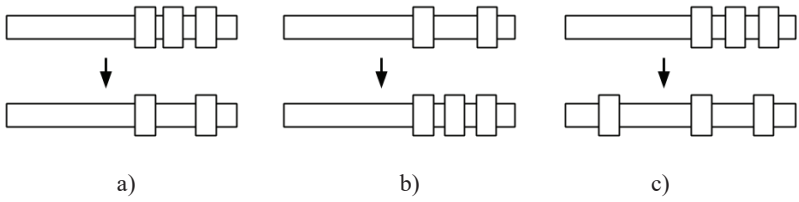


Fig. 4-1. Machine configuration change at a production line: a) Removal of one machine; b) Addition of one machine; c) Change of one machine. Adapted from Andrés et al. (2005, 276–77).

4.2 Classifications of setup times

The treatment of setup times in a scheduling problem depends on the problem assumptions. With respect to the processing time, the setup times are *separable or non-separable*. In the majority of researches dedicated to production planning and scheduling, it is assumed that setups are constant for all jobs and therefore, they are included into the processing times, or they are negligible and hence are ignored. The *non-separable* setup time assumption simplifies the analysis. These problems can be formulated and solved as standard scheduling statements.

An operation is *separable* from the processing if it is not a part of the processing operation. Whenever separable setup/removal times are not negligible in a scheduling problem, they should be explicitly considered. In some cases, non-separable setup and removal operations, which are treated as a part of the processing, require an explicit consideration, for example, loading and unloading. Such situations arise mainly in automatic production systems, which involve intermediate material handling devices such as automatic guided vehicles (AGVs) and robots. These non-separable setup operations and the processing operations must be closely coordinated (Cheng, Gupta, and Wang, 2000, 263).

Separable job/batch setup times, which are involved into the scheduling problem considerations, can be classified as follows:

- Sequence-independent (SI) setup times;
- Sequence-dependent (SD) setup times;
- Machine/resource-dependent (MD) setup times;
- Time-dependent (TD) setup times.

Separable setup times may be independent or dependent on the sequence of jobs on the machine. When the setup activities for a job on any machine in the shop do not depend on any other job in the sequence of the jobs to be processed, such a setup time is defined as *sequence-independent*. In this case, the setup duration (cost) depends solely on the current job/batch to be processed. On the contrary, when the duration (cost) of a setup depends on both the current and the immediately preceding jobs/batches on a specific machine, i.e., *dependent on the sequence* of jobs, such a setup is defined as *sequence-dependent*. An explicit consideration of sequence-dependent setup times produces a broad class of real production problems. A short review on this class of problems is given in Section 4.3.

Sequence-dependent setup times can be explained by the following example. Let us suppose that a part of the setup adjustments for a job k can

be used for processing the next job *j*. In such a case, after processing job *k*, this part of the setup will be conserved to be used by the next job *j*, i.e., it will not be removed. Consecutively, the removal time of job *k* depends on the setup activities for job *j*, and the setups for job *j* will depend on job *k*. By this reason, sequence-dependent setup times are of anticipatory and detached type as the setup information and details cannot be included into the processing time and cannot move with the job.

Sequence-independent setup times could be either anticipatory (detached) or non-anticipatory (attached). The difference consists in the treatment of the machine idle time. A setup is *anticipatory* if it can be started before the corresponding job or batch becomes available on the machine. Unless stated otherwise, the setups are assumed anticipatory, which means that a setup on a machine does not require the presence of a job (Potts and Kovalyov 2000, 230). Sequence-dependent setup times are only of anticipatory type. In such a situation, the idle time of a machine can be used to complete the setup for this job. *Anticipatory* setup times are separated from the processing times (detached). In this case, the setup time and the processing time are associated with every job. The setup information and details *do not move with the job*, if the setup times are of anticipatory type.

Otherwise, a setup is *non-anticipatory* and the setup operations can start only when the job arrives at a machine as the *setup is attached to the job*. In this case, the idle time of a machine cannot be used and hence, the setup time is considered as a part of the processing time. Consecutively, the problem is formulated and solved as a standard scheduling problem without setup consideration. If it is not stated explicitly, setups are considered as non-anticipatory. This assumption can be used for the optimization goals in specific problem statements (Khurana and Bagga 1984; Allahverdi 2000). If the setup type is not stated explicitly, it is considered sequence-independent. A graphical representation of the described setup time models is given in Fig. 4-2.

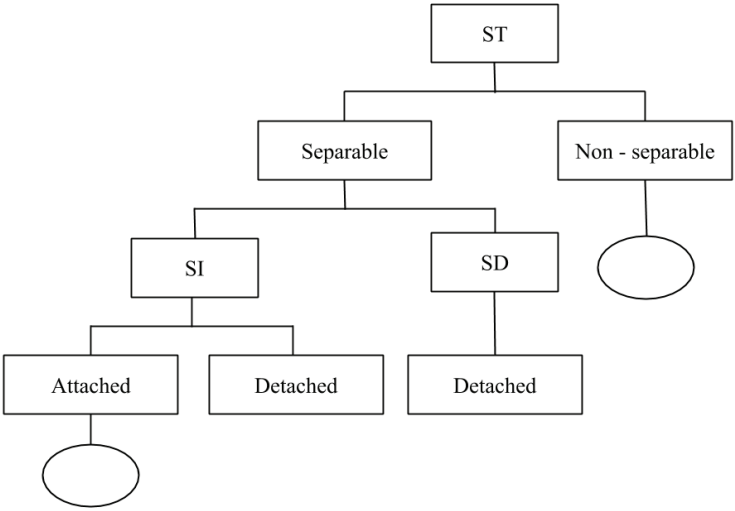


Fig. 4-2. Relationships between setup time (ST) models. A circle represents a standard scheduling model.

Summarizing the previous considerations, the breakdown structure of a job on a machine can be discomposed into various activities as follows (T. C. E. Cheng, Gupta, and Wang 2000, 263):

1. The setup activities, which are independent of the job to be processed. This operation consists of activities, such as fetching the required jigs and fixtures, then setting them up on the machine;
2. The setup activities, which are dependent on the job to be processed. This operation includes the activities required to set the job in the jigs and fixtures and to adjust the tools as required;
3. The processing of the job;
4. The removal activities, which are dependent on the job that has been processed. This operation includes activities such as disengaging the tools from the job and releasing the job from the jigs and fixtures;
5. The removal activities, which are independent of the job that has been processed. This operation includes activities such as dismounting the jigs, the fixtures and/or tools, inspecting/sharpening of the tools, and cleaning the machine and the adjacent area.

With this decomposition, the three general phases, namely, setup, processing, and removal, can be given in detail as follows: separable setup

time (operations 1 and 2); processing time (3); and separable removal time (4, 5).

In a non-batch processing environment, a setup time (cost) is incurred prior to the processing of each job. The corresponding model can also be viewed as a particular case of a batch setup time (cost) model, in which each family consists of a single-job.

In discrete production systems, the setup time is usually a *discrete variable* but for a continuous-time representation, all activities are scheduled on a continuous time scale including a *continuous variable*, which defines the duration of a setup before the next lot is handled. It is combined with setup types, such as sequence-independent/dependent setup times for a classification and modeling, see Stefansdottiret, Grunow, and Akkerman (2017).

The setup times can be represented as a quadratic non-symmetric matrix $S(k, j)$, where every element s_{kj} is the setup time required to adjust the machine from product k to j for $k, j = 1,\ldots, n$. For the diagonal elements, we have $s_{kk} = 0$, because it is referring to the same product. In the matrix, we have $s_{kj} \neq s_{jk}$ because the time, which is required for preparing the equipment, depends on the features of the previous product and on the configuration of the machine. The setup time to prepare the machine for processing the product k after j is in general distinct for the change from j to k.

The setup matrices are used to schedule the jobs and to group the products into families to minimize the setup times at each stage (see Section 4.4).

4.3 Scheduling problems with setup times

Scheduling problems, which involve separable setup time assumptions, attract the interest of the manufacturers due to their high applicability in real HV/HM planning problems and the attention of researchers due to the variability of the computational complexity. The importance of setup times has been investigated in several studies. In the following, a short state-of-the-art survey on recent publications is presented. Some theoretical and practical problems with setup time assumptions are considered for different shop models. One of the goals of this section is to show the variety of problems with setup times.

4.3.1 Sequence-independent setup times

The works, which treated explicitly sequence-independent setup times and their variations, are dedicated mainly to a single machine environment. Nevertheless, there is a variation in the setup interpretation.

Janiak and Kovalyov (1995) were among the first researchers who studied the *single machine group scheduling problem*, in which jobs of the same group have to be processed jointly, and the machine setup time is *independent* of the group sequence. The authors have investigated as further research a more general problem, where the groups are allowed to be partitioned into batches. The setup times are dependent on both the current group and the group to be processed next. This setup model is evidently a more realistic one. Pan, Chen, and Cheng (2001) assumed that the jobs were classified into classes in their problem, and a setup was required between jobs of different classes but not of the same class. The setup time was fixed and depended only on the current job. A heuristic algorithm, which generates a near optimal solution, was proposed to find an approximate schedule that minimizes maximum lateness for a set of jobs. T. C. E. Cheng, Ng, and Yuan (2003a) proved that the single machine batching problem with family setup times to minimize maximum lateness is NP-hard. The same authors proved also the NP-hardness of the single machine multi-operation job scheduling problem with *batch setup times* and arbitrary due dates when the objective is the minimization of the number of tardy jobs (T. C. E. Cheng, Ng, and Yuan 2003b). Liu and Cheng (2004) considered the single machine scheduling problem of minimizing total completion time subject to job release dates and preemption penalties. In this problem, each time a job is started, independently whether initially or after being preempted, a *job-independent setup* must take place. This situation may occur, for example, when the operating system must interrupt current tasks temporarily and later, when the interrupted tasks are resumed, some extra time must be expended. The problem was proved to be NP-hard even if the setup time is one unit.

In the paper by Mor and Mosheiov (2014), the job processing times can be controlled (i.e., compressed) by allocating additional resources, and the batch processing time is identical to the total processing time of the jobs contained in the batch plus a constant setup time, which is incurred when starting a new batch, i.e., it is sequence-*independent*. The cases of a *single machine* and *identical parallel machines* were considered with the objective of minimizing the flow time. A rounding procedure was introduced for a relaxed version and shown to generate extremely close-to-optimal integer solutions.

Pei et al. (2016) considered the scheduling of jobs on a single serial batching machine with dynamic job arrivals and multiple job types, where an independent constant setup time was associated with each new batch. A two-phase hybrid algorithm (TPHA) was developed. A valid lower bound for the problem was also derived.

Song, Yi, and Shen (2016) focused on single machine scheduling with sequence-independent setup times, multi-item production, and small lot sizes. Many setup times are difficult to estimate accurately in this statement. This uncertainty influences the ability to achieve accurate production cycles and costs of products. On the base of a survey, the authors of the study concluded that the length of the setup time depended on the level of the knowledge of the employers. They proposed a method for determining the *standard setup time quota* based on this level.

Kong et al. (2019) studied the bounded parallel-batching scheduling problem considering job rejection, deteriorating jobs and non-identical job sizes. The objective was to minimize the sum of the makespan and the total penalty. A sequence-independent setup time before the processing of each batch was included. A hybrid algorithm combining a heuristic with a dynamic programming algorithm (H-DP) was proposed to obtain satisfactory solutions within a reasonable time.

4.3.2 Sequence-dependent setup times

Models with sequence-dependent setup times are studied in all machine environments, such as a single machine (Miller et al. 1999; T. Chen, Long, and Fung 2006; Giglio 2015; Y.-C. Choi 2016; Pei, Liu, Pardalos, Migdalas, et al. 2017); parallel machines (Y. H. Lee and Pinedo 1997; S.-I. Kim, Choi, and Lee 2006); flow shops and assembly lines (H. M. Cheng and Ying 2011; Jung, Woo, and Kim 2017); job shops (Flynn 1987; I.-C. Choi and Korkmaz 1997; Mönch and Drießel 2005); FFS/HFS (C.-Y. Liu 1996; C.-Y. Liu and Chang 2000; Yaurima, Burtseva, and Tchernykh 2009). To the best of our knowledge, the concept of sequence-dependent setup times was first introduced by Barbara B. Flynn in 1987 for scheduling repetitive lots in GT and traditional shops (Flynn 1987).

Miller et al. (1999) proposed a hybrid genetic algorithm (HGA) for the *single machine*, single-stage scheduling problem in a sequence-dependent setup time environment within a fixed planning horizon. It incorporates the elitist ranking method, genetic operators, and a hill-climbing technique in each searching area. To improve the performance and efficiency, hill climbing was developed by uniting the Wagner-Whitin algorithm, see the paper by Wagner and Whitin (1958, 1773) and Section 8.1. The objective

of the HGA was to minimize the sum of setup cost, inventory cost, and backlog cost. The HGA showed a performance up to 50% better than JIT heuristics and 30% better than complete batching heuristics. The investigation was dedicated to solve a scheduling problem faced by a plant in an automotive supply chain, where heuristics for determining a job sequence do not produce the best solution, for example, a heuristic for grouping products only by the *master die*. A detailed review for this class of scheduling problems was given.

Chen, Long, and Fung (2006) considered a single machine batch scheduling problem with sequence-dependent setup times to minimize maximum lateness. The problem was solved by a genetic algorithm. In the single machine job scheduling problem discussed by Giglio (2015), equivalent jobs were grouped into classes (*family scheduling model*). The *generalized due date model* was adopted for each class of jobs. It was possible to reduce the processing time of a job, at the price of the payment of an extra cost (*controllable processing times*). A costly sequence-dependent setup was required when switching between jobs of different classes. The single machine was assumed unreliable, and then perturbations, such as breakdowns and slowdowns, may affect the nominal behavior of the system. The problem was solved by optimal control strategies.

The research by Choi (2016) was focused on the problem of scheduling jobs and energy requirements. Jobs of multiple types arrive dynamically over time. A setup operation is required to change over the job types, and it strongly depends on the sequence of the job types. During the setup operations, the machine tools consume an idle energy for non-machining on the machine tool. Moreover, frequent setups and long setup times have a negative impact on the completion of the jobs as well as the idle energy consumption. Each job type has alternative process plans with different electricity machining energy requirements. Two energy efficient dispatching rule-based algorithms were applied with the objective of minimizing the average energy consumption (with machining and non-machining) and mean tardiness of the finished jobs on a single machine with sequence-dependent setup times and energy requirements.

Pei, Liu, Pardalos, Migdalas, et al. (2017) combined deteriorating jobs, serial batches, multiple job types, and sequence-dependent setup times. In this model, the jobs of each type are first partitioned into serial batches, and then all batches of different job types are processed on a single serial batching machine. The actual job processing time is an increasing function of its starting time. A setup time is required only when a new batch is processed first on the machine or immediately after a batch belonging to another job type. Some optimization algorithms were developed to solve the

makespan minimization problem, the maximum tardiness minimization problem, the maximum lateness minimization problem, and the maximum earliness minimization problem, respectively. Two special cases of the total completion time minimization problem were also discussed.

Lee and Pinedo (1997) considered a total weighted tardiness scheduling problem with *identical parallel machines* to process a number of jobs. A job has a processing time, a weight and a due date. In order to process job k after job j, a setup time s_{jk} is required. This time depends on job j as well as on job k, but it does not depend on the machine. A dispatching rule and a simulated annealing procedure were presented. Kim, Choi, and Lee (2006) combined sequence-dependent setup and distinct ready times, i.e., the time at which a job is available for processing in parallel machine scheduling. Tabu search heuristics were suggested due to the complexity of the problem.

The problems considering sequence-dependent setup times in *flow shop environments* were presented in the papers by Cheng and Ying (2011); Jung, Woo, and Kim (2017). Cheng and Ying (2011) examined the flow line manufacturing cell scheduling problem (FMCSP) with sequence-dependent family setup times, which is one of the most intractable NP-hard combinatorial optimization problems. It is also characterized by a high practical importance for many industrial applications, in particular cellular manufacturing. This problem was denoted as $Fm|fmls,s_{ghj}|C_{\max}$ using the triplet notation. Prior to processing each family, a sequence-dependent family setup time s_{ghj} is incurred on machine j when family f_h is processed immediately following family f_g, where $s_{ghj} = 0$ for all f_g, j. The individual job setup times were very small compared to the sequence-dependent setup times: They were considered as a part of the processing time of a job in each family. A two-level iterated greedy (TLIG) heuristic was proposed to minimize the makespan. Jung, Woo, and Kim (2017) addressed the two-stage assembly flow shop scheduling problem (TSAFSP) containing a machining operation and an assembly operation to manufacture products having dynamic component sizes. This model is common in many real-life industrial applications such as fire engine assembly and personal computer manufacturing. At the machining stage, a single machining machine produces various types of components to assemble the products. A sequence-dependent setup time is required whenever the machining machine starts to process a new component or processes a different component. When the required components are available for the associated product from the machining stage, a single assembly machine can assemble these components into the ready product. To solve the problem, a MILP model was derived. Three genetic algorithms with different chromosome representations were proposed.

For a *job shop environment*, Choi and Korkmaz (1997) considered a basic variant of the problem, which was weighted by sequence-dependent setup times. A MILP model and a heuristic procedure were proposed. Mönch and Drießel (2005) studied a *complex job shop* for the manufacturing of integrated circuits on silicon wafers. The machine environment contained parallel batching machines, machines with sequence-dependent setup times, and reentrant process flows. The goal was to minimize TWT. A two-layer hierarchical approach was suggested in order to decompose the overall scheduling problem.

Scheduling with sequence-dependent setup time consideration in the *FFS/HFS* context is one of the most difficult classes of scheduling problems, both in terms of statement and modeling, as well as in the solution. One of the first works in this shop environment was due to Liu (1996). The author detected that processing in large batches may increase machine utilization and reduce the setup time. However, it also increases the flow time. Therefore, scheduling problems with setup time consideration make a trade-off of the WIP level between flow time, machine utilization by selecting the batch size, and the schedule. The flow of works was unidirectional in the problem statement. A solution must: 1) meet the customers' due dates in the context of the JIT philosophy; 2) reduce the WIP; 3) reduce the machine setup times. While the system constraints are: a) machine capacity; b) product demands; c) precedence relationship. In a feasible schedule, the production must be synchronized with the machine schedule. This NP-hard problem was formulated as a large-scale integer linear programming model. A Lagrangian relaxation method was adopted to solve the problem in the dual space, where the synchronization constraint between the machine setup and part processing was relaxed. An efficient heuristic was developed to adjust the dual solution to primal feasibility.

Liu and Chang (2000) used also a Lagrangian relaxation approach with respect to the synchronization constraints. The scheduling problem was decomposed into two classes of subproblems: 1) part production and 2) machine scheduling. In these subproblems, there were network flow structures in the equality constraints describing the machine status change and the production flow balance. An iterative algorithm of minimum cost network flow was applied to solve the individual subproblems and an efficient subgradient method was adopted to optimize the Lagrangian multipliers. Finally, a machine availability searching heuristic adjusted the solution to satisfy all synchronization constraints by exploiting the network structure, the economic interpretation of the Lagrangian multipliers, and the slack time policy. Yaurima, Burtseva, and Tchernykh (2009) studied a problem in which unrelated machines, sequence-dependent setup times,

availability constraints, and limited buffers in an HFS environment were simultaneously considered. A family of genetic algorithms was proposed as a solution.

4.3.3 Machine/resource-dependent setup times

Machine/resource-dependent setup times were presented in the works by Cheng, Janiak, and Kovalyov (2001) and Chen (2008). In the first paper, a variant of the single machine batch scheduling problem with setup and job processing times dependent on a *continuously divisible recourse* is considered. The authors denoted this problem by BR. It was motivated by part manufacturing on a multi-purpose machining center, where the parts were mounted on, processed and taken off pallets. The pallets were removed and installed by a robot. The setup/removal time of a pallet depended on the robot productivity and the energy consumption. The goal was to produce all part orders by their deadlines subject to saving energy. A linear programming (LP) based polynomial time algorithm was presented to find an optimal batch sequence and the resource values.

Chen (2008) modeled disruptive events as 'machine vacations'. These events may be weekends when a machine operated by workers must be stopped. Consequently, the setup time was affected. An efficient heuristic algorithm was developed to solve the problem of minimizing maximum tardiness subject to the family setup time and vacation constraints.

4.3.4 Time-dependent setup times

The concept of *time-dependent setup times,* which is also referred to as *past-sequence-dependent (p-s-d*) *setup times*, was introduced almost simultaneously by Koulamas and Kyparisis (2008, 1046) and Kuo and Yang (2007, 22) into the consideration of scheduling problems. This means that the setup time is proportional to the length of the already scheduled jobs. This kind of setups is usually presented in problem statements with specific assumptions such as learning/deteriorating effects. These phenomena can affect not only the job processing times but also the setup process lengths.

Koulamas and Kyparisis (2008) explained the motivation for the introduction of this new form of setup times by certain situations in HT manufacturing, in which a batch of jobs consists of a group of electronic components mounted together on an IC board. These jobs must be processed one-by-one by a machine while they are mounted together on the board. The machine operation on any of these components has an adverse effect on the '*readiness*' of all other components, which have not yet been processed due

to the flow of electrical current through the IC board while the machine is operating. Once a component is fully processed, its condition is not affected by the subsequent operation of the machine even if it remains mounted on the IC board. The degree of '*un-readiness*' of any component is proportional to the amount of time it has been exposed to the machine operation on other components. Consequently, prior to a component processing, a setup operation is needed to restore it to '*full-readiness*' status. This setup operation is proportional to the degree of un-readiness of the respective component, and it has no effect on the '*readiness*' of the remaining unprocessed components. When all components on the IC board are managed by the machine, the overall manufacturing process is completed. The authors noted that it is reasonable to model the degree of job 'un-readiness' in the form of a required setup time followed by the actual constant job processing time.

These authors extended also this concept to a learning environment, in which the p-s-d setup times are no longer linear functions of the already elapsed processing time due to learning effects. It was proved that the standard single machine scheduling problem with p-s-d setup times can be solved in $O(n\log n)$ time by a sorting procedure for the following objective functions: the maximum completion time (makespan), the total completion time, the total absolute differences in the completion times, and a bi-criteria combination of the last two objective functions. They also extended the results to nonlinear p-s-d setup times.

Koulamas and Kyparisis (2008, 1046) gave a formal statement of such a single-machine problem as follows.

A standard non-preemptive single machine scheduling problem with a batch of n jobs available at time zero and with a continuously available machine is considered. The notations used are as follows:

p_j - processing time of job J_j , j =1,…, n;
J_j - job occupying the position j in the sequence;
p_j - processing time of the job occupying the position j in the sequence.

The processing of J_j must be preceded by a p-s-d setup time s_j, which can be computed as

$$s_j = \gamma \sum_{i=1}^{j-1} p_j, \quad j=2,\ldots, n; s_1 = 0,$$

where $\gamma \geq 0$ is a normalizing constant.

The value of the normalizing constant γ determines the actual lengths of the required setups. When $\gamma \geq 0$, there is no need for any p-s-d. In such a case, the problem is reduced to the standard single machine scheduling

problem without setups. The setup times can theoretically grow substantially when the batch size n is large. This can be prevented by setting the normalizing constant γ to a very small value or by introducing some type of learning effect on the setup times. In the latter case, the setup times do not grow proportionally with the total length of the already processed jobs. This procedure is repeated with each new batch of jobs.

Kuo and Yang (2007) combined single machine scheduling with a job-independent/job-dependent learning effect and p-s-d setup times. It was assumed that the learning process reflected a decrease in the process time as a function of the number of repetitions, i.e., as a function of the job position in the sequence. The following objectives were considered: the makespan, total completion time, total absolute differences in the completion times and sum of an earliness, tardiness, and common due date penalty. Polynomial time algorithms were proposed to solve optimally the problems with the above objective functions. Later, Wang (2008) extended the result of Kuo and Yang (2007) to the single machine scheduling problem with p-s-d setup times and a time-dependent learning effect.

Wu and Lee (2008) first introduced the *deteriorating setup time* into group technology scheduling in which the group setup times were usually assumed to be known and fixed. The setup or preparation time often requires more time as the food quality deteriorates or a patient's condition worsens. In the paper, if a job has to wait for processing, both the setup times and job processing times are enlarged. The shop environment represented two single machine group-scheduling problems, where the group setup times and the job processing times were both increasing functions of their starting times. The authors first proved that the makespan minimization problem remains polynomially solvable when deterioration is present. In addition, they have shown that the problem with the sum of the completion times is polynomially solvable when the numbers of jobs in each group are equal. For the case of unequal job sizes, a heuristic algorithm was proposed. The computational experiments showed that the performance of the heuristic was fairly accurate when the deterioration rate was small.

Lee (2014) noted that with short product cycle times in many production lines, the workers must constantly learn new skills and technologies. Thus, *forgetting effects* might occur in these situations. The author considered a model, where both the effects of learning/forgetting (deteriorating jobs) and p-s-d setup times were present simultaneously. Optimal schedules for some single machine problems were provided.

Pei, Liu, Pardalos, Migdalas, et al. (2017) combined the *time-dependent setup time* assumption with the effects of time-dependent deterioration and position-dependent learning to constitute a new model for the job processing

time in serial batching problems. The setup time of a batch was a linear function of its starting time. In the next work of the authors (Pei et al. 2018), the batch setup time was defined as a linear function:

$$s_k = \theta t,$$

where

- θ - a parameter denoting the deteriorating rate of the setup time for each batch;
- t - starting time of processing the batch b_k.

The minimization of the makespan was the objective of the studied problem under the constraint that the total resource consumption did not exceed a given limit. Structural properties were proposed for job batching policies and batching sequencing. An optimal batching policy was derived using these properties. A hybrid algorithm, which combined a gravitational search algorithm and a tabu search algorithm, was developed to solve the general case of the problem.

4.4 Family/batch setup times

In a family scheduling model, the jobs are partitioned into F families according to some specific characteristics (see Ch. 5 for more details). This means that no setup is required between jobs of the same family. Let n_f denote the number of jobs in family f, $f = 1, \ldots, F$. However, there exists a *family setup time* on machine i if a job of family g is immediately processed after a job of a different family f, $g \neq f$, and it is denoted as s_{ifg}, or s_{i0g} if there is no preceding job. If for every family g, the condition $s_{ifg} = s_{i0g} = s_{ig}$ is met for all g, $f \neq g$, then the setup times on machine i are *sequence-independent*; otherwise they are *sequence-dependent*. If, for each machine i, the condition $s_{ifg} = s_{fg} = s_{i0g}$ is met for all f, g, $f \neq g$, including the case $f = 0$, then the setup times are *machine-independent*; otherwise they are *machine-dependent*. For the case of a single machine, setup times are, by definition, machine-independent.

The triangular inequality is retained for most production environments. The changeover time between any two products is always less than in the case when a third product is processed between these two products. Formally, the *triangle inequality* assumption holds for each machine i. This means that $s_{ifh} \leq s_{ifg} + s_{igh}$ for all f, g, h, $f \neq g \neq h$, including the case $f = 0$.

As usual, the setups are assumed anticipatory unless stated otherwise. The setups are *non-anticipatory* if the setup, which anticipates the processing of a batch, cannot start on the current machine before all jobs of this batch are released after the processing on any previous machine. This assumption is used in shop problems when there occur release dates in the model. *Sequence-independent family setup times* and *batch availability* may occur if there are two or more successive batches of the same family. In such a case, a family setup time is required before a batch is processed even if it belongs to the same family as the previous batch. If the maximum batch size is limited for a machine i, this is denoted by b_i. If the batch sizes have the same upper bound on all machines, the maximum batch size is denoted by b.

An example of the classification triplet is $\tilde{F}2|b_1 = 1, b_2 = n|C_{\max}$. It denotes the problem of the makespan minimization in a two-machine flow shop, where the first machine is a classical machine and the second one is a batching machine that can process simultaneously up to n jobs. Another example is the problem $1|s_{fg}|\sum C_j$, which denotes the minimization of the total completion time on a single (classical) machine with job families and sequence-dependent family setup times.

4.5 Systematic classification of changeovers

A systematic classification of the changeovers was developed by Stefansdottir, Grunow, and Akkerman (2017) mainly for the food industry, but it is also applicable to any discrete production system.

Studying setup time components in the food and pharmaceutical industries in the context of lot sizing and scheduling, the authors put attention to cleaning activities. They detected that in the process industries, the setup and cleaning classes are often interrelated. The changeovers consist mainly of cleanings, which must be performed while the production is stopped. The cleanings cause long downtimes of the machines and waste of material. Moreover, these activities consume energy, water, and cleaning agents. For lot sizing and scheduling in the process industries, it is therefore fundamental to consider a time for cleanings in addition to setups. The authors developed a general classification scheme for setups and cleanings, which included three different *classes*: *batch-*, *time-, and volume-dependent setups and cleanings*.

Sequence-dependent changeovers are one of the basic characteristics of process industries. A *batch-dependent changeover* (B) may be required when switching between batches. *Time-dependent changeovers* (T) are common in the food industry, in which machines are frequently cleaned at

the end of the day to meet hygiene regulations. For *volume-dependent changeovers* (V), the requirements depend on the produced volume or the frequency of use. These are, for example, often-scheduled changeovers in the chemical industry after a specific number of batches to prevent a buildup of impurities in the processing items. A combination of *time and volume dependency* of changeovers is also possible. For example, fouling of pipelines in process industries could depend on both the time and the volume of the product pumped.

A systematic classification of the changeover characteristics is given in Table 4-1.

The changeovers are classified according to the following general characteristics:

1. Separability;
2. Substitutability;
3. Reference point;
4. Flexibility.

Separability: An inseparable changeover requires that the machine is idle during this activity. This is typical for process industries. On the other hand, a separable changeover is realized offline. Such changeovers are common in discrete manufacturing plants. For instance, machine tools are often prepared offline in assembly plants. However, they also arise in process industries, for example, cheese production, in which the cheese forms are cleaned offline. For the JIT SMED tools, inseparable and separable (or internal and external) changeovers are distinguished, and as many of the changeover operations as possible are treated as separable while running the machines.

Characteristics	Value	Changeover class		
		B	T	V
General				
Separability	Inseparable (insep)	+	+	+
	Separable (sep)	+	+	+
Substitutability	Insubstitutable (insub)	+	+	+
	Across classes (ac)	+	+	+
	Within and across classes (wac)	+	+	+
Reference point	Start/finish of batch (sfb)	+	-	-
	Start of production (sop)	-	+	+
Flexibility	Exact (ex)	+	+	+
	Maximum (max)	+	+	+
	Time window (tw)	+	+	+
Changeover matrix: time/costs				
Product dependency	Product(s)-independent (pi)	+	+	+
	Predecessor product-dependent (ppd) Family (f)/ no family (-f)	+	-	-
	Successor product-dependent (spd) Family (f) / no family (-f)	+	-	-
	Product sequence-dependent (seqd) Family (f) / no family (-f) Natural seq(ns)/no natural seq(-ns) Triangular inequality kept (Δ) / violated (-Δ)	+	-	-
	Predecessor products history (pph)	-	+	+
Batch-size dependency	Batch-size(s)-independent (bi)	+	+	+
	Predecessor batch size-dependent (pbd)	+	-	-
	Successor batch size-dependent (sbd)	+	-	-
	Predecessor and successor batch size-dependent (psbd)	+	-	-
	Predecessor batch sizes history (pbh)	-	+	+

Table 4-1. Classification scheme of changeover characteristics. Adapted from Stefansdottir, Grunow, and Akkerman (2017, 581).

Substitutability: An insubstitutable changeover must take place and cannot be substituted by any other changeover. Changeovers may be substitutable within a class. For example, a time-dependent changeover, which is scheduled every second hour, may be substituted by a daily time-dependent changeover. Furthermore, changeovers may be substitutable across the classes. For example, in the food industry, a less intensive batch-dependent cleaning may be substituted by an intensive time-dependent cleaning. Finally, changeovers may be substitutable within and across the classes. A time-dependent changeover may be substituted by another time-dependent changeover and by also a volume-dependent changeover.

Reference point: It represents a base for the starting time of a changeover. The starting time of a batch-dependent changeover is always based on the start or finish of a batch. On the other hand, for time- and volume-dependent changeovers, the starting time is based on the start of the production, the start of the last changeover, or on a fixed time point. A fixed time point is, for example, the beginning of the planning horizon. The start of the production and the beginning of the planning horizon are the same for the problems with makespan minimization.

Flexibility: It characterizes the changeover starting time. Changeovers may require an exact starting time with no flexibility. These are, for example, time-dependent cleanings in a food production may be programmed at the end of each production day. Changeovers may also be more flexible, i.e., an upper bound on the starting time (maximum) or both upper and lower bounds may be used.

The general changeover characteristics and the structure of the changeover matrix are illustrated graphically in Fig. 4-3.

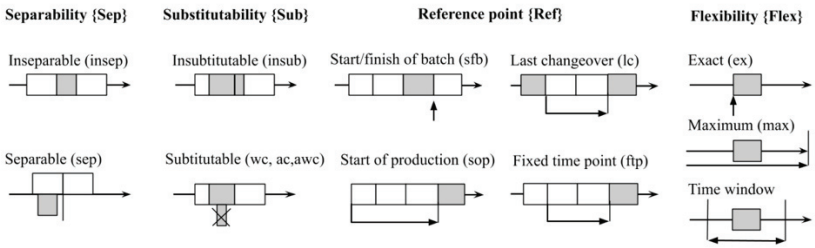


Fig. 4-3. Graphical representation of general changeover characteristics. Adapted from Stefansdottir, Grunow, and Akkerman (2017, 581).

The changeover matrix-related characteristics are classified according to a) product dependency and b) batch size dependency.

The changeover matrix characteristics can be formulated in two variants: time and/or costs. The product dependency of a changeover may be *product-independent* as it frequently occurs in the case of time- and volume-dependent changeovers. This is also usual for *batch-dependent changeovers*. *Batch-dependent changeovers* may also depend on the *predecessor* or *successor batch*. The family structure is a property of predecessor- and successor-dependent changeovers. A changeover is *minor* when switching between products of the same family. A changeover is *major* when switching between families. A batch-dependent changeover may be *product-sequence-dependent*. In such a case, a changeover depends on both the predecessor and successor batch (Fig. 4-4).

Using the classification system represented in Table 4-1, any changeover can be categorized according to the class (B, T, V), the general characteristics (*Sep/Sub/Ref/Flex*) as well as the changeover matrix with respect to time *t* (*Prod/Size*) or costs *c* (*Prod/Size*). Then the classification record of a changeover takes the form:

Class; (*Sep/Sub/Ref/Flex*); *t*(*Prod/Size*); *c*(*Prod/Size*).

This classification may be illustrated by two examples from the semiconductor industry, where the changeovers often take the form of time-dependent cleanings, which are done on the machines at the end of a production day. Assuming that this cleaning is substitutable across the classes and with a product- and batch-size-independent changeover matrix defined for the times, the changeover classification takes the form:

T; (*insep/ac/ftp/ex*); *t*(*pi/bi*); *c*(-/-).

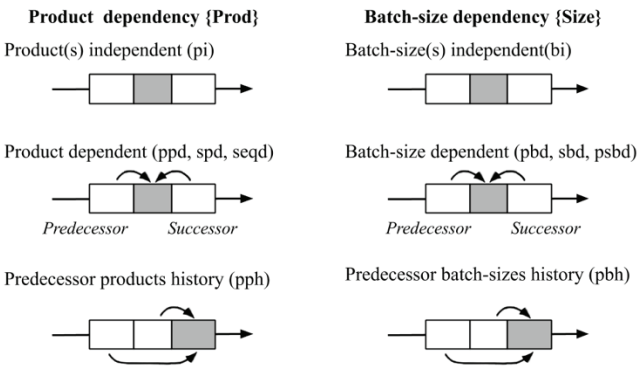


Fig. 4-4. Graphical representation of changeover matrix setup/cost characteristics.

The second example concerns a typical changeover from one product to another one on a production line. This is often an inseparable and insubstitutable sequence-dependent batch changeover. If it is assumed that a changeover matrix is defined for costs, it has not a family structure or natural sequence in which the triangular inequality is maintained, and it is batch-size-independent, then this changeover is classified as:

B; (*insep/insub/sfb/max*)*; t*(-/-); *c*(*seq*d*(-f;-ns;* Δ)*/bi)* .

4.6 Setup and changeover reduction schemes

According to Gest et al. (1995, 205), the need to reduce changeover times, called *set-up reduction* (SUR), was first declared by Shingo (1985, 11–20). This philosophy arose from Toyota's JIT-based production system. Since then, an active investigation of setup times started from different viewpoints, such as detailing the setup structure and developing setup time reduction methods. In the literature, one can meet two classes of setup schemes: GSU/SDS and more recently, SMED/OTED, which start with a detailed analysis of the setup/changeover structure.

4.6.1 Categorization of techniques

Although the available information about changeover reduction is extensive, it is usually unstructured and very difficult to access. Nevertheless, it was categorized into the thirteen primary classes listed below (Gest et al. 1995, 206):

- Elimination;
- Motion systems;
- Move internal to external;
- Access;
- Clamping;
- Adjustment;
- Clean out/purging;
- Machine design;
- Product design;
- Tooling design;
- Fixturing aspects;
- Housekeeping;
- Preparatory work.

Elimination
An ideal changeover would be one in which there is no time lost during its course, that is '*the best changeover is no changeover*' (Gest et al. 1995, 206). However, to achieve this, a very high investment capital is usually required such as extra machines, duplicate tooling or a new machine and system design. Typically, two methods are available for the virtual elimination of setups: 1) *excess capacity* and 2) *automation*. In the first case, excess tooling or machine capacity is employed to eliminate the need for setups by using duplicate machines or tooling, which can be simply substituted for the old product set. Alternatively, an excessive capacity allows an adjustment of the machines before the run of new products. The automation of setups has been widely realized by the use of machine centers and a computerized control.

Motion systems
The gathering and delivery of tools and equipment, which are required for a changeover, have a significant effect on the overall changeover time. AGVs should be used for the transport of material to/from the machines, although some systems include the use of an AGV to deliver a pre-set tooling. For example, the vehicles cease to be an appropriate solution when the press or tooling of a large capacity range (1000-ton press) must be transported. In this area, railed carts and bolsters are more common. The simplest way to solve the problem of tool delivery is the use of roller and ball trolleys. A set of air die rollers can be embedded into the bolster of such a press to facilitate the delivery of the dies.

Move internal to external
The easiest way to achieve a significant reduction in a setup time is to move certain aspects to an external stage, where adjustments can be made beforehand and tools can be prepared before the actual machine stops the production. Shingo highlighted the importance of moving internal operations to external ones. He described the case of Arakawa Auto Body, where the provision of an external setup table reduced the setup time from 10.5 minutes to 30 seconds. However, an effective scheduling system is required to enable changeover elements to be carried out as an external operation and for more efficient setups in general. This ensures a correct preparation for a changeover, i.e., the presence of the correct tools and materials at the machine at exactly the right time.

Access
When the access to the tools and resources is difficult, it can take a significant effect on the setup time.

Securing
Securing operations, such as clamping, play a significant role in the changeover time due to two factors:

1. Engineers use a too high safety factor in the design of clamping, sometimes up to 30 times;
2. It is due to the use of screw thread devices for clamping purposes. A number of different methods have been developed, where a clamp handle simply needs to be turned slightly to clamp or release.

Adjustment
Despite having, a very low tool change time (unclamp, unload, load, clamp), the setup time could still be excessive due to a high proportion of the time taken to adjust the machine and/or the system to an operating state. The losses of time due to the adjustment can reach ten times of that of the physical changeover time. The use of GT permits to reduce the range of the features within the part families, which are produced by a machine due to the similarities between parts. The adjustment can even be eliminated by settings, which use the exactly defined positions. The standard gauges can greatly reduce the adjustment time and increase the accuracy. Simply slotting in a new gauge can immediately change the settings on a machine with a desired accuracy.

Clean Out/Purging
Purging between batches can cause severe problems within the process industry such as food, paint, plastics, and chemical production. An automatic changeover and purging can be realized, for example, by using hoppers mounted on an automatic shuttle system to simultaneously unload the old tools and load the new tools. Then a changeover occurs almost instantaneously. In the printing industry, duplicate drums are utilized for a quick changeover by removing the clean out to the external arena.

Machine design
An effective machine design is essential to reach a zero-setup time. Most of the previously discussed aspects such as access, location, adjustment, etc., can be improved by an appropriate design of the machines. The design of a machine, its interaction with tooling and associated systems, as well as the

amount, by which ergonomics are taken into account, are critical aspects of the setup problem. The following aspects are highlighted in this category:

- The standardization of clamping and location can drastically reduce the changeover time;
- An over-adjustability of machine tools should be revised;
- The use of many small dedicated machines is preferable to one large machine that requires a great deal of changeover and adjustment.

Product design
If the parts are designed using as many common features as possible, then the setup between the runs can be significantly reduced and many problems can be countered at the design stage.

Tooling design
When a tool such as one dedicated to work in the presswork field, has top and bottom halves, which are joined together, any adjustment in-between can be eliminated. As a result, the tool can be instantly loaded and located. Additionally, the need for soft tooling is eliminated along with the associated rework and shearing.

The use of 'indexable' tools for the machining industry simplifies the changeover process. In such a case, the operator is no longer required to undo the previous tool, get the new tool, pack it up to height and then check its center.

The weight of tooling has a significant effect on the delivering methods used, for example, if the transportation equipment used for this process is employed elsewhere. A welding jig weighting 30 kg requires a hoist to be loaded into the welding machine. Delays in the hoist delivery result in excessively long setups in such a case. The load and setup times can be reduced drastically if the jigs are sectioned in such a manner that only the required parts are changed.

Housekeeping methodology
This aspect is about the organization of the workplace, including the techniques to maintain several points, such as following ones:

- The changeover personnel must have skills to quickly and easily locate tools and make efficiently required changeovers;
- Coding dies and keeping them adjacent to the dedicated machine;
- Ensuring that all tools, gauges, etc., are available for the production;
- Using two people rather than one person;

- Arranging the workplace so that everything can be easily found;
- Ensuring that all tools, etc., are where they are supposed to be in a clean state and ready for use, etc.

Preparatory work for a setup reduction
Some of the following suggestions should be taken into account:

- Be concentrated on one line, usually on the bottleneck;
- Develop and implement a training program for the shop floor and support the staff;
- No cost/low cost ideas first;
- Set targets, for example, 50% reduction in six months;
- Time savings are used for more frequent setups;
- Standardize the setup method once improved.

4.6.2 Practical approaches

There are several practical approaches used to reduce the overall setup time as well as the sequence-dependent setup time. These approaches are mainly described in earlier publications. Afentakis, Gavish, and Karmarkar (1984, 223) proposed to enlarge the lot sizes. Nevertheless, this method leads to an accumulation of the WIP. It may also be impossible to create large lot sizes. A second method proposed by Boyle (1986, 1042) consists in reducing the setup frequency. It is essentially based on the GT concept, which was initially proposed for a single machine environment. A similar method was proposed by Kusiak, Vanelli, and Kumar (1985, 245–52). It was based on grouping parts and fixtures in FMSs.

The idea of the SDS method is that the PCB types should be sequenced such that a following PCB will have a maximum of the common components as the current PCB, thus eliminating many setups in-between. The goal is to minimize the number of component changes required during the sequence. The products requiring the same limited resources (jigs, fixtures, etc.) must be scheduled separately from each other to reduce the waiting period of these machines. The SDS method sacrifices several reductions in the setup time but has some advantageous qualities for such a type of a production system. This approach maintains a low WIP level (Fig. 4-5a). A description of the SDS method can be met in the paper by Maimon et al. (1993, 179–80).

The GSU method was introduced by Carmon, Maimon, and Dar-El (1989, 1795) for a multi-machine environment trying to reduce the cost and to increase the throughput of small-lot PCB assembly lines. The main idea

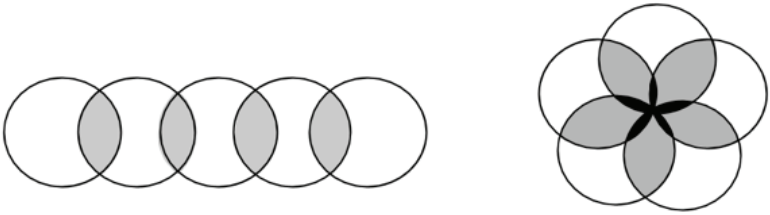
of this method is that the PCB products are divided into groups. Each group is processed in two stages:

- The *first stage* is characterized by the common setup and assembly operation. The *common components* share a setup on the machines, once only for the whole group and then for the assembly onto their respective PCBs. The common components are the components, which are shared among two or more PCB types in the group. This stage is referred to as *common setup and production*;
- The *second stage* is referred to as *residual setup and assembly*. During this stage, a separate setup and an assembly of the remaining components on each PCB type are sequentially performed.

Therefore, the production stages on each machine are as follows:

1. Setup of common components;
2. Assembly of common components on all PCBs in the group;
3. Setup of residual components;
4. Assembly of residual components on each PCB separately.

The grouping problem can be viewed as a clustering problem. The GSU method was shown to result in a high throughput but also in a high WIP inventory level. A graphical presentation of the GSU method is given in Fig. 4-5b. The spaces marked represent the savings in the setup time.



a) *b*)

Fig. 4-5. Graphical illustration: a) SDS method; b) GSU production method.

The reader can also meet a description of the GSU method in the paper by Maimon, Dar-El, and Carmon (1993, 179–179).

The savings under GSU should exceed those under SDS, in which some common components may need to be setup more than once. As the group size is enlarged, the saving in the setup increases since each product type added to the group typically contains some common components that are

already set up on the machine. However, each PCB added also increases the production makespan and the lead time of all PCBs in the group. For this reason, while defining the groups of products, the due dates of all product types should be considered.

Ovacik and Uzsoy (1996) presented some dispatching rules to decompose the general complex job shop problem for testing facilities into a number of work centers and then to simplify the management of setups with the goal to reduce the WIP. Leon and Petters (1996) suggested a partial setup strategy for replanning purposes on a single-placement multi-product machine in a PCB assembly system. The partial setup proposed is a combination of a unique setup for each product and a group setup for a group or family of similar products. Lambert et al. (1997) considered both approaches, SDS and GSU, combined with the family shortest processing time (FSPT) first scheduling rule for a surface mount technology (SMT) production line.

4.6.3 SMED/OTED method

Nowadays, customers require a wide range of products of high quality delivered with quick response times and sold at reasonable prices. To survive under such increasingly competitive conditions, there is a need for a continuous improvement in every type of industry. The companies are forced to produce smaller lots with the same demand as in previous years, without affecting their global productivity. However, the fabrication of more products at smaller lot sizes requires a larger number of changeovers. Consecutively, a rapid changeover capability is critical for being able to produce small quantities of a large diversity of the products, which is the basis for a pull production.

Van Goubergen and Van Landeghem (2002) indicated three main reasons why setup reduction initiatives are appropriate for any company:

1. Increase of the flexibility by conducting more changeovers and reducing the lot size;
2. Increase of the bottleneck capacities in order to maximize the line availability for the production;
3. Minimization of the production costs since these are related to the equipment effectiveness.

The SMED methodology was developed by the Japanese engineer Shigeo Shingo in the 1950s in response to these challenges of the production with short changeover times (Shingo 1985). SMED is focused on the

reduction from hours to minutes, the time required to move from producing one product to another one. With the reduced setup time, the production flexibility expanded as it was able to afford more frequent product mix changes and to detect the production bottlenecks. Generally, SMED aims to standardize and simplify the manufacturing operations. An important point is that by this means, the need for special skilled workers is also minimized.

SMED is one of the lean production methods for reducing the waste in a manufacturing process. It provides a rapid and efficient way to convert a manufacturing process from running the current product to running the next product. This rapid changeover is a key to reduce the production lot sizes and thereby to improve the flow. The words 'single minute' do not mean exactly that all changeovers and setups will take no more than one minute, but that they should take less than 10 minutes (meaning 'single-digit minute'). SMED was originally developed through the analysis of a die change process to improve die press and machine tool setups, but its principles are applicable to changeovers in all types of processes. The SMED technique is used as a part of the TPM system (Nakajima and Bodek 1988) and as the philosophy of continuous improvement called *kaizen*[9] in various studies to reach lean manufacturing.

Shingo divides the setup operation into two parts:

1. *Internal setup*: the setup operation can be done only when the machine is shut down (for example, attaching or removing the dies);
2. *External setup*: the setup operation can be done when the machine is still running.

These operations can be performed either before or after the machine is shut down (for example, getting the equipment ready for the setup operation can be done before the machine is shut down). The four conceptual stages of SMED can be defined as follows (Fig. 4-6):

Preliminary stage: The internal and external setup conditions are not distinguished;

Stage 1: Separate the internal and external setups;
Stage 2: Convert an internal setup to an external setup;
Stage 3: Streamline all aspects of the setup operation.

The application of Shingo's methodology usually results in two main benefits:

[9] The word *kaizen* comes from two Japanese words: '*kai*' meaning 'change' and '*zen*' meaning 'good'.

- Increasing the manufacturing capacity;
- Improving the equipment flexibility.

This allows working with smaller batch sizes, creating a flow of materials by eliminating waiting.

The SMED concept was further expanded with One-Touch Exchange of Die (OTED), which is closely associated with SMED but a more difficult concept. OTED states that the changeovers can and should take less than 100 seconds. This means also an ideal reduction or elimination of the setup effort required between the operations on the same equipment (Shingo 1985).

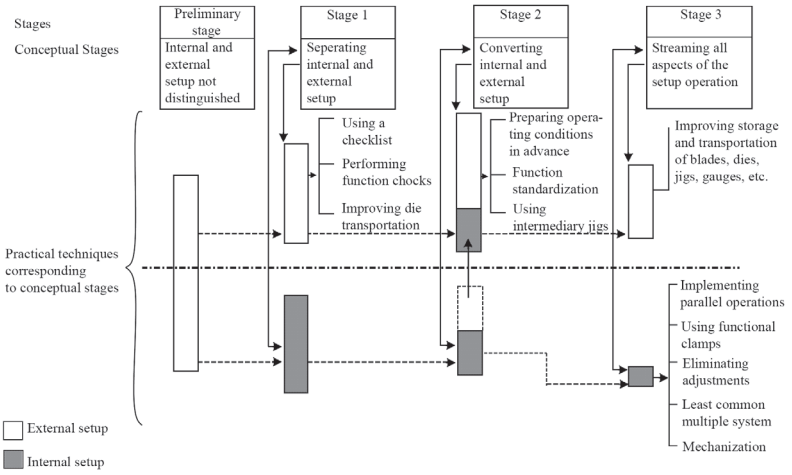


Fig. 4-6. SMED conceptual stages and practical techniques. Adapted from Ulutas (2011, 1195).

4.7 Conclusions

It is difficult to imagine now that the researches considering setup/changeover times started only in the 1980s when serial manufacturing companies had already a high level of the environment and the scheduling theory had already a 30-years history starting from Johnson's seminal work. A deep analysis of the structure and time of the setup activities gave the origin to change the production philosophy to a continuous improvement provoking a rapid growth of the scheduling theory, an appearance of numerous models and approaches.

The consideration of setup times has two aspects, namely:

- Reduction of time resulting from the improvement in the manufacturers' activities;
- Development and application of scheduling methods.

The first part is close to Toyota's improvement philosophy. Nevertheless, there is still a big interest of researchers to scheduling problems with setup consideration. Allahverdi, Gupta, and Aldowaisan (1999, 219) published a detailed review on scheduling problems involving setup considerations for different classes of models and noted the following:

> 'For a separable setup, two problem types exist. In the first type, setup depends only on the job to be processed, hence it is called sequence-independent. In the second, setup depends on both the job to be processed and the immediately preceding job, hence it is called sequence-dependent'.

Actually, with an artificial intelligence outlook, we can see that there appear new and practical kinds of models, which are every time closer to the challenges of the industry. Particularly, new types of a setup appeared, such as resource-dependent and time-dependent setup times. With the presented state-of-the-art survey, the reader can evaluate a growing interest to p-s-d setup times, setup times with deteriorating jobs, as well as setup times with forgetting and learning effects. These new concepts appeared first in single machine models, but must be studied further also for other environments. In addition, surveys involving these new concepts are required.

The reader has a variety of reviews to study setup concepts in different contexts. Surveys on scheduling problems involving setup times/costs were presented by Allahverdi in co-authorship with other recognized researchers in 1999, 2008 and 2015. The authors gave a deep comprehensive analysis of the up-to-date situation in this area. A literature review about scheduling

and lot sizing with sequence-dependent setup times was given by Zhu and Wilhelm (2006). Jungwattanakit et al. (2009) reviewed algorithms for FFS problems with unrelated parallel machines, setup times, and dual criteria. Sharma and Jain (2015) dedicated their review to job shop scheduling with setup times. Recently, Godina et al. (2018), and Silva and Godinho Filho (2019) presented systematic literature reviews on the latest trends in the SMED, where the reader can meet references to related publications classified into numerous categories.

CHAPTER FIVE

GROUP TECHNOLOGY

GT provides, in fact, the foundation for an evolutionary approach to complete automation.
(*Burbidge 1991, 26*)

If one looks at a plant, which manufactures many different products, it can be found that some products are similar to each other in the shape, the construction, the production process, and the resources they require. For example, a company produces *N* different products. However, if we would analyze all nomenclature, it can be noted that some products are similar and can be grouped, say, one group of *x* products, another group of *y* products, and so on. To be efficient, all resources, which each group needs, should be placed in a separate area, called a *machine cell* (MC). In such a case, the production of a group of components, called *part family* (PF), would be done efficiently because the products are similar and all resources required are located in close proximity. The main idea of GT is to explore the similarities, which exist among the components of the products, then based on simplification and standardization methods, to organize the manufacturing activities in such a manner that the production efficiency is improved. This approach has various aspects and methods, which are considered in this chapter.

5.1 Group technology history and philosophy

HV/HM industries are characterized by the manufacturing of a large number of components of the same size and configuration. The equipment necessary for this type of production requires a high capital investment and a high operating flexibility. However, during recent years, manufacturing companies direct their steps towards a *multi-product* and *small batch* production with the goal of adapting to the market movements. These movements are characterized by diversified and specialized demands of the society as well as short product life cycles. The management of a large variety of products to be manufactured with small batch sizes brings

enormous complexities into the production process, where frequent problems may arise, such as manufacturing inefficiencies, poor workflow, high machine cost, high setup time, large inventories, and delivery waiting, among many others. In order to face such difficulties in dynamic environments, the production management methodologies, such as SMED, are insufficient to maintain a high flexibility of the manufacturing. GT and Cellular Manufacturing (CM) approaches dispone internal instruments to improve the production system while reducing the lead times as well as seek to eliminate or minimize the planning complexity and to improve or maximize the productivity.

The term *Group Technology* has numerous definitions. Maybe, the most frequently cited definition belongs to one of the earliest authors, Professor V. B. Solaja, who was the Director of the Institute of Machine Tools, Belgrade (former Yugoslavia) (Solaja and Urosevic 1969, 157):

> 'Group Technology is the realization that many problems are similar that, by grouping, a single solution can be found to a set of problems thus saving time and effort'.

In a first survey dedicated to CM, Greene and Sadowski (1984, 85) defined GT as

> 'bringing together and organizing (grouping) common concepts, principles, problems, and tasks (technology), meanwhile CM is the physical division of the functional job shop's manufacturing machinery into production cells, where each cell is designed to produce a part family'.

One of the most recognized authors in the GT area, John L. Burbidge, offered in his last published paper in 1996 the following definition (Burbidge 1996, 261):

> 'Group technology is a method of organization for factories in which the machine tools, other processing facilities, and people, are divided into groups. Each group completes all the parts it makes, at the processing stage at which it operates. The machine in each group, must be laid out together in one place'.

For many years, GT did not receive the formal recognition, which it deserved, and had not been rigorously practiced as an integral approach to the productivity improvement. Nevertheless, its basic concepts have been practiced around the world for many years. They can be identified in earlier publications under different names and in various forms of engineering and manufacturing functions, such as 'good engineering practice' and 'scientific

management', for example, in a paper by Hathaway, which was published in 1920.

Systematic studies of the organization of the production commenced in the USSR since the 1930s by A. P. Sokolovskiy, S. P. Mitrofanov, and other leading Russian engineers, which made some pioneering works in the areas of simplification of job processing and reduction of the setup times. According to Burbidge (1991, 8), the term *Group Technology* was introduced by Sergey P. Mitrofanov, a Professor of the Leningrad State University, USSR, as the title of his research about the relationships between the component shape and processing methods. The new theory was formalized in his book entitled *The Scientific Principles of Group Technology*, first in Russian in 1946, and after then translated into English in 1966 (Mitrofanov 1966, 1946).

According to Zhang (2013, 33), Mitrofanov's contributions were as follows:

- He found that considerable reductions in the setup times, which lead to an increasing capacity, could be achieved with lathes. In this case, a *group* of similar parts is created. Then these parts are loaded on the machine one after another with the same setup;
- He also specified a simplification of design and a standardization of process as imperative prerequisites to a GT program, and suggested a dual classification and coding system: one part for the simplification of design and the other one for the standardization of process;
- He continuously affirmed that the classification of technical operations based upon component shape, surfaces, and features afforded the best solution for this problem. He illustrated the role of classification as the basic problem or solution on which GT is based.

From 1960 on, the factories in the Soviet Union used this GT, and there was created a great bibliographic collection on this subject, including the fundamental works (Ivanov 1968; Petrov 1968), where, in particular, it has been noted that a redesign of the production planning and scheduling system is required when applying GT principles to the organization of the production. The interest to GT in the USSR may be explained by the specifics of a centralized planned economic system, which was peculiarly adequate to emerge GT and strong national standardization policies such as establishing detailed ranges, which were useful for the development of this technology.

At this time, there existed several reasons to make the GT implementation attractive for industrial companies, among others:

- An intensive economic progress and the appearance of big manufacturing companies had difficulties in the fabrication of complex products;
- There was a request to reduce the production costs;
- At this time electronic devices appeared for the industrial use and data processing, which required an efficient classification and coding systems for the manufacturing components;
- There was a common idea to switch from mass manufacturing to small batch production;
- Due to the rapid technological progress, the manufacturers asked for a flexible organization and highly productive manufacturing systems.

Since the release of Mitrofanov's book, GT has attracted a worldwide interest. This methodology originally emerged as a single machine concept that was created to reduce setup times. This method was further extended by Burbidge in a series of works in the years from 1963 to 1996, which served as a base for the creation of the CM concept.

According to Burbidge (1991, 6), an engineering company in Alsace followed Mitrofanov's mode. They took a section of lathes and added two milling machines. Then two drilling machines were added to form a group, which completed all parts it made. At this stage, the meaning of the term *group* was changed from a *set of parts* to a *set of machines*. The new term of a *family* (*famille des pièces*) was introduced to describe a set of parts. In the 1960s, the German Professor H. Opitz developed at the Aachen Technical University the most commonly used GT classification system, which received his name OPITZ, for the design selection, the process planning, and the cell formation (Opitz, Eversheim, and Wiendahl 1969). A hierarchical code system was developed in the VUOSO research institute, located in former Czechoslovakia, for the optimization of the machine tool design.

Meanwhile, in the 1950s, various research institutes in the USA, such as the Brich, Brin & Partners Consulting Company, Alliss-Chalmers Manufacturing Company, and others, worked on the development of GT techniques. E. G. Brisch and J. Gombinski adopted the ideas of Mitrofanov's seminal works. They popularized new paradigms in scientific and engineering communities for the development of classification and encoding techniques as well as the application of them in all aspects of industrial activities. In England, the companies Serck Audco, Ferrodo, and Ferranti implemented the GT with the support of Brisch's consulting company. At that time, it was probably the most advanced approach in the utilization of GT concepts among all industrial countries. The government

of England has supported a Group Technology Center since 1968, and the British Institution of Production Engineers has formed a Group Technology Section. Nevertheless, Eng. Joseph Gombinski, the Managing Director of the Birch Company, highlighted in his paper the urgent need to improve the productivity in the British industry. He also noted a lack of attempts, which have been made to apply GT in the country because with GT, a substantial increase can be achieved in the productivity of engineering industries, particularly when the production is on a one-off or small batch basis. These increases arise from an improved utilization both of the manpower and the machines with a corresponding cost reduction. Giving such a high importance to the GT application, Gombinski concluded that the resistance to change may be one reason for the slow acceptance of GT (Gombinski 1967, 557). More than 50 years after the publication of this paper, one can conclude that the principles and methods of GT are still implemented insufficiently in modern manufacturing.

In the 1970s, the concept of standardization was stabilized in GT, while a general classification of the attributes of the components was made, namely:

- Geometric or graphic characteristics: size, shape, etc.;
- Functions of the components: handle, clamp, etc.;
- Depending on the manufacturing type: lot size, process path, etc.;
- Raw material of the piece.

From the 1970s on, Professor John L. Burbidge conducted work in Turin, Italy, and then at the Cranfield University, UK. He made a significant contribution to the methods of Production Flow Analysis. Then in the 1980s, different coding methods were developed in the USA, Japan, and Europe. At the same time, many companies implemented the JIT method. The JIT philosophy implied changes at all levels of the production, from the manufacturing by the functional departments to the manufacturing cells, where the maximum number of operations is made and the staff is responsible for the product quality, in addition to seeking a continuous improvement at all production stages, from the design to the final product. Then the theory was extended to grouping and scheduling problems in shops with setup times dependent on the job sequence (Flynn 1987).

The concept of the *coefficient of similarity* between the individuals for the group formation attracted attention first for the taxonomic studies at the end of the 1960s (Rubin 1967, 108; Gower 1971). Initially, it was defined and used as a parameter, allowing products to be grouped by a heuristic method, instead of the basic concept of 'exploiting similarities', which was

taken from the GT philosophy and which has been used to address the classification problem in a creative way. From the pioneering work by McAuley (1972) onwards, a systematic exploitation of similarity coefficients in GT has begun to group parts into families and create machine cells.

Although technological improvements are typical for modern manufacturing plants, nowadays it is rather common to have high setup times, up to 8–24 hours, in some phases of the production processes like in the tile or semiconductor manufacturing (Delgado-Arana et al. 2017, 208). Therefore, is still important to seek for a reduction of the lead time by using the JIT, SMED, and other philosophies. It is the reason to follow with the development and implementation of GT into practice. In Fig. 5-1, one can note a stable interest of researchers to the GT theory and applications.

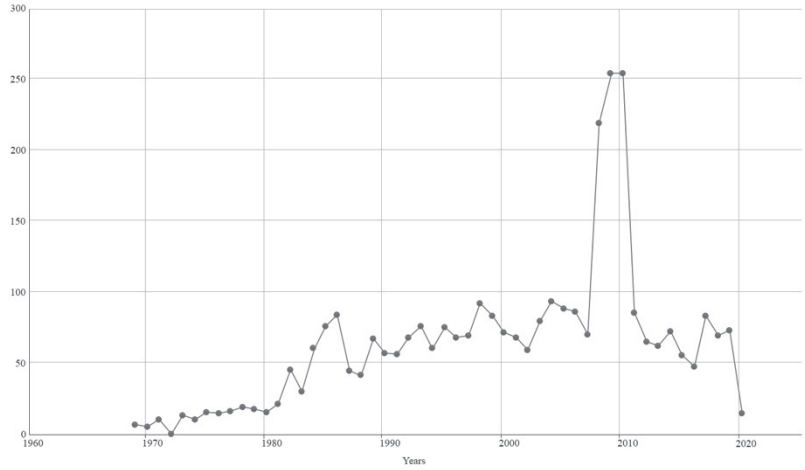


Fig. 5-1. Number of documents with references to GT in Scopus in the period 1969-2019.

According to the Scopus[10] database, there are 3,278 documents with the words 'Group Technology' in the article title, abstract, and keywords areas in the period from 1969 (first reference) to March, 2020, and 865 documents in the period 2010-2019.

GT principles may be applied to any industrial entity ranging from the manufactured parts and the capital equipment to the decision processes, and to human characteristics with the aims to take advantage of the similarities that exist among the items, with the goal to increase the corresponding effectiveness. Principal aspects and approaches of GT are considered thereafter.

5.2 Part family

A *part family* is a collection of similar parts that share a specific design and/or manufacturing characteristics, identified for a well-defined purpose (Tatikonda and Wemmerövr 1992, 2088). The part family is the key concept of GT. Components having similar characteristics are grouped into families, and a new family member can be created by modifying the characteristics of an existing component from the same family.

The advantage of the use of similarities that exist among the items appears through increasing the effectiveness by:

- Allowing similar, recurring activities to be realized together (for example, part family scheduling);
- Standardizing similar actions to control new activities and the efficient utilization of the resources;
- Supporting the access and recovering the actual and historical information retrieval to make that information accessible and usable.

The group attributes, which are usually considered, are as follows:

- *Design*: geometry, shape, tolerances, finishing, material;
- *Manufacturing*: production process, processing time, sequence of the operations, tools required, fixtures required, lot/batch size;
- *Combined*: features, which combine the best characteristics of both the design and manufacturing attributes.

The parts in a family may use similar treatment, handling and control methods. These methods do not require a considerable adjustment in-

[10] Scopus is the abstract and citation database of the Elsevier publisher.

between, therefore the setup times are eliminated or reduced when they are processed together. An efficiency increment is achieved due to various indices, such as joint scheduling, facilitated control, standardized product design, group layouts, and advanced personal learning, among others.

It is typical that the parts, which belong to the same design family, are also produced by using similar manufacturing processes. However, there may occur a different situation. Let us suppose that there are two parts with identical design, but one part is made from plastic and the other one from steel. The manufacturing processes would be an injection molding for the plastic reel and a turning operation for the metal reel. In this case, the family design is common; however, the production processes are unrelated. It is the reason why the part families are assigned to groups along with the machines required to produce these parts.

5.3 Grouping methods

Part families may be formed in one of the two ways as follows:

1. Grouping parts, which have similar design attributes within a certain dimensional range and most or perhaps all machining operations in common;
2. Grouping parts, which have dissimilar geometry but one or more machining operations in common. This is rather a similarity in the production process than in the form or size.

Therefore, before grouping, the *objectives* of generating the part families must be made clear, whether they are related to the product design or manufacturing applications. Once the objectives are determined, the relevant attributes for a part family are identified. Then the part families may be formed on the base of an appropriate grouping method.

Various methods are available for creating the part/machine groups and formation of cells in the literature. As can be seen from Fig. 5-2, the principal approaches are:

1. Visual inspection;
2. Classification and coding;
3. Production flow analysis (PFA).

These methods are briefly reviewed below.

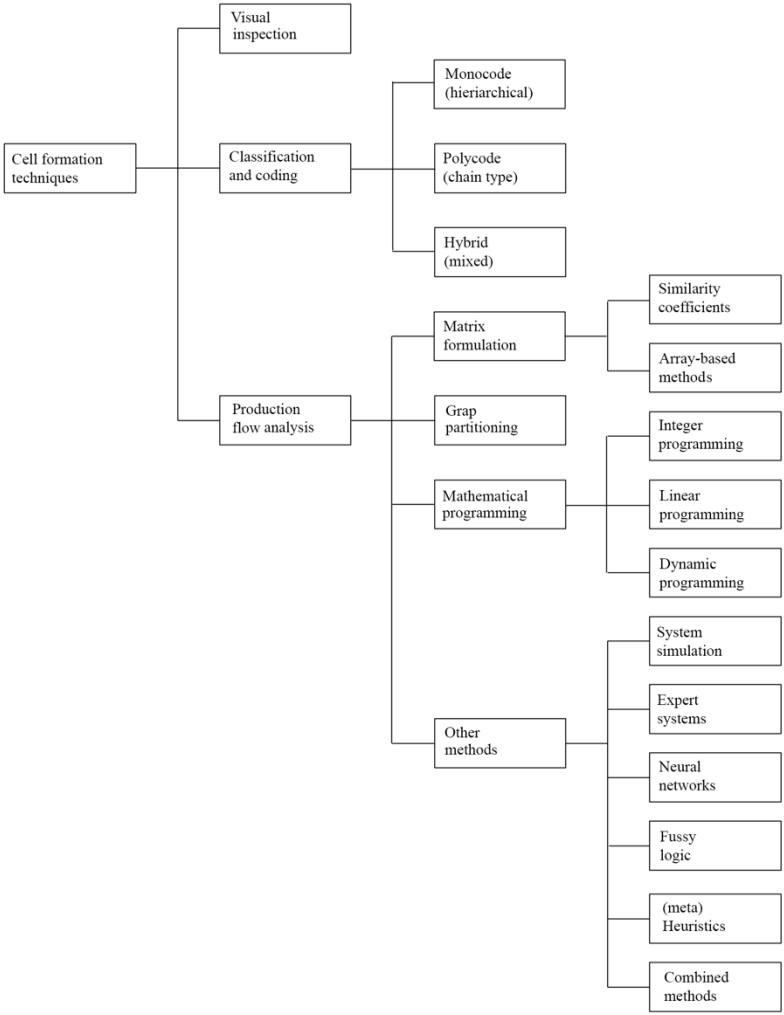


Fig. 5-2. Taxonomic framework for GT.

Visual inspection is an archaic method to form part families. It is simple and the least expensive one. An expert evaluates empirically the parts with similar features and classifies them into appropriate families on the base of the critical attributes and the proper experience. This method is prone to error, relies very heavily on the expertise conditions and the experience of the inspector. Actually, it is rarely used in practice, except, maybe for a

small number of parts. It may be employed as a starting point by which one can begin to judge, whether GT seems to make sense in his environment.

The *part classification and coding* approach identifies similarities and differences among the parts and assigns codes depending on their characteristics and according to a coding scheme. Based on these codes, parts are grouped into part families. This method can also be used for the inverse procedure: identification of part families using a coding scheme. The classification and coding systems are considered more in detail in Section 5.4.

Production flow-based system, which is also referred to as PFA, is a technique for finding both GT groups and their associated families by analyzing the operation sequence and the routing of the components over the machines in the plant. The concept of PFA was first introduced by Burbidge (1963). Then the author extended this concept in a series of papers and books. He proposed the change of the GT paradigm from the classification and coding meaning (Burbidge 1991, 10):

> 'It is probable that parts which are similar in shape or function can be made by the same group (set) of machines',

to PFA:

> 'Parts which are made using the same set of machines can be made in the same group'.

Burbidge indicates also another difference between classification and coding from one side and PFA from the other side:

> 'PFA is a technique for simplifying material flow systems, while classification and coding ignores this aspect of the problem'.

Actually, classification and coding systems are mainly developed, but the attention of researchers still attracts the PFA approach. Various methods, which have been developed since then for machine/part grouping methods, belong to this category. In this book, the PFA method is described more in detail in Section 5.5. The extensions of this method are discussed in Sections 5.6-5.7.

5.4 Classification and coding systems

Classification is a basic concept in GT. It represents a logical and systematic way to declare the similarity of things to form groups and detect differences inside of these groups to form subgroups. In the following section, the main

definitions of this part of the GT theory are given, a taxonomy of the classification and coding systems is discussed, and the Opitz code, which is a comprehensive coding and classification system for work pieces, is described.

5.4.1 Basic classification assumptions

Three types of activities are necessary for applying the GT (Tatikonda and Wemmeröv 1992, 2088):

- Determination of critical part attributes that represent the criteria for a part family membership;
- Allocation of parts to established families;
- Retrieval of part family members and related information.

Classification and coding systems assist in these activities by providing a code structure for jointing parts into groups, which are based on the selected attributes and by assigning a code to each part.

A classification and coding system groups the parts into families. The *classification* assists in assigning a group to a part on the base of the selected attributes, and the *coding* serves for the identification of a unit by its code. The classification precedes the coding of parts, that is, the assignment to the class, to which a part belongs, is first performed for each critical part attribute.

Some reasons for applying the classification and coding procedures to the parts are:

- A fast identification of a similar family member in the database. It may be useful for the design of new parts;
- The part code for a new part can be employed for planning using already existing parts, which have identical or similar codes;
- The part codes can be taken to design the machine cells capable of producing all members of a particular part family using the composite part concept.

There exist numerous possible coding strategies. It is difficult to know in advance, which attributes can be identified and which software can be used. Therefore, it is important to guarantee the flexibility of the database structure to add enough attributes and to modify the coding schemes as necessary for new applications.

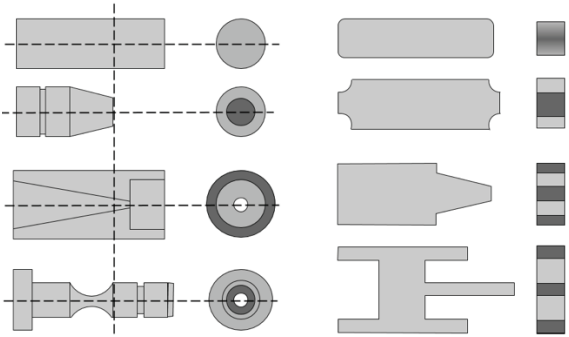
There is a variety of classification and coding systems, which were developed for parts or machined components, and many efforts have been made to show the advantages of a particular system. These systems may be completely manual, computer-assisted with an interactive expert system or mixed. The classification systems can be seen just as a tool to solve problems, which arise in the manufacturing process and which are concerned with a great variety of components. These systems can be evaluated only by the contribution to the solution of the stated problems (Opitz and Wiendahl 1971, 183). For this goal, a company can develop its proper classification system based upon either a universal system or another tailor-made system installed by outside consultants or in-house experts.

