Operations Research/Computer Science Interfaces Series

Volume 52

For further volumes:
http://www.springer.com/series/6375

Lars Mönch • John W. Fowler • Scott J. Mason

Production Planning and Control for Semiconductor Wafer Fabrication Facilities

Modeling, Analysis, and Systems

Springer

Lars Mönch
Department of Mathematics
and Computer Science
University of Hagen
Universitätsstraße 1
Hagen, Germany

John W. Fowler
W.P. Carey School of Business
Department of Supply Chain
Management
Arizona State University
Tempe, AZ, USA

Scott J. Mason
Department of Industrial Engineering
Clemson University
Freeman Hall 124
Clemson, SC, USA

ISSN 1387-666X
ISBN 978-1-4614-4471-8 ISBN 978-1-4614-4472-5 (ebook)
DOI 10.1007/978-1-4614-4472-5
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012944171

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Semiconductor manufacturing is one of the most important segments of the global manufacturing sector. Today semiconductor wafer fabrication facilities, for short called wafer fabs in the rest of this monograph, can be found in the USA, Europe, and Asia. Starting in the mid-1980s, the number of people in academia that deal with modeling and analysis of wafer fab operations has constantly increased. While in the beginning the number of academics working on these problems was quite small, today there are very active research groups around the world. A growing number of academics have contributed to the literature related to modeling and analysis of wafer fabs in such different areas as simulation modeling, dispatching and shop-floor scheduling, queueing models, production planning models, supply network planning models, and design and implementation of information systems for decision support. Furthermore, a wide range of other engineering models to support yield and quality improvement have been developed (cf. Chien et al. [49]). The vast academic interest in modeling and analysis of wafer fabs is caused by the fact that wafer fabs are one of the most complex and challenging industrial environments in use today.

The number of scholarly publications in this area has also increased significantly over the years. However, there are only a few survey papers that attempt to give a complete picture of various aspects of modeling and analysis of wafer fabs. The most popular among these papers are those by Uzsoy et al. [306, 307]. Except for the monographs by Atherton and Atherton [14] and Ovacik and Uzsoy [223], there are no further books in this area. The book [14] discusses modeling and analysis issues only briefly and from a different point of view. The second related book deals mainly with certain decomposition strategies based on disjunctive graphs and the shifting bottleneck heuristic for scheduling the back-end stage of semiconductor manufacturing.

In this monograph, we are interested in covering a broader area, and we attempt to take recent research trends into account. To our best knowledge, there is no book on modeling and analysis in semiconductor manufacturing that simultaneously considers production planning, production control, and

the related information systems. In this book, after presenting basic concepts in the semiconductor manufacturing process and in basic modeling and analysis tools, we introduce production control schemes that are based on dispatching rules as they are predominately used in practice. Next, we discuss recent scheduling approaches. We continue with a description of order release strategies for wafer fabs. We then introduce different production planning approaches with a focus on capacity planning. In the second to last chapter, we present research related to the important field of automated material handling systems. Finally, we describe various aspects of decision support provided by manufacturing execution systems and advanced planning systems.

Based on our experience and research interests, we mainly suggest heuristics throughout the book. However, when practical, we also discuss methods that lead to optimal solutions. It is an important feature of this book that we consider discrete-event simulation in different situations as a modeling and analysis tool [89].

We have been helped by many people in the course of preparing this book. We would like to thank Cheryl Dwyer for carefully reading the entire manuscript and for providing many helpful suggestions for improvements. Ulrike Schmidt helped us by preparing parts of the figures and by checking the references. We would also like to thank Stefan Voß who strongly supported the inclusion of this book into the Springer Operations Research/Computer Science Interfaces Series.

Finally, we want to give special thanks to our friends and colleagues Oliver Rose, Stéphane Dauzère-Pérès, and Leon McGinnis. Many of the results in this monograph represent joint work with these scientists, and their insights, criticism, and support have been an important component of our research efforts. Furthermore, we also thank Reha Uzsoy, Robert C. Leachmann, Tae-Eog Lee, and Chen-Fu Chien among others for fruitful discussions that led to insights into semiconductor manufacturing and helped us to structure our knowledge on semiconductor manufacturing and finally to write this monograph.

Many people from the industry helped us with insights, datasets, and challenging problems. We especially thank Hans Ehm and Andreas Klemmt, Infineon Technologies AG; Volker Schmalfuß, X-Fab Semiconductor Foundries AG; Karl Kempf, Intel Corporation; You-In Choung, Samsung; Shekar Krishnaswamy, Applied Materials; and Detlef Pabst and Marcel Stehli, GLOBALFOUNDRIES.

Finally, we would like to thank Neil Levine and Matthew Amboy from Springer for their support and patience during our work on this monograph.

Hagen, Germany — Lars Mönch
Tempe, AZ, USA — John W. Fowler
Clemson, SC, USA — Scott J. Mason

Contents

Notation

The following symbols and notation will typically be used throughout the book. We have reused some of them for several purposes due to the limited supply of symbols from the alphabet and to be consistent with papers from the literature. Their usage should be clear from the context. We also introduce additional notation when needed throughout the book.

Notation	Explanation
ACT	Average cycle time
AL	Average lateness
AT	Average tardiness
AWT	Average weighted tardiness
aux	Auxiliary resource
B	Maximum batch size
BS	Base system
BP	Base process
CDT	Carrier delivery time
C_j	Completion time of job j
$C_{\max}$	Makespan
CP	Control process
CS	Control system
CT	Cycle time
CT_j	Cycle time of job j
d_i	Individual desirability function for objective i
d_j	Due date of job j
d_{jk}	Due date for process step k of job j
D	Combined desirability function
$DU[a,b]$	Discrete uniform distribution over the integer set $\{a,\ldots,b\}$
ε	Small number or a random error $\sim N(0,\sigma^2)$
f	Number of incompatible job families
F	Number of FOUPs in a MOJ scheduling problem
$F(j)$	Family of job j
FF	Flow factor
FJm	Flexible job shop with m machine groups
h	Planning horizon
i	Machine index

Notation	Explanation
IS	Information system
j	Job index
Jm	Job shop with m machines
JS	Job processing system
k	Process step index
κ	Look-ahead parameter
L_j	Lateness of job j
$L_{\max}$	Maximum lateness
λ	(Arrival) rate
m	Number of machines
M_j	Set of machines that are possible for job j
MS	Material flow system
n	Number of jobs
n_j	Number of process steps of job j
NTJ	Number of tardy jobs
$N(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
O_j	Process flow of job j
O_{jk}	Operation k of job j
OS	Operational system
$\bar{p}$	Average processing time
$\mathrm{p-batch}$	Parallel batching
p_j	Processing time of job j
p_{jk}	Processing time of process step k of job j
Pm	Parallel identical machines, where the number of machines is m
PP	Planning process
PS	Planning system
recrc	Recirculation, i.e., reentrant flow
r_j	Ready time of job j
Rm	Unrelated parallel machines, where the number of machines is m
$\mathbb{R}_+$	The set of non-negative real numbers
$\mathrm{s-batch}$	Serial batching
s_{jk}	Setup time to process job k after job j
$s_{kl,ji}$	Setup time to process step l of job k after step i of job j
t	Current time
TC	Total completion time
T_j	Tardiness of job j
TP	Throughput
TT	Total tardiness
TWC	Total weighted completion time
TWT	Total weighted tardiness
τ_Δ	Planning interval
τ_{ah}	Additional planning horizon
$U(a,b)$	Continuous uniform distribution over the interval (a,b)
U_j	Indicator variable that is 1 if job j is tardy
Var(CT)	Variance of the cycle time
Var(L)	Variance of the lateness
w_j	Weight of job j
WNTJ	Weighted number of tardy jobs
x^+	$\max(x,0)$
z_i	Weight for the individual desirability function d_i
$\mathbb{Z}_+$	The set of non-negative integers

Chapter 1
Introduction

The purpose of this chapter is to provide an overview of the book. Therefore, we start by discussing motivation for modeling and analysis of semiconductor manufacturing. Semiconductor manufacturing is an extreme environment for production planning and control, scheduling, and simulation models. The enormous size of the facilities and supply chains in the semiconductor industry, the permanent appearance of uncertainty, and rapid technological changes lead to an environment that brings approaches developed for other industries under stress (see Chien et al. [49] for a related discussion). The capital intensive nature of the semiconductor industry requires manufacturing systems to run consistently at high utilization levels, reentrant flows create complex competition for limited resource capacity, and the ever-increasing level of automation reduces the ability to rely solely on people for production planning and control. Models that are successful in the semiconductor industry will likely find reasonable applications in other areas. A second source of academic interest in modeling and analysis of semiconductor manufacturing is the insight that the semiconductor manufacturing environment initiates on the formulation of some problems that had not been widely studied in other industries (cf. Chien et al. [49]).