A description of the methodology for the selection and use of an appropriate classification and coding system can be met in the paper by Tatikonda and Wemmerlöv (1992), where six case studies are considered for real manufacturing plants. A short history of former classification and coding systems was also presented.

A classification system should be in line with the four principles developed by Brisch as follows (Benhabib 2003, 84):

1. *All-embracing* – an adopted classification system must be inclusive to embark all available parts within the population and be capable to classify future product features;
2. *Mutually exclusive and unambiguous* – every part must belong to one class only;
3. *Based on permanent features* – only final features are considered and not any intermediate one;
4. *Be obvious for the terminal users*.

The development of a classification system begins with a detailed review of past products and the identification of the primary attributes. A common practice is to separate first rotational and non-rotational parts, see Fig. 5-3. Once the overall classes have been determined, the next step is the examination of each class for differences. For this reason, other critical attributes, such as external features (mortises, grooves, slots), internal features (holes, threads, cavities), the ratios 'diameter-to-length' for rotational parts or ratios ‘maximal dimension-to-minimal dimension’ for non-rotational parts must be considered and systematized, etc.



a) Rotational parts b) Non-rotational parts

Fig. 5-3. Overall review of geometrical similarity. Adapted from Benhabib (2003, 85).

The classification by shape usually determines the manufacturing process, whereas the classification by function is of interest to the designer. In the case when a new part has similar functions with an already existed one, the designer does not have to duplicate it.

5.4.2 Code structures

The *code* of a part is a string of alphanumeric characters, which provides information about the part. This is in contrast to a *part number*, whose purpose is the identification of an item and not its description.

A code is created on the base of an *alphabet*. It includes numbers, letters, and other symbols, and has a determined length and a structure. The symbols are usually alphanumeric, although most systems use only numbers. The *length* is the number of the positions in the code. It should be as short as possible and have a fixed length and pattern. The *structure* is the order of the code symbols, which are used for the identification of the selected classifying attributes.

There exist various coding systems. The most used ones for the part family coding are of the next three types: polycode, monocode, and mixed structure.

a) *Polycode* (*chain-type structure*)
In this code structure, every digit represents one feature. The value of any given digit or position within the code is not related to the other digits. It can consist of various sections for different groups of attributes. It is a

position-based code, in which each position in the sequence has always the same interpretation. Each digit of the code represent an attribute as shown in Fig. 5-4.

Advantage:
- Easy to formulate.

Disadvantages:
- Very specific information is stored per digit and therefore, to get an exhaustive information about a part, a long code is required;
- Polycodes are longer than monocodes;
- The comparison of codes to seek the similarities between the parts requires a longer time than more advanced code structures.

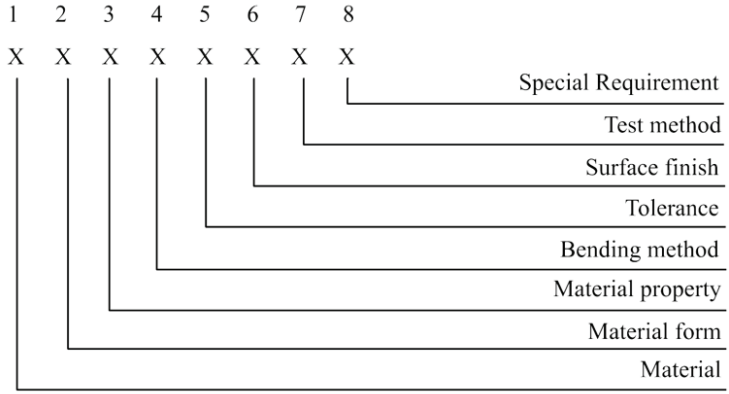


Fig. 5-4. An example of a chain-structured code.

b) *Monocode* (*hierarchical structure*)
The code is formed using a hierarchical structure, where the value of every next level is a refinement of the previous level value, and each element of the structure has a range of values.

Advantages:
- A shorter length of the code compared to the chain-type structure;
- More information can be contained in a smaller number of symbols in the code;
- The hierarchical codes are easily to interpret by a computer.

However, the *codified information must have a hierarchical structure* (Fig. 5-5). The majority of known coding systems have monocode sections.

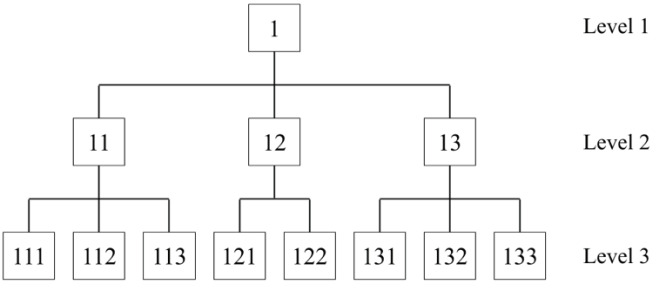


Fig. 5-5. An example of a hierarchical structure of a code.

c) *Hybrid* (*mixed structure*)
This type represents a combination of hierarchical and chain-type structures, i.e., a hybrid coding system is a combination of both polycode and monocode sections, taking advantage of both structures. Most of the available coding systems use this type of structure. Opitz' coding system has characteristics of a mixed structure. A hybrid structure combines the best qualities of a monocode (synthesis into subgroups) and the best qualities of a polycode (individual feature coding independence). The first digit in a hybrid structured coding system is usually a monocode, and the polycode makes up the rest of the digits. A formal example of a hybrid code is given in Fig. 5-6.

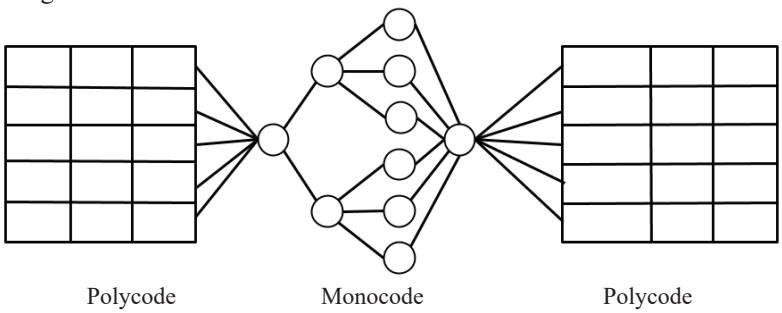


Fig. 5-6. An example of a code with a hybrid structure.

5.4.3 Taxonomy of classification and coding systems

Code-based systems were the primary tools disponed by GT in the 1960s and 1970s. This approach permitted to form the families by coding the parts with respect to their design-based features, such as shape, size, and tolerance. S. P. Mitrofanov introduced the concept of using design features for the purpose of describing and grouping similar parts in 1946. The *Brisch Birn* system, which was the first GT-oriented coding system, appeared in 1964 in UK due to E.G. Brisch and J. Gombinski for the Brisch & Partners Consulting Company's necessities. One of the most popular systems, the Opitz non-proprietary system, was developed in 1965 at the University of Aachen, West Germany, as a part of the PFA method. Later, Opitz with various coauthors have extended the idea of part classification to the *production cell*. This last concept means the grouping of machineries, which are used to process an individual part family. They have developed a comprehensive coding and classification system for work pieces.

Since these pioneering works, several coding and classification systems have been developed to facilitate the part grouping. Some known coding systems are enumerated and characterized in Tables 5-1a and 5-1b, and then shortly discussed.

5.4.4 The Opitz classification and coding system

One of the first classification systems has been proposed by Opitz and Wiendahl (1971). It served for the identification and grouping of similar work pieces by means of symbols. Although originally developed for work pieces, the methodology was of general nature. It can be adapted to structurally defined machine tools.

The basic idea for a work piece classification includes the following steps:

1. Identifying single parts;
2. Classifying and giving a survey;
3. Ordering;
4. Clearing, unifying.

The code has a hybrid structure (semi-polycode) and three sections with up to 13 digits. It starts with a design-oriented 5-digit *geometric form code*, followed by a 4-digit *supplementary code*. The supplementary code describes other information about the work piece, which was not covered by the form code. At the end, the code may be followed by a company-

specific 4-digit *secondary code* for description of production operations and sequencing (Fig. 5-7):

- The form Code 1 2 3 4 5 for the primary geometrical design attributes;
- The supplementary Code 6 7 8 9 for the manufacturing attributes;
- The secondary Code A B C D for the production operation type and sequence.

Fig. 5-8 shows the layout of the Opitz classification and coding system.

The first digit of the Opitz code is the most important digit. It characterizes the general shape of the work piece, meaning that the part class is subdivided into the rotational and the non-rotational parts. For example, the first digit numbers 0 to 2 attribute the rotational parts with a certain longitude/diameter (L/D) ratio. For the rotational pieces, a subdivision according to the length-to-diameter ratio was performed to recognize disc-type, medium and shaft-like parts. In the part class 4, denoting rotational parts with deviations, there is a subdivision into a short and a long form.

The digits 2 and 3 are subsequent to the first digit numbers. The second digit defines the external shape and its relevant form elements. The third digit describes the form and position of the main bores. The fourth digit identifies the plane machining, and the fifth digit gives the auxiliary holes, the gears and the forming.

Name	Developer	Characteristic	Description Reference
1. Brisch-Birn system	Brisch & Partners, UK-USA, Brisch and Gombinski, 1964	Monocode-oriented structure. Has 6 sections: 16 digits for the family name (class-3, subclass-4, gro-up-4, subgroup-1, series-1, subseries-3), and a 3-digit sector (Christian name).	Hallett (1964); Gombinski (1968); Patel (1991, 54)
2. OPITZ (Work piece Describing Classificatio n System)	Girardot Verlag, Essen; University of Aachen, Germany, Opitz and Wiendhal, 1965-1971,	Semi-polycode structured code with up to 13 digits. It is applicable to both machined and non-machined parts. Has 3 sections: a 5-digit geometric code; a 4-digit supplementary code; at the end it may be followed by a company-specific 4-digit secondary code for the description of the production operations and sequencing.	Opitz, Eversheim, and Wiendhal (1969); Opitz and Wiendhal (1971, 183–87)
3. SAGT (Systematic Approach to GT)	West Lafayette, IN, USA, Purdue University, Abou-Zeid, 1973	Polycode structured code. It is both design and production oriented. Has an 18-digit alphanumeric coding system for cylindrical work pieces	Abou-Zeid (1975); Offodile (1984, 40–41)
4. MICLASS (Metal Institute Classification System)	Organization for Applied Scientific Research (TNO), Netherlands, 1974	Semi-polycode structured code. Has from 12 to 30 digits in 2 sections: 12 digits are a universal code that can be applied to any part; 18 additional digits are specific for the particular company or industry.	Houtzeel and Schilperoot (1975); Hyde (1981);Houtze el (1981); Patel (1991); Elanchezhian, Selwyn, and Sundar (2008); Narayan and Mallikarjuna (2008); Hout-zeel and Schil-peroot (1975); Hyde (1981)

Table 5-1a. A survey of coding systems for manufactures with lot processing (I).

5. MultiClass	Organization for Industrial Research (OIR)	Monocode-oriented or decision-tree coding structure. The main part contains 18 digits.	GT in MFG system (n.d., ch 4, 9)
6. CODE	Jay Bergen and Associates, Manufacturing Data Systems, Inc. (MDSI), USA, 1975	Semi-polycode structured code. Has 8 hexadecimal digits 0-9 for the design of parts and A-F for manufacturing characteristics.	Patel (1991, 36) Elanchezhian, Selwyn, and Sundar (2008)
7. KK-3	The Japan Society for the Promotion of the Machining Industry (JSPMI), 1976	Chain-structured code. Has 21 decimal digits. A general-purpose classification and coding system for machined parts.	Patel (1991, 38–39); Narayan and Mallikarjuna (2008)
8. COFORM (COding FOR Machining)	Rose at Purdue University, USA	A generative process planning system with a coded input system. The code describes each surface in terms of attributes needed to select the appropriate machining processes and the related parameters (feed, speed, and depth of cut). It does not contain a description of nonmachined features, so it cannot be used for a CAD system.	Wysk (1977); Patel (1991, 45)
9. DCLASS (Design and Classification Information System)	Brigham Young University, USA; Allen K. Dell, 1985	Semi-polycode structured decision-logic handling system. Has an 8-digit code in 5 code segments (basic shape-3, form features-1, size-1, precisi-on-1, and material-2). Supports a generative process planning system	Kunzler and Nakornachal (1982); Patel (1991, 43); Narayan and Mallikarjuna (2008)

Table 5-1b. A survey of coding systems for manufactures with lot processing (II).

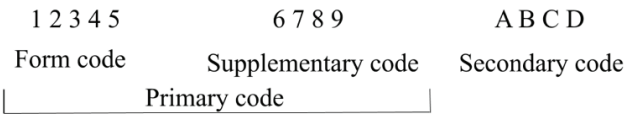


Fig. 5-7. The Opitz code structure.

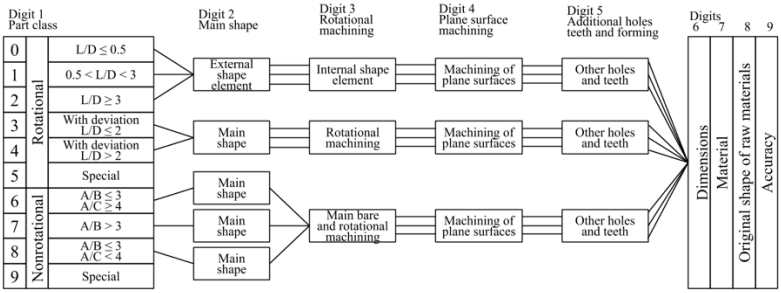


Fig. 5-8. The Opitz classification and coding system. Adapted from Opitz and Wiendahl (1971, 184).

In a similar fashion, the non-rotational parts are divided into flat, long, and cubic parts according to the ratio of length and width.

An example of the Opitz code is given in Fig. 5-9.

The Opitz coding system can also be interpreted in a matrix form (Fig. 5-10). The parts to be coded are listed on the vertical axis. Their respective design and processing features are listed on the horizontal axis. Each cell entry represents a particular attribute of a given characteristic on the horizontal axis. The string of numbers on each row represents the code number in the coding system.

Besides the form codes, additional information is necessary for a complete classification, which is collected in a so-called *additional code*. After the classification of the components in the first step, an analyst obtains a total panorama of the parts. In the next step, groups of similar parts are put into classes. In the last step, clearing and unification can take place.

Similarities between parts, captured by the GT code, are used by manufacturing engineering, purchasing, and sales. A manufacturer can drastically reduce the time and the effort spent deciding how a part should be produced if this information is available for a similar part.

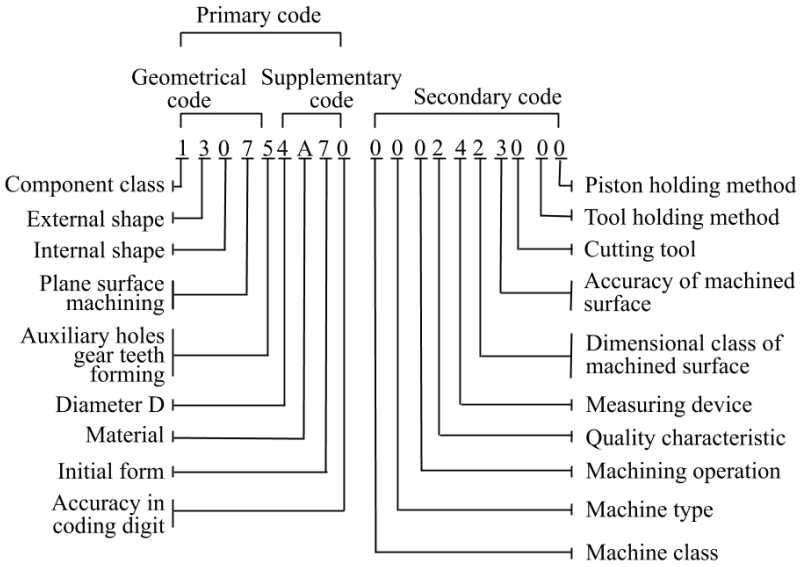


Fig. 5-9. An example of a part code of rotational shape.

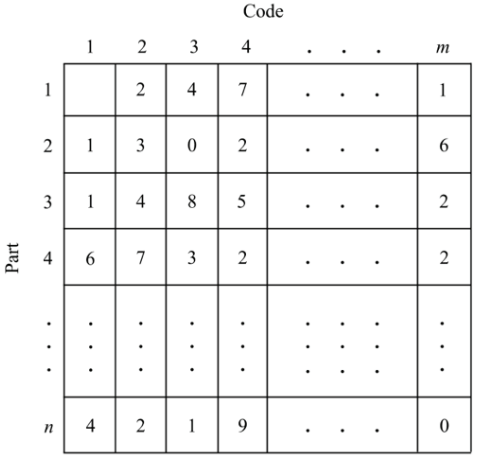


Fig. 5-10. A part/code matrix.

5.5 Production flow analysis

As it has been pointed out by Burbidge (1991, 10), the PFA consists of five sub-techniques used progressively to simplify the material flow system in an enterprise, namely:

- *Company flow analysis* (CFA) - analysis of the material flow between different factories of the company and development of a new simpler system;
- *Factory Flow Analysis* (FFA) – grouping the departments of the plant. Each of these groups completes all parts it makes. The development of a simple unidirectional flow system between these groups of departments;
- *Group analysis* (GA) – each department is divided into groups by using the matrix resolution and starting with the departments, which complete parts, avoiding backflow or crossflow (between the n groups) and buying any additional equipment;
- *Line analysis* (LA) – the flow between the machines in each group to detect the plant layout;
- *Tooling analysis* (TA) - matrix part/tools resolution to reduce the setup times.

Burbidge considered FFA and GA as the main techniques to find the machine groups and the part families.

One of the advantages of the PFA method is that a part family can be formed without using a classification and coding system, since the part families are determined by the data from the operation orders. There are also a number of drawbacks in the implementation due to the complexity of the analysis of the production data.

Some systematic methods were growing from the PFA methodology to front with the grouping problem. The most frequently used ones are based on the machine/part incidence matrix, see the reviews by Offodile, Mehrez, and Grznar (1994), Selim, Askin, and Vakharia (1998), as well as subsequent sections of this book.

A *machine/part incidence matrix* is determined and then rearranged to identify the groups of machines and parts. This matrix representation is based on cell admissibility principles.

A cell is *admissible* only if the part requires processing on the corresponding machine. Otherwise, it is not admissible. The elements a_{ij} of the matrix are defined as follows:

$$a_{ij} = \begin{cases} 1, \text{if part } j \text{ visits machine } i; \\ 0, \text{otherwise.} \end{cases} \quad (5\text{-}1)$$

This is referred to as the machine/part matrix. If '1' is written at the intersection of row i and column j in the matrix ($a_{ij} = 1$), it indicates that part j has an operation on machine i, whereas a '0' indicates that this is not the case. The order of visiting the machines by every part is not important.

The *array-based techniques* try to allocate machines to groups and parts to associated families by appropriately rearranging the order of rows and columns to find a pattern called *block-diagonal form* with the values $a_{ij} = 1$ in the incidence machine/part matrix. A value '1' outside the block is known as an *exceptional element*. A value '0' inside a cell is known as *void*. The objective is to minimize the exceptional elements and voids (Ünler and Güngör 2009, 1180).

The rearrangement process is quite subjective and difficult, especially for large matrices. An example of an incidence matrix and the corresponding block-diagonal form matrix are given in Fig. 5-11 and Fig. 5-12, respectively.

Machine No.	**Part No.**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
1			1		1				**2**
2		1	1			1			**3**
3	1			1			1	1	**4**
4		1	1		1	1			**4**
5	1			1			1	1	**4**
	2	**2**	**3**	**2**	**2**	**2**	**2**	**2**	

Fig. 5-11. An example of the incidence matrix resolution in the PFA approach: initial matrix.

Column 1 (Fig. 5-11) shows that part 1 visits two machines, 3 and 5. Furthermore, row 2 displays that machine 2 processes three parts, namely, 2, 3, and 6. The last column and last row contain a complementary information about the total value of each machine/part, respectively. Two part families and two machine cells are visible in the rearranged matrix (Fig. 5-12). These machine cells are: MC1={3,5} and MC2={1,2,4}. Consecutively, the two part families are: PF1={1,4,7,8} and PF2={2,3,5,6}.

The PFA theory has been extended by other authors (King 1980; King and Nakornchai 1982).

Ei-Essawy and Torrance (1972) proposed a technique similar to PFA called the *Component Flow Analysis* (CFA). The methodology of the CFA has differences with Burbidge's PFA procedure since the latter one first partitions the problem, whereas the former does not.

MC		PF1				PF2				
		4	**1**	**8**	**7**	**2**	**6**	**3**	**5**	
MC1	**5**	1	1	1	1					**4**
	3	1	1	1	1					**4**
MC2	**4**					1	1	1	1	**4**
	2					1	1	1		**3**
	1							1	1	**2**
		2	**2**	**2**	**2**	**2**	**2**	**3**	**2**	

Fig. 5-12. The incidence matrix resolution in the PFA approach: rearranged matrix.

5.6 Similarity coefficient

The *similarity coefficient* is a principal measure, which is employed in GT applications for the formation of the part families and the machine cells. Compared to the other methods, similarity-based methods incorporate more flexibility into the machine-component grouping process and more easily lend itself to the computer application.

The similarity coefficients can be classified into two distinct groups:

- Problem-oriented;
- General-purpose.

General-purpose similarity coefficients are widely used in different disciplines, such as biology, sociology, medicine, economics, engineering, etc., in numerical studies to build up taxonomic classifications.

Taxonomy is initially a science of a biological classification of objects based on their possession or lack of relevant characteristics. When these characteristics can be expressed numerically, the objects can be classified by a numerical taxonomy (Carrie 1973). The *problem-oriented similarity*

was designed to solve a specific problem, particularly, the cell formation problem.

In GT, the numerical taxonomy is a method of analysis, which involves three stages:

1. Prepare the data matrix;
2. Compute the similarity coefficient matrix;
3. Perform a cluster analysis.

After introducing the PFA methodology by Prof. J. L. Burbidge (1963), there was a lack of a quick, simple, and cheap method for finding the families and cells. The similarity coefficient approach filled this gap.

Below, this subject and relevant methodologies are discussed in more detail.

5.6.1 Definition and properties

A similarity coefficient measures the resemblance between two individuals based on either one or both of two logically distinct kinds of information pertaining to *K* attributes and allowing for possible missing information (Gower 1971, 858–59). Two individuals *i* and *j* may be compared on an attribute *k*. A score $s_{ijk} = 0$ is assigned when *i* and *j* are considered to be different, and a positive fraction $s_{ijk} = 1$ when both individuals have some degree of agreement or similarity. If $s_{ijk} = \{0,1\}$, the attribute *k* is either present or absent in both individuals. It indicates a *dichotomous* character of the attributes, where the presence of the attribute *k* is denoted by '+' and its absence by '-'.

There are many ways of calculating s_{ijk}, some of them are described below. Sometimes no comparison is possible because the information is missing or in the case of dichotomous variables, an attribute is non-existent in both *i* and *j*. The possibility of making a comparison is represented by the value $\delta_{ijk} = 1$ if the attribute *k* allows to compare the individuals *i* and *j*, and $\delta_{ijk} = 0$ otherwise. If $\delta_{ijk} = 0$, the score s_{ijk} is unknown and set to be equal to zero. The similarity between *i* and *j* is defined as the average score taken over all possible comparisons:

$$S_{ij} = \frac{\sum_{k=1}^{K} s_{ijk}}{\sum_{k=1}^{K} \delta_{ijk}}. \qquad (5\text{-}2)$$

Obviously, if $\delta_{ijk} = 0, \forall k, k = 1, \ldots, K$, the similarity S_{ij} is undefined. If all comparisons are possible, $\sum_{k=1}^{K} \delta_{ijk} = K$, where K is the total number of attributes. Otherwise, it is the number of characters over which the comparison is made (Fig. 5-13).

	Values of character k			
Individual i	+	+	-	-
Individual j	+	-	+	-
S_{ijk}	1	0	0	0
δ_{ijk}	1	1	1	0

Fig. 5-13. Comparison of dichotomous components.

A weighted similarity coefficient can be defined by the following formula:

$$S_{ij} = \frac{\sum_{k=1}^{K} s_{ijk} \varpi_k}{\sum_{k=1}^{K} \delta_{ijk} \varpi_k}, \tag{5-3}$$

where ϖ_k is a constant weight of the attribute k.

Three types of similarity coefficients can be identified in the literature:

1. Association;
2. Correlation;
3. Distance.

The *association type* of a similarity coefficient measures the value of similarity between pairs over a given data set using binary codes. A similarity coefficient by association has been defined in several ways (Rajagopalan and Batra 1975, 571; King and Nakornchai 1982, 123). For the *correlation type* of similarity, the mutual relationship between pairs of data to be grouped is established by the correlation equations (McAuley 1972, 54; Carrie 1973, 404; Hachicha, Masmoudi, and Haddar 2008, 1158–60). A *distance type* in a similarity coefficient is used to measure the dispersity between two points within a data set. In most cases, this coefficient

is defined by a distance formula (Chandrasekharan and Rajagopalan 1986, 453; Andrés et al. 2005, 277–78; Shafer and Rogers 1993a; 1993b).

The main difference between an association-type similarity coefficient and distance type coefficient measures is that the first one is normalized to 1. For GT problems, an association-type similarity coefficient is preferred to a distance measure coefficient. The distance coefficients tend to favor dissimilar pairs of data more than similar pairs, since the coefficient becomes larger for the more distant (dissimilar) pairs. However, the ratio of the total number of observed matches to the total number of possible matches received most acceptance.

5.6.2 Part grouping

Carrie (1973, 404) suggested the following definition of a similarity coefficient for the *part grouping* problem in GT:

$$s_{ij} = \frac{M(\beta_i \cap \beta_j)}{M(\beta_i \cup \beta_j)}, \tag{5-4}$$

where

s_{ij} - similarity coefficient between the parts *i* and *j*;
M - number of discrete variables in the data set;
β_i - set of operations required for part *i*;
β_j - set of operations required for part *j*.

This definition implies that s_{ij} has always a positive value between 0 for the maximum dissimilarity and 1 for the maximum similarity. This means:

1. $0 \leq s_{ij} \leq 1$;
2. $s_{ij} = 1$ implies a maximal similarity;
3. $s_{ij} = 0$ implies a minimal similarity;
4. $s_{ij} = s_{ji}$ implies symmetrical properties.

In some cluster analysis problems, the *dissimilarity coefficient* is sometimes used. Dissimilarity and similarity measures are complementary. Thus, the dissimilarity between the two parts *i* and *j* is:

$$d_{ij} = s_{ij} - 1. \tag{5-5}$$

5.6.3 Shape-based similarity for the part family formation

Offodile and Grznar (1997) introduced a similarity coefficient-based methodology for a part coding and classification system. A mathematical formulation and a grouping algorithm were proposed with the objective to maximize the sum of the similarity measures.

The similarity between any two parts is as follows:

$$S_{ij} = \frac{\sum_{k=1}^{K} s_{ijk}}{\sum_{k=1}^{K} \delta_{ijk}}, \tag{5-6}$$

$$s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{R_k}; \tag{5-7}$$

$$\delta_{ijk} = \begin{cases} 1, \text{if parts } i \text{ and } j \text{ are comparable for attribute } k; \\ 0, \text{ otherwise.} \end{cases} \tag{5-8}$$

Where

S_{ij} - similarity between part *i* and part *j*;
s_{ijk} - score between part *i* and part *j* for the attribute *k*;
x_{ik} - weight assigned to part *i* for the attribute *k*;
x_{jk} - weight assigned to part *j* for the attribute *k*;
R_k - range of the attribute *k* taken over the population of parts;
K - number of attributes.

Following such a definition, if $x_{ik} = x_{jk}$, k =1,…,K, then $s_{ijk} = 1$ and $S_{ij} = 1$. This implies a maximal similarity. In the opposite case, $x_{ik} \neq x_{jk}$, and either [$x_{ik} = 0$ or $x_{ik} = R_k$] and [$x_{jk} = R_k$ or $x_{jk} = 0$], k =1,…, K, for a minimal similarity.

The case when $\delta_{ijk} = 0$ implies that there is no basis to compare the parts *i* and *j* for the attribute *k*. Therefore, the score s_{ijk} between part *i* and part *j* is unknown and can be set equal to zero.

If $\delta_{ijk} = 0$, k =1,…, K, then the similarity S_{ij} is undefined, but it can also be set equal to zero for convenience. Otherwise, we have $0 \leq S_{ij} \leq 1$, $\forall i \neq j$, i, j =1,…, n; and $S_{ij} = 0$, $\forall i = j$, i, j =1,…, n.

However, it is unlikely that $\delta_{ijk} = 0$, $\forall k$, k =1,…, K. This is because in practice, the same coding system is often used to code all parts in an installation. Therefore, the parts will share a common database. However, such a case implies that the parts have nothing in common, and there is no basis for GT. Nevertheless, such a situation is very rarely occurring in practice.

The last term in Formula (5-7) is a dissimilarity measure, which displays the relative difference between the codes for part *i* and any other part *j*, compared on the base of the attribute *k*. Therefore, the term shows how dissimilar parts *i* and *j* are, and whether these parts can be grouped together.

5.6.4 The Jaccard similarity coefficient for the cell formation

McAuley (1972, 54) was the first researcher to apply the Jaccard similarity coefficient to the cell formation problem. McAuley calculated the Jaccard similarity coefficient for each pair of machines. These similarity coefficients give a measure to calculate how similar the two machines in the pair are, taking into account the number of parts, which visit both machines, and the number of parts, which visit one of the two machines only. An example of the *incidence machine/part matrix* is given in Fig. 5-14. In this matrix, a cell (*i*, *j*) takes the value '1', if part *j* visits machine *i*, and '0' otherwise.

Machine No. *i*	**Part No. *j***			
	1	**2**	**3**	**4**
1	1	1		1
2	1	1		
3	1		1	1

Fig. 5-14. Machine/part incidence matrix.

The basic arrangement of the *pairs of machines* for the computation of the similarity coefficient is a 2×2 matrix as it is shown in Fig. 5-15. The elements of the matrix are as follows. The upper case (capital letter) of a subscript indicates a 1-state, i.e., the part visits the machine. The lower case of a subscript denotes a 0-state when a part does not visit the machine. Conversely, if the machines *j* and *k* are considered, N_{JK} is the number of parts visiting both machines; N_{jk} is the number of parts, which visit none of the machines; N_{Jk} is the number of parts, which visit machine *J* and not machine *k*; N_K is the number of parts, which do not visit machine *j* but visit machine *k*.

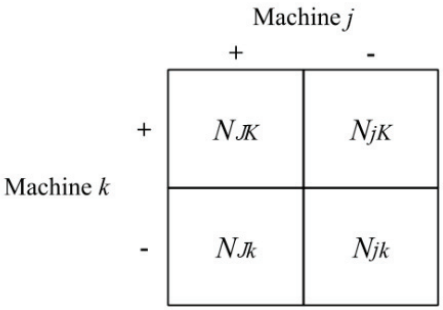


Fig. 5-15. Matrix of the two-machine relationship.

The similarity coefficient for the machines *j* and *k* is given as the number of components N_{JK}, which visit both machines, divided by the sum of the same number N_{JK} and the number U of components, which visit one of the machines, as follows by the formula:

$$S_{jk} = \frac{N_{JK}}{N_{JK}+U}; \tag{5-9}$$

where $U = N_{Jk} + N_{jK}$.

According to this formula, the possible similarity coefficients are:

$$S_{1,2} = \frac{2}{2+1} = 0.67;$$
$$S_{1,3} = \frac{2}{2+(1+1)} = 0.5;$$
$$S_{2,3} = \frac{1}{1+(1+2)} = 0.25.$$

For *m* machines, the number of coefficients to be calculated is given by the number of combinations of pairs of machines:

$$\binom{m}{2} = \frac{(m-1)m}{2}. \tag{5-10}$$

The calculations can be reduced because machines *j* and *k* with at least one part in common have non-zero values. In addition, since $S_{jk} = S_{kj}$, it is only necessary to calculate a triangular matrix of the similarities with the leading diagonal omitted.

The grouping of machines can be justified by a *dendrogram* (Fig. 5-16). In such a dendrogram, the abscissa-axis has no special meaning. It is used only for the separation of the involved machines, while the ordinate shows

the similarity coefficient, commonly in percent. The points of junction between the stems mean that the resemblance between the two stems is at the similarity coefficient level, which is shown on the ordinate.

5.6.5 Modified McAuley similarity coefficient for the cell design

A variant of the McAuley similarity coefficient was proposed by Rajagopalan and Batra (1975, 571). It was defined by the formula:

$$s_{ik} = \frac{x_{ik}}{x_{ii}+x_{kk}-x_{ik}}, \quad (5\text{-}11)$$

where

s_{ik} - similarity coefficient between machines *i* and *k*;
x_{ik} - number of components using both machines *i* and *k*;
x_{ii} - number of components using only machine *i*;
x_{kk} - number of components using only machine *k*.

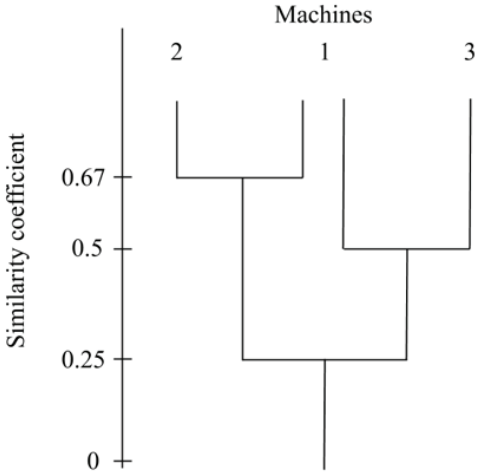


Fig. 5-16. A dendrogram for grouping machines on the base of the similarity level.

In this definition, s_{ik} lies between 0 and 1 for a maximum dissimilarity and similarity, respectively.

Once the similarities between the different pairs of machines are calculated, the machine cells are then formed. Machines can be combined into a group if and only if the similarity coefficients are greater than a

threshold value *T*, which is established by an expert and verified by a dendrogram.

5.6.6 Similarity-based distance between parts

Part families, determined by common manufacturing and design attributes, are not necessarily connected. Andres et al. (2005, 277–78) proposed an approach to calculate a coefficient of similarity between parts, which is based on the production data. This coefficient is different from the traditional one, which is based on the part attributes. The definition of this coefficient has been established following the subsequent three statements:

- S_1: Two parts (A and B) belong to the same family when the setup time from A to B (B to A) is small.
- S_2: Two parts (A and B) belong to the same family when the setup time from producing any part (except A or B) is similar.
- S_3: Inversely, two parts (A and B) belong to the same family when the setup time from producing A or B to any other part is similar.

These statements are illustrated in Fig. 5-17.

The statement S_1 is based on a low difficulty of the adjustment of the machines to change the layout from A to B (B to A). The statement S_2 is based on the fact that both parts can be treated as a family, i.e., the setups from any part to A or B are similar. The statement S_3 is the same but considers the inverse case of statement S_2.

These three statements can be summarized by a measure of distance, which is a *similarity coefficient*:

$$d(i,r) = \sqrt{S_1 + S_2 + S_3}\,, \forall i, r \neq k, \tag{5-12}$$

where

$$S1 = S(i,r)^2 + S(r,i)^2;$$
$$S2 = \sum_{k=1}^{p} \big(S(i,k) - S(r,k)\big)^2;$$
$$S3 = \sum_{k=1}^{p} \big(S(i,k) - S(r,k)\big)^2.$$

The authors implemented these metrics in a methodology to estimate the setup times and to apply GT in the tile industry. The resulting part families were used to develop production sequences requiring shorter setup times.

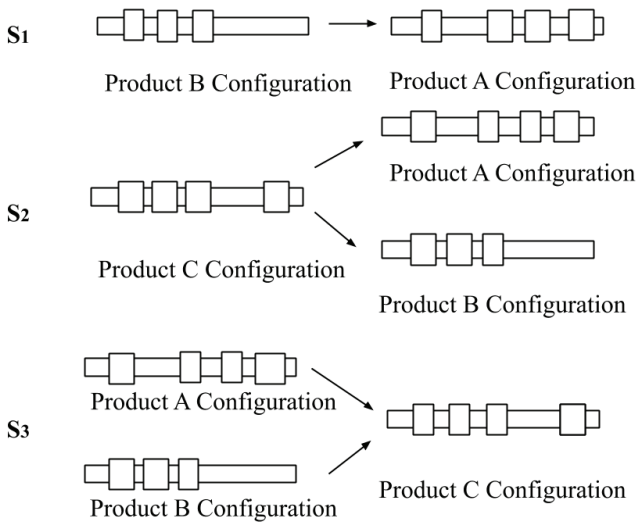


Fig. 5-17. Graphical interpretation of grouping criteria. Adapted from Andrés et al. (2005, 278).

5.7 Cluster analysis

A cluster analysis has been utilized to form groups of entities in computer science and engineering. It was employed for pattern recognition and artificial intelligence studies as a method for uncovering the fine underlining structure of a given data set, see McAuley (1972, 54); Offodile (1984, 47–49). In GT, a cluster analysis was used for grouping parts and machines to form separate machine cells in manufacturing systems.

5.7.1 Taxonomy of clustering approaches

Typically, the initial machine/part incidence matrix $[a_{ij}]$ does not give an immediate decision about the cells of machines and families of parts. A clustering algorithm allows transforming the initial incidence matrix into a structured form, a block-diagonal form, if possible. A cluster analysis is one of the most frequently applied mathematical tools in GT. There are three basic formulations of clustering methods:

1. Graphical formulation;
2. Matrix formulation;
3. Mathematical programming formulation.

A *graphical formulation* is important for the initial study of the problem as well as for an analysis of the results. A *matrix formulation* technique for clustering the machines, called Single Linkage Cluster Analysis (SLCA), was proposed by McAuley (1972). This approach was initially a part of the Numerical Taxonomy, which was developed in the 1960s to be used in the fields of entomology and microbiology as well as paleontology. McAuley applied this technique to the cell formation in GT. Similarity coefficients were extensively used for a cluster analysis to summarize the relationship between pairwise combinations of the entities in a given data set.

Finally, a *mathematical programming* formulation permitted the formalization of the problem for its efficient computational solution. The used models are integer programming, linear programming, and dynamic programming. The most usual model is a programming formulation, which is more convenient for dichotomic binary variables.

Considering the final objectives of clustering, the techniques applied for the cell formation fall into one of the three following categories:

- Grouping part families or machine cells only;
- Forming part families and then machine cells;
- Forming part families and machine cells simultaneously.

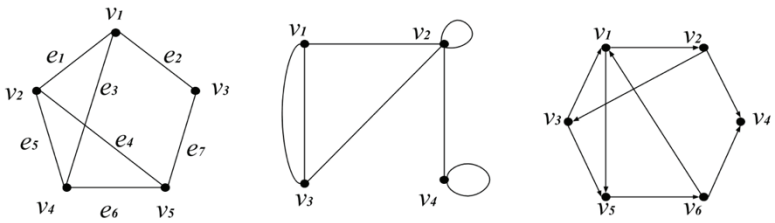
Many computational techniques have been developed to solve the clustering problem, which is of NP-hard nature. The main approaches are described below.

5.7.2 Graphical formulation

A graph representation, which exploits the similarities Part-Part and Machine-Machine, gives some useful tools for part families grouping and machine clustering.

A graph G consists of a set V of *vertices* and a set E of *edges*. An edge e is associated with a unique pair of vertices v and u. Consecutively, $e = (v_1, v_2)$ or $e = (v_2, v_1)$ denotes an edge in an *undirected graph*. If a non-directed graph admits the existence of loops, such a graph is called a graph with *loops*. If the existence of more than one edge between two vertices is allowed in an undirected graph, such a graph is called a *multigraph*.

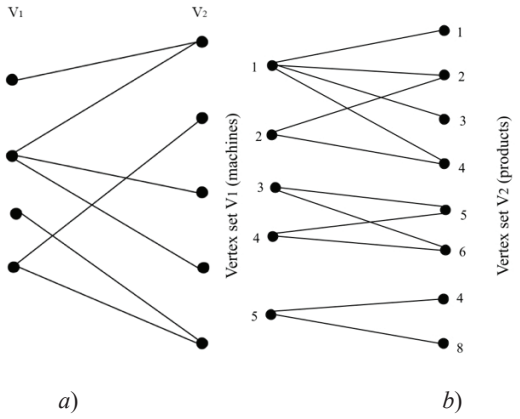
A *directed graph* (or *digraph*) G consists of a set V of vertices and a set E of edges such that every edge $e \in E$ is associated with an ordered pair of vertices. If such an ordered pair is unique, then $e = (v_1, v_2)$, see the graph examples in Fig. 5-18.



a) b) c)

Fig. 5-18. Examples of graphs: a) undirected graph; b) undirected multigraph with loops; c) directed graph.

A graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $E' \subseteq E$, $V' \subseteq V$, and the edges of E' are incident only with the vertices of V'. An undirected graph is *bipartite* if the set of its vertices V is separated into two mutually exclusive subsets V_1 and V_2 in such a manner that every edge joins a vertex in V_1 with a vertex in V_2 (Fig. 5-19a). An undirected graph is *connected* if there exists a path between any two vertices, and it is *disconnected* otherwise. An undirected graph may be discomposed into a number of separated *connected components* (Fig. 5-19b).



a) *b*)

Fig. 5-19 Examples: *a*) bipartite graph; *b*) disconnected bipartite graph.

The most usual and convenient representation of the results of a cluster analysis is by a *dendrogram*. The machine/part matrix (preferably a rearranged matrix) can also be illustrated by a *network model* using a *bipartite graph*.

King and Nakornchai (1982, 121) suggested that a GT problem should be represented by a bipartite graph to derive some important decisions. The vertices on the left-hand side of such a graph form the machine numbers and on the right-hand side, the part numbers are written. An edge connects a left/right pair of vertices if the corresponding machine processes the corresponding part. The machines and parts are the two vertex sets. An edge connects a machine with a part, which visits that machine. Therefore, every machine is connected with all parts, which it is able to process. From the other side, every part is connected with all machines, which are able to process this part. Obviously, such a graph can have no isolated vertices both in the set of machines as well in the set of parts.

The machine/part clustering problem results in the problem of the decomposition of a bipartite graph into a set of disconnected components. A detailed analysis of bipartite graphs from the GT viewpoint has been given by Chandrasekharan and Rajagopalan (1986, 453–54).

A grouping problem can also be illustrated by a multigraph. A vertex in such a multigraph corresponds to a part. Two vertices are connected by an edge if the same machine can process the corresponding parts. A decomposition of a multigraph into a set of disconnected components gives a solution for grouping the parts into families.

A similar graph representation can be also used for detecting a machine cell. In such a multigraph, a vertex corresponds to a machine. Two vertices are connected by an edge if the corresponding machines are able to process the same part. The solution of the clustering problem is to detect disconnected components, which represent machine cells.

Illustrative examples of the application of the graphical formulation are given in Sections 5.7.1-5.7.2.

5.7.3 Matrix formulation

According to this method, the machines, which have mutually the highest possible similarity coefficient, are first combined in a cluster. Then step-by-step the machines by other levels of similarity are taken. The criterion of a single linkage is used for the admission of a machine, or a group of machines, into another group. Analogously, any pair of machines, one in each cluster, with the critical level is linked with corresponding cluster. The disadvantage of this method is that a situation may occur when two clusters

are linked on the basis of a single bond, while many of the members of the two clusters may be quite far removed from each other considering their similarity.

Typically, when the initial machine/part incidence matrix $[a_{ik}]$ is constructed, the clusters of machines and parts are not visible. A perfect cell formation is achieved when the cells on the diagonal contain '1'. The clustering concept for grouping parts and machines is illustrated in the following two examples.

Example 5.1.
A job shop with 6 machines producing 6 parts is considered. The machine/part incidence matrix is given in Fig. 5-20. Column *j* of this matrix, $j = 1,\ldots,6$, displays the machines, which are required to produce part *j*. These machines are marked by '1'. On the other side, the row *i* indicates the parts, which visit the machine *i*, $i = 1,\ldots,5$. Therefore, column 2 shows that part 2 visits the machines 1, 3, and 5. Row 2 demonstrates that machine 2 processes parts 1 and 3. The sequence, in which the parts visit the machines, is not taken into account since it does not affect the groupings. Nevertheless, it can influence the layout of the machines within each group and the positioning of the groups in relation to each other.

In Fig. 5-21, a matrix with a *block-diagonal* pattern obtained by a manual grouping of the machines is given. It is received by moving the rows and columns until they reach a quadratic pattern. With this, one can conclude that the five machines of the plant should be grouped into two cells. More detailed information is received by means of the similarity coefficients for all pairs of machines. It is evident that only non-empty cells give non-zero similarity coefficients.

Machine No.	**Part No.**					
	1	**2**	**3**	**4**	**5**	**6**
1		1		1	1	
2	1		1			
3		1		1		
4			1			1
5		1				

Fig. 5-20. A machine/part incidence matrix.

MC		PF1			PF2		
		4	5	2	6	3	1
MC1	1	1	1	1			
	3	1		1			
	5			1			
MC2	4				1	1	
	2					1	1

Fig. 5-21. A machine/part matrix after rearranging the rows and columns.

The possible Jaccard-type non-zero similarity coefficients are given for all pairs of machines as follows:

$$S_{1,3} = \frac{2}{2+1} = 0.67,$$
$$S_{1,5} = \frac{1}{1+2} = 0.33,$$
$$S_{3,5} = \frac{1}{1+(1+2)} = 0.25,$$
$$S_{2,4} = \frac{2}{2+1} = 0.67.$$

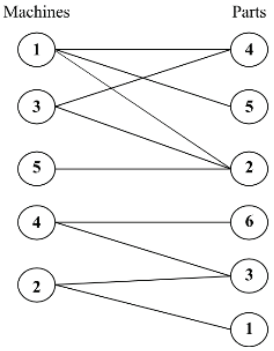


Fig. 5-22. A bipartite graph.

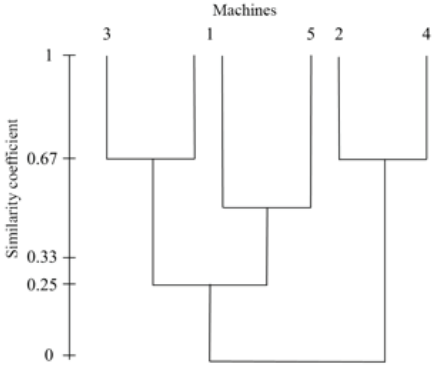


Fig. 5-23. A dendrogram.

The result of the cluster analysis on the base of the similarity coefficients between the pairs of machines is shown in Fig. 5-22 and Fig. 5-23. A dendrogram illustrates the level of similarity for these pairs. Machines 1 and 3 should be linked due to a high similarity of 67%. If a plant designer requires a similarity no less than 33%, machines 3 and 5 cannot be grouped.

However, if 25% suffices, the machines 1, 3, and 5 can be included into one group. The second group is formed by the machines with 67% of similarity and without any intersection with the first group.

Example 5-2.
A part/machine incidence matrix is given in Fig. 5-24. After rearranging, the parts were grouped into three PFs and the machines into three MCs. The rearranged matrix is given in Fig. 5-25. The similarity coefficients are calculated using the data of the incidence matrix from Fig. 5-25. These coefficients are represented in a machine/machine matrix, formed according to the grouping of machines into cells (Fig. 5-26). The machines form three groups: {1,4,7}, {2,3,8,10}, and {5,6,9}. These groups process the part families PF1, PF2, and PF3, respectively. One can see that this matrix is symmetric with respect to the main diagonal and has no diagonal elements. The results of grouping can also be visualized as a network by a bipartite undirected graph, which is decomposed into three connected components corresponding to the machine/part grouping (Fig. 5-27). The dendrogram in Fig. 5-28 shows that MC1 is formed by 17% of similarity, MC2 has 33%, and MC3 has 40% of similarity, respectively, between the machines, which constitute the cells.

The matrix formulation is a simple and useful method but has two big disadvantages:

1. For matrices, which have a large number of rows and columns, it is difficult to represent and visualize clusters;
2. In many cases, it is difficult to obtain a diagonal or close to diagonal structure of the clustered matrix, even for low dimensions.

An integer programming formulation of the clustering problem, known as the *p*-median mode, does not have these disadvantages.

Machine No.	Part No.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1				1		1		1			1				
2	1		1		1										
3	1		1		1				1						
4				1		1							1		
5		1					1					1		1	
6		1					1					1		1	
7				1				1			1				
8	1		1						1						
9		1										1			1
10	1		1		1					1					

Fig. 5-24. Initial machine/part incidence matrix for Example 5-2.

MC		PF1					PF2					PF3				
		4	6	8	11	13	1	3	5	9	10	12	2	7	14	15
MC1	1	1	1	1	1											
	4	1	1			1										
	7	1		1	1											
MC2	2						1	1	1							
	3						1	1	1	1						
	8						1	1		1						
	10						1	1	1		1					
MC3	5											1	1	1	1	
	6											1	1	1	1	
	9											1	1			1

Fig. 5-25. Block-diagonal structure of the machine/part incidence matrix for Example 5-2.

MC		MC1			MC2				MC3		
		1	**4**	**7**	**2**	**3**	**8**	**10**	**5**	**6**	**9**
MC1	**1**	-	0.17	0.75							
	4	0.17	-	0.2							
	7	0.75	0.2	-							
MC2	**2**				-	0.75	0.33	0.75			
	3				0.75	-	0.75	0.6			
	8				0.33	0.75	-	0.4			
	10				0.75	0.6	0.4	-			
MC3	**5**								-	1	0.4
	6								1	-	0.4
	9								0.4	0.4	-

Fig. 5-26. Diagonal machine/machine matrix of the similarity coefficients for MCs.

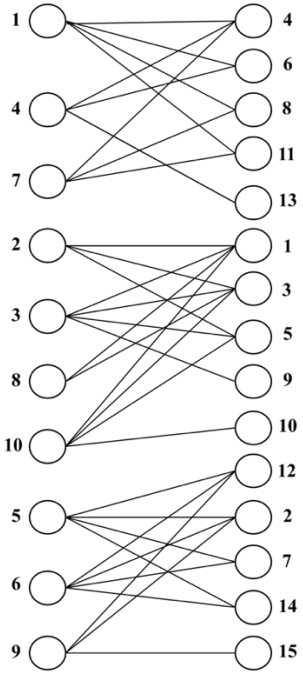


Fig. 5-27. A bipartite graph for Example 5-2. The initial graph is discomposed into three disconnected components (MCs).

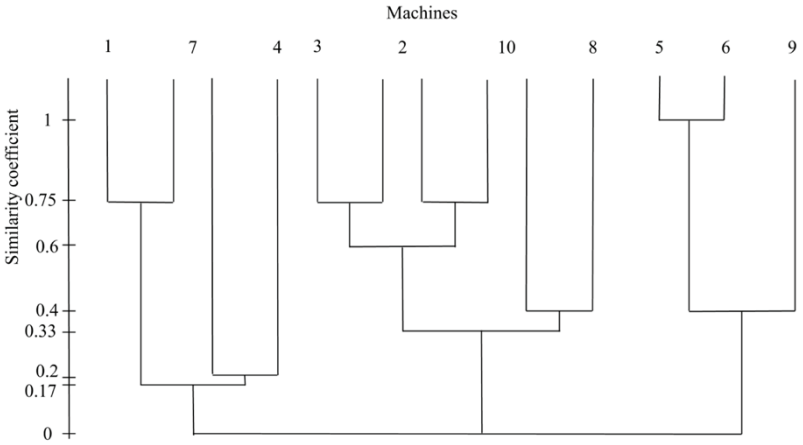


Fig. 5-28. A dendrogram for Example 5-2

5.7.4 Integer programming formulation

The *p-median* model was introduced by Kusiak (1987) to avoid the first disadvantage of the matrix formulation. The author noted that the only difference between the matrix formulation and the corresponding *p-median* formulation is that in the first case, the number of part families is determined *a posteriori* while in the second one, it is determined *a priori*.

The following notations and definitions are used.

Let a part/machine incidence matrix be given by the *p-median* model, where *p* part families and *p* machine cells are formed.

The notations used are:

n - number of parts;
m - number of machines;
p - required number of part families.

For the matrix representation, the combinations of two 0-1 vectors are used:

$$P_i = [a_{i1}, a_{i2}, \dots, a_{ik}, \dots, a_{im}]^T, \quad (5\text{-}13)$$

$$P_j = [a_{j1}, a_{j2}, \dots, a_{jk}, \dots, a_{jm}]^T, \quad (5\text{-}14)$$

s_{ij} - similarity between two parts, say, between part *i* and part *j*; $s_{ij} \geq 0$, $\forall$ *i*, *j* =1,…, *n*; and $s_{jj} = 0$, $\forall$ *j*=1,…,*n*;

The similarity is defined according to the formula:

$$s_{ij} = \sum_{k=1}^{m} \delta(a_{ik}, a_{jk}), \quad (5\text{-}15)$$

where

$$\delta(a_{ik}, a_{jk}) = \begin{cases} 1 \text{ if } a_{ik} = a_{jk} \\ 0 \text{ otherwise} \end{cases}, \quad (5\text{-}16)$$

$$x_{ij} = \begin{cases} 1 \text{ if part } i \text{ belongs to part family } j \\ 0 \text{ otherwise} \end{cases}, \quad (5\text{-}17)$$

An element $x_{jj} = 1$ forms a part family together with the elements of column j, for which $x_{ij} = 1$, $i = 1,\ldots, n$. The case when $x_{jj} = 0$ means that family j is not formed. So, $\sum_{j=1}^{n} x_{jj} = p$.

The *p-median* model is as follows.

The objective is to maximize the total sum of the similarities:

$$z = \sum_{i=1}^{n} \sum_{j=1}^{n} s_{ij} x_{ij} \rightarrow \max \quad (5\text{-}18)$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1, \forall\ i = 1,\ldots, n, \quad (5\text{-}19)$$

$$\sum_{j=1}^{n} x_{jj} = p, \quad (5\text{-}20)$$

$$x_{ij} = \{0,1\}, \forall\ i, j = 1,\ldots, n \quad (5\text{-}21)$$

$$x_{ij} \leq x_{jj}, \forall\ i, j = 1,\ldots, n \quad (5\text{-}22)$$

Constraint (5-19) makes sure that each part belongs to exactly one family. Constraint (5-20) defines the required number of part families. Constraint (5-21) ensures the integrality of the binomial components. Constraint (5-22) guarantees that part i belongs to part family j only when this part family is formed.

An illustrative example is described below.

Example 5-3.
Let the incidence matrix be as follows.

$$[a_{ik}] = \begin{matrix} \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} \begin{matrix} \text{Part No.} \\ 1 \quad 2 \quad 3 \quad 4 \quad 5 \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \text{Machine No.}$$

Solving the clustering problem by the block-diagonal method, the obtained structure is as follows (Fig. 5-29):

MC		PF1		PF2		
		1	**3**	**2**	**4**	**5**
MC1	**2**	1	1			
	4	1	1			
MC2	**1**			1	1	1
	3			1	1	

Fig. 5-29. A machine/part matrix after rearranging the rows and columns.

After the transposition of the matrix $[a_{ik}]$ according to representations (5-13) and (5-14), and using expressions (5-15) and (5-16) of the *p*-median model, the similarities s_{ij}, $i, j = 1,\ldots,5$, are calculated comparing every pair of family parts. The resulting matrix $[s_{ij}]$ has the dimension $n \times n$. The first row of the matrix $[s_{ij}]$ for the given matrix $[a_{ij}]$ is as follows:

$s_{11} = 0;$

$s_{12} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} = 0;$

$s_{13} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} = 4;$

$s_{14} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} = 0;$

$s_{15} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} = 1,$ etc.

Using the notations of the model for every pair P_i and P_j:

$s_{11} = s_{22} = s_{33} = s_{44} = s_{55} = 0;$

$s_{12} = \delta(a_{11}, a_{21}) + \delta(a_{12}, a_{22}) + \delta(a_{13}, a_{23}) + \delta(a_{14}, a_{24}) = 0 + 0 + 0 + 0 = 0;$

$s_{13} = \delta(a_{11}, a_{31}) + \delta(a_{12}, a_{32}) + \delta(a_{13}, a_{33}) + \delta(a_{14}, a_{34}) = 1 + 1 + 1 + 1 = 4;$

$s_{14} = \delta(a_{11}, a_{41}) + \delta(a_{12}, a_{42}) + \delta(a_{13}, a_{43}) + \delta(a_{14}, a_{44}) = 0 + 0 + 0 + 0 = 0;$

$s_{15} = \delta(a_{11}, a_{51}) + \delta(a_{12}, a_{52}) + \delta(a_{13}, a_{53}) + \delta(a_{14}, a_{54}) = 0 + 0 + 1 + 0 = 1;$
etc.

The complete similarity matrix is:

$$[s_{ij}] = \begin{bmatrix} 0 & 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 4 & 3 \\ 4 & 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 3 \\ 1 & 3 & 1 & 3 & 0 \end{bmatrix}.$$

The matrix of the unknown binomial components is:

$$[x_{ij}] = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{42} & x_{x_{44}} & x_{45} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \end{bmatrix}.$$

Let the number of family groups be $p = 2$. For the given similarity matrix, the objective function z is:

$z = \sum_{i=1}^{n} \sum_{j=1}^{n} s_{ij} x_{ij} =$
$(4x_{13} + x_{15}) + (4x_{24} + 3x_{25}) + (4x_{31} + x_{35}) + (4x_{42} + 3x_{45}) +$
$(x_{51} + 3x_{52} + x_{53} + 3x_{54}),$

with the constraints:

$\sum_{j=1}^{5} x_{ij} = 1$, $\forall i = 1,\ldots, 5$, and
$\sum_{j=1}^{5} x_{jj} = 2$.

The solution is obtained by a linear programming algorithm for the model with the incidence matrix $[a_{ij}]$. It is given by the following matrix:

$$[x_{ij}^*] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

The maximal value of the objective function is:

$$z^* = 4 + 4 + 3 = 11.$$

The following two part families are formed on the base of Definition (5-17) of x_{ij} and the given value of p:

PF1 = {1,3};
PF2 = {2,4,5}.

The corresponding two machine cells are found by analyzing the incidence matrix $[a_{ij}]$:

MC1: {2,4};
MC2: {1,3}.

In addition, the machines can be clustered applying the *similarity approach*. The similarity matrix $[s_{kl}^m]$ for the machines is:

$$[s_{kl}^m] = \begin{bmatrix} 0 & 0 & 3 & 0 \\ 0 & 0 & 1 & 5 \\ 3 & 1 & 0 & 1 \\ 0 & 5 & 3 & 1 \end{bmatrix}.$$

Here k, $l = 1,\ldots, m$, and the decomposition of the machine cells into groups is evident.

It can be easily verified that the solution by the p-median model is identical to the one obtained by the block-diagonal method (Fig. 5-29).

However, in contrast to the block-diagonal method, using the integer programming formulation, the number p of families is fixed.

Example 5.4.
The given incidence matrix is:

$$[a_{ij}] = \begin{array}{c} \text{Part No.} \\ \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \end{array} \quad \text{Machine No.}$$

The similarity matrix for the given incidence matrix $[a_{ij}]$ is:

$$[s_{ij}] = \begin{bmatrix} 0 & 1 & 4 & 2 & 3 & 3 \\ 1 & 0 & 0 & 4 & 3 & 1 \\ 4 & 0 & 0 & 1 & 2 & 4 \\ 2 & 4 & 1 & 0 & 4 & 2 \\ 3 & 3 & 2 & 4 & 0 & 3 \\ 3 & 1 & 4 & 2 & 3 & 0 \end{bmatrix}.$$

Let us suppose that the number of family groups is $p = 2$. Applying the pivoting algorithm, the following solution is obtained:

$$[x_{ij}^*] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For this example, the objective function is:

$$\begin{aligned} z = \quad & 0 + 1x_{12} + 4x_{13} + 2x_{14} + 3x_{15} + 3x_{16} \\ & + 1x_{21} + 0 + 0 + 4x_{24} + 3x_{25} + 1x_{26} \\ & + 4x_{31} + 0 + 0 + 1x_{34} + 2x_{35} + 4x_{36} \\ & + 2x_{41} + 4x_{42} + 1x_{43} + 0 + 4x_{45} + 2x_{46} \\ & + 3x_{51} + 3x_{52} + 2x_{53} + 4x_{54} + 0 + 3x_{56} \\ & + 3x_{61} + 3x_{62} + 4x_{63} + 2x_{64} + 3x_{65} + 0 \end{aligned}$$

After inserting the values x_{ij}^{*}, we get:

$z^{*} = 4 + 4 + 3 + 3 = 14.$

Therefore, two following part families are formed:

PF1 = {1,3,6};
PF2 = {2,4,5}.

The corresponding two machine cells are found from the matrix $[a_{ij}]$:

MC1: {2,4};
MC2: {1,3,5}.

Examples 5-3 and 5-4 can be interpreted and solved graphically. A visual verification shows that the obtained solutions correspond with the *p*-median and block-diagonal methods.

In Fig. 5-30, a multigraph illustrates the incidence matrix for Example 5-3. The vertices correspond to the parts. Two vertices are connected by an edge if the same machine can process the corresponding parts.

It is easy to see the two disconnected components, which represent the part families: {1,3} and {2,4,5}.

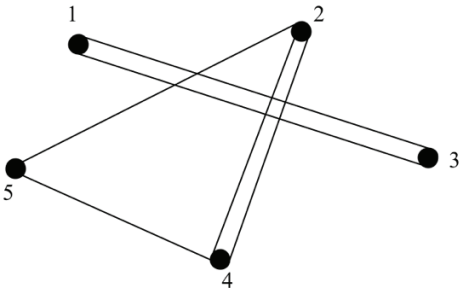


Fig. 5-30. The multigraph for Example 5-3. Part family detection.

A similar graph representation can be made for machine cell detecting. In such a graph, a vertex corresponds to a machine. An edge connects two vertices if the corresponding machines are able to process the same part (Fig. 5-31). One can detect two disconnected components, which represent machine cells: {2,4} and {1,3}.

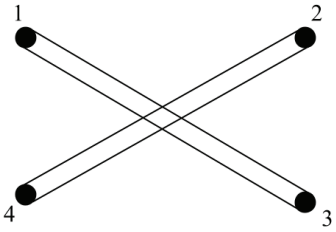


Fig. 5-31. The multigraph for Example 5-3. Machine cell detection.

The graphs illustrating the incidence matrix for Example 5-4 are given in Fig. 5-32 and Fig. 5-33.

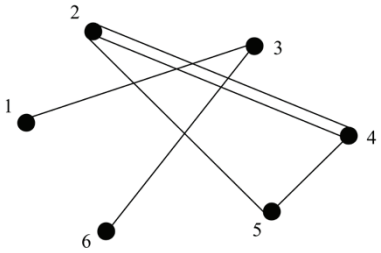


Fig. 5-32. The multigraph for Example 5-4. Part family detection.

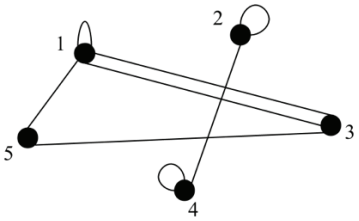


Fig. 5-33. The multigraph for Example 5-4. Machine cell detection.

5.7.5 Separability of clusters

The processing of an incidence matrix with the goal to obtain a block-diagonal pattern may result in the following two categories of clusters (Kusiak and Cho 1992, 2634):

1. Mutually separable clusters;
2. Partially separable clusters.

In the previous sections, *mutually separable clusters* were presented in Figs. 5-12, 5-21, 5-25, 5-26, and 5-29. The impossibility to form an ideal block-diagonal matrix gives space to *partially separable clusters*, which cannot be reduced to such a form. In this situation, the objective is to form machine-component cells, where the number of odd elements in the off-diagonal is minimized. This transformation minimizes the number of inter-cell moves. An example of clusters of the second type is given in Fig. 5-34.

The matrix in Fig. 5-34 cannot be separated into two disjoint clusters due to part 5, which is to be processed in both cells, MC1 and MC2. When part 5 is removed from the matrix, its decomposition into two separable machine cells can be achieved, MC1={1,2} and MC2={3,4}, and two part families, PF1={1,2} and PF2 = {3,4}.

Machine No.	**Part No.**				
	1	**2**	**3**	**4**	**5**
1			1	1	1
2			1	1	
3	1	1			1
4	1	1			

Fig. 5-34. Incidence matrix with two partially separable clusters.

These two cells are called *partially separable clusters*. The overlapping part, which is processed on machines belonging to two or more machine cells, is called a *bottleneck part*. A popular technique to eliminate a bottleneck part is to use an *alternative process plan*. Given the incidence matrix of Fig. 5-34, an alternative process plan for part 5 of the involved machines 1 and 3 results in two mutually separable machine cells.

Analogously to the bottleneck part, a bottleneck machine can be defined. A *bottleneck machine,* also called an *exceptional machine*, is one that processes parts belonging to more than one cell. Therefore, it does not allow the decomposition of the machine-part incidence matrix into disjoint blocks (Hachicha, Masmoudi, and Haddar 2008, 1158–60). In the matrix given in Fig. 5-35, machine 3 is the bottleneck, because it does not permit a decomposition of this matrix into a block-diagonal form.

Machine No.	Part No.					
	1	2	3	4	5	6
1	1	1				
2	1	1				
3	1	1	1		1	1
4			1	1	1	1
5			1		1	1

Fig. 5-35. Incidence matrix with the bottleneck machine 3.

Alternative process plans are frequently available for many parts. A way to decompose the matrix in Fig. 5-35 into two disjoint blocks is to create an additional copy of machine 3. With this transformation, two separated machine cells are obtained: MCl = {1,2,3(l)}, MC2 = {3(2),4,5}, and two corresponding part families are formed: PFl = {1,2} and PF2 = {3,4,5,6}, as it is shown in Fig. 5-36.

MC		PF1		PF2			
		1	2	3	4	5	6
MC1	1	1	1				
	2	1	1				
	3(1)	1	1				
MC2	3(2)			1		1	1
	4			1	1	1	1
	5			1		1	1

Fig. 5-36. Incidence matrix with partially separable clusters.

5.7.6 Other clustering methods

Various methods of a cluster analysis can be found in the literature after the definition given by Gower (1971, 859) for a general coefficient to measure the similarity between two sampling units and the pioneering work of McAuley (1972, 54–55). King (1980) proposed a *rank order clustering* (ROC) algorithm for a machine-component grouping in PFA. Relaxation and regrouping procedures were developed, in which the basic ROC method was extended to the case with bottleneck machines. Chandrasekharan and Rajagopalan (1986) developed an ideal seed non-hierarchical clustering

algorithm for cellular manufacturing. The problem was first formulated by means of a bipartite graph, which was a good illustration. An expression for the upper limit on the number of groups was derived. Using this limit, a non-hierarchical clustering method was adopted for grouping components into families and machines into cells. After diagonally correlating the groups, an ideal-seed method was used to improve the initial grouping. A quantitative criterion called grouping efficiency was then developed for comparing alternative solutions. Kumar, Kusiak, and Vannelli (1986) proposed a model for the grouping problem as an optimal *k*-decomposition of weighted networks. Large problems were modeled by using a computer to find an initial solution and later refining this solution. Bounds on the performance of the algorithm were constructed to give an estimated quality on the generated solution. Seifoddini and Wolfe (1986; 1987; 1988) suggested an Average Linkage Clustering (ALC) algorithm, which improved the previous models based on similarity coefficients by dealing with the duplication of bottleneck machines and by employing special data storage and some analysis techniques. This algorithm greatly simplified the machine-component grouping process. The duplication process in this model was based on the number of inter-cellular moves. Duplication starts with the machine generating the largest number of inter-cellular moves and continues until no machine generates more inter-cellular moves than specified by a threshold value. By changing the threshold value, alternative solutions can be examined. This model employed the bit-level data storage technique to reduce the storage and computational requirements of the machine-component grouping process. Kusiak (1987, 565) proposed a generalized model for the cases when rearranging rows and columns of the incidence matrix does not result in a block-diagonal matrix. Srinivasan, Narendran, and Mahadevan (1990, 147–49) presented an attractive *p*-median-based assignment model to solve the part grouping problem. A similarity coefficient matrix was used as input. Closed loops in the form of subtours were identified after solving the problem and were used as the basis for grouping. The method was applied to a number of examples. The assignment method emerged as a distinctly superior technique both in terms of the quality of the solution and computational time being compared with the earlier mathematical programming model and the *p*-median model. Cedeño and Süer (1997) discussed the use of a similarity coefficient-based method to perform a cluster analysis to a large set of data with dissimilar parts. The case was considered when many part families have a low number of parts and a few families were formed by a high number of parts. This issue led to the definition of the *remainder cluster*. In this manner, those parts or clusters of parts are designated, which at a certain threshold value

do not join any part family and do not have sufficient parts to be considered as a feasible part family. An approach was proposed to deal with the remainder clusters with the objective to form part families, each with a feasible number of parts.

A combinatorial approach, which used *Benders' decomposition*, was proposed by Heragu and Chen (1998) for an optimal solution of a cellular manufacturing system design. A mathematical model was formulated. The method incorporated three critical aspects:

1. Resource utilization;
2. Alternate routings;
3. Practical constraints.

The problem was shown to be NP-hard. A linear mixed integer version of the model and its optimal solution using Benders' decomposition approach were described together with an example of the obtained machine/part cluster.

Jeon and Leep (2006, 267–69) studied the problem of forming part families and designing machine cells under demand changes. A two-phase methodology was developed. A new similarity coefficient was suggested in Phase I to identify part families by using a genetic algorithm. The similarity coefficient considered the number of available alternative routes during a machine failure. A new methodology was introduced in Phase II for the cell formation. It considered scheduling and operational aspects in cell design under demand changes. Machines were assigned to part families by using an optimization technique. This optimization technique employed sequential and simultaneous mixed integer programming models for a given period to minimize the total costs, which were related to the scheduling and operational aspects.

Hachicha, Masmoudi, and Haddar (2008, 1160–61) presented a multivariate approach called *principal component analysis* (PCA) to form a machine/part block-diagonal matrix. It was a logical and systematic approach to the design of cellular manufacturing systems. The authors used a correlation matrix as similarity coefficient matrix, which later was used as an input for the PCA to identify similar groups.

There are various surveys and taxonomies in the literature related to clustering.

King and Nakornchai (1982) reviewed various approaches, which have been adopted in an attempt to solve the problem of joining the machines into groups and the components into associated families. The authors described also a new version of the ROC algorithm. Shafer and Rogers

(1993a) proposed a survey on similarity/dissimilarity and distance measures for cellular manufacturing. The evolution of similarity metrics was also discussed. In the second part of their work (Shafer and Rogers 1993b), some extensions and comparisons of these metrics were proposed. Offodile, Mehrez, and Grznar (1994) employed a taxonomic framework for a comprehensive review on cellular manufacturing systems. Three classes of machine-part grouping techniques have been identified in cellular manufacturing: visual inspection, part coding and classification, and analysis of the production process. A comprehensive review and discussions of various models were provided. The presented assumptions and characteristics were summarized using a tabular framework. Yin and Yasuda (2005) presented a comparative investigation on similarity coefficient methods applied to the cell formation problem. Nine performance measures were used for evaluating the quality of the cell formation solutions. Two characteristics, *discriminability* and *stability* of the similarity coefficients, were tested under different data conditions. Three similarity coefficients were found to be more discriminable. The Jaccard-based coefficient was found to be *the most stable* one. Four similarity coefficients were not recommendable due to their poor performance.

A short resume on the literature related to part family grouping and machine cell clustering is given in Table 5-2.

5.8 Cellular manufacturing

Traditionally organized manufacturing systems, such as a job shop or a flow shop, are inefficient to control the massive production in advanced productions, which are subject to an extreme pressure due to a high variability of the products, every time shorter product life-cycles, impulsive and unexpected customer demands, etc. The satisfaction of flexibility and JIT processing capabilities forces the manufacturing companies with discrete processing to take attention on the optimization of the configuration of their facilities.

By the application of GT, which is a philosophy that utilizes similarities to simplify the production processes and the product design through standardization approaches, CM has emerged as a viable replacement of traditional systems. The origins of the CM concept are traced to the 1980s and J. L. Burbidge, who introduced in his book *The Introduction of Group Technology* (1975), a concept referred to as *cell technology*, which later has been transformed into CM. CM can be defined as the organization of a manufacturing system that integrates the machine cells, and every cell is

configured to produce a family of parts. In this manner, a cell is a small scaled production unit within a larger factory.

CM is a lean manufacturing approach that helps the companies in the production of a variety of products for the customers with as little waste as possible. For a contemporary plant, CM is a stepping stone to achieve a world-class manufacturing status.