We obviously cannot give a complete account of modeling and analysis of semiconductor manufacturing in a book of only a few hundred pages. Hence, instead of attempting the impossible, we have chosen production planning and control of wafer fabs from the perspective of our own interests and research programs. In the second section of this chapter, we provide an outline of the content of the book.

1.1 Motivation

In the last decade, the electronics industry has become one of the largest industries in the world. At the heart of this industry is the manufacturing of integrated circuits (ICs or chips) on thin silicon discs (wafers). The fabrication

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5_1,

of ICs on silicon wafers is arguably the most complex manufacturing process in existence [14, 110, 276, 306, 307]. This complexity is caused by many factors including multiple products, routes with several hundred process steps, and a large number of machines (tools). There are four basic steps in manufacturing ICs (cf. Uzsoy et al. [306]):

- Wafer fabrication
- Sort (or probe)
- Assembly
- Test

In wafer fabrication, the layers of the ICs are fabricated onto raw silicon wafers. Next, the completed wafers are sent to sort where electronic probes perform an electrical test on each IC to determine basic functionality. Then, the probed wafers are sent to assembly, where the wafers are cut into individual ICs and the functioning ones are put into a package that allows connection with higher level devices such as PCs, cell phones, etc. Finally, the packaged ICs are tested and labeled. Wafer fabrication is the most time-consuming and the most costly step and is the primary focus of this monograph. It is characterized by the following process conditions:

- Reentrant flow, i.e., a lot of wafers, called jobs to be consistent with the scheduling literature, may visit the same machine several times
- A mix of different process types, for example, batch processes, i.e., several jobs, can be processed simultaneously on the same machine vs. single wafer processes
- Unrelated parallel machines that are often highly unreliable or require considerable preventive maintenance to keep them reliable
- Sequence-dependent setup times that can in some cases take considerably longer than the time to process a job
- Variety of products with a changing product mix
- Customer due dates that are very aggressive

In addition, the machines used for processing jobs are extremely expensive, some as high as US$40 million, and thus are scarce resources. This is the main reason for reentrant flow of the jobs through the wafer fabrication facility (wafer fab). This type of flow causes problems related to production control of wafer fabs that are different than production control problems in classical job shops, for example, the occurrence of dynamic bottlenecks. The cost of today's fab, up to US$5 billion, leads to competition between production jobs and prototype (or engineering) jobs for processing time on the machines. Many companies do process development for the next generation of products in the same fab that produces the current generation products in high volume.

In the past, sources of reducing costs in semiconductor manufacturing were decreasing the size of the chips, increasing the wafer sizes, and improving the yield, simultaneously with efforts to improve operational processes inside the wafer fabs (cf. Schömig and Fowler [276]). While shrinking the size of the chips will likely continue to significantly reduce costs of semiconductor

manufacturing for the next several generations of products, the productivity gains from increasing wafer sizes and improving yields will not likely continue at historic levels. There will be wafer size increases, but the increased costs of the larger wafers will offset some of the productivity gains. In the case of yield, the gains will not be as large as in the past because yields are already high, which leaves less room for improvement. Currently, it seems that the improvement of operational processes creates the best opportunity to realize the necessary cost reductions. Therefore, the development of efficient planning and control strategies is highly desirable in the semiconductor wafer fabrication domain. In the course of the development of new planning and control algorithms, researchers and developers have to take into account the new opportunities in advanced software and hardware technologies.

1.2 Outline of the Book

In this monograph, we consider these problems. We show that from our point of view productivity improvements in the semiconductor industry will have to come through the implementation of operations research and industrial engineering tools and techniques and through application of state-of-the-art computing technologies.

In Chap. 2, we provide a detailed process description of semiconductor manufacturing. We use the notion of base system, base process, control system, control process, planning system, planning process, and finally of the information system. A manufacturing system consists of a base system that contains all the resources, i.e., tools, secondary resources, and operators. The corresponding base process is given by jobs that consume capacities of the resources during processing. The resource allocation process of the jobs is influenced by the production control process that is performed by using the production control system. The production control system consists of the computers and the software used to produce production control instructions, i.e., software with dispatching and scheduling capabilities. The production control process determines when and under which circumstances a certain control algorithm is used to determine production control instructions. The production planning system is given by a set of computers and software that is used to determine production planning instructions. The production control system and production planning system combined with human decision makers and the operational system form the information system. The production planning system determines when certain planning actions have to be performed. The main results of production planning are quantities and points of time for releasing orders into the base system. In this book, we are mainly interested in the design of the production control and production planning system and also in the production control and planning process. Less attention is paid to the design of the base system and process.

We introduce the notion of complex job shops (cf. [172, 223]). Complex job shops are a specific class of job shops that are characterized by unrelated parallel machines, batch machines, reentrant process flows, and process

variability. We describe the basic process steps of semiconductor manufacturing and introduce a production planning and control hierarchy that includes the enterprise level, the factory level, and finally the work center level from a resource point of view.

Most enterprises consist of several factories that are geographically distributed. The following questions are interesting on the enterprise level:

- How do we maximize enterprise capacity?
- What are the effects of product mix?
- How do we minimize enterprise costs?

However, in this monograph, we mainly concentrate on the factory and the work center level instead of the enterprise level. We are interested in the following types of questions for the factory level:

- What is the best dispatching strategy?
- Is it beneficial to use a scheduling system?
- What is the impact of different lot sizes?
- What is the impact of different order/job release strategies?
- Is it worthwhile to integrate machine-related and automated material handling system-related decisions?

A single factory consists of different work centers. The following issues will be discussed for the work center level:

- Is it necessary to use different dispatching strategies for different types of work centers?
- What is a good batching strategy?
- What is an effective way to manage reticles?
- How should cluster tools be scheduled?

A batch is a collection of jobs that are processed at the same time on the same machine. Reticles are secondary resources that contain the information for specific integrated chips. Finally, a cluster tool may perform several consecutive process steps of a certain job.

From an operations management point of view, we differentiate between planning at the highest level, order release, scheduling, and dispatching at the lowest level. Each of these different functionalities is related to a certain horizon. The decisions on the higher levels are generally made on a periodic, but infrequent basis. Each decision typically has a huge financial impact. The plans from the planning level are used to make order/job release decisions. Finally, priorities are assigned to each job in order to process the jobs on single machines.

In this book, we start by describing dispatching and end up with production planning. The resulting planning and control hierarchy consists of the following layers:

- Planning with a time horizon ranging from months to years
- Order release in a weekly or bi-weekly frequency

- Scheduling each shift/day
- Dispatching in a minute-by-minute manner depending on the speed of the material flow

Production plans are the output of the planning layer. These plans are then used to release jobs. These job starts are an input of the scheduling layer. A detailed description of activities at the machine level is the result of the scheduling layer. A schedule can be used to establish priorities for operations. These priorities are important for dispatching decisions.

In this monograph, of course, we cannot completely answer all questions. However, the aim of this monograph is to provide tools and techniques from operations research, industrial engineering, and computer science to model and control wafer fabs effectively. We hope that readers of this book are willing to accept the abstract modeling and operations research language.

Therefore, after a description of the base process and base system in Chap. 2, we provide the necessary modeling and analysis tools in Chap. 3. Models are used within the production planning and control process for representing the base system and base process and for decision-making. They can be part of the production control and also the production planning system. We differentiate between dynamic and static, deterministic and stochastic, and descriptive and prescriptive models. Dynamic models contain a time dependency, while static models do not. Stochastic models include model attributes that are specified by probability distributions, while model attributes in deterministic models are not random. Descriptive models are used to describe how a system behaves. For the purpose of this book, descriptive models usually are given by queueing models and simulation models of the base system and process. Prescriptive models are generally used for the immediate derivation of planning and control instructions. These models are used to modify the future evolution of the system. Scheduling models are examples for this class of models. Furthermore, we also describe statistical models for the design of experiments (cf. Montgomery [208]). These models are important for the performance assessment of new production planning and control approaches and will be used in the subsequent chapters of the book.

We describe various decision methods including branch-and-bound techniques, linear and mixed integer programming, stochastic programming, dynamic programming, metaheuristics, queueing theory, and discrete-event simulation for the sake of completeness. These descriptions are simply meant to equip the readers with the necessary tools to understand the models and methods to solve specific problems in the remaining chapters of the monograph.

We also deal with basic questions of performance assessment and therefore introduce important performance measures used in this monograph. We describe simulation-based methods to assess the performance of the production planning and the production control system within a dynamic and stochastic environment due to Mönch [192].

In Chap. 4, we deal with dispatching rules. A dispatching rule selects the next job to be processed among the jobs that are waiting in front of a machine group (cf. [29, 116]). Dispatching rules are generally myopic in time and space, and it may be difficult to adapt them to different situations on the shop floor. However, their decision logic is easy to understand, and they can be implemented with less effort on the shop floor. We introduce several simple dispatching rules that are in common use in the semiconductor industry like earliest due date, critical ratio, and least slack (cf. Sarin et al. [274]). Simple dispatching rules answer only the question of which job should be processed on which machine next. In this monograph, we provide results of simulation experiments with different commonly used dispatching rules.

Chapter 4 continues by combining these simple rules into composite rules. We discuss several variants of the apparent tardiness cost rule (cf. Vepsalainen and Morton [311]). A third decision is important for batching machines in addition to assignment and sequencing decisions. In this situation, we have to decide which jobs should form the batch. This decision is typically made by batching rules, and we describe several of the most important of these rules. Finally, we also introduce look-ahead rules (cf. [84, 85, 101]) that take into account the situation of downstream work centers. Look-ahead rules are important in manufacturing systems with sequence-dependent setups and batching.

In contrast to dispatching, scheduling approaches consider a time horizon for decision-making and not only a discrete set of points of time. We discuss the use of scheduling techniques in semiconductor manufacturing in Chap. 5. Scheduling is defined as the process of allocation of scarce resources over time [34, 240]. The goal of scheduling is to optimize one or more objectives in a decision-making process. The two major categories in scheduling are deterministic and stochastic scheduling. Deterministic scheduling is characterized by processing times, setup times, and job priorities that are known in advance. They are not influenced by uncertainty. In contrast, stochastic scheduling problems do not assume the existence of deterministic values for processing times, setup times, or other quantities that are used within the scheduling model. The deterministic values are replaced by corresponding probability distributions. Deterministic scheduling problems can be further differentiated into static problems where all jobs to be scheduled are available at time $t = 0$. Dynamic scheduling problems relax this condition. In this situation, jobs are ready at different points in time, i.e., $t \geq 0$.

Simulation-based scheduling means that simulation is used to determine schedules with a horizon ranging from several hours to a day. Dispatching rules that are already part of the simulation engine are used to determine what will be processed next on each machine. The assignment and the sequencing of jobs observed in the simulation are used to produce a control instruction in the original production control system that is used to influence the base system. Simulation-based scheduling relies to a large extent upon the capability to produce simulation models that represent the base

system and the base process in a very detailed manner. Automated or semi-automated simulation model generation abilities based on data in operational systems like the manufacturing execution system (MES) are necessary in order to run a simulation-based scheduling system. The selection of a final schedule as production control instructions can be based on several criteria (cf. Sivakumar [284] for a more detailed description of this approach). Usually, all stochastic effects, for example, machine breakdowns, are turned off because of the short time horizon. Appropriate model initialization is a non-trivial issue in simulation-based scheduling. Simulation-based scheduling is somewhere between dispatching and more traditional scheduling.

According to the suggested planning and control hierarchy, we consider single machine-related and work center-related scheduling problems. These scheduling problems may be the result of decomposition techniques that divide the overall, full factory scheduling problem into scheduling problems for single machines or work centers. Hence, single machine or parallel machine scheduling problems are the building blocks of full factory scheduling problems. On the other hand, this type of scheduling problem may arise independently for the processing of jobs on bottleneck machines. Many scheduling problems are known to be NP-hard (cf. Brucker [34]). Therefore, we often resort to using efficient heuristics. In this monograph, we describe mainly dispatching rule-based techniques and approaches based on genetic algorithms. We consider scheduling problems for single and parallel batch machines and for parallel machines with sequence-dependent setup times.

Furthermore, we also consider cluster tools as special mini fabs. We discuss modeling issues for cluster tools. The scheduling of jobs on cluster tools is challenging because cluster tools consist of parallel chambers that do not allow for a straightforward estimation of processing times of a job on these machines. Simulations and neural networks are used to perform this task. We discuss the scheduling of single and parallel cluster tools.

After the discussion of single and parallel machine scheduling models, we consider full factory scheduling problems. Until recently, full factory scheduling methods seemed to be too costly in comparison to dispatching methods. However, with the recent dramatic increase in computer efficiency, full fab scheduling methods have become more competitive. Because we can model a wafer fab as a complex job shop, we also have to deal with scheduling approaches for large-scale job shops.

We describe the shifting bottleneck heuristic by Adams et al. [1] as an important representative of scheduling heuristics for complex job shops. The main idea of the shifting bottleneck heuristic consists in using disjunctive graphs to model the dependency of job processing on different machines. Based on the calculation of longest paths within the disjunctive graphs, the overall scheduling problem is decomposed into smaller, more tractable scheduling problems for single or parallel machines. After the solution of these subproblems, the structure of the graph has to be updated to incorporate the scheduling decisions that were made.

We then describe modifications of the shifting bottleneck heuristic suggested by Mason et al. [172, 175]. These modifications take batching machines and reentrant flows into account with the goal of minimizing weighted total tardiness of the jobs. While subproblem solution techniques are often based on dispatching rules, we also present more sophisticated approaches based on genetic algorithms (cf. Mönch et al. [206]). We then describe simulation experiments that allow for the application of the shifting bottleneck heuristic in a rolling horizon manner in a dynamic environment (cf. Mönch et al. [202]).

Because of the reduction of solution complexity, distributed solution heuristics for production control problems seem to provide some advantage. Usually, it is difficult and time-consuming to collect in one place all the required data for centralized algorithms in real-world manufacturing systems. From this point of view, distributed algorithms working on local data provide a highly desirable approach. Therefore, we also discuss a distributed variant of the shifting bottleneck heuristic (cf. Mönch and Driessel [193]). This heuristic is based on a two-level hierarchical approach. The upper level determines expected start dates and completion dates for the jobs with respect to a certain work area, i.e., a collection of work centers, in a first step. Then in a second step, we use this information in order to apply the shifting bottleneck heuristic for the jobs in each work area. The schedules for the single work areas can be improved by using an iterative improvement technique. The distributed shifting bottleneck heuristic requires less memory and can be distributed on several computers.

We also describe an extension of the shifting bottleneck heuristic from the single-objective case to the multiobjective case (cf. Pfund et al. [235]) via a desirability function approach. By using this approach, we can model preferences for certain objectives by appropriate weight settings.