The *principal objective of CM* is to minimize the total material handling cost caused by inter-cellular moves to maximize the intra-cellular utilization of the machines and to minimize the duplication of machines in a cell. This objective is reached through the design of the machine cells in such a way that critical measures of the production performance are optimized. The performance could be measured by the productivity, the cycle time, and logistic indexes. Typical practical measures include also pieces-per-man-hour, unit cost, on-time delivery, lead time, and defect rates.

Employing a CM strategy in a production system leads to a physical division of the functional manufacturing machinery into a set of machine cells. The *paradigm of the organization of a production system* on the base of the *machine similarity* shifts to the identification and grouping different machines into cells. For this, it is necessary to identify a part family or a set of part families, which require similar machinery, machine operations, and/or jigs and fixtures. Therefore, these parts may be processed together. These groups discompose a production into *manufacturing cells*. The parts within the family are normally transformed from a raw material to a finished good within a single cell. In each production cell, the equipment and the workstations are arranged in a sequence that supports a smooth flow of materials and components through the process with minimal transport or delay. This type of layout is commonly referred to as a *group* or *cellular layout* (Fig. 5-37).

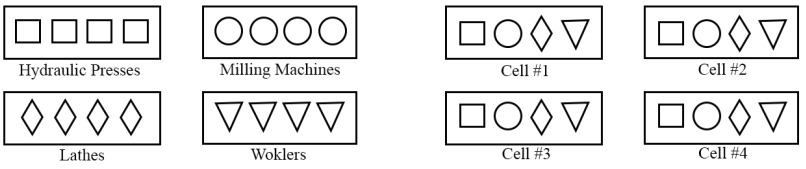


Fig. 5-37. Design of a manufacturing system: *a*) Process layout; *b*) Cellular layout.

Approach base	Reference
General coefficient measuring	Gower (1971)
Single Linkage Cluster Analysis (SLCA)	McAuley (1972)
Bond energy algorithms (BEA)	McCormick, Schweitzer, and White (1972); Slagle, Chang, and Heller (1975); Bhat and Haupt (1976)
Modified McAuley similarity for cell design	Rajagopalan and Batra (1975)
Rank order clustering (ROC)	King (1980); King and Nakornchai (1982)
Direct Clustering Algorithm for Group Formation (RCA)	Chan and Milner (1982)
Weighted Average Linkage Clustering (WALC)	Chandrasekharan and Rajagopalan (1986)
Optimal *k*-decomposition of weighted networks	Kumar, Kusiak, and Vannelli (1986)
Average Linkage Clustering (ALC) algorithm; Similarity Coefficient Method (SCM)	Seifoddini and Wolfe (1986; 1987); Seifoddini (1988)
Within-cell utilization-based heuristic	Ballakur and Steudel (1987)
p-median model	Kusiak (1987)
Cost-based heuristics	Kusiak and Chow (1987) Askin and Subramanian (1993)
Production flow analysis (PFA)	Burbidge (1989)
Identification, clustering, refinement, merging and allocation heuristic (ICRMA)	Tabucanon and Ojha (1987)
Extended cluster identification algorithm	Kusiak and Cheng (1990)
An assignment model for the part-families problem. Closed loops identification algorithm.	Srinivasan, Narendran, and Mahadevan (1990)
Clustering of a binary incidence matrix with bottlenecks	Kusiak and Cho (1992)
Group formation by neuronal networks	De La Fuente García, Diez and Fernandez (1995)
Fixed family approach to a large set of data with dissimilar parts	Cedeño and Süer (1997)
A similarity coefficient-based methodology	Offodile and Grzna (1997)
Benders' optimal decomposition approach	Heragu and Chen (1998)
Similarity-based distance between parts	Andrés et al (2005); Ghosh and Dan (2011)
Kohonen self-organizing map (KSOM) networks	Venkumar and Haq (2006)
ART-modified single linkage clustering	Murugan and Selladurai (2011, 205–7)
Principal component analysis	Hachicha, Masmoudi, and Haddar (2008, 1160–61)
Surveys and taxonomies	King and Nakornchai (1982); Greene and Sadowski (1984); Shafer and Rogers (1993a; 1993b); Offodile, Mehrez, and Grznar (1994); Selim, Askin, and Vakharia (1998); Yin and Yasuda (2005)

Table 5-2. A review of the grouping/clustering literature.

The design, subsequent control, and operation of a CM system are constrained by a set of implied assumptions. They are typically applied in such a manner that the advantages of CM are maximized, while the disadvantages are minimized. The generally recognized *assumptions* include (Greene and Sadowski 1984, 84):

1. Parts are grouped on the base of their specific shape and required manufacturing operations;
2. If possible, the manufacturing machines should be grouped so that all operations on parts belonging to the same family are completed in a single cell;
3. Operations required by any job should not be split between the cells;
4. Cells can share the machinery. Nevertheless, the number of shared machines should be minimized;
5. Each cell is designed as a modified flow shop;
6. Machines, which are not grouped into specialized cells, are grouped into a *remainder cell*;
7. Some machinery cannot be grouped, for example, paint booths and toxic degreasing equipment;
8. For any job, there is at least one feasible cell, where all operations can be completed;
9. Jobs have more than one feasible cell;
10. If there is a specialized cell with assigned jobs, these jobs should be assigned to a specialized cell instead of being assigned to the remainder cell;
11. The efficiency of a cell and/or machines within the cell, which perform the operations on a job, is partially correlated to the characteristics of the jobs;
12. Most machines in a cell have some flexibility to perform multiple operations.

CM is usually implemented into a production process by the following four stages:

1. *Cell formation*: Grouping parts into part families and the corresponding machines into machine cells;
2. *Intra-cell layout*: Layout of the machines within each cell;
3. *Inter-cell layout*: Layout of the cells within the factory or shop floor;
4. *Scheduling*: Scheduling of the jobs in each cell.

5.9 Cell formation problem

Among the CM problems, *the cell formation* (CF), also referred to as *the cell layout*, is considered fundamental and the foremost problem in designing a CM system. The problem can be summarized as follows (Wei and Gaither 1990, 222):

> 'If the number, types, and capacities of the production machines, the number and types of parts to be manufactured, and the routing plans and machine standards for each part are known, which machines and their associated parts should be grouped together to form a cell? '

For the solution of a CF problem, it is usually assumed that 1) a specific set of parts is identified to be suitable for manufacturing on a specified group of machines or machine types, and 2) there exists a basic relationship between a part and a set of machines, for instance, a *part routing*. The parts can be assigned to families in such a way that the same group of machines processes all parts in the family, and machines are grouped into cells if they process the same set of parts. Most procedures for CF rely on this type of relationship to establish part families and machine cells. Once the part and machine populations have been identified, the CF problem can be reduced to three major decisions:

1. Identification of part families;
2. Identification of machine cells;
3. Allocation of the families to cells or vice versa.

These three decisions are interrelated and compose subproblems of the CF problem.

In an ideal situation, a *cell* is a group of dissimilar machines physically located in a close proximity so that all operations of parts in a part family are processed from the start to finish in a single continuous flow without any backtracking. However, a fully independent layout is rare in practice. In the majority of layouts, the degree of independence is restricted by exceptional parts and exceptional machines. An *exceptional part* requires the processing by at least one machine, which belongs to more than one machine cell. Similarly, an *exceptional machine* processes parts from more than a one-part family. Such situations may occur when an exceptional machine is unique, or the utilization of the machine must be increased (C. -H. Cheng, Kumar, and Motwani 1995, 87). On the other hand, a job assigned to a cell may only require a fraction of the machines or all of them.

The cells are created for efficient manufacturing plants. Nevertheless, it is not always possible to convert the entire job shop into a number of production cells. Usually, there are some parts, which cannot be associated with a family and therefore, they cannot be placed into a specific cell. There exists also a specialized machinery, which cannot technologically be placed into a separated cell because of its general utilization. Therefore, the CF problem could be defined as identifying and grouping parts into families and machines into cells, and then assigning families to cells based on routing sheet information such that one or more of the following *objectives* are satisfied (Ballakur and Steudel 1987, 640):

1. Minimize the number of inter-cellular moves;
2. Minimize the number of cells;
3. Maximize the utilization of machines;
4. Minimize the duplication of machines in different cells;
5. Maximize the percentage of operations of a part processed within a single cell;
6. Maximize the number of parts handled by the cells as a percentage of the total number of parts processed through the shop;
7. Minimize the total manufacturing costs (by a reduction of setup times and WIP inventory levels);
8. Minimize the job throughput time;
9. Minimize the job lateness.

It should be noted that many of the above mentioned objectives, such as minimizing the job throughput times, are not solely determined by CF, but also by the operating policies of the shop (for example, dispatching rules, scheduling policies, etc.), which are used in the system.

A machine cell, where all operations for a product or service are performed in close proximity, is often configured in a *U-shaped layout*, to ensure the production flow and to allow a quick feedback between operations when problems and other issues arise. The workers in the cells are typically cross-trained and able to perform multiple tasks as needed. Common forms of a single cell are also a straight line, an L-shape, an S-shape. The number of workers inside these cells depends on the current demand and can be changed according to increasing or decreasing the production (Fig. 5-38).

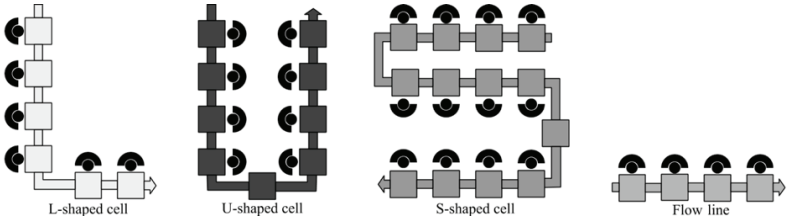


Fig. 5-38. Shapes of a machine cell.

CF, machine layout, and cell layout problems in a CM system are known to be NP-hard optimization problems. Exact solution methods are inefficient when the problem size is large. Therefore, heuristics, metaheuristics and hybrid methods to solve them efficiently have been developed (Lamba et al. 2020; Shashikumar et al. 2019; Zandieh 2019; Forghani, Fatemi Ghomi, and Kia 2020).

5.10 Conclusions

For many years, the industrial production has faced an increase in the complexity and a decline in productivity due to an increase in the part mix, the volume of the parts, the plant size, the machine production rates, and the part complication. The development of GT and CM is a response to the increased complexity and an effort to maintain the productivity. CM is not the same as GT. The latter one may include CM, but it is a much broader concept. So, one can say that CM is an application of GT to the production. CM uses the principle of GT by grouping parts with similar characteristics into part families and the corresponding machines into machine cells in order to achieve a higher production efficiency compared to traditional manufacturing.

Classification and coding systems are traditional tools used to implement CM. A classification and coding system allows assigning codes to parts. Based on these codes, the parts can be grouped into families. The disadvantage of a classification and coding system is that its implementation is time consuming to assign a code to every part. In addition, parts of similar size, shape, and function may not use the same set of machine tools and other resources. Therefore, recent works are focused on the use of cluster analysis.

A cluster analysis identifies similar and dissimilar object features as well as groups objects into homogeneous groups. An underlying assumption is that homogeneous clusters exist in raw data. A usual task in a cluster analysis is to develop efficient and effective clustering algorithms that

identify homogeneous clusters. Unlike classification and coding systems, a cluster analysis uses only information that is available in a production system.

The following reviews have been presented in the GT literature. The first survey found was due to Pullen (1976). It was dedicated to the description of CM features. King and Nakornchai (1982) reviewed various approaches that have been adopted in an attempt to solve the problem of grouping machines into cells and components into associated families. Green and Sadowski (1984) proposed a review of CM assumptions, advantages, and design techniques. Shafer and Rogers (1993a; 1993b) presented an overview on similarity and distance measures for CM, a survey, and a comparison of techniques. The first taxonomic review on CM was due to Offodile, Mehrez, and Grznar (1994). Mosier, Yelle, and Walker (1997) published a survey of similarity-oriented metrics for the GT configuration problem (GTCP). Selim, Askin, and Vakharia (1998) discussed, reviewed and classified up-to-day methods for solving a CF problem, which is a fundamental issue in CM. Allahverdi, Gupta, and Aldowaisan (1999) revised advances in scheduling problems with setup times, including results for batch setup times with GT assumptions. Yin and Yasuda (2005) dedicated their comparative investigation to similarity coefficient methods applied to the CF problem. The most recent overview of relevant studies was due to Esmaeilian, Behdad, and Wang (2016). The authors studied recent publications in the organization of CM, starting from past and current trends to future developments. By this extensive survey of the literature, future directions of this changing field were suggested. As relevant books, the following two ones can be recommended: Benhabib (2003) and Kamrani et al. (2013).

The basic theory of GT has mainly been developed. Nevertheless, its application in practice is still actual and insufficient. There are recent publications, where new algorithms and approaches are offered, but there is a lack of related literature, such as books and surveys, which explain the bases and summarize the advances in the theory for industrial employers and research requests.

CHAPTER SIX

BATCH SCHEDULING

...job scheduling in batches is an important issue in the manufacturing industry.
(*Mathirajan and Sivakumar 2006, 990*)

In a manufacturing process with a bottleneck operation, it is desirable to keep changeovers on a machine as few as possible in order to reduce the setup time required, which is a non-production idle time. With this reason, when setups are very costly in terms of money or time, jobs with similar characteristics are grouped and processed together. Consecutively, the required number of setups is reduced. In such scenarios, processing jobs in a batch and allowing a single setup per batch may give a sound operational advantage. In this case, minimizing total flow time, a reduction of the WIP inventory, improving the customer service responsiveness, and other production indexes can be reached.

A *batch scheduling problem* consists in grouping the lots on each machine into batches and then scheduling these batches. The concepts of *batch scheduling* and a *batch machine* have arisen from *burn-in operations* in semiconductor manufacturing industries, which represent today one of the most complex industries with batched and singular wafer processes. These concepts are combined with numerous production restrictions, such as different kinds of setup times, prescribed customer due dates for the lots, very expensive equipment, reentrant process flows, etc. (Mathirajan and Sivakumar 2006; Mönch et al. 2011). In such a changeable scenario, maintaining a competitive advantage and remaining profitable in the operational terms requires the minimization of the product cycle time and the WIP inventory, the maximization of throughput, etc. It is particularly reached by an efficient batching.

In this chapter, the main concepts of batch scheduling are collected and described on the base of a review of the contemporary literature. The burn-in operation, which gave rise to the batch machine concept, is described in detail. The treatment of different batching models and typical cell architectures are explained and illustrated by examples in this chapter. Some concluding remarks are also presented.

6.1 Basic steps of semiconductor manufacturing

The semiconductor production is a very complex production process due to the diversity of the environments, the variety of models, and a high automation level of the manufacturing process. On the other side, the same characteristics make the application of scheduling methods very efficient. It is the reason of the interest of researchers in batching models in general, and particularly in the burn-in operation, which caused a multitude of studies.

In any semiconductor manufacturing plant, the fabrication of *integrated circuits* (IC) can be divided into *four basic steps*: wafer fabrication, wafer probe, assembly or packaging, and final testing, as it is shown in Fig. 6-1.

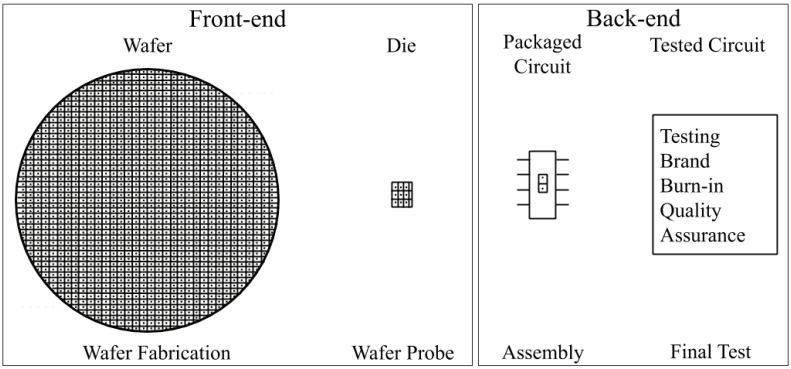


Fig. 6-1. Basic steps of the semiconductor manufacturing process. Adapted from Uzsoy, Lee, and Martin-Vega (1992, 48).

6.1.1 Front-end operations

The first stage, *wafer fabrication*, is the most technology consuming and capital-intensive one. At this stage, the layers and patterns of metal and wafer material are built up over its surface in order to manufacture the required chips. For a complex component, such as a microprocessor, the number of operations can reach hundreds, and one wafer may contain hundreds of components. The production of wafers involves the use of precious metals and a clean-room environment to prevent a particulate contamination of the wafers. The wafers move through the plant in lots, which are deposited in standard containers of a constant size. The *wafer probe* operation is as follows. The individual circuits on each wafer are subjected to an electrical test by means of thin probes. The failed circuits,

which do not correspond to the required specifications, are marked by an ink dot. The wafers are then cut up into the individual circuits, and the defective circuits are discarded. These two stages, the wafer fabrication and wafer probe are generally referred to as the *front-end* operations.

6.1.2 Back-end operations

The two following stages, the assembly and final test, are referred to as the *back-end*. In the back-end operations, the lots may vary significantly in the size from several individual circuits to various thousands. The sequence of operations, which follows a lot, depends on the product and on the customer specification. Therefore, a lot is generally more closely associated with a particular order and a customer than with the wafer fabrication or the probe (Uzsoy, Lee, and Martin-Vega 1992, 48–49). In the *assembly* stage, the circuits are placed into plastic or ceramic packages to protect the integrity and influence of the environment. Once the package is sealed and tested for the leaks and other defects, the product is moved to the *final test*. The objective of this stage is to test all circuits produced by the company to ensure that the defective products are not passed to a customer. In the testing process, the automated equipment is used to check each IC and to determine whether it is able to operate at the required specifications.

6.1.3 Burn-in operation

Due to various operations employed in the wafer fabrication process, some chips are *fragile* and may fail after a short time. It is essential to identify and scrap these devices as *infant mortality*. Identifying and scrapping the fragile devices is realized by the *burn-in operation* (C.-Y. Lee, Uzsoy, and Martin-Vega 1992, 764–65; Sung et al. 2002, 996–97; Mathirajan and Sivakumar 2006, 991–92; Mönch et al. 2011, 3380–82). The purpose of burn-in is to detect fragile IC chips by a test. A heat stress is applied to the chips, placed in an *oven*. This operation forces a failure of weak or fragile devices. IC chips arrive at the burn-in area in lots consisting of pieces of the same product type, where each lot is referred to as a job. IC chips of each job are loaded onto boards to process a burn-in operation. Each job may have different lot sizes so that those job sizes (number of required boards) are not identical. The boards are often product-specific, and a job cannot be processed if there are not the necessary boards. Once IC chips have been loaded onto a board, the last one is placed into an oven. Typically, the oven capacity is larger than the job size. Therefore, the number of boards in an oven can keep the oven capacity and the number of the boards required

defines the size of a job. Each IC chip has a pre-specified minimum burn-in time, which depends on its type and/or the customer's requirements. Since IC chips may stay in the oven for a period longer than their minimum required burn-in time, it is possible to place different products (jobs) in an oven simultaneously. The processing time of each batch equals the longest minimum-exposure time among all products (jobs) in the batch.

During the testing, it may occur that a circuit did not correspond to an original specification, but met another one, which is less strict. Therefore, a number of different grades of the product may be applied when a lot is tested, and as a result, the pieces, which did not pass the test, are not moved to the scrap and can still be used in other ranges of the tested characteristic for *lower-grade products*. It is also possible to use the inventory of a higher-grade product for a demand of a lower-grade product. It may occur that a part of components did not pass a test. These components can also be sent back for *reworking*. These factors, among others, create difficulties for the production planning and scheduling process in the semiconductor industry. This leads to an enormous number of resulting scheduling problems.

Usually, the processing time of the burn-in operation is extremely long compared to other testing operations. It is generally considered as the *bottleneck* process in the final testing step. Therefore, an optimal scheduling of the burn-in operation is very important to improve the productivity of the whole process of chip manufacturing.

The semiconductor *burn-in scheduling problem* was first introduced by Lee, Uzsoy, and Martin-Vega (1992). In that paper, burn-in ovens were first modeled as batch processing machines, which are dedicated to handle a number of jobs simultaneously. An effective burn-in operation scheduling is a key issue because it causes frequently a bottleneck due to long processing times relative to other testing operations - days in contrast to hours. Other reason is that it occurs at the end of the manufacturing process and thus, it has a strong influence on the shipping dates.

6.2 Batching models

In the literature, a *batch* is defined as a maximal set of jobs, which are scheduled contiguously on a machine to share a setup (Potts and Kovalyov 2000, 228). A job represents usually an item or a *lot* of identical items. A lot and a batch can be interpreted as equivalent concepts, meaning a number of identical items. Nevertheless, a batch can also represent *a number of lots*. According to this definition, a batch is a generalization of the lot concept.

One must differ between a batch and a family. The scheduling approaches, which use these two concepts, can be employed for sequencing

part families as well as the parts within each family. In the scheduling problems, the *family* term, when used, denotes the initial partitioning of the jobs, while the term of a *batch* is used to denote a part of the solution. In a *family scheduling problem*, the jobs are partitioned into families according to their similarity so that no setup is required between two sequential jobs if both belong to the same family. A setup time is required at the start of the schedule and every time when the machine switches from processing jobs belonging to one family to jobs belonging to another family. When one batch is completed, the resource has to be adjusted for the next batch. The time needed for the setup activities depends on the families of both adjacent batches. A batch is called *feasible* if it can be processed without any tool switch.

In this model, a *batch* is a maximal set of jobs, which are scheduled contiguously on a machine and share a setup. Large batches have the advantage of high machine utilization because the number of setups is reduced. On the other hand, large batches increase the flow time. Moreover, processing a large batch may delay the processing of jobs of high priority, which belong to a different family. In the problems, which involve the selection of a batch size and creation of a schedule, there is a trade-off between *flow time* and *machine utilization*. This is the main reason to optimize the batch size.

When GT assumptions are respected, i.e., each family is processed as a single batch in each manufacturing cell, such an order is referred to as *group scheduling* (GS), and the families of the jobs are referred to as the *groups*. However, if a family of jobs is split into smaller batches by violating the GT assumptions, it is referred to as *batching and scheduling* (BS), wherein sub-families belonging to the same family are referred to as batches (C.-Y. Liu and Chang 2000; Shen, Gupta, and Buscher 2014, 353–54; Shahvari and Logendran 2016, 239–41).

Yimer and Demirli (2009, 119–20) proposed a methodology to deal with a batch scheduling problem. It requires five distinct but interdependent decisions to be made:

- *Grouping decision* – classify the set of job orders into families on the base of their setup similarity;
- *Batching decision* – find out the jobs of the same family, which have to be included into each batch;
- *Allocating decision* – resolve how many batches are assigned to the available machines at each stage of the operation;
- *Sequencing decision* – determine the order in which the batches and the jobs within each batch have to be processed;

- *Sorting decision* – regroup the finished jobs based on their due date and the customer ID.

Batch scheduling models can be classified into batch availability and job availability models, depending on when the jobs become available for the next operation, either to be processed on the next machine or for dispatching to the customer (Potts and Kovalyov 2000, 228–29; Allahverdi et al. 2008, 978). According to the *batch availability model*, all jobs of the same batch become available for processing at the same time and leave the machine together (Danneberg, Tautenhahn, and Werner 1999, 102; Selvarajah and Steiner 2006, 228–29; Mosheiov and Oron 2008, 1283; Ben-Dati, Mosheiov, and Oron 2009, 2–3). This situation occurs, for example, when the jobs of a batch are placed on a pallet, and the pallet is only moved from the machine when all jobs are processed. An alternative assumption is *job availability*, also referred to as the *item availability* (Shen and Buscher 2012, 15; Kress, Barketau, and Pesch 2018, 596). According to this model, the jobs enter and leave the machine in a batch. Therefore, a job becomes available immediately after its processing is completed. Its completion time is independent of the other jobs of the batch. In the batch scheduling problems, the assumption of job availability is adopted, unless it is stated otherwise.

Depending on the availability model, batch scheduling assumes two types of grouping jobs, referring to serial batching and parallel batching models. Consecutively, the processing time of a batch is determined as follows:

- In *serial batching*, also known as *s-batch* or *sum-batch*, the processing time of a batch is equal to the sum of the processing times of the included jobs;
- In *parallel batching*, also known as *p-batch* or *max-batch*, the processing time of a batch is usually equal to the largest processing time of the jobs it includes. It can also be fixed for an individual batching machine, see Sung et al. (2002, 997).

6.3 Serial batching

In serial batching, if a batch is assigned to an available machine, a setup is required at the beginning of the first job in this batch. In this case, a schedule defines: 1) the way in which the batches are created from the independent jobs, 2) the processing order of the batches, and 3) the processing order of

the jobs within the batches. A machine can only process one job at a time, and it cannot perform any processing while undergoing a setup.

6.3.1 Single machine

The most studied environment is the *single machine s-batching problem*. Despite seeming simplicity, only a few basic problems can be solved in polynomial time. Chen, Long, and Fung (2006) considered the single machine scheduling problem with sequence-dependent setup times to minimize maximum lateness. A genetic algorithm was proposed. Crauwels et al. (2005) addressed the problem of minimizing the number of late jobs under the same assumptions. Some branch-and-bound algorithms were developed, and some lower bounds were derived by relaxing either the setup times or the due dates. Yuan et al. (2006) considered the single machine s-batch scheduling problem with family setup times and release dates to minimize the makespan. The authors showed that this problem is NP-hard. They proposed dynamic programming algorithms for two variants of the problem. A heuristic with a performance ratio of two and a polynomial time approximation scheme (PTAS) for the problem were also given. Erel and Ghosh (2007) dealt with the single machine scheduling problem with due dates and batch setup times to minimize the weighted number of tardy jobs. A pseudo-polynomial dynamic program and a fully polynomial approximation scheme were given for the case when the due dates are uniform within a family.

The problem of scheduling *groups of jobs* on a single machine under the GT assumptions was studied by Cheng et al. (2008). Jobs of the same group were worked up contiguously and a sequence-independent setup time preceded the processing of each group. All jobs have a common fixed due date, which can be either *unrestrictively large* or *restrictively small*. The objective was to minimize the total weighted earliness–tardiness penalties. Properties of optimal solutions were established and dynamic programming algorithms were derived to solve several special cases of this problem.

Suppiah and Omar (2014) addressed the batching and sequencing of jobs originating from *incompatible families*. Sequence-dependent setup times exist on a single machine, what is typical for manufacturing practices. The total number of jobs is divisible by the batch size, and all parameters of the model are known in advance. Splitting jobs into batches is not allowed. The jobs in each batch may have different weights. A hybrid tabu search (HTS) algorithm was proposed to minimize TWT. The authors developed a testing methodology to determine the quality of the HTS solution. A MILP model was also developed to compare with the heuristic solution.

Pei et al. (2015) investigated a single serial batching scheduling problem with *deteriorating jobs* in a practical production with the goal to enhance the productivity of an aluminum manufacturing factory. In this problem, all jobs were first partitioned into serial batches. These batches were then processed on a single serial batching machine. Before each batch was processed, an independent constant setup time was required. The next model of the same group of authors was dedicated to a scheduling model with certain co-existing features of serial batching, dynamic job arrival, multi-types of job, and setup time assumptions (Pei et al. 2016). The jobs of all types were first partitioned into serial batches, which were then processed on a single serial batching machine with an *independent constant setup time* for each new batch. In order to obtain a solution, this scheduling problem was divided into two phases on the basis of the job arrival times. A corresponding two-phase hybrid algorithm (TPHA) was proposed. After this research on a process in aluminum manufacturing, the authors proposed another type of scheduling problem, which included deteriorating jobs (Pei, Liu, Pardalos, Fan, et al. 2017). Aluminum ingots were of multiple types, each of which included a certain number of aluminum ingots. Each type of aluminum ingots was first partitioned into multiple batches. Then all batches of different job types were processed on a single s-batch machine. The setup time before processing a batch was *sequence-dependent*. In this model, the actual job processing time was an increasing function of its starting time, and a setup time was required only when a new batch was processed first on the machine or immediately after a batch belonging to another job type. A set of optimization algorithms was developed to solve the makespan minimization problem, the maximum tardiness minimization problem, the maximum lateness minimization problem, and the maximum earliness minimization problem, respectively. Some optimization algorithms were also proposed to solve the problem of minimizing the number of tardy jobs under a certain agreeable condition. Later, the authors modified the model, introducing a *time-dependent setup time* with the *effects of deterioration and learning* (Pei, Liu, Pardalos, Migdalas, et al. 2017). The setup time for the batches was a linear function of its starting time. Structural properties were derived for the problems of minimizing the makespan, the number of tardy jobs, and maximum earliness. Three optimization algorithms were developed to solve the respective problems. Then the authors extended this problem by a simultaneous consideration of serial batching, a learning effect, and resource-dependent processing times in the processing model. The objective was minimizing the makespan under the constraint that the total resource consumption does not exceed a given limit (Pei et al. 2018). Structural properties for the job batching policies and

the batching sequencing were first proposed for the special case when the resource allocation was given. An optimal batching policy was derived on the basis of these properties. A novel hybrid GSA-TS algorithm, which combines a gravitational search and a tabu search techniques, was developed to solve the general case. Then Pei et al. (2019), extended the environment to the parallel-machine serial batching problem, where a position-based learning effect and a linear setup time were also considered.

Kress, Barketau, and Pesch (2018) studied a situation in an industrial environment, which was characterized by sequentially processed jobs on a single machine. In this problem, the setup operations may result in a *significant setup cost*, when the machine switched from processing a job of one family to processing a job of another family. These setups do not require time but are associated with a fixed cost, which is identical for all setup operations. For example, this cost may be due to the need to modify the machine by installing a different set of tools or loading a new software. In such an environment, it can be beneficial to group the jobs into batches and process contiguously the jobs, which belong to the same batch, in order to share the same setup requirements. Each job has a processing time and an associated deadline. It is known that the processing of large batches may delay the processing of important jobs, which form parts of other batches. Such a delay may eventually result in violating the respective deadlines. Hence, there is a trade-off between minimizing the total setup cost of a schedule and the need to guarantee the on-time production of the jobs to satisfy the deadlines. The starting and the completion times of each job, i.e., the time instants when the job is started to be processed and when its processing is completed, are independent of the other jobs of its batch. The job availability model was adopted. The schedule must satisfy on-time production all jobs with respect to their deadlines, and the total setup cost must be minimized. It was shown that the decision version of this problem is NP-hard in the strong sense. Properties of an optimal solution were presented. An $O(n\log n + nF)$ algorithm that approximates the cost of an optimal schedule by a factor of F, where F is the number of families, was developed.

6.3.2 Parallel machines

Mor and Mosheiov (2014) solved a batch scheduling problem for identical jobs with *controllable processing times* and a linear compression cost function in both a single machine and parallel identical machines environments. The job processing times can be controlled (i.e., compressed) by allocating additional resources. Batching of jobs was allowed under the s-batch assumption. Two versions of the problem were considered. In the

first one, the sum of the total flow time and the compression cost was minimized. In the second problem, the total flow time subject to an upper bound on the maximum compression was minimized. In all cases, close-to-optimal integer solutions for the relaxed version were generated by allowing non-integer batch sizes.

6.3.3 Flow shop

One of the first works considering the s-batch idea for permutation flow shop scheduling was due to Sotskov, Tautenhahn, and Werner (1996). There are given processing times t_{ij} of job i on machine j and setup times s_{rj} on machine j when a job of the batch r is processed after a job of another batch. The objectives were minimizing the makespan as well as the sum of completion times. Batch or item availability of the jobs was assumed. For these problems, various constructive and iterative algorithms were given.

When only *permutation schedules* are considered in a flow shop, in which the order of job processing is the same on all machines, the corresponding problems are denoted as *permutation flow shop group scheduling problem* (PFGSP) and *permutation flow shop batch scheduling problem* (PFBSP), respectively. Otherwise, such a scheduling problem involving *non-permutation* schedules are referred to as the *non-permutation flow shop group scheduling problem* (NPFGSP) and *non-permutation flow shop batch scheduling problem* (NPFBSP) (Shen, Gupta, and Buscher 2014, 353–54). Fig 6-2 illustrates the corresponding pairs of models. In both cases, four jobs are given. Jobs 1, 2, and 3 belong to the same family, and job 4 does not. In Fig.6-2a, each family forms a single batch according to the PFGS model. In Figure 6-2b, the families are divided into separate batches for the FBSP. One can see that PFBSP is more efficient than PFGSP. In both cases, non-permutation schedules can improve the makespan.

The PFGSP with *sequence-dependent family setup times* (SDFST) is NP-hard even in the simplest case for the two-machine flow shop with only one job for each family (J. N. D. Gupta and Darrow 1986). Moreover, all versions of FGSP and FBSP with SDFST are NP-hard (T. C. E. Cheng, Gupta, and Wang 2000, 265). In view of this, the corresponding research for solving the FGSP with SDFST makes an emphasis on branch-and-bound procedures and approximation algorithms. Schaller, Gupta, and Vakharia (2000, 327–32) and Shen, Gupta, and Buscher (2014, 355–58) gave an exhaustive collection of contemporary available approximate algorithms for solving PFGSPs with SDFST. These algorithms provide good permutation schedules within a reasonable computational time, but they do not guarantee the optimality of the schedule.

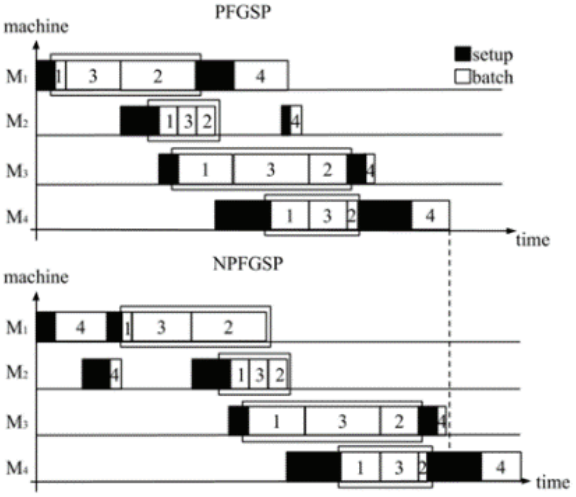
These algorithms can be classified into two categories:

1. *Constructive heuristics*, where once a job sequence is determined, it is fixed and cannot be reversed;
2. *Improvement heuristics*, which start with an initial solution and then provide an iterative scheme for obtaining an improved solution.

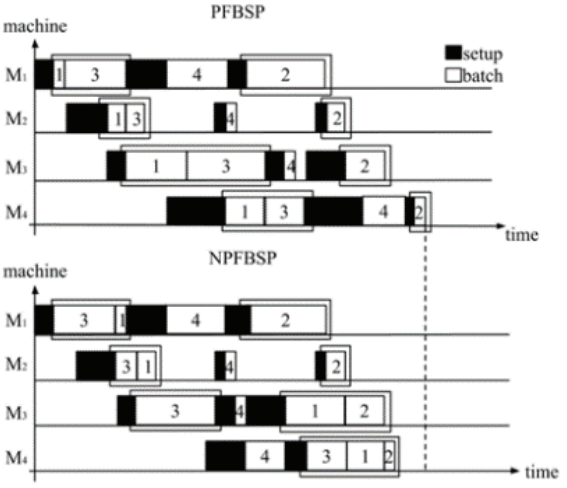
In order to reduce the setup times, Shen, Gupta, and Buscher (2014, 353) divided the product families into *inconsistent batches* on different machines. This procedure leads to non-permutation schedules. A tabu search algorithm, which used several neighborhood functions, was developed to solve the NPFBSP.

6.3.4 HFS

Xuan and Tang (2007) addressed the *s-stage* HFS problem to schedule *n* jobs with the s-batch processing model at the last stage. The authors reduced the problem to a two-stage HFS. *The transportation times* were considered separately from the processing times. The objective was to minimize a given criterion with respect to the completion time. When the jobs are grouped at stage *s*, each batch *l* has a given size b_l, i.e., the batch size is different for all batches. All jobs from the same batch must be processed on a machine at stage *s* consecutively while satisfying the given precedence constraints among the jobs within this batch. Each job *j* has a weight, and waiting for the job processing between two adjacent stages causes a penalty cost. A sequence-independent setup time is considered separately from the processing time before the first job of batch *l* starts its processing. It could be anticipatory, meaning that the setup of the next batch can start as soon as a machine becomes free to process the batch. An integer programming model and a batch decoupling-based Lagrangian relaxation algorithm were proposed. The problem was found in the iron and steel industry.



a)



b)

Fig. 6-2. Flow shop scheduling problem: *a*) Flow shop group scheduling problem (FGSP); *b*) Flow shop batching and scheduling problem (FBSP). Adapted from Shen, Gupta and Buscher (2014, 354).

Shahvari and Logendran (2016) dealt with the HFS batching and scheduling problem, where SDFST were present. The objective was to minimize simultaneously the weighted sum of the total weighted completion time and total weighted tardiness. The GT assumptions were disregarded by allowing the possibility of splitting pre-determined groups of jobs into inconsistent batches, in order to improve the operational efficiency, as it was made by Shen, Gupta, and Buscher (2014). A benchmark of small size problems was considered to show the benefits of batching on group scheduling. Since the problem is NP-hard, several tabu search-based algorithms were developed at three levels, which move back and forward between the batching and scheduling phases. Two algorithms incorporated tabu search into the framework of path-relinking to exploit the information on good solutions. An initial solution was implemented to trigger the search into the solution space.

6.3.5 Job shop and open shop

Shen and Buscher (2012) addressed the serial batch scheduling problem embedded in a *job shop environment* to minimize the makespan. The SDFST and the job availability assumption were also taken into account. A tabu search algorithm was proposed.

The paper by Mosheiov and Oron (2008) addressed the batch scheduling problems in an *m-machine open shop* environment. The objectives were a minimum makespan and a minimum flow time assuming an identical processing time of the jobs, machine- and sequence-independent setup times as well as batch availability. It was shown that the optimal number of batches was either m or $[n/(n/m)]$, where m and n are the number of machines and the number of jobs, respectively. The complexity of the minimum flow time problem was unknown. An $O(n)$ algorithm, which extended the solution of the single machine case and produced close-to-optimal solutions, was proposed.

6.4 Batch Processing Machines

The jobs may be batched if they belong to the same family or share the same setup on a machine. Another reason for batching occurs when a machine is able to process several jobs simultaneously, as it happens in the manufacturing of circuit boards during the burn-in operation. As it was noted before, the burn-in operation is performed in ovens that are able to accommodate several jobs (C.-Y. Lee, Uzsoy, and Martin-Vega 1992, 764–65; Mathirajan and Sivakumar 2006, 991–92). This assumption produces a

big class of scheduling problems, which consider a simultaneous processing of the jobs, and the batch size is restricted by the capacity of the machine used.

A *batch processing machine* (BPM) is a work center, which is dedicated to process a number of jobs *simultaneously*. Once the processing of a batch is initiated on a BPM, no job can be removed or added to the batch. On such a machine, a number of jobs may be processed together, or a large job can be broken up into smaller lots according to the machine capacity. These jobs can be similar or have different processing times. In the latter case, different jobs are batched together. The processing time of the entire batch is given by the lot, which has the longest processing time among all lots in the batch. It can also be independent of the processing time of the jobs included into the batch and can be fixed for an individual batching machine. An example of a batch scheduling problem with similar jobs for the final test stage in burn-in ovens was given in the paper by S. S. Chang and Young (2003, 645).

One can meet BPMs in many different environments such as chemical processes performed in the tanks or kilns and the burn-in operations in semiconductor industry. They are commonly used in wafer fabrication, kilns, environmental stress screening (ESS) chambers, chemical, food, and mineral-processing industries, pharmaceutical and construction materials industries, to name a few.

In serial manufacturing facilities, a BPM is typically a bottleneck in an assembly line, in which it is included. The BPMs produce many problems in the programming of the manufacturing process. Different scheduling problems involving BPMs have been addressed extensively in the literature for many discrete manufacturing industries, for example, shoe manufacturing, aircraft industry, furniture manufacturing, ion plating industry, iron and steel industry, steel casting and glass container production, etc. Results of the researches in this area are relevant to all shop environments: a single BPM, parallel BPMs, and flow shops. Ikura and Gimple (1986) were probably the first researchers to address the problem of scheduling BPMs from a deterministic perspective. The problems, which considered BPM scheduling, have received attention with the advances in the semiconductor industry, see Lee, Uzsoy, and Martin-Vega (1992); Uzsoy, Lee, and Martin-Vega (1992; 1994); Li and Lee (1997); Brucker et al. (1998); Lee and Uzsoy (1999); Damodaran and Srihari (2004); Jolai (2005); Mathirajan and Sivakumar (2006); Damodaran, Srihari, and Lam (2007); Manjeshwar, Damodaran, and Damodaran (2009); Trindade et al. (2018); Jia et al. (2019).

6.5 Parallel batching

6.5.1 Model properties

In the literature, p-batch scheduling forms a class of problems referred to as *batch parallel processing* or *BPM scheduling.* The two important decisions made on BPM-including environments are:

- Grouping part families into batches;
- Scheduling the batches to improve a performance measure.

For the p-batch assumption, there are two cases, which consider the batch size B. The special case when $B = 1$ is a classical scheduling problem that is solvable in $O(n\log n)$ time. If the sizes of the jobs are taken into account, the total size of all jobs must not exceed the machine capacity. This case is called a problem with *bounded batches*. As far as each job may have a different size, the number of jobs in each batch may be different. A batch is called *full* if it contains exactly B jobs; otherwise, it is called a *partial batch*. It is easy to see that there are at most n and at least $\lceil n/B \rceil$ batches in a feasible schedule, where $\lceil n/B \rceil$ denotes the smallest integer greater than or equal to n/B. As it was noted just before, a schedule for processing jobs on a BPM consists of a batching decision and finding the sequence of the batches.

When an unlimited number of jobs is allowed to be included into a batch, this is referred to as an *unbounded batch*. In this case, the number of jobs is not restricted to be processed in one batch (L. L. Liu, Ng, and Cheng 2010, 814; Yazdani Sabouni and Jolai 2010, 315). Scheduling problems involving a single BPM with bounded and unbounded batches have very different computational complexity. Thus, for a single BPM with an unbounded batch, i.e., $B \geq n$, where n is the number of jobs to be processed, the problem of minimizing the makespan is solved trivially by putting all jobs into one batch B_1. The minimum makespan is then $C_{\max} = p(B_1) = \max_{1 \leq j \leq n}\{p_j\}$.

Lee, Uzsoy, and Martin-Vega (1992, 771) showed that finding a batching scheme with minimum makespan on parallel BPMs with non-identical job sizes is NP-hard. It is NP-hard even for a single BPM. Table 6-1 shows an overview of the complexities for different objective functions.

Due to the complexity of BPM problems, the scheduling research mainly focuses on single and parallel BPMs.

Objective function	Unbounded	Bounded	
	$b \geq n$	$b = 1$	$b \geq 2$
f_{max}	polynomial	$O(n^2)$	NP – hard
C_{max}	$O(n)$	$O(n)$	$min\{O(n \log n), O(n^2/b)\}$
L_{max}	$O(n^2)$	$O(n \log n)$	NP – hard
$\sum_{j=1}^{n} f_j$	$O(n^2 P)$ NP - hard	NP - hard	NP – hard
$\sum_{j=1}^{n} C_j$	$O(n \log n)$	$O(n \log n)$	$O(n^{b(b-1)})$
$\sum_{j=1}^{n} w_j C_j$	$O(n \log n)$	$O(n \log n)$	Open
$\sum_{j=1}^{n} U_j$	$O(n^3)$	$O(n \log n)$	NP – hard
$\sum_{j=1}^{n} w_j U_j$	$O(n^2 P)$ NP - hard	$O(nP)$ NP - hard	NP – hard
$\sum_{j=1}^{n} T_j$	$O(n^2 P)$ Open	$O(n^4 P)$ NP - hard	NP – hard
$\sum_{j=1}^{n} w_j T_j$	$O(n^2 P)$ NP - hard	NP - hard	NP – hard

Table 6-1. Overview of the computational complexities for a single BPM with equal release dates. Adapted from Brucker et al. (1998, 32).

6.5.2 Single BPM

When a burn-in oven is modeled as a BPM, the typical assumptions are as follows:

Assumption 1. There are *n* jobs to be processed. The following attributes are associated with each job *j*: a processing time p_j, a due date d_j and a release time r_j, which corresponds to the time the job becomes available for processing on the batch machine. All data are assumed deterministic. The processing times are known in advance from the product test specification.

Assumption 2. Each machine can process up to *B* jobs simultaneously. All jobs require the same amount of the oven capacity. In practice, this corresponds to an offline assignation of the available boards to the lots awaiting burn-in and then treating each board as an individual job. The maximum number of *B* jobs, which can be processed simultaneously, is then given by the number of the boards, which an oven is able to accommodate.

Assumption 3. Once the processing of a batch is initiated, it cannot be interrupted, and other jobs cannot be deposited into the oven until the

processing is completed. Therefore, it is acceptable to keep a lot of chips in the oven for a longer period than its specified burn-in time, but not for a shorter period. The processing time of a batch is given by the processing time of the longest job in the batch.

The dispatching rules, applied to BPM scheduling, have a specificity compared with the traditional ones:

A sequence is in *batch-EDD order* if for any two batches P and Q in the sequence, where batch P is processed before batch Q, there is no pair of jobs i, j such that $i \in P, j \in Q$ and $d_i > d_j$.

A sequence is in *batch-LPT order* if for any two batches P and Q in the sequence, where batch P is processed before batch Q, there is no pair of jobs i, j such that $i \in P, j \in Q$ and $p_i < p_j$.

In the burn-in operation, the batches in an optimal schedule are not necessarily full, but the processing time of a batch is always equal to the longest processing time of the jobs in the batch. For example, let $B = 2$, $n = 2$, $p_1 = 10$, $p_2 = 30$, $d_1 = 10$, $d_2 = 40$ be given, and the goal is to minimize $T_{\max}$. If both jobs belong to the same batch, they both complete at time 30, and $T_{\max} = 20$. If they belong to separate batches and job 1 is scheduled before job 2, then $T_{\max} = 0$.

S*ingle BPM* problems are the most studied ones in the p-batch scheduling literature. One of the first works dedicated to a single BPM was due to Lee, Uzsoy, and Vega (1992). The burn-in oven was modeled as a BPM. The authors introduced and extended some important definitions: the triplet-notation to describe a single BPM scheduling problem as well as the usual assumptions and dispatching policies. For example, the notation $1|B|C_{\max}$ denotes the problem of minimizing the makespan on a single BPM; the notation $1|\ r_j,\ p_j = p,\ B|\sum U_{\max}$ denotes the problem of minimizing the number of the tardy jobs on a single BPM, where all jobs have equal processing times and job j is available at time r_j.

The research by Chandru, Lee, and Uzsoy (1993) was also motivated by the burn-in operation. A basic variant of the BPM model with a fixed number of job families was considered. The jobs of the same family had the same processing time. Properties of an optimal schedule were analyzed, and a dynamic programming algorithm of polynomial time complexity was developed. In the work by Sung et al. (2002), each job belongs to one of a given number of families. The release times of the jobs are different from each other. The objective is the minimization of the makespan. A dynamic programming algorithm was proposed for the situation, where a fixed number of job families was given.

Liu, Ng, and Cheng (2007) considered the problem of scheduling jobs with agreeable processing times and due dates to minimize total tardiness

and the weighted number of tardy jobs. The processing times and the due dates are *agreeable* if $p_i < p_j$ implies that $d_i \le d_j$, $1\ i < j \le n$. In the problems with agreeable processing times and due dates, the jobs should be re-indexed in non-decreasing order of their processing times and due dates such that $p_1 \le p_2 \le \ldots \le p_n$ and $d_1 \le d_2 \le \ldots \le d_n$. The problems under study were denoted by the three-field notation as $1|B$, agr(p_j, d_j)$|\sum T_j$ and $1|B$, agr(p_j, d_j)$|\sum w_j\ U_j$. A strict analysis of the NP-hardness of these problems was provided, and pseudo-polynomial time algorithms for NP-hard special cases of these problems were suggested. Applications of these problems were found in a variety of manufacturing environments, for example, heat treatment in the metalworking industry, very large-scale integrated circuits manufacturing, and diffusion or oxidation in wafer fabrication of semiconductor manufacturing.

A class of BPM scheduling problems is dedicated to the consideration of *incompatible job families*, where a set of n jobs is partitioned into m incompatible families and jobs of different families cannot be processed together (Uzsoy 1995; Jolai 2005; Koh et al. 2005). Jolai (2005) studied the problem of minimizing the number of tardy jobs on a single bounded BPM. The processing times of all jobs belonging to the same family are equivalent, and jobs of different families cannot be processed together. It was shown that this problem is NP-hard. A dynamic programming algorithm was presented. It has a polynomial time complexity when the number of job families and the batch machine capacity are fixed. This work was motivated by a diffusion operation in wafer fabrication facilities. A similar model, where the job sizes are arbitrary and the performance measures included the makespan, total completion time, and total weighted completion time was studied by Koh et al. (2005).

Ridouard, Richard, and Martineau (2008) proposed an *online scheduling* algorithm on a bounded BPM to minimize the makespan. The jobs in the same batch are completed at the same time at the final testing stage in a burn-in oven. Job release dates were also considered. This general problem with an infinite machine capacity was denoted as $1|$ p-batch, r_j, $B = \infty| C_{\max}$. Some simple algorithms were proposed.

In the scheduling literature, BPMs were also included into *more complex production environments*.

6.5.3 Parallel BPMs

Li, Li, and Zhang (2005) studied the problem of minimizing the makespan of *n* jobs on *m* identical *parallel* unbounded batching machines. Each job is characterized by a release time and a processing time. A PTAS was presented. Perez, Fowler, and Carlyle (2005) considered the diffusion step in the semiconductor wafer fabrication. It is very time consuming, compared to other steps. The authors modeled the diffusion furnaces as parallel BPMs with incompatible job families and focused on minimizing total weighted tardiness in this environment. The resulting problem is NP-hard. Therefore, it was decomposed into two sequential decision problems: 1) assigning the lots to batches, 2) sequencing the batches. Several heuristics were developed. In the problem studied by Mönch and Unbehaun (2007), the objective was to minimize the sum of the absolute deviations of the completion times from the due date (earliness–tardiness) of all jobs on parallel burn-in ovens. All jobs were assumed to have the same due date. Three decomposition heuristics were suggested: 1) to separate the sets of early and tardy jobs for each of the parallel burn-in ovens; 2) to assign the jobs to each single burn-in oven, and 3) to assign the jobs to *m* early job sets and *m* tardy jobs sets in the case of *m* burn-in ovens in parallel. Genetic algorithms, dynamic programming and sequencing rules were applied.

6.5.4 Flow shop

Sung and Kim (2003) dealt with an FS2 problem with two identically-bounded BPMs. Three due date-related objectives were studied, namely: 1) maximum tardiness, 2) the number of tardy jobs, and 3) total tardiness. The processing time of a batch depends on the individual machine, but not on the jobs in the batch, which is deterministic and known a priori. The problem was found in a wafer fabrication process, where the wafers were processed and moved through a series of subprocesses including cleaning, oxidation, lithography, etching and ion implantation. The batch transportation time between the machines was considered to be negligible. Three efficient polynomial time algorithms were developed for minimizing the due date related measures.

A BPM-containing FS2 variant with waiting time restrictions was considered in the paper by Su (2003). The production system included a bounded batch processor at stage 1 and a single processor at stage 2. For each job, the waiting time for the second stage cannot be greater than a given upper bound. The objective was to minimize the makespan. The application of this model was found, particularly, in the semiconductor wafer

fabrication. Diffusion and oxidation ovens are there working as batch processors, while the laser ablation, inspection, and repair operations are executed by discrete processors. The two-stage flow shop sequencing problem with limited waiting constraints was shown to be NP-hard. A heuristic algorithm and a mixed integer program were proposed.

6.5.5 BPM combined with deterioration effect

Tang and Liu (2009) dealt also with a similar FS2 scheduling environment, where a single machine was followed by a BPM. There existed a *transportation* as well as a *deterioration* phenomenon. This scheduling problem arose from the ingot teeming and the heating process in a steel plant. Gong, Tang, and Duin (2010) studied the same FS2 structure with a batching machine subject to the blocking constraint and a discrete machine with shared setup times. The problem of minimizing the makespan was shown to be NP-hard. This research was motivated by applications in the iron and steel industries.

Tang et al. (2017) formulated a scheduling problem, in which the jobs were generated by *two agents* and had *time-dependent proportional-linear deteriorating processing times*. The two agents compete for a common single batching machine to process their jobs, and each agent has its own criterion to optimize. The jobs may have identical or different release dates. The problem was to determine a schedule for processing the jobs such that the objective of one agent was minimized, while the objective of the other agent was maintained under a fixed value. Various combinations of the unbounded model were considered for regular objectives on the basis of the compatibility of the two agents. For the bounded model, two different objectives for incompatible and compatible agents were studied: 1) minimizing the makespan of one agent subject to an upper bound on the makespan of the other agent, and 2) minimizing the number of tardy jobs of one agent subject to an upper bound on the number of tardy jobs of the other agent. Different versions of the problem were analyzed and shown to be intractable. Nevertheless, an efficient exact algorithm was provided. The study was motivated by a production scheduling problem for the ingot soaking process of a primary rolling plant in the steel industry. Up to now, it is the only research dedicated to two-agent scheduling for problems with p-batch processing. Kong et al. (2019) stated a complex problem with a bounded p-batch machine, where *job rejection, deteriorating jobs, setup times,* and *non-identical job sizes* were considered. Each job can be either rejected with a certain penalty cost, or accepted and further processed in batches on a single machine. There was a setup time before processing each

batch and the objective was to minimize the sum of the makespan and the total penalty. Several useful preliminaries for arranging accepted jobs of identical size were proposed. Based on these preliminaries, the special case was first investigated, in which all jobs had an identical size. A dynamic programming algorithm was developed to solve this problem. These preliminaries helped to reduce the complexity of the dynamic programming algorithm.

6.5.6 Job compatibility on a BPM

The *job compatibility assumption* produces a class of scheduling problems with BPM included environments (Bellanger and Oulamara 2009; Oulamara, Finke, and Kamgaing Kuiteing 2009; Bellanger, Oulamara, and Kovalyov 2010). The study was first motivated by a scheduling problem in the tire manufacturing industry. Such a processing is also typical for galvanic operations, chemical milling operations, and temperature testing operations. The jobs in a batch have to be compatible on the batching machine, that is, they must share the same characteristics, particularly, the job processing time compatibility. This means that the jobs are compatible if their processing time intervals have a non-empty intersection.

Formally, the *job processing time compatibility* is defined as follows. There are n jobs to be processed on a batching machine. The processing time p_j of job j is given by the interval $[a_j, b_j]$ with the initial and terminal endpoints a_j and b_j, respectively. The batching machine has a capacity k, which means that at most k jobs can be processed simultaneously in a batch. The jobs of the same batch have to be compatible, i.e., the time processing intervals of the jobs must have a non-empty intersection. The processing time of a batch is given by the longest processing time of the jobs in the batch. This time corresponds to the left endpoint of the intersection of the job processing time intervals in the batch, which is the maximum initial endpoint a_j of the compatible jobs. Consecutively, the processing time of a batch B on the batching machine is $q(B) = \max_{j \in B}\{a_j\}$.

Compatibility is a symmetric binary relation. A pair (i, j) of jobs is compatible if they share a similar processing time on the machine, $[a_i, b_i] \cap [a_j, b_j] = \emptyset$, $i \neq j$. A compatibility relation, which is defined between each pair of operations, gives rise to an *undirected compatibility graph* $G = (V, E)$, where V is the set of vertices, which represent the jobs, and E is the edge set. A pair of jobs is connected if and only if they are compatible. A schedule is fully characterized by a partition of the jobs into batches and the batch sequence. The minimum completion time of the latest batch in a schedule is the minimal value of the makespan, $C_{\max} = \max_{1 \leq j \leq n}\{C_j\}$.

An example illustrates these concepts.

Example 6-1.
The problem by Oulamara, Finke, and Kamgaing Kuiteing (2009) describes a variant of such an environment in an FS2 with a discrete machine at the first stage and a batching machine at the second stage. There are $n = 8$ jobs in the system. All jobs visit the machines in the same order. The batching machine can process up to two jobs simultaneously, $k = 2$. This problem is denoted as $F2|p-batch(2), G_p = INT, k < n\ |C_{max}$, where $p-batch(2)$ means that machine 2 is a batching machine, $G_p = INT$ specifies that the compatibility graph is an interval graph, and $k < n$ indicates that the capacity of the batching machines is a part of the input (i.e., k is a constant).

The job processing times p_j on the first machine and the processing intervals q_j for the batching machine are given in Table 6-2. Fig. 6-3 shows the intervals of the processing time for every job. The corresponding compatibility graph is presented in Fig. 6-4. A feasible schedule contains the four batches $S = \langle \{J1, J8\}, \{J6, J7\}, \{J3, J5\}, \{J2, J4\} \rangle$. According to the definition and the input data, the processing times of the batches are determined and given in Table 6-3. A Gantt chart in Fig. 6-5 shows a feasible schedule S.

Job	***J*1**	***J*2**	***J*3**	***J*4**	***J*5**	***J*6**	***J*7**	***J*8**
Time p_j	4	7	5	6	8	6	10	4
Interval q_j	[5,15]	[3,6]	[7,10]	[3,11]	[9,12]	[11,16]	[15,18]	[14,19]

Table 6-2. Job processing times.

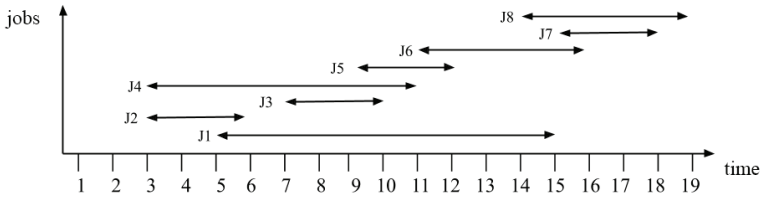


Fig. 6-3. The job processing times for the batching machine are given in form of time intervals.

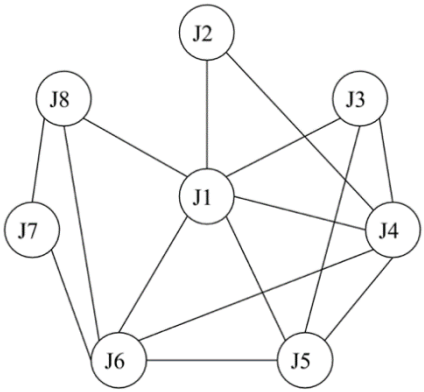


Fig. 6-4. Undirected compatibility graph

Batches	{*J*1, *J*8}	{*J*6, *J*7}	{*J*3, *J*5}	{*J*2, *J*4}
Batch processing time	14	15	9	3

Table 6-3. Batch processing times for a feasible partition of the jobs.

Fig. 6-5. A feasible schedule with $C_{\max} = 53$.

An extension of this model was presented by Bellanger and Oulamara (2009). A two-stage HFS with several identical discrete machines at the first stage and several identical BPMs, which process batches of compatible operations, were considered. The goal was to minimize the makespan. This problem was denoted as $FH2B(m_1,m_2)|p-batch(\text{II}), G_p = INT, k < n\ |C_{\max}$. Here m_1 and m_2 represent the number of machines at the first and second stages, respectively; $p-batch(\text{II})$ means that stage two is composed of BPMs; $G_p = INT$ specifies that the compatibility graph is an interval graph; $k < n$ indicates that the capacity of the batching machines is a variable and it is a part of the input. Since the problem is NP-hard, several heuristics were developed along with their worst cases analysis. The case when the jobs have the same processing time at the first stage was also considered, and a

PTAS algorithm was presented. Bellanger, Oulamara, and Kovalyov (2010) addressed an unbounded batching machine with the objective to minimize total completion time. The *normal job processing times* are given. An *actual job processing time* can exceed its normal value up to a certain percentage. If the corresponding processing time intervals intersect, this percentage is the same for all jobs. The job processing times are compatible. The problem was denoted as $1|p-batch(\text{II}), G = \alpha - INT \mid \sum C_j$. Here $G = \alpha - INT$ stands for the specific interval compatibility relations in the problem. The objective was to find a schedule, which minimizes total completion time. A dynamic programming algorithm was provided.

6.5.7 Multicriteria problems

The complexity of the production process management in advanced manufacturing systems with batch processing stimulated a recent interest of researchers to studies, in which one criterion is insufficient. Liu, Ng, and Cheng (2009) studied some *bicriteria* scheduling problems with equal processing times on a bounded p-batch BPM. This model was motivated by the problem of scheduling a burn-in operation in very large-scale integrated circuit manufacturing, where several criteria may be considered at the same time, for example, customer satisfaction, on/time delivery, and WIP inventory. As usual, a schedule for processing the jobs on a BPM consists of a batching decision and sequencing the batches. Two types of criteria in a multi-criteria problem can be distinguished: regular and additional ones. A *regular criterion* is nondecreasing in the job completion times. Regular criteria for machine scheduling, among others, include the makespan, total completion time, maximum lateness, number of tardy jobs, and total tardiness, denoted by $C_{\max}$, $\sum C_j$, $L_{\max}$, $\sum U_j$, and $\sum T_j$, respectively. If the jobs have different weights, then the corresponding *additional criteria* are $\sum w_j C_j$, $\sum w_j U_j$, and $\sum w_j T_j$, respectively. A useful characterization of an optimal schedule class for minimizing any regular single criterion and bicriteria scheduling problems were first developed. The bicriteria scheduling problems on a single BPM were transformed into two extended assignment problems. Then optimal polynomial time algorithms were provided for various combinations of these criteria. The results for the single machine case can be extended to the case of m identical parallel BPMs.

In the model by Kashan, Karini, and Jolai (2010), the processing times are non-identical. Two different multi-objective genetic algorithms based on different representation schemes were proposed for the simultaneous minimization of the makespan and maximum tardiness. Yazdani Sabouni and Jolai (2010) considered the problem of scheduling n jobs on a single

BPM, which can be both bounded and unbounded. In this problem, two customers ordered the jobs. The jobs belonging to different customers were processed according to an individual criterion. The considered criteria were the minimization of the makespan and maximum lateness for the problem with incompatible groups and unbounded batches. Optimal algorithms for three special cases were developed.

Luo et al. (2009) proposed a genetic algorithm for scheduling an HFS2 problem with sequence-dependent setup times, blocking, and machine availability constraints. The first stage consisted of multiple *parallel* bounded-batch machines with the job availability model, and the second stage had only one machine. A *blocking* environment existed between the two stages with no intermediate buffer storage. The scheduling problem with blocking and no-wait arises in serial product lines, where no intermediate buffer storage is available. In *no-wait scheduling*, the jobs must be processed from the start until the completion without any interruption. In *blocking scheduling*, a job completed at a machine must remain there until the downstream machine becomes available. Preventive maintenance and a machine breakdown were considered. Two types of machine unavailability, namely deterministic and stochastic cases were identified in this problem. The former one occurs on the machine at stage two, where both the start time and the end time were known in advance. The stochastic case occurs on one of the parallel BPM at stage one, and a real-time rescheduling is triggered. Minimizing the makespan was considered as the objective to develop an optimal scheduling algorithm. A genetic algorithm was used to obtain a near-optimal solution. This research has been motivated by a real-life problem faced by a company. It was specialized in metalworking, where the pre-heating section combined with the rolling section formed a two-stage HFS system.

6.6 No-wait batching models

Various practical scheduling problems include no-wait constraints between operations. These situations are studied in FS2 environments, where the two machines are continuously available and arranged in a pipeline fashion. That is, it may be necessary to delay the operation of a job on machine one to ensure that there is no waiting for the availability of machine two when its operation on machine one is finished. This necessity increases the complexity of finding an optimal schedule for the problem. This problem can be viewed as a variant of Johnson's classical F2 problem with n jobs, which can be solved optimally by a simple $O(n\log n)$ algorithm, see Pinedo (2008, 156). This algorithm finds an optimal schedule that minimizes the

makespan, i.e., the maximum completion time of all jobs, however, this solution admits an idle time between the two machines, therefore, it cannot be applied to FS2 with the no-wait constraint. The no-wait constraint naturally occurs in many production environments, such as, for example, in steel or plastic industries, in which no interruption or waiting is allowed during the processing of a job between successive machines. Recently, the FS2 no-wait problem was formulated for environments with batch processing assumptions. The interest of researchers was provoked, as usual, by practical necessities and theoretical challenges, but there is still a lack of publications on this subject.

In a basic variant with two s-batching machines, this problem is defined as follows. The jobs are grouped into batches before the processing. A constant setup time is incurred for preparing each batch on machine one. After the completion of a batch on machine one, all jobs contained in this batch must be immediately transferred to the second machine in a no-wait manner. A constant setup time is also incurred on machine two for these batches. The minimization of the makespan is the usual criterion. Following Graham's three-field notation, this problem may be denoted as $F2|nwt, batch|C_{\max}$, where *nwt* and *batch* represent the characteristics of the no-wait constraint and batch scheduling, respectively.

To explain the problem, an illustrative example is given below. It shows that the makespan length depends on both the batching and the scheduling decisions.

Example 6-2.
Let us consider a set of six jobs to be processed in an FS2 no-wait system with batching. The processing times are given in Table 6-4. A constant setup time $s = 1$ is assumed.

Job	**1**	**2**	**3**	**4**	**5**	**6**
$\boldsymbol{p_j}$	1	2	2	5	3	1
$\boldsymbol{q_j}$	3	4	1	2	2	2

Table 6-4. Processing times of the jobs.

In Fig. 6-6, the schedule S_1 contains the two batches $B_1 = \{2,4,6\}$ and $B_2 = \{1,3,5\}$, four setups, and we get $C_{\max}(B_1) = 25$, while the schedule S_2 contains three batches $B_1 = \{1,2\}$, $B_2 = \{3,4\}$, $B_3 = \{5,6\}$; six setups, and $C_{\max}(B_2) = 22$, i.e., three time units shorter than S_1. One can see that, although schedule B_2 has more setup operations, it is shorter due to smaller idle times on both machines.

The problem $F2|nwt, batch|C_{\max}$ is NP-hard, but it has polynomially solvable variants, for example, $F2|nwt, batch, p_i=q_i=p|C_{\max}$ and $F2|nwt, batch, p_i=p, q_i=q\ |C_{\max}$ (Lin and Cheng, 2001). A no-wait FS2 problem with makespan minimization as the criterion can be formulated as a restricted traveling salesman problem (TSP). Therefore, the Gilmore and Gomory algorithm can be used to solve the problem in polynomial time (Gilmore and Gomory 1964; Vairaktarakis 2003, 503–5).

In the paper by Oulamara (2007), a no-wait problem was also considered. In a comprehensible mode, it was denoted as $F2|$ *p-batch*(1), *s-batch*(2), *b, nwt*$|C_{\max}$. The motivation for the problem studied in that paper came from processing products in the iron and steel industry. During each working phase, the metal sheets first pass through a multi-head hole-punching machine, which simultaneously punches batches of holes according to their position on the sheet. Then, following a short preparation period, the sheet undergoes a series of sequential bending operations. This problem was shown to be NP-hard. Some properties of the model were studied.

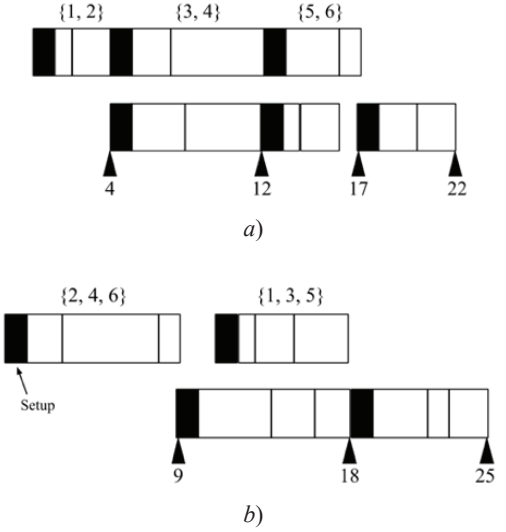


Fig. 6-6. Schedules with different batching solutions: *a*) schedule S_1; *b*) schedule S_2. Adapted from Lin and Cheng (2001, 616).

6.7 Cell architectures

A manufacturing cell in a flexible system consists of more than one machine assisted by a robot for loading and unloading of parts, and an automated material handling system. In general, the flexibility of the cells in a plant leads to benefits such as easy adaptation to product nomenclature changes, increased productivity, reduced inventory, reduced production cost, and improved quality. The manufacturing cell architecture has numerous aspects in its design and functionality. Many models for decision-making in flexible manufacturing are concerned with the key issue of the *resource management* due to the versatility and flexibility of the production, that is, how to optimally allocate and synchronize machining resources and parts in order to minimize the production costs, which is usually equivalent to maximizing the productivity. This objective leads to different models of the cell architecture, depending on the resource system at hand.

The selected architecture has a direct effect on the performance of the FMS through different issues. One of such moments is the presence of setups. In flexible manufacturing cells, each part is mounted at a fixed position and does not move until the processing machine, which is usually assisted by a robot, has completed all required operations. Significant time losses occur in situations when the processing resource (machining center, robot, and numerical control center) switches its operation mode, affecting the performance indexes. Nevertheless, a certain time must be allowed to a resource to interchange the tools before it starts working in a new mode.

Numerous methods have been proposed to solve the tool switching problem (Stecke and Kim 1988, 9; Co, Biermann, and Chen 1990, 2172–73; Crama 1997, 144–45; Agnetis, Lucertini, and Nicolo 1993, 104). Tool/part grouping problems are closely related to the cell formation (CF) problem in GT. Various methods, which are based on similarity coefficients, are described in Sections 5.6-5.7.

A general formulation of the *tool switching problem* was given by Crama (1997, 144) as follows: Determine a part input sequence and an associated sequence of tool loadings such that all tools required by the part j are present in the tool loading j, and minimize the total number of tool switches.

The tool switching problem is NP-hard for any fixed $C \geq 2$, where C is the capacity of the tool. The problem of scheduling jobs requiring exactly M tool setups, where M is the total number of tools needed to process all parts, is also NP-hard (Crama et al. 1994, 36). When all setup times are equal, i.e., when the objective is only to minimize the total number of switches, then the integer program can be solved by a greedy algorithm,

which turns out to be equivalent to the so-called *keep tool needed soonest* policy (KTNS) (C. S. Tang and Denardo 1988). In some situations, the number of batches (switching instants) is a more relevant performance criterion. This is the case when the setup time of the operations is proportional to the number of tool changeovers, or when the tool transportation system is congested (C. S. Tang and Denardo 1988, 769).

There are generally two different setup types concerning the resource handling in the batch machinery:

- *Part replacement*, which occurs when a finished part is removed and replaced by a new part;
- *Tool switch*, which happens when the machine switches from one operation type to another one, even if the same part is processed.

A part replacement setup takes usually more time than a tool switch.

A part replacement may be executed in two different ways named *FMS management policies*:

- *Batching replacement*: All parts (up to a total of k) currently being processed must be completed before new parts are loaded. A batch setup occurs whenever a new set of parts is loaded. A tool setup occurs at each switch of tools. The parts cannot be removed without stopping the cell operation;
- *Flexible replacement*: A part replacement takes place when it is completed. These setups occur only at the tool switches.

Modeling the part replacement requires to solve simultaneously two subproblems:

- Forming batches; each one contains at most k parts;
- Scheduling the tools and the robot moves, which are required by the execution program for each batch.

Using these definitions, the cell architecture can be described by the following assumptions.

There is a set of parts to be processed in a cell. At any time, up to k parts can be accommodated simultaneously in special fixtures. The quantity k is defined as the *degree of parallelism* of the cell. Each part requires a given sequence of the operations called *part program*, and each part operation requires a certain tool. The *length of a part program* is the number of operations in a part program. The same tool may be used by different

operations, even non-consecutive ones, in the same part program. The robot can hold only one tool at a time. When the robot is holding a given tool, same of the parts present in the cell may need this tool to carry out the current operation of their part program. In such a case, the robot performs all these operations before switching to a different tool. The tools are kept in a tool rack, which is accessed by the robot during the tool switch.

A *tool schedule* is the sequence of tools (possibly repeated), which are loaded by the machine when a given batch is processed. The *total time* required to process a batch is determined by the total duration of the operations plus the total tool switching time, which must be minimized. The *length of a tool schedule* is the length of the shortest tool schedule for this batch. Hence, for each batch, the scheduling subproblem consists of finding the batch length and the corresponding tool schedule.

Two illustrative examples for a robot, where any tool schedule is feasible, are given below to explain the tool scheduling problem. The examples are adapted from Agnetis, Alfieri, and Nicosia (2003, 89).

Example 6-3.
There are three tools *a*, *b*, *c*, and a batch *B*, which consists of three parts of the program: *P*1 = *abac*, *P*2 = *bacb*, and *P*3 = *abcb*. For instance, the tool schedule in the order *abcabc* allows to complete *P*1 and *P*3 but not *P*2. If the tools are scheduled in the order *abcabcb*, all part programs may be completed, but this requires six tool switches (Fig. 6-7 a). However, if the tools are scheduled in the order *abacb*, three part programs may be completed using only *four tool switches*. Therefore, the *schedule length* of *B* is *five* (Fig. 6-7 b).