Dispatching and scheduling systems assume jobs have already been started in the base system. In Chap. 6, we discuss order release approaches (cf. Fowler et al. [87]). After a brief overview of the general push and pull philosophies, we describe the starvation avoidance approach (cf. Glassey and Resende [100]) and the workload regulation approach (cf. Wein [318]). We then discuss the use of CONWIP-like (cf. Spearman et al. [290]) order release strategies in semiconductor manufacturing. We also present work that investigates the interaction of order release schemes with the full factory scheduling approaches presented in Chap. 5. The main results of a simulation study to find appropriate order release schemes are also discussed. An optimization formulation is presented that supports order release decisions in wafer fabs.

One prerequisite for order release is (operational) capacity planning. Therefore, we consider capacity planning approaches in semiconductor manufacturing in Chap. 7. The basic problem consists of allocating production capacity to alternative products over time in the occurrence of forecasted demands to optimize some performance measure of interest.

We describe simple (static) spreadsheet-type models for this production planning problem (cf. Ozturk et al. [224]). Furthermore, we also discuss the

use of simulation models that take the dynamics of the wafer fab into account better than the spreadsheet models do. Besides short-term capacity planning schemes, we also discuss models used for medium- and long-term capacity planning. Linear and stochastic programming are used to solve these kind of problems. We discuss the problem of modeling load-dependent cycle times. Therefore, we consider iterative simulation techniques due to Hung and Leachman [120], provide a methodology to efficiently generate cycle time throughput curves (cf. Fowler et al. [86]), and introduce clearing functions due to Srinivasan et al. [292], Karmarkar [136], and Asmundsson et al. [12].

Agrawal and Heragu [3] indicate that semiconductor wafer fabs are highly automated manufacturing systems. Compared to other industries, accurate and up-to-date data are available. Therefore, we have a rather good starting position for establishing more advanced production planning and control approaches in wafer fabs. On the other hand, we also have to deal with the operational information systems on the shop floor. We have to analyze the current state of these systems. Furthermore, based on the presented methodological framework in the monograph, we derive future needs for production planning and control systems in Chap. 8.

An MES is an operational information system that is between the enterprise resource planning (ERP) systems and the base process. We describe the core functionality of an MES, which consists of providing correct information about the process flows, the machine set, the status of machines, and also the status of jobs (cf. McClellan [178]). An MES also sometimes supports the implementation of dispatching and scheduling algorithms and decisions. Scheduling and dispatching heuristics have to use the data from the base process that is contained in the MES. Furthermore, an MES is used to provide maintenance and quality assessment functionality. These information systems work together with higher level operational systems like ERP systems and with different databases. Because most of the commercial MESs have difficulties with the integration of more sophisticated dispatching and scheduling approaches, we suggest the use of scheduling systems in a plug-and-play manner via an object-oriented data layer. The data layer acts as a mirror of the base process, and its objects are updated in an event-driven manner (cf. Mönch et al. [190]).

Usually, an MES does not provide adequate dispatching and scheduling functionality. Therefore, we discuss several extensions of MESs for this purpose. These extensions are typically software systems on their own. We start with dispatching systems that are quite common in the semiconductor industry (cf. Pfund et al. [234]). We describe the main architecture of such a system and its interaction with both the MES and ERP systems. Scheduling systems are also discussed.

Software agents allow for the implementation of distributed planning and control algorithms. The agents are able to act autonomously; on the other hand, their communication abilities ensure a cooperative behavior and the fulfillment of global system goals. Furthermore, agent-based systems

facilitate maintenance and further development tasks of the software [319, 324]. In this book, we present the design and the architecture of a multi-agent-system for wafer fabs called FABMAS. The FABMAS system is aimed at production control of wafer fabs. We differentiate between decision-making agents and staff agents. Staff agents encapsulate scheduling logic. They support the decision-making agents in the course of their decision-making. The suggested architecture is used to implement the distributed version of the shifting bottleneck heuristic on a cluster of computers.

Manual material handling is typical for 200-mm wafer fabs. A 300-mm wafer usually visits several hundred machines to perform hundreds of different process steps. Modern 300-mm wafer fabs consist of several bays. To transport wafers, front-opening unified pods (FOUPs) are used as wafer carriers. The FOUPs have to be transported not only within a bay but also from one bay to another. Material control systems (MCSs) are used to initiate and coordinate concurrent movements of carriers within the automated material handling system (AMHS). Therefore, material handling is a critical issue in wafer fabs. An AMHS is an important tool to achieve the goal of reducing cycle time and improving yield rates (cf. Agrawal and Heragu [3]). Running and controlling an AMHS is challenging. Advanced software is required to run an AMHS that performs all the material handling requirements. We discuss the functionality of such MCSs for an AMHS and their interaction with the MES.

Wafer fabs, i.e., the front-ends, are geographically distributed over North America, Europe, and Asia, but most assembly and test sites, the back-end, are in the Pacific Rim. Thus, production orders have to be coordinated between the front-end and back-end facilities, and the management and coordination of the entire supply chain is an important issue. Supply chain management functionality is usually provided by advanced planning systems (APSs). We describe the main functionality of such systems. Furthermore, we also discuss the interaction of an APS with ERP systems and with the MES.

Chapter 2
Semiconductor Manufacturing Process Description

In this chapter, we provide a process description of semiconductor manufacturing. Therefore, we describe the front-end and back-end areas in some detail. We introduce the notion of base system, base process, control system, control process, planning system, planning process, and finally of the information system from systems theory. Then we discuss important wafer fabrication operations including a description of the most important characteristics of the semiconductor manufacturing process. We also introduce the notion of complex job shops because this is the way wafer fabs are organized. Finally, we discuss the production planning and control (PPC) hierarchy in semiconductor manufacturing.

From an operations management point of view, we differentiate between planning at the highest level, order release, scheduling, and finally dispatching at the lowest level. Each of these different functionalities is related to a certain horizon. Planning has a time horizon ranging from months to years. Order release takes place in a weekly or biweekly frequency. Scheduling is performed each shift or day. Finally, dispatching is carried out in a minute-by-minute manner depending on the speed of the material flow. We develop this PPC hierarchy because it forms the skeleton for the remainder of this monograph.

2.1 Semiconductor Manufacturing Overview

A semiconductor chip is a highly miniaturized, integrated electronic circuit consisting of thousands of components. Every semiconductor manufacturing process starts with raw wafers, thin discs made of silicon or gallium arsenide. Depending on the diameter of the wafer, up to several thousand identical chips can be made on each wafer by building up the electronic circuits layer-by-layer in a wafer fab. There are about 40 layers for the most advanced technologies. Next, the wafers are sent to sort or probe, where electrical tests identify the individual dies that are not likely to be good when packaged.

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5_2,

Historically, bad dies were physically marked so that they would not be put in a package. Today, this has been replaced by producing an electronic map to identify the bad dies. The probed wafers are sent to an assembly facility where the dies with a reasonable quality are put into an appropriate package. Finally, the packaged dies are sent to a test facility where they are tested in order to ensure that only good products are sent to customers. Wafer fab and sort are often called front-end, and assembly and test are often called back-end. While front-end operations are often performed in highly industrialized nations, back-end operations are typically carried out in countries where labor rates are cheaper.

Considering the scale of integration, the type of chip, the type of package, and customer specifications, the whole manufacturing process may require up to 700 single process steps and up to 3 months to produce. The four main stages of semiconductor manufacturing are shown in Fig. 2.1.

In the past, all that was necessary for a semiconductor company to make money was to design a good product. However, over the last decade, increased competition has required semiconductor companies to also be able to manufacture their products in an efficient and cost-effective manner.