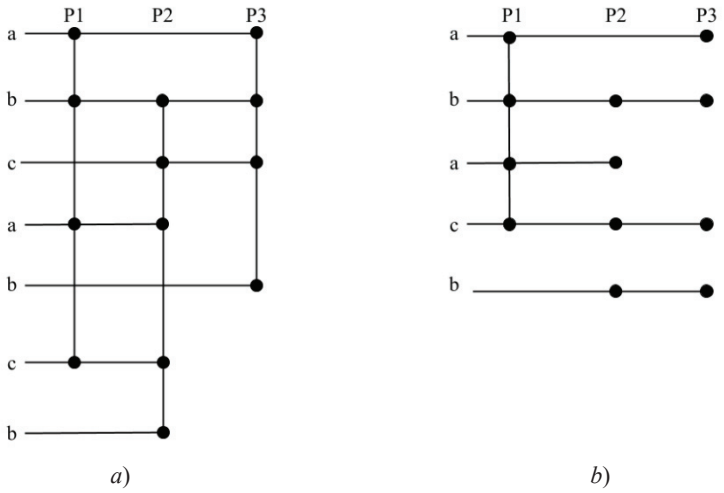


Fig. 6-7. Tool schedules: *a*) *abcabcb*; *b*) *abacb*.

The robot performs only one operation at a time, and the total duration of all operations is given. Therefore, the objective depends only on the amount of time spent for the setup activities (both batch and tool setups). The total batch setup time is equal to T_s (b - 1), where T_s is the batch setup time, and b is the number of batches. The total setup time is obtained by summing up the tool setup times of all batches.

Example 6-4.
Let us suppose that besides there are the batch B with the tooling as it is described in the previous example. Another batch B' formed by the part programs $P4$ = *bcba*, $P5$ = *caba*, and $P6$ = *bcab*. The length of B' is also five. It is obtained by the tool schedule *bcaba*. The overall time spent for setup activities during processing the batches B and B' is given as $T_s + 8t_s$, where T_s is the setup time used for loading a new batch, and t_s is the tool switch setup time.

The problem *Part Batching and Tool Scheduling* is NP-hard even for one processor and $k = 3$ parts presented in a flexible manufacturing cell at the same time (Agnetis, Alfieri, and Nicosia 2003).

6.8 Conclusions

A variety of problems, which were described in the literature for batching models, motivated the introduction of numerous new concepts and approaches. This attention to the effective organization of manufacturing industries with lot processing had inspired an evolution of the scheduling theory, and on the other hand, it implied a huge number of researches related to the study of new extensions and generalizations of classical scheduling problems. Therefore, we can conclude that the investigation of the batching production order introduced a big progress in the scheduling theory.

The main purpose to join similar lots into a batch is the reduction of lot changeover times and costs, because processing the jobs/lots as a batch may be cheaper or faster than processing the jobs individually. This simple idea gave rise to numerous aspects to be studied. In this chapter, the main points of the batching problematic were discussed. The theoretical batching models were explained and related practical applications were discussed. Various examples of the notation for these complex problems were given and explained.

The description and review of batching problems showed that basic environments, such as a single machine, FS2, HFS2 are mainly studied. Nevertheless, there is a lack of works, which deal with generalized environments. Job shop and open shop models, which consider s-batch and p-batch processing, are practically absent in the literature. In the last few years, there has also been an increasing interest in multi-criteria batch scheduling problems because of their great application potential, but there are only a few publications, which consider the batching assumption. The scientific literature on these addressed problems has a small number of contributions. Recent reviews are particularly absent. A deep study of batching problems is still ahead due to its practicality in real-life manufacturing settings and the versatility of the applications.

For the interested reader, there exist several reviews with a more detailed study of this subject. The first review by Uzsoy, Lee, and Martin-Vega (1992; 1994) gave the bases for studying the batching phenomenon in planning and scheduling models for the semiconductor industry as a fundamental area of applications. Rippin (1993) proposed a retrospective and perspective review of batch process systems, affecting the attention on economical and marketing related problems. Reklaitis (1996) defined the essential elements of batch scheduling and planning problems, which are characteristic for chemical manufacturing. Selim et al. (1998) discussed and reviewed a fundamental issue in cellular manufacturing, which is CF. The authors first provided a comprehensive mathematical formulation of the CF

problem and then proposed a methodology-based classification of prior research. Some directions for future studies are also highlighted. A classical review on scheduling with batching was given by Potts and Kovalyov (2000). Strict definitions of the principal concepts were introduced as well as the specifications of batching problems and main solution approaches. A literature review, a classification, and a simple meta-analysis on scheduling of batch processors in semiconductor manufacturing were proposed by Mathirajan and Sivakumar (2006). Williams and Magazine (2007) described heuristic approaches for batching jobs in a PCB assembly. Mönch et al. (2011) dedicated their survey to problems, solution techniques, and future challenges for scheduling in semiconductor manufacturing.

CHAPTER SEVEN

LOT STREAMING

By assuming that a batch can be split, we can handle the problems that occur due to dynamic nature of shop floor more elegantly.
(*Jeong et al. 1997, 781*)

In most multi-stage scheduling studies associated with processing discrete products, a *production batch* (lot) is treated as a single entity called *job*, which consists of only one part. As a result, a schedule cannot be improved anymore, even if there are plenty of idle times on the machines; that is, a partial transfer of completed items belonging to the same job between the machines is assumed impossible. Nevertheless, a much better solution can be obtained by considering alternative schedules, which allow the splitting of the current batch into smaller batches. Since the production lots are often large, the items, which are already processed on a machine, need to wait for a long time in an output buffer of this machine, even if the downstream machine becomes idle. It leads to elevated WIP inventories between the machines and extends the makespan. When the above assumption is relaxed in the corresponding scheduling problem, i.e., when it is supposed that a job can be split, it may be possible to improve the quality of the resulting schedule. Hence, the production leading times, the WIP inventory, the interim storage with its space requirements and the material handling system capacity requirements may be reduced.

The term "*lot streaming*" denotes techniques to split given jobs, each consisting of a number of identical items, into sublots to allow an overlapping during the processing on successive (downstream) machines in a *multi-stage* production system. This approach may be implemented in practice in a shorter time and at a lower cost than a structural reorganization.

In many practical situations, splitting a lot is both possible and desirable. Large lot sizes accumulate downstream at the process workstations. The lots must be divided into smaller transfer lots to avoid a blocking of the production process. Such situations occur often in electronic manufacturing, where many identical or very similar components are required to compose a final product. Nevertheless, making the lot size smaller, the number of shop orders to produce the same amount of parts is increased. This may cause

difficulties due to increased work loading and transportation costs. The computational complexity is also increased. Therefore, convenient methodologies are required to split a lot in an optimal way.

In this chapter, some related concepts are explained. The main results of the studies in this area are resumed from recently published papers. Various practical models are classified and described together with the solution ideas and methods.

7.1 Splitting jobs

The term *lot splitting* refers to breaking given lot into sublots of smaller size during the production (Baker and Pyke 1990, 475). In the scheduling models, two cases of splitting jobs may appear:

- Preemption: the interruption of the production run for a more urgent job is allowed;
- Lot streaming: overlapping operations on downstream machines are permitted.

In the second case, different sublots of the same job may be processed simultaneously at different stages. As a result of an operation overlapping, the production can be remarkably accelerated and consecutively, a reduction of the idle time on successive machines can be reached. *Lot streaming* is the process of splitting an entire production job (*process batch*) into sublots (*transfer batches*) and scheduling those sublots in an *overlapping* fashion, in order to accelerate the progress of an order in the production. A job is defined here as a production order (*lot*) composed of a number of identical items.

When a job is split into a number of sublots, a sublot can be processed on a machine even if the other sublots still have not been processed on the upstream machines. The sublots can be independently processed on several machines in order to finish the processing of all demands as soon as possible. The number *of sublots* and *the sublot size*, i.e., the number of items in each sublot, are the *decision variables*.

It is typically assumed that the number of items in a lot is large. Therefore, the lots can be treated as *infinitely divisible*. Ideally, the sublots are *integral*, but this version of the problem leads to the more difficult continuous versions (Potts and Baker 1989, 297).

One of the first attempts to derive optimal sublot sizes for one cyclic job with an overlapping operation was undertaken by Graves and Kostreva (1986, 285–89). The case with two successive workstations was considered.

It was a rather complex algorithm for a simple environment. Nevertheless, this work gave an impulse to study this subject.

Graves and Kostreva (1986, 285–285) gave the notion of *overlapping operations* as a kind of parallelism in processing. In traditional MRP system models, the items flow from a workstation to another workstation in the same batches. Overlapping operations occur when the transportation of partial batches to a *downstream* workstation is allowed while the work proceeds to complete the batch at the upstream workstation. The main *advantage* of splitting jobs into a sublot to be gained, among others, is the reduction of the following indexes:

- Total lead time;
- WIP inventory;
- Size of transfer vehicles;
- Timely delivery to the customers.

First, splitting jobs may improve the customer service; each sublot can be delivered to the customer immediately upon completion, without waiting for the remaining sublots of the same job to be processed. Another motivation for splitting jobs in multi-stage production systems is to enable various operations of the same job to be overlapped (by allowing the processing of downstream operations for any sublot immediately after being processed at the current stage.)

The advantage of lot streaming in production planning and scheduling systems is also evident in the context of synchronous manufacturing. Nevertheless, additional costs may accrue due to costs of the transportation of the partial batches and due to the costs of the control.

The size of a scheduling problem may become too large to be solved within a practical time limit. Unexpected dynamic events, such as a machine failure, rush order and require simple dispatching rules analogously to the traditional scheduling policies to get an improved schedule by splitting the original lot into smaller lots, and thereby to meet the due date requirements. Jeong et al. (1997, 782) proposed four specific *batch/lot splitting rules* for a job shop scheduling problem. However, they are also applicable to other environments. These rules are:

- Minimum Batch Size (MBS);
- Required Batch Size (RBS);
- Full Batch Size (FBS);
- Equal Batch Size (EBS).

The *MBS* rule assigns the minimum possible batch size as the batch size of the first small batch being created by splitting. Adopting this strategy implies intentionally increasing the batch size for later small batches. The makespan can be reduced by splitting a batch as many times as possible. Therefore, one expects a better performance by taking this rule while there is less idle time at later stages than at an earlier stage.

The *RBS* strategy assigns the required batch size dynamically as the batch size of the first batch being created by the split. When using this strategy, an acceptable solution by a rather small number of splits is obtained.

The *FBS* strategy splits an entire batch into batches, which has a processing time approximately equal to the idle time of the machine just before processing the operation of the job. This strategy may also work better if there are more idle times at later stages compared to the MBS strategy.

The *EBS* strategy consists in splitting a batch into equal sized two batches. When using this strategy, many precedence relationships are set in advance. This strategy leads to a considerable reduction of the computational efforts to find a solution.

7.2 Lot streaming problem

The *lot streaming problem* is to determine:

1. The optimal *number of sublots* for each job;
2. The optimal *size of each sublot;*
3. An optimal *sequence* for processing the sublots, such that the production lead time is minimized. It combines lot sizing and scheduling decisions, which are traditionally treated separately.

The concept and practice of lot streaming are not new. The lot streaming concept was first formally introduced by Reiter (1966, 374) for the cases when certain operations with extremely long total processing times are followed by operations with short processing times. According to his definition, each work is restricted by a critical total processing time k (TPT). If the TPT of an operation exceeds its critical value k, the lot is split into TPT/k+1 sublots, each of which is scheduled separately in succeeding operations. The start time of the next operation of a sublot is the completion time of the last piece in that sublot in the current operation. Sublots may be further streamed at subsequent operations. The completion time at any work center is the sum of the following terms, which are calculated using the

adjusted lot size: the completion time at the preceding work center, the transportation time, the delay, the setup time and TPT. This concept and related methods were systematically considered in the scheduling literature since 1980 for the production management in big manufacturing plants such as Toyota.

In general, the makespan can be optimized if there is just one item in each sublot (Vickson and Aldredsson 1992, 1552). However, there may exist practical considerations when it is undesirable to have a large number of unit-sized sublots. It may also be possible to attain the minimum makespan with fewer sublots, or it might be difficult in tracking a large number of small sublots. Therefore, the lot splitting problem finds a *compromise* between the sizes of the production batch and the sublots when setups are long and difficult.

An example of lot streaming benefits is shown in Fig. 7-1. There is a three-machine flow shop, a single job with the processing times 6, 3, and 6 time units on the machines M1, M2, and M3, respectively. When the job is not split into sublots, the job completion time is 15 (Fig. 7-1a). When the job is split into three sublots and a *no-idling* production interruption time is allowed between any two adjacent sublots, the job completion time is reduced to 11 (Fig. 7-1b). In the case when an *intermittent idling* is allowed between the sublots, the job completion time is further reduced to 9 (Fig. 7-1c). Obviously, the completion time of a job under the idling case is shorter than the one under the no-idling case with the same sublot sizes. However, there are many practical applications of lot streaming flow shop scheduling under the no-idling assumption.

A comprehensive description of the lot streaming problem is given by Baker and Jia (1993, 562) as follows.

A production lot containing Q units requires the processing on m machines in a sequence. The processing time per unit on machine i is p_i. Therefore, if lot streaming is not applied, the makespan for the lot would be $Q\sum_{i=1\ldots m} p_i$. In this case, there is one transfer between the machines i and (i+l). This transfer occurs at the time when the processing completes on machine i. In contrast, in the lot streaming problem, the processing on machine i is broken down into n separated sublots. Every sublot is transferred individually to machine (i+1), each one at the time its processing completes. Consecutively, there are n transfers between each pair of machines.

Let C denote the makespan of a schedule, Q_{ij} the size of the sublot j on machine i, and let L be the entire lot size. The variables Q_{ij} can be fractional, or equivalently, the original size L of the lot is infinitely divisible. In the majority of real problems, Q_{ij} are integer values, although this requirement

might be met simply by rounding the fractional solution. In the most general version of the lot streaming problem, the variables Q_{ij} can be assigned freely conserving the condition $\sum_{j=1\ldots n} Q_{ij} = L$. This case is referred to as *variable* sublots. If $Q_{ij} = Q_{1j}$ for a sublot j on all machines $i = 1,\ldots, m$, then the sublots are defined as *consistent*.

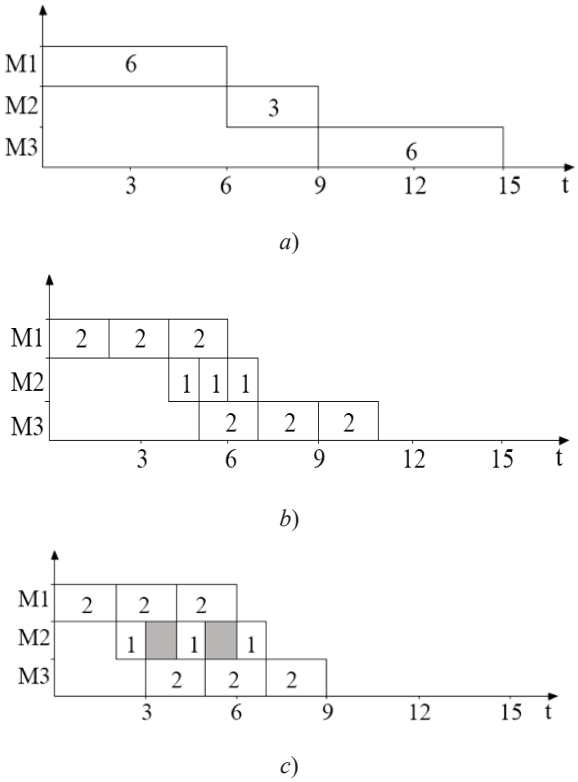


Fig. 7-1. An example of the lot streaming effect in the solution of a FS3 scheduling problem: *a*) without sublots; *b*) with sublots under the *no-idling* case; *c*) with sublots under the *intermittent idling* case. Adapted from Pan et al. (2011, 2457).

If the sublots are consistent, their integrity is preserved throughout the schedule. An item, which is produced as a part of sublot j on machine 1, is a part of sublot j on all other machines. In addition, all sublots j may be fixed or identical, i.e., $Q_{ij} = Q$, $i = 1,\ldots, n$.

A lot streaming problem may include several restrictions permitting or avoiding breaks in the processing on a machine. It may be desirable for machines to process a work intermittently, with some *idleness* between certain pairs of sublots. Other cases include a *no-idling* assumption, which means that each machine must operate continuously, once it begins processing. It may be due to the specific of the manufacturing process, for example, the parts must be processed quickly to avoid cooling or chemical deterioration.

Studying the m-stage flow shop lot streaming, the *inverse problem*, in which the processing time on machine i is p_{m-i+1} for $1 \leq i \leq m$, can also be considered. It was observed that a lot streaming flow shop problem and its inverse problem are equivalent. Therefore, for $m = 2$ and $m = 3$, there exists an optimal schedule with consistent sublots. The inverse problem is also used to study the properties of the original problem (Potts and Baker 1989; Glass, Gupta, and Potts 1994, 380).

7.3 Model structure and notations

There exist some variations in the formal descriptions of lot streaming models. These variations contain specific constraints, which are imposed on the scheduling decisions. These constraints reflect several features of the production technology, which must be taken as given, or they reflect simplified assumptions, which are made for an analytic treatment.

Consider N jobs to be processed on a machine. Each job j, $j = 1,\ldots, N$, becomes available for processing at time zero and requires a processing time p_j. Let job j contain q_j identical items. One item requires a processing time of p_j/q_j units. Three *splitting models* can be specified as follows:

- *Item completion time model*, in which item i is deemed to be completed immediately after its processing is finished;
- *Sublot completion time model*, in which item i is completed only when the processing of sublot k is finished;
- *Job completion time model*, in which item i is completed only when the processing of job j is finished.

The N jobs can be partitioned into k sublots in different manners in a schedule. The models, in which lot streaming is allowed, assume a broad

variety of the characteristics compared with traditional scheduling to specify the model under study. The principal characteristics are given below.

Due to the specific of lot streaming problems, some authors proposed original classification schemes to denote a problem, while the highlighted characteristics are mainly the same in all variants. Nevertheless, any classification system should be understandable. Trietsch and Baker (1993, 1068) used the following scheme to classify a lot streaming problem:

> {Number of machines}|{Whether sublots are variable (*V*), consistent (*C*) or equal (*E*)}|{Whether there is intermittent idling (*II*) or no-idling (*NI*){Whether the continuous version (*CV*) or discrete version (*DV*) is applied}.

The following examples illustrate the use of this scheme:

> 3|*V*|*II* – denotes a 3-machine problem with variable sublots and no-idling, without specifying whether the solution has to be continuous or discrete.
> 3|*E*|*NI*|*CV* - denotes a 3-machine problem with equal sublots and no-idling in its continuous version.

This scheme has several limitations. For example, it does not include a detailed description of the machine environment and the objective. Moreover, it does not specify a problem as a lot streaming one.

The version by Cheng, Mukherjee, and Sarin (2013, 1–2) looks complete but it is also very specific. The authors proposed a modification of the traditional 3-field classification scheme to a 9-field one to describe a problem with lot streaming as follows:

> {Machine configuration}{Number of machines}|{Number of product types}|{Sublot type}|{Idling}|{Number of sublots}|{Sublot sizes}|{Setups}|{Transfer or removal} |{Objective function}

Nevertheless, the traditional triplet $\alpha|\beta|\gamma$ by Graham et al. (1979) remains to be dominant (see Section 3.1), of course updated with specific features (Potts and Van Wassenhove 1992, 396–99; Potts and Kovalyov 2000, 229–31):

System configuration (α-field):

The shop environment is denoted as usual. There can also be specified the *number of product types*:

- *Single product* (1);
- *Multiple product types* (*n*).

Sublot-related features (β-field):

1. Depending on the sublot size, there are two *versions* of lot splitting with respect to the divisibility of the sublot size:

- *Discrete sublots (DV)* - the processing time for each sublot is p_j/q_j times multiplied by the (integer) number of items in a sublot;
- *Continuous sublots (CV)* - any split of the processing time p_j defines the sublots of job *j*. In this case, a sublot with processing time *p*, where $0 < p \leq p_j$, contains pq_j/p_j items, independently whether this quantity is an integer or not.

In the *discrete version*, the sublot sizes are integers that correspond to discrete numbers of units in each sublot. Such a problem can be formulated as an integer linear program. In the *continuous version* of the problem, the integer restrictions are relaxed and the sizes of sublots are real-valued. Such a routing may be acceptable if the lot size is large and the number of sublots is small. The optimal makespan in the continuous version serves also as a lower bound on the optimal makespan in the discrete version, and the makespan produced by rounding the continuous solution serves as an upper bound.

2. In general, the transfer lots between the machines *m* and (*m*+1) may differ from the transfer lots between the machines (*m*+1) and (*m*+2). So, a model may have one of the following *sublot types* (Trietsch and Baker 1993, 1067–68; Biskup and Feldmann 2007, 199):

- *Fixed* (*F*) - all sublots for all products consist of an identical number of items at all stages;
- *Equal* (*E*) – the sublot sizes are fixed for each product/lot;
- *Consistent* (*C*) – the sizes of the sublots remain the same on all machines;
- *Variable* (*V*) - no restrictions are given on the sublot sizes. They can change from machine to machine.

The distinction between fixed and equal sublots is only necessary for multiple product models. When the size of sublots is variable, it can change from machine to machine.

The restriction to consistent sublots does not affect the optimality of the makespan. An optimal solution for the lot streaming problem can be obtained by using consistent sublots when there are *two* or *three* machines. For *larger problems*, this restriction may increase the optimal makespan. An additional restriction imposed that all sublots are of a consistent type and *equal* size may increase the optimal makespan. An intermediate variant between consistent and variable sublots, where lots are allowed to vary, is also possible but with restrictions. This version yields an intermediate makespan.

The variable sublot case is dominant over the consistent sublot case, which is *dominant* over the equal sublot case. This means that a model with variable sublots should have a shorter makespan than the makespan of the same model with consistent or equal sublots, or an equal makespan.

3. When an *idling* requirement is imposed to a machine, once started, the entire lot must be processed continuously without any idling:

- *No-idling* (*NI*) – the sublots at a particular stage have to be processed directly one after the other;
- *Intermitted idling* (*II*) – idle times between sublots may occur.

This restriction when imposed on the first machine does not affect the optimal makespan. Symmetrically, it does not affect the last machine. Consecutively, in a two-machine problem, the no-idling restriction does not affect the optimal makespan. When imposed on intermediate machines in larger environments, however, this restriction may increase the optimal makespan (Fig. 7-1 b, c).

It is clear that *II* dominates the *NI* case, and *CV* dominates the *DV* case. According to the dominance relationships, the least restrictive model is $m|V|II|CV$.

4. Inter-stage *waiting* conditions may be defined:

- *No-wait* (*NWT*) - each sublot has to be transferred to and processed at the next stage immediately after it has been finished at the preceding stage;
- *Wait* (wait) - a sublot may wait for processing between consecutive stages.

This requirement can be found in many practical situations, for example, in chemical and food industries. Particularly, in the metal processing industry, metal is rolled while it is hot. Therefore, delays between operations are prohibited.

5. *Setup* requirements can be imposed to a whole lot and also to sublots:

- *Lot/Sublot-attached setup* ($L(a)$/ $S(a)$) - a setup required to process a lot/sublot on a machine can start only after the arrival of the lot at the machine;
- *Lot/Sublot-detached setup* ($L(d)$/ $S(d)$) - a setup can be performed on a machine even before the arrival of the lot to that machine.

6. The *number of sublots* can be indicated. There are two options:

- *FixN* - known a priori or
- *FlexN* - is to be determined.

In general, even if there is just one item in each sublot, the makespan will be minimized. Nevertheless, there may be practical considerations that make it undesirable to have a large number of unit-sized sublots. It may be possible to attain the minimum makespan with fewer sublots, or there may be difficulties in tracking a large number of small sublots. Thus, the basic lot streaming problem specifies the number of sublots. Alternatively, there may be a constraint on the availability of the transporting equipment. For more than two machines, however, the problem of limited transport equipment is difficult to solve.

7. An *availability* option during the transfer or removal operations can be indicated as follows:

- *Transfer time* (T) - refers to the time required to move a lot or a sublot from one machine to another one. During this time, the machine is available for processing the next lot or sublot;
- *Removal time* (R) - refers to the time required to remove a lot or sublot from a machine, but during this time the machine is not available to process the next lot or sublot.

8. In a multi-product problem, a *change in the sequence* of sublots can also be set:

- *Intermingling* sublots - the sequence of sublots of product *j* may be interrupted by sublots of product *k*;
- *Non-intermingling* sublots - no interruption in the sequence of sublots of a product is allowed.

Performance measures (γ-field):

Having specified the model, the completion time C_i and other performance measures of each item *i* may be easily found for any schedule, for example:

$L_i = C_i - d_i$ - lateness of item *i*;
$T_i = \max\{C_i - d_i, 0\}$ - tardiness of item *i*.

Possible objectives to be minimized are, among others:

- Maximum completion time (makespan) $C_{\max} = \max_i\{C_i\}$;
- Maximum lateness $L_{\max} = \max_i\{L_i\}$;
- Total (weighted) completion time $\sum_i w_i C_i$;
- Total (weighted) tardiness $\sum_i w_i T_i$.

In some special cases, the makespan value $C_{\max}$ can be easily calculated. The following three formulas correspond to three well-solved versions of the lot streaming model. Each solution holds for any number of machines and any number of sublots, but restrictive assumptions (equal sublots, no-idling or both) are required in each case. The respective formulas are as follows.

Equal sublots (*E*):

$$C = \frac{L}{n}\left[\sum_{j=1}^{m} p_j + (n-1)p_{\max}\right],$$

where $p_{\max}$ denotes the largest p_j value.

Equal sublots and no-idling (*E*, *NI*):

$$C = \frac{L}{n}\left[\sum_{j=1}^{m} p_j + \left((n-1)\sum_{j=1}^{m}(p_j - p_{j-1})\,\delta(p_j - p_{j-1})\right)\right],$$

where

$$\delta(x) = \begin{cases} 1, \text{if } x > 0 \\ 0, \text{if } x = 0 \end{cases}.$$

No-idling (*NI*):

$$C = L\left[p_m + \sum_{j=1}^{m}\left(\frac{p_j - p_{j+1}}{1-(p_j/\,p_{j+1})^n}\right)\right].$$

In the case when $p_j = p_{j+1}$, the fraction is substituted by p_j/n.

Illustrative examples of the triplet notations for notations for problems with lot streaming are given below:

$F3, 2|s_f|C_{\max}$ is the three-machine flow shop problem of scheduling two families of jobs, where each job contains a single item. Sequence-independent family setup times are only necessary when the machine switches to the processing of jobs from a different family. The objective is to minimize the makespan.

$F2|q_j(DV), NI| L_{\max}$ is a two-machine flow shop scheduling problem, where each job contains several items. The jobs may be split into discrete sublots. Sublot completion times and a no-idling order are assumed. The objective is to minimize maximum lateness.

Independently of the classification system used, the notation must be comprehensible.

7.4 Problems with transfer batches

The contributions of lot streaming theory to the environment are given in recent publications. These studies, which consider transfer batches, can be categorized into two main groups:

- *Single-job* lot streaming, which includes decisions over the *sublot sizing*;
- *Multiple-job* lot streaming, which includes simultaneous decisions over the *sublot sizing* and the *job sequencing*, given various job and shop characteristics.

The problem statements and applied approaches are analyzed below for the principal shop environments, which are grouped into three parts:

1. Flow shops;
2. Flexible and hybrid flow shops;

3. Job shops and open shops, including their flexible and mixed variants.

The influenced characteristics of lot streaming are highlighted for every problem. Various practical algorithms and rules are described.

7.4.1 Flow shops

The literature on lot streaming is limited. Most studies focus on problems in pure flow shops with a number of machines in series. The first models considered are two machines or special cases of three machines, only sometimes with more than three machines (Potts and Baker 1989; Trietsch and Baker 1993; Baker and Jia 1993);

The first formal results on lot streaming were obtained by Potts and Baker (1989) dealing with the *one-job* case in a flow shop with *two* and *three stages*. The problem of minimizing the length of a schedule with *consistent* sublots of *equal* sizes was studied. This apparently easy problem is difficult to solve, even when a job is infinitely divisible. The authors showed that it is sufficient to consider consistent sublots, which have the same size at both stages, to minimize the length of a flow shop schedule. It was also indicated that an optimal policy with consistent sublots can always be found for $m \leq 3$; otherwise, it may be desirable to allocate a job differently at different parts of the schedule. Some difficulties involved in the n-job problem were also demonstrated. In the concluding part of the paper, the authors addressed the problem of lot streaming with *two products* at *two stages*. They give a counterexample to show that it is not possible to solve the n-job problem simply by applying lot streaming individually to each job in an optimal schedule, which has been created without lot streaming. However, Potts and Baker did not present a general solution procedure for lot streaming with multiple products.

Glass, Gupta and Potts (1994) extended the results of Potts and Baker (1989) by solving the flow shop lot streaming problem for the *one-job* model in *three-stage* production processes, including *flow shop*, *job shop* and *open shop* environments. At each stage, the job was to be partitioned into s sublots. There were no setup activities between the sublots. The objective was to find the sublot sizes, for which the makespan is minimized. The sublots were not constrained to contain an integer number of items. A *critical path* algorithm was proposed. It finds the minimum makespan in $O(\log s)$ time.

Trietsch and Baker (1993) considered *one-job* models with continuous and *discrete* sublot sizes, models *with and without intermittent idling* of

machines, and models with *consistent* and *variable* sublots. The authors introduced a model with *limited transporter capacity*, which looks for a compromise between the lot size and the capacity of the transporter. Some studies of the computational complexity were presented. It was shown that some *m*-machine versions, such as $m|V|NI|CV$, $m|V|NI|DV$ and $m|C|CV$ are polynomially solvable (see Section 7.3 for the classification system by Triesch).

Vickson and Alfredsson (1992, 1557–58) proposed a *modified Johnson algorithm* to solve the *multiple-job*, *two-machine* lot streaming scheduling problem *without setup times* to minimize the *makespan*. In this algorithm, every item of a job is a separate transfer batch. Using the notation system by Rinnooy Kan (1976, 28–29), the problem was classified as $N|2|F$, $t.b.|C_{\max}$, which means an environment with N jobs, a two-machine flow shop with transfer batches and $C_{\max}$ as the objective. In ordinary form, this means: $K|2|F|C_{\max}$.

The modified Johnson algorithm is as follows:

The total number of items of all jobs is
$K = \sum_{i=1}^{N} n_i$.

All K transfer batches are scheduled, one-by-one, using Johnson's algorithm. If job i satisfies

$$a_i = \min_{j=1}^{N}\{a_j, b_j, \},$$

any transfer batch of job i can be scheduled at the first position, according to Johnson's algorithm. Since n_j transfer batches satisfy this minimality condition, another transfer batch of job i is scheduled as the second one, then another is scheduled as the third one, and so on. This means that there is an optimal solution for problem $K|2|F|C_{\max}$, in which every transfer batch of job i precedes the transfer batches of all jobs $j \in S\text{-}\{i\}$. Similarly, if there is a job i satisfying the condition

$$b_i = \min_{j=1}^{N}\{a_j, b_j, \},$$

there is an optimal sequence in which every transfer batch of job i follows the transfer batches of all jobs $j \in S\text{-}\{i\}$. Continuing in this way, an optimal solution for problem $K|2|F|C_{\max}$ is found, in which the transfer batches of each job i are sequenced in an unbroken string. The optimal job sequence is

given by Johnson's algorithm for the ordinary problem $N|2|F|C_{max}$ with the processing times (a_i, b_i) on (M_1, M_2).

The following example illustrates the idea that for a given set of jobs, a different optimal sequence may result dependent on whether transfer batches are used or not.

Example 7-1.
The data are given in Table 7-1. Without transfer batches, Johnson's algorithm applied to the processing times (A_i, B_i), gives the optimal sequence J_2J_1 with the makespan value 35 (Fig. 7-2a). With transfer batches, Johnson's algorithm applied to the processing times (a_i, b_i), gives the optimal sequence J_1J_2 with the makespan value 30 (Fig. 7-2b). The makespan is improved due to the transfer batches themselves rather than by different sequences. Note that the sequence J_2J_1 with transfer batches has the makespan value 31.

i	a_i	b_i	n_i	A_i	B_i
1	3	5	3	9	15
2	4	6	2	8	12

Table 7-1. Processing times of the jobs on the machines. Adapted from Vickson and Alfredsson (1992, 1558).

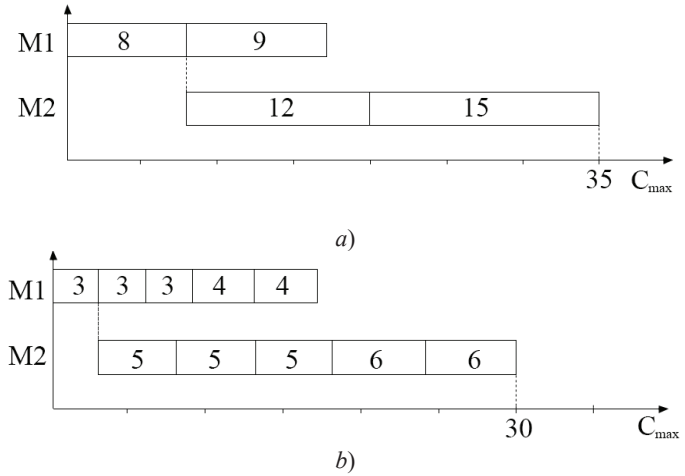


Fig. 7-2. Optimal sequence *a*) without job splitting; *b*) with job splitting.

Yoon and Ventura (2002) studied a *multi-job m*-machine flow shop lot streaming problem with the objective of minimizing the *mean weighted absolute deviation from the due dates* when the jobs are already scheduled. This objective emphasizes a timely delivery, what is consistent with the JIT philosophy. For a given sequence, the insertion of idle times between sublots and between jobs may improve the objective function value in some cases. Thus, the solution methodology includes two stages:

- Sequencing of the jobs;
- Insertion of idle times between sublots and between jobs.

The authors presented two linear programming formulations for different lot streaming flow shop problems to find the optimal sublot completion times considering a given job sequence. These formulations are as follows.

1) The equal-size sublot problem for the following cases:
   - Infinite capacity buffers between successive machines;
   - No-wait flow shop;
   - Limited capacity buffer;
   - Blocking is allowed;
   - Blocking is not allowed.

2) Consistent sublot problem.

A *pairwise interchange* (PI) *method* was applied to generate initial sequences. This method provides a near-optimal solution in reasonable time (Ding 1990; Della Croce 1995). The basic elements of the PI method are:

1. An initial sequence;
2. A neighborhood search mechanism;
3. A rule of selecting a particular sequence in the neighborhood to be the new sequence.

The four rules used to generate the initial sequences are based on the EDD, the smallest slack time on the last machine (LSL), the smallest overall slack time (OSL), and the smallest overall weighted slack time (OWSL) policies.

With these rules, the initial sequence $\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(n)\}$ is obtained as follows:

For $\sigma = 2,\ldots, n$,
EDD: $d_{\sigma(u-1)} \le d_{\sigma(u)}$;
LSL: $d_{\sigma(u-1)} - p_{m,\sigma(u-1)} \le d_{\sigma(u)} - p_{m,\sigma(u)}$;
OSL: $d_{\sigma(u-1)} - \sum_{i=1}^{m} p_{i,\sigma(u-1)} \le d_{\sigma(u)} - \sum_{i=1}^{m} p_{i,\sigma(u)}$;
OWSL:

$$\alpha_{\sigma(u-1)}\max\left\{0, d_{\sigma(u-1)} - \sum_{i=1}^{m} p_{i,\sigma(u-1)}\right\} + \beta_{\sigma(u-1)}\max\left\{0, \sum_{i=1}^{m} p_{i,\sigma(u-1)} - d_{\sigma(u-1)}\right\} \le$$

$$\alpha_{\sigma(u)}\max\left\{0, d_{\sigma(u)} - \sum_{i=1}^{m} p_{i,\sigma(u)}\right\} + \beta_{\sigma(u)}\max\left\{0, \sum_{i=1}^{m} p_{i,\sigma(u)} - d_{\sigma(u)}\right\}.$$

The neighborhood search generates a collection of corresponding sequences by making successively small changes. Four neighborhood search algorithms were used to improve a job sequence:

1. An adjacent pairwise interchange (API), switches two adjacent jobs in a sequence;
2. A non-adjacent pairwise interchange (NAPI), does not consider switches if two jobs are adjacent;
3. An extraction and forward shifted reinsertion (EFSR) chooses two jobs and moves a job in the front just after another job;
4. An extraction and backward shifted reinsertion (EBSR) chooses two jobs and moves a job in the front just before another job.

The new sequence is recorded when the objective function value of the best sequence obtained by a neighborhood search algorithm is less than that of the current best sequence. This process continues until there is no improvement of the objective function value, which was obtained by solving the LP problem. It was confirmed by a computational experiment that the use of consistent sublots improves the solution of the equal-size sublot lot-streaming flow shop problem. It was also concluded that the use of infinite capacity buffers improves the solution of the no-wait flow shop problem. In the case of negligible material handling costs and infinitely divisible jobs into sublots, the use of consistent sublots has to be considered.

Çetinkaya (1994) examined an integrated transfer batch sizing and scheduling problem for *two-stage* flow shops. In this problem, the changeover times are independent of the processing times. The makespan is taken as the measure of performance. Both *single-job* models and *multi-job* models were considered. A polynomial time algorithm was presented for

problem $N|2|F$, tb, S_{nsd}, $R_{nsd}|C_{\max}$ in the classification scheme by Rinnooy Kan (1976, 28–29), where N is the number of jobs at the two stages; tb, S_{nsd} and R_{nsd} denote the transfer batches, separable and sequence-independent setup and removal times, respectively. The algorithm gives an optimal solution for this integrated problem of transfer batch sizing and scheduling.

The *multi-job* problem by Feldmann and Biskup (2008) involved splitting given orders of different products into sublots and determining their optimal sequence for the M-machine flow shop. Each sublot has to be processed successively on all machines. An important detail is that the sublots of particular products are allowed to *intermingle*, that is, sublots of different jobs may be interleaved. A MIP formulation was given, which simultaneously determined the lot sizes and the sequence of the sublots to guarantee overall optimal solutions. However, the authors have been only able to solve problems with two or three products and up to seven sublots per product to optimality in a reasonable time. The model was extended to deal with different settings and objectives.

Mortezaei and Zulkifli (2013) integrated *lot sizing* and *lot streaming* with *multiple products* into a *flow shop* scheduling problem. A mixed integer programming formulation was presented. It was able to find optimal production quantities, optimal inventory levels, optimal sublot sizes, and an optimal sequence simultaneously. Numerical examples were given to test eight different lot streaming problems:

1. Consistent sublots with intermingling;
2. Consistent sublots and no intermingling between the sublots of the products (without intermingling);
3. Equal sublots with intermingling;
4. Equal sublots without intermingling;
5. No-wait consistent sublots with intermingling;
6. No-wait equal sublots with intermingling;
7. No-wait consistent sublots without intermingling;
8. No-wait equal sublots without intermingling.

It was shown that consistent sublots with the intermingling case achieved the best makespan value.

The paper by Mukherjee and Sarin (2014) was the first one found, which considers a *learning effect* in the context of the lot streaming problem. The case deals with the situation, where the processing time required for a job depends on the number of previously processed identical items. As a result, the time required to process a production lot continuously decreases.

The idea to solve this class of problems was to transform a general lot streaming problem, in which the processing times were influenced by a general learning curve, into an equivalent problem without learning, for which a solution method may already exist. The use of this transformation was demonstrated on some basic lot streaming problems with consistent lot sizes. Later, Mukherjee, Sarin, and Singh (2016) used this method to solve a *two-stage flow shop* lot streaming problem, in which the learning effect was observed in the processing times, sublot-attached setup times, or both. The objective was to determine the number of sublots and the sublot sizes in order to minimize the makespan. Closed-form expressions for optimal sublot sizes and efficient search schemes to determine the optimal number of sublots were presented.

The *learning* effect in a lot streaming problem was also studied by Fattahi, Azizi, and Jabbari (2015). A *no-wait multi-product M-machine flow shop* with *sequence-dependent setup times* and *position-based learning factors* was considered. The last restriction was interpreted as follows.

If the sublot *r* of job *i* is put at position τ on machine *m*, its required processing time p_{im}^{τ} is calculated according to the *truncation learning formula*:

$$p_{im}^{\tau} = p_{im}^{\tau}{}' * \max\{\tau^{\alpha_m}, \beta_m\},$$

where

$p_{im}^{\tau}{}'$ - actual process time of the product *i*;
α_m - learning effect parameter for machine *m*;
β_m - control parameter limiting the learning parameter, which prevents that the processing time becomes zero.

The objective was to minimize the makespan. A mathematical model for the problem was presented.

Two hybrid metaheuristics were proposed. These metaheuristics were based on variable neighborhood search (VNS). The VNS starts from an initial solution and proceeds with local changes in order to improve the value of the objective function. The neighborhood structure is a mechanism leading to a new set of neighboring solutions. This structure eliminates unnecessary moves. In this algorithm, three neighborhood structures were used:

- The first structure randomly selects and exchanges two elements in the current solution (*swap*);

- The second structure chooses an element and put it just before another randomly determined element (*insert*);
- The third structure reverses the elements, which are located between two randomly chosen elements (*reversion*).

The search procedure is continued until the stopping condition is met. The exploration structure of VNS was embedded into simulated annealing and tabu search. These metaheuristics enhance the performance of VNS.

7.4.2 HFS

Lot streaming is very useful for many practical production systems, which can be modeled as an HFS. However, such models have received very limited research attention. The more important papers are analyzed below.

One of the first works, which were dedicated to the HFS lot streaming problems, was a paper due to Li (1997), which dealt with a typical *two-stage* 1+*m* HFS with part families and batch production, where a lot is allowed as *split and as no split at stage two*. The parts within a family share a *major setup* and the parts in a batch share a *minor setup*. The objective is to minimize the *makespan*. The problem studied was found in the blade line in the airplane engine plant Pratt and Whitney, Inc. (PWI).

Two allocation policies were developed, *forward heuristic* (FH) and *backward heuristic* (BH), which were based on two critical issues:

1. To determine an optimal process sequence of the parts at stage 1;
2. To allocate the parts to the machines of stage 2.

Li adapted the SPT and LPT rules and developed three useful sequencing AVG rules for this problem. These rules are as follows:

For all sequencing rules, the lots in every family were first sequenced in an ascending order of the lot processing times plus their minor setup times.

SPT1: The parts are sequenced in an ascending order of the family processing times including the major, minor, and lot processing times at stage 1. This allocation rule is very popular both in practice and in the reported literature. The drawback of this rule is that it does not consider the processing times at stage 2.

SPT2: The parts at stage 1 are sequenced in an ascending order of the family processing times including the major, minor, and lot processing times at stage 2. It makes sense to sequence the long jobs at the end and the short jobs at the beginning.

LPT2: The parts at stage 1 are sequenced in a descending order of the family processing times including the major, minor, and lot processing times at stage 2. This rule sequences the families by their family processing times, but it does not consider the lot processing times and the number of lots in the families.

AVG1: The average processing time per lot for each family at stage 2 is calculated. Then the family is sequenced in a descending order of the ratios. A family that has a significantly long processing time may have many lots, each of which has a very short processing time. On the other hand, a family that has a relatively short processing time may contain only one lot.

AVG2: The ratios of the total family processing times at stage 2 over the total family processing times at stage 1 are calculated. Then the families are sequenced in a descending order of the ratios. The families with large ratios are scheduled at the beginning of the sequence so that the machines of stage 2 will be busy at the beginning. With this rule, the machines of stage 2 are utilized more effectively. The logic of this rule is similar to Johnson's algorithm.

AVG3: The ratios of the unit processing times at stage 2 over the unit processing times at stage 1 are first calculated and then summed up over the number of lots in a family. The sum is then divided by the number of lots to obtain the average ratios. Then the families are sequenced in descending order of the ratios. This rule considers both the ratio of the processing times between the two stages and the number of lots in each family. Therefore, any family with a large ratio will be given a higher priority. The idea of this rule is also to sequence short jobs at the end of the sequence.

These rules are very practical and possess a generality to be applied to similar problems. The AVG rules showed also a better performance than the traditional sequencing rules. A mathematical interpretation was given for all these rules in the paper.

Zhang, Liu, and Linn (2003) and Zhang et al. (2005) studied a *discrete* version with *integer* lot sizes of the m+1 HFS2 lot streaming problem with m identical machines at the first stage, and one of the stages is obviously a *bottleneck*. The size of each sublot is kept *consistent* at the two stages. A job can be continuously *divisible*. The problem is then to decide the number and sizes of sublots for each job and to schedule these sublots, so as a given objective is minimized. In the paper by Zhang, Liu, and Linn (2003), the

single-job version of the problem was solved. The objective was to minimize the *makespan*. Zhang et al. (2005), dealt with the *multi-job* version of the problem. The objective was to minimize the *mean completion time* of the jobs. In both works, an efficient solution was obtained for the *equal* sublot version using a similar approach. The general problem was formulated as a MILP model and solved using *two heuristics*. The first one sequences the jobs by enumerating the sublots and making them as equal as possible. For a given number of sublots, the procedure allocates the sublots as evenly as possible to the machines of stage 1. The second heuristic schedules (with splitting) the jobs in the sequence one at a time by using a smaller MILP model.

Liu (2008) used the same m+1 machine model with the minimization of the *makespan* as the objective for the *single-job* version. He applied a similar approach consisting of two steps, where the sizes of the sublots are *not restricted to be equal or integers*, but they must not be smaller than a *lower bound*, $x_0 \geq 0$.

The total size of the job is U units. The job can be split into sublots. A sublot is treated as an independent entity during the production in the system. Each sublot requires the processing on any of the machines at the first stage and then on the machine of the second stage. The processing times of a sublot at the two stages are proportional to the size of the sublot. The unit processing times on the machines of stage 1 and stage 2 are $p^{(1)}$ and $p^{(2)}$, respectively. Each machine can process at most one sublot at a time. Each sublot can be processed by at most one machine at any time. A setup time is considered for loading the sublot onto the machine. It is independent of the processing sequence and sizes of the sublots. The problem was to determine the number and sizes of the sublots and a schedule for processing them on the machines, such that the makespan is minimized.

The problem was solved efficiently in two steps:

1. The sublot allocation and sequencing were first decided for a fixed number of sublots;
2. The sublot sizes were optimized.

For the problem with a *fixed number of sublots*, Liu (2008) formulated a simple and practical rotation method for allocating and sequencing the sublots on the machines, which was previously used by Zhang, Liu, and Linn (2003). Then it was proved that this method was optimal.

The *rotation method* is as follows:

1. Let the numbers of the sublots be 1, 2,.., *l*;
2. Allocate and sequence these sublots at stage 1 machines in a 'rotation' order, i.e., assign sublot 1 to machine 1, sublot 2 to machine 2, ..., sublot *m* to machine *m*, sublot *m* + 1 to machine 1, and so on;
3. Sequence the sublots on the machine of stage 2 in the order of their numbers.

In this way, the batches of a number of sublots are obtained for every machine. The following features characterize the schedule given by the rotation method:

1. The numbers of sublots processed on the machines of stage 1 are *balanced*, i.e., each machine processes at least $\lfloor l/m \rfloor$ sublots and at most $\lceil l/m \rceil$ sublots;
2. The sublots are processed on the machine of stage 2 in the order of their batch numbers;
3. For the sublots in the same batch, the processing on the machine of stage 2 is in the order of their first-stage machine numbers.

When the sublots are allocated using the rotation method, the only remaining decision is to determine the *optimal sizes of the sublots*. Since all discrete decisions are fixed, this problem can be formulated as an LP model, and the optimal number of sublots can be determined. Then, it only remains to solve a scheduling problem with equivalent sublot sizes. In this paper, optimal and heuristic solution methods for different configurations of the problem were proposed. The worst-case performance of the equal-sublot solution was analyzed. Computational experiments on a wide range of problem settings showed that the approach by Zhang, Liu, and Linn (2003) and Liu (2008) is very close to optimality.

The work by Defersha (2011) was motivated by bridging the gap between the research efforts in flow shop lot streaming and HFS scheduling. A comprehensive MILP model for scheduling a *multi-product f*lexible hybrid flow shop with lot streaming was presented. In this problem, the number of stages is not limited to two. Each stage has a known number of unrelated parallel machines. The batch of each job is to be split into several *unequal consistent* sublots. The sublots are to be processed in the order of the stages. Sublots of certain products may skip some stages. At a given stage, a sublot of a job can be assigned to one of the eligible parallel machines to process that particular job. For each job, there is a *sequence-*

dependent setup time on each eligible machine. This setup may be *anticipatory* or *non-anticipatory* at different stages. Each machine can process at most one sublot at a time. Sublots of different products can be interleaved. The problem was to determine 1) the size of each sublot of each job, 2) the assignment, and 3) the processing sequence of these sublots on each machine at each stage. The objective function was to minimize the *completion time* of the last sublot to be processed in the system. This work was an extension of the research by Ruiz, Sivrikaya Şerifoglu, and Urlings (2008). In that paper, a similar model was considered, but overlapping operations were introduced by means of a negative time lag (see Section 3.4.4). Defersha's model was optimally solved for a small problem size.

A special machine environment was studied in the paper by Lalitha, Mohan, and Pillai (2017). This HFS contains Q jobs, which are in a lot to be processed at N stages. The first N-1 stages contain a single machine, and the last stage contains m machines in parallel. Every job can be divided into a maximum of $N^j_{\max}$ *consistent* sublots. The jobs have to be scheduled *without intermingling* the sublots. There is *no setup time* for the jobs. This scheduling problem involved the determination of the number of sublots, the sublot size, the sublot sequence, and the job sequence with the objective of minimizing the makespan. A MILP model was proposed and solved for *short-size instances* using the LINGO solver. The model gave the optimal makespan with the optimal number of sublots, the sublot sizes, the sublot sequence, and the job sequence. However, the computational price of the solution was very high. Therefore, for large-size instances a *heuristic algorithm* was developed.

This algorithm consisted of two parts:

1. Splitting the lots;
2. Sequencing the jobs.

The lot splitting was based on the average processing times of the jobs at the first (N-1) stages and the cycle time at stage N. The sum of the processing times of all jobs at each stage of the first (N-1) ones were considered for sequencing.

This algorithm reached a near optimal solution for problems within a very small computational time compared to the MILP model. A comprehensible flowchart of the complete algorithm was given.

Cheng, Sarin, and Singh (2016) addressed the two-stage 1+2 HFS, *single-lot*, lot streaming problem (TSHF-LSP) for both the *continuous* and *integer* sublot size versions, which were denoted as TSHF-LSP(C) and TSHF-LSP(I), respectively, in the presence of a sublot-attached *removal time* for each sublot at stage 1 and with the minimization of the makespan

as the objective. An upper bound on the number of sublots, N_{pos} was determined. Then a *TF heuristic algorithm* of complexity $O(N_{pos})$ in conjunction with closed-form expressions for the sublot sizes was developed in two versions to obtain an optimal solution:

- TF-C determines the optimal continuous sublot sizes of the TSHF-LSP(C) problem;
- TF-HI determines the number of sublots and integer sublot sizes for the solution of the TSHF-LSP(I) problem.

As an extension of the previous work, which began before the publication of the paper, Cheng and Sarin (2013) introduced a *multi-objective* 1+2 HFS lot splitting problem with a sublot-attached setup time before the processing of each sublot on the machine at stage 1. The problem was to determine an optimal sequence of the lots, the number of sublots for each lot, the sublot sizes, and an optimal allocation of the sublots to the machines of stage 2 in order to minimize two objectives: the *makespan* and the *sum of the lot completion times*. The cases of both *continuous* and *integer* sublot sizes were considered. An optimal solution for single-lot cases was given. For multiple-lot cases, the TF-C method (M. Cheng, Sarin, and Singh 2016), which was adapted for multiple-lot cases, was used to obtain an optimal number of sublots. Once the number of sublots is determined, the sublot sizes can be obtained. Various LPT- and SPT-based TF-C heuristics for multiple-lot cases were presented to sequencing the sublots.

The problem studied by Nejati et al. (2014) included *multi-job* lot streaming in a *m-stage* HFS with work shift constraint and sequence-dependent setup times. The *work shifts constraint* is as follows. Each job has m operations. Each job i is split into b_i sublots at each shift and all sublots are simultaneously processed. Each sublot of job i is transferred immediately to the next stage after the completion of its processing on a machine. If the completion time of a *sublot* at a specific stage is less than the end time of the work shift, the sublot is sent to the next stage *in the same shift*. If it is greater than the end time of the work shift, it will be completely sent to the *next shift* at the same stage and will be considered as a *WIP job*. If the completion time of the *WIP job* at a stage is less than the end time of the work shift, the WIP job is sent to the next stage *in the same shift*. If it is larger than the end time of the work shift, it will be sent to the *next shift* at the same stage. It is assumed that 1) each job has a *fixed number of sublots*, 2) the sublots are *consistent*, and 3) a setup should be made at the beginning of each shift.

The aim is to minimize the *weighted completion time* of jobs in each shift in order to finish the jobs at earlier shifts without being transferred to the next shift. The proposed model schedules the jobs on the machines, sequences the operations on the allocated machines, as well as finds the size of each sublot in each work shift, the completion times of the works in each shift, and the jobs at each stage as the WIP jobs. A genetic algorithm and a simulated annealing procedure were used to solve this problem.

7.4.3 Job shops and open shops

Job shop lot streaming started in the literature due to Jeong et al. (1997), who used an *iterative batch splitting method* to get an improved schedule for a dynamic *job shop* scheduling problem by splitting the original batch into two smaller batches. A family setup time before the arrival, a job release date, a transportation time, and a due date were considered. The *iterative batch splitting method* consists of two steps:

1. The first step starts by building a conjunctive graph, which represents a solution for a job shop scheduling problem. The graph is used to select the batch to be split and to determine its size;
2. The second step is to improve the schedule for the new job shop problem, where one more job is added due to the split of the batch.

The suggested algorithm splits batches iteratively and scans the improvement of the schedule. It generates schedules, which do not only meet the due date requirements but also reduce the number of batches to an acceptable limit. The batch size was determined by the following four proposed rules: MBS, RBS, FBS and EBS, and selecting the one, which delivered the best solution (see Section 7.1).

Dauzère-Péres and Lasserre (1994) dealt with lot streaming in a job shop environment. An integrated model for *job shop lot sizing and scheduling* was considered in order to determine a feasible plan, which contained at least one feasible schedule. To compute the best possible (feasible) plan, a *modified shifting bottleneck* (MSB) procedure has been presented. It was an adapted variant of the shifting bottleneck procedure for job shop lot streaming, see Dauzère-Péres and Lasserre (1993) and Section 3.3.2 for a description of the method. Starting with an initial plan, the MBS procedure alternates between the two levels in order to find the best feasible plan. The lot sizing calculation with *consistent* sublots for a fixed ordering of the products that share a common resource was first executed and at the second

level, the scheduling decisions were considered for the fixed production plan.

Lot streaming in an *open shop* environment was first studied by Şen and Benli (1998). Some results for scheduling a *single job* in *multi-stage open shops* were presented, considering *single* and *multiple routing* for each sublot. Lot streaming of *multiple jobs* was also discussed for two-machine open shops. Some conditions were stated, under which improvements can be achieved. For such a case, the authors derived the optimal sublot sizes and their sequence.

A *two-machine mixed shop* with *two types of jobs* and the minimization of the makespan as the goal was considered in the paper by Çetinkaya and Duman (2010). An interesting shop model was studied. This model generalized three environments: a flow shop, a job shop, and an open shop. As it is known, the simplest model is the *flow shop*, in which the operations of all jobs are processed in the same order of the machines, and each job has exactly one operation on each machine. A more flexible model is the *job shop*, in which the operations of each job are processed in a predefined but different order. The most general model is the *open shop*, in which the operations of each job may be realized in any order. In the real world problems, planning and control systems need usually to consider a mixture of these shop characteristics. In this work, it was shown that the makespan in a two-machine mixed shop scheduling problem can be improved by using lot streaming if there is an open shop type job j satisfying the condition

$$\max_{j\in J_O}\{a_j + b_j\} > \max\{A_F + A_O, B_F + B_O, \mathrm{CF}^*\}.$$

Meanwhile, the completion time CF^* of the flow shop satisfies the condition

$$\mathrm{CF}^* > \max\{A_F + A_O, B_F + B_O, \max_{j\in J_O}\{a_j + b_j\}\},$$

where

a_j - processing time of job j on M1;
b_j - processing time of job j on M2;
$A_F = \sum_{j\in J_F} a_j$ - sum of the processing times of the flow shop type jobs on M1;
$A_O = \sum_{j\in J_O} a_j$ - sum of the processing times of the open shop type jobs on M1;
$B_F = \sum_{j\in J_F} b_j$ - sum of the processing times of the flow shop type jobs on M2;

$B_O = \sum_{j \in J_O} b_j$ - sum of the processing times of the open shop type jobs on M2;

CF^* - optimal makespan for the flow shop type jobs, which are ordered according to Johnson's rule.

An optimal solution method was developed for shops, which satisfy these conditions. The minimal makespan for the multiple-job scheduling problem in such a two-machine mixed shop is given by:

$$C_{\max} = \max\{A_F + A_O, B_F + B_O, \mathrm{CF}^*\}.$$

In the paper by Defersha and Chen (2012), a complex *multi-job* lot streaming problem for a *flexible job shop environment* was studied considering several pragmatic issues such as: 1) routing flexibility, 2) sequence-dependent setup times, 3) attached/detached nature of setups, 4) machine release dates, 5) lag times, and 6) high-performance parallel computation. The problem studied was classified as flexible job shop scheduling with lot streaming. A MILP model was developed to formalize the problem. This problem is NP-hard and difficult to solve even for small-sized problems using off-the-shelf optimization software. To this end, *an island-model parallel genetic algorithm* was developed. It runs in a high-performance parallel computing environment. This technique reduces the makespan. The authors characterized the results as 'very encouraging'.

A study by Yavuz (2013, 5056–57) introduced the notion of *flexible lot streaming* (FLS) as an integration of the process plans at the shop-floor level into a generalized manufacturing system with lot streaming. In an FLS problem, there exist p process plans to produce a given lot consisting of d identical items. Each process plan i is represented by a linear processing sequence on a set P_i of machines. A part visits the same machine in a given process plan no more than once. For instance, three process plans are defined as P_1 ={M1, M2, M3}, P_2 ={M2, M3, M1}, and P_3 ={M3, M2, M1}, respectively. Each machine $k \in P_i$ has an associated unit processing time $t_{i,k}$ and a setup time $h_{i,k}$, if it exists. A single type of a part, consistent sublots and sublot availability were assumed. Since sublots processed through different process plans can be treated as different jobs, the FLS problem dealt not only with job splitting but also with the sequencing subproblem. For a given number of sublots, the problem consisted of determining simultaneously the sizes of each sublot and sequencing the sublots on each machine with the objective of minimizing the makespan. Two integer programming models were developed and solved for this FLS problem with/without setup times. The computational results showed that

the consideration of FLS may yield a substantial improvement in the makespan.

Bożek and Werner (2018) addressed the sublot size optimization in *flexible job shop* scheduling with lot streaming, where the sublot size is *variable*. A *two-stage optimization procedure* was proposed. First, the makespan value was minimized with the smallest sublots defined for the problem instance. In the second stage, the sizes of the sublots were maximized without increasing the obtained makespan value. In this way, the number of sublots and transport activities were limited together with the related manufacturing cost. Two objectives were defined for the second stage: 1) maximization of the *sum of the sublot sizes* of all operations, and 2) maximization of the *number of operations*, which do not need to be split at all. The problem was solved by MILP, constraint programming and graph-based models as well as two optimization approaches: the first one was based on a third-party solver and the second one on a tabu search and a greedy constructive heuristic.

Yousefi Yegane, Kamalabadi, and Khanlarzade (2017) studied a lot streaming problem in a *flexible job shop* environment with *multiple jobs* and the objective of minimizing the maximum *completion time*. In this study, first, the considered problems were solved both for permitted and not permitted lot streaming by means of a *memetic algorithm*. Then the obtained solutions were optimized by adapting a *critical path-based heuristic technique*.

7.5 Conclusions

Lot streaming is one of the current trends in deterministic scheduling due to its practical utility. It is a key element in synchronous manufacturing, JIT and OPT philosophies, which address each item in the lot as a single unit or a job searching for a minimum makespan. Overlapping operations are sometimes enforced by a lack of floor space.

Jeong et al. (1997, 781) highlighted the importance of splitting jobs in order to make the scheduling decisions less sensible to unexpected events:

> 'Most past solutions to scheduling problems assume that a manufacturing shop is operating steady and peacefully. But a real factory is full of unexpectedness and dynamics. The causes of such dynamism are very diverse, e.g. human errors, rush orders, requests from customers to shorten the delivery dates, and hardware related events - tool breakage, machine failure, etc. These events cannot be expected beforehand, and therefore, it is very hard to strictly observe the original schedule.'

He concluded:

> 'we can get much better solution considering alternative schedules which allow the splitting of current batches in smaller batches'.

Lot streaming combines lot sizing and scheduling decisions that were traditionally made separately. The application of the lot streaming idea has received considerable attention in the scheduling literature.

It is a relatively new approach and therefore, the theory is given directly by the published papers. The state-of-the-art shows that there is still a lack of studies in this area, and the theory is not sufficiently strong. After reading the related papers, one can note that the problem statements are frequently vague, the authors sometimes do not give a clear identification of the problem and do not use any classification system to denote the structure of the problem. The complexity of lot streaming problems is very sensible to the assumptions declared, changing drastically from an NP-hard problem to one, which can be solved by a simple algorithm. Nevertheless, many authors refer only to the main scheduling problem without lot streaming consideration to confirm the NP-hardness of the studied problem. Therefore, a strict classification system and the corresponding complexity analysis are urgently required.

This chapter shows an evolution of the lot streaming theory from the first definitions and simplest models with two-machine flow shop environments and a single-job (Graves and Kostreva 1986; Potts and Baker 1989; Glass, Gupta, and Potts 1994; Çetinkaya 1994), and minimal configurations of the HFS, such as 1+m and m+1 (S. Li 1997; W. Zhang, Liu, and Linn 2003; W. Zhang et al. 2005; J. Liu 2008), to multi-job and multi-machine general schemes (Yoon and Ventura 2002; Feldmann and Biskup 2008; Defersha 2011; Defersha and Chen 2012; Mortezaei and Zulkifli 2013; Yavuz 2013; Nejati et al. 2014; Lalitha, Mohan, and Pillai 2017), and multi-objectives goals (M. Cheng and Sarin 2013; Bożek and Werner 2018). In recent years, there appeared papers, which studied a learning effect in the lot streaming context (Mukherjee and Sarin 2014; Mukherjee, Sarin, and Singh 2016; Fattahi, Azizi, and Jabbari 2015). It is logical to wait for publications, which connect the no-wait lot streaming assumption with the deteriorating job restriction.

One also can note a progress in the solution algorithms used, from simple rules (MBS, et al.; FH; BH; adapted SPT and LPT; AVG) and well-known heuristics (a critical patch algorithm; a modified Johnson algorithm; an iterative batch splitting method; a modified shifting bottleneck procedure; a critical path-based heuristic technique), to hybrid algorithms

looking for more exact solutions (VNS, SA, TS). Following this trend, we can wait for a big interest of the researchers in this subject.

Reviews, which were dedicated to lot streaming problems, have been given by Potts and Baker (1989); Potts and Van Wassenhove (1992); Trietsch and Baker (1993); Chang and Chiu (2005); Cheng, Mukherjee, and Sarin (2013); Gómez-Gasquet, Segura-Andrés, and Andrés-Romano (2013).

CHAPTER EIGHT

LOT SIZING

Essentially, the problem of short-term production planning turns out to be a lot sizing and scheduling problem, then. If we ask about how to solve this production planning problem, we first need a deeper understanding of its basic attributes. (*Drexl and Kimms 1997, 222*)

The complexity of planning problems arising in big plants with lot processing, such as semiconductor manufacturing plants, impedes an efficient solution of many problems due to various reasons, such as high dimensionality and production costs. The lot sizing optimization problem is one of these problems.

In discrete manufacturing environments, the term *lot sizing* defines the process of determining the production quantity of each product over a finite multi-period planning horizon. In the planning and scheduling literature, a lot sizing problem refers to the decision on when and how to split a production job consisting of a number of identical items into lots.

Lot sizing is a fundamental problem in different production environments. It is one of the key problems for any production planning system in big production plants. A typical variant of such a problem can be described as follows.

Several *items* must be produced in order to meet a number of *demands*, which can be known beforehand or estimated. *Backlogging* and *stocking* are not desirable. The finite *planning horizon*, which is usually less than six months, is subdivided into a number of discrete time *periods*, such as hours, days, or weeks. During the production, the items share a common machine. The *capacity* of that machine is limited and may vary over time. Producing one item requires an item-specific amount of the available capacity of that machine. The items, which are produced in the actual period to meet a demand in the future, must be stored in the *inventory* and therefore, they cause item-specific holding costs. The production of an item can only be realized if a corresponding *setup* is performed. The machine setup for manufacturing of a particular item incurs item-specific setup costs. The goal is to determine the optimal number of the production items by balancing the trade-off between the production, inventory, and setup costs.

Production planning addresses the acquisition, utilization, and allocation of the production resources, which are required to transform the raw materials into finished goods in the most efficient and economical way. An inappropriate lot sizing causes an instable material flow in the shop, a high setup frequency or WIP level. It influences negatively the manufacturing performance metrics. Planning the sizes of the lots for the production processes with a dynamic demand and under a tight capacity restriction is an important and difficult subject, which is not really a new one. Nevertheless, recently it reached an extensive research interest.

Lot sizing is sometimes called batching, because both are used to induce a time-phased production and make it synchronized with the actual consumption or the demand pattern. In some sense, it is opposite to batching, which is a decision on whether or not to schedule similar jobs contiguously. Lot sizing can also be seen as a job splitting problem, because it supposes the division of an entire job into sublots. Lot sizing is also strictly connected with decisions taken by lot streaming problems in the scheduling theory.

In this chapter, the history of the problem, some principal concepts, and basic models are described, complemented by a broad state-of-the-art survey for the readers, which like to study the subject more in depth.

8.1 Background

The origin of the lot sizing problem takes us to the paper by Ford Whitman Harris, entitled 'How Many Parts to Make at Once', which was published in Factory, The Magazine of Management (Harris 1913). This publication was dedicated to determine economic lot sizes for production or order, by balancing inventory and setup or order costs. In this paper, Harris presented a formula for the order quantity calculation called the *economic order quantity* (EOQ), which is known now as the famous *square-root formula* derived by Harris (Wagner and Whitin 1958, 1773):

$$Q = \sqrt{2Ds_c/i}, \tag{8-1}$$

where

Q - order quantity;
D - demand quantity;
s_c - ordering setup cost;
i - interest charge ($i = 1$ if no other considerations).

Today, the EOQ model is the classical one and so well known in the area that this basic structure is accepted as obvious. Nevertheless, in 1913, it was a modeling achievement of high elegance. After that, Harris' original paper was lost sight of many years. His name was almost forgotten, while the square-root formula was used referring to other authors until his rediscovery in 1988. An interesting historical sketch, which was dedicated to Eng. F.W. Harris as the originator of the EOQ model, also describing his personality and contributions, was written by Prof. Donald Erlenkotter (1990), who made this investigation.

The EOQ formula proposed by Harris is a simple and efficient tool to avoid an excessive inventory, but it has several restrictions. The assumptions for the EOQ model are a single-level production process with no capacity constraints, which turns the formulation into a single-item problem. The demand for that item is assumed stationary, i.e., the demand occurs continuously with a constant rate. An optimal solution is easy to derive.

A few years later, on the base of the EOQ formula, Taft (1918) developed the *economic production lot* (EPL) formula by replacing the replenishment with a continuous rate production-function (Jodlbauer 2006). Both formulas represent *continuous time models* with an *infinite planning horizon.*

Since Harris' publication, the subject has received attention both in the research literature and in practice. In a modern interpretation, the *lot sizing problem* became an active research area since the seminal work by Wagner and Whitin. This formula was one of the first models for a *dynamic demand in discrete time* over a *finite planning horizon*. It was based on the *Wagner–Whitin fundamental property*: under an optimal lot sizing policy, either the *inventory* carried to period t+1 from a previous period will be zero, or the *production quantity* in period t+1 will be zero (Wagner and Whitin 1958, 1771). Almost at the same time, *discrete lot sizing models* were developed for more complex manufacturing situations by Manne (1958), Zangwill (1966) and later by Lasdon and Terjung (1979). These models considered already multiple products, capacity restrictions, setups, and multi-level production structures.

Typically, the objective of lot sizing is to minimize the sum of the setup and holding costs. Other costs might also be considered. Boucher (1984) noted a traceable relationship between the lot size and the WIP inventory in GT. Moreover, the WIP inventory level was found as a function of the lot size.

Rogers (1958) was the first researcher who noted an intrinsic connection between lot sizing and scheduling. Since the works by Potts and Van Wassenhove (1992); Haase (1996); Tsubone, Ohba, and Uetake (1996),

many authors highlighted this connection. Indeed, when the sequence-dependent setup costs must be accurately computed, the integration of sequencing into lot sizing is inevitable.

Potts and Van Wassenhove (1992) were the researchers who first integrated explicitly scheduling with batching and lot sizing approaches into the production management. However, the formulation of the lot sizing models and the framework used are different from the above-mentioned subjects due to its planning background. Nevertheless, in advanced manufacturing processes, batching, lot sizing, and scheduling decisions, which minimize the schedule, must be taken concurrently, i.e., they should be integrated and computer-controlled.

8.2 Parameters and assumptions

Before modeling, a problem must be strictly formulated. There are various specific system parameters and assumptions, which are involved in a lot sizing problem statement and have a direct influence on the modeling procedure. Some basic ones are discussed below (Kuik, Salomon, and Van Wassenhove 1994).

8.2.1 Parameters

The main system parameters are as follows.

Planning horizon and time scale. First of all, a certain period, denoted as the planning horizon, must be defined. The planning horizon is the time interval, in which an MPS extends into the future. The planning horizon may be finite or infinite. A *finite* planning horizon is usually complemented by a dynamic demand, while an *infinite* planning horizon is complemented by a stationary demand. Then, the production system may be represented continuously or at discrete time points, which correspond to a system of *continuous* or *discrete time scale*.

A discrete scale represents a partitioning of the time into a number of *time buckets* or *planning periods*. In this case, the real-world events, which happen in a continuous time, have to be transformed into the events and decisions, which occur according to a discrete time scale. To make this translation reasonable, valid, and accurate, one has to select carefully the size of the time bucket.

With respect to the time periods, lot sizing problems are classified into big bucket or small bucket problems, as it is defined by Belvaux and Wolse (2000). A *big bucket* problem is one, where the time period is long enough

to produce multiple items, what corresponds usually to the *multi-item* problem case. For a *small bucket* problem, the time period is so short that only a *single item* can be produced. Also, a *rolling planning horizon* can be considered, see Kimms (1998); Clark and Clark (2000); Mohammadi et al. (2010). It usually appears when there is an uncertainty in the data.

Levels. Single-level and multi-level production systems are distinguished. In a *single-level* system, the final product is usually simple to manufacture. The raw materials, after being processed by a single operation, are changed to a final product. A final good item is directly produced from the raw or purchased materials with no-intermediate operations. The product demands are taken directly from the customer orders or the market forecasts. This kind of demand is referred to as an *independent demand*. In a *multi-level* system, there is a precedence relationship between the item components. Several operations are required to change the raw materials into an end product. The output of an operation/level is the input for another operation/level. Therefore, the demand at one level depends on the demand for its parent's level. This kind of a demand is denoted as *dependent demand*. Evidently, multi-level problems are more difficult to solve than single-level problems.

Multi-level systems are further distinguished by the production system structure, such as *serial*, *assembly*, *arborescent*, and *general* structures, which can be represented by a *Gozinto Graph*[11] (see Section 8.3.15).

Products. Two principal types of production systems are used in terms of the number of products, namely, single-item and multi-item production planning. In a *single-item* production planning system, there is only one final product (end item), while in a *multi-item* production planning system, there are several end items. The complexity of multi-item planning problems is much higher than the complexity of single-item problems.

Capacity or resource constraints. *The capacities* in a production system, also referred to as *resources*, include manpower, equipment, transport, robots, machines, the budget, etc. When there is no restriction on the resources, the problem is defined as *uncapacitated*. When the capacity constraints are explicitly stated, the problem is called *capacitated*. The presence of capacity constraints makes a problem more difficult to solve.

[11] A *Gozinto Graph* is a tree-like graphical representation of raw materials, parts, intermediates, and subassemblies, in which a particular production process transforms raw material into an end product through a sequence of (production) operations.

Demand rate. The nature of a demand is an input parameter of a lot sizing problem. A *static* demand, also called a *stationary* demand, assumes a constant demand pattern, while a *dynamic* demand permits a demand to vary over time. If all relevant demand data are available before the planning, such a demand is assumed deterministic. In the opposite case, in which the demand values are based on expected probabilities, the demand is considered stochastic.

A demand can be either independent or dependent. When the demands are *independent*, the imposed requirements on one item do not depend on the decisions regarding the lot size of another item. This kind of demand is typical in *single-level* production systems.

In *multi-level* lot sizing with a parent-component relationship among the items, the demands are *dependent*, because the demand requirements at one level depend on the demand for its previous level, and the production at one level leads to a demand for the components at a lower level.

Obviously, problems with dynamic and dependent demands are much more complex than problems with static and/or independent demands. In addition, problems with probabilistic demand are more complex than problems with deterministic demand.

Setup structure. Setup costs and times are usually modelled by introducing binary zero–one variables in the mathematical model of the problem and make the solution of the problem more difficult. Usually, a production changeover between different products incurs a setup time and a setup cost. There are different types of setups, see Chapter 4 for more details.

The setups can first be considered according to the *conservation of the setup state*. This means that the setup state may either be conserved (*continuous setup*) or get *lost* after idle periods.

The setups may also be grouped globally into two big categories:

1. *Simple* structure of a setup, if the setup time and cost in a period are independent of the item sequence and the decisions made in the previous periods;
2. *Complex* structure of a setup, if it depends on the item sequence or the previous periods.

In the context of the lot sizing problematic, complex setups can be further subdivided into the following types:

1. *Carry-over setup*, if it is possible to continue the production run from the previous period into the current period without the need for an additional setup, thus reducing the setup cost and time;
2. *Family setup* or *major setup*, which is caused by similarities in the manufacturing process and the design of a group of items;
3. *Item setup* or *minor setup*, which occurs when changing the production among the items within the same family;
4. *Sequence-dependent setup*, when the item setup cost and time depend on the production sequence.

It is obvious that complex setup structures have a harder difficulty in both modeling and solving lot sizing problems.

Inventory shortage. If a shortage is allowed, this means that one of the following two variants is possible:

- *Backlogging* - a demand of the current period is satisfied in a future period;
- *Lost sales* - a demand will not be satisfied at all.

The combination of backlogging and lost sales is also possible. In such a case, a shortage cost is introduced into the objective function. If a deterioration effect occurs in the problem, it must be included in the inventory holding time restrictions. This feature increases the problem complexity.

Problems with a shortage are more difficult to solve than those without a shortage.

Costs. There are two types of costs, which are taken into account:

- Setup costs;
- Inventory holding costs.

Setup costs are the costs incurred when changing the resource configuration from processing one type of products to another one. They account for the loss of the potential production and the resources, which were consumed during the setup, such as additional workforce and additional raw material. *Inventory holding costs* account for opportunity costs of capital as well as for direct costs of storing goods, such as warehousing, handling, transporting, etc.

To minimize the setups costs, a production is run with large batches but at the expense of high inventory costs. On the contrary, the inventory levels

can be kept low if the manufacturing of a product is run in frequent and small batches, but at the expense of high setup costs. Therefore, a lot sizing model aims at finding an optimal compromise between the setup and the inventory holding costs, while complying with the given capacity constraints and insuring that the demand for all products is satisfied without backlogging.

8.2.2 General Assumptions

A planning activity is first motivated by identifying a subject that is worth to be investigated in an economical sense. It is frequently caused by large inventories. Direct costs of storing goods and holding items in the inventory, which produce *holding costs*, should be avoided. On the other hand, if different parts use common resources, say the machines, and the setup activities take place to prepare a corresponding operation, then *setup costs* are incurred when the production is delayed. Another aspect of sharing recourses is that different parts need different setups. Hence, the production orders must be sequenced. As a result, the goal is a trade-off between low setup costs favoring large production lots and low holding costs favoring a lot-for-lot-like production, in which the sequencing decisions have to be made due to sharing the common resources. Therefore, the problem of short-term production planning is mainly a lot sizing and scheduling problem.

Any production plan must respect the *precedence relationships* between the operations, if they exist. Hence, *multi-level structures* and *multi-item* production must be respected. The production planning must take into account a *scarce capacity*. In the opposite case, the optimization decisions have no sense.

Holding costs are incurred for the items in the inventory. The demand for the items has to be satisfied without a delay, i.e., shortages are not allowed. The external demands, either known or estimated ones, are given at the end of each period. Backlogging and shortages must be included into the model, if they are considered.

8.3 Main problems

A brief description of the main problems for lot sizing is given below, including the principal features and mathematical formulations. For additional information, the following papers can be consulted (Kuik, Salomon, and van Wassenhove 1994; Haase 1996; Drexl and Kimms 1997; Fleischmann and Meyr 1997; Staggemeier and Clark 2001; Brahimi et al. 2006; Gicquel, Minoux, and Dallery 2008; Ramezanian, Saidi-Mehrabad, and Fattahi 2013).

All lot sizing problems are seeking for an optimal timing and level of the production. Nevertheless, these problems have several variations in the mathematical formulation. The selection of a convenient variant of the problem is an important methodological point in the problem solution.

8.3.1 Notations

The notations used are as follow:

Indexes:
j - item, $j = 1,\ldots, J$;
t - time period, $t = 1,\ldots, T$;
n - position, $n = 1,\ldots, N_T$.

Decision variables:
I_{jt} - inventory for item j at the end of period t;
q_{jt} - production quantity for item j in period t;
q_{jn} - production quantity for item j at position n;
x_{jt} - binary variable, $x_{jt} = 1$ if a setup for item j occurs in period t, and $x_{jt} = 0$ in the opposite case;
x_{jn} - binary variable, $x_{jn} = 1$ if a setup for item j occurs in position n, and $x_{jn} = 0$ in the opposite case;
y_{jt} - binary variable, $y_{jt} = 1$ if a setup for item j occurs in period t, and $y_{jt} = 0$ in the opposite case;
y_{jn} - binary variable, $y_{jn} = 1$ if the machine is ready to produce item j at position n, and $y_{jn} = 0$ in the opposite case.

Parameters:
c_t - available capacity of the machine in period t;
c_j^1 - capacity need for producing one unit of item j;
d_{jt} - external demand for item j in period t;
h_j - non-negative holding costs for item j;

I_{0t} - initial inventory for item j;
N_t - maximum number of lots in period t;
s_j - non-negative setup costs for item j;
y_{0t} - binary value, $y_{0t} = 1$ if the machine setup for item j occurs at the beginning of period 1, $y_{j0} = 0$, $\sum_{j=1}^{J} y_{j0} \leq 1$.

8.3.2 Basic models

Several models have been proposed for the mathematical formulation of different lot sizing problems. The following variants are the principal ones:

- Economic order quantity model;
- Deterministic dynamic lot sizing Wagner-Whitin model;
- Economic lot scheduling model;
- Uncapacitated lot sizing model;
- Capacitated lot sizing model;
- Discrete lot sizing and scheduling model;
- Continuous setup lot sizing model;
- Proportional lot sizing and scheduling model;
- General lot sizing and scheduling model;
- Multi-level lot sizing and scheduling model.

These models and some important extensions are described below. An adapted interpretation of the mathematical representation, which was given by Drexl and Kimms (1997), was used. The reader can also consult the representations of these models given by Jans and Degraeve (2008).

8.3.3 Economic order quantity (EOQ) model

The EOQ model[12] is characterized by a *single-item* and *single-level* production process without capacity constraints. It is a famous model, which assumes a *continuous* time scale, a constant demand rate, and an infinite planning horizon. It is easy to solve by Harris square-root formula (see

[12] The EOQ model is also referred to by some early authors as the *Economic Lot Size* (ELS) model, see Wagner and Whitin (1958, 1770). This abbreviation introduces a confusion with *the economic lot scheduling* (ELS) model. The readers can meet another confusion between the *capacitated lot sizing* and the *continuous setup lot sizing* models, which are abbreviated in the literature in a very similar manner, CLS and CSL, respectively. To avoid this confusion, the abbreviation CSLS is used for the *continuous setup lot sizing* in this book.

Formula 8-1) (Harris 1913; Wagner and Whitin 1958, 1773; Brahimi et al. 2006).

Rogers (1958, 265) noted that this formula solves only a very specific case of lot sizing. When additional variables (for example, the price as a function of the lot size, the setup cost as a function of the lot size, or the production cost as a function of the production rate) and conditions on the variables must be considered, particularly when some of the variables are discontinuous, the cost-minimizing expressions may become so complex that solutions are hard to reach. Additional difficulties appear when the number of items must also be considered. Another difficulty, highlighted by Rogers (1958, 266), is that the usual ELQ approach makes no explicit provision for scheduling, proceeding as if each item, for which the ELQ calculations are made, can be considered independently.

8.3.4 Deterministic dynamic lot sizing Wagner-Whitin (WW) model

It is a dynamic version of the EOQ model, which is characterized as a *single-level*, *single-item* model. A *finite* planning horizon is subdivided into several discrete periods. A *dynamic* demand is given per period. It may vary over time. It is assumed that each production system incurs a *fixed setup cost* in every period, in which it produces one (1) item. However, capacity limits are not considered. This means that the single level WW problem is a single-item problem.

The deterministic dynamic lot sizing problem is one of the most famous discrete decision problems of production and inventory planning. It has been first introduced and solved by Wagner and Whitin in 1958. The way, in which this problem is modeled, differs from the classical EOQ model, where the demand is neither stationary nor continuous. Instead, both the inventory replenishments and the demand are instantaneous and discretized over a finite planning horizon of several periods. The demand may change from one period to another one, what means that it is *time-varying*. The WW model can be viewed as a shortest path problem. Wagner and Whitin (1958) developed a well-known exact algorithm based on dynamic programming. Another known method solving the dynamic lot sizing model was proposed by Edward A. Silver (1973). This approximation algorithm is known as the *Silver–Meal heuristic*. It is essentially more complex than the WW algorithm.

This model was also studied by Karmarkar, Kekre, and Kekre (1987); Federgruen and Tzur (1991); Wagelmans, van Hoesel, and Kolen (1992);

Aggarwal and Park (1993); Kuik, Salomon, and Van Wassenhove (1994); Aksen (2007). Some capacitated versions were also formulated.

8.3.5 Economic lot scheduling (ELS) model

It is a *single-level*, *multi-item* problem with capacity constraints. This model generalizes the EOQ considering multiple items that share a constrained resource. While the EOQ model is simple to solve, the ELS problem is NP-hard (Hsu 1983; Gallego and Shaw 1997). It is a *continuous time* model with infinite planning horizon as well. This problem has various extensions, see the papers by Elmaghraby (1978); Moon, Giri, and Choi (2002); Raza and Akgunduz (2008), for reviews. Rogers (1958) was the first researcher who noted an intrinsic connection between lot sizing and scheduling. Rogers also incorporated sequencing techniques into the economic lot sizing model, while Maxwell (1964) proposed a *time-varying lot sizes* approach, which is one of the most used approaches up-to-date.

The basic ELS problem, as it was described by Maxwell, has the following features:

1. Only one product can be produced at a time on the machine;
2. Each product has a deterministic and constant demand and deterministic and constant production rates;
3. The setup cost and the setup times are independent of the production sequence;
4. The production facility is assumed capable of satisfying the demand predicted for the planning horizon;
5. The inventory holding cost is directly proportional to the amount of the inventory.

One of the frequently used methods to solve a CLS problem is a dynamic programming-based *Common Cycle* (CC) approach, which restricts all product cycle times to equal length. It is the simplest one to implement, however, in some cases, the solution when compared to a lower bound is of poor quality. The main advantage of this approach is that it always provides a feasible schedule (Bomberger 1966; Jones and Inman 1989; Gallego 1990).

8.3.6 Uncapacitated lot sizing (ULS) model

It is a generalized variant of the WW single-item, single-level problem, which can include setups. That is, it can be extended to a multi-item

problem, but *capacity restrictions are not included*. The production cost of a lot is decomposed into a fixed cost, which is independent of the lot size, and a unit cost for each unit, which is produced in the lot. The inventory costs are modelled by charging an inventory cost per unit held in the inventory at the end of each period. Any demand in a period is satisfied by the production or the inventory. Backlogging is not allowed. The production capacity in each period is not considered in the model. Therefore, it is assumed infinite. The ULS model can be solved by the WW dynamic algorithm (Wagner and Whitin 1958).

ULS models are mainly used in the practice since the implementation of the capacitated approaches requires big efforts for collecting, maintaining and processing the manufacturing data. They are also employed as a basic stage in modeling and solving complex multi-item and multi-level problems (Fleischmann 1990, 340; Pochet 2001; Aksen 2007; Jans and Degraeve 2008; Gicquel, Minoux, and Dallery 2008). The ULS model is the core subproblem in production planning, because it is solved repeatedly for each item in the material requirements (MRP-I) of a sequential planning system. All capacitated models have uncapacitated extensions.

It is useful to employ a network presentation for the resolution of a lot sizing problem. A simple example for an uncapacitated multi-level lot sizing problem is given in Fig. 8-1.

ULS models were reviewed in the papers by Brahimi et al. (2006), Robinson, Narayanan, and Sahin (2009).

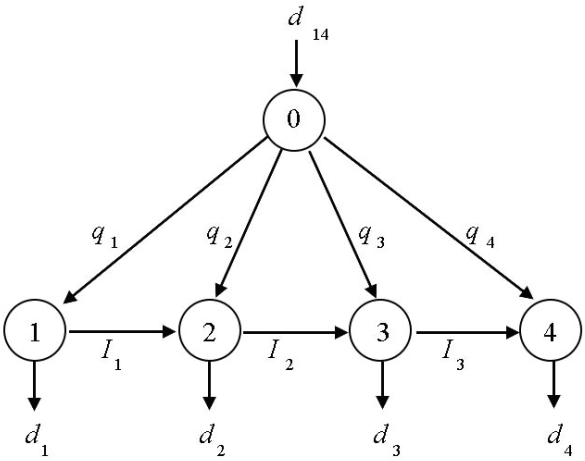


Fig. 8-1. An uncapacitated lot sizing problem with 4 demands for 4 periods and holding inventory. Adapted from Jans and Degraeve (2008, 1622).

8.3.7 Capacitated lot sizing (CLS) model

This lot sizing variant is also called *big-bucket model* or *large time window* of a *multi-item single-stage system*. It is a typical example of a big-bucket problem, where many different items are produced on the *same resource* in one time period. The production resources can manufacture only one type of product at a time and therefore, they have a limited capacity. This means that a significant setup work is required for the machine adjustments from the production of one type of items to another one. The general assumption is that there is exactly one setup for each item, which is produced in the period. Moreover, the CLS model in general does not address scheduling. The planning interval is subdivided into many short periods, shifts or days, and then the jobs are scheduled in each period separately. A lot consists of a quantity of items of the same product, which is processed in one or several consecutive periods. A setup occurs at the beginning of the first period of every lot. The main difference with an *uncapacitated* model is the addition of capacity constraints. This model is useful for developing short-term production schedules for single facilities. The planning horizon is usually less than six months.

Detailed scheduling decisions are not integrated into the CLS model. The usual approach consists in solving the CLS problem first and then the corresponding scheduling problem for each period separately afterwards.

The CLS model is as follows:

$$\sum_{j=1}^{J}\sum_{t=1}^{T}\left(s_j x_{jt} + h_j I_{jt}\right) \rightarrow \min \tag{8-2}$$

s.t.

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt},\ j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-3}$$

$$c_j^1 q_{jt} \leq c_t x_{jt}, j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-4}$$

$$\sum_{j=1}^{J} c_j^1 q_{jt} \leq c_t,\ t = 1,\ldots, T; \tag{8-5}$$

$$x_{jt} \in \{0,1\}, j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-6}$$

$$I_{jt}, q_{jt} \geq 0, j = 1, \ldots, J,\ t = 1,\ldots, T. \tag{8-7}$$

The objective (8-2) is to minimize the sum of the setup and holding costs for all items and all periods. The restrictions (8-3) ensure the inventory balances. The constraints (8-4) indicate that the production of an item can only be realized, if the machine is setup for that item. The restrictions (8-5) ensure the capacity constraints. The restrictions (8-6) define binary setup variables, and the inequalities (8-7) are the non-negativity conditions.

A detailed description of the model can be found in the papers by Fleischmann (1990) and Armentano, França, and de Toledo (1999).

The CLS problem in known to be NP-hard (Florian, Lenstra, and Rinnooy Kan 1980). Armentano, França, and de Toledo (1999) proposed a mathematical model, which represented the CLS as a minimum cost network flow problem. A setup time and a setup cost were considered. A branch-and-bound method was employed for solving the model. Heuvel and Wagelmans (2006) proposed an $O(T^2)$ dynamic programming algorithm for a special case of the CLS problem with non-increasing setup costs, general holding costs, non-increasing production costs, and non-decreasing capacities over time, where T is the length of the model horizon.

A usual approach to solve a CLS model is the application of Lagrangian relaxation to induce a problem decomposition into a set of ULS subproblems that are then solved by the WW algorithm or its variants, see Toledo and Armentano (2006); Caserta and Quiñonez Rico (2009). Karimi, Fatemi Ghomi, and Wilson (2003) gave a detailed analysis of solution methods for the CLS problem.

There are several surveys and reviews of extensions, as well as solution algorithms for the CLS problem (Drexl and Kimms 1997; Karimi, Fatemi Ghomi, and Wilson 2003; Brahimi et al. 2006; Gicquel, Minoux, and Dallery 2008).

8.3.8 Capacitated lot sizing model with sequence-dependent setup costs (CLSD)

In this model, more than one setup is allowed per period, and the setup state can be preserved over an idle time. The CLSD problem describes the case when one or more machines (or production facilities) are used to meet a forecasted demand for multiple products over multiple periods. It must be decided, which products must be made in which periods to meet the exact production sequence and the production quantities in order to minimize the sum of the setup and inventory holding costs. The difficulty of this problem consists in the fact that the capacity is tight, the setup costs are large and sequence-dependent, and the setup times are non-zero. This model was introduced by Haase (1996).

Trigeiro, Thoam, and McClain (1989) observed that the *bin packing problem* (Garey and Johnson 1979, 226) is a special case of the CLS problem with setup times. With this reason, the CLSD problem is NP-hard. Therefore, heuristic methods are used. Haase (1996) developed a *backward-oriented heuristic* for solving the CLSD problem and some extensions. Gupta and Magnusson (2005) tested the accuracy of several different lower bounding linear relaxations with a heuristic and obtained an approximate solution with an average deviation from the corresponding exact solution in a range of 10-16%. The authors also presented an extensive state-of-the-art survey for the CLSD problem.

Trigeiro (1989) developed a simple and efficient heuristic algorithm called *simple heuristic* (SH) for CLSD. It consists of several passes through the production schedule for a problem, starting with a lot-by-lot solution and provides reasonably good solutions. Nevertheless, a modified *Silver-Meal heuristic* is initially applied to each item (Silver 1973).

8.3.9 Discrete lot sizing and scheduling (DLS) model

This is a *multi-item small-bucket model* or a *small-time* window of a *multi-item single-stage system*. In this problem, the macro-periods of the CLS are subdivided into micro-periods, in which only one type of parts may be processed at the full capacity. The fundamental assumption of the DLS model is that the production process always runs over full periods without

changeover. Such an "*all-or-nothing*" assumption implies that at most one item is produced per period. A discrete production *policy* corresponds to such a lot sizing problem: if there is any production in a period, it must be at the full capacity, meaning that the production process always runs full periods without a changeover and that the setup state is not preserved over the idle time.

The major advantage of small-time-bucket problems in DLS models against the CLS is the exact control of the lot sequence and hence, the possibility to include sequence-dependent setup costs.

The DLS problem was introduced by Fleischmann (1990) and later studied by Brüggemann and Jahnke (2000), and by Gicquel, Minoux, and Dallery (2008).

The DLS model is as follows:

$$\sum_{j=1}^{J}\sum_{t=1}^{T}\left(s_j x_{jt} + h_j I_{jt}\right) \rightarrow \min \tag{8-8}$$

s.t.

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt},\ j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-9}$$

$$c_j^1 q_{jt} = c_t y_{jt}, j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-10}$$

$$\sum_{j=1}^{J} y_{jt} \leq 1,\ t = 1,\ldots, T; \tag{8-11}$$

$$x_{jt} \geq y_{jt} - y_{j(t-1)}, j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-12}$$

$$y_{jt} \in \{0,1\}, j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-13}$$

$$I_{jt}, q_{jt}, x_{jt} \geq 0, j = 1, \ldots, J,\ t = 1,\ldots, T. \tag{8-14}$$

The objective (8-8) is to minimize the sum of the setup and holding costs for all items and all periods. The restrictions (8-9) assure the inventory balances. In contrast to the restrictions (8-4) in the CLS model, here in the restrictions (8-10), the left and right sides are equivalent. This corresponds to the assumption 'all-or-nothing'. The restrictions (8-11) ensure that at most one item is produced per period, in accordance with the capacity limit constraints (8-10). Usually, it is assumed that the capacity remains constant over the time, i.e., $c_1 = c_2 = \ldots = c_T$. The inequalities (8-12) indicate the beginning of a new lot. The restrictions (8-13) define binary setup variables, and the inequalities (8-14) are the non-negativity conditions.

Finding an optimal solution of the DLS problem was discussed by Salomon et al. (1991, 805–10). A feasible solution can be obtained in polynomial time. If setup times or parallel machines are considered, even finding a feasible solution of the DLS problem is NP-hard.

All solution methods presented so far for the DLS problem are restricted to *sequence-independent setup costs*, because a preferred procedure consists in decomposing the problem into single-product problems. Fleischmann (1990, 330) described a procedure for solving the DLS problem with capacity constraints, which is based on branch-and-bound and Lagrangian relaxation algorithms. A dynamic programming based Lagrangian relaxation leads to single-product problems, but it is distinct from the WW type. The branching strategy consists in the application of the Last-In, First-Out (LIFO) rule for the selection of the product to be fixed in the current period. Fleischmann compared the performance of his approach for both CLS and DLS models. He showed that the DLS problem can be used for modeling capacitated multi-product dynamic lot sizing problems. This method is also very suitable for determining both feasible solutions and relatively sharp lower bounds.

Some reviews of DLS model extensions and solution methods can be found in the papers by Drexl and Kimms (1997); Salomon et al. (1991); Staggemeier and Clark (2001); Brahimi et al.(2006); Copil et al.(2017).

8.3.10 Discrete lot sizing with sequence-dependent setup costs (DLSDSD) model

This model was introduced by Fleischmann (1994) and studied by Gicquel, Minoux, and Dallery (2008). It is similar to the CLSD problem. The main difference between the CLSD and DLSDSD models is that in the first one, continuous lot sizes are allowed and the setup state can be preserved over an idle time.

8.3.11 Continuous setup lot sizing (CSLS) model

This is a *small bucket* model. In this kind of problems, a machine can only process *one product type* in one period.

The CSLS model is an extension of the CLS model, obtained by relaxing the assumption that the available capacity is fully utilized in the period, where a product is manufactured. It is very similar to the DLS model, and only one item may be produced per period. The decision variables are basically the same. The difference is that the '*all-or-nothing*' assumption is not applied. It allows at most one part type in each period by using less than

the full capacity. Nevertheless, the difference between these models is sensible. In the DLSP problem, the setup costs are incurred whenever a new lot begins. In the CSLS problem, however, setup costs can occur only once. The CSLS model is based on the assumption that at most one item can be produced per period.

The CSLS model is considered to be less realistic than the DLS model and less studied in the literature, see Karmarkar and Schrag (1985); Drexl and Kimms (1997) for details.

The CSLS model is as follows:

$$\sum_{j=1}^{J}\sum_{t=1}^{T}\left(s_j x_{jt} + h_j I_{jt}\right) \rightarrow \min \tag{8-15}$$

s.t.

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt},\ j = 1, \ldots, J, t = 1,\ldots, T; \tag{8-16}$$

$$c_j^1 q_{jt} \leq c_t y_{jt}, j = 1, \ldots, J, t = 1,\ldots, T; \tag{8-17}$$

$$\sum_{j=1}^{J} y_{jt} \leq 1, t = 1,\ldots, T; \tag{8-18}$$

$$x_{jt} \geq y_{jt} - y_{j(t-1)}, j = 1, \ldots, J, t = 1,\ldots, T; \tag{8-19}$$

$$y_{jt} \in \{0,1\}, j = 1, \ldots, J, t = 1,\ldots, T; \tag{8-20}$$

$$I_{jt}, q_{jt}, x_{jt} \geq 0, j = 1, \ldots, J, t = 1,\ldots, T. \tag{8-21}$$

The CSLS differs from the DLS model only in the restrictions (8-10) and (8-17), meaning that the production quantities can take a continuous value.

Bitran and Yanasse (1982) proved that the CSLS problem is NP-hard as a special case of the CLS problem.

8.3.12 Proportional lot sizing and scheduling (PLS) model

It is a *multi-item small-bucket* model. The basic idea is to use the remaining capacity for scheduling a second item in a particular period. It adapts the CSLS model to the case when the capacity of a period is not fully used. The remaining capacity is left unused. The PLS problem allows an unused capacity to process a second part type in a period. The objective function and most of the constraints are equal to the CSLS model. Similar to the

CSLS model, the idle periods between two lots of the same item do not cause additional setup costs.

The PLS model was proposed by Drexl and Haase (1995). The authors described the PLS as a model that produces continuous lot sizes over one or several adjacent or nonadjacent periods preserving the setup state over idle periods. This model also allows one changeover within each period. The PLS model assumes that at most one changeover is allowed within each period. According to this assumption, the continuous lot sizes can be computed over one or several adjacent or nonadjacent periods. In addition, the setup costs are calculated by looking back for one or several periods. As for the DLS and CSLS models, the PLS model is restricted to short-term production scheduling. The authors also developed an extension of the PLS problem, which included setup times (PLSPST), and the PLS problem with multiple machines (PLSPMM) as well as with multiple stages (Drexl and Haase 1995, 81–83).

The PLS model is as follows:

$$\sum_{j=1}^{J}\sum_{t=1}^{T}\left(s_j x_{jt} + h_j I_{jt}\right) \rightarrow \min \tag{8-22}$$

s.t.

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt},\ j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-23}$$

$$c_j^1 q_{jt} \leq c_t(y_{j(t-1)} - y_{jt}), j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-24}$$

$$\sum_{j=1}^{J} c_j^1 q_{jt} \leq c_t,\ t = 1,\ldots, T; \tag{8-25}$$

$$\sum_{j=1}^{N} y_{jt} \leq 1,\ t = 1,\ldots, T; \tag{8-26}$$

$$x_{jt} \geq y_{jt} - y_{j(t-1)}, j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-27}$$

$$y_{jt} \in \{0,1\}, j = 1, \ldots, J,\ t = 1,\ldots, T; \tag{8-28}$$

$$I_{jt}, q_{jt}, x_{jt} \geq 0, j = 1, \ldots, J,\ t = 1,\ldots, T. \tag{8-29}$$

The objective (8-22) is also to minimize the sum of the setup and holding costs for all items and all periods. The inequalities (8-24) ensure that the production of an item in every period can only be realized, if the machine is setup either at the beginning or at the end of that period. The restrictions (8-25) keep the total capacity requirement per period within the limits. The rest of the restrictions is similar to the previous models.

Drexl and Haase (1995) presented a *backward-oriented regret-based biased random sampling method*, which solved efficiently this model and its extensions. Other approaches can be found in the papers by Staggemeier and Clark (2001); Karimi, Fatemi Ghomi, and Wilson (2003); Jans and Degraeve (2008); Stadtler (2011); Copil et al. (2017).

8.3.13 General lot sizing and scheduling (GLS) model

The GLS problem consists in determining continuous lot sizes of several products and scheduling them on a single machine subject to capacity constraints. Deterministic dynamic demands, given over a finite planning horizon, are to be fulfilled without backlogging, so that the inventory holding and sequence-dependent setup costs are minimized. The GLS problem integrates lot sizing and scheduling of several products in macro-periods by subdividing the macro-periods into a predefined number of non-overlapping micro-periods. The computational complexity of the models was increased using this procedure. The GLS problem turned to be NP-hard (Drexl and Kimms 1997).

GLS is a *large bucket* model. It can be viewed as an extension of the CLS and DLS models, in which the scheduling decisions and sequence-dependent setup costs are incorporated. It also incorporates a user-defined parameter to restrict the number of lots per period. The parameters and decision variables are the same as for the DLS model. The objective is to minimize the total sum of the setup and holding costs.

The GLS problem was introduced by Fleischmann and Meir (1997), and by Drexl and Kimms (1997) a few months later. Then, Meyr (2000) extended the GLS problem to the general lot sizing and scheduling problem with setup times (GLSPST), which considered sequence-dependent setup times. Both problems were solved by the author using two solution heuristics, *threshold accepting* (TA) and *simulated annealing*, both were based on local search procedures.

The fundamental assumption for the GLS model is that a user-defined parameter N_T restricts the number of lots per period.

Let each lot be uniquely assigned to a position in order to define a sequence, and these positions are numbered as N_1, N_2,…, N_T. The first position in period t is denoted by

$$F_t = 1 + \sum_{\tau=1}^{t-1} N_\tau,$$

and the last position in period t is denoted by

$$L_t = F_t + N_t - 1.$$

The total number of positions and thus, the maximum number of lots, which can be formed, is:

$$N = \sum_{t=1}^{T} N_t.$$

The GLS model is as follows:

$$\sum_{j=1}^{J}\sum_{t=1}^{T} s_j x_{jt} + \sum_{j=1}^{J}\sum_{t=1}^{T} h_j I_{jt} \rightarrow \min \quad (8\text{-}30)$$

s.t.

$$I_{jt} = I_{j(t-1)} + \sum_{n=F_t}^{L_t} q_{jn} - d_{jt},\ j = 1, \ldots, J,\ t = 1,\ldots, T; \quad (8\text{-}31)$$

$$c_j^1 q_{jn} \leq c_t y_{jn}, j = 1, \ldots, J, t = 1,\ldots, T, n = F_t,\ldots, L_t; \quad (8\text{-}32)$$

$$\sum_{j=1}^{J}\sum_{n=F_t}^{L_t} c_j^1 q_{jn} \leq c_t, t = 1,\ldots, T; \quad (8\text{-}33)$$

$$\sum_{j=1}^{J} y_{jt} \leq 1, n = 1,\ldots, N; \quad (8\text{-}34)$$

$$x_{jn} \geq y_{jn} - y_{j(n-1)}, j = 1, \ldots, J, n = 1,\ldots, N; \quad (8\text{-}35)$$

$$y_{jt} \in \{0,1\}, j = 1, \ldots, J, n = 1,\ldots, N; \quad (8\text{-}36)$$

$$I_{jt} \geq 0, j = 1, \ldots, J, t = 1,\ldots, T; \quad (8\text{-}37)$$

$$q_{jn}, x_{jn} \geq 0, j = 1, \ldots, J, n = 1,\ldots, N. \quad (8\text{-}38)$$

The objective (8-30) is to minimize the total sum of the setup and holding costs. Equality (8-31) maintains the inventory balances. Considering that a particular item is allowed to be produced at several positions in a period, inequalities (8-32) guarantee that, if a lot for item j is scheduled at position n, the machine is in the correct setup state. Capacity restrictions are taken into account by the constraints (8-33). The restrictions (8-34) enforce a unique setup state. The position, at which a setup must take place, is determined by the inequalities (8-35). The binary conditions for the setup state variables are given in (8-36). The restrictions (8-37) and (8-38) are the non-negativity constraints.

If $N_t = 1$ for all $t = 1,\ldots, T$, then the GLS is equal to the CSLS.

Fleischmann and Meir (1997) presented a heuristic approach, which includes three variants of a local search algorithm, based on threshold accepting.

8.3.14 Batching and scheduling problem (BSP)

The usual assumptions are as follows. Each demand is characterized by a deadline and a size. The demands are interpreted as jobs, and the demand size determines the processing time of a job. The capacity of a machine, for example the speed, is constant over time and thus, the processing time of a job does not depend on the schedule. The jobs are not allowed to be split. This means that a certain demand is interpreted as one piece. Several demands (jobs) for the same item can be grouped together to form one lot with the goal to save the setup costs. This problem is referred to as a BSP rather than a lot sizing and scheduling problem. The objective is to minimize the total sum of the setup and holding costs. A solution of the BSP is uniquely characterized by the sequence, in which the jobs are scheduled, and by the completion time for each job. For more information, see Chapter 6, as well as the papers by Kuik, Salomon, and Van Wassenhove (1994); Bogaschewsky, Buscher, and Lindner (2001); Gaafar (2006).

8.3.15 Multi-level lot sizing and scheduling model

In a multi-level system, the raw materials are changed to end products by executing several operations. The production planning is not only considered for the final level, i.e., for the end products, but also for the components and subsystems that are involved into manufacturing. The parent-component relationship between items is considered, because the output of an operation (level) is an input for another operation. Therefore, the fulfillment of a demand at one level depends on the lot sizing decisions made at predecessor levels. As a result, multi-level problems are more difficult to solve than single-level problems. The usual objective is to determine the order quantities over a finite horizon and the inventory levels at each stage of the production process so that all demand requirements are satisfied at the minimum cost.

The parent-component relationship between items, also known in planning systems as BOM, is usually represented by an acyclic directed graph, where every vertex is an item, an arc represents the assembly or distribution relation between items, and the weight of an arc is the quantity relation, also called the '*gozinto factor*', between the two terminal vertices of the arc. Generally, for lot sizing problems there are considered four basic patterns of an inventory system, depending on the production flow:

- *Serial structure* - Each stage facility has at most one immediate predecessor and one immediate successor stage. It is the simplest

multi-layer configuration (Lambrecht, Vander Eecken, and Vanderveken 1983; Pochet 2001; Pitakaso et al. 2007; Ramezanian, Saidi-Mehrabad, and Fattahi 2013) (Fig. 8-2, a);

- *Assembly structure* - Each stage or facility in the process requires inputs from a number of immediate predecessor stages, while supplying, in turn, requires at most one immediate successor. An assembly system finishes with one final product (Lambrecht, Vander Eecken, and Vanderveken 1983; Kuik, Salomon, and van Wassenhove 1994) (Fig. 8-2, b);
- *Arborescent structure* - Each production stage has either a unique immediate predecessor or no predecessors at all, whereas the number of successors is unlimited (Kuik, Salomon, and van Wassenhove 1994) (Fig. 8-2, c);
- *General structure* – A multi-stage configuration, which does not satisfy one of the above definitions, is denoted as general structure. There may exist several end products that have some components in common. This situation is sometimes referred to as *component commonality* (Kuik, Salomon, and van Wassenhove 1994; Gicquel, Minoux, and Dallery 2008) (Fig. 8-2, d).

The system structure in a multi-stage production process (Fig. 8-2) can be represented by a directed acyclic graph $G(V, E)$, where V is the set of vertices, which represent the items, either parts or components, and E is the set of directed arcs, which denote the processing operations. An arc (i, j) leads from item i to the parent item j. The total number of parts in the process is $|V| = N$. The vertices have been numbered in a *topological order* by the integers $\{1, 2,\ldots, N\}$, i.e., the vertex $i \in$ V has been labeled by an integer $v(i)$ such that all arcs (i, j) satisfy the inequality $v(i) > v(j)$ for $i > j$.

A weight r_{ij} is associated with every arc (i, j). This weight denotes the number of item i used to produce an item j. In the assembly structure graph, the unique immediate successor of i is denoted by $s(i)$ and the immediate predecessor by $p(i)$ (Fig. 8-2, b). The set of all successors of i is denoted by $S(i)$ and the set of all predecessors by $P(i)$. A path $Q(i)$ from vertex i to vertex 1 is the set of vertices $(i = i_1, i_2,..., i_k = 1)$ of V satisfying $(i_j, i_{j+1}) \in E$ for $j = 1, \ldots, k - 1$. In serial structures, these sets $S(i)$ are a singleton for all items i and for a finished product i, $S(i) = \varnothing$. Note that the vertex, which produces the final product, is numbered by 1.

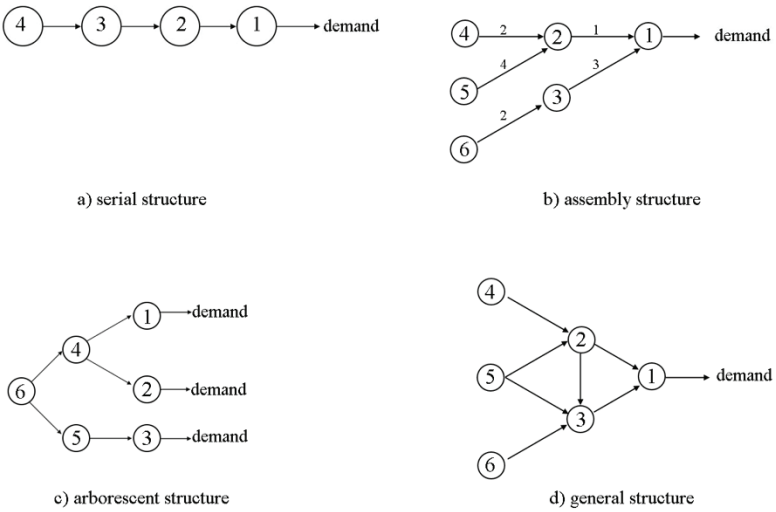


Fig. 8-2. Basic structures of multi-level inventory systems.

According to the above definitions, the corresponding values for Fig. 8-2, b are:

- Required quantities:
$r_{21} = 1$, $r_{31} = 3$, $r_{42} = 2$, $r_{52} = 4$, $r_{63} = 2$;

- Immediate successors:
$s(2) = 1$, $s(3) = 1$, $s(4) = 2$, $s(5) = 2$, $s(6) = 3$;

- Sets of all successors:
$S(2) = \{1\}$, $S(3) = \{1\}$, $S(4) = \{2, 1\}$, $S(5) = \{2, 1\}$, $S(6) = \{3, 1\}$;

- Sets of immediate predecessors:
$p(1) = \{2,3\}$, $p(2) = \{4,5\}$, $p(3) = \{6\}$;

- Sets of all predecessors:
$P(1) = \{2,3,4,5,6\}$, $P(2) = \{4,5\}$, $P(3) = \{6\}$;

- Paths:
$Q(1) = \{1\}$, $Q(2) = \{2, 1\}$, $Q(3) = \{3, 1\}$, $Q(4) = \{4, 2, 1\}$, $Q(5) = \{5, 2, 1\}$, $Q(6) = \{6,3, 1\}$.

Multi-level production planning models in a small bucket time variant include the following model classes:

- Multi-level discrete lot sizing and scheduling (MLDLS) model (Kimms 1996);
- Multi-level proportional lot sizing and scheduling (MLPLS) model (Kimms and Drexl 1998);
- Multi-level general lot sizing and scheduling (MLGLS) model (Fandel and Stammen-Hegene 2006).

The MLDLS and MLPLS models enable simultaneous lot sizing and scheduling. Nevertheless, they consider a limited number of products to be elaborated per period.

The MLGLS model was proposed by Fandel and Stammen-Hegene (2006). This model pretends to take the advantages of the MLPLS and MLCLS models based on subdividing the macro-period into a fixed number of micro-periods. It integrates lot sizing and scheduling of several products in each period.

These models have a high level of complexity caused by the large number of variables. Therefore, only problems with a reduced number of products, machines, and periods can be optimally solved. Mohammadi et al. (2010) proposed the formulation of a mathematical model for lot sizing and scheduling in a flow shop environment with sequence-dependent setup times.

The multi-level capacitated lot sizing (MLCLS) problem is a large bucket model, in which several products can be processed in a given macro-period, but it cannot determine the lot sizes and schedules simultaneously. Most contributions on the multi-level lot sizing problem use large bucket models and a general product structure (Gicquel, Minoux, and Dallery 2008),

Due to the complexity of the problem, the main approaches are branch-and-bound and dynamic programming algorithms for an optimal solution and heuristic approaches for efficient solutions. There are two general heuristic approaches: level-by-level and period-by-period. *Level-by-level* procedures assign the production quantities to a specific facility over all periods and use the resulting production program as requirements for all immediate predecessor stages. These heuristics start with the finishing facility (the facility producing the final product, i.e., stage 1). On the contrary, multi-level procedures assign the production quantities to all stages on a *period*-by-*period* basis. A review of the corresponding heuristics can be found in the paper by Lambrecht, Vander Eecken, and Vanderveken

(1983). The period-by-period approach is of practical importance because it considers the cost structure and the product structure of the system simultaneously.

8.4 Conclusions

The importance of analyzing the production planning problem in all aspects of its complexity, considering its immanent dynamic, is always highlighted in the corresponding literature. One of the most challenging and rewarding subjects in this area is shop floor lot sizing and scheduling, which directly affect the plant performance metrics, such as the productivity and the utilization rate of the machinery and the space. The decisions, which are made to meet the final product demand requirements and to minimize the system costs, including the setup, production, and holding costs, are essential to enhance the competitiveness of a company in the market. The lot sizing optimization problem is common for big manufacturing plants. For instance, the semiconductor industry is such an example. The complexity of the problem impedes the modeling because of its high dimensionality and high production costs.

Since Wagner and Whitin (1958) have published their seminal paper on the dynamic lot sizing problem, referring to Ford Whitman Harris' formula (Harris 1913), this model is still the starting point for a variety of models, which deal with a wide spectrum of medium to long term planning problems in industries with lot processing. This subject has remained a topic of interest for researchers and planners in developing variants of the basic models. Different real-world phenomena are there captured, such as capacity or resource restrictions, setup times and costs, multi-level and multi-item versions, backlogging of demand, rolling planning horizons, deteriorating inventory due to a limited shelf life of items, among many others. These extensions lead usually to an increase of the problem complexity, making worse the solution time and the decision quality.

Lot sizing problems can be classified according to the principal features, such as the time scale, the time horizon, the demand variety, the shop floor structure, the inclusion of the capacity constraints, and others more. The existing approaches have arisen from the need to shorten the gap between traditional theoretical models and planning problems that occur in real life. Both experimental and theoretical results point the necessity to design and implement flexible tools that are capable of handling dynamics from different production aspects.

The solution methods can be classified into three main categories: exact methods, common sense or specialized heuristics, and mathematical

programming-based heuristics. Exact methods improve the MIP formulation of the problem using generic models, so that commercial solvers such as CPLEX or XPRESS-MP are able to process practical instances using branch-and-bound-type procedures, see Belvaux and Wolsey (2000). These solvers can also provide lower bounds for practical heuristic algorithms. A mathematical programming-based heuristics use optimal search procedures to generate a solution for different problems.

This study of the subject revealed a lack of literature for practical use, which would be used by planners, teachers, and students of different levels. The lot sizing theory requires a systematization and a popularization. There is also observed an insufficient attention of the researchers to problems with a deterioration effect in different production environments, which are common for the actual level of real-life modeling, see the papers by Abad (2003); J. C. Ho, Solis and Chang (2007); Chakraborty, Giri and Chaudhuri (2009). Various problems with data uncertainty are considered by Liu (2008); Li and Hu (2017). Sometimes, both effects appear together in a lot sizing problem (J. C. Ho, Solis, and Chang 2007).

A typical assumption in the actual literature is that the lot sizing input data are deterministic, i.e., all information that defines a problem instance is known with certainty in advance. Nevertheless, in real life, many forms of uncertainty affect a production process. These uncertainties can be categorized into two groups:

- Environmental uncertainty, for example, demand and supply uncertainty;
- System uncertainty, for example, operation yield, quality, and system failure uncertainty.

Therefore, random fluctuations and a sensitivity analysis should be discussed in incoming models to investigate the impact of random factors on the decision quality.

In this chapter, a formal modification was made. During the study, it was noted that *capacitated lot sizing* and *continuous setup lot sizing* models are abbreviated in the literature in a very similar manner, namely as CLS and CSL models, respectively. This similarity provokes confusions and difficulties for the reader. To avoid such mistakes, the abbreviation CSLS is proposed in this book for the *continuous setup lot sizing* model.

For more detailed surveys and reviews of the relevant works done in this area, the interested readers are referred to the papers by Elmaghraby (1978); Potts and Van Wassenhove (1992); Drexl and Kimms (1997); Staggemeier and Clark (2001); Karimi, Fatemi Ghomi, and Wilson (2003); Brahimi et

al. (2006); Zhu and Wilhelm (2006); Ben-Daya, Darwish, and Ertogral (2008); Gicquel, Minoux, and Dallery (2008); Jans and Degraeve (2008); Robinson, Narayanan, and Sahin (2009); Glock, Grosse, and Ries (2014).

CHAPTER NINE

RESCHEDULING

Researches on rescheduling help bridge the gap between the theory and practice of production scheduling.
(*Qiao et al. 2018, 9*)

An important aspect of the production control in manufacturing plants is the generation of a schedule and its update when needed. It is a very complex decision-making activity focused on doing the best allocation of a set of limited resources to jobs over time when optimizing an objective function. A Semiconductor Manufacturing System (SMS) is a representative example of the most complex and stochastic production environments, where the scheduling problems are among the most difficult ones in the industry (Wen, Fu, and Huang 2001, 3559; Senties et al. 2010, 555; Yao et al. 2011, 125).

It is unusual that the original schedule can be completely executed without any alteration since most of the manufacturing environments are dynamic and stochastic. Consequently, managers and production planners must fulfill two objectives at the same time: a) generation of good quality schedules, and b) quick reaction to unforeseen events, reviewing the schedule and only updating it when the impact of the disturbance warrants it. Although the occurrence of some random events can be anticipated applying predictive scheduling methodologies, it is practically impossible to anticipate precisely in advance what events will occur in real time. Commonly, those events, which are not considered during the initial schedule generation, can greatly affect the system, making the schedule non-viable in the execution due to the differences between the original state and the actual conditions on the shop floor. When such a situation arises, a partial or complete rescheduling is mandatory to minimize the impact of the disturbance in the performance of the system. There are many types of disturbances, which upset the plan. In Section 9.2, the most common ones in manufacturing systems are described. Usually, in the theoretical research it is supposed that rescheduling is done periodically with a fixed horizon.

But in practice, it has been reported that a moving horizon is employed in most industrial settings (D. Gupta, Maravelias, and Wassick 2016, 91).

As the schedule execution evolves, also the probability that unexpected events happen, and it becomes less probable that a schedule follows the initially specified sequences. Eventually, the changes in the shop floor can be of such a magnitude that rescheduling is needed.

This chapter is focused on the rescheduling problem, covering: a) basic aspects of uncertainty; b) uncertainty sources in manufacturing systems; c) rescheduling framework, including the environments, methods, and strategies. A concise overview of some representative cases for each strategy studied is also included.

9.1 Basic aspects of uncertainty

In highly dynamic and stochastic manufacturing environments, a schedule can rarely be executed as it was originally generated because of unexpected disruptions and random events that alter the shop-floor state. Hence, one of the major issues in the scheduling problems for those environments is to determine how to deal effectively with unexpected disruptions. The inclusion of uncertainty into a schedule and the evaluation of the schedule quality after including uncertainty are not easy tasks. The difficulty is mainly caused by the uncertainty being a complex phenomenon in real manufacturing situations. Each type of uncertainty has a different impact on the situation. For example, the impact of a machine breakdown in the production is different, if it occurs just when the schedule begins its execution or it occurs in the final stages. Therefore, it is important to identify the most common types of uncertainties for each particular shop floor: how they arise, what they imply for a particular situation, and which types of uncertainty should be incorporated into a schedule. According to the taxonomy of uncertainty proposed by Aytug et al. (2005, 91), there are three key dimensions of uncertainty - *cause, context, impact* - which are useful for the formulation of a problem. Additionally, the authors suggested that the same dimension can have different facets.

A *cause* can be viewed as an *object* or a *state*. Possible instances of a *cause* as an *object* are: material process, resource, tooling, employees, etc. Possible instances of a *cause* as a *state* can be: ready, not ready, high quality, low quality, damaged, healthy, etc.

A context represents an environmental situation at a certain time. Such situations can be defined as *context-free* or *context-sensitive* depending on whether the *context* has or does not have an influence on the expectations about some particular performance metrics.

An *impact* measures how the uncertainty affects the schedule execution from different viewpoints, such as time, material, quality, dependency, and context. Another important aspect of uncertainty, which is considered in the problem definition, is the cost imposed on the system by disruptions. Aytug et al. (2005, 93) identified three types of costs to be incurred in anticipation of a disruption:

- A disruption does not occur. The time or the resources are wasted;
- A perturbation effectively take place;
- The system is reconfigured, either after or during the disruption.

Until now, most of the scheduling researches were *context-free*, since the contextual information was not included into the problem during the modeling. Recently, with the emergence of intelligent manufacturing, the necessity to establish a smart factory arose. For the implementation in a smart factory, the status quo and manufacturing requirements should be taken into account. Production planning and scheduling are the key processes in a smart factory, and they are highly dependent on the contextual information and the physical environment context, such as the status of the shop floor at the schedule execution time, to deal adequately with the uncertainties and the data flux. There exist many research proposals reiterating the use of ontologies for representing a context in a smart factory. Ontology-based applications are able to optimize the scheduling of the manufacturing resources. Also, important advantages are offered, such as the possibility of a high-level knowledge inference, checking the consistency and the soundness of the knowledge represented in the ontology (B. Chen et al. 2017, 6515–16).

9.2 Uncertainty sources in manufacturing systems

In the production environments, there exist a myriad of uncertainty sources, which can change the system status affecting its performance. Not every event triggers an update of a schedule. Ideally, rescheduling must be applied when the deterioration in the performance is significant, with the intention to reduce the impact of the disturbance and restore the schedule operability. These events are called *rescheduling factors*.

The most common factors identified in rescheduling studies are related to the load capacity, job orders, or both (Geng, Jiang, and Chen 2009, 900–6). According to Katragjini, Vallada, and Ruiz (2013, 783), the most common disruptions related to the capacity load are:

- Machine breakdowns (most frequently studied);
- Non-available tools;
- Absence of operators;
- Deterioration of machine efficiencies.

On the other hand, the most common disruptions related to job orders are:

- Rush orders;
- Priority changes;
- Variations in the processing times;
- Order cancellations;
- Rework;
- Lack of material or material transportation delays.

In an SMS, the production process is variable, dynamic and has plenty of uncertainties. The uncertainties are derived from the production process, which involves complex product flows, jobs with reentrancy, a quick change of the orders, concurrent manufacturing of distinct products, and multiple steps of a variable length in the production cycles. The complexity of an SMS induces a huge quantity of interruptions. Some of them are related to the job orders or the load capacity (similar to the ones described in Section 9.1). Other ones are particularly related to the volatility of the SMS's market, which is constantly changing the product demands (Chien, Wu, and Chiang 2012, 860–61). Other unexpected events are directly related to the production process itself, for example interruptions derived from simultaneous manufacturing of different products.

9.3 Framework

While two different types of notation and classification schemes have been developed in a concurrent way for scheduling problems (Graham et al. 1979; Brucker et al. 1999), a mathematical classification scheme for rescheduling has not been developed up to now being a problem intrinsically more complex, large, and stochastic than traditional scheduling. Nonetheless, there are a few researchers, which have tried to understand, organize, and systematize the knowledge about rescheduling. In this direction, it is worth mentioning the contributions by Vieira, Herrmann, and Lin (2003, 43–48); Aytug et al.(2005, 94–104); Ouelhadj and Petrovic (2009, 419–21); García-Mata, Márquez-Gutiérrez, and Burtseva (2015); Gupta, Maravelias, and Wassick (2016, 92). The results of

these organizational attempts have been translated into classification schemes sharing some common concepts but they are different enough from each other, so that a brief summary is recommendable.

One of the most relevant frameworks oriented to the manufacturing domain has been proposed by Vieira, Herrmann, and Lin (2003, 43–48). The authors classified the rescheduling problems according to the environment, strategies and methods. The main concepts of this framework are described in the subsequent sections.

Later, Aytug et al. (2005, 94–103) reviewed the literature in the domain of scheduling with uncertainties. This research was driven on how the problems were formulated, reporting the following classification for scheduling under uncertainty:

- Completely reactive approaches;
- Robust scheduling approaches;
- Predictive–reactive scheduling.

Ouelhadj and Petrovic (2009, 419–21) focused the research on dynamic scheduling (scheduling in the presence of an ample variety of real-time events) and classified the related problems into four categories:

- Completely reactive scheduling;
- Predictive-reactive scheduling;
- Robust predictive-reactive scheduling;
- Robust proactive scheduling.

Most recently, D. Gupta, Maravelias, and Wassick (2016, 83–89) reviewed the advances in rescheduling. The analyzed literature was classified into two categories:

- Reactive scheduling;
- Scheduling under uncertainty.

According to these authors, the *reactive scheduling* term comes from the fact that rescheduling is traditionally performed as a reaction to an event. Under this concept, a nominal schedule is generated without taking into account any kind of uncertainties. Upon unexpected disturbances, the interrupted schedule is reorganized to update the remaining (un-executed) part. Updating a schedule can be achieved either by heuristics (task-time shifting, mixing-splitting of batches, etc.), or by a regeneration and re-optimization of the execution of the remaining plan. *Scheduling under*

uncertainty includes those models that account for uncertainty a priori, i.e., resilient schedules capable of absorbing the effect of uncertainty to avoid changes in the shop floor. *Robust optimization* techniques produce "robust" solutions, that is, solutions that are immune to an extent against uncertain data. *Stochastic programming* provides a way to protect the decisions against imperfect future information, using probability theory to model uncertain values of diverse information, for example: job processing times, product demands, the reduction in the plant capacity, etc. At the end, Gupta, Maravelias, and Wassick (2016, 90–92) conclude that in practice, event-triggered rescheduling has some shortcomings, which can be addressed if rescheduling is approached as an online problem.

A comparison of the differences and similarities among these taxonomies shows that the rescheduling frameworks by Vieira, Herrmann, and Lin (2003, 95–96); Aytug et al. (2005, 95–96) share two strategy classes in common: *dynamic*, also called *online* or *completely reactive*, and *predictive-reactive*. Aytug et al. (2005, 95–96) added a third strategy called *robust scheduling*. The objective in the *robust scheduling* approaches by Aytug consists in reacting to the interruptions creating a new schedule, but also in taking care about the schedule performance, minimizing as much as possible the impact of the disturbances.

Then Ouelhadj and Petrovic (2009) subdivided the *robust strategy* into two further ones, namely the *robust predictive* and the *robust proactive* strategies plus the two strategies originally proposed by Vieira, Herrmann and Lin (2003, 49–54) (*completely reactive*, and *predictive-reactive*). In the rescheduling classification by D. Gupta, Maravelias, and Wassick (2016, 84–89), reactive scheduling and scheduling under uncertainty have similarities with the classifications proposed by other researchers. The *reactive scheduling strategy* coincides both with the dynamic strategy as well as with the completely reactive strategy described by Vieira, Herrmann, and Lin (2003, 45–51), while the class of *scheduling under uncertainty* corresponds to the domain of the literature reviewed by Aytug et al. (2005, 94–103).

In Fig. 9-1, a rescheduling framework is presented. It is mainly based on Vieira's proposal and extended with additional strategies introduced by Ouelhadj and Petrovic (2009). Three aspects were appreciated to be considered in solving rescheduling problems: environment, strategies, and methods.

Rescheduling environments are classified into two types, which in turn are subdivided into two and three classes, respectively. These are: *static (deterministic, stochastic)* and *dynamic (cyclic production*, *flow shop*, and *job shop) classes*.

Rescheduling strategies comprise four categories: *dynamic, predictive-reactive, robust predictive-reactive*, and *robust proactive*.

Usually, the approaches using a *dynamic strategy* either build a partial schedule or do not build any schedule at all. These strategies only use dispatching rules to react to disturbances. The use of dispatching rules appears to be intuitive, easy to understand, and it does not impose a high computational load.

Rescheduling Enviroments				
Static (finite set of jobs)		Dynamic(infinite set of jobs)		
Deterministic (all information given)	Stochastic (some information uncertain)	No arrival variability (cyclic production)	Arrival variablility (flow shop)	Process flow variability (job shop)

Rescheduling Strategies				
Dynamic (no schedule)		Predictive-reactive	Robust predictive-reactive	Robust pro-active
Dispatch rules	Control-theoretic	Rescheduling policies		
		Periodic	Event-driven	Hybrid

Rescheduling Methods				
Schedule generation		Schedule repair		
Nominal schedules	Robust schedules	Right-shift rescheduling	Partial rescheduling	Complete regeneration

Fig. 9-1. Rescheduling framework, extended from Vieira, Herrmann, and Lin (2003, 44).

The *predictive-reactive* strategy is the most frequently applied one in stochastic environments. It operates in two phases: 1) in the *offline phase*, a predictive schedule is designed; 2) in the *execution phase*, as a reaction to some disruption, the original schedule is partially or completely rebuilt.

Both *robust predictive-reactive* and *robust proactive* strategies share commonalities with a predictive-reactive strategy and include different types and levels of uncertainties that are embedded into the initial and predictive schedule. Likewise, the interruptions caused by some unexpected events are attended according to the rescheduling policies previously

established. Furthermore, robustness and stability objectives are also considered in these strategies (Fig. 9-1).

The next sections about this subject are dedicated to study in detail the rescheduling environments, methods, and strategies. Some illustrative examples based on the published works about scheduling strategies are also included. It is important to clarify that many of these references come from diverse applications in SMSs.

9.4 Environments

There are two main classes of *manufacturing environments: statics* and *dynamics* (Vieira, Herrmann, and Lin 2003, 45–47). In a *static environment*, the number of jobs is finite. Besides, a static environment can be deterministic or stochastic. In a *deterministic static environment*, all information is known in advance. In a *stochastic static environment*, there are random variables that affect the schedule execution. Dispatching rules or other policies are typically used to attend a disruption. Instead, a *dynamic environment* operates with an infinite stream of jobs. Depending on the production model, *dynamic environments* have been classified into: a *cyclic* one, a *flow shop*, and a *job shop.* In a *cyclic dynamic environment*, there is no variability or uncertainty in the job flow. The scheduling is designed only once, and then production scheduling is repeated continuously. A d*ynamic flow shop* presents only variability in the job arrival to the production plant. However, once a job starts its processing, it flows from the first until the last machine. *A dynamic job shop* is variable in the process flow, meaning that every job follows a proper processing pattern.

The previously reviewed classification offers a synthetic vision on manufacturing systems, which are complex, dynamic, and stochastic systems. In manufacturing environments, there exists a great variety of products, processes, and manufacturing methods. The objective of a production schedule is to obtain a better coordination among these activities, increasing the productivity and minimizing the operating costs. A production schedule is useful to identify a conflict in the consumption of the resources to control the job release and to guarantee that raw materials have been available on time. Once the schedule is finished, the execution starts, assuming that the major part of the execution time will fit into the initial schedule as much as possible. In practice, it is possible that small deviations happen with respect to both the start and end points. However, when an unexpected event interrupts the schedule execution and the original schedule tightly follows, a situation occurs, which is called *smooth* shop. However, when an unexpected event interrupts the original schedule, a reaction is required, for

instance a delay or finishing of a job out of date. The unexpected or stochastic events change the state of the system and affect the production performance. If the deterioration is very significant, the event activates rescheduling in order to reduce the impact.

McKay and Wiers (2006, 55–57) proposed a taxonomy and a categorization for the solution methods based on the grade of uncertainty.

A dynamic shop with a significant uncertainty is called a *stress* shop, and the real-time events, which provoke such an uncertainty, are classified into two categories:

- *Resource-related* - machine breakdown, operator illness or absenteeism, unavailability or tool failures, loading limits, delay in the arrival or shortage of materials, defective material loading (material with a wrong specification), etc.;
- *Job-related* - rush jobs, job cancellation, due date changes, early or late arrival of jobs, change in the job priority, changes in the job processing time, rework or quality problems, over- or underestimation of the process time, etc.

These interruptions trigger other actions, for example overtime, in-process subcontracting, process change or rerouting, machine substitution, limited manpower, setup time, etc.

9.5 Methods

Rescheduling methods are employed in predictive-reactive, robust predictive-reactive and robust proactive strategies. Basically, these methods consist of updating or rebuilding an interrupted schedule in different ways. Vieira's taxonomy (Vieira, Herrmann, and Lin 2003, 53–56) proposed to categorize rescheduling methods into two subclasses:

- Schedule generation;
- Schedule repair.

At the same time, the *schedule generation* considers a nominal and a robust schedule. *Nominal schedules* are focused on optimizing the performance. However, not much attention is paid to include uncertainties or to indicate a way to handle disruptive events. This type of schedules has been vastly studied in the literature.

Uncertainties are not included into deterministic scheduling, but they are included into reactive scheduling. It is required to include some methods to

react to the disturbances. The selection depends on the chosen repair method, which defines when and how a disturbed schedule will be updated. The basic methods are:

- *Right-shift rescheduling* - the original schedule is repaired for all jobs that are affected by a shift to the right. The size of the shift is equal to the time of the machine breakdown;
- *Partial rescheduling* - only the affected jobs are considered for the repair. Unlike right-shift rescheduling, the sequence of the execution in the new schedule can be reassessed and modified in order to improve the schedule performance;
- *Complete rescheduling* - all parts of the original schedule are reworked, even those jobs, which were not affected by the disruption, except those parts related to the jobs that have been already processed.

The concepts of right-shift rescheduling and partial rescheduling are illustrated in Fig. 9-2. There are seven jobs (J1, J2, J3, J4, J5, J6, and J7) and three machines (M1, M2, and M3). The jobs J1, J2, J3, J4, and J6 are processed by the three machines, while job J5 is only processed by machines M2 and M3. Job J7 is only processed by M3. Machine M2 is down for a time period of length *t.* This time interval is illustrated in the figure by the pattern of crossing lines.

9.6 Policies

There exist many ways to solve a scheduling problem with uncertainties, from only reacting to the disturbances using dispatching rules or other heuristics to prioritize jobs waiting for processing, to design schedules with different kind of robustness. While the strategy specifies how rescheduling is performed, a rescheduling policy specifies when it should be done.

Three types of *rescheduling policies* are known (Vieira, Herrmann, and Lin 2003, 56–58):

- *An event-driven policy* is the most popular and most frequently used one. Rescheduling is triggered by some random event that modifies the current system status;
- *A periodic policy* is also studied and widely applied. This policy is used to generate schedules in the regular periods previously established. Typically, a dynamic problem is broken into static problems, which can be then solved using classical scheduling

techniques. The resultant schedule is executed without updating until the next period;

- *Hybrid policies* combine *event-driven* and *periodic policies*. One popular way to combine these two policies is known under the concept *rolling time horizon* (see Section 8.2.1).

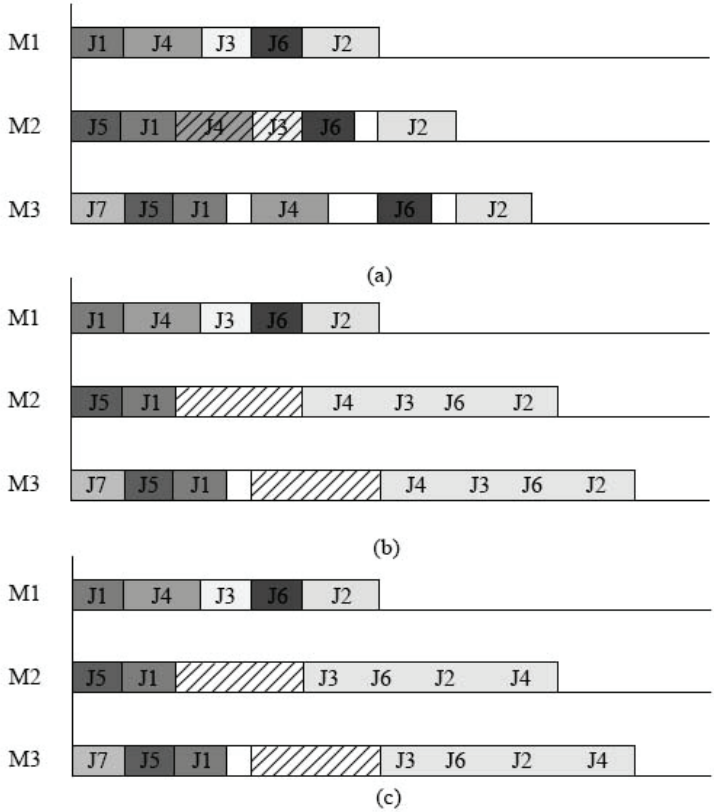


Fig. 9-2. The Gantt chart for the example: (a) The original schedule for the machines M1, M2, and M3. For M2, the pattern of the crossing lines represents the time period when machine M2 is down; (b) A modified schedule after applying the right-shift method for repairing. For clarity, all rescheduled jobs are colored in gray; (c) The reschedule obtained after applying the partial method for repairing. Notice that by this method, the jobs affected by the rupture of M2 are also displaced for M2 and M3, but the job processing order can be readjusted, as it happened in this case.

A great variety of ways exists to approach the rolling time horizon, depending on diverse factors, such as:

a) Application type;
b) Aspects of uncertainty, which are taken into account (cause, context, impact, and schedule uncertainty inclusion);
c) Objective function;
d) Implementation technique.

In general, the application of a *rolling time horizon* policy is as follows. First, a deterministic schedule for a finite period and for all machines is developed deliberately ignoring every event that could happen in the future. After generation of the first schedule, the next schedule is generated for the subsequent period, and so on (Z. Li and Ierapetritou 2010, 5888–5901). A hybrid policy uses a rolling time horizon. It updates the schedule periodically and also whenever an urgent event happens, for example, a machine breakdown.

Usually, rescheduling strategies are combined with rescheduling policies. Four of the most common strategies for controlling the production in dynamic rescheduling environments are described in Section 9.7.

When no schedules are created and the jobs are processed according to dispatching rules based on the actual condition of the shop floor, the strategy is known as *dynamic scheduling, online scheduling or reactive scheduling.*

For the case when a schedule has been generated, three rescheduling strategies are known: *predictive-reactive, robust predictive-reactive,* and *robust proactive.* In the *predictive-reactive strategy*, any kind of uncertainty is included into the original schedule. In the *robust predictive-reactive* and *robust proactive rescheduling strategies,* different kinds of uncertainties are included, generating either a robust schedule or a robust and stable schedule. Depending on the rescheduling policy and the rescheduling strategy selected, it is then decided how and when reactions to a disruption will be activated during the schedule execution.

9.7 Strategies

In this section, a brief description of each rescheduling strategy is presented. Additionally, some study cases are analyzed and classified according to the strategy used to deal with the uncertainties.

9.7.1 Dynamic scheduling

It is important to highlight that there is no unique definition of *dynamic scheduling*. Vieira, Herrmann, and Lin (2003, 44) wrote that *dynamic scheduling* does not create schedules and at most, it creates partial schedules. These partial schedules are generated by means of heuristics or dispatching rules. Dynamic scheduling is used in factories handling a high-volume but low-mix production and also in automated systems that need to react quickly to disruptions. Vieira's definition for *dynamic scheduling* is called *completely reactive* by Aytug et al. (2005, 94). Other authors interchangeably use the terms *dynamic scheduling, reactive scheduling,* and *online scheduling* (Sabuncuoglu and Goren 2009, 139), while Suwa and Sandoh (2013, 55–58) classify *online scheduling* into *dispatching* and *schedule revisions* with the respective sub-classifications.

The proposed subclasses for the *schedule revisio*n strategy are:

- Periodic schedule revision;
- Reactive scheduling.

The revision of a *periodic schedule* is done both cyclically and by unexpected events. Suwa's *reactive scheduling* is also called *predictive-reactive* by many other researchers. It is studied in Section 9.6.2. Suwa's definition of dispatching matches with Vieira's concept for *dynamic scheduling* and with Aytug's definition of *completely reactive scheduling*, while D. Gupta, Maravelias, and Wassick (2016, 91) considered rescheduling as a special case of *online scheduling*. They also posited a framework for online scheduling. Gupta's definition of *online scheduling* is the same definition that Suwa and Sandoh (2013, 50) assigned to a *periodic schedule.*

The determination of the most convenient dispatching rule for each application is not an easy task, because there are many different types of rules that have been developed over the years. To avoid a blind search within the set of dispatching rules to be selected, simulation methods are frequently used. In view of the complexity of an SMS, many authors first model the complete fabric environment or at least a section under study. Considering the enormously large, dynamic, and stochastic environment of an SMS, mathematical modeling languages such as Petri nets are usually used for the description of distributed systems.

Modeling the behavior of the system is necessary to obtain the information about the actual state of a factory, including one of the most relevant characteristics of the SMS, the *reentrancy* (pattern flow of the

manufacturing lines, where a product must pass through a production step more than once). For example, Wen, Fu, and Huang (2001, 3560–64) modeled and simulated a scheduling problem for a wafer factory by a combination of queuing theory and color-timed Petri nets (CTPNs). A genetic algorithm that dynamically searched for an appropriate dispatching rule implemented a scheduler. The experimental results showed that the genetic algorithm-based scheduler had a superior performance compared with conventional dispatching rules. Qiao et al. (2013, 199–202), reported a case of SMS scheduling, which was modeled by hierarchical colored timed Petri nets (HCTPNs). An approach, which employed the HCTPN and an extended genetic algorithm (EGA), was used to study how to optimize a combination of scheduling policies. The results obtained by a simulation system showed a near to optimal schedule.

In a similar way, Lee, Jiang, and Liu (2009, 869) as well as Liu, Jiang, and Fung (2009, 124–27) proposed an approach, which was based on time extended object-oriented Petri nets (TEOPNs). This approach was used for the SWFS performance modeling, real-time dispatching and simulation. The TEOPNs were applied to describe the SWFS as a series of objects. Coincidentally, in both reports previously indicated, some dispatching rules were developed via a *dynamic bottleneck dispatching algorithm*. The performance was evaluated by Liu, Jiang, and Fung (2009, 127), employing the simulation architecture SWFS. Lee, Jiang, and Liu (2009, 867) proposed a new multi-objective scheduling and real-time dispatching (MSRD) approach, which basically consists of two main modules:

- Offline multi-objective scheduling;
- Online real-time dispatching.

The performance evaluation was done via a simulation built on a platform, which combined the MSRD prototype and the TEOPNs.

In the carried out studies, the virtual SWFS was derived from a real plant located in Shanghai (Y. F. Lee, Jiang, and Liu 2009; H. Liu, Jiang, and Fung 2009). The authors obtained similar results showing that a dynamic dispatching had a better performance than both the critical ratio (CR) + FIFO) and EDD dispatching policies.

A different approximation to modeling a reentrancy problem in a semiconductor manufacturing plant was reported by Coron, Kawski, and Zhiqiang (2009). The authors characterized the reentrancy problem as an optimal control problem governed by the scalar hyperbolic conservation law using partial differential equations.

Another methodology that seems well situated for solving complex scheduling problems, such as those in SMSs, is named multi-agent system (MAS). MASs are well suited for ill-structured problems and have some advantages, such as heterogeneity, high variability, high complexity, high flexibility, and high robustness against failures. The most remarkable advantage of an MAS is that the agents react locally and independently to the changes required (Leitão, Mařík, and Vrba 2013). Many researchers have taken advantage of the superior capacities of an MAS to deal with the randomness and dynamism of complex problems to solve the scheduling problem in the manufacturing environment. A pioneer research of this type was presented by Mönch, Stehli, and Zimmermann (2003), who proposed a new architecture of an agent-based system for the production control of an SMS. This model includes scheduling implemented by a multi-layer hierarchical scheme. Other similar researches can be met in the literature (J. Lin and Long 2011; Vrba, Mařík, and Kadera 2012; D. Z. Zhang and Anosike 2012).

Lin and Long (2011, 5232–34) developed a distributed simulation platform for semiconductor manufacturing. The platform architecture was structured into three layers:

- Network communication layer;
- Middle-ware layer (based on JADE);
- Multi-agent simulation layer.

This platform was tested simulating the manufacturing process of a real semiconductor factory in Shanghai.

In order to fulfill the overwhelming demand requirements in modern production systems and keep the firm competitive, the production scheduling and execution control should be tightly coupled. This strong integration in manufacturing execution systems (@MES) is essential to achieve an adaptive behavior due to the interactions between a set of agents acting as autonomous managers, as it was reached by the agent-based modeling and simulation (ABMS) tools, proposed by Rolón and Martínez (2012) for a @MES distributed design. In these approaches, a bio-inspired technique in Holon manufacturing systems was used (Giret and Botti 2004). The agents showed emergent behaviors comparable to a complex adaptive system. Besides, the agents were well suited for the modeling processes, where each agent must adapt and modify its own behavior over time. Each agent has an autonomy to solve the disruptions related to its function. The fulfillment of the goals was achieved by collaborating with each other, and each agent individually defined and performed its own actions. The agents

communicated with each other and were coordinated by a dynamic Gantt chart. According to the simulation results, the interaction mechanisms among the agents were stable and robust in spite of the total autonomy of all agents and the absence of a master schedule.

9.7.2 Predictive-reactive scheduling

The predictive-reactive strategy has been approached in various ways by the researchers. The initial schedules are generally designed offline, and whenever an unanticipated event occurs during the running time, a partial or complete rescheduling is done based on the previously established rescheduling policies.