Several performance measures are commonly used to describe and assess semiconductor manufacturing systems including machine utilization, production yield, throughput, cycle time, and on-time delivery performance-related measures. Machine utilization is extremely important because the machines account for around 70% of the cost of a new wafer fab, which can be as high as $5 billion US. In this context, cycle time is defined as the time needed for a lot of wafers, called a job, to travel through the semiconductor

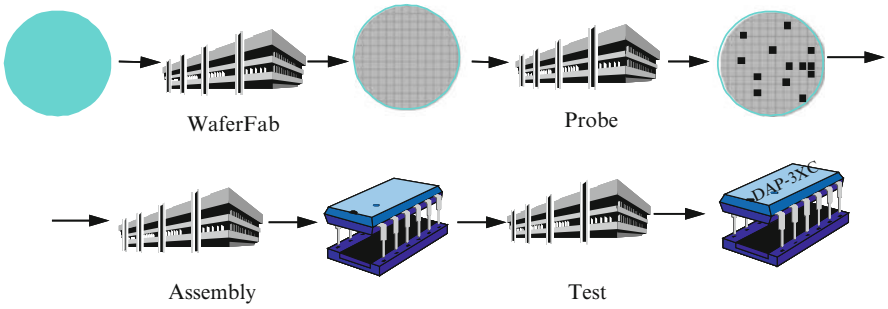

Figure 2.1: Stages of semiconductor manufacturing

manufacturing system including queue time, processing time, and transit time. Each job contains a fixed number of wafers. A high on-time delivery performance is important to satisfy customers. We also refer to Sect. 3.3 where these performance measures are introduced in a more formal way.

The competitiveness of a semiconductor manufacturer often depends on the ability to rapidly incorporate advanced technologies in electronic

products, continuous improvement of manufacturing processes, and the capability of meeting customer due dates. In a situation where prices as well as the state of technology have settled at a certain level, the capability of meeting due dates along with the reduction of cycle time has become the most decisive factor in the fierce competition in the global market place. Consequently, short and predictable cycle times are highly desirable.

Semiconductor companies have increasingly turned to data-intensive modeling and analysis tools and techniques because of their potential to significantly improve these performance measures, and hence the bottom line. The semiconductor manufacturing modeling and analysis community has been working over the last 20 years to modify general purpose manufacturing modeling tools and techniques to handle the intricacies and complexity of semiconductor manufacturing.

2.2 Front-End and Back-End Operations

In this section, we start by an overall framework for manufacturing systems. Then, we discuss the base system and finally the base process of semiconductor manufacturing.

2.2.1 Overall Framework for Manufacturing Systems

Before we describe front-end and back-end operations in detail, we present a framework that is used in the remainder of this monograph to discuss PPC problems in semiconductor manufacturing.

We start in a general manner with systems. A system consists of a set of interacting components. Each single component of a system has a state. We introduce processes to deal with dynamic aspects of systems. A process is defined as a mapping between a partially ordered set of events E and actions A, i.e., exactly one action $a \in A$ is assigned to each event $e \in E$. The partial order of the elements of E might refer, for example, to the points of time where the events happen, but it can also represent precedence constraints among the events. Typically, we describe the system and at the same time the corresponding processes. The actions of the processes are performed on system components.

In this monograph, we study manufacturing systems. Manufacturing systems are systems that have the purpose to produce goods. A manufacturing system consists of a base system (BS) and an information system (IS). The BS is formed by system components that are used to transform raw materials and intermediate products into final products. It contains a job processing system (JS) and a material flow system (MS) as subsystems. The JS consists of all the system components that allow for value-added processing of working objects, i.e., jobs. The system components of the JS offer capacity for processing. Resources, like machines, operators, and auxiliary resources, form the JS. On the other hand, all the facilities that are

necessary to store, transport, and supply raw material, working objects, and auxiliary, also called secondary resources, form the MS.

The base process (BP) is resonsible for the usage of system components of the BS by working objects. Of course, we can differentiate between subprocesses related to the JS and the MS, respectively. The BP is specified by process flows, also called routes, and a given set of working objects. A process flow in semiconductor manufacturing is a sequence of process steps. A set of possible machines is assigned to each single process step within a process flow. A recipe is an execution program at a certain machine that is associated with a process step.

The IS is responsible for the control of the production of goods. It is given by the planning system (PS), the control system (CS), and the operational system (OS). The PS consists of a set of computers and software that are used to determine production planning instructions *mp*. Production planning results in quantities and points of time for releasing working objects into the BS. The production planning process (PP) determines when and under which circumstances certain production planning actions have to be performed. Similarly, the CS is given by a set of computers and software that are used to determine production control instructions *mc* that influence the BP. As a consequence, production control decisions only have impact on working objects that are already part of the BP. The corresponding control process (CP) determines when and in which situations a certain production control algorithm is used to determine production control instructions.

Finally, the OS is responsible for immediate control of the BP. The OS usually consists of hardware and software to represent the state of system components of the BS and working objects of the BP. It acts as a mirror of the BS and the BP. Usually, databases are used to implement the OS. The PS, the CS, the OS, and the human decision makers form the IS.

The OS, the CS, and the PS interact by instructions and feedback. A more detailed description of these interactions is provided in Sect. 2.3. The different subsystems and subprocesses of a manufacturing system and the corresponding manufacturing process are summarized in Fig. 2.2.

We will use the notation from Fig. 2.2 when we describe the manufacturing system and process for semiconductor manufacturing. In Sects. 2.2.2 and 2.2.3, we start with the BS and the BP, whereas the PS, the PP, the CS, and the CP are discussed in Sect. 2.3.

2.2.2 Description of the Base System

In this monograph, we will mainly focus on modeling and analyzing of wafer fab operations. These operations generally account for more than 75% of the total cycle time and are also the largest component of cost. However, for the sake of completeness, we will also briefly discuss back-end issues.

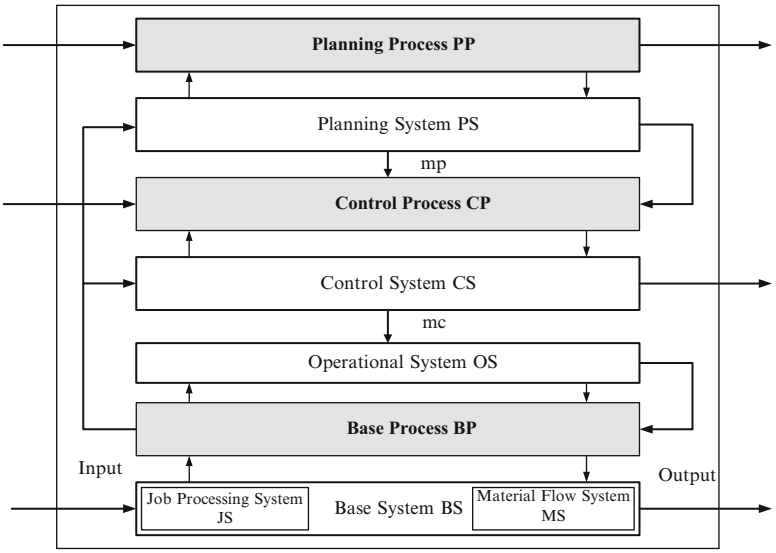

Figure 2.2: Subsystems and subprocesses of a manufacturing system and process

We start by describing the BS for semiconductor manufacturing. The entire enterprise may consist of several wafer fabs and back-end facilities, i.e., the BS of an enterprise is a collection of such factories.

In a first step, we will discuss the JS of a single wafer fab. The JS of a wafer fab consists of several work areas that are used for wafer processing and sorting in a clean-room environment. Each work area is a collection of work centers that are closely related logically or due to their location. Work areas are also called bays when the relation among the work centers is based on the location. A work center is a collection of machines that provide similar processing capabilities. Work centers are also called tool groups in semiconductor manufacturing. A single machine is a non-human resource with a fixed location that is able to process jobs. It can have a buffer where jobs are stored before, during, or after processing. For 200-mm wafer fabs, these buffers are often assumed to be practically unlimited, whereas they usually have a limited capacity in 300-mm wafer fabs. The following machine states are possible according to SEMATECH [280]:

- Productive state: a period of time during which the machine is performing its intended function
- Standby state: a period of time during which the machine is not operated, although it is in a condition to perform its intended function, and the chemicals and facilities are available
- Engineering state: a period of time during which a machine is in a condition to perform its intended function but is operated for the purpose of conducting engineering experiments

- Scheduled downtime state: a period of time during which the machine is not available to perform its intended function because of planned downtime actions
- Unscheduled downtime state: a period of time during which the machine is not available to perform its intended function because of unplanned downtime actions
- Nonscheduled state: a period of time during which the machine is not scheduled to be utilized in production including off-line training, unworked shifts, weekends, and holidays

Some of the machines are able to process only one wafer or a job at a certain point of time, whereas some types of machines process two jobs in an overlapping manner, i.e., the second job is already started after the first one has been processed for a certain period of time. These machines are called pipeline tools. Another example of specific machines that can be encountered in wafer fabs are X-piece machines, in which X wafers of a job are loaded at a time in the machines and where X is smaller than the total number of wafers in a job, except for small jobs. Other types of machines can process entire batches. A batch is defined in semiconductor manufacturing as a collection of jobs that are processed at the same time on the same machine (cf. Mathirajan and Sivakumar [176]). Besides batch-processing machines (for short, batch machines in the remainder of this book), we also introduce cluster tools that are typical for many modern wafer fabs.