Usually, a predictive-reactive schedule is built through an iterative process consisting of the following three steps:

1. *Evaluation.* The objective of this step is to evaluate the impact of a disruption. If the impact does not significantly affect the schedule execution, any programmed action follows, and the schedule continues running without a change.
2. *Solution*. This step is dedicated to find rescheduling solutions that improve the schedule performance. There is no predetermined way to decide which is a better solution for a given rescheduling problem. Therefore, the solution step is usually the most difficult one.
3. *Revision*. In this step, the executing schedule is updated or a new schedule with more convenient results is generated.

Recently, new theories and methodologies are being introduced to represent the domain knowledge for rescheduling problems. Muñoz et al. (2011) reported a study case found in a chemical factory with multi-product batches. The unexpected event considered was an increasing operation time. The objective function was the maximization of the profit of the plant considering income and energy costs. The modeling of the problem was based on approximated dynamic models and ontologies. Usually, the online and historical information among different decision levels is independent and not properly integrated during the rescheduling process. However, the ontologies can be used to incorporate this information as it was done in this study case.

H.-H. Hung, Liang, and Chen (2013) employed a predictive-reactive strategy for a scheduling problem in the photolithography area of a semiconductor wafer factory. The goal was to compare the effectiveness and efficiency of three algorithms: simulated annealing, a genetic algorithm,

and tabu search to get an approximate reschedule. The authors also proposed a *sensitivity search method* to improve the performance. This method works as follows: independently of the moment when rescheduling is required, the initial schedule is used as the starting point to search for a new schedule. The experimental results showed that a sensitive search improved the performance. In particular, tabu search turned out to be superior to the other proposed algorithms.

A *mixed-line production* is popular in certain types of current production systems and emerged as a way to meet the market demand. Under this production scheme, a customer order consists of high-mix and low-volume products. Typical manufacturing systems with a mixed-line production are electronics and wireless communication industries. (H.-H. Huang et al. 2013, 65–68) solved a scheduling problem for this kind of systems, adopting the DBR technique (see Section 2.6.3) of the Theory of Constraints (TOC) and the buffer management. The application of these techniques helps the managers to detect early production problems and to evaluate the desirability of rescheduling in advance. Rescheduling is done depending on the evaluation results of different random events such as:

- Product combination (changes, cancelation or rush orders);
- Materials (shortage of raw materials);
- Resources and products (machine breakdown, absenteeism of workers, quality problems);
- Others (rework, processing time is over or underestimated, delayed progress).

Depending on the evaluation results, rescheduling is applied using a DBR algorithm. It is based on the bottleneck workstation as the critical scheduling target and applies a buffer management method to reduce the frequency of rescheduling. Another research objective was to compare the DBR and EDD techniques and to study the impact of the delayed orders on the performance metrics, such as the maximum tardiness time and the cost as well as the total completion/flow time. The results showed that the DBR method was better than the EDD technique, because the first one minimized the longest tardiness in the customer orders. It also offered both a greater flexibility and ability to manage the scheduling activities of a plant with an overloaded capacity and frequent disruptions.

9.7.3 Robust predictive-reactive scheduling

The objective of a robust predictive-reactive scheduling is the generation of robust schedules by minimizing the interruption effects in the performance. This strategy is applicable to highly stochastic systems, where the predictive schedule being interrupted during its execution is common. Occasionally, the magnitude of a disruption forces a rescheduling. However, it is not convenient to reschedule at each disruption. Instead of applying a rescheduling policy triggered every time when a disturbance happens, a better option is to reschedule just whenever the interruption dimension makes the running schedule non-viable (Church and Uzsoy 1992).

There are numerous ways to generate a robust schedule. Typically, two criteria are concurrently considered: shop efficiency and stability. The *stability* measures the deviation from the original schedule in relation to the new schedule. Generally, the instability grows as the number of changes during the updating grows, meaning that the updated schedule differs significantly from the predictive schedule.

A robust solution for any predictive-reactive scheduling problem is still an open question. Nevertheless, multiple approaches and methods are constantly being proposed and tested.

In the paper by Van de Vonder, Demeulemeester, and Herroelen (2007), the results of an experiment designed to evaluate several predictive reactive resource-constrained project schedules were reported. The projects of this kind are stochastic versions of a basic scheduling problem in a deterministic setting, known as the resource-constrained project schedule problem (RCPSP). In an extended RCPSP, stochastic durations of the activities were considered. The optimization objective was to minimize the expected makespan. Another objective was to guarantee that the schedule robustness would not be affected by disturbances. Vonder's complete experiment consisted in evaluating all possible combinations of the three baseline schedules obtained for different procedures with four reactive rescheduling procedures. The impact on the performance was measured whenever there are range variations in the following parameters:

1. Level of uncertainty in the activity span;
2. Weighting parameter (ratio of the dummy end activity to the average of the rest of the activities);
3. Project due date (timely project completion probability, TPCP).

The baseline scheduling methods evaluated were:

1. RCPSP-predictive exact procedure with the average duration of the activities;

2. Suboptimal procedure consisting of simple priority-based scheduling heuristics, specifically of the latest start time (LST) priority rule;
3. Resource flow dependent float factor (RFDFF), which is a suboptimal procedure targeted to minimize the stability cost function.

The four reactive scheduling methods were:

1. An RCPSPS-reactive method, consisting of a complete rescheduling by an exact procedure, where only the pending activities at the disruption time are considered for rescheduling;
2. The Fix Flow heuristic, which is no complete rescheduling under the railway concept, that is, an activity never starts earlier than its assigned starting point in the baseline schedule;
3. Activity-based priority (ABR) rules. The problem is solved by heuristics. The solution is a list of activities rather than a schedule;
4. An exact solution procedure for the resource-constrained project scheduling problem with weighted earliness/tardiness penalty costs (RCPSPWET) is turned into a reactive scheduling, making sure that at each rescheduling point, a new projected schedule is created with the lowest stability cost (in terms of the deviation from the original baseline schedule).

The generation of a new projected schedule includes the following steps to fulfill these objectives:

- The due date for an activity is set to its projected finish time in the baseline;
- The dummy end activity is equal to the project due date. It is assumed that the unit earliness and tardiness costs of a non-dummy activity j are identical and equal to the activity weight w_j;
- The tardiness cost of the dummy end activity is set to zero, while the tardiness cost is equal to the weight w_n.

The conclusion from this experiment was that even when exact procedures were used to generate proactive and reactive schedules, the TPCP optimization objective showed the best results. However, the stability objective was not improved. In general, considering the results obtained in all experiments, the authors concluded that for strong requirements in a TPCP with not too tight due dates and low values of the duration variability, it is preferable to generate a robust (stable) proactive scheduling based on

the RFDFF heuristic. However, for highly variable environments and low values of the TPCP, the RFDFF heuristic is not the best option. In these cases, a better option is to combine a procedure that generates a minimum duration baseline schedule with a stability-improving reactive policy, such as the Weighted Earliness/Tardiness (WET) one. In conclusion, Van de Vonder, Demeulemeester, and Herroelen (2007, 206) advised to conduct more studies about robust reactive scheduling in order to improve the WET results.

Kuster, Jannach, and Friedrich (2010) proposed the x-RCPSP extension to surpass the inherent limitations of the RCPSP for repairing the schedule. The proposed x-RCPSP can be used to formulate a wide range of practical problems for the disruption management. In essence, the x-RCPSP is based on a distinction between valid and invalid activities. The activities were turned on the base of predefined substitution rules and different constraints, however, only valid activities were considered for rescheduling. Applying this procedure, an x-RCPSP instance can be continuously modified. Additionally, these authors proposed a generic approach to partial schedulable stochastic realistic environments, called Local Rescheduling (LRS). The LRS was based on a time window that was extended in a bidirectional way to search for potential solutions. These potential solutions must fulfill the new requirements imposed by the occurrence of stochastic events. Experimentally, it was found that LRS outperforms other approaches.

A similar approach and objectives were addressed by Huang et al. (2010, 1277–80) in their Job Shop Rescheduling Repair (JSSR) problem. The goal of this problem was to obtain a stable repair of the schedule through a commitment between the makespan optimization and the performance deviation during rescheduling. The problem was formulated as a Disjunctive Temporal Problem (DT), framed as an Optimal Constraint Satisfaction (OCS), and solved by an algorithm integrating an incremental consistency and an efficient candidate generation.

9.7.4 Robust proactive scheduling

In this strategy, the objective is to obtain a robust and stable schedule. To accomplish this goal, which include the uncertainties, are offline created. In the theory, the resulting schedule must be insensitive to the disruptions. Unavoidably, some non-anticipated exogenous events occur during the execution of a proactive schedule. These disturbances cause frequently the schedule to be partially or completely repaired. The updated plan has to maintain the same stability and performance as the original one.

Robust proactive approaches are classified into three subcategories (Lou et al. 2012, 312):

- *Redundancy-based approaches* reduce the impact of the uncertainties by allocating the time and extra resources;
- *Probabilistic approaches* obtain probability density functions of the uncertainties;
- *Contingent/policy-based approaches* establish scheduling policies attending any particular sequence of the events.

Besides the stability measures, it is also important to know how the disturbances and rescheduling policies affect the system performance. In robust schedules, the number of disturbances can be correlated with the system performance. Diverse studies have been developed to measure how the disruptions affect the system performance (V. Kumar et al. 2008; Ghezail, Pierreval, and Hajri-Gabouj 2010).

Bonfill, Camarasa, and Puigjaner (2008, 1692) suggested a stochastic model and an optimization approach to solve a rescheduling problem of batch processing. Initially, a proactive schedule was generated including the uncertainty measures of loading, heating, and discharging. The uncertainties were characterized by a uniform distribution. The optimization objective was a combination of the makespan and the waiting times. The schedule was obtained using an optimized genetic algorithm. Whenever a machine was broken or the processing times were larger than expected, a new schedule was calculated applying the right-shift rule. Finally, the authors developed an experiment comparing the performance of the algorithms under deterministic and stochastic approaches. They found that even when the makespan and the waiting times were optimized for a deterministic environment, the makespan increases under a stochastic scenario by about 4%. Despite the simplicity of this problem, the experimental results showed a significant information supporting benefits when the uncertainties were incorporated into the schedule from the beginning.

In the research reported by Zakaria and Petrovic (2012), a predictive-reactive approach with non-reshuffle and reshuffle strategies was proposed for FMSs. The only source of disturbances considered by the authors was a situation when new job orders arrive before finishing the scheduled jobs. The challenge was to integrate the new job orders into the existing production schedule immediately, while preserving the factory performance and stability. By the non-reshuffle strategy, new orders are assigned to the machines just in the available idle times, whereas in the reshuffle strategy, the operations are re-sequenced to generate a partial solution within the

rescheduling horizon. The performance measure was a commitment between the sum of weighted squared tardiness, the makespan and the stability. The stability was calculated on the base of three aspects: machine migration, job start time, and sequence deviations. The implementation of the reshuffling and non-reshuffling strategies was done by genetic algorithms. The experimental results showed that the non-reshuffle strategy was a better option than the reshuffle one because it improved the sum of weighted squared tardiness. At the same time, the stability was highly increased without increasing the cost of the makespan.

Novas and Henning (2010) proposed a framework oriented towards multi-product and multi-stage plants addressing a repair-based reactive scheduling problem. The framework objective was to represent the context knowledge at the disturbance time to properly identifying the rescheduling problem class and the suitable rescheduling action types (for example, shift-jump, reassign, freeze, etc.). Once the problem specification was ready, this was used to create a Constraint Programming (CP) model. Alternative solution scenarios focusing on stability and regular performance measures were proposed. Multi-product multi-stage batch plants operate with the following inter-stage storage and operational policies:

- Unlimited intermediate storage (UIS);
- Non-intermediate storage, unlimited wait (NIS-UW);
- Non-intermediate storage, zero-wait (NIS-ZW).

The knowledge about the manufacturing environment and the production plan was explicitly represented and modeled with object-oriented techniques. Certain static information about the resources (for example, the properties of the most relevant entities and solution methods) was included into the domain knowledge. The temporal attributes of the resources were considered as well. The chosen scheduling policy was an event-driven heuristic together with a partial rescheduling method. The goal of this proposal was to give an immediate response to the events without introducing excessive changes into the schedule and at the same time, to maintain the system stability. By means of the domain representation, the current state of the scheduling in the process can be known whenever an unanticipated event occurs. Thus, at any moment an event happens, the context can be precisely obtained and used to render the rescheduling problem specifications. Otherwise, the incorporation of contextual information could allow a more precise evaluation of the impact of an event. However, the authors proposed this improvement as a future work. The contextual information was only used to evaluate whether a schedule

becomes useless by the effect of the occurrence of some events, and to decide whether rescheduling proceeds. Finally, once the rescheduling was completely specified and the performance measures have been selected, the model was generated by a CP updating module.

Nova's approach has the effect of reducing nervousness in the production line and at the same time, it maintains an acceptable optimization. The use of the contextual information demonstrates that the evaluation of the impact of an event allows us to distinguish the cases, where the size of the disturbance mandates a rescheduling reducing the rescheduling cases and avoiding an unnecessary updating. These experimental results also demonstrated that better values of the optimization objectives were obtained when a more complete knowledge about the manufacturing process was included.

9.8 Conclusions

Since the inception, most of the scheduling problems have been addressed by a deterministic approach. However, in the last two decades, a paradigm shift has emerged going from a static scheduling point of view to a more realistic perspective. This paradigm shift can be explained by two main reasons:

1. It is imperative to fill the gap between purely theoretical investigations and scheduling solutions towards practical problems in the industry;
2. The difficulty of scheduling problems in manufacturing plants continues to increase making the production processes extremely complex.

Therefore, uncertainties are playing a transcendental role, and more researchers are focusing on dynamic and non-deterministic scheduling problems. The paradigm shift makes it indispensable paying attention to the formalization of the replies to disturbing events, which affect the schedule execution. In other words, it is important to have a reference framework to understand and classify different aspects involved into rescheduling.

Following this order of the ideas, and despite the time elapsed since Vieira, Herrmann, and Lin (2003, 42–58) published their seminal paper about rescheduling, this work continues being one of the most influential and referenced papers about this topic. Therefore, the backbone of this chapter is built around the rescheduling framework developed by these authors. However, considering that other rescheduling strategies have emerged since then, the original rescheduling strategies proposed by Vieira

et al. were extended to robust predictive and robust proactive strategies (Ouelhadj and Petrovic 2009, 418). In addition, an analysis of some selected cases published in the literature regarding to each one of the studied strategies was also included into this section. The majority of the analyzed references correspond to applications in SMSs. These papers were selected considering that SMSs are highly complex, stochastic, and dynamic, implying a great variety of scheduling problems, mainly with lot processing.

It is important to highlight that despite an overabundance of methodologies, which address uncertainty in different production and scheduling environments, these proposals are coming mainly from the academic studies and in reality, they are not implemented in practice. The practical utility of the theoretic approaches on rescheduling is compromised for the little attention that is paid to incorporating all possible uncertainties in the modeling phase. This dichotomy between the industrial and the academic worlds are due to the fact that in real plants, uncertainty permeates everything from the system parameters, such as the job processing times, yields, etc.

Additionally, in stochastic and dynamic production environments, it is impossible to anticipate every unexpected event, which frequently has a very disruptive and strong impact in meeting the planned objectives. In recent years, some researchers have pointed out the shortcomings of rescheduling to deal with scheduling problems in complex and stochastic environments, and instead they are proposing that this problem can be approached as an online problem. Online scheduling is an ongoing process, in which evolving and changing circumstances force continually the reconsideration and revision of the pre-established plans. Online scheduling has a goal to maintain a balance between stability and efficiency versus optimality.

Due to the randomness and complexity of manufacturing systems plus diverse factors such as changes of the market, the emergence of new technologies and mainly the emergence of new manufacturing paradigms, such as Industry 4.0, both, industries and researchers, are faced to new challenges. This means that even smart factories must be flexible, reconfigurable, adaptable, and efficient. However, to endow a factory with these qualities is not an easy task because the resultant manufacturing system will be every time bigger and more complex. It becomes evident that a methodology, which is different from the traditional one, is required to fulfill all these challenges. The self-organizing manufacturing systems (SOMS) methodology has been identified by some researchers as a methodology with a potential to approach the solution of smart factories (J. Zhang et al. 2017) Although it is common knowledge that this methodology

has technical limitations, which restrict its applicability, the emergence of new technologies offers new opportunities, making the SOMS paradigm viable.

As a final note, it is important to highlight that many terms are used in rescheduling in an imprecise form and the knowledge is not properly organized, for instance, the classification schemes are not uniform, distinct terms are used to identify the same concepts, and even the same nomenclature means different things by distinct authors. This proves that rescheduling is still an immature field, in which there is no common vocabulary yet, and a deeper and updated research is required to understand and organize the domain knowledge.

For more detailed surveys and reviews of the relevant works done in this area, the readers are referred to the papers by Ayutug et al. (2005); Wazed, Ahmed, and Nukman (2010) Mönch et al. (2011); Esmaeilian, Behdad, and Wang (2016).

CHAPTER TEN

LOT PROCESSING IN HT INDUSTRIES

As new concepts are introduced in manufacturing, the research community should be able to identify future research frontiers.
(*Esmaeilian, Behdad, and Wang 2016, 80*)

The necessity to cover the gap between the studied models and real requirements of production systems is permanently highlighted in the modern planning/scheduling literature. In this chapter, three projects are described that were developed by the authors of this book for corporations of the Baja California state, Mexico, which is a region with a high presence of HT industries, where the efficiency of the production organization has an extremal cost.

The first project was related to the application of some ideas of GT to the electrical test area in a semiconductor packaging factory with the goal to reduce the setup times. In the second project, an HFS structure with several specific characteristics was detected in the production of televisions. A genetic algorithm was proposed to solve the problem of minimizing the makespan. The last project was related to an efficient handling of the WIP. A heuristic algorithm was proposed for the selection of the prime material with appropriated characteristics. In fact, in the plant, there did not exist a formal tool for selecting the material. The decisions were taken empirically, based on former experience. This has led to the generation of an excessive WIP and other negative effects. All these projects were supported by the plant administrations.

10. 1 Product family formation in a packaging factory

This project was focused on the planning system at the areas of an electrical test in a semiconductor packaging factory, which has high-volume and high-mix (HV/HM) characteristics of the production. There are realized some assembly operations and an electrical test of the manufactured items. This process is time consuming and requires hundreds of machines, which occupy big areas of the plant. A model was proposed to allow the best use of the installed capacity. A GT approach was employed to minimize the

machine idle time due to tooling changeovers. A paradigm shift, which handles the production planning on the family level rather than on the product level, was suggested focusing on the market priorities. The research was made for Skyworks Solutions, Inc., Mexicali, Mexico.

10.1.1 Problem description

In the electrical test area of a semiconductor HT company with the characteristics of HV/HM, the machine downtime caused by a lot changeover was out of control. The detected root cause was the method of the machine allocation order when the first machine available is assigned to the next lot. Moreover, it did not consider that the lot change time was different for different sequences of the products.

The production model is composed of two machine platforms (M1 and M2), each one with a number of parallel identical machines. There is a difference in the performance of the production process on different microcircuits due to the nature of the electrical test on the electronic components. This observation implicates preferences in allocating a product to a predetermined platform for processing to avoid additional adjustments. The individual adjustment of a machine may take from a few minutes to some hours before processing a production lot, depending on the similarity of the adjacent products in the workflow. Consecutively, the lot changeover time on a machine is strictly dependent on the sequence of the lots. Given the high volumes, diversity, and the frequent changes of the product nomenclature at the plant, the minimization of the machine break times due to a setup implies a considerable reduction of the flow time. It also decreases the penalties and quantity of the involved machines, facilitates rescheduling, improves the machine loading and consecutively, decreases the production costs.

The objective of this project was to develop a flexible model for the company planning to realize: a) a makespan reduction, b) a delivery on time, and c) the minimization of the manufacturing cost.

To achieve these goals, the workflow in the company and the information flow in the electrical test area have been studied, the setup structure was analyzed, and the manufactured products were grouped into families on the basis of their similarity in the geometries. The grouping was performed by employing the main idea of GT: the products must be sorted out into groups according to their design or manufacturing attributes, such as shape, size, surface texture, material type, raw material, etc., see Kusiak, Vanelli and Kumar (1985); Tatikonda and Wemmerlöv (1992); Mathirajan and Sivakumar (2006); Pickardt and Branke (2012); Mutingi and Mbohwa

(2016). The technical similarities of the products within a group were used to schedule lots in batches. With this, the setup time on the machines may be reduced essentially. It was proposed to limit the changeover options for the lots, which have the same product geometry, to form production batches, which do not require major setups in-between.

This batching categorically reduces the idle time due to the activities of the change or adjustment of a tool (Delgado-Arana et al. 2017).

10.1.2 Workflow model of the production planning of the company

The production planning follows the workflow described below, also described in Fig. 10-1:

- Supply chain management (SCM) - prepares the MPS;
- Industrial engineering - receives, validates, and provides a feedback to the master production plan (MPP) to determinate the production capacity constraints along the production line. The Industrial Engineering Department is responsible to indicate whether the processing of the required volume is feasible;
- Raw materials - receives, validates, and provides a feedback for the raw material constraints to avoid delays;
- Production Control - develops a detailed production plan for the current week (t) and for the next week (t+1); reviews the production plan proposed by SCM to agree with the final version of the volume planned to deliver; prepares a detailed daily production plan for the plant with the following goals:
    a) Fill the installed capacity;
    b) Achieve cost absorption levels;
    c) Fulfill the *on-time delivery* (OTD) orders;
    d) Release the orders to the workshop floor.

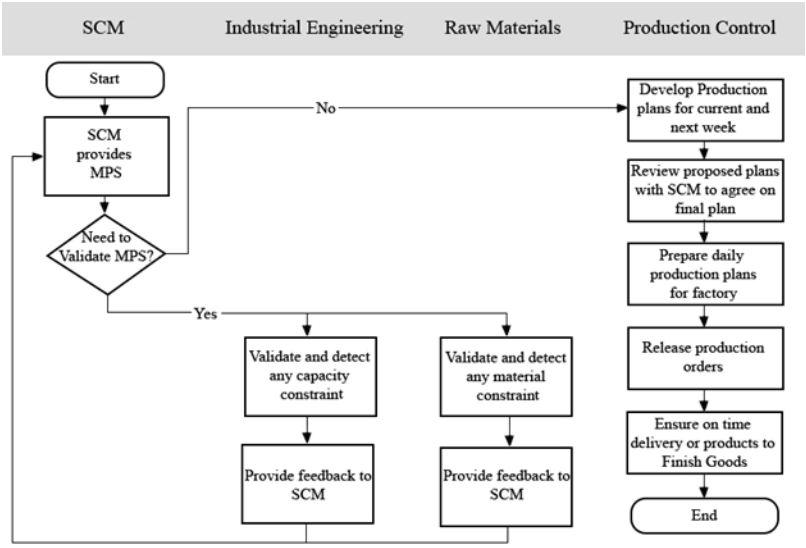


Fig. 10-1. General workflow of the production plan.

10.1.3 Grouping the products into families

The size of a microcircuit or package, which is referred to as *Package Size*, and its height H (*Mold Cap*) are taken as geometric attributes of a microcircuit as it is shown in Fig. 10-2.

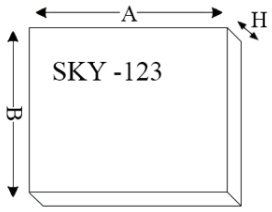


Fig. 10-2. Product geometry support. Package size: A × B; Mold Cap: H.

The company product catalog and the product portfolio were analyzed to extract all geometries of the products announced in the manufacturing process. Table 10-1 shows the number of geometric variations in the portfolio, where approximately 67% of the geometries are active and form the main part in the product categorization.

Products	Quantity	Geometries	Heights
Active	566	70	11
Inactive	533	34	5
Totals	**1089**	**104**	**16**

Table 10-1. Product portfolio characteristics.

As it can be seen in Table 10-1, there are well-founded reasons to manage the planning with a focus on a product family rather than just a part number level. To gain additional capacity, there are advances in the process flexibility, since it is commonly known that the plans are never carried out as planned.

Current business planning systems carry out their processes at the product level and therefore, there is no visibility of similar products, which can resemble each other. That is why for a planner, a transition from one product to another one is realized at the same planning level. However, at the operational level, there are resource restrictions for tool changes, in addition to the fact that a small change (recipe or tool) and a product dimension change are not the same, and a great part of the production capacity is lost due to setups.

A family includes all those products that share the same attributes, i.e., in the given case the geometric sizes of a microcircuit. When these attributes are similar, major adjustments between the lots are not required. Consequently, sequencing becomes simpler and the decision time is smaller (Andrés et al. 2005, 273). When planning is focused on a product family rather than on a part number (product) level, the planner flexibility is improved by the information about the compatibility of the products, which belong to the same family. The knowledge about the compatibility of the products is enough to protect the execution of the plan by replacing the part numbers, which wait for processing.

The three product types are established according to the A-B-C categories of the inventories by the APICS[13] classification as a function of the volumes required by every geometry. In this classification, category A has the highest priority, and C has the lowest one. Table 10-2 shows the different types of the geometry together with the volumes required within the period. As a result, 65.90% of the demands are concentrated in 35 part numbers (high volume and high frequency for the products of priority type A).

[13] American Production and Inventory Control Society.

Priority	**A**	**B**	**C**
Characteristics	High volume/ High frequency	Middle volume/ Middle frequency	Low volume/ Low frequency
Quantity	65.90%	24.75%	9.35%
Parts	35	114	407
Number of Geometries	8	12	50

Table 10-2. Volume-priority relationship considering the type and geometry of the products.

Based on this analysis, the following policies were proposed for the planning according to the priorities that consider the grouping of products into families:

1. Load the equipment capacity assigned to a family with products of priority type *A* first (confirmed orders by the customers);
2. Once a product of priority type *A* completed its allocation, a product of priority type *B* of the same family is assigned (to buffer demand peaks);
3. Once the products of priority type *B* complete their allocation, a family of priority type *C* is assigned (to forecast the future demands).

Following these ideas, an algorithm was proposed to generate and maintain product families according to their geometries (Fig. 10-3). The obtained results were used to calculate the theoretical amount of the equipment required to cover the needs of the period.

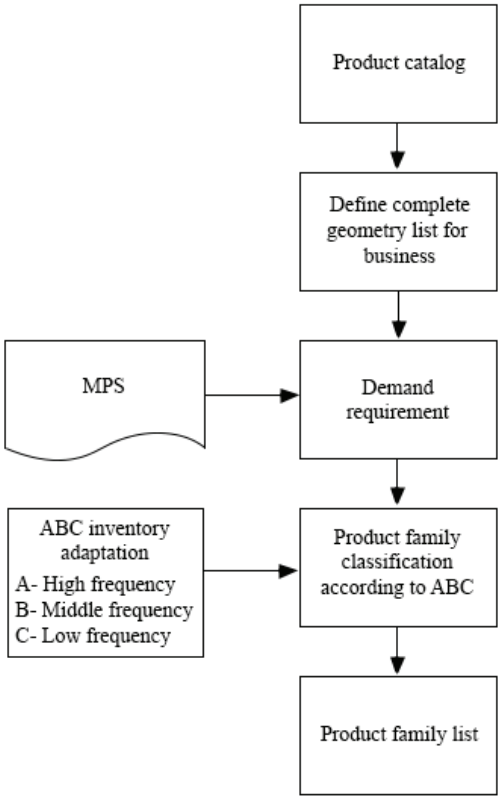


Fig. 10-3. An algorithm to generate and group the product families by geometries.

10.1.4 Tool changeover taxonomy

Any change in the production process leads to the generation of an idle time, in which a machine is not productive, that is, it does not process items and therefore, planning and reducing these changeovers help to improve the effectiveness of the company and the results of the business. A study of the operating times was carried out for different changeover processes on the equipment at the electrical test area. The following elements were found to be time consuming, see Fig. 10-4 for more details.

The setup types were sorted according to the length, from low to high, as follows:

A lot setup: the raw material is taken from the warehouse and placed at the equipment. This activity involves an internal cleaning and removal of any piece from the previous lot in order to eliminate any risk of contamination. Beginning with the lot on the equipment, place trays with raw material to be tested at the entrance of the handler. Empty black and red trays in the corresponding exit spaces.

A recipe setup: a recipe is searched in the database, downloaded to the computer, and saved. A lot is fed. The corresponding formats are filled to register the change. Finally, the information in the monitoring system is updated.

A tool setup: this change is made whenever the product is changed and the symmetry of the test tools (*contactors*) used does not coincide with those of the previous product. This activity consists of the following steps: uninstallation/installation of the electrical test tools; obtaining the calibration pattern, installing the pattern; correlation of the variables to calibrate the electrical test; and finally, removal of the pattern from the equipment, which is returned to the tool store.

A family setup: this change is made as long as there is no geometric compatibility in the product attributes in the sequence, and it leads to activities such as a change of the handler, tools, the recipe, and the lot.

Setup time	Setup type	Setup characteristics	Activities
-	Lot setup	Same product	Purge equipment Blower and cleaning New lot feed
+	Recipe setup	Different product Same tool Same geometry	Recipe uploading Variable correlation Lot setup
++	Tool setup	Different product Different tool Same geometry	Tool installation Tool fine tuning Recipe setup
+++	Family setup	Different geometry	Handler installation Handler fine tuning Tool setup

Fig. 10-4. Setup comparison.

It can be seen in Table 10-3 that the changeover times vary depending on the type of the setup, the smallest one being the lot change, and the largest one being the change of the family.

Geometric ranges	Changeover type	M1 (minutes)	M2 (minutes)
M1 3 to 3.9	Lot setup	10 +/- 2.5	8 +/- 2
	Recipe setup	30 +/- 5.5	45 +/- 12.3
M2 1.6 to 2.8	Tool setup	90 +/- 13.2	135 +/- 51.4
	Family setup	290 +/- 62.3	430 +/- 93.2
M1 4 to 6.9	Lot setup	10 +/- 2.5	8 +/- 2
	Recipe setup	30 +/- 4.8	45 +/- 7.9
M2 2.9 to 4.5	Tool setup	90 +/- 7.8	98.2 +/- 35.4
	Family setup	210 +/- 42.1	340 +/- 38.4
M1 7 to 11	Lot setup	10 +/- 2.5	8 +/- 2
	Recipe setup	30 +/- 3.2	45 +/- 5.4
M2 4.6 to 5.5	Tool setup	90 +/- 6.2	89 +/- 22.1
	Family setup	170 +/- 33.5	260 +/- 25.1

Table 10-3. Standard time of changeovers.

10.1.5 Modeling the lot sequence

To model the lot sequence, many activities were performed, starting with the definition of the setup types to set the relationship 'product geometry - setup time'. It was fixed, which adjustment is required when a lot change occurs. Subsequently, a study of the workshop information flows was realized. This analysis allowed building a general model of lot sequencing. From the study of the information flows in the test area and an analysis of the setup structure, a general planning model was obtained, as it is shown in Fig. 10-5. The demand signal and the capacity analysis report, which are raised by the MPS, were used as input in the general lot sequencing model of the company. The amount of necessary machinery, which may be theoretically assigned to each family, was defined. It was made involving the information on the demands, the knowledge of the characteristics of the products and flows that provide data to obtain a classification at the family level, and using the delivery priority when sequencing on the machines. In the model, the lots are sequenced according to their priority: first those with the highest priority (produced on demand), followed by those with medium priority (produced for the inventory) and finally the sequence of orders with the lowest priority (produced for forecast).

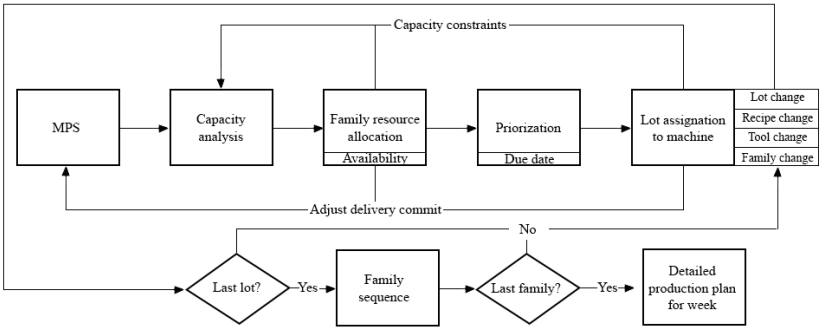


Fig. 10-5. General model of lot sequencing.

The matrix of the machine changeovers according to the setup types was developed for the products, which belong to the same family (Fig. 10-6). It was assumed that a minor setup corresponds to a lot change. If the next product in the sequence shares the same installed tools, then a recipe change is performed. If the next product in the sequence is not compatible with the installed tools, then a tool setup is done.

Tool	Part No	A	B	C	D	E	F	G
XY	A	Lot setup	Recipe setup		Tool setup			
	B		Lot setup					
	C	Recipe change		Lot setup				
VW	D	Tool setup			Lot setup	Recipe change		
	E				Recipe change	Lot setup		
TU	F						Lot setup	
RS	G							Lot setup

Fig. 10-6. Morphology matrix of the change within a family.

Only in the case when the geometry of the next product is different, a family setup is incurred. The individual family matrices were consolidated into a single matrix that includes all families, which were extracted from the product catalog as stated in Fig. 10-7. It was assumed that to move from one family to another one, a changeover time must be taken to perform the

corresponding activities: setting the handler, tool, recipe, lot, and cleaning. If a change occurs between products belonging to the same family, these times are minimal. Currently, there are 83 product families included into the matrix.

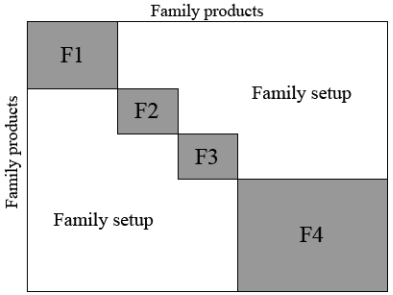


Fig. 10-7. Morphology matrix of the family tool changes.

Since each family has different adjustment times, three standard ranges of the geometry combining with the type of the assigned test machine were created. This can be seen in Table 10-3. The duration of the changeover corresponds to its type. These times are first separated by the two main platforms that the process uses (M1 and M2), followed by three levels of geometric ranges. When the last lot of the family is prioritized, the sequence model is obtained and one continues to proceed with deliveries and assigning machines. Upon reaching the last lot of the last family, the production plan for the current period is acquired. The normal period of the sequence is one week, and the plan is made to start on Saturday at 12:00 AM and to end on Friday at 11:59 PM. The above means that each of the machines has 168 available hours according to the standard that the company has. Usually, 10% of the available time is considered as slack, which leaves only 151.2 hours per week and machine. This time contains the production processing time plus the idle time due to a changeover (lot, recipe, tool, and family).

By designing the changeovers on the machines, the cycle time of the activities corresponding to each setup type was validated. In addition, the setup types were documented by the taxonomy data, which were taken directly from the plant and by the standards presented in Table 10-3. This information was used to quantify the transition time between the lots and batches.

10.1.6 Conclusions

This work helps the planner to make a detailed short-term plan at the family level for the following decisions:

- Assign the required machines;
- Attend the product grouping into families;
- Act quickly when a part number does not arrive as planned.

In this case, it is clearly observed that there are major advantages of planning at the family level. As it was demonstrated, a gain of 25.93% in the installed capacity was saved. Saving a quarter of the capacity, the company could attract more customers, produce more products, increase the delivered volume to customers, and definitely reduce the operating cost. This enhances the profitability of the given semiconductor company.

The robustness of this model was tested, first under simulation conditions, using for this purpose the real information from the company databases, and then testing the model under controlled conditions or in a pilot test (Delgado-Arana et al. 2017). The data from a sample production month were used first to perform a simulation and then to compare the process data by different stated scenarios. Once these activities were performed, a planning algorithm was implemented for the selected representative family to confirm the efficiency of the lot sequencing model. The proposed model can be used for any discrete manufacturing business, which has to sequence the production orders.

10.2 Minimization of the makespan with multiple restrictions for the production of televisions

This section presents a solution for the problem of minimizing the makespan with multiple restrictions for a real production environment of an electronic television industry, specifically in a part of the plant called "Auto Insertion", where automated machines participate. The model represents an HFS with unrelated machines, sequence-dependent setup time, machine eligibility, and limited buffer.

The investigation was made for SONY de Mexicali, S.A. de C.V.

10.2.1 Problem description

The television manufacturing process has three sections (Fig. 10-8):

1. Auto Insertion: Components are inserted into a PCB in an automated way;
2. Manual Assembly: Operators assemble and insert large components manually;
3. Final: Tests are carried out and final packaging of the finished product is made for the transportation to the distributors.

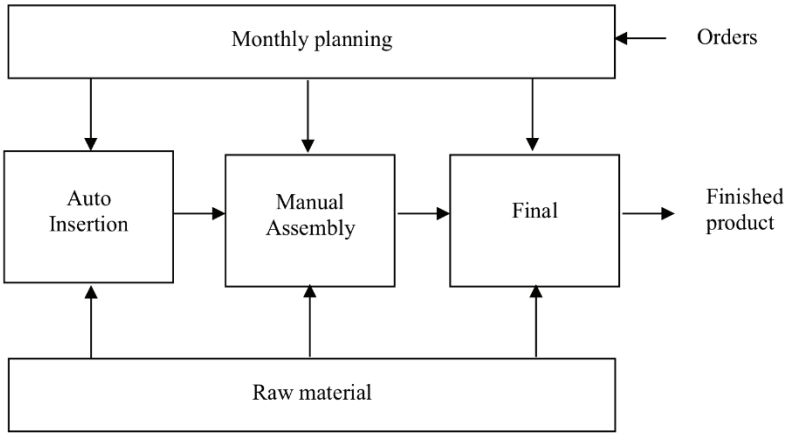


Fig. 10-8. Process of manufacturing televisions.

The number of PCBs needed for the assembly of the televisions was calculated from the monthly production plan, and the order of their processing on the machines was determined. In this factory, the processing of 500,000 boards monthly on average for different TV models is usual. The monthly planning is subject to changes during the current month due to variations in the orders and external circumstances that are managed daily in the 'Final' section.

The Auto Insertion section is addressed because it is a section, where automated machines are involved. The processing times on each machine are statistically established. The section has machines distributed according to the processes they perform:

1. Eyelet (Ey) - make holes to insert components;
2. Jumper (Jv) - insert jumpers to connect circuits;
3. Axial (Ax) - insert axial components of fixed length;
4. Axial variable (Av) - insert variable length axial components;
5. Radial (Rd) - insert radial components;
6. Surface Mount (Sm) - apply welding.

The characteristics of the shop environment are as follows:

1. A PCB passes successively through several processes, depending on the type of the PCB;
2. In each process, there are groups of machines of different brands, speeds, and capacities. The processing time of each operation depends of the particular machine;
3. Each machine has a *feeder* of components, which allows the entry of a number of electronic components to process a board. The set of the components depends on the type of the board, because each PCB requires a certain number and types of components.

The feeders have different capacities. For example, a machine has a feeder with the capacity of 60 components, another one of 80. The feeder of components at a machine must be prepared before processing the jobs. The number of components arranged in a feeder depends on the type of the board.

The time for the preparation of each feeder at a machine and its corresponding adjustment required by the change of the board type depends essentially on the board that previously passed through that machine. This means that there is a time of adjustment dependent on the sequence.

1. A machine is not enabled when it is in maintenance or because it is not capable of processing a certain type of work. This implies the consideration of the *availability* or *eligibility* of the machines (Machine availability/eligibility);
2. Each machine has a limited physical space for a temporary storage (buffer) of the processed jobs. These jobs in the buffer represent a part of the WIP. If the limit of the buffer capacity is reached in the pace of a machine, it is not able to continue the production, so there would be a *block* in the production. This feature is known as *limited buffer*;
3. It is required to minimize the total time for processing a given set of jobs, called the makespan.

This production environment represents a six-stage HFS with unrelated parallel machines, sequence-dependent setup times, machine eligibility, and limited buffering. An increase of the productivity implies the determination of an order in the workflow that allows a greater use of the machines in such a way that the time of the fulfilment of all works is minimized.

The characteristics and restrictions of the production environment lead to a complex decision-making. This represents a problem for the company because it wants to increase the productivity. The programming of the production was usually done by a group of planners on the base of their experience by manual methods, and these methods did not guarantee good and fast solutions. Therefore, increasing the productivity required an improved decision-making in the scheduling process.

10.2.2 Production model

An HFS scheduling problem is considered. This problem can be characterized as follows: A set of N jobs j, given at time 0, has to be processed without preemption at M consecutive production stages with the objective of minimizing the total completion time. At every stage $i = 1, 2, \ldots, M$, a set $M_i = \{1,2,\ldots,m_i\}$ of m_i unrelated parallel machines is given, where $|M_i| \geq 1$. Every job passes through all stages and must be processed by exactly one machine at every stage. Let $p_{i,l,j}$ be the processing time of job j, $j = 1, 2, \ldots, N$, on machine $l \in M_i$, at the stage i, $i = 1, 2, \ldots, M$. Machine-based sequence-dependent setup times are considered. Let $s_{i,l,j,k}$ be the setup time on machine l at stage i when processing job k after processing job j, where $j, k = 1, 2, \ldots, N, j \neq k$. For the stage i, $E_{i,j}$ is the set of eligible machines that can process job j, $1 \leq |E_{ij}| \leq m_i$. For every machine $m = 1, 2, \ldots, m_i$ at stage i, a limited buffer for the jobs is given. The maximal storage capacity in front of every machine m is b_m, where $1 \leq |b_m| \leq m_i$.

Gourgand, Grangeon and Norre (1999) have shown that for a similar problem the total number of possible solutions is $N! \left(\prod_{i=1}^{M} m_i\right)^N$. Moreover, Gupta and Tunc (1998) have proved that the FFS problem with only two stages ($M = 2$) is *NP*-hard even if one of two stages contains a single machine. Since the HFS is a general case of the FFS, the HFS problem is also *NP*-hard.

Using the well-known three-field notation $\alpha|\beta|\gamma$ for scheduling problems (see Section 3.1) and its extension for the HFS problem proposed by Vignier, Billaut, and Proust (1999, 122), the problem can be denoted as:

$$HFk, \left(Rm^{(i)}\right)_{i=1}^{(m)} | s_{sd}, M_j, block | C_{\max} \ .$$

10.2.3 Algorithm

A genetic algorithm was proposed, considering the characteristics of the model under study. By a genetic algorithm, approximate solutions for problems of great computational complexity were sought by means of a mathematically simulated evolution procedure on a computer (Holland 1975)

The basic genetic algorithm consists of well-defined phases. Candidate solutions are encoded by *chromosomes* (also called genomes or individuals). The set of initial individuals forms the *population*. The *fitness* values are defined for the individuals to measure the quality of the represented solution. The genomes are evolved through genetic operators, generation by generation, to find an approximate solution or even an optimal one. Three genetic operators were repeatedly applied: selection, crossover, and mutation. The *selection* picks the chromosomes to mate and produce offspring. The *crossover* combines two selected chromosomes to generate the next generation of chromosomes. The *mutation* reorganizes the structure of the genes in a chromosome randomly so that a new combination of the genes may appear in the next generation. The individuals evolve until some stopping criterion is met (Fig. 10-9).

Regarding the coding, in several works a chain of integers was used, which is a permutation of the numbers 1,2, ..., *n*, and each integer represents a job.

The proposed algorithm GA_{SBC} is described below.

Input: A population of P_{size} individuals.
Output: An individual of length *n*.

```
01. Generate_population()
02. while not stopping_criterion do
03.         for i = 0 to Psize
04.                 generate_individual(i)
05.                 evaluate_objective(i)
06.         Select individuals by the tournament selection
07.         Keep_the_best_individual_found()
08.         if iterations_no_improvement = 25
09.                 regenerate_population
10.                 if regenerate = 10
11.                     stopping_criterion=true
12.                 else
13.                     regenerate = regenerate+1
14.                 iterations_no_improvement = 0
15.         else
```

```
16.                 if actual_minimum_makespan =
                    previous_minimum_makespan
17.                 iterations_no_improvement
                        =iterations_no_improvement +1
18.         crossover with probability Pc
19.         mutation with probability Pm.
```

The population is formed by P_{size} individuals. According to the algorithm, the *generate_population* function generates a set of individuals with their genes in a random and uniform manner. The function *evaluate_objective* calculates the fitness (makespan) of each individual taking into account the size of the buffer at each machine. When the buffer of a machine becomes saturated, the flow continues through another machine of the same stage, considering the same selection criterion: Each job is assigned to the machine that can finish the job in the shortest possible time at each stage, considering the different speeds, setup times, machine availability, and buffer sizes.

The *keep_the_best_individual_found* function saves the best individual found in the previous evaluation of the population in each iteration. The final result will be the best individual found for the objective function.

A reset mechanism is applied when, after ordering by aptitudes, the best 20% of the individuals are maintained, 50% of the rest (80%) are replaced by the *shift* mutation of the best individual. The rest is generated again in a random way. The evaluations are stopped when the minimum time to complete the works has not changed within 25 generations and after 10 restarts have been executed. This criterion allows both good solutions and reasonable execution times.

A selection by binary tournament was considered as well as eight types of crossovers, five of them were taken from the literature:

OBX – *Order-Based Crossover* (Gen and Cheng 1999);
PPX - *Precedence Preservative Crossover* (Bierwirth, Mattfeld, and Kopfer 1996);
OSX - *One Segment Crossover* (Gen and Cheng 1999);
TP - *Two Point* (Michalewicz 1996);
SB2OX - Similar Block 2-Point Order Crossover (Ruiz and Maroto 2006, 788–91).

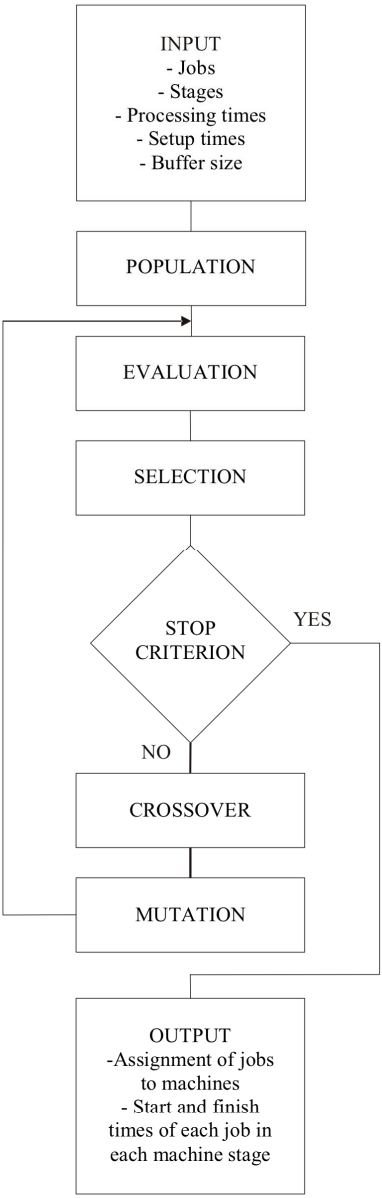


Fig. 10-9. Chart of algorithm GA_{SBC}.

Three further crossovers were proposed:

TPI (*Two Point Inverse*);
OBSTX (*Order-Based Setup Time Crossover*);
ST2PX (*Setup Time Two Point Crossover*).

Three well-known mutation operators, which are frequently used in the literature, were also considered (*insert*, *swap*, and *switch*) (Michalewicz 1996; Gen and Cheng 1999).

A calibration process was carried out, applying 1800 alternative algorithms to a set of 60 instances, in total 108,000 evaluations, taking into account the following factors: type of crossover, type of mutation, probability of a crossover, probability of a mutation, and size of the population. The response variable was the percentage of the relative distance increase over the best known solution. The results of the experiments were subjected to a multivariate variance analysis to test the null hypothesis: there is no significant effect of the factors on the response variable. The contribution of each factor was measured removing the effects of the other factors. The results of this analysis have shown that five of the considered factors (population, crossover operator, mutation operator, crossover probability, mutation probability) had a significant effect on the response variable at a 95% confidence level, for which the null hypothesis was rejected, with a significant effect of the factors on the response variable.

10.2.4 Computer experiment

To evaluate the quality of the calibrated algorithm, two experiments were performed. The first experiment was conducted to analyze the behavior of the algorithm under limited time circumstances. The results showed the superiority of the proposed algorithm.

As the reference algorithm, a genetic algorithm GA_H was used in the second experiment. This algorithm was proposed by Ruiz and Maroto (2006, 788–91) for solving a similar problem without limited buffers. It was compared with nine variants of metaheuristic methods: NEH heuristic, simulated annealing, tabu search, ant-based algorithms, etc. It turned out to be best algorithm for solving similar problems without limited buffers. To compare algorithm GA_{SBC} with GA_H, an adaptation of the last one to limited buffers was made. This version was denoted as GA_{HBC}. The mean relative distance of the GA_{SBC} and GA_{HBC} from the best results are given in Table 10-4. GA_{SBC} improved the results of GA_{HBC} in all experiments.

Algorithm GA_{SBC} had an efficient performance due to the proposed

crossover operator (ST2PX). Based on a multivariate analysis of variance, it was shown that this crossover was the best among the eight ones considered (Yaurima, Burtseva, and Tchernykh 2009). Its success can be explained by the fact that this crossover operator takes into account the adjustment times of the machines according to the sequence of the jobs. The adjustment time in this model is an important circumstance when making decisions regarding the assignment of jobs to the unrelated parallel machines. The restarting conditions within the suggested scheme, together with the stopping criterion, were other innovations that were successfully carried out in this algorithm.

50 Jobs instances			**100 Jobs instances**		
Instance	GA_{SBC}	GA_{HBC}	***Instance***	GA_{SBC}	GA_{HBC}
50 x 2	1.3557	6.4789	100 x 2	2.7658	5.5054
50 x 3	1.8877	5.7557	100 x 3	2.8016	6.8553
50 x 6	2.4963	6.5712	100 x 6	1.8996	4.8614
Average	1.9132	6.2686	Average	2.489	5.7407

Table 10-4. Average percentages of the relative distance from the best known solutions.

10.2.5 Conclusions

An efficient genetic algorithm was presented to solve a complex HFS problem with sequence-dependent setup times, unrelated parallel machines, machine availability constraints, and limited buffers. This problem occurs in the television electronics industry, in particular, in the line of auto insertion of components into PCBs. It can be concluded that genetic algorithms are a useful tool in solving real and complex problems such as the HFS.

10.3 Optimization of the material use in the reed switch production

A planning problem that occurs in the production of reed switches was examined in this project. Reed switches are used as parts in the

manufacturing of electronic sensors and relays. The company suffers from the uncertainty in the choice of the raw material, which comes in lots of four types, to fulfill the customer orders without violating the respective deadlines. An incorrect selection causes the processing of additional lots and may provoke delays in the fulfilment of the orders. This also generates an excessive WIP. It was required to improve the production efficiency giving the planners a tool to make the decisions for the selection of the lot types.

Similar situations occur in the manufacturing of semiconductors and digital devices as well as in the ceramic, food and agricultural industries, when the items of a production lot are classified according to quality, color, size, etc., to be used for different purposes.

The lot type selection represents a complex combinatorial problem. The proposed model can be interpreted as an extension of the bin packing problem. A heuristic was proposed to calculate the minimal number of lots to fulfill a given set of customer orders selecting the correct raw material and the sequence of processing the lots. This heuristic represents a modification of the north-west corner rule. The investigation was made for COTO Technology, Inc., Mexicali, Mexico.

10.3.1 Problem description

The components of a reed switch are two metallic contact blades (reeds) positioned into a hermetically sealed glass capsule (tube). There is a gap between the blades in the contact area (Fig. 10-10). The magnetic field expressed in NI or ampere-turns (AT) is known as the *operating value*. When applied to the switch, the operating value forces this to actuate closing the blades.

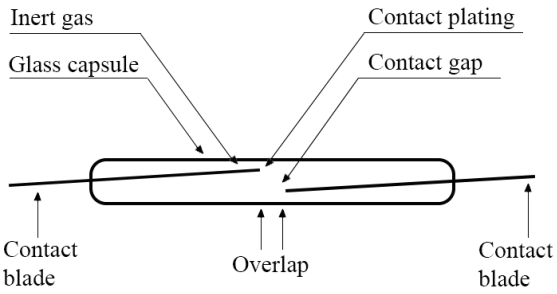


Fig. 10-10. The components of a reed switch.

The glass tubes used distinguish in-between by the core diameter. Each lot of the raw material contains 5500 glass tubes with a fixed range of the core diameters, which is referred to as the *glass type* in the plant. In total, there are *four glass types*, i.e., four types of glass tubes, which have a specific range of the core diameters so that these ranges do not intersect.

A *customer order* includes the part number (the code of a product), the number of items, the delivery date, and it needs several lots to be completed. The production planning is realized per lots for the planning horizon, which defines the set of consumer orders. The *part number* defines the range of the acceptable *operating value* and one of four possible *shapes of the blades*. There are approximately 50 part numbers, which this plant can manage.

The technical route, which is used for manufacturing the switches, is composed of a number of successive operations divided in a natural manner into two parts by an external operation in-between with the duration of one day. The investigation is focused on the second part, where the presented problem occurs: the classification of the reed switches according to the operating value and the formation of the contact blades. The investigated resource model is presented in Fig. 10-11.

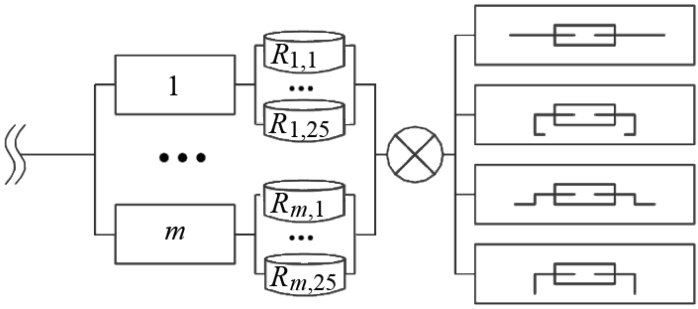


Fig. 10-11.The *m* identical classification machines with 25 repositories every one.

The classification of the reed switches is realized per lots by a group of identical parallel machines, so as one lot is completely processed by one machine without an interruption. The processing time of a lot is constant for any glass type. A classification machine contains 25 successively enumerated repositories to deposit the processed pieces. The classification process is as follows. A machine takes one item, measures its operating value and then deposits it into a corresponding repository.

At the end of this operation, a lot is split into 25 parts/repositories, according to the operating value of the switch. The part number indicates

the acceptable range of the operating value, i.e., the range of the successive repositories, so as the same repository can be used for the fulfillment of the different part numbers. After the classification, when a lot is distributed between the repositories, the product is treated in items.

The items, which are used for carrying out an order, are taken from the corresponding repositories and assigned to one of the four blade shape lines according to the order indications. Then pieces for the next order are taken from the repositories, and so on. This process is similar for the other classification machines. The rest of the pieces, which are not reclaimed for the orders, form the WIP inventory to be used in the future.

Due to the complexity of the information, the production planning and scheduling at the plant are realized empirically, based on the previous experience and therefore, they suffer from the stochastic results after the lot classification.

10.3.2 Modeling

It was observed in the plant that the core diameter of a tube has a direct effect on the operating value of a resulting switch. Therefore, a lot of raw material with a convenient glass type should be used when it is required to obtain the majority of items within a given range of the operating value. The uncertainty in the lot type selection can be reduced if the results of the classification operation are known or evaluated beforehand. Then the number of the lots, which are needed for the fulfillment of a given set of orders with the same due date, can be minimized through finding a best linear combination of the lot types. The classification is a bottleneck operation. It lasts about 8 hs/lot, while the blade shape forming operation is significantly shorter. Therefore, a big effect on the total processing time is also reached by minimizing the number of the lots.

After an analysis of the historical data of the classification results, it was noted that the distribution of the lot items between the repositories has a central tendency that is specific for each glass type, and those empirical distributions tend to be near-normal (Fig. 10-12) (Romero-Parra and Burtseva 2010).

A deterministic approach was used to solve the problem, i.e., all distributions were supposed to be known and fixed for each glass type such that the number of items in every repository can be anticipatorily calculated to be assigned to an order. An illustrative example is given in Fig. 10-13. Assume that the orders $Z1$ and $Z2$ require pieces in the operating ranges (1,2) and (5,6), respectively. The lot type $l1$ is the preferable selection, because it has the maximal quantities of items in these repositories,

compared to the other lot types. For the order *Z*3, the lot type *l*2 is preferable. The lot type *l*3 is convenient for the orders *Z*4 and *Z*5. The lot type *l*4 is the best selection for the order *Z*6. Nevertheless, the selection of the lot types is not so evident.

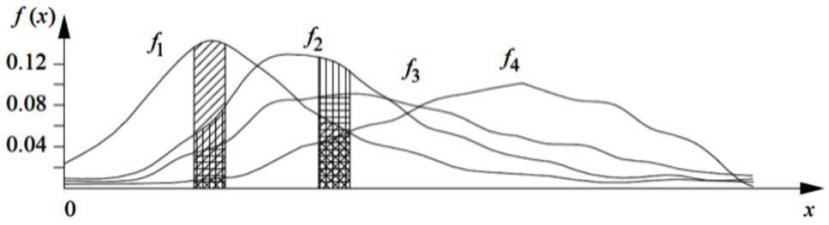


Fig. 10-12. Probability density functions $f(x)$ of the distributions of the items between repositories for different lot types. The corresponding quantities of the items are highlighted for each function (lot type).

The dependency between the characteristics of the raw material and the effect of these characteristics on the final product quantities can be formulated as an optimization problem selecting a linear combination of the lots, everyone with the glass type, which minimizes the total number of the lots to satisfy all orders in the required operating values of the switches.

The problem complexity is conditioned by the following reasons. The number of the lots for the completion of the orders depends essentially on the selection of the lot type as a consequence of the differences of the distribution functions. Moreover, the sets of the repositories, which are used for different part numbers, may overlap. Therefore, the allocation of the orders on the repositories is not evident, for example, orders 1 and 2 need repositories 1 and 2, and order 3 needs repositories 2 and 3 (Fig. 10-14).

Given this situation, the addressed lot minimization problem can be summarized as follows. There are containers of G types with an equivalent capacity U to pack N fractionable objects consisting of D_j items, $j = 1,\ldots, N$. Every container l_g is distributed among R repositories with the capacities of k_{gr} items, forming the matrix $K = [k_{gr}]$, where k_{gr} is the capacity of the repository r, $r = 1,\ldots, R$, for a container of type g, $g = 1,\ldots, G$, and $\sum_{r=1}^{R} k_{gr} = U$, $\forall g$. The items of object D_j can be only deposited into the repositories associated with object j. The restrictions in the allocation of the items of object j on the repositories are given by a binary matrix $O = [o_{jr}]$ and do not depend on the selection of the container type g. The element o_{jr} is equal to 1 if repository r is allowed to pack items of object j, and 0 otherwise. As a result of the allocation of object j to the container of number

l, d_{lrj} items are packed into repository r of container l whose type is g. Then a list of containers $A = \{g_1, g_2,\ldots, g_l,\ldots, g_L\}$ is formed. Its elements are the container types g_l. For example, $A = \{2,4,1,1,4\}$ means that first a container of type 2 is used, then a container of type 4, and the cardinality of A is the required number of containers, $L = |A|$, which is equal to the sum n_g of containers of every type g. In our example, $g_1 = 2$, $g_2 = 1$, $g_3 = 0$, $g_4 = 2$, and $L = 5$.

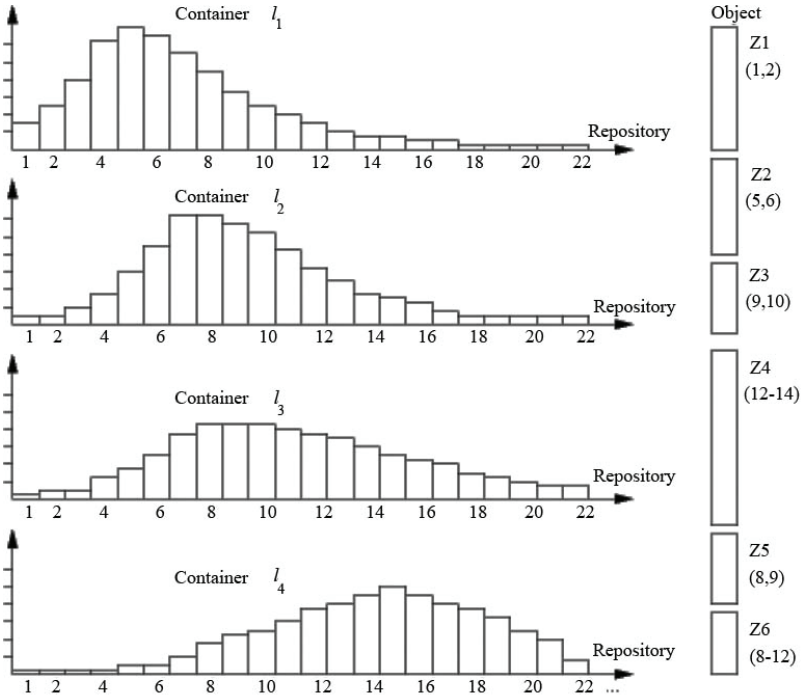


Fig. 10-13. Distribution of the items between the repositories for every lot type.

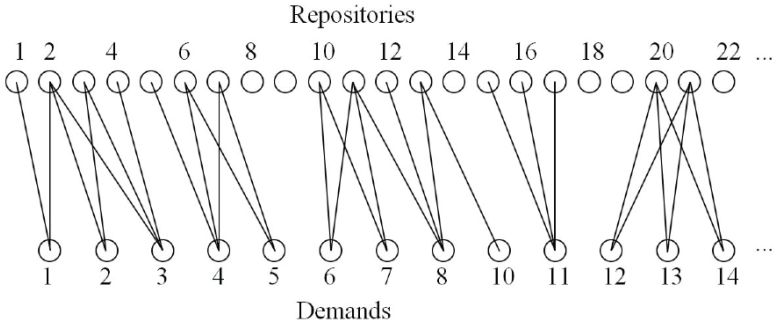


Fig. 10-14. The relationship between the repositories. The orders represented as a bipartite graph with a possible overlapping of the repositories (an example).

Therefore, the problem of minimizing the number of lots is as follows:

$$L = \sum_{g=1}^{G} n_g \to \min \tag{10-2}$$

s.t.

$$\sum_{g=1}^{G} \sum_{l_g=1}^{n_g} \sum_{j=1}^{N} d_{l_g rj}\, o_{rj} \le \sum_{g=1}^{G} n_g k_{gr},\ r = 1,\ldots, R \tag{10-3}$$

$$\sum_{r=1}^{R} \sum_{j=1}^{N} d_{l_g rj} \le U,\ \forall g, l_g, \tag{10-4}$$

$$o_{rj} = \begin{cases} 1, \text{if } r \in V_j \\ 0, \text{otherwise} \end{cases} \quad \forall r, j \tag{10-5}$$

$$\sum_{g=1}^{G} \sum_{l_g=1}^{n_g} \sum_{r=1}^{R} d_{l_g rj}\, o_{rj} = D_j, j = 1,\ldots, N, \tag{10-6}$$

$$k_{gr}, d_{l_g rj} \in Z^0,\ \forall g, l_g, r, j. \tag{10-7}$$

The constraints (10-3) describe the relationship between the assigned items and the total capacity of repository r, where the total sum of the pieces assigned to container l_g must not exceed its capacity U (10-4). The binary values o_{rj} in (10-5) are employed to restrict the object allocation only to allowed repositories. Equalities (10-6) ensure that all objects must be allocated completely. The sizes k_{gr} and $d_{l_g rj}$ are non-negative integers (10-7). The objective is to minimize the number of containers (10-2). This problem is denoted as the *fractionable object bin packing with variable sized repositories and allocation constraints* (*FOBP-VAC*).

Problem (10-2) is an extension of the well-known *bin packing* problem. In the classical NP-hard bin packing problem, a given list of *indivisible objects* is required to be packed into the smallest possible number of unit-sized *indivisible bins*, also called *containers*. The difference from the classical variant is that in the considered case, a container is divided into a

fixed number of R repositories (sub-containers) (Fig. 10-13). The repository capacities in a container vary depending on a parameter g (the glass type of a lot in our problem). These capacities are calculated according to the density distribution function given by row g of matrix K. The sum $k_{g1}+ k_{g2} +\ldots+ k_{gR}$ is the capacity U of a container (lot size), which is constant for any container type g. A fractionable object j is associated with a customer order of D_j items. For each object, the number of items and the repositories that are allowed to be used by this order are indicated.

Similar problems have been presented in the literature. Nevertheless, never before all considered aspects were modeled together. Shachnai and Tamir (2004) formulated a problem called the *class-constrained bin packing* problem, where the bins have a capacity of v and c compartments. In their problem, every item has the same size and a color. The items must be deposited into bins subject to the capacity constraints so that items of different colors are placed in different compartments. The goal was to minimize the number of used bins. Friesen and Langston (1986) defined a variable sized bin packing problem, where some fixed (finite) number of given sizes was available. The cost for using a container was simply its size. The goal of the problem was to pack the items into those containers, whose sum of sizes was minimal. Epstein and Levin (2008) proposed a problem called *Generalized Cost Variable Sized Bin Packing*. There are given an infinite supply of bins of r types whose sizes are denoted by $b_r < ... < b_1 = 1$. Items of sizes in a range (0, 1] are to be partitioned into subsets. A bin type i is associated with a cost c_i. It was assumed that $c_1 = 1$. The goal was to find a feasible solution with minimal total cost. Langston (1984) investigated the problem of maximizing the number of items packed into m available bins, where the bin sizes can be different. Menakerman and Rom (2001) investigated a bin packing problem variant, in which the items may be fragmented into smaller size pieces called fragments. The model was derived from a scheduling problem presented in the data from a CATV network. Xing (2002) introduced a problem called *bin packing with oversized items*, where the items may have a size larger than the largest allowed bin size. The bins cannot be overpacked. An oversized item is free to be divided into parts, which are no larger than a limit of the bin size. Mandal, Chakrabarti, and Ghose (1998) have shown that the decision version of the bin packing problem with N fragmentable objects is NP-hard for $N \geq 2$. Therefore, problem (10-2) is NP-hard as well, and heuristic algorithms should be applied.

10.3.3 Algorithm

Several works were dedicated to study the bin packing problem in offline and online environments. An online algorithm assigns items to the bins in the order in which they are given in the original list, without using any knowledge about subsequent items in the list, see Mandal, Chakrabarti, and Ghose (1998); Ye and Zhang (2009). In an offline version, the entire list of the objects is available to compute the packing. The Next Fit strategy and its variants are a typical solution approach, see Coffman, Garey, and Johnson (1996).

The following heuristic offline algorithm provides a solution for problem (10-2). The algorithm is based on the *North West Corner rule* and finds:

1. The number of containers of the corresponding types $L = n_1 + n_2 + \ldots + n_G$;
2. The number of items d_{lrj} allocated to repository r of container l in order to pack object j for all l, r, j;
3. The container sequence for each type (lot processing sequence).

The steps of the algorithm are as follows.

Input: D[N], K[G,R], O[N,R]
Create a table T with N+1 rows and R+1 columns, where the first N rows are used for the allocation of items of object j, j = 1,…, N , to R repositories. In the cell (j, R+1) of the table, the number D_j of items to allocate is written. The cell (N+1, r) is initially the capacity of the repository r. The element (j, r), j = 1,…, N, r = 1,…, R, of the table is available if o_{jr} = 1. The unavailable cells are blocked. Initially all cells (j, r,), j = 1,…, N, r = 1,…, R, are zero; n_g = 0, g = 1,…, G.

1. Sort the N rows of the table T using a weight rule (more items, fewer items, larger number of available repositories, etc.).
3. Create G copies of the table.
   3.1. In the copy T_g, g = 1,…, G, add k_{gr} to the value of cell r in row N+1, r = 1,..., R.
   3.2. Process the unblocked cells of the table per rows, starting in the upper left corner while the corresponding value in column R+1 is different from zero. The cell values of row N+1 are assigned to the corresponding cells of column r so that the assigned values do not exceed the value in cell (j, R+1). The assigned value is subtracted from the cells (N+1, r) and (j, R+1).

   It continues until the values of all cells in row N+1 are assigned to the available cells or the value of cell (j, R+1) is zero.

4. Calculate the sums in each table T_g, $g = 1,\ldots, G$, as the sum of the values of column $R+1$.
5. Select the table T_{g*}, $i \in \{1,\ldots, G\}$, whose sum is minimal.
   - If the total values in two or more tables are minimal, select the table with minimal total of row $N + 1$.
   - If there is a tie in the two previous rules, select the first table.

   $n_{g*} = n_{g*} + 1$.

   Keep the packing history: the table state T_{g*} and the selected index $g*$.
6. If there exist values different from zero in column $R+1$, go to 2.
7. End

The following example illustrates the execution of the algorithm:

Consider three objects of the sizes 100(1, 2), 75(2, 3), 70(2), where the values in parentheses give the number of allowed repositories to be used. There are 2 types of containers, $g = 1, 2$, with a capacity of 90, and every container has 3 repositories. The sizes of the repositories in a container of type 1 are: 10, 30, and 60, respectively; the repository sizes in a container of type 2 are: 20, 30, and 50. Fig. 10-15 shows the first iteration of the algorithm. After four iterations, the algorithm receives the following result: L =4, which means two containers of type 1 and two containers of type 2 in the sequence {1,2,2,1}; 110 excessive pieces (WIP) are generated. If only containers of type 1 are used, the result is 5 containers and 255 excessive pieces.

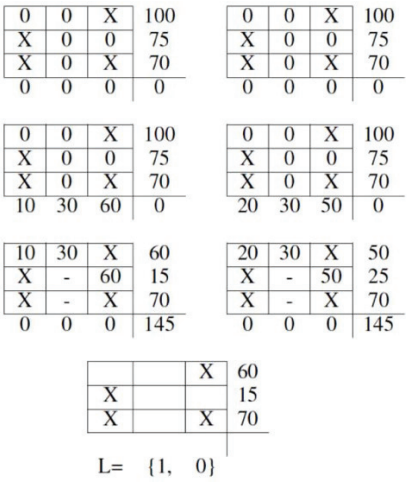


Fig. 10-15. The first iteration of the algorithm selects a container of type 1.

10.3.4 Conclusions

The efficiency of a plant, which is dedicated to the manufacturing of reed switches, is formulated as the problem of minimizing the number of lots of raw material for a given set of customer orders. An extension of the bin packing problem was used for the modeling. A simple and fast heuristic algorithm was proposed to obtain a feasible and practical solution for the plant. It turned out that the density function distributions do not affect the presented algorithm.

To compensate the variation of the classification results with respect to the performance of the real process, it is further necessary to: update the samples of the company; investigate the behavior of the scrap; maintain a certain level of the WIP. These topics are subjects of future research**.**

REFERENCES

Abad, P. L. 2003. "Optimal Pricing and Lot-Sizing under Conditions of Perishability, Finite Production and Partial Backordering and Lost Sale." *European Journal of Operational Research* 144 (3): 677–85. https://doi.org/10.1016/S0377-2217(02)00159-5.

Abou-Zeid, M. R. 1975. "Group Technology." *Ind Eng* 7 (May): 32–39.

Achugbue, J. O., and F. Y. Chin. 1982. "Scheduling the Open Shop to Minimize Mean Flow Time." *SIAM Journal on Computing* 11 (4): 709–20. https://doi.org/10.1137/0211058.

Adams, J., E. Balas, and D. Zawack. 1988. "The Shifting Bottleneck Procedure for Job Shop Scheduling." *Management Science* 34 (3): 391–401.

Afentakis, P., B. Gavish, and U. Karmarkar. 1984. "Computationally Efficient Optimal Solutions to the Lot-Sizing Problem in Multistage Assembly Systems." *Management Science* 30 (2): 222–39.

Aggarwal, A., and J. K. Park. 1993. "Improved Algorithms for Economic Lot Size Problems." *Operations Research* 41 (3): 549–71. https://doi.org/10.1287/opre.41.3.549.

Aghezzaf, E.-H., and H. V. Landeghem. 2002. "An Integrated Model for Inventory and Production Planning in a Two-Stage Hybrid Production System." *International Journal of Production Research* 40 (17): 4323–39. https://doi.org/10.1080/00207540210159617.

Agnetis, A., A. Alfieri, and G. Nicosia. 2003. "Part Batching and Scheduling in a Flexible Cell to Minimize Setup Costs." *Journal of Scheduling* 6 (1): 87–108. https://doi.org/10.1023/A:1022239620866.

Agnetis, A., M. Lucertini, and F. Nicolo. 1993. "Flow Management in Flexible Manufacturing Cells with Pipeline Operations." *Management Science* 39 (3): 294–306. https://doi.org/10.1287/mnsc.39.3.294.

Aksen, D. 2007. "Loss of Customer Goodwill in the Uncapacitated Lot-Sizing Problem." *Computers & Operations Research* 34 (9): 2805–23. https://doi.org/10.1016/j.cor.2005.10.012.

Alidaee, B., and N. K. Womer. 1999. "Scheduling with Time Dependent Processing Times: Review and Extensions." *The Journal of the Operational Research Society* 50 (7): 711–20. https://doi.org/10.2307/3010325.

Allahverdi, A. 2000. "Minimizing Mean Flowtime in a Two-Machine Flowshop with Sequence-Independent Setup Times." *Computers & Operations Research* 27 (2): 111–27. https://doi.org/10.1016/S0305-0548(99)00010-6.

—. 2015. "The Third Comprehensive Survey on Scheduling Problems with Setup Times/Costs." *European Journal of Operational Research* 246 (2): 345–78. https://doi.org/10.1016/j.ejor.2015.04.004.

Allahverdi, A., H. Aydilek, and A. Aydilek. 2016. "Two-Stage Assembly Scheduling Problem for Minimizing Total Tardiness with Setup Times." *Applied Mathematical Modelling* 40 (17): 7796–7815. https://doi.org/10.1016/j.apm.2016.03.037.

Allahverdi, A., J. N. D Gupta, and T. Aldowaisan. 1999. "A Review of Scheduling Research Involving Setup Considerations." *Omega* 27 (2): 219–39. https://doi.org/10.1016/S0305-0483(98)00042-5.

Allahverdi, A., C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov. 2008. "A Survey of Scheduling Problems with Setup Times or Costs." *European Journal of Operational Research* 187 (3): 985–1032. https://doi.org/10.1016/j.ejor.2006.06.060.

Anand, E., and R. Panneerselvam. 2015. "Literature Review of Open Shop Scheduling Problems." *Intelligent Information Management* 7 (1): 33–52. https://doi.org/10.4236/iim.2015.71004.

Andersson, H., A. Hoff, M. Christiansen, G. Hasle, and A. Løkketangen. 2010. "Industrial Aspects and Literature Survey: Combined Inventory Management and Routing." *Computers & Operations Research* 37 (9): 1515–36. https://doi.org/10.1016/j.cor.2009.11.009.

Andrés, C., J. M. Albarracín, G. Tormo, E. Vicens, and J. P. García-Sabater. 2005. "Group Technology in a Hybrid Flowshop Environment: A Case Study." *European Journal of Operational Research* 167 (1): 272–81. https://doi.org/10.1016/j.ejor.2004.03.026.

Anthony, R. N. 1965. *Planning and Control Systems; a Framework for Analysis*. Boston: Division of Research, Graduate School of Business Administration, Harvard University.

Anupindi, R., S. Chopra, and S. D. Deshmukh. 2012. *Managing Business Process Flows: Principles of Operations Management*. Prentice Hall.

Applegate, D., and W. Cook. 1991. "A Computational Study of the Job-Shop Scheduling Problem." *ORSA Journal on Computing* 3 (2): 149–56. https://doi.org/10.1287/ijoc.3.2.149.

Armentano, V. A., Paulo M. França, and F. M. B. de Toledo. 1999. "A Network Flow Model for the Capacitated Lot-Sizing Problem." *Omega* 27 (2): 275–84. https://doi.org/10.1016/S0305-0483(98)00045-0.

Askin, R. G., and C. R. Standridge. 1993. "Modeling and Analysis of Manufacturing Systems | Wiley." Wiley.Com. 1993. https://www.wiley.com/en-mx/Modeling+and+Analysis+of+Manufacturing+Systems-p-9780471514183.

Aytug, H., M. A. Lawley, K. McKay, S. Mohan, and R. Uzsoy. 2005. "Executing Production Schedules in the Face of Uncertainties: A Review and Some Future Directions." *European Journal of Operational Research*, IEPM: Focus on Scheduling, 161 (1): 86–110. https://doi.org/10.1016/j.ejor.2003.08.027.

Azzi, A., M. Faccio, A. Persona, and F. Sgarbossa. 2012. "Lot Splitting Scheduling Procedure for Makespan Reduction and Machine Capacity Increase in a Hybrid Flow Shop with Batch Production." *The International Journal of Advanced Manufacturing Technology* 59 (5): 775–86. https://doi.org/10.1007/s00170-011-3525-x.

Bachman, A., and A. Janiak. 2004. "Scheduling Jobs with Position-Dependent Processing Times." *Journal of the Operational Research Society* 55 (3): 257–64. https://doi.org/10.1057/palgrave.jors.2601689.

Bai, D., Z.-H. Zhang, and Q. Zhang. 2016. "Flexible Open Shop Scheduling Problem to Minimize Makespan." *Computers & Operations Research* 67 (March): 207–15. https://doi.org/10.1016/j.cor.2015.10.012.

Bai, S. X., and S. B. Gershwin. 1990. "Scheduling Manufacturing Systems with Work-in-Process Inventory." In *29th IEEE Conference on Decision and Control*, 557–64 vol.2. https://doi.org/10.1109/CDC.1990.203658.

Baker, K. R., and D. Jia. 1993. "A Comparative Study of Lot Streaming Procedures." *Omega* 21 (5): 561–66. https://doi.org/10.1016/0305-0483(93)90024-F.

Baker, K. R., and D. F. Pyke. 1990. "Solution Procedures for the Lot-Streaming Problem." *Decision Sciences* 21 (3): 475–91. https://doi.org/10.1111/j.1540-5915.1990.tb00328.x.

Ballakur, A., and H. J. Steudel. 1987. "A Within-Cell Utilization Based Heuristic for Designing Cellular Manufacturing Systems." *International Journal of Production Research* 25 (5): 639–65. https://doi.org/10.1080/00207548708919868.

Bang, J.-Y., and Y.-D. Kim. 2011. "Scheduling Algorithms for a Semiconductor Probing Facility." *Computers & Operations Research* 38 (3): 666–73. https://doi.org/10.1016/j.cor.2010.08.010.

Bard, J. F., Z. Gao, R. Chacon, and J. Stuber. 2013. "Daily Scheduling of Multi-Pass Lots at Assembly and Test Facilities." *International Journal of Production Research* 51 (23–24): 7047–70. https://doi.org/10.1080/00207543.2012.758393.

Bellanger, A., and A. Oulamara. 2009. “Scheduling Hybrid Flowshop with Parallel Batching Machines and Compatibilities.” *Computers & Operations Research* 36 (6): 1982–92. https://doi.org/10.1016/j.cor.2008.06.011.

Bellanger, A., A. Oulamara, and M. Y. Kovalyov. 2010. “Minimizing Total Completion Time on a Batching Machine with Job Processing Time Compatibilities.” *Electronic Notes in Discrete Mathematics*, ISCO 2010 - International Symposium on Combinatorial Optimization, 36 (August): 1295–1302. https://doi.org/10.1016/j.endm.2010.05.164.

Belvaux, G., and L. A. Wolsey. 2000. “Bc — Prod: A Specialized Branch-and-Cut System for Lot-Sizing Problems.” *Management Science* 46 (5): 724–38. https://doi.org/10.1287/mnsc.46.5.724.12048.

Ben-Dati, L., G. Mosheiov, and D. Oron. 2009. “Batch Scheduling on Two-Machine Flowshop with Machine-Dependent Setup Times.” Research Article. Advances in Operations Research. Hindawi. August 16, 2009. https://doi.org/10.1155/2009/153910.

Ben-Daya, M., M. Darwish, and K. Ertogral. 2008. “The Joint Economic Lot Sizing Problem: Review and Extensions.” *European Journal of Operational Research* 185 (2): 726–42. https://doi.org/10.1016/j.ejor.2006.12.026.

Benhabib, B. 2003. *Manufacturing: Design, Production, Automation, and Integration*. CRC Press.

Bhat, M. V., and A. Haupt. 1976. “An Efficient Clustering Algorithm.” *IEEE Transactions on Systems, Man, and Cybernetics* SMC-6 (1): 61–64. https://doi.org/10.1109/TSMC.1976.5408399.

Bierwirth, C., D. C. Mattfeld, and H. Kopfer. 1996. “On Permutation Representations for Scheduling Problems.” In *Parallel Problem Solving from Nature — PPSN IV*, edited by H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, 310–18. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-61723-X_995.

Biskup, D. 1999. “Single-Machine Scheduling with Learning Considerations.” *European Journal of Operational Research* 115 (1): 173–78. https://doi.org/10.1016/S0377-2217(98)00246-X.

—. 2008. “A State-of-the-Art Review on Scheduling with Learning Effects.” *European Journal of Operational Research* 188 (2): 315–29. https://doi.org/10.1016/j.ejor.2007.05.040.

Biskup, D., and M. Feldmann. 2007. “Lot Streaming in a Multiple Product Permutation Flow Shop with Intermingling.” *International Journal of Production Research* 46 (01): 197–216. https://doi.org/10.1080/00207540600930065.

Bitran, G. R., and H. H. Yanasse. 1982. "Computational Complexity of the Capacitated Lot Size Problem." *Management Science* 28 (10): 1174–86.

Błażewicz, J., K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. 2007. *Handbook on Scheduling: From Theory to Applications*. International Handbooks on Information Systems. Berlin Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-540-32220-7.

Bogaschewsky, R. W., U. D. Buscher, and G. Lindner. 2001. "Optimizing Multi-Stage Production with Constant Lot Size and Varying Number of Unequal Sized Batches." *Omega* 29 (2): 183–91. https://doi.org/10.1016/S0305-0483(00)00040-2.

Bomberger, E. E. 1966. "A Dynamic Programming Approach to a Lot Size Scheduling Problem." *Management Science* 12 (11): 778–84. https://doi.org/10.1287/mnsc.12.11.778.

Bonfill, A., A. Espuña, and L. Puigjaner. 2008. "Proactive Approach to Address the Uncertainty in Short-Term Scheduling." *Computers & Chemical Engineering* 32 (8): 1689–1706. https://doi.org/10.1016/j.compchemeng.2007.08.014.

Bonney, M. C., Z. Zhang, M. A Head, C. C Tien, and R. J. Barson. 1999. "Are Push and Pull Systems Really so Different?" *International Journal of Production Economics* 59 (1): 53–64. https://doi.org/10.1016/S0925-5273(98)00094-2.

Boucher, T. O. 1984. "Lot Sizing in Group Technology Production Systems." *International Journal of Production Research* 22 (1): 85–93. https://doi.org/10.1080/00207548408942432.

Boyle, C. 1986. "The Benefits of Group Technology Software for Manufacturing." In *Proceedings of the IMTS Conference*, 1042. Chicago, ll1inois.

Bożek, A., and F. Werner. 2018. "Flexible Job Shop Scheduling with Lot Streaming and Sublot Size Optimisation." *International Journal of Production Research* 56 (19): 6391–6411. https://doi.org/10.1080/00207543.2017.1346322.

Brahimi, N., S. Dauzere-Peres, N. M. Najid, and A. Nordli. 2006. "Single Item Lot Sizing Problems." *European Journal of Operational Research* 168 (1): 1–16. https://doi.org/10.1016/j.ejor.2004.01.054.

Browne, S., and U. Yechiali. 1990. "Scheduling Deteriorating Jobs on a Single Processor." *Operations Research* 38 (3): 495–98. https://doi.org/10.1287/opre.38.3.495.

Brucker, P., A. Drexl, R. Möhring, K. Neumann, and E. Pesch. 1999. "Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods." *European Journal of Operational Research* 112 (1): 3–41. https://doi.org/10.1016/S0377-2217(98)00204-5.

Brucker, P., A. Gladky, H. Hoogeveen, M. Y. Kovalyov, C. N. Potts, T. Tautenhahn, and S. L. van de Velde. 1998. “Scheduling a Batching Machine.” *Journal of Scheduling* 1 (1): 31–54. https://doi.org/10.1002/(SICI)1099-1425(199806)1:1<31::AID-JOS4>3.0.CO;2-R.

Brüggemann, W., and H. Jahnke. 2000. “The Discrete Lot-Sizing and Scheduling Problem: Complexity and Modification for Batch Availability.” *European Journal of Operational Research* 124 (3): 511–28.

Burbidge, J. L. 1963. “Production Flow Analysis.” *Production Engineer* 42 (12): 742–52. https://doi.org/10.1049/tpe.1963.0114.

—. 1989. *Production Flow Analysis for Planning Group Technology*. Oxford : New York.

—. 1991. “Production Flow Analysis for Planning Group Technology.” *Journal of Operations Management* 10 (1): 5–27. https://doi.org/10.1016/0272-6963(91)90033-T.

—. 1996. “The First Step in Planning Group Technology.” *International Journal of Production Economics* 43 (2): 261–66. https://doi.org/10.1016/0925-5273(96)00027-8.

Carmon, T. F., O. Z. Maimon, and E. M. Dar-El. 1989. “Group Set-up for Printed Circuit Board Assembly.” *International Journal of Production Research* 27 (10): 1795–1810. https://doi.org/10.1080/00207548908942655.

Carrie, A. S. 1973. “Numerical Taxonomy Applied to Group Technology and Plant Layout.” *International Journal of Production Research* 11 (4): 399–416. https://doi.org/10.1080/00207547308929988.

Caserta, M., and E. Q. Rico. 2009. “A Cross Entropy-Lagrangean Hybrid Algorithm for the Multi-Item Capacitated Lot-Sizing Problem with Setup Times.” *Computers & Operations Research*, Scheduling for Modern Manufacturing, Logistics, and Supply Chains, 36 (2): 530–48. https://doi.org/10.1016/j.cor.2007.10.014.

Cedeño, A. A., and G. A. Süer. 1997. “The Use of a Similarity Coefficient-Based Method to Perform Clustering Analysis to a Large Set of Data with Dissimilar Parts.” *Computers & Industrial Engineering*, Proceedings of the 21st International Conference on Computers and Industrial Engineering, 33 (1): 225–28. https://doi.org/10.1016/S0360-8352(97)00080-6.

Çetinkaya, F. C. 1994. “Lot Streaming in a Two-Stage Flow Shop with Set-up, Processing and Removal Times Separated.” *The Journal of the Operational Research Society* 45 (12): 1445–55. https://doi.org/10.2307/2583938.

Çetinkaya, F. C., and M. Duman. 2010. "Lot Streaming in a Two-Machine Mixed Shop." *The International Journal of Advanced Manufacturing Technology* 49 (9): 1161–73. https://doi.org/10.1007/s00170-009-2473-1.

Chakraborty, T., B. C. Giri, and K. S. Chaudhuri. 2009. "Production Lot Sizing with Process Deterioration and Machine Breakdown under Inspection Schedule." *Omega* 37 (2): 257–71. https://doi.org/10.1016/j.omega.2006.12.001.

Chakravarthy, K., and C. Rajendran. 1999. "A Heuristic for Scheduling in a Flowshop with the Bicriteria of Makespan and Maximum Tardiness Minimization." *Production Planning & Control* 10 (7): 707–14. https://doi.org/10.1080/095372899232777.

Chamnanlor, C., K. Sethanan, C.-F. Chien, and M. Gen. 2013. "Hybrid Genetic Algorithms for Solving Reentrant Flow-Shop Scheduling with Time Windows." *Industrial Engineering and Management Systems* 12 (4): 306–16. https://doi.org/10.7232/iems.2013.12.4.306.

Chan, H. M., and D. A. Milner. 1982. "Direct Clustering Algorithm for Group Formation in Cellular Manufacture." *Journal of Manufacturing Systems* 1 (1): 65–75. https://doi.org/10.1016/S0278-6125(82)80068-X.

Chandrasekharan, M. P., and R. Rajagopalan. 1986. "An Ideal Seed Non-Hierarchical Clustering Algorithm for Cellular Manufacturing." *International Journal of Production Research* 24 (2): 451–63. https://doi.org/10.1080/00207548608919741.

Chandru, V., C.-Y. Lee, and R. Uzsoy. 1993. "Minimizing Total Completion Time on a Batch Processing Machine with Job Families." *Operations Research Letters* 13 (2): 61–65. https://doi.org/10.1016/0167-6377(93)90030-K.

Chang, J. H., and H. N. Chiu. 2005. "A Comprehensive Review of Lot Streaming." *International Journal of Production Research* 43 (8): 1515–36. https://doi.org/10.1080/00207540412331325396.

Chang, S. S., and H. K. Young. 2003. "Minimizing Due Date Related Performance Measures on Two Batch Processing Machines." *European Journal of Operational Research* 147 (3): 644–56. https://doi.org/10.1016/S0377-2217(02)00352-1.

Che, A., M. Chabrol, M. Gourgand, and Y. Wang. 2012. "Scheduling Multiple Robots in a No-Wait Re-Entrant Robotic Flowshop." *International Journal of Production Economics*, Advances in Optimization and Design of Supply Chains, 135 (1): 199–208. https://doi.org/10.1016/j.ijpe.2011.07.008.

Chen, B., J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin. 2017. "Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges." *IEEE Access* 6: 6505–19. https://doi.org/10.1109/ACCESS.2017.2783682.

Chen, B., Y. Ye, and J. Zhang. 2006. "Lot-Sizing Scheduling with Batch Setup Times." *Journal of Scheduling* 9 (3): 299–310. https://doi.org/10.1007/s10951-006-8265-7.

Chen, C., T. Gärling, and R. Kitamura. 2004. "Activity Rescheduling: Reasoned or Habitual?" *Transportation Research Part F: Traffic Psychology and Behaviour* 7 (6): 351–71. https://doi.org/10.1016/j.trf.2004.10.003.

Chen, C.-l., C.-l. Liu, X.-d. Feng, and H.-h. Shao. 2002. "Optimal Short-Term Scheduling of Multiproduct Single-Stage Batch Plants with Parallel Lines." *Industrial & Engineering Chemistry Research* 41 (5): 1249–60. https://doi.org/10.1021/ie010465d.

Chen, J.-F. 2009. "Scheduling on Unrelated Parallel Machines with Sequence- and Machine-Dependent Setup Times and Due-Date Constraints." *The International Journal of Advanced Manufacturing Technology* 44 (11): 1204–12. https://doi.org/10.1007/s00170-008-1917-3.

—. 2015. "Unrelated Parallel-Machine Scheduling to Minimize Total Weighted Completion Time." *Journal of Intelligent Manufacturing* 26 (6): 1099–1112. https://doi.org/10.1007/s10845-013-0842-y.

Chen, J.-S., J. C.-H. Pan, and C.-M. Lin. 2008. "A Hybrid Genetic Algorithm for the Re-Entrant Flow-Shop Scheduling Problem." *Expert Systems with Applications* 34 (1): 570–77. https://doi.org/10.1016/j.eswa.2006.09.021.

Chen, L., and E. L. Plambeck. 2008. "Dynamic Inventory Management with Learning About the Demand Distribution and Substitution Probability." *Manufacturing & Service Operations Management* 10 (2): 236–56. https://doi.org/10.1287/msom.1070.0165.

Chen, T., L. Long, and R. Y. K. Fung. 2006. "A Genetic Algorithm for the Batch Scheduling with Sequence-Dependent Setup Times." In *Intelligent Computing in Signal Processing and Pattern Recognition: International Conference on Intelligent Computing, ICIC 2006 Kunming, China, August 16–19, 2006*, edited by D.-S. Huang, K. Li, and G. W. Irwin, 1137–44. Lecture Notes in Control and Information Sciences. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-37258-5_147.

Chen, T., Y.-H Qin, J.-N Wan, D.-J Wang, and B. Liu. 2015. "Re-Entrant Flexible Scheduling: Models, Algorithms and Applications." *Xitong*

Gongcheng Lilun Yu Shijian/System Engineering Theory and Practice 35 (May): 1187–1201.

Chen, W.-J. 2008. "Single-Machine Scheduling with Family Setup Times in a Manufacturing System." *Engineering Optimization* 40 (6): 579–89. https://doi.org/10.1080/03052150801888540.

—. 2009. "Minimizing Number of Tardy Jobs on a Single Machine Subject to Periodic Maintenance." *Omega* 37 (3): 591–99. https://doi.org/10.1016/j.omega.2008.01.001.

Chen, Z.-L., and W. B. Powell. 2003. "Exact Algorithms for Scheduling Multiple Families of Jobs on Parallel Machines." *Naval Research Logistics (NRL)* 50 (7): 823–40. https://doi.org/10.1002/nav.10091.

Cheng, C.-H., A. Kumar, and J. Motwani. 1995. "A Comparative Examination of Selected Cellular Manufacturing Clustering Algorithms." *International Journal of Operations & Production Management* 15 (12): 86–97. https://doi.org/10.1108/01443579510104538.

Cheng, H. M., and K. C. Ying. 2011. "Minimizing Makespan in a Flow-Line Manufacturing Cell with Sequence Dependent Family Setup Times." *Expert Systems with Applications* 38 (12): 15517–22. https://doi.org/10.1016/j.eswa.2011.06.008.

Cheng, M., N. J. Mukherjee, and S. C. Sarin. 2013. "A Review of Lot Streaming." *International Journal of Production Research* 51 (23–24): 7023–46. https://doi.org/10.1080/00207543.2013.774506.

Cheng, M., and S. C. Sarin. 2013. "Two-Stage, Multiple-Lot, Lot Streaming Problem for a 1 + 2 Hybrid Flow Shop." *IFAC Proceedings Volumes*, 7th IFAC Conference on Manufacturing Modelling, Management, and Control, 46 (9): 448–53. https://doi.org/10.3182/20130619-3-RU-3018.00310.

Cheng, M., S. C. Sarin, and S. Singh. 2016. "Two-Stage, Single-Lot, Lot Streaming Problem for a 1+2 Hybrid Flow Shop." *Journal of Global Optimization* 66 (2): 263–90. https://doi.org/10.1007/s10898-015-0298-z.

Cheng, R., M. Gen, and Y. Tsujimura. 1996. "A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms—I. Representation." *Computers & Industrial Engineering* 30 (4): 983–97. https://doi.org/10.1016/0360-8352(96)00047-2.

Cheng, T. C. E., Q. Ding, M. Y. Kovalyov, A. Bachman, and A. Janiak. 2003. "Scheduling Jobs with Piecewise Linear Decreasing Processing Times." *Naval Research Logistics (NRL)* 50 (6): 531–54. https://doi.org/10.1002/nav.10073.

Cheng, T. C. E., Q Ding, and B. M. T Lin. 2004. "A Concise Survey of Scheduling with Time-Dependent Processing Times." *European Journal of Operational Research* 152 (1): 1–13. https://doi.org/10.1016/S0377-2217(02)00909-8.

Cheng, T. C. E., J. N. D. Gupta, and G. Wang. 2000. "A Review of Flowshop Scheduling Research with Setup Times." *Production and Operations Management* 9 (3): 262–82. https://doi.org/10.1111/j.1937-5956.2000.tb00137.x.

Cheng, T. C. E., A. Janiak, and M. Y. Kovalyov. 2001. "Single Machine Batch Scheduling with Resource Dependent Setup and Processing Times." *European Journal of Operational Research* 135 (1): 177–83. https://doi.org/10.1016/S0377-2217(00)00312-X.

Cheng, T. C. E., and M. Y. Kovalyov. 1995. "Single Machine Batch Scheduling with Deadlines and Resource Dependent Processing Times." *Operations Research Letters* 17 (5): 243–49. https://doi.org/10.1016/0167-6377(95)00011-8.

Cheng, T. C. E., C. T. Ng, and J. J. Yuan. 2003a. "The Single Machine Batching Problem with Family Setup Times to Minimize Maximum Lateness Is Strongly NP-Hard." *Journal of Scheduling* 6 (5): 483–90. https://doi.org/10.1023/A:1024858623282.

—. 2003b. "A Stronger Complexity Result for the Single Machine Multi-Operation Jobs Scheduling Problem to Minimize the Number of Tardy Jobs." *Journal of Scheduling* 6 (6): 551–55. https://doi.org/10.1023/A:1026276627133.

Cheng, T. C. E., and C. C. S. Sin. 1990. "A State-of-the-Art Review of Parallel-Machine Scheduling Research." *European Journal of Operational Research* 47 (3): 271–92. https://doi.org/10.1016/0377-2217(90)90215-W.

Cheng, T. C. E., and G. Wang. 2000. "Single Machine Scheduling with Learning Effect Considerations." *Annals of Operations Research* 98 (1): 273–90. https://doi.org/10.1023/A:1019216726076.

Cheng, T. C. E., C.-C. Wu, and W.-C. Lee. 2008. "Some Scheduling Problems with Deteriorating Jobs and Learning Effects." *Computers & Industrial Engineering* 54 (4): 972–82. https://doi.org/10.1016/j.cie.2007.11.006.

Chien, C.-F., C.-H. Wu, and Y.-S. Chiang. 2012. "Coordinated Capacity Migration and Expansion Planning for Semiconductor Manufacturing under Demand Uncertainties." *International Journal of Production Economics*, Green Manufacturing and Distribution in the Fashion and Apparel Industries, 135 (2): 860–69. https://doi.org/10.1016/j.ijpe.2011.10.024.

Choi, I.-C., and O. Korkmaz. 1997. "Job Shop Scheduling with Separable Sequence-Dependent Setups." *Annals of Operations Research* 70 (0): 155–70. https://doi.org/10.1023/A:1018918003761.

Choi, Y.-C. 2016. "Dispatching Rule-Based Scheduling Algorithms in a Single Machine with Sequence-Dependent Setup Times and Energy Requirements." *Procedia CIRP*, Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems, 41 (January): 135–40. https://doi.org/10.1016/j.procir.2015.12.109.

Chung, T., J. N. D. Gupta, and M. Qiu. 2019. "Single Machine Scheduling Problem with Batch Setups Involving Positional Deterioration Effects and Multiple Rate-Modifying Activities." *Engineering Optimization* 51 (10): 1743–60. https://doi.org/10.1080/0305215X.2018.1552269.

Church, L. K., and R. Uzsoy. 1992. "Analysis of Periodic and Event-Driven Rescheduling Policies in Dynamic Shops." *International Journal of Computer Integrated Manufacturing* 5 (3): 153–63. https://doi.org/10.1080/09511929208944524.

Clark, A. R., and S. J. Clark. 2000. "Rolling-Horizon Lot-Sizing When Set-up Times Are Sequence-Dependent." *International Journal of Production Research* 38 (10): 2287–2307. https://doi.org/10.1080/00207540050028106.

Co, H. C., J. S. Biermann, and S. K. Chen. 1990. "A Methodical Approach to the Flexible-Manufacturing-System Batching, Loading and Tool Configuration Problems." *International Journal of Production Research* 28 (12): 2171–86. https://doi.org/10.1080/00207549008942860.

Coffman, E. G., M. R. Garey, and D. S. Johnson. 1996. "Approximation Algorithms for Bin Packing: A Survey." In *Approximation Algorithms for NP-Hard Problems*, 46–93. USA: PWS Publishing Co.

Conway, R, W. Maxwell, J. O. McClain, and L. J. Thomas. 1988. "The Role of Work-in-Process Inventory in Serial Production Lines." *Operations Research* 36 (2): 229–41. https://doi.org/10.1287/opre.36.2.229.

Copil, K., M. Wörbelauer, H. Meyr, and H. Tempelmeier. 2017. "Simultaneous Lotsizing and Scheduling Problems: A Classification and Review of Models." *OR Spectrum: Quantitative Approaches in Management* 39 (1): 1–64.

Coron, J.-M., M. Kawski, and Z. Wang. 2009. "Analysis of a Conservation Law Modeling a Highly Re-Entrant Manufacturing System," July. https://arxiv.org/abs/0907.1274v1.

Crama, Y. 1997. "Combinatorial Optimization Models for Production Scheduling in Automated Manufacturing Systems." *European Journal of Operational Research* 99 (1): 136–53. https://doi.org/10.1016/S0377-2217(96)00388-8.

Crama, Y., A. W. J. Kolen, A. G. Oerlemans, and F. C. R. Spieksma. 1994. "Minimizing the Number of Tool Switches on a Flexible Machine." *International Journal of Flexible Manufacturing Systems* 6 (1): 33–54. https://doi.org/10.1007/BF01324874.

Crauwels, H. A. J., C. N. Potts, D. V. Oudheusden, and L. N. Van Wassenhove. 2005. "Branch and Bound Algorithms for Single Machine Scheduling with Batching to Minimize the Number of Late Jobs." *Journal of Scheduling* 8 (2): 161–77. https://doi.org/10.1007/s10951-005-6365-4.

Cuatrecasas-Arbós, L., J. Fortuny-Santos, P. Ruiz-de-Arbulo-López, and C. Vintró-Sanchez. 2015. "Monitoring Processes through Inventory and Manufacturing Lead Time." *Industrial Management & Data Systems* 115 (5): 951–70. https://doi.org/10.1108/IMDS-12-2014-0375.

Damodaran, P., and K. Srihari. 2004. "Mixed Integer Formulation to Minimize Makespan in a Flow Shop with Batch Processing Machines." *Mathematical and Computer Modelling* 40 (13): 1465–72. https://doi.org/10.1016/j.mcm.2005.01.005.

Damodaran, P., K. Srihari, and S. S. Lam. 2007. "Scheduling a Capacitated Batch-Processing Machine to Minimize Makespan." *Robotics and Computer-Integrated Manufacturing* 23 (2): 208–16. https://doi.org/10.1016/j.rcim.2006.02.012.

Danneberg, D., T. Tautenhahn, and F. Werner. 1999. "A Comparison of Heuristic Algorithms for Flow Shop Scheduling Problems with Setup Times and Limited Batch Size." *Mathematical and Computer Modelling* 29 (9): 101–26. https://doi.org/10.1016/S0895-7177(99)00085-0.

Danping, L., and Carman K. M. Lee. 2011. "A Review of the Research Methodology for the Re-Entrant Scheduling Problem." *International Journal of Production Research* 49 (8): 2221–42. https://doi.org/10.1080/00207541003720350.

Dauzere-Péres, S., and J.-B. Lasserre. 1993. "A Modified Shifting Bottleneck Procedure for Job-Shop Scheduling." *International Journal of Production Research* 31 (4): 923–32. https://doi.org/10.1080/00207549308956766.

Dauzère-Péres, S., and J.-B. Lasserre. 1994. "Integration of Lotsizing and Scheduling Decisions in a Job-Shop." *European Journal of Operational*

Research, Lotsizing models for production planning, 75 (2): 413–26. https://doi.org/10.1016/0377-2217(94)90085-X.

De La Fuente García, D., R. Pino Diez, and J. Parreño Fernandez. 1995. "Formación de celdas trabajo-máquina en tecnología de grupos mediante redes neuronales artificiales." *Investigaciones Europeas de Dirección y Economía de la Empresa* 1 (2): 51–68.

Defersha, F. M. 2011. "A Comprehensive Mathematical Model for Hybrid Flexible Flowshop Lot Streaming Problem." *International Journal of Industrial Engineering Computations* 2 (2): 283–94. https://doi.org/10.5267/j.ijiec.2010.07.006.

Defersha, F. M., and M. Chen. 2012. "Jobshop Lot Streaming with Routing Flexibility, Sequence-Dependent Setups, Machine Release Dates and Lag Time." *International Journal of Production Research* 50 (8): 2331–52. https://doi.org/10.1080/00207543.2011.574952.

Delgado-Arana, E. M., L. Burtseva, B. Flores-Rios, R. Ibarra, and F. Werner. 2017. "A Batch Sequencing Model for a Semiconductor Packaging Company." *Engineering Letters* 25 (2): 255–213.

Della Croce, F. 1995. "Generalized Pairwise Interchanges and Machine Scheduling." *European Journal of Operational Research*, EURO Summer Institute Combinatorial Optimization, 83 (2): 310–19. https://doi.org/10.1016/0377-2217(95)00009-F.

Ding, F.-Y. 1990. "A Pairwise Interchange Solution Procedure for a Scheduling Problem with Production of Components at a Single Facility." *Computers & Industrial Engineering* 18 (3): 325–31. https://doi.org/10.1016/0360-8352(90)90054-P.

Dong, X., H. Huang, and P. Chen. 2008. "An Improved NEH-Based Heuristic for the Permutation Flowshop Problem." *Computers & Operations Research*, Part Special Issue: Telecommunications Network Engineering, 35 (12): 3962–68. https://doi.org/10.1016/j.cor.2007.05.005.

Drexl, A., and K. Haase. 1995. "Proportional Lotsizing and Scheduling." *International Journal of Production Economics* 40 (1): 73–87. https://doi.org/10.1016/0925-5273(95)00040-U.

Drexl, A., and A. Kimms. 1997. "Lot Sizing and Scheduling — Survey and Extensions." *European Journal of Operational Research* 99 (2): 221–35. https://doi.org/10.1016/S0377-2217(97)00030-1.

Duwayri, Z., M. Mollaghasemi, D. Nazzal, and G. Rabadi. 2006. "Scheduling Setup Changes at Bottleneck Workstations in Semiconductor Manufacturing." *Production Planning & Control* 17 (7): 717–27. https://doi.org/10.1080/09537280600901426.

Ebrahimi, M., S. M. T. Fatemi Ghomi, and B. Karimi. 2014. "Hybrid Flow Shop Scheduling with Sequence Dependent Family Setup Time and Uncertain Due Dates." *Applied Mathematical Modelling* 38 (9): 2490–2504. https://doi.org/10.1016/j.apm.2013.10.061.

Ei-Essawy, I. G. K., and J. Torrance. 1972. "Component Flow Analysis — an Effective Approach to Production Systems' Design." *Production Engineer* 51 (5): 165–70. https://doi.org/10.1049/tpe.1972.0027.

Elanchezhian, C., S. T. Selwyn, and S. Sundar. 2008. *Computer Aided Manufacturing*. New Delhi; Boston.

Elmaghraby, S. E. 1978. "The Economic Lot Scheduling Problem (ELSP): Review and Extensions." *Management Science* 24 (6): 587–98.

Epstein, L., and A. Levin. 2008. "An APTAS for Generalized Cost Variable-Sized Bin Packing." *SIAM Journal on Computing* 38 (1): 411–28. https://doi.org/10.1137/060670328.

Erel, E., and J. B. Ghosh. 2007. "Batch Scheduling to Minimize the Weighted Number of Tardy Jobs." *Computers and Industrial Engineering* 53 (3): 394–400. https://doi.org/10.1016/j.cie.2007.03.006.

Eren, T., and E. Güner. 2008. "The Tricriteria Flowshop Scheduling Problem." *The International Journal of Advanced Manufacturing Technology* 36 (11): 1210–20. https://doi.org/10.1007/s00170-007-0931-1.

Erlenkotter, D. 1990. "Ford Whitman Harris and the Economic Order Quantity Model." *Operations Research* 38 (6): 937–46. https://doi.org/10.1287/opre.38.6.937.

Esmaeilian, B., S. Behdad, and B. Wang. 2016. "The Evolution and Future of Manufacturing: A Review." *Journal of Manufacturing Systems* 39 (April): 79–100. https://doi.org/10.1016/j.jmsy.2016.03.001.

Fakher, H. B., M. Nourelfath, and M. Gendreau. 2017. "A Cost Minimisation Model for Joint Production and Maintenance Planning under Quality Constraints." *International Journal of Production Research* 55 (8): 2163–76. https://doi.org/10.1080/00207543.2016.1201605.

Fan, W., J. Pei, X. Liu, P. M. Pardalos, and M. Kong. 2018. "Serial-Batching Group Scheduling with Release Times and the Combined Effects of Deterioration and Truncated Job-Dependent Learning." *Journal of Global Optimization* 71 (1): 147–63. https://doi.org/10.1007/s10898-017-0536-7.

Fandel, G., and C. Stammen-Hegene. 2006. "Simultaneous Lot Sizing and Scheduling for Multi-Product Multi-Level Production." *International Journal of Production Economics*, Theoretical Issues in Production

Scheduling and Control & Planning and Control of Supply Chains and Production, 104 (2): 308–16. https://doi.org/10.1016/j.ijpe.2005.04.011.

Fattahi, P., V. Azizi, and M. Jabbari. 2015. "Lot Streaming in No-Wait Multi Product Flowshop Considering Sequence Dependent Setup Times and Position Based Learning Factors." *International Journal of Engineering* 28 (7): 1031–39.

Federgruen, A., and M. Tzur. 1991. "A Simple Forward Algorithm to Solve General Dynamic Lot Sizing Models with n Periods in 0(n Log n) or 0(n) Time." *Management Science* 37 (8): 909–25.

Feldmann, M., and D. Biskup. 2008. "Lot Streaming in a Multiple Product Permutation Flow Shop with Intermingling." *International Journal of Production Research* 46 (1): 197–216. https://doi.org/10.1080/00207540600930065.

FINCH, BY RON J., and JAMES F. COX. 1986. "An Examination of Just-in-Time Management for the Small Manufacturer: With an Illustration." *International Journal of Production Research* 24 (2): 329–42. https://doi.org/10.1080/00207548608919732.

Fleischmann, B. 1990. "The Discrete Lot-Sizing and Scheduling Problem." *European Journal of Operational Research* 44 (3): 337–48. https://doi.org/10.1016/0377-2217(90)90245-7.

—. 1994. "The Discrete Lot-Sizing and Scheduling Problem with Sequence-Dependent Setup Costs." *European Journal of Operational Research*, Lotsizing models for production planning, 75 (2): 395–404. https://doi.org/10.1016/0377-2217(94)90083-3.

Fleischmann, B., and H. Meyr. 1997. "The General Lotsizing and Scheduling Problem." *Operations-Research-Spektrum* 19 (1): 11–21. https://doi.org/10.1007/BF01539800.

Florian, M., J. K. Lenstra, and A. H. G. R. Rinnooy Kan. 1980. "Deterministic Production Planning: Algorithms and Complexity." *Management Science* 26 (7): 669–79.

Flynn, B. B. 1987. "Repetitive Lots: The Use of a Sequence-Dependent Set-Up Time Scheduling Procedure in Group Technology and Traditional Shops." *Journal of Operations Management* 7 (1 and 2): 203–16.

Forghani, K., S. M. T. Fatemi Ghomi, and R. Kia. 2020. "Solving an Integrated Cell Formation and Group Layout Problem Using a Simulated Annealing Enhanced by Linear Programming." *Soft Computing* 24 (15): 11621–39. https://doi.org/10.1007/s00500-019-04626-8.

Framinan, J. M., and P. Perez-Gonzalez. 2017. "The 2-Stage Assembly Flowshop Scheduling Problem with Total Completion Time: Efficient

Constructive Heuristic and Metaheuristic." *Computers & Operations Research* 88 (December): 237–46. https://doi.org/10.1016/j.cor.2017.07.012.

Friesen, D. K., and M. A. Langston. 1986. "Variable Sized Bin Packing." *SIAM Journal on Computing* 15 (1): 222–30. https://doi.org/10.1137/0215016.

Fu, R., T. C. E. Cheng, C. T. Ng, and J. Yuan. 2010. "Online Scheduling on Two Parallel-Batching Machines with Limited Restarts to Minimize the Makespan." *Information Processing Letters* 110 (11): 444–50. https://doi.org/10.1016/j.ipl.2010.04.008.

Gaafar, L. 2006. "Applying Genetic Algorithms to Dynamic Lot Sizing with Batch Ordering." *Computers & Industrial Engineering*, Special Issue on Selected Papers from the 34th. International Conference on Computers and Industrial Engineering (ICC&IE), 51 (3): 433–44. https://doi.org/10.1016/j.cie.2006.08.006.

Gallego, G. 1990. "An Extension to the Class of Easy Economic Lot Scheduling Problems." *IIE Transactions* 22 (2): 189–90. https://doi.org/10.1080/07408179008964172.

Gallego, G., and D. X. Shaw. 1997. "Complexity of the ELSP with General Cyclic Schedules." *IIE Transactions* 29 (2): 109–13. https://doi.org/10.1080/07408179708966318.

García-Mata, C. L., P. R. Márquez-Gutiérrez, and L. Burtseva. 2015. "Rescheduling in Industrial Environments: Emerging Technologies and Forthcoming Trends." *International Journal of Combinatorial Optimization Problems and Informatics* 6 (3): 34–48.

Garey, M. R., and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

Garey, M. R., D. S. Johnson, and R. Sethi. 1976. "The Complexity of Flowshop and Jobshop Scheduling." *Mathematics of Operations Research* 1 (2): 117–29. https://doi.org/10.1287/moor.1.2.117.

Gaury, E. G. A., J. P. C. Kleijnen, and H. Pierreval. 2001. "A Methodology to Customize Pull Control Systems." *The Journal of the Operational Research Society* 52 (7): 789–99.

Geiger, C. D., R. Uzsoy, and H. Aytuğ. 2006. "Rapid Modeling and Discovery of Priority Dispatching Rules: An Autonomous Learning Approach." *Journal of Scheduling* 9 (1): 7–34. https://doi.org/10.1007/s10951-006-5591-8.

Geismar, H. N., M. Dawande, and C. Sriskandarajah. 2004. "Robotic Cells with Parallel Machines: Throughput Maximization in Constant Travel-Time Cells." *Journal of Scheduling* 7 (5): 375–95. https://doi.org/10.1023/B:JOSH.0000036861.28456.5d.

Gen, M., and R. Cheng. 1999. *Genetic Algorithms and Engineering Optimization*. New York.

Gen, M., W. Zhang, and L. Lin. 2015. "Multiobjective Hybrid Genetic Algorithms for Manufacturing Scheduling: Part II Case Studies of HDD and TFT-LCD." In *Proceedings of the Ninth International Conference on Management Science and Engineering Management*, edited by J. Xu, S. Nickel, V. Machado, and A. Hajiyev, 27–54. Advances in Intelligent Systems and Computing. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-662-47241-5_2.

Geng, N., Z. Jiang, and F. Chen. 2009. "Stochastic Programming Based Capacity Planning for Semiconductor Wafer Fab with Uncertain Demand and Capacity." *European Journal of Operational Research* 198 (3): 899–908. https://doi.org/10.1016/j.ejor.2008.09.029.

Gest, G., S. J. Culley, R. I. McIntosh, A. R. Mileham, and G. W. Owen. 1995. "Review of Fast Tool Change Systems." *Computer Integrated Manufacturing Systems* 8 (3): 205–10. https://doi.org/10.1016/0951-5240(95)00011-H.

Ghezail, F., H. Pierreval, and S. Hajri-Gabouj. 2010. "Analysis of Robustness in Proactive Scheduling: A Graphical Approach." *Computers & Industrial Engineering*, Scheduling in Healthcare and Industrial Systems, 58 (2): 193–98. https://doi.org/10.1016/j.cie.2009.03.004.

Ghosh, T., and P. K. Dan. 2011. "Effective Clustering Method for Group Technology Problems: A Short Communication." *E -Journal of Science & Technology* 6 (22): 23–28.

Gicquel, C., M. Minoux, and Y. Dallery. 2008. "Capacitated Lot Sizing Models: A Literature Review." https://hal.archives-ouvertes.fr/hal-00255830.

Giglio, D. 2015. "Optimal Control Strategies for Single-Machine Family Scheduling with Sequence-Dependent Batch Setup and Controllable Processing Times." *Journal of Scheduling* 18 (5): 525–43. https://doi.org/10.1007/s10951-015-0440-2.

Gilmore, P. C., and R. E. Gomory. 1964. "Sequencing a One State-Variable Machine: A Solvable Case of the Traveling Salesman Problem." *Operations Research* 12 (5): 655–79. https://doi.org/10.1287/opre.12.5.655.

Giret, A., and V. Botti. 2004. "Holons and Agents." *Journal of Intelligent Manufacturing* 15 (5): 645–59. https://doi.org/10.1023/B:JIMS.0000037714.56201.a3.

Glass, C. A., J. N. D. Gupta, and C. N. Potts. 1994. “Lot Streaming in Three-Stage Production Processes.” *European Journal of Operational Research*, Lotsizing models for production planning, 75 (2): 378–94. https://doi.org/10.1016/0377-2217(94)90082-5.

Glock, C. H., E. H. Grosse, and J. M. Ries. 2014. “The Lot Sizing Problem: A Tertiary Study.” *International Journal of Production Economics*, Celebrating a century of the economic order quantity model, 155 (September): 39–51. https://doi.org/10.1016/j.ijpe.2013.12.009.

Godina, R., C. Pimentel, F. J. G. Silva, and J. C. O. Matias. 2018. “A Structural Literature Review of the Single Minute Exchange of Die: The Latest Trends.” *Procedia Manufacturing*, 28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USAGlobal Integration of Intelligent Manufacturing and Smart Industry for Good of Humanity, 17 (January): 783–90. https://doi.org/10.1016/j.promfg.2018.10.129.

Goldratt, E. M. 2012. *The Goal: A Process of Ongoing Improvement - 30th Anniversary Edition*. Great Barrington, Mass.

Golhar, D. Y., and C. L. Stamm. 1991. “The Just-in-Time Philosophy: A Literature Review.” *International Journal of Production Research* 29 (4): 657–76. https://doi.org/10.1080/00207549108930094.

Gombinski, J. 1967. “Group Technology - An Introduction.” *Production Engineer* 46 (9): 557–64. https://doi.org/10.1049/tpe:19670075.

—. 1968. “Component Classification- Why and How?” In *Machinery and Production Engineering*, 547–50.

Gómez-Gasquet, P., R. Segura-Andrés, and C. Andrés-Romano. 2013. “A Review of Lot Streaming in a Flow Shop Environment with Makespan Criteria.” *Journal of Industrial Engineering and Management* 6 (3): 761–70. https://doi.org/10.3926/jiem.553.

Gong, H., L. Tang, and C. W. Duin. 2010. “A Two-Stage Flow Shop Scheduling Problem on a Batching Machine and a Discrete Machine with Blocking and Shared Setup Times.” *Computers & Operations Research*, Disruption Management, 37 (5): 960–69. https://doi.org/10.1016/j.cor.2009.08.001.

Gonzalez, T. F. 2004. “Open Shop Scheduling.” In *Handbook of Scheduling Algorithms, Models, and Performance Analysis*, Joseph Y-T. Leung. Chapman and Hall/CRC.

Gonzalez, T. F., and S. Sahni. 1976. “Open Shop Scheduling to Minimize Finish Time.” *Journal of the ACM* 23 (4): 665–79. https://doi.org/10.1145/321978.321985.

Gourgand, M., N. Grangeon, and S. Norre. 1999. "Metaheuristics for the Deterministic Hybrid Flow Shop Problem." In *Proceedings of the International Conference on Industrial Engineering and Production Management*, 136–45. FUCAM – INRIA.

Gower, J. C. 1971. "A General Coefficient of Similarity and Some of Its Properties." *Biometrics* 27 (4): 857–71. https://doi.org/10.2307/2528823.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. 1979. "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey." In *Annals of Discrete Mathematics*, edited by P. L. Hammer, E. L. Johnson, and B. H. Korte, 5:287–326. Discrete Optimization II. Elsevier. https://doi.org/10.1016/S0167-5060(08)70356-X.

Graves, S. C., and M. M. Kostreva. 1986. "Overlapping Operations in Material Requirements Planning." *Journal of Operations Management*, Special Combined Issue, 6 (3): 283–94. https://doi.org/10.1016/0272-6963(86)90004-5.

Greene, T. J., and R. P. Sadowski. 1984. "A Review of Cellular Manufacturing Assumptions, Advantages and Design Techniques." *Journal of Operations Management* 4 (2): 85–97. https://doi.org/10.1016/0272-6963(84)90025-1.

GT in MFG system. n.d. "Group-Technology-in-MFG." Yumpu.Com. Accessed November 24, 2020. https://www.yumpu.com/en/document/view/36324026/chapter4-group-technology-in-mfg-systemspdf.

Gupta, D., and T. Magnusson. 2005. "The Capacitated Lot-Sizing and Scheduling Problem with Sequence-Dependent Setup Costs and Setup Times." *Computers & Operations Research* 32 (4): 727–47. https://doi.org/10.1016/j.cor.2003.08.014.

Gupta, D., C. T. Maravelias, and J. M. Wassick. 2016. "From Rescheduling to Online Scheduling." *Chemical Engineering Research and Design*, Process Systems Engineering - A Celebration in Professor Roger Sargent's 90th Year, 116 (December): 83–97. https://doi.org/10.1016/j.cherd.2016.10.035.

Gupta, J. N. D. 1986. "Flowshop Schedules with Sequence Dependent Setup Times." *Journal of the Operations Research Society of Japan* 29 (3): 206–19. https://doi.org/10.15807/jorsj.29.206.

—. 1988. "Two-Stage, Hybrid Flowshop Scheduling Problem." *The Journal of the Operational Research Society* 39 (4): 359–64. https://doi.org/10.2307/2582115.

Gupta, J. N. D., and W. P. Darrow. 1986. "The Two-Machine Sequence Dependent Flowshop Scheduling Problem." *European Journal of Operational Research*, Flexible Manufacturing Systems, 24 (3): 439–46. https://doi.org/10.1016/0377-2217(86)90037-8.

Gupta, J. N. D., and S. K. Gupta. 1988. "Single Facility Scheduling with Nonlinear Processing Times." *Computers & Industrial Engineering* 14 (4): 387–93. https://doi.org/10.1016/0360-8352(88)90041-1.

Gupta, J. N. D., and E. A. Tunc. 1998. "Minimizing Tardy Jobs in a Two-Stage Hybrid Flowshop." *International Journal of Production Research* 36 (9): 2397–2417. https://doi.org/10.1080/002075498192599.

Haase, K. 1996. "Capacitated Lot-Sizing with Sequence Dependent Setup Costs." *Operations-Research-Spektrum* 18 (1): 51–59. https://doi.org/10.1007/BF01539882.

Hachicha, W., F. Masmoudi, and M. Haddar. 2008. "Formation of Machine Groups and Part Families in Cellular Manufacturing Systems Using a Correlation Analysis Approach." *The International Journal of Advanced Manufacturing Technology* 36 (11): 1157–69. https://doi.org/10.1007/s00170-007-0928-9.

Hall, L. A., and D. B. Shmoys. 1992. "Jackson's Rule for Single-Machine Scheduling: Making a Good Heuristic Better | Mathematics of Operations Research." *Mathematics of Operations Research* 17 (1). https://doi.org/10.1287/moor.17.1.22.

Hall, N. G., G. Laporte, E. Selvarajah, and C. Sriskandarajah. 2005. "Scheduling and Lot Streaming in Two-Machine Open Shops with No-Wait in Process." *Naval Research Logistics (NRL)* 52 (3): 261–75. https://doi.org/10.1002/nav.20065.

Hallett, W. J. 1964. "Classifying 250,000 Drawings by Brisch System." *Machine Design*.

Harris, F. W. 1913. "How Many Parts to Make at Once Factory." *The Magazine of Management* 10 (2): 135–36.

Hathaway, H. D. 1920. "The Mnemonic Systems of Classification; As Used in the Taylor System of Management." *Industrial Management* 9 (3): 173–83.

Hazır, Ö., and S. Kedad-Sidhoum. 2014. "Batch Sizing and Just-in-Time Scheduling with Common Due Date." *Annals of Operations Research* 213 (1): 187–202. https://doi.org/10.1007/s10479-012-1289-9.

Heragu, S. S., and J.-S. Chen. 1998. "Optimal Solution of Cellular Manufacturing System Design: Benders' Decomposition Approach." *European Journal of Operational Research* 107 (1): 175–92. https://doi.org/10.1016/S0377-2217(97)00256-7.

Heuvel, W. van den, and A. P. M. Wagelmans. 2006. "An Efficient Dynamic Programming Algorithm for a Special Case of the Capacitated Lot-Sizing Problem." *Computers & Operations Research*, Part Special Issue: Recent Algorithmic Advances for Arc Routing Problems, 33 (12): 3583–99. https://doi.org/10.1016/j.cor.2005.02.046.

Ho, J. C., A. O. Solis, and Y.-L. Chang. 2007. "An Evaluation of Lot-Sizing Heuristics for Deteriorating Inventory in Material Requirements Planning Systems." *Computers & Operations Research* 34 (9): 2562–75. https://doi.org/10.1016/j.cor.2005.09.020.

Ho, K. I-J., J. Y-T. Leung, and W-D. Wei. 1993. "Complexity of Scheduling Tasks with Time-Dependent Execution Times." *Information Processing Letters* 48 (6): 315–20. https://doi.org/10.1016/0020-0190(93)90175-9.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Oxford, England: U Michigan Press.

Hoogeveen, J. A., J. K. Lenstra, and B. Veltman. 1996. "Preemptive Scheduling in a Two-Stage Multiprocessor Flow Shop Is NP-Hard." *European Journal of Operational Research* 89 (1): 172–75. https://doi.org/10.1016/S0377-2217(96)90070-3.

Hopp, W. J., and M. L. Roof. 1998. "Setting WIP Levels with Statistical Throughput Control (STC) in CONWIP Production Lines." *International Journal of Production Research* 36 (4): 867–82. https://doi.org/10.1080/002075498193435.

Houtzeel, A. 1981. "Computer Assisted Process Planning Minimizes Design and Manufacturing Costs, Organization for Industrial Research Inc." *IE Transactions*, 60–64.

Houtzeel, A., and B. Schilperoot,. 1975. "Group Technology via Computer (MICLASS System)." *American Machinist* 119 (17): 39–44.

Hsu, W.-L. 1983. "On the General Feasibility Test of Scheduling Lot Sizes for Several Products on One Machine." *Management Science* 29 (1): 93–105. https://doi.org/10.1287/mnsc.29.1.93.

Huang, Hsiang-Hsi, W. Pei, H.-H. Wu, and M.-D. May. 2013. "A Research on Problems of Mixed-Line Production and the Re-Scheduling." *Robotics and Computer-Integrated Manufacturing*, Extended Papers Selected from FAIM 2011, 29 (3): 64–72. https://doi.org/10.1016/j.rcim.2012.04.014.

Huang, W., C.-C. Wu, and C. Liu. 2018. “Single-Machine Batch Scheduling Problem with Job Rejection and Resource Dependent Processing Times.” *RAIRO - Operations Research* 52 (2): 315–34. https://doi.org/10.1051/ro/2017040.

Huang, Y., L. Zheng, B. C. Williams, L. Tang, and H. Yang. 2010. “Incremental Temporal Reasoning in Job Shop Scheduling Repair.” In *2010 IEEE International Conference on Industrial Engineering and Engineering Management*, 1276–80. https://doi.org/10.1109/IEEM.2010.5674383.

Hum, S.-H., and C.-K. Lee. 1998. “JIT Scheduling Rules: A Simulation Evaluation.” *Omega* 26 (3): 381–95. https://doi.org/10.1016/S0305-0483(97)00066-2.

Hung, Y.-F., C.-H. Liang, and J. C. Chen. 2013. “Sensitivity Search for the Rescheduling of Semiconductor Photolithography Operations.” *The International Journal of Advanced Manufacturing Technology* 67 (1): 73–84. https://doi.org/10.1007/s00170-013-4754-y.

Husseinzadeh Kashan, A., B. Karimi, and F. Jolai. 2010. “An Effective Hybrid Multi-Objective Genetic Algorithm for Bi-Criteria Scheduling on a Single Batch Processing Machine with Non-Identical Job Sizes.” *Engineering Applications of Artificial Intelligence* 23 (6): 911–22. https://doi.org/10.1016/j.engappai.2010.01.031.

Hyde, W. F. 1981. *Improving Productivity by Classification, Coding and Data Base Standardization: The Key to Maximizing Cad/Cam and Group Technology;Manufacturing Engineering & Materials Processing*. New York.

Ikura, Y., and M. Gimple. 1986. “Scheduling Algorithms for a Single Batch Processing Machine.” *Operational Research Letters* 5: 61–65.

Inderfurth, K., A. Janiak, M. Y. Kovalyov, and F. Werner. 2006. “Batching Work and Rework Processes with Limited Deterioration of Reworkables.” *Computers & Operations Research* 33 (6): 1595–1605. https://doi.org/10.1016/j.cor.2004.11.009.

Ivanov, E. K. 1968. *Group Production Organization and Technology*. Business Publications.

Jackson, J. R. 1955. “Scheduling a Production Line to Minimize Maximum Tardiness.” *Management Science Research Project*. https://ci.nii.ac.jp/naid/10015080697/.

Jacobs, F. R., W. L. Berry, D. C. Whybark, and T. E. Vollmann. 2011. *Manufacturing Planning and Control for Supply Chain Management*. McGraw Hill Professional.

Janiak, A., and M. Y. Kovalyov. 1995. "Single Machine Group Scheduling with Ordered Criteria." *Annals of Operations Research* 57 (1): 191–201. https://doi.org/10.1007/BF02099697.

Jans, Raf, and Z. Degraeve. 2008. "Modeling Industrial Lot Sizing Problems: A Review." *International Journal of Production Research* 46 (6): 1619–43. https://doi.org/10.1080/00207540600902262.

Jeon, G., and G. R. Leep. 2006. "Forming Part Families by Using Genetic Algorithm and Designing Machine Cells under Demand Changes." *Computers & Operations Research* 33 (1): 263–83. https://doi.org/10.1016/j.cor.2005.03.033.

Jeong, H., S. Woo, S. Kang, and J. Park. 1997. "A Batch Splitting Heuristic for Dynamic Job Shop Scheduling Problem." *Computers & Industrial Engineering*, Selected Papers from the Proceedings of 1996 ICC&IC, 33 (3): 781–84. https://doi.org/10.1016/S0360-8352(97)00252-0.

Ji, M., and T. C. E. Cheng. 2008. "Parallel-Machine Scheduling with Simple Linear Deterioration to Minimize Total Completion Time." *European Journal of Operational Research* 188 (2): 342–47. https://doi.org/10.1016/j.ejor.2007.04.050.

Jia, Z.-h., X.-x Zhuo, J. Y-T. Leung, and K. Li. 2019. "Integrated Production and Transportation on Parallel Batch Machines to Minimize Total Weighted Delivery Time." *Computers & Operations Research* 102 (February): 39–51. https://doi.org/10.1016/j.cor.2018.07.026.

Jin, Z. H., K. Ohno, T. Ito, and S. E. Elmaghraby. 2002. "Scheduling Hybrid Flowshops in Printed Circuit Board Assembly Lines*." *Production and Operations Management* 11 (2): 216–30. https://doi.org/10.1111/j.1937-5956.2002.tb00492.x.

Jodlbauer, H. 2006. "An Approach for Integrated Scheduling and Lot-Sizing." *European Journal of Operational Research* 172 (2): 386–400. https://doi.org/10.1016/j.ejor.2004.09.050.

Johnson, S. M. 1954. "Optimal Two- and Three-Stage Production Schedules with Setup Times Included." *Naval Research Logistics Quarterly* 1 (1): 61–68. https://doi.org/10.1002/nav.3800010110.

Jolai, F. 2005. "Minimizing Number of Tardy Jobs on a Batch Processing Machine with Incompatible Job Families." *European Journal of Operational Research*, Logistics: From Theory to Application, 162 (1): 184–90. https://doi.org/10.1016/j.ejor.2003.10.011.

Jones, P. C., and R. R. Inman. 1989. "When Is The Economic Lot Scheduling Problem Easy?" *IIE Transactions* 21 (1): 11–20. https://doi.org/10.1080/07408178908966202.

Jung, S., Y.-B. Woo, and B. S. Kim. 2017. “Two-Stage Assembly Scheduling Problem for Processing Products with Dynamic Component-Sizes and a Setup Time.” *Computers & Industrial Engineering* 104 (February): 98–113. https://doi.org/10.1016/j.cie.2016.12.030.

Jungwattanakit, J., M. Reodecha, P. Chaovalitwongse, and F. Werner. 2009. “A Comparison of Scheduling Algorithms for Flexible Flow Shop Problems with Unrelated Parallel Machines, Setup Times, and Dual Criteria.” *Computers & Operations Research*, Scheduling for Modern Manufacturing, Logistics, and Supply Chains, 36 (2): 358–78. https://doi.org/10.1016/j.cor.2007.10.004.

Junqueira, R. de Á. R., and R. Morabito. 2017. “Abordagens de otimização para a programação e sequenciamento das frentes de colheita de cana-de-açúcar.” *Gestão & Produção* 24 (2): 407–22. https://doi.org/10.1590/0104-530x1882-16.

Kaihara, T., S. Kurose, and N. Fujii. 2012. “A Proposal on Optimized Scheduling Methodology and Its Application to an Actual-Scale Semiconductor Manufacturing Problem.” *CIRP Annals* 61 (1): 467–70. https://doi.org/10.1016/j.cirp.2012.03.077.

Kamrani, Ali K., Maryam Azimi, Abdulrahman M. Al-Ahmari, Maryam Azimi, and Abdulrahman M. Al-Ahmari. 2013. *Methods in Product Design : New Strategies in Reengineering*. CRC Press. https://doi.org/10.4324/9781315300252.

Karimi, B., S. M. T. Fatemi Ghomi, and J. M. Wilson. 2003. “The Capacitated Lot Sizing Problem: A Review of Models and Algorithms.” *Omega* 31 (5): 365–78. https://doi.org/10.1016/S0305-0483(03)00059-8.

Karmarkar, U. S., Sh. Kekre, and Su. Kekre. 1987. “The Dynamic Lot-Sizing Problem with Startup and Reservation Costs.” *Operations Research* 35 (3): 389–98. https://doi.org/10.1287/opre.35.3.389.

Karmarkar, U. S., and L. Schrage. 1985. “The Deterministic Dynamic Product Cycling Problem.” *Operations Research* 33 (2): 326–45.

Katragjini, K., E. Vallada, and R.. Ruiz. 2013. “Flow Shop Rescheduling under Different Types of Disruption.” *International Journal of Production Research* 51 (3): 780–97. https://doi.org/10.1080/00207543.2012.666856.

Khurana, K., and P. C. Bagga. 1984. “Minimizing the Makespan in a 2-Machine Flowshop with Time Lags and Setup Conditions.” *Zeitschrift Für Operations Research* 28 (5): 163–74. https://doi.org/10.1007/BF01920918.

Kilic, O. A., and S. A. Tarim. 2011. “An Investigation of Setup Instability in Non-Stationary Stochastic Inventory Systems.” *International Journal*

of Production Economics, Leading Edge of Inventory Research, 133 (1): 286–92. https://doi.org/10.1016/j.ijpe.2010.04.021.

Kim, J. S., and R. C. Leachman. 1994. "Decomposition Method Application to a Large Scale Linear Programming WIP Projection Model." *European Journal of Operational Research* 74 (1): 152–60. https://doi.org/10.1016/0377-2217(94)90213-5.

Kim, S., and R. Uzsoy. 2009. "Heuristics for Capacity Planning Problems with Congestion." *Computers & Operations Research* 36 (6): 1924–34. https://doi.org/10.1016/j.cor.2008.06.006.

Kim, S.-Il, H.-S. Choi, and D.-H. Lee. 2006. "Tabu Search Heuristics for Parallel Machine Scheduling with Sequence-Dependent Setup and Ready Times." In *Computational Science and Its Applications - ICCSA 2006*, edited by M. Gavrilova, O. Gervasi, V. Kumar, C. J. K. Tan, D. Taniar, A. Laganá, Y. Mun, and H. Choo, 728–37. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/11751595_77.

Kimms, A. 1996. "Stability Measures for Rolling Schedules with Applications to Capacity Expansion Planning, Master Production Scheduling, and Lot Sizing." 418. *Manuskripte Aus Den Instituten Für Betriebswirtschaftslehre Der Universität Kiel*. Manuskripte Aus Den Instituten Für Betriebswirtschaftslehre Der Universität Kiel. Christian-Albrechts-Universität zu Kiel, Institut für Betriebswirtschaftslehre. https://ideas.repec.org/p/zbw/cauman/418.html.

Kimms, A. 1998. "Stability Measures for Rolling Schedules with Applications to Capacity Expansion Planning, Master Production Scheduling, and Lot Sizing." *Omega* 26 (3): 355–66. https://doi.org/10.1016/S0305-0483(97)00056-X.

Kimms, A., and A. Drexl. 1998. "Some Insights into Proportional Lot Sizing and Scheduling." *Journal of the Operational Research Society* 49 (11): 1196–1205. https://doi.org/10.1057/palgrave.jors.2600632.

King, J. R. 1980. "Machine-Component Grouping in Production Flow Analysis: An Approach Using a Rank Order Clustering Algorithm." *International Journal of Production Research* 18 (2): 213–32. https://doi.org/10.1080/00207548008919662.

King, J. R., and V. Nakornchai. 1982. "Machine-Component Group Formation in Group Technology: Review and Extension." *International Journal of Production Research* 20 (2): 117–33. https://doi.org/10.1080/00207548208947754.

Kochhar, S., and R. J. T. Morris. 1987. "Heuristic Methods for Flexible Flow Line Scheduling." *Journal of Manufacturing Systems* 6 (4): 299–314. https://doi.org/10.1016/0278-6125(87)90006-9.

Koh, S.-G., P.-H. Koo, D.-C. Kim, and W.-S. Hur. 2005. "Scheduling a Single Batch Processing Machine with Arbitrary Job Sizes and Incompatible Job Families." *International Journal of Production Economics* 98 (1): 81–96. https://doi.org/10.1016/j.ijpe.2004.10.001.

Komaki, G. M., E. Teymourian, V. Kayvanfar, and Z. Booyavi. 2017. "Improved Discrete Cuckoo Optimization Algorithm for the Three-Stage Assembly Flowshop Scheduling Problem." *Computers & Industrial Engineering* 105 (March): 158–73. https://doi.org/10.1016/j.cie.2017.01.006.

Kong, M., X. Liu, J. Pei, Z. Zhou, and P. M. Pardalos. 2019. "Parallel-Batching Scheduling of Deteriorating Jobs with Non-Identical Sizes and Rejection on a Single Machine." *Optimization Letters* 14 (4): 857–71. https://doi.org/10.1007/s11590-019-01389-x.

Koulamas, C., and G. J. Kyparisis. 2007. "Single-Machine and Two-Machine Flowshop Scheduling with General Learning Functions." *European Journal of Operational Research* 178 (2): 402–7. https://doi.org/10.1016/j.ejor.2006.01.030.

—. 2008. "Single-Machine Scheduling Problems with Past-Sequence-Dependent Setup Times." *European Journal of Operational Research* 187 (3): 1045–49. https://doi.org/10.1016/j.ejor.2006.03.066.

Kress, D., M. Barketau, and E. Pesch. 2018. "Single-Machine Batch Scheduling to Minimize the Total Setup Cost in the Presence of Deadlines." *Journal of Scheduling* 21 (6): 595–606. https://doi.org/10.1007/s10951-018-0561-5.

Kuik, R., M. Salomon, and L. N. van Wassenhove. 1994. "Batching Decisions: Structure and Models." *European Journal of Operational Research*, Lotsizing models for production planning, 75 (2): 243–63. https://doi.org/10.1016/0377-2217(94)90072-8.

Kumar, Katuru Phani Raja, Dr V. Kamala, and Dr Acs Kumar. 2013. "Optimisation Of Job Shop Scheduling Problem For Batch Production." *International Journal of Engineering Research & Technology* 2 (7): 1634–36.

Kumar Manjeshwar, P., P. Damodaran, and K. Srihari. 2009. "Minimizing Makespan in a Flow Shop with Two Batch-Processing Machines Using Simulated Annealing." *Robotics and Computer-Integrated Manufacturing* 25 (3): 667–79. https://doi.org/10.1016/j.rcim.2008.05.003.

Kumar, R., A. Kusiak, and A. Vannelli. 1986. "Grouping of Parts and Components in Flexible Manufacturing Systems." *European Journal of Operational Research*, Flexible Manufacturing Systems, 24 (3): 387–97. https://doi.org/10.1016/0377-2217(86)90032-9.

Kumar, V., S. Kumar, M. K. Tiwari, and F. Chan. 2008. "Performance Evaluation of Flexible Manufacturing Systems under Uncertain and Dynamic Situations." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 222 (7): 915–34. https://doi.org/10.1243/09544054JEM950.

Kunzler, J., and V. Nakornachal. 1982. "CAD/CAM Education and Status of the Manufacturing Consortium." In *On Computer Aided Manufacturing and Productivity*.

Kuo, W.-H., and D.-L. Yang. 2007. "Single Machine Scheduling with Past-Sequence-Dependent Setup Times and Learning Effects." *Information Processing Letters* 102 (1): 22–26. https://doi.org/10.1016/j.ipl.2006.11.002.

Kušar, J., T. Berlec, F. Žefran, and M. Starbek. 2010. "Reduction of Machine Setup Time." *Journal of Mechanical Engineering* 56: 833–45.

Kusiak, A. 1987. "The Generalized Group Technology Concept." *International Journal of Production Research* 25 (4): 561–69. https://doi.org/10.1080/00207548708919861.

Kusiak, A., and C. H. Cheng. 1990. "A Branch-and-Bound Algorithm for Solving the Group Technology Problem." *Annals of Operations Research* 26 (1): 415–31. https://doi.org/10.1007/BF03543078.

Kusiak, A., and M. Cho. 1992. "Similarity Coefficient Algorithms for Solving the Group Technology Problem." *International Journal of Production Research* 30 (11): 2633–46. https://doi.org/10.1080/00207549208948181.

Kusiak, A., and W. S. Chow. 1987. "Efficient Solving of the Group Technology Problem." *Journal of Manufacturing Systems* 6 (2): 117–24. https://doi.org/10.1016/0278-6125(87)90035-5.

Kusiak, A., A. Vannelli, and K. R. Kumar. 1985. "Grouping Problem in Scheduling Flexible Manufacturing Systems." *Robotica* 3 (4): 245–52. https://doi.org/10.1017/S0263574700002344.

Kuster, J., D. Jannach, and G. Friedrich. 2010. "Applying Local Rescheduling in Response to Schedule Disruptions." *Annals of Operations Research* 180 (1): 265–82. https://doi.org/10.1007/s10479-008-0488-x.

Lalitha, J. L., N. Mohan, and V. M. Pillai. 2017. "Lot Streaming in [N-1](1)+N(m) Hybrid Flow Shop." *Journal of Manufacturing Systems* 44 (July): 12–21. https://doi.org/10.1016/j.jmsy.2017.04.018.

Lamba, K., R. Kumar, S. Mishra, and S. Rajput. 2020. "Sustainable Dynamic Cellular Facility Layout: A Solution Approach Using Simulated Annealing-Based Meta-Heuristic." *Annals of Operations Research* 290 (1): 5–26. https://doi.org/10.1007/s10479-019-03340-w.

Lambert, S., B. Cyr, G. AbdulNour, and J. Drolet. 1997. "Comparison Study of Scheduling Rules and Set-up Policies for a SMT Production Line." *Computers & Industrial Engineering*, Proceedings of the 21st International Conference on Computers and Industrial Engineering, 33 (1): 369–72. https://doi.org/10.1016/S0360-8352(97)00114-9.

Lambrecht, M. R., J. Vander Eecken, and H. Vanderveken. 1983. "A Comparative Study of Lot Sizing Procedures for Multi-Stage Assembly Systems." *Operations-Research-Spektrum* 5 (1): 33–43. https://doi.org/10.1007/BF01720285.

Langston, M. A. 1984. "Performance of Heuristics for a Computer Resource Allocation Problem." *SIAM Journal on Algebraic Discrete Methods* 5 (2): 154–61. https://doi.org/10.1137/0605017.

Lasdon, L. S., and R. C. Terjung. 1979. "An Efficient Algorithm for Multi-Item Scheduling." In *Disaggregation: Problems in Manufacturing and Service Organizations*, edited by L. P. Ritzman, L. J. Krajewski, W. L. Berry, Stephen H. Goodman, Stanley T. Hardy, and Lawrence D. Vitt, 157–82. Dordrecht: Springer Netherlands. https://doi.org/10.1007/978-94-015-7636-9_10.

Lawler, E. L., M. G. Luby, and V. V. Vazirani. 1982. "Scheduling Open Shops with Parallel Machines." *Operations Research Letters* 1 (4): 161–64. https://doi.org/10.1016/0167-6377(82)90021-9.

Lee, C.-Y., T. C. E. Cheng, and B. M. T. Lin. 1993. "Minimizing the Makespan in the 3-Machine Assembly-Type Flowshop Scheduling Problem." *Management Science* 39 (5): 616–25.

Lee, C.-Y., and R. Uzsoy. 1999. "Minimizing Makespan on a Single Batch Processing Machine with Dynamic Job Arrivals." *International Journal of Production Research* 37 (1): 219–36. https://doi.org/10.1080/002075499192020.

Lee, C.-Y., R. Uzsoy, and L. A. Martin-Vega. 1992. "Efficient Algorithms for Scheduling Semiconductor Burn-In Operations." *Operations Research* 40 (4): 764–75.

Lee, W.-C. 2014. "Single-Machine Scheduling with Past-Sequence-Dependent Setup Times and General Effects of Deterioration and Learning." *Optimization Letters* 8 (1): 135–44. https://doi.org/10.1007/s11590-012-0481-9.

Lee, Y. F., Z. B. Jiang, and H. R. Liu. 2009. "Multiple-Objective Scheduling and Real-Time Dispatching for the Semiconductor Manufacturing System." *Computers & Operations Research* 36 (3): 866–84. https://doi.org/10.1016/j.cor.2007.11.006.

Lee, Y. H., and M. Pinedo. 1997. "Scheduling Jobs on Parallel Machines with Sequence-Dependent Setup Times." *European Journal of*

Operational Research 100 (3): 464–74. https://doi.org/10.1016/S0377-2217(95)00376-2.

Leitão, P., V. Mařík, and P. Vrba. 2013. "Past, Present, and Future of Industrial Agent Applications." *IEEE Transactions on Industrial Informatics* 9 (4): 2360–72. https://doi.org/10.1109/TII.2012.2222034.

Lenstra, J. K., A. H. G. Rinnooy Kan, and P. Brucker. 1977. "Complexity of Machine Scheduling Problems." In *Annals of Discrete Mathematics*, edited by P. L. Hammer, E. L. Johnson, B. H. Korte, and G. L. Nemhauser, 1:343–62. Studies in Integer Programming. Elsevier. https://doi.org/10.1016/S0167-5060(08)70743-X.