Cluster tools are special integrated tools for wafer processing in semiconductor manufacturing (cf. Lee [157]). They are used to maximize quality performance at the cost of very complex behavior. Since wafers with different types of process steps can circulate in a cluster tool simultaneously, it can be regarded as a fully automated machine environment. Cluster tools work under vacuum conditions inside the tool, which means very few particles could possibly contaminate wafers. As a consequence, the clean-room quality outside the cluster tool is allowed to be lower than in traditional wafer fabs. The basic components of a cluster tool are as follows:

- A vacuum mainframe with one or two wafer-handling robots
- Two (or more) load locks to pump to vacuum or vent to atmospheric conditions
- Several processing chambers, where some of them can be dedicated to identical processes and hence used in parallel
- Optional transfer chambers if there is more than one wafer-handling robot
- An equipment front-end module (EFEM), which is attached to the load locks, with an atmospheric wafer-handling robot and several load ports

These basic components of a cluster tool are shown in Fig. 2.3. The processing of wafers on cluster tools is included in the description of the semiconductor BP later in Sect. 2.2.3.

In wafer fabs, we typically have dozens of different work centers and several work areas. A more detailed description of the functionality of work

centers and work areas is also included in the BP description in Sect. 2.2.3. For each machine, there is a list of which products are allowed to be performed on the machine because of quality considerations, i.e., we find machine dedications. They are mainly influenced by the fact that qualifying a certain machine for a specific process is time-consuming and therefore expensive (see Johnzén [133]). The machines of a work center are heterogeneous because they tend to have a different age. A total of several hundred machines can be found in most wafer fabs. Machines are expensive, ranging in

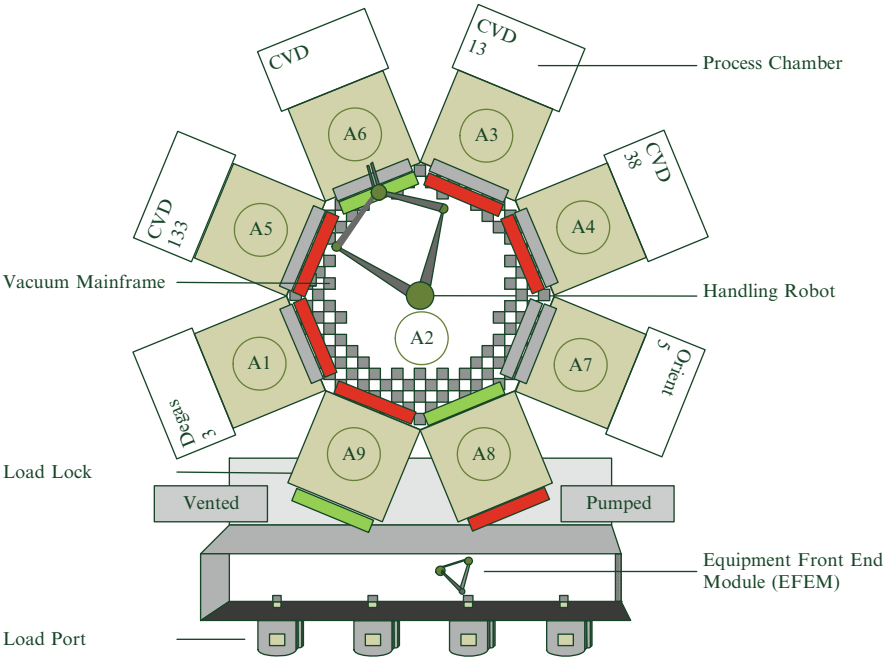

Figure 2.3: Basic components of a cluster tool

price from a couple of US$100 thousand to over US$40 million per machine. Note that in a few cases there are borderless wafer fabs, i.e., facilities that are in close geographical proximity (cf. Gan et al. [93]). In this situation, the different wafer fabs can share some of their machines to produce ICs.

Back-end facilities also consist of work areas and work centers, but the number of machines is usually smaller than in wafer fabs. Furthermore, the clean-room conditions that are required are less strict than those for wafer fabs.

Operators as special human resources are necessary to run a wafer fab. In all but the most highly automated wafer fabs, humans load and unload wafers to machines, run the machines, and perform inspection steps. Often highly qualified and experienced operators are assigned to supervise work areas. In addition, there are generally several people responsible for the overall performance of the work areas; these people may spend the majority of their time outside the shop floor.

Typical auxiliary resources used in semiconductor manufacturing are reticles. A reticle is a photo mask that is a carrier of an IC pattern. They are used in the photolithography process. In back-end facilities, load boards are applied to load and place chips into burn-in ovens to subject them to thermal stress. The type of board required to test chips depends on the packaging of the circuits. Both reticles and load boards are quite expensive, and therefore their number is often limited. In a certain sense, operators might also be considered as an auxiliary resource that is necessary to process wafers on machines.

After the description of the JS, we discuss the MS of a wafer fab [82, 131]. In modern 300-mm wafer fabs, wafers and reticles are transported fully automatically in carriers called FOUPs, using an AMHS. A FOUP is a container that holds up to 25 wafer jobs of 300-mm wafers in an inert, nitrogen atmosphere. Automated material handling is always a critical operation in wafer fabs [3, 209]. In many 200-mm wafer fabs and in most back-end facilities, material handling is usually carried out manually. Note that machines are the main resources in the JS while vehicles take over this role in the MS. Jobs compete for the scarce machine and vehicle capacities.

We now need to differentiate between interbay systems and intrabay systems. Interbay systems are used to store and transport wafers or reticles between the various bays of a wafer fab. On the other hand, intrabay systems have the purpose to move carriers for wafers and reticles within a bay. In 200-mm wafer fabs automated interbay systems are common, but the transport within a bay is done manually. The automated transport in 300-mm wafer fabs also covers transport within a bay. This is caused by the increase in area and weight of the wafers. Furthermore, because of more advanced software and hardware solutions in 300-mm wafer fabs, a better integration and control of the interaction between machines and automated transportation systems is possible.

We start with the interbay situation. The interbay MS consists of carriers for wafers and reticles, stockers, and the transportation system itself. A stocker is an automated high-rack storage area where wafers and reticles can be stored before and after being processed. A robot moves the carrier into a shelf when it is inside the stocker. There are different

possibilities for stocker placement within a wafer fab. Often a single stocker is assigned to exactly one bay by locating it near the entrance of a bay. A discussion of different AMHS layouts can be found in Agrawal and Heragu [3]. Each stocker may contain several input/output stations, called load ports. Load ports are used for manual or automated load and unload operations of carriers by fab-operating personnel or the automated transportation system, respectively. If processing of wafers is done on different levels, i.e., floors, of a wafer fab, then interlevel lift systems are used to transport carriers between these multiple levels. The transport system transports carriers between different bays of a wafer fab. Overhead monorail-tracked vehicle-type systems are in widespread use in industry. When an overhead transport is not practical, then a floor running automated guided vehicle (AGV) system is typical for interbay transportation as shown by Foster and Pillai [82]. Note that interbay transportation takes place from stocker to stocker or from stocker to an interlevel lift system and vice versa.

In contrast, in the intrabay situation, the transportation is carried out between stockers and machines or directly between machines of a bay. Floor-based transport systems like AGVs and rail guided vehicles (RGVs) have been used in industry. In many 300-mm wafer fabs, ceiling-based overhead hoist vehicles (OHVs) are run instead of AGVs or RGVs. This type of transportation is called overhead hoist transportation (OHT). Stockers are equipped with intrabay input/output ports to allow carriers to enter and exit the intrabay system. Stockers are usually far away from a specific machine where a job is needed. This can lead to long delivery times and under track storage (UTS) is proposed to avoid this disadvantage (Fischmann et al. [80]). UTSs are single buffer storages that are mounted overhead but under the track system. They are passive shelves that do not require floor space in the clean room. OHVs can place carriers for temporary buffering in route to their destinations. They provide additional queue positions for high throughput machines (Foster and Pillai [82]). Load ports at the machines are the primary buffers besides stockers and UTSs. A single machine typically has three to four ports. The intrabay vehicles deliver unprocessed FOUPs to the ports and pick up finished wafers. A single OHV can transport a carrier directly from machine load port to machine load port.

Operators are often elements of the nonautomated part of the MS. They drive manual carts or personal guided vehicles (PGVs), especially in 200-mm wafer fabs or in ramp-up situations where the automated transport system is not running in 300-mm wafer fabs.

Two intrabay system configurations are in current use in industry. We differentiate between a unified transport configuration and a non-unified one. In the unified configuration, the track network for interbay and intrabay is connected directly, i.e., no load and unload at stockers are necessary to decouple the two transport loops. The track elevations of the interbay and intrabay system have to be the same.

In the second configuration, stockers are used to exchange the load of the vehicles between interbay and intrabay segments. The track elevations of the interbay and intrabay systems can be different. Within each bay, intrabay vehicles are in place to transport carriers from the bay stocker to the storage facilities of the bay. On the other hand, interbay vehicles transport carriers to the destination bay stockers. We finish this brief description of the MS by making the comment that often the MS is as complex and difficult to control as the JS and a source of problems in many wafer fabs.

2.2.3 Description of the Base Process

We continue by describing the BP of a wafer fab. Many authors have discussed the difficulties of the semiconductor manufacturing process (cf. Wein [318], Uzsoy et al. [306], Atherton and Atherton [14], Ovacik and Uzsoy [223], Sze [294], Sarin et al. [274], among others). Up to now, work areas are identified as an important component in the JS hierarchy. Now we describe the basic process steps, i.e., the operations, that can be performed in different work areas. The following process steps have to be performed in a wafer fab after starting the raw wafer [14, 122]:

1. Oxidation/diffusion: A layer of material is grown or deposited on the surface of a cleaned wafer. Oxidation aims at growing a dioxide layer on a wafer. Diffusion is a high temperature process that disperses material on the wafer surface. Diffusion furnaces and rapid thermal processing equipment are in place at the oxidation/diffusion work area. The furnaces are typical batch machines.
2. Film deposition: Deposition is used to deposit films onto wafers. The corresponding steps deposit dielectric or metal layers. There can be a dozen or more such deposition layers in an advanced circuit. Deposition can be executed by different processes, such as physical vapor deposition (PVD) or chemical vapor deposition (CVD), epitaxy, or metalization.
3. Photolithography: Coating, exposure, developing, and process control are the main steps of the photolithography process. In the first step, the wafer is coated with a thin film of a photosensitive polymer, called photoresist strip. Accurate and precise three-dimensional patterns are produced on the silicon wafer's surface when an IC pattern is transferred via a photo mask, i.e., reticle, onto the photosensitive polymer, which replicates the pattern in the underlying layer. Exposure tools, called steppers, transfer the pattern onto the wafer by projecting light through the reticle to expose the wafer using ultraviolet light. The exposed wafer is then developed by removing polymerized sections of photoresist from the wafer. Every wafer passes through the photolithography area up to 40 times because the circuits are made up of layers. The photolithography work area is a typical example of a bottleneck in a wafer fab because steppers are very expensive machines.

4. Etch: This step is responsible for removing material from the wafer surface. The wafers are partially covered by photoresist strip after the photolithography step. Areas on the wafer that are not covered are then removed from the wafer. We differentiate between wet and dry etching. In the first case, liquids are used, whereas gases are necessary for the latter case.
5. Ion implantation: Dopant ions are selectively deposited on the surface of the wafer. Doping material is deposited where parts of the wafer have been etched. Ion implanters are used for between four and eight applications for most modern ICs.
6. Planarization: This step cleans and levels the wafer surface. It is called chemical-mechanical polishing (CMP). A chemical slurry is applied to a wafer and the surface is equalized. This results in the thickness of the wafers being diminished before adding a new layer.

Before the wafers are entered into the oxidation/deposition/diffusion work area, a cleaning step is performed. Several inspection and measurement steps are necessary to control the processes within and between work areas. Inspection machines can be found in all work areas.

At certain process steps, it can happen that jobs, wafers, or dies are processed in a way that they become damaged. In some situations, rework is possible to repair the wafer. When rework is not allowed, the useless wafers are called scrapped material. The yield is the percentage of dies that meet their electrical specifications.

Wafer fabrication has a number of unusual facets that are described below. In a typical wafer fab, there often are dozens of process flows. Products that follow the same basic process flow are often said to be of the same technology. Typically, the only differences among products in the same technology are the photolithography reticles used. Individual products are often referred to as devices. Depending on the number of different products, or product variety, wafer fabs can be classified as low- or high-mix wafer fabs. In low-mix wafer fabs, machines can be dedicated to products, whereas, in high-mix wafer fabs, the same machine can be shared by many products of various technologies, i.e., requiring different setup and processing times. Hence, production control is more complex in high-mix fabs, and efficient production control is usually more critical.

Each process flow contains 300–700 process steps on more than one hundred machines. The economic necessity to reduce capital spending dictates that such expensive machines be shared by all jobs requiring the particular processing capabilities provided by the machine, even though they may be at different stages of their manufacturing cycle. This results in a manufacturing environment that is different in several ways from both traditional flow shops as well as traditional job shops. A job shop is characterized by an individual process flow of each single product using different machines, whereas a flow shop is described by the fact that all products have a fixed machine sequence, i.e., the jobs are processed on the same sequence of machines or work centers. The main consequence of the reentrant flow nature is that wafers at different

stages in their manufacturing cycle have to compete with each other for the same machines.

A simplification of the typical reentrant flow of a wafer fab is shown in Fig. 2.4 where the different work areas within a wafer fab are also summarized.

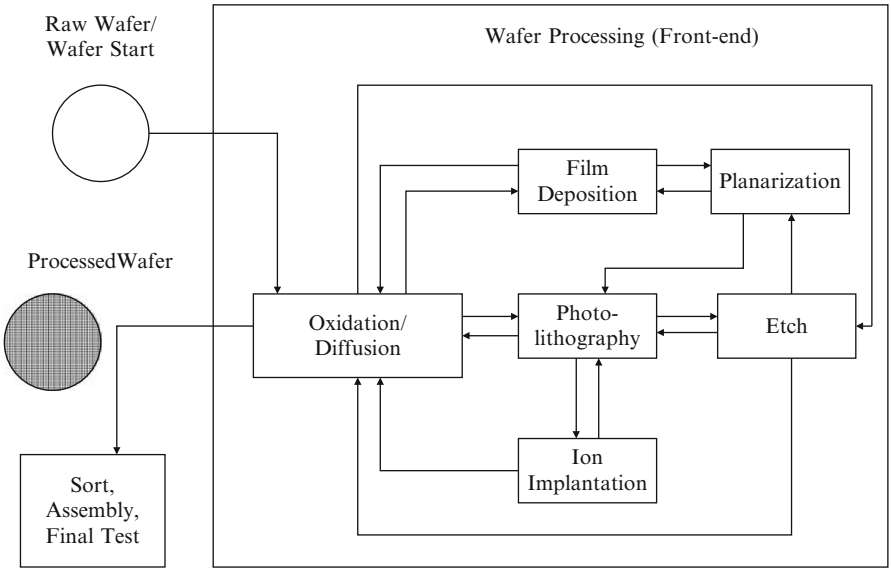

Figure 2.4: Operations in a wafer fab

Furthermore, the nature and duration of the various process steps in a process flow differ significantly. Some process steps require 15 min or less to process a job, while others may require over 12 h.