Leon, V. J., and B. A. Peters. 1996. "Replanning and Analysis of Partial Setup Strategies in Printed Circuit Board Assembly Systems." *International Journal of Flexible Manufacturing Systems* 8 (4): 389–411. https://doi.org/10.1007/BF00170019.

Li, C.-L., and C.-Y. Lee. 1997. "Scheduling with Agreeable Release Times and Due Dates on a Batch Processing Machine." *European Journal of Operational Research* 96 (3): 564–69. https://doi.org/10.1016/0377-2217(95)00332-0.

Li, M., F. Yang, R. Uzsoy, and Jie Xu. 2016. "A Metamodel-Based Monte Carlo Simulation Approach for Responsive Production Planning of Manufacturing Systems." *Journal of Manufacturing Systems* 38 (January): 114–33. https://doi.org/10.1016/j.jmsy.2015.11.004.

Li, S. 1997. "A Hybrid Two-Stage Flowshop with Part Family, Batch Production, Major and Minor Set-Ups." *European Journal of Operational Research* 102 (1): 142–56. https://doi.org/10.1016/S0377-2217(96)00213-5.

Li, S., G. Li, and S. Zhang. 2005. "Minimizing Makespan with Release Times on Identical Parallel Batching Machines." *Discrete Applied Mathematics* 148 (1): 127–34. https://doi.org/10.1016/j.dam.2004.11.004.

Li, Shisheng, T. C. E. Cheng, C. T. Ng, and Jinjiang Yuan. 2018. "Two-Agent Scheduling on a Single Sequential and Compatible Batching Machine." *Naval Research Logistics (NRL)* 64 (8): 628–41. https://doi.org/10.1002/nav.21779.

Li, Y., and G. Hu. 2017. "Shop Floor Lot-Sizing and Scheduling with a Two-Stage Stochastic Programming Model Considering Uncertain Demand and Workforce Efficiency." *Computers & Industrial Engineering* 111 (September): 263–71. https://doi.org/10.1016/j.cie.2017.07.014.

Li, Z., and M. G. Ierapetritou. 2010. “Rolling Horizon Based Planning and Scheduling Integration with Production Capacity Consideration.” *Chemical Engineering Science* 65 (22): 5887–5900. https://doi.org/10.1016/j.ces.2010.08.010.

Liaw, C.-F., C.-Y. Cheng, and M. Chen. 2005. “Scheduling Two-Machine No-Wait Open Shops to Minimize Makespan.” *Computers & Operations Research* 32 (4): 901–17. https://doi.org/10.1016/j.cor.2003.09.005.

Lin, B. M. T., and T. C. E. Cheng. 2001. “Batch Scheduling in the No-Wait Two-Machine Flowshop to Minimize the Makespan.” *Computers & Operations Research* 28 (7): 613–24. https://doi.org/10.1016/S0305-0548(99)00138-0.

Lin, B. M. T., T. C. E. Cheng, and A. S. C. Chou. 2007. “Scheduling in an Assembly-Type Production Chain with Batch Transfer.” *Omega* 35 (2): 143–51. https://doi.org/10.1016/j.omega.2005.04.004.

Lin, H.-T., and C.-J. Liao. 2003. “A Case Study in a Two-Stage Hybrid Flow Shop with Setup Time and Dedicated Machines.” *International Journal of Production Economics* 86 (2): 133–43. https://doi.org/10.1016/S0925-5273(03)00011-2.

Lin, J., and Q. Long. 2011. “Development of a Multi-Agent-Based Distributed Simulation Platform for Semiconductor Manufacturing.” *Expert Systems with Applications* 38 (5): 5231–39. https://doi.org/10.1016/j.eswa.2010.10.035.

Lin, James T., and Chien-Ming Chen. 2015. “Simulation Optimization Approach for Hybrid Flow Shop Scheduling Problem in Semiconductor Back-End Manufacturing.” *Simulation Modelling Practice and Theory* 51 (February): 100–114. https://doi.org/10.1016/j.simpat.2014.10.008.

Lin, Y.-H., J.-R. Shie, and C.-H. Tsai. 2009. “Using an Artificial Neural Network Prediction Model to Optimize Work-in-Process Inventory Level for Wafer Fabrication.” *Expert Systems with Applications* 36 (2, Part 2): 3421–27. https://doi.org/10.1016/j.eswa.2008.02.009.

Linn, R., and W. Zhang. 1999. “Hybrid Flow Shop Scheduling: A Survey.” *Computers & Industrial Engineering*, Proceedings of the 24th international conference on computers and industrial engineering, 37 (1): 57–61. https://doi.org/10.1016/S0360-8352(99)00023-6.

Little, J. D. C., and S. C. Graves. 2008. “Little’s Law.” In *Building Intuition: Insights From Basic Operations Management Models and Principles*, edited by D. Chhajed and T. J. Lowe, 81–100. International Series in Operations Research & Management Science. Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-73699-0_5.

Liu, C.-Y. 1996. “Scheduling Flexible Flow Shops with Sequence-Dependent Setup Effect.” In , 35:1757–62. Kobe, Japan.

Liu, C.-Y., and S.-C. Chang. 2000. “Scheduling Flexible Flow Shops with Sequence-Dependent Setup Effects.” *IEEE Transactions on Robotics and Automation* 16 (4): 408–19. https://doi.org/10.1109/70.864235.

Liu, H., Z. Jiang, and R. Y. K. Fung. 2009. “Performance Modeling, Real-Time Dispatching and Simulation of Wafer Fabrication Systems Using Timed Extended Object-Oriented Petri Nets.” *Computers & Industrial Engineering* 56 (1): 121–37. https://doi.org/10.1016/j.cie.2008.04.006.

Liu, J. 2008. “Single-Job Lot Streaming in M−1 Two-Stage Hybrid Flowshops.” *European Journal of Operational Research* 187 (3): 1171–83. https://doi.org/10.1016/j.ejor.2006.06.066.

Liu, L. L., C. T. Ng, and T. C. E. Cheng. 2007. “Scheduling Jobs with Agreeable Processing Times and Due Dates on a Single Batch Processing Machine.” *Theoretical Computer Science* 374 (1): 159–69. https://doi.org/10.1016/j.tcs.2006.12.039.

—. 2009. “Bicriterion Scheduling with Equal Processing Times on a Batch Processing Machine.” *Computers & Operations Research*, Part Special Issue: Operations Research Approaches for Disaster Recovery Planning, 36 (1): 110–18. https://doi.org/10.1016/j.cor.2007.07.007.

—. 2010. “On Scheduling Unbounded Batch Processing Machine(s).” *Computers & Industrial Engineering* 58 (4): 814–17. https://doi.org/10.1016/j.cie.2010.02.003.

Liu, Z., and T.C. E. Cheng. 2004. “Minimizing Total Completion Time Subject to Job Release Dates and Preemption Penalties.” *Journal of Scheduling* 7 (4): 313–27. https://doi.org/10.1023/B:JOSH.0000031424.35504.c4.

Lopez de Haro, S., S. B. Gershwin, and D. B. Rosenfield. 2009. “Schedule Evaluation in Unstable Manufacturing Environments.” *International Journal of Production Economics*, Modelling and Control of Productive Systems: Concepts and Applications, 121 (1): 183–94. https://doi.org/10.1016/j.ijpe.2009.05.004.

Lou, P., Q. Liu, Z. Zhou, H. Wang, and S. X. Sun. 2012. “Multi-Agent-Based Proactive–Reactive Scheduling for a Job Shop.” *The International Journal of Advanced Manufacturing Technology* 59 (1): 311–24. https://doi.org/10.1007/s00170-011-3482-4.

Luo, Hao, G. Q. Huang, Y. Zhang, Q. Dai, and X. Chen. 2009. “Two-Stage Hybrid Batching Flowshop Scheduling with Blocking and Machine Availability Constraints Using Genetic Algorithm.” *Robotics and Computer-Integrated Manufacturing*, 18th International Conference on

Flexible Automation and Intelligent Manufacturing, 25 (6): 962–71. https://doi.org/10.1016/j.rcim.2009.06.001.

Lushchakova, I. N., and V. A. Strusevich. 2010. "Scheduling Incompatible Tasks on Two Machines." *European Journal of Operational Research* 200 (2): 334–46. https://doi.org/10.1016/j.ejor.2009.01.029.

Mahdieh, M., M. Bijari, and A. Clark. 2011. "Simultaneous Lot Sizing and Scheduling in a Flexible Flow Line." *Journal of Industrial and Systems Engineering* 5 (2): 107–19.

Maimon, O. Z., E. M. Dar-El, and T. F. Carmon. 1993. "Set-up Saving Schemes for Printed Circuit Boards Assembly." *European Journal of Operational Research* 70 (2): 177–90. https://doi.org/10.1016/0377-2217(93)90037-N.

Mandal, C. A., P. P. Chakrabarti, and S. Ghose. 1998. "Complexity of Fragmentable Object Bin Packing and an Application." *Computers & Mathematics with Applications* 35 (11): 91–97. https://doi.org/10.1016/S0898-1221(98)00087-X.

Manne, A. S. 1958. "Programming of Economic Lot Sizes." *Management Science* 4 (2): 115–35. https://doi.org/10.1287/mnsc.4.2.115.

Mason, S.J., and J.-S. Chen. 2010. "Scheduling Multiple Orders per Job in a Single Machine to Minimize Total Completion Time." *European Journal of Operational Research* 207 (1): 70–77. https://doi.org/10.1016/j.ejor.2010.03.034.

Masuda, T., H. Ishii, and T. Nishida. 1985. "The Mixed Shop Scheduling Problem." *Discrete Applied Mathematics* 11 (2): 175–86. https://doi.org/10.1016/S0166-218X(85)80007-X.

Mathirajan, M., and A.I. Sivakumar. 2006. "A Literature Review, Classification and Simple Meta-Analysis on Scheduling of Batch Processors in Semiconductor." *The International Journal of Advanced Manufacturing Technology* 29 (9): 990–1001. https://doi.org/10.1007/s00170-005-2585-1.

Maxwell, W. L. 1964. "The Scheduling of Economic Lot Sizes." *Naval Research Logistics Quarterly* 11 (2): 89–124. https://doi.org/10.1002/nav.3800110202.

McAuley, J. 1972. "Machine Grouping for Efficient Production." *Production Engineer* 51 (2): 53–57. https://doi.org/10.1049/tpe.1972.0006.

McCormick, W. T., P. J. Schweitzer, and Thomas W. White. 1972. "Problem Decomposition and Data Reorganization by a Clustering Technique." *Operations Research* 20 (5): 993–1009.

McKay, K. N., and V. C. S. Wiers. 2006. "The Human Factor in Planning and Scheduling." In *Handbook of Production Scheduling*, edited by J.

W. Herrmann, 23–57. International Series in Operations Research & Management Science. Boston, MA: Springer US. https://doi.org/10.1007/0-387-33117-4_2.

Mejabi, O., and Gary S. Wasserman. 1992. "Basic Concepts of JIT Modelling." *International Journal of Production Research* 30 (1): 141–49. https://doi.org/10.1080/00207549208942883.

Menakerman, N., and R. Rom. 2001. "Bin Packing with Item Fragmentation." In *Algorithms and Data Structures*, edited by . Dehne, J.-R. Sack, and R. Tamassia, 313–24. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-44634-6_29.

Meyr, H. 2000. "Simultaneous Lotsizing and Scheduling by Combining Local Search with Dual Reoptimization." *European Journal of Operational Research* 120 (2): 311–26. https://doi.org/10.1016/S0377-2217(99)00159-9.

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd ed. Berlin Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-662-03315-9.

Miller, D. M., H.-C. Chen, J. Matson, and Q. Liu. 1999. "A Hybrid Genetic Algorithm for the Single Machine Scheduling Problem." *Journal of Heuristics* 5 (4): 437–54. https://doi.org/10.1023/A:1009684406579.

Minella, G., R. Ruiz, and M. Ciavotta. 2008. "A Review and Evaluation of Multiobjective Algorithms for the Flowshop Scheduling Problem." *INFORMS Journal on Computing* 20 (3): 451–71. https://doi.org/10.1287/ijoc.1070.0258.

Mistry, P., and M. Desai. 2015. "Set Up Change Time Optimization Using Single Minute Exchange of Die (SMED) Methodology." *International Journal of Research in Electronics and Communication Technology* 2 (1): 6.

Mitrofanov, S. P. 1966. *Scientific Principles of Group Technology*. Yorkshire, UK: National Lending Library.

Mohammadi, M., S. M. T. Fatemi Ghomi, B. Karimi, and S. A. Torabi. 2010. "Rolling-Horizon and Fix-and-Relax Heuristics for the Multi-Product Multi-Level Capacitated Lotsizing Problem with Sequence-Dependent Setups." *Journal of Intelligent Manufacturing* 21 (4): 501–10. https://doi.org/10.1007/s10845-008-0207-0.

Mohammadi, M., and N. Jafari. 2011. "A New Mathematical Model for Integrating Lot Sizing, Loading, and Scheduling Decisions in Flexible Flow Shops." *The International Journal of Advanced Manufacturing Technology* 55 (5): 709–21. https://doi.org/10.1007/s00170-010-3101-9.

Mönch, L., and R. Drießel. 2005. "A Distributed Shifting Bottleneck Heuristic for Complex Job Shops." *Computers & Industrial Engineering* 49 (3): 363–80. https://doi.org/10.1016/j.cie.2005.06.004.

Mönch, L., J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose. 2011. "A Survey of Problems, Solution Techniques, and Future Challenges in Scheduling Semiconductor Manufacturing Operations." *Journal of Scheduling* 14 (6): 583–99. https://doi.org/10.1007/s10951-010-0222-9.

Mönch, L., M. Stehli, and . Zimmermann. 2003. "FABMAS: An Agent-Based System for Production Control of Semiconductor Manufacturing Processes." In *Holonic and Multi-Agent Systems for Manufacturing*, edited by V. Mařík, D. McFarlane, and P. Valckenaers, 258–67. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-45185-3_24.

Mönch, L., and R. Unbehaun. 2007. "Decomposition Heuristics for Minimizing Earliness–Tardiness on Parallel Burn-in Ovens with a Common Due Date." *Computers & Operations Research* 34 (11): 3380–96. https://doi.org/10.1016/j.cor.2006.02.003.

Moon, I., B. C. Giri, and K. Choi. 2002. "Economic Lot Scheduling Problem with Imperfect Production Processes and Setup Times." *Journal of the Operational Research Society* 53 (6): 620–29. https://doi.org/10.1057/palgrave.jors.2601350.

Moore, J. M. 1968. "An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs." *Management Science* 15 (1): 102–9.

Mor, B., and G. Mosheiov. 2014. "Batch Scheduling of Identical Jobs with Controllable Processing Times." *Computers & Operations Research* 41 (January): 115–24. https://doi.org/10.1016/j.cor.2013.08.007.

Mortezaei, N., and N. Zulkifli. 2013. "Integration of Lot Sizing and Flow Shop Scheduling with Lot Streaming." Research Article. Journal of Applied Mathematics. Hindawi. November 20, 2013. https://doi.org/10.1155/2013/216595.

Mosheiov, G. 1994. "Scheduling Jobs under Simple Linear Deterioration." *Computers & Operations Research* 21 (6): 653–59. https://doi.org/10.1016/0305-0548(94)90080-9.

Mosheiov, G., and D. Oron. 2008. "Open-Shop Batch Scheduling with Identical Jobs." *European Journal of Operational Research* 187 (3): 1282–92. https://doi.org/10.1016/j.ejor.2006.03.068.

Mosier, C. T., J. Yelle, and G. Walker. 1997. "Survey of Similarity Coefficient Based Methods as Applied to the Group Technology Configuration Problem." *Omega* 25 (1): 65–79. https://doi.org/10.1016/S0305-0483(96)00045-X.

Motwani, J. 2003. "A Business Process Change Framework for Examining Lean Manufacturing: A Case Study." *Industrial Management & Data Systems* 103 (5): 339–46. https://doi.org/10.1108/02635570310477398.

Mukherjee, N. J., and S. C. Sarin. 2014. "Lot Streaming in the Presence of Learning." *Int. J. Planning and Scheduling* 2 (1): 40–52.

Mukherjee, N. J., S. C. Sarin, and S. Singh. 2017. "Lot Streaming in the Presence of Learning in Sublot-Attached Setup Times and Processing Times." *International Journal of Production Research* 55 (6): 1623–39. https://doi.org/10.1080/00207543.2016.1200760.

Muñoz, E., E. Capón-García, M. Moreno-Benito, A. Espuña, and L. Puigjaner. 2011. "Scheduling and Control Decision-Making under an Integrated Information Environment." *Computers & Chemical Engineering*, Selected Papers from ESCAPE-20 (European Symposium of Computer Aided Process Engineering - 20), 6-9 June 2010, Ischia, Italy, 35 (5): 774–86. https://doi.org/10.1016/j.compchemeng.2011.01.025.

Murugan, M., and V. Selladurai. 2011. "Formation of Machine Cells/ Part Families in Cellular Manufacturing Systems Using an ART-Modified Single Linkage Clustering Approach ± A Comparative Study." *Jordan Journal of Mechanical and Industrial Engineering* 5 (3): 199–212.

Mutingi, M., and C. Mbohwa. 2016. "An Exploratory Study of Computational Challenges in Industrial Grouping Problems." In *Proceedings of The World Congress on Engineering and Computer Science 2016*, 502–7. San Francisco, USA. http://www.iaeng.org/publication/WCECS2016/WCECS2016_pp502-507.pdf.

Naderi, B., and M. Zandieh. 2014. "Modeling and Scheduling No-Wait Open Shop Problems." *International Journal of Production Economics* 158 (December): 256–66. https://doi.org/10.1016/j.ijpe.2014.06.011.

Naderi, B., M. Zandieh, A. Khaleghi Ghoshe Balagh, and V. Roshanaei. 2009. "An Improved Simulated Annealing for Hybrid Flowshops with Sequence-Dependent Setup and Transportation Times to Minimize Total Completion Time and Total Tardiness." *Expert Systems with Applications* 36 (6): 9625–33. https://doi.org/10.1016/j.eswa.2008.09.063.

Nakajima, S., and N. Bodek. 1988. *Introduction to TPM: Total Productive Maintenance*. Cambridge, Mass.

Narahari, Y., and L. M. Khan. 1996. "Performance Analysis of Scheduling Policies in Re-Entrant Manufacturing Systems." *Computers & Operations Research* 23 (1): 37–51. https://doi.org/10.1016/0305-0548(95)00003-5.

Narayan, L. K., and K. R. Mallikarjuna. 2008. *Computer Aided Design and Manufacturing*. India: PHI Learning.

Nawaz, M., E. E. Enscore, and I. Ham. 1983. "A Heuristic Algorithm for the M-Machine, n-Job Flow-Shop Sequencing Problem." *Omega* 11 (1): 91–95. https://doi.org/10.1016/0305-0483(83)90088-9.

Nejati, M., I. Mahdavi, R. Hassanzadeh, N. Mahdavi-Amiri, and M. Mojarad. 2014. "Multi-Job Lot Streaming to Minimize the Weighted Completion Time in a Hybrid Flow Shop Scheduling Problem with Work Shift Constraint." *The International Journal of Advanced Manufacturing Technology* 70 (1): 501–14. https://doi.org/10.1007/s00170-013-5265-6.

Nguyen, V., and H. P. Bao. 2016. "An Efficient Solution to the Mixed Shop Scheduling Problem Using a Modified Genetic Algorithm." *Procedia Computer Science*, Complex Adaptive Systems Los Angeles, CA November 2-4, 2016, 95 (January): 475–82. https://doi.org/10.1016/j.procs.2016.09.324.

Ni, Y., and F. Werner. 2017. "Material Handling Tools for a Discrete Manufacturing System: A Comparison of Optimization and Simulation." In *2017 International Conference on Circuits, System and Simulation (ICCSS)*, 97–102. London / UK. https://doi.org/10.1109/CIRSYSSIM.2017.8023190.

Nobil, A. H, E. Nobil, and L. E. Cárdenas-Barrón. 2017. "Some Observations to: Lot Sizing with Non-Zero Setup Times for Rework." *International Journal of Applied and Computational Mathematics* 3 (1): 1511–17. https://doi.org/10.1007/s40819-017-0340-6.

Novas, J. M., and G. P. Henning. 2010. "Reactive Scheduling Framework Based on Domain Knowledge and Constraint Programming." *Computers & Chemical Engineering*, 10th International Symposium on Process Systems Engineering, Salvador, Bahia, Brasil, 16-20 August 2009, 34 (12): 2129–48. https://doi.org/10.1016/j.compchemeng.2010.07.011.

Offodile, O. F. 1984. "Design and Analysis of a Coding and Classification System for a Systematic Interactive Computer-Aided Robot Selection Procedure (CARSP),." PhD thesis, Texas Tech University.

Offodile, O. F., and J. Grznar. 1997. "Part Family Formation for Variety Reduction in Flexible Manufacturing Systems." *International Journal of Operations & Production Management* 17 (3): 291–304. https://doi.org/10.1108/01443579710159914.

Offodile, O. F., A. Mehrez, and J. Grznar. 1994. "Cellular Manufacturing: A Taxonomic Review Framework." *Journal of Manufacturing Systems* 13 (3): 196–220. https://doi.org/10.1016/0278-6125(94)90005-1.

Olaitan, O., Q. Yu, and E. Alfnes. 2017. "Work In Process Control for a High Product Mix Manufacturing System." *Procedia CIRP*, Manufacturing Systems 4.0 – Proceedings of the 50th CIRP Conference on Manufacturing Systems, 63 (January): 277–82. https://doi.org/10.1016/j.procir.2017.03.352.

Opitz, H., W. Eversheim, and H. P. Wiendahl. 1969. "Workpiece Classification and Its Industrial Application." *International Journal of Machine Tool Design and Research* 9 (1): 39–50. https://doi.org/10.1016/0020-7357(69)90027-4.

Opitz, H., and H.-P. Wiendahl. 1971. "Group Technology and Manufacturing Systems for Small and Medium Quantity Production." *International Journal of Production Research* 9 (1): 181–203. https://doi.org/10.1080/00207547108929870.

Orlicky, Joseph. 1975. *Material Requirements Planning: The New Way of Life in Production and Inventory Management*. New York.

Ouelhadj, D., and S. Petrovic. 2009. "A Survey of Dynamic Scheduling in Manufacturing Systems." *Journal of Scheduling* 12 (4): 417. https://doi.org/10.1007/s10951-008-0090-8.

Oulamara, A. 2007. "Makespan Minimization in a No-Wait Flow Shop Problem with Two Batching Machines." *Computers & Operations Research* 34 (4): 1033–50. https://doi.org/10.1016/j.cor.2005.05.028.

Oulamara, A., G. Finke, and A. Kamgaing Kuiteing. 2009. "Flowshop Scheduling Problem with a Batching Machine and Task Compatibilities." *Computers & Operations Research*, Scheduling for Modern Manufacturing, Logistics, and Supply Chains, 36 (2): 391–401. https://doi.org/10.1016/j.cor.2007.10.006.

Ovacik, I. M., and R. Uzsoy. 1996. "Decomposition Methods for Scheduling Semiconductor Testing Facilities." *International Journal of Flexible Manufacturing Systems* 8 (4): 357–87. https://doi.org/10.1007/BF00170018.

Pan, J. C.-H., J.-S. Chen, and H.-L. Cheng. 2001. "A Heuristic Approach for Single-Machine Scheduling with Due Dates and Class Setups." *Computers & Operations Research* 28 (11): 1111–30. https://doi.org/10.1016/S0305-0548(00)00031-9.

Pan, Q.-K., M. Fatih Tasgetiren, P. N. Suganthan, and T. J. Chua. 2011. "A Discrete Artificial Bee Colony Algorithm for the Lot-Streaming Flow Shop Scheduling Problem." *Information Sciences* 181 (12): 2455–68. https://doi.org/10.1016/j.ins.2009.12.025.

Papadopoulos, H. T., and M. I. Vidalis. 2001. "Minimizing WIP Inventory in Reliable Production Lines." *International Journal of Production Economics* 70 (2): 185–97.

https://doi.org/10.1016/S0925-5273(00)00056-6.

Parveen, Sultana, and Hafiz Ullah. 2010. "REVIEW ON JOB-SHOP AND FLOW-SHOP SCHEDULING USING." *Journal of Mechanical Engineering* 41 (2): 130–46. https://doi.org/10.3329/jme.v41i2.7508.

Patel, shilpan. 1991. "DEVELOPMENT OF A CLASSIFICATION AND CODING SYSTEM FOR COMPUTER AIDED PROCESS PLANNING." Master of Science in Manufacturing Engineering, New Jersey Institute of Technology.

Pei, Jun, Bayi Cheng, Xinbao Liu, Panos M. Pardalos, and Min Kong. 2019. "Single-Machine and Parallel-Machine Serial-Batching Scheduling Problems with Position-Based Learning Effect and Linear Setup Time." *Annals of Operations Research* 272 (1): 217–41. https://doi.org/10.1007/s10479-017-2481-8.

Pei, Jun, Xinbao Liu, Wenjuan Fan, Panos M. Pardalos, Athanasios Migdalas, and Shanlin Yang. 2016. "Scheduling Jobs on a Single Serial-Batching Machine with Dynamic Job Arrivals and Multiple Job Types." *Annals of Mathematics and Artificial Intelligence* 76 (1): 215–28. https://doi.org/10.1007/s10472-015-9449-7.

Pei, Jun, Xinbao Liu, Baoyu Liao, Panos M. Pardalos, and Min Kong. 2018. "Single-Machine Scheduling with Learning Effect and Resource-Dependent Processing Times in the Serial-Batching Production." *Applied Mathematical Modelling* 58 (June): 245–53. https://doi.org/10.1016/j.apm.2017.07.028.

Pei, Jun, Xinbao Liu, Panos M. Pardalos, Wenjuan Fan, and Shanlin Yang. 2015. "Single Machine Serial-Batching Scheduling with Independent Setup Time and Deteriorating Job Processing Times." *Optimization Letters* 9 (1): 91–104. https://doi.org/10.1007/s11590-014-0740-z.

—. 2017. "Scheduling Deteriorating Jobs on a Single Serial-Batching Machine with Multiple Job Types and Sequence-Dependent Setup Times." *Annals of Operations Research* 249 (1): 175–95. https://doi.org/10.1007/s10479-015-1824-6.

Pei, Jun, Xinbao Liu, Panos M. Pardalos, Athanasios Migdalas, and Shanlin Yang. 2017. "Serial-Batching Scheduling with Time-Dependent Setup Time and Effects of Deterioration and Learning on a Single-Machine." *Journal of Global Optimization* 67 (1): 251–62. https://doi.org/10.1007/s10898-015-0320-5.

Perez, Imelda C., John W. Fowler, and W. Matthew Carlyle. 2005. "Minimizing Total Weighted Tardiness on a Single Batch Process Machine with Incompatible Job Families." *Computers & Operations Research* 32 (2): 327–41. https://doi.org/10.1016/S0305-0548(03)00239-9.

Petrov, V. A. 1968. *Flowline group production planning*. London.

Pfund, Michele, John W. Fowler, Amit Gadkari, and Yan Chen. 2008. "Scheduling Jobs on Parallel Machines with Setup Times and Ready Times." *Computers & Industrial Engineering* 54 (4): 764–82. https://doi.org/10.1016/j.cie.2007.08.011.

Pickardt, Christoph W., and Jürgen Branke. 2012. "Setup-Oriented Dispatching Rules – a Survey." *International Journal of Production Research* 50 (20): 5823–42. https://doi.org/10.1080/00207543.2011.629634.

Pinedo, Michael L. 2008. *Scheduling: Theory, Algorithms, and Systems*. 4th ed. New York: Springer-Verlag. https://doi.org/10.1007/978-1-4614-2361-4.

Pinxten, Joost Van, Umar Waqas, Marc Geilen, Twan Basten, and Lou Somers. 2017. "Online Scheduling of 2-Re-Entrant Flexible Manufacturing Systems." *ACM Transactions on Embedded Computing Systems* 16 (5s): 160:1-160:20. https://doi.org/10.1145/3126551.

Pitakaso, Rapeepan, Christian Almeder, Karl F. Doerner, and Richard F. Hartl. 2007. "A MAX-MIN Ant System for Unconstrained Multi-Level Lot-Sizing Problems." *Computers & Operations Research* 34 (9): 2533–52. https://doi.org/10.1016/j.cor.2005.09.022.

Pochet, Yves. 2001. "Mathematical Programming Models and Formulations for Deterministic Production Planning Problems." In *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, edited by Michael Jünger and Denis Naddef, 57–111. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-45586-8_3.

Potts, C. N., and K. R. Baker. 1989. "Flow Shop Scheduling with Lot Streaming." *Operations Research Letters* 8 (6): 297–303. https://doi.org/10.1016/0167-6377(89)90013-8.

Potts, C. N., and M. Y. Kovalyov. 2000. "Scheduling with Batching: A Review." *European Journal of Operational Research* 120 (2): 228–49. https://doi.org/10.1016/S0377-2217(99)00153-8.

Potts, C. N., S. V. Sevast'janov, V. A. Strusevich, L. N. Van Wassenhove, and C. M. Zwaneveld. 1995. "The Two-Stage Assembly Scheduling Problem: Complexity and Approximation." *Operations Research* 43 (2): 346–55. https://doi.org/10.1287/opre.43.2.346.

Potts, C. N., and L. N. Van Wassenhove. 1992. "Integrating Scheduling with Batching and Lot-Sizing: A Review of Algorithms and Complexity." *The Journal of the Operational Research Society* 43 (5): 395–406. https://doi.org/10.2307/2583559.

Price, Wilson, Marc Gravel, and Aaron Luntala Nsakanda. 1994. "A Review of Optimisation Models of Kanban-Based Production Systems." *European Journal of Operational Research* 75 (1): 1–12. https://doi.org/10.1016/0377-2217(94)90182-1.

Ptak, Carol, and Chad Smith. 2016. *Demand Driven Material Requirements Planning*. South Norwalk, Connecticut.

Pullen, R. D. 1976. "A Survey of Cellular Manufacturing Cells." In *The Production Engineer*, 451–54. Rockwall Machine Tool Companiy Ltd.

Qiao, F., Y. Ma, L. Li, and H. Yu. 2013. "A Petri Net and Extended Genetic Algorithm Combined Scheduling Method for Wafer Fabrication." *IEEE Transactions on Automation Science and Engineering* 10 (1): 197–204. https://doi.org/10.1109/TASE.2012.2204049.

Qiao, F., Y. Ma, M. Zhou, and Q. Wu. 2018. "A Novel Rescheduling Method for Dynamic Semiconductor Manufacturing Systems." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50 (5): 1679–89. https://doi.org/10.1109/TSMC.2017.2782009.

Qiu, Robin G. 2005. "Virtual Production Line Based WIP Control for Semiconductor Manufacturing Systems." *International Journal of Production Economics* 95 (2): 165–78. https://doi.org/10.1016/j.ijpe.2003.12.004.

Rajagopalan, R., and J. L. Batra. 1975. "Design of Cellular Production Systems A Graph-Theoretic Approach." *International Journal of Production Research* 13 (6): 567–79. https://doi.org/10.1080/00207547508943029.

Ramezanian, Reza, Mohammad Saidi-Mehrabad, and Parviz Fattahi. 2013. "MIP Formulation and Heuristics for Multi-Stage Capacitated Lot-Sizing and Scheduling Problem with Availability Constraints." *Journal of Manufacturing Systems* 32 (2): 392–401. https://doi.org/10.1016/j.jmsy.2013.01.002.

Raza, Asif S., and Ali Akgunduz. 2008. "A Comparative Study of Heuristic Algorithms on Economic Lot Scheduling Problem." *Computers & Industrial Engineering* 55 (1): 94–109. https://doi.org/10.1016/j.cie.2007.12.004.

Reisman, A., A. Kumar, and J. Motwani. 1997. "Flowshop Scheduling/Sequencing Research: A Statistical Review of the Literature, 1952-1994." *IEEE Transactions on Engineering Management* 44 (3): 316–29. https://doi.org/10.1109/17.618173.

Reiter, Stanley. 1966. "A System for Managing Job-Shop Production." *The Journal of Business* 39 (3): 371–93.

Reklaitis, G. V. 1996. "Overview of Scheduling and Planning of Batch Process Operations." In *Batch Processing Systems Engineering*, edited

by Ginraras V. Reklaitis, Aydin K. Sunol, David W. T. Rippin, and Öner Hortaçsu, 660–705. NATO ASI Series. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-60972-5_27.

Ribas, Imma, Rainer Leisten, and Jose M. Framiñan. 2010. "Review and Classification of Hybrid Flow Shop Scheduling Problems from a Production System and a Solutions Procedure Perspective." *Computers & Operations Research*, Operations Research and Data Mining in Biological Systems, 37 (8): 1439–54. https://doi.org/10.1016/j.cor.2009.11.001.

Ridouard, Frédéric, Pascal Richard, and Patrick Martineau. 2008. "On-Line Scheduling on a Batch Processing Machine with Unbounded Batch Size to Minimize the Makespan." *European Journal of Operational Research* 189 (3): 1327–42. https://doi.org/10.1016/j.ejor.2006.06.080.

Rinnooy Kan, A. H. G. 1976. *Machine Scheduling Problems: Classification, Complexity and Computations*. Springer US. https://doi.org/10.1007/978-1-4613-4383-7.

Rippin, D. W. T. 1993. "Batch Process Systems Engineering: A Retrospective and Prospective Review." *Computers & Chemical Engineering*, European Symposium on Computer Aided Process Engineering—2, 17 (January): S1–13. https://doi.org/10.1016/0098-1354(93)80201-W.

Robinson, Powell, Arunachalam Narayanan, and Funda Sahin. 2009. "Coordinated Deterministic Dynamic Demand Lot-Sizing Problem: A Review of Models and Algorithms." *Omega* 37 (1): 3–15. https://doi.org/10.1016/j.omega.2006.11.004.

Rogers, Jack. 1958. "A Computational Approach to the Economic Lot Scheduling Problem." *Management Science* 4 (3): 264–91.

Rolón, Milagros, and Ernesto Martínez. 2012. "Agent-Based Modeling and Simulation of an Autonomic Manufacturing Execution System." *Computers in Industry* 63 (1): 53–78. https://doi.org/10.1016/j.compind.2011.10.005.

Romero-Parra, R., and L. Burtseva. 2010. "Implementation of Bin Packing Model for Reed Switch Production Planning." *AIP Conference Proceedings* 1247 (1): 403–12. https://doi.org/10.1063/1.3460247.

Rossit, D., F. Tohmé, M. Frutos, J. Bard, and D. Broz. 2016. "A Non-Permutation Flowshop Scheduling Problem with Lot Streaming: A Mathematical Model." *International Journal of Industrial Engineering Computations* 7 (3): 507–16.

Rubin, J. 1967. "Optimal Classification into Groups: An Approach for Solving the Taxonomy Problem." *Journal of Theoretical Biology* 15 (1): 103–44. https://doi.org/10.1016/0022-5193(67)90046-X.

Ruiz, R., and Concepción Maroto. 2006. “A Genetic Algorithm for Hybrid Flowshops with Sequence Dependent Setup Times and Machine Eligibility.” *European Journal of Operational Research* 169 (3): 781–800. https://doi.org/10.1016/j.ejor.2004.06.038.

Ruiz, R., F. S. Şerifoğlu, and T. Urlings. 2008. “Modeling Realistic Hybrid Flexible Flowshop Scheduling Problems.” *Computers & Operations Research* 35 (4): 1151–75. https://doi.org/10.1016/j.cor.2006.07.014.

Ruiz, R., and J. A. Vázquez-Rodríguez. 2010. “The Hybrid Flow Shop Scheduling Problem.” *European Journal of Operational Research* 205 (1): 1–18. https://doi.org/10.1016/j.ejor.2009.09.024.

Sabuncuoglu, I., and S. Goren. 2009. “Hedging Production Schedules against Uncertainty in Manufacturing Environment with a Review of Robustness and Stability Research.” *International Journal of Computer Integrated Manufacturing* 22 (2): 138–57. https://doi.org/10.1080/09511920802209033.

Salomon, M., L. G. Kroon, R. Kuik, and Luk N. Van Wassenhove. 1991. “Some Extensions of the Discrete Lotsizing and Scheduling Problem.” *Management Science* 37 (7): 801–12. https://doi.org/10.1287/mnsc.37.7.801.

Schaller, J. E, J. N. D Gupta, and A. J Vakharia. 2000. “Scheduling a Flowline Manufacturing Cell with Sequence Dependent Family Setup Times.” *European Journal of Operational Research* 125 (2): 324–39. https://doi.org/10.1016/S0377-2217(99)00387-2.

Seifoddini, H. 1988. “Machine Grouping — Expert Systems: Comparison between Single Linkage and Average Linkage Clustering Techniques in Forming Machine Cells.” *Computers & Industrial Engineering* 15 (1): 210–16. https://doi.org/10.1016/0360-8352(88)90088-5.

Seifoddini, H., and P. M. Wolfe. 1986. “Application of the Similarity Coefficient Method in Group Technology.” *IIE Transactions* 18 (3): 271–77. https://doi.org/10.1080/07408178608974704.

—. 1987. “Selection of a Threshold Value Based on Material Handling Cost in Machine-Component Grouping.” *IIE Transactions* 19 (3): 266–70. https://doi.org/10.1080/07408178708975395.

Selim, H. M., R. G. Askin, and A. J. Vakharia. 1998. “Cell Formation in Group Technology: Review, Evaluation and Directions for Future Research.” *Computers & Industrial Engineering*, Cellular manufacturing systems: Design, Analysis and Implementation, 34 (1): 3–20. https://doi.org/10.1016/S0360-8352(97)00147-2.

Selvarajah, E., and G. Steiner. 2006. "Batch Scheduling in a Two-Level Supply Chain—a Focus on the Supplier." *European Journal of Operational Research* 173 (1): 226–40. https://doi.org/10.1016/j.ejor.2004.12.007.

Şen, A., and Ö. S. Benli. 1998. "Lot Streaming in Open Shops." *Operations Research Letters* 23 (3): 135–42. https://doi.org/10.1016/S0167-6377(98)00026-1.

Sendil Kumar, C., and R. Panneerselvam. 2007. "Literature Review of JIT-KANBAN System." *The International Journal of Advanced Manufacturing Technology* 32 (3): 393–408. https://doi.org/10.1007/s00170-005-0340-2.

Senties, O. B., C. Azzaro-Pantel, L. Pibouleau, and S. Domenech. 2010. "Multiobjective Scheduling for Semiconductor Manufacturing Plants." *Computers & Chemical Engineering* 34 (4): 555–66. https://doi.org/10.1016/j.compchemeng.2010.01.010.

Shabtay, D., and G. Steiner. 2007. "A Survey of Scheduling with Controllable Processing Times." *Discrete Applied Mathematics* 155 (13): 1643–66. https://doi.org/10.1016/j.dam.2007.02.003.

Shachnai, H., and T. Tamir. 2004. "Tight Bounds for Online Class-Constrained Packing." *Theoretical Computer Science*, Latin American Theoretical Informatics, 321 (1): 103–23. https://doi.org/10.1016/j.tcs.2003.05.006.

Shafer, S. M., and D. F. Rogers. 1993a. "Similarity and Distance Measures for Cellular Manufacturing. Part I. A Survey." *International Journal of Production Research* 31 (5): 1133–42. https://doi.org/10.1080/00207549308956779.

—. 1993b. "Similarity and Distance Measures for Cellular Manufacturing. Part II. An Extension and Comparison." *International Journal of Production Research* 31 (6): 1315–26. https://doi.org/10.1080/00207549308956793.

Shahvari, O., and R. Logendran. 2016. "Hybrid Flow Shop Batching and Scheduling with a Bi-Criteria Objective." *International Journal of Production Economics* 179 (September): 239–58. https://doi.org/10.1016/j.ijpe.2016.06.005.

Shakhlevich, N. V., Y. N. Sotskov, and F. Werner. 2000. "Complexity of Mixed Shop Scheduling Problems: A Survey." *European Journal of Operational Research* 120 (2): 343–51. https://doi.org/10.1016/S0377-2217(99)00161-7.

Shakhlevich, N. V., and V. A. Strusevich. 2006. "Single Machine Scheduling with Controllable Release and Processing Parameters."

Discrete Applied Mathematics, International Symposium on Combinatorial Optimization CO'02, 154 (15): 2178–99. https://doi.org/10.1016/j.dam.2005.04.014.

Sharda, B., and A. Banerjee. 2013. "Robust Manufacturing System Design Using Multi Objective Genetic Algorithms, Petri Nets and Bayesian Uncertainty Representation." *Journal of Manufacturing Systems* 32 (2): 315–24. https://doi.org/10.1016/j.jmsy.2013.01.001.

Sharma, P., and A. Jain. 2015. "A Review on Job Shop Scheduling with Setup Times:" *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, January. https://doi.org/10.1177/0954405414560617.

Shashikumar, S., R. D. Raut, V. S. Narwane, B. B. Gardas, B. E. Narkhede, and A. Awasthi. 2019. "A Novel Approach to Determine the Cell Formation Using Heuristics Approach." *OPSEARCH* 56 (3): 628–56. https://doi.org/10.1007/s12597-019-00381-4.

Shen, L., and U. Buscher. 2012. "Solving the Serial Batching Problem in Job Shop Manufacturing Systems." *European Journal of Operational Research* 221 (1): 14–26. https://doi.org/10.1016/j.ejor.2012.03.001.

Shen, L., J. N. D. Gupta, and U. Buscher. 2014. "Flow Shop Batching and Scheduling with Sequence-Dependent Setup Times." *Journal of Scheduling* 17 (4): 353–70. https://doi.org/10.1007/s10951-014-0369-x.

Shingo, S. 1985. *A Revolution in Manufacturing: The SMED System*. CRC Press.

Silva, I. B. da, and M. Godinho Filho. 2019. "Single-Minute Exchange of Die (SMED): A State-of-the-Art Literature Review." *The International Journal of Advanced Manufacturing Technology* 102 (9): 4289–4307. https://doi.org/10.1007/s00170-019-03484-w.

Silver, E. A. 1973. "A Heuristic for Selecting Lot Size Quantities for the Case of a Deterministic Time-Varying Demand Rate and Discrete Opportunities for Replenishment." *Prod. Inventory Manage.* 2: 64–74.

Sivakumar, B., and G. Arivarignan. 2009. "A Stochastic Inventory System with Postponed Demands." *Performance Evaluation* 66 (1): 47–58. https://doi.org/10.1016/j.peva.2008.08.001.

Slagle, J. R., C. L. Chang, and S. R. Heller. 1975. "A Clustering and Data-Reorganizing Algorithm." *IEEE Transactions on Systems, Man, and Cybernetics* SMC-5 (1): 125–28. https://doi.org/10.1109/TSMC.1975.5409163.

Smith, W. E. 1956. "Various Optimizers for Single-Stage Production." *Naval Research Logistics Quarterly* 3 (1–2): 59–66. https://doi.org/10.1002/nav.3800030106.

Sobeyko, O., and L. Mönch. 2015. "Grouping Genetic Algorithms for Solving Single Machine Multiple Orders per Job Scheduling Problems." *Annals of Operations Research* 235 (1): 709–39. https://doi.org/10.1007/s10479-015-1976-4.

Solaja, V., and S. M. Urosevic. 1969. "Optimization of Group Technology Lines by Methods Developed in the Institute for Machine Tools and Tooling (IAMA) in Belgrade." In *Proceeding of International Seminar*, 157–76. Turin Italy: Bulbridge, J.L.

Soleimani, M., and S. Majid. 2012. "Machine Maintenance in Open Shop Problems with Genetic Algorithm." *Journal of Mechatronics, Electrical and Computer Technology* 1 (5): 181–97.

Song, H., S. Yi, and H. Shen. 2016. "A Study of Job Shop Standard Setup Time Quota." *Production Engineering* 10 (2): 185–96. https://doi.org/10.1007/s11740-015-0649-0.

Sotskov, Y. N., T. Tautenhahn, and F. Werner. 1996. "Heuristics for Permutation Flow Shop Scheduling with Batch Setup Times." *Operations-Research-Spektrum* 18 (2): 67–80. https://doi.org/10.1007/BF01539731.

Spearman, M. L., D. L. Woodruff, and W. J. Hopp. 1990. "CONWIP: A Pull Alternative to Kanban." *International Journal of Production Research* 28 (5): 879–94. https://doi.org/10.1080/00207549008942761.

Srinivasan, G., T. T. Narendran, and B. Mahadevan. 1990. "An Assignment Model for the Part-Families Problem in Group Technology." *International Journal of Production Research* 28 (1): 145–52. https://doi.org/10.1080/00207549008942689.

Stadtler, H. 2011. "Multi-Level Single Machine Lot-Sizing and Scheduling with Zero Lead Times." *European Journal of Operational Research* 209 (3): 241–52. https://doi.org/10.1016/j.ejor.2010.09.022.

Staggemeier, A. T., and A. Clark. 2001. "A Survey of Lot-Sizing and Scheduling Models," November. https://uwe-repository.worktribe.com/output/1092068/a-survey-of-lot-sizing-and-scheduling-models.

Stecke, K. E., and I. Kim. 1988. "A Study of FMS Part Type Selection Approaches for Short-Term Production Planning." *International Journal of Flexible Manufacturing Systems* 1 (1): 7–29. https://doi.org/10.1007/BF00713157.

Stefansdottir, B., M. Grunow, and R. Akkerman. 2017. "Classifying and Modeling Setups and Cleanings in Lot Sizing and Scheduling." *European Journal of Operational Research* 261 (3): 849–65. https://doi.org/10.1016/j.ejor.2017.03.023.

Strusevich, V. A. 1999. "A Heuristic for the Two-Machine Open-Shop Scheduling Problem with Transportation Times." *Discrete Applied Mathematics* 93 (2): 287–304. https://doi.org/10.1016/S0166-218X(99)00115-8.

Su, L.-H. 2003. "A Hybrid Two-Stage Flowshop with Limited Waiting Time Constraints." *Computers & Industrial Engineering* 44 (3): 409–24. https://doi.org/10.1016/S0360-8352(02)00216-4.

Su, L.-H., and C.-Y. Lien. 2009. "Scheduling Parallel Machines with Resource-Dependent Processing Times." *International Journal of Production Economics* 117 (2): 256–66. https://doi.org/10.1016/j.ijpe.2008.10.014.

Sun, Y., C. Zhang, L. Gao, and X. Wang. 2011. "Multi-Objective Optimization Algorithms for Flow Shop Scheduling Problem: A Review and Prospects." *The International Journal of Advanced Manufacturing Technology* 55 (5): 723–39. https://doi.org/10.1007/s00170-010-3094-4.

Sung, C. S., Y. I. Choung, J. M. Hong, and Y. H. Kim. 2002. "Minimizing Makespan on a Single Burn-in Oven with Job Families and Dynamic Job Arrivals." *Computers & Operations Research* 29 (8): 995–1007. https://doi.org/10.1016/S0305-0548(00)00098-8.

Suppiah, Y., and M. K. Omar. 2014. "A Hybrid Tabu Search for Batching and Sequencing Decisions in a Single Machine Environment." *Computers & Industrial Engineering* 78 (December): 135–47. https://doi.org/10.1016/j.cie.2014.10.010.

Suwa, H., and H. Sandoh. 2013. *Online Scheduling in Manufacturing: A Cumulative Delay Approach*. London: Springer-Verlag. https://doi.org/10.1007/978-1-4471-4561-5.

Tabucanon, M. T., and R. Ojha. 1987. "ICRM—a Heuristic Approach for Intercell Flow Reduction in Cellular Manufacturing." *Material Flow* 4: 189–97.

Taft, E. W. 1918. "The Most Economical Production Lot." *The Iron Age* 101: 1410–12.

Tang, C. S., and E. V. Denardo. 1988. "Models Arising from a Flexible Manufacturing Machine, Part I: Minimization of the Number of Tool Switches." *Operations Research* 36 (5): 767–77. https://doi.org/10.1287/opre.36.5.767.

Tang, L., and P. Liu. 2009. "Two-Machine Flowshop Scheduling Problems Involving a Batching Machine with Transportation or Deterioration Consideration." *Applied Mathematical Modelling* 33 (2): 1187–99. https://doi.org/10.1016/j.apm.2008.01.013.

Tang, L., P. B. Luh, J. Liu, and L. Fang. 2002. "Steel-Making Process Scheduling Using Lagrangian Relaxation." *International Journal of Production Research* 40 (1): 55–70. https://doi.org/10.1080/00207540110073000.

Tang, L., X. Zhao, J. Liu, and J. Y. -T. Leung. 2017. "Competitive Two-Agent Scheduling with Deteriorating Jobs on a Single Parallel-Batching Machine." *European Journal of Operational Research* 263 (2): 401–11. https://doi.org/10.1016/j.ejor.2017.05.019.

Tatikonda, M. V., and U. Wemmerövr. 1992. "Adoption and Implementation of Group Technology Classification and Coding Systems: Insights from Seven Case Studies." *International Journal of Production Research* 30 (9): 2087–2110. https://doi.org/10.1080/00207549208948139.

Toledo, F. M. B., and V. A. Armentano. 2006. "A Lagrangian-Based Heuristic for the Capacitated Lot-Sizing Problem in Parallel Machines." *European Journal of Operational Research* 175 (2): 1070–83. https://doi.org/10.1016/j.ejor.2005.06.029.

Trietsch, Dan, and K. R. Baker. 1993. "Basic Techniques for Lot Streaming." *Operations Research* 41 (6): 1065–76. https://doi.org/10.1287/opre.41.6.1065.

Trigeiro, W. W. 1989. "A Simple Heuristic for Lot Sizing with Setup Times." *Decision Sciences* 20 (2): 294–303. https://doi.org/10.1111/j.1540-5915.1989.tb01879.x.

Trigeiro, W. W., L. J. Thomas, and J. O. McClain. 1989. "Capacitated Lot Sizing with Setup Times." *Management Science* 35 (3): 353–66. https://doi.org/10.1287/mnsc.35.3.353.

Trindade, R. S., O. C. de Araújo Bassi, M. H. Costa Fampa, and F. M. Müller. 2018. "Modelling and Symmetry Breaking in Scheduling Problems on Batch Processing Machines." *International Journal of Production Research* 56 (22): 7031–48. https://doi.org/10.1080/00207543.2018.1424371.

Tsourveloudis, N. C. 2010. "On the Evolutionary-Fuzzy Control of WIP in Manufacturing Systems." *Neurocomputing*, Bayesian Networks / Design and Application of Neural Networks and Intelligent Learning Systems (KES 2008 / Bio-inspired Computing: Theories and Applications (BIC-TA 2007), 73 (4): 648–54. https://doi.org/10.1016/j.neucom.2009.06.020.

Tsubone, H., M. Ohba, and T. Uetake. 1996. "The Impact of Lot Sizing and Sequencing on Manufacturing Performance in a Two-Stage Hybrid Flow Shop." *International Journal of Production Research* 34 (11): 3037–53. https://doi.org/10.1080/00207549608905076.

Uetake, T., H. Tsubone, and M. Ohba. 1995. "A Production Scheduling System in a Hybrid Flow Shop." *International Journal of Production Economics*, Proceedings of the 12th International Conference on Production Research, 41 (1): 395–98. https://doi.org/10.1016/0925-5273(94)00081-6.

Ulutas, B. 2011. "An Application of SMED Methodology." *World Academy of Science, Engineering and Technology* 79 (July): 100–103.

Ünler, A., and Z. Güngör. 2009. "Applying K-Harmonic Means Clustering to the Part-Machine Classification Problem." *Expert Systems with Applications* 36 (2, Part 1): 1179–94. https://doi.org/10.1016/j.eswa.2007.11.048.

Uzsoy, R. 1995. "Scheduling Batch Processing Machines with Incompatible Job Families." *International Journal of Production Research* 33 (10): 2685–2708. https://doi.org/10.1080/00207549508904839.

Uzsoy, R., C.-Y. Lee, and L. A. Martin-Vega. 1992. "A Review of Production Planning and Scheduling Models in the Semiconductor Industry Part I: System Characteristics, Performance Evaluation and Production Planning." *IIE Transactions* 24 (4): 47–60. https://doi.org/10.1080/07408179208964233.

—. 1994. "A Review of Production Planning and Scheduling Models in the Semiconductor Industry Part II: Shop-Floor Control." *IIE Transactions* 26 (5): 44–55. https://doi.org/10.1080/07408179408966627.

Vairaktarakis, G. L. 2003. "Simple Algorithms for Gilmore–Gomory's Traveling Salesman and Related Problems." *Journal of Scheduling* 6 (6): 499–520. https://doi.org/10.1023/A:1026200209386.

—. 2004. "Flexible Hybrid Flowshops." In *Handbook of Scheduling*, 86–119. Chapman and Hall/CRC. https://doi.org/10.1201/9780203489802-12.

Van de Vonder, S., E. Demeulemeester, and W. Herroelen. 2007. "A Classification of Predictive-Reactive Project Scheduling Procedures." *Journal of Scheduling* 10 (3): 195–207. https://doi.org/10.1007/s10951-007-0011-2.

Van Goubergen, D., and H. Van Landeghem. 2002. "Rules for Integrating Fast Changeover Capabilities into New Equipment Design." *Robotics and Computer-Integrated Manufacturing*, 11th International Conference on Flexible Automation and Intelligent Manufacturing, 18 (3): 205–14. https://doi.org/10.1016/S0736-5845(02)00011-X.

Varela, M. L. R., J. Trojanowska, J. Machado, S. Carmo-Silva, and N. M.L. Costa. 2017. "Comparative Simulation Study of Production Scheduling

in the Hybrid and the Parallel Flow." *Management and Production Engineering Review* 8 (2): 69–80. https://doi.org/10.1515/mper-2017-0019.

Venkumar, P., and A. N. Haq. 2006. "Complete and Fractional Cell Formation Using Kohonen Self-Organizing Map Networks in a Cellular Manufacturing System." *International Journal of Production Research* 44 (20): 4257–71. https://doi.org/10.1080/00207540500507450.

Vickson, R. G. 1980a. "Two Single Machine Sequencing Problems Involving Controllable Job Processing Times." *A I I E Transactions* 12 (3): 258–62. https://doi.org/10.1080/05695558008974515.

—. 1980b. "Choosing the Job Sequence and Processing Times to Minimize Total Processing Plus Flow Cost on a Single Machine." *Operations Research* 28 (5): 1155–67. https://doi.org/10.1287/opre.28.5.1155.

Vickson, R. G., and B. E. Aldredsson. 1992. "Two- and Three-Machine Flow Shop Scheduling Problems with Equal Sized Transfer Batches." *International Journal of Production Research* 30 (7): 1551–74. https://doi.org/10.1080/00207549208948107.

Vidalis, M. I., C. T. Papadopoulos, and C. Heavey. 2005. "On the Workload and 'Phaseload' Allocation Problems of Short Reliable Production Lines with Finite Buffers." *Computers & Industrial Engineering*, Selected Papers from The 30th International Conference on Computers & Industrial Engineering, 48 (4): 825–37. https://doi.org/10.1016/j.cie.2004.12.011.

Vieira, G. E., J. W. Herrmann, and E. Lin. 2003. "Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods." *Journal of Scheduling* 6 (1): 39–62. https://doi.org/10.1023/A:1022235519958.

Vignier, A., J.-C. Billaut, and C. Proust. 1999. "Les Problèmes d'ordonnancement de Type « flow-Shop » Hybride : État de l'art." *RAIRO - Operations Research - Recherche Opérationnelle* 33 (2): 117–83.

Vollmann, T. E., W. L. Berry, D. C. Whybark, and F. R. Jacobs. 2005. *MANUFACTURING PLANNING AND CONTROL SYSTEMS FOR SUPPLY CHAIN MANAGEMENT: The Definitive Guide for Professionals*. McGraw-Hill Education.

Vrba, P., V. Mařík, and P. Kadera. 2012. "MAST: From a Toy to Real-Life Manufacturing Control." In *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 428–33.

https://doi.org/10.1109/SNPD.2012.137.
Wagelmans, A., S. van Hoesel, and A. Kolen. 1992. "Economic Lot Sizing: An O(n Log n) Algorithm That Runs in Linear Time in the Wagner-Whitin Case." *Operations Research* 40 (1-supplement-1): S145–56. https://doi.org/10.1287/opre.40.1.S145.
Wagner, H. M., and T. M. Whitin. 1958. "Dynamic Version of the Economic Lot Size Model." *Management Science* 50 (12): 1770–74.
Wang, H. 2005. "Flexible Flow Shop Scheduling: Optimum, Heuristics and Artificial Intelligence Solutions." *Expert Systems* 22 (2): 78–85. https://doi.org/10.1111/j.1468-0394.2005.00297.x.
Wang, I-L., Y.-C. Wang, and C.-W. Chen. 2013. "Scheduling Unrelated Parallel Machines in Semiconductor Manufacturing by Problem Reduction and Local Search Heuristics." *Flexible Services and Manufacturing Journal* 25 (3): 343–66. https://doi.org/10.1007/s10696-012-9150-7.
Wang, J.-B. 2007a. "Flow Shop Scheduling Problems with Decreasing Linear Deterioration under Dominant Machines." *Computers & Operations Research* 34 (7): 2043–58. https://doi.org/10.1016/j.cor.2005.08.008.
—. 2007b. "Single-Machine Scheduling Problems with the Effects of Learning and Deterioration." *Omega* 35 (4): 397–402. https://doi.org/10.1016/j.omega.2005.07.008.
—. 2008. "Single-Machine Scheduling with Past-Sequence-Dependent Setup Times and Time-Dependent Learning Effect." *Computers & Industrial Engineering* 55 (3): 584–91. https://doi.org/10.1016/j.cie.2008.01.017.
Wang, J.-B., X. Huang, Y.-B. Wu, and P. Ji. 2012. "Group Scheduling with Independent Setup Times, Ready Times, and Deteriorating Job Processing Times." *The International Journal of Advanced Manufacturing Technology* 60 (5): 643–49. https://doi.org/10.1007/s00170-011-3639-1.
Wang, J.-B., and J.-X. Li. 2011. "Single Machine Past-Sequence-Dependent Setup Times Scheduling with General Position-Dependent and Time-Dependent Learning Effects." *Applied Mathematical Modelling* 35 (3): 1388–95. https://doi.org/10.1016/j.apm.2010.09.017.
Wang, L., S. Wang, and X. Zheng. 2016. "A Hybrid Estimation of Distribution Algorithm for Unrelated Parallel Machine Scheduling with Sequence-Dependent Setup Times." *IEEE/CAA Journal of Automatica Sinica* 3 (3): 235–46. https://doi.org/10.1109/JAS.2016.7508797.

Wang, X., and L. Tang. 2009. "A Tabu Search Heuristic for the Hybrid Flowshop Scheduling with Finite Intermediate Buffers." *Computers & Operations Research* 36 (3): 907–18. https://doi.org/10.1016/j.cor.2007.11.004.

Wazed, M. A., S. Ahmed, and Y. Nukman. 2010. "A Review of Manufacturing Resources Planning Models under Different Uncertainties: State-of-the-Art and Future Directions." *South African Journal of Industrial Engineering* 21 (1): 17–34.

Wei, J. C., and N. Gaither. 1990. "A Capacity Constrained Multiobjective Cell Formation Method." *Journal of Manufacturing Systems* 9 (3): 222–32. https://doi.org/10.1016/0278-6125(90)90053-K.

Wen, H.-W., L.-C. Fu, and S.-S. Huang. 2001. "Modeling, Scheduling, and Prediction in Wafer Fabrication Systems Using Queueing Petri Net and Genetic Algorithm." In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, 4:3559–64 vol.4. https://doi.org/10.1109/ROBOT.2001.933169.

Wenqi, H., and Y. Aihua. 2004. "An Improved Shifting Bottleneck Procedure for the Job Shop Scheduling Problem." *Computers & Operations Research* 31 (12): 2093–2110. https://doi.org/10.1016/S0305-0548(03)00243-0.

Williams, S. K., and M. J. Magazine. 2007. "Heuristic Approaches for Batching Jobs in Printed Circuit Board Assembly." *Computers & Operations Research* 34 (7): 1943–62. https://doi.org/10.1016/j.cor.2005.07.023.

Wilson, J. M. 2016. "The Origin of Materials Requirements Planning in Frederick W. Taylor's Planning Office." *International Journal of Production Research* 54 (5): 1535–53.

Wong, T. C., and S. C. Ngan. 2013. "A Comparison of Hybrid Genetic Algorithm and Hybrid Particle Swarm Optimization to Minimize Makespan for Assembly Job Shop." *Applied Soft Computing*, Hybrid evolutionary systems for manufacturing processes, 13 (3): 1391–99. https://doi.org/10.1016/j.asoc.2012.04.007.

Wright, T. P. 1936. "Factors Affecting the Cost of Airplanes." *Journal of the Aeronautical Sciences* 3 (4): 122–28. https://doi.org/10.2514/8.155.

Wu, C.-C., and W.-C. Lee. 2008. "Single-Machine Group-Scheduling Problems with Deteriorating Setup Times and Job-Processing Times." *International Journal of Production Economics* 115 (1): 128–33. https://doi.org/10.1016/j.ijpe.2008.05.004.

Wu, C.-C., Y.-R. Shiau, and W.-C. Lee. 2008. "Single-Machine Group Scheduling Problems with Deterioration Consideration." *Computers &*

Operations Research, Part Special Issue: Algorithms and Computational Methods in Feasibility and Infeasibility, 35 (5): 1652–59. https://doi.org/10.1016/j.cor.2006.09.008.

Wysk, R. A. 1977. "Automated Process Planning and Selection Program: APPAS." Ph. D. Thesis, West Layfette, Indiana: School of Industrial Engineering, Purdue University.

Xanthopoulos, A. S., and D. E. Koulouriotis. 2014. "Multi-Objective Optimization of Production Control Mechanisms for Multi-Stage Serial Manufacturing-Inventory Systems." *The International Journal of Advanced Manufacturing Technology* 74 (9): 1507–19. https://doi.org/10.1007/s00170-014-6052-8.

Xie, J., and X. Wang. 2005. "Complexity and Algorithms for Two-Stage Flexible Flowshop Scheduling with Availability Constraints." *Computers & Mathematics with Applications* 50 (10): 1629–38. https://doi.org/10.1016/j.camwa.2005.07.008.

Xing, W. 2002. "A Bin Packing Problem with Over-Sized Items." *Operations Research Letters* 30 (2): 83–88. https://doi.org/10.1016/S0167-6377(02)00118-9.

Xuan, H., and L. Tang. 2007. "Scheduling a Hybrid Flowshop with Batch Production at the Last Stage." *Computers & Operations Research* 34 (9): 2718–33. https://doi.org/10.1016/j.cor.2005.10.014.

Yang, J., and M. E. Posner. 2010. "Flow Shops with WIP and Value Added Costs." *Journal of Scheduling* 13 (1): 3–16. https://doi.org/10.1007/s10951-009-0130-z.

Yao, S., Z. Jiang, N. Li, H. Zhang, and N. Geng. 2011. "A Multi-Objective Dynamic Scheduling Approach Using Multiple Attribute Decision Making in Semiconductor Manufacturing." *International Journal of Production Economics* 130 (1): 125–33. https://doi.org/10.1016/j.ijpe.2010.12.014.

Yaurima, V., L. Burtseva, and A. Tchernykh. 2009. "Hybrid Flowshop with Unrelated Machines, Sequence-Dependent Setup Time, Availability Constraints and Limited Buffers." *Computers & Industrial Engineering* 56 (4): 1452–63. https://doi.org/10.1016/j.cie.2008.09.004.

Yavuz, Y. 2013. "Lot Streaming with Flexible Process Plans." *International Journal of Production Research* 51 (17): 5055–72. https://doi.org/10.1080/00207543.2013.784413.

Yazdani Sabouni, M. T., and F. Jolai. 2010. "Optimal Methods for Batch Processing Problem with Makespan and Maximum Lateness Objectives." *Applied Mathematical Modelling* 34 (2): 314–24. https://doi.org/10.1016/j.apm.2009.04.007.

Ye, D., and G. Zhang. 2009. "On-Line Extensible Bin Packing with Unequal Bin Sizes." *Discrete Mathematics & Theoretical Computer Science* Vol. 11 no. 1 (January). https://dmtcs.episciences.org/472/pdf.

Yimer, A. D., and K. Demirli. 2009. "Fuzzy Scheduling of Job Orders in a Two-Stage Flowshop with Batch-Processing Machines." *International Journal of Approximate Reasoning*, Special Section on Recent advances in soft computing in image processing and Special Section on Selected papers from NAFIPS 2006, 50 (1): 117–37. https://doi.org/10.1016/j.ijar.2007.08.013.

Yin, Y., D. Li, D. Wang, and T. C. E. Cheng. 2018. "Single-Machine Serial-Batch Delivery Scheduling with Two Competing Agents and Due Date Assignment." *Annals of Operations Research*, April. https://doi.org/10.1007/s10479-018-2839-6.

Yin, Y., and K. Yasuda. 2005. "Similarity Coefficient Methods Applied to the Cell Formation Problem: A Comparative Investigation." *Computers & Industrial Engineering*, GroupTechnology/Cellular Manufacturing, 48 (3): 471–89. https://doi.org/10.1016/j.cie.2003.01.001.

Yoon, S. H., and J. A. Ventura. 2002. "Minimizing the Mean Weighted Absolute Deviation from Due Dates in Lot-Streaming Flow Shop Scheduling." *Computers and Operations Research* 29 (10): 1301–15. https://doi.org/10.1016/S0305-0548(01)00032-6.

Yousefi Yegane, B., N. Khanlarzade, and I. Nakhai Kamalabadi. 2017. "Critical Path Method for Flexible Job Shop Scheduling Problem with Preemption." *International Journal of Engineering* 30 (2): 261–69.

Yu, T.-S., H.-J. Kim, and T.-E. Lee. 2017. "Minimization of Waiting Time Variation in a Generalized Two-Machine Flowshop With Waiting Time Constraints and Skipping Jobs." *IEEE Transactions on Semiconductor Manufacturing* 30 (2): 155–65. https://doi.org/10.1109/TSM.2017.2662231.

Yuan, J. J., Z. H. Liu, C. T. Ng, and T. C. E. Cheng. 2006. "Single Machine Batch Scheduling Problem with Family Setup Times and Release Dates to Minimize Makespan." *Journal of Scheduling* 9 (6): 499–513. https://doi.org/10.1007/s10951-006-8776-2.

Zakaria, Z., and S. Petrovic. 2012. "Genetic Algorithms for Match-up Rescheduling of the Flexible Manufacturing Systems." *Computers & Industrial Engineering* 62 (2): 670–86. https://doi.org/10.1016/j.cie.2011.12.001.

Zandieh, M. 2019. "Scheduling of Virtual Cellular Manufacturing Systems: A Biogeography-Based Optimization Algorithm." *Applied Artificial Intelligence* 33 (7): 594–620.

https://doi.org/10.1080/08839514.2019.1577021.
Zandieh, M., S. M. T. Fatemi Ghomi, and S. M. Moattar Husseini. 2006. "An Immune Algorithm Approach to Hybrid Flow Shops Scheduling with Sequence-Dependent Setup Times." *Applied Mathematics and Computation* 180 (1): 111–27. https://doi.org/10.1016/j.amc.2005.11.136.
Zangwill, W. I. 1966. "A Deterministic Multiproduct, Multi-Facility Production and Inventory Model." *Operations Research* 14 (3): 486–507. https://doi.org/10.1287/opre.14.3.486.
Zhang, A. 2013. "Cellular Manufacturing Systems." In *Methods in Product Design*, 1st ed., 342. Boca Raton: Taylor & Francis Group. https://doi.org/10.4324/9781315300252.
Zhang, B., Q.-k. Pan, L. Gao, X.-l. Zhang, H.-y. Sang, and J.-q. Li. 2017. "An Effective Modified Migrating Birds Optimization for Hybrid Flowshop Scheduling Problem with Lot Streaming." *Applied Soft Computing* 52 (March): 14–27. https://doi.org/10.1016/j.asoc.2016.12.021.
Zhang, D. Z., and A. I. Anosike. 2012. "Modelling and Simulation of Dynamically Integrated Manufacturing Systems." *Journal of Intelligent Manufacturing* 23 (6): 2367–82. https://doi.org/10.1007/s10845-010-0494-0.
Zhang, G., J. Zhang, and S. Y. Tian. 2011. "RFID-Enabled Real-Time Production Tracking System for PCB Assembly Industry." Applied Mechanics and Materials. Trans Tech Publications Ltd. 2011. https://doi.org/10.4028/www.scientific.net/AMM.80-81.1330.
Zhang, H., Z. Jiang, and C. Guo. 2009. "Simulation-Based Optimization of Dispatching Rules for Semiconductor Wafer Fabrication System Scheduling by the Response Surface Methodology." *The International Journal of Advanced Manufacturing Technology* 41 (1): 110–21. https://doi.org/10.1007/s00170-008-1462-0.
Zhang, J., X. Yao, J. Zhou, J. Jiang, and X. Chen. 2017. "Self-Organizing Manufacturing: Current Status and Prospect for Industry 4.0." In *2017 5th International Conference on Enterprise Systems (ES)*, 319–26. https://doi.org/10.1109/ES.2017.59.
Zhang, W., J. Liu, and R. J. Linn. 2003. "Model and Heuristics for Lot Streaming of One Job in M-1 Hybrid Flowshops." *International Journal of Operations and Quantitative Management* 9 (1): 49–64.
Zhang, W., C. Yin, J. Liu, and R. J. Linn. 2005. "Multi-Job Lot Streaming to Minimize the Mean Completion Time in m-1 Hybrid Flowshops." *International Journal of Production Economics* 96 (2): 189–200. https://doi.org/10.1016/j.ijpe.2004.04.005.

Zhu, X., and W. E. Wilhelm. 2006. "Scheduling and Lot Sizing with Sequence-Dependent Setup: A Literature Review." *IIE Transactions* 38 (11): 987–1007. https://doi.org/10.1080/07408170600559706.

ABBREVIATIONS

AFS	Assembly Flow Shop
AGV	Automated Guided Vehicle
AJS	Assembly Job Shop
AOS	Assembly Open Shop
BOM	Bill of Materials
BPM	Batch Processing Machine
CONWIP	CONstant Work-In-Process
CF	Cell Formation (problem)
CLS	Capacitated Lot Sizing (model)
CP	Constraint Programming (model)
CR	Critical Ration (rule)
DDMRP	Demand-Driven MRP (system)
DBR	Drum-Buffer-Rope (method)
DLS	Discrete Lot sizing and Scheduling (model)
DP	Dynamic Programming
EDD	Earliest Due Date first (rule)
ELS	Economic Lot Scheduling (model)
EOQ	Economic Order Quantity (model)
EPL	Economic Production Lot (formula)
ERP	Enterprise Resource Planning (system)
FCFS	First Come, First Served (rule)
FFL	Flexible flow line
FFS	Flexible Flow Shop
FIFO	First-In, First-Out (rule)
FLS	Flexible Lot Streaming
FMS	flexible manufacturing system
FS	Flow Shop
FSMP	Flow shop with multiple processors
GSU	Grouped Set-Up
IC	Integrated Circuit
ISB	Improved Shifting Bottleneck (procedure)
JIS	Just-In-Sequence (paradigm)
JIT	Just-In-Time (paradigm)
JS	Job Shop
HDD	Hard Disc Device

HFS	Hybrid Flow Shop
HPP	Hierarchical Production Planning (system)
HT	High Tech
HV/HM	High Volume/High Mixture (of nomenclatures)
LAPT	Longest Alternate Processing Time first (rule)
LBFS	Last Buffer-First Serve (rule)
LCD	Liquid Crystal Display
LCS	Least Setup Cost
LIFO	Last-In, First-Out (rule)
LT	Lead Time
LPT	Longest Processing Time first (rule)
LV/HM	Low Volume/High Mix (paradigm)
MAS	Multi-Agent System
MILP	mixed integer linear programming (model)
MIP	Mixed Integer Programming (model)
MPC	Manufacturing Planning and Control (system)
MPS	Master Production Schedule
MRP, MRP-I	Material Requirements Planning (system)
MRP-II	Manufacturing Resource Planning (system)
MS	Mixed Shop
OPT	Optimized Production Technology
OS	Open Shop
PCB	Printed Circuit Board
PTAS	Polynomial Time Approximation Scheme
RFID	Radio Frequency Identification
SB	Shifting Bottleneck (heuristic)
SDS	Sequence-Dependent Scheduling
SDST	Sequence-Dependent Setup Times
SMED	Single Minute Exchange of Die (setup reduction paradigm)
SMS	Semiconductor Manufacturing System
SPT	Shortest Processing Time first (rule)
TOC	Theory of Constraints
TPM	Total Productivity Maintenance (system)
TPS	Toyota Production System
TWT	Total Weighted Tardiness (criterio)
VPL	Virtual Production Line
ULS	Uncapacitated Lot Sizing (model)
WIP	Work-In-Process
WLPT	Weighted Longest Processing Time first (rule)
WSPT	Weighted Shortest Processing Time first (rule)

INDEX