Many of these long operations involve batch processes. In reality, it is not uncommon for one-third of the wafer fab operations to be batch operations. Batch machines tend to off-load multiple jobs, 1 to 12, onto machines that are capable of processing only one job at a time. This leads to the formation of long queues in front of these serial machines and ultimately to a nonlinear flow of products in the wafer fabs. The diffusion furnaces in wafer fabs are an example of batch machines. Here, the jobs are assigned to incompatible job families. While several jobs can be processed at the same time, jobs of different families cannot be processed together due to the chemical nature of the processes. The processing time of all jobs within a family is the same.

The combination of decreased line widths and more area per wafer in 300-mm wafer fabs results in fewer wafers being needed to fill an IC order of a customer. Each wafer fab will have only a limited number of FOUPs as they are expensive. A large number of FOUPs have the potential to cause the AMHS to become overloaded. In addition, some machines have the same processing times regardless of the number of wafers in the batch. Thus, it

is not reasonable to assign an individual FOUP to each order. Therefore, 300-mm manufacturers often have the need and the incentive to group orders from different customers into one or more FOUPs to form production jobs. We refer to this as the multiple orders per job problem (cf. Mönch et al. [207]).

Historically, equipment reliability has been a major source of uncertainty in wafer fabs. The continual drive to reduce the size of ICs has led to the adoption of machines that have innovative processing modes but have been built by manufacturers without a long history of developing manufacturing equipment. The failure of equipment or processes is often not a hard failure in the sense that something obviously breaks or goes wrong but rather a soft failure in which the equipment begins to produce out of the tolerance region. Due to the nature of the product and process, one may not detect this fact for some time. Therefore, many inspection steps are added to the process flow. The large downtime of some wafer fabrication machines, for example, ion implanters may be down 30–40% of the time, has significant impact on the production control function. The probabilistic occurrence of long machine failures results in large variability in the time a job spends in process. High variability in cycle times prevents accurate prediction of production cycle times, resulting in longer lead-time commitments.

Preventive maintenance operations are used to reduce the number and the duration of machine failures. But at the same time, they reduce machine capacity. There is also a competition between production jobs and prototype jobs for processing times on the machines. Many prototype/engineering jobs are necessary because of the difficulty of the technological processes. Prototype jobs also consume machine capacities.

Often, certain jobs in a wafer fab are more important than others. Then these jobs will be expedited to meet their due dates. They are called hot or rocket jobs. Because of these hot jobs, the congestion of many wafer fabs is further increased.

Time constraints between consecutive process steps are another important restriction. For example, there is often a time restriction between operations in the etch work area and oxidation/diffusion work area (cf. Scholl and Domaschke [275]). Time windows are installed by the process engineering department to respect the time constraints. This is important to prevent native oxidation and contamination effects on the wafer surface. More than two consecutive process steps might be involved, and these time constraints might be nested. Jobs with a violation of the recommended time windows often have to be scrapped since rework is generally not allowed.

Sequence-dependent setup times occur in some work areas and are related to changing temperature, gas pressure, metal composition, etc. It is not only important which product is going to be processed next but also which was the last product processed before the current one. For example, in the ion implantation work area, dopants have to be changed frequently. The effort to do this change depends on the predecessor dopant. If the setups are not

treated correctly by the production control staff, the corresponding machines can become bottlenecks.

Finally, some process steps require an auxiliary resource in order to process the job. For example, reticles are required in photolithography to process wafers on steppers. Therefore, the challenge is to ensure that the machine and the auxiliary resource are available at the same time.

Next, we consider cluster tools as they cause specific processing restrictions. Each wafer of a job has to undergo the same process steps in the cluster tool. This sequence of process steps is usually referred to as a recipe. A typical product flow in a cluster tool starts with loading jobs into one of the load ports. After that, single wafers are consecutively transferred from the load port to the load lock by an atmospheric robot. Then the load lock will pump to achieve vacuum conditions. Next, the mainframe robot can transfer the wafer to its destination chamber where it will be processed. The next step depends on whether the wafer shall leave the system or is required to be processed in another chamber according to its recipe. After the last process step, the wafer will be guided through the load lock back to the load port. With more than one load port occupied, the controller of the cluster tool will always process jobs of the same recipe sequentially one after another and jobs of different recipes in parallel. Usually, the mainframe robot is a dual blade robot with the two blades either on the same side or opposite to each other. Advantages compared to single blade robots are reduced wafer transfer times and, with regard to multiple product flows, a reduced amount of possible deadlocks as well.

Cluster tools can be seen as a collective term for a certain type of machine with a wide range of varying configurations. The number of load ports varies usually between two and four, an EFEM may not be given, and loading stations are directly attached to the load locks. Some cluster tools have two mainframes with a transport robot and transfer chambers to connect them.

Cluster tools can process a certain number of jobs in parallel, which is determined by the number of given load ports at a certain tool. The logic of job processing requires jobs of different recipes to be processed in parallel and jobs of the same recipe to be processed sequentially one after another. Through parallel job processing, resource conflicts arise that result from a shared use of the handling robots, load locks, and process chambers. The conflicts lead to a job processing time that will be increased compared to its stand-alone processing time.

Another effect that needs to be considered is called pipelining, which implies that two jobs of the same recipe need to be processed sequentially. The recipe is such that each wafer needs to go through more than one process chamber. If this is the case, the first wafer of the second job may already start processing in the first chamber if the last wafer of the first job is finished and

no wafer of this job will return to the chamber. Hence, there will be a certain overlap time between the start of the second job and the completion of the first job.

In summary, concluding the description of BS and BP, wafer fabs can be considered as complex job shops [172, 223]. A job shop is called flexible when the same processing capabilities are offered by more than one machine, i.e., machines are run in parallel in the work centers. According to Mason et al. [172], a flexible job shop is called complex when the following processing conditions appear:

- Unequal release dates of the jobs
- Sequence-dependent setup times
- Prescribed due dates of the jobs
- Reentrant flows of the jobs
- Different types of processes, for example, single job vs. batch processing
- Frequent machine breakdowns and other types of disturbances

We do not describe the part of the BP that is related to the MS because it is simpler and it will not be covered in detail in the remainder of this book.

Finally, we briefly consider sort operations to complete the description of the BP of the front-end part of semiconductor manufacturing. The sort operations are performed within the inspection work area. Inspections for defects, film composition, critical measurements, and wafer profiling are performed on automatic inspection stations.

Next, we briefly discuss back-end operations. We differentiate between assembly and final test operations that are performed in the corresponding facilities (Hutcheson [122]):

1. Assembly: In the main assembly work area, typically dicing saw, die attach, wire bonding, and optical inspection operations are performed. Packaging, molding, lid sealing, and environmental testing are usually carried out in work areas that need less strict clean-room conditions.
2. Final test: A series of electrical tests similar to those in wafer sort is performed for the individual ICs to make sure that they meet complex specifications. A heat-stress test of ICs is performed in burn-in ovens. Before the ICs can be processed in the burn-in ovens, they have to be inserted onto a load board. They are kept at a specific temperature for a certain period of time. Then they are packed into tubes and delivered to customers.

There are usually more product types being made in an assembly factory than in a wafer fab, but each product type requires 10–30 steps instead of 300–700 in wafer fabs. One difficulty of these operations is the fact that a job is often divided into subjobs with each subjob being sent to the next machine when it completes an operation. Thus, one job may be being processed across several machines at different steps at the same time.

Another difficulty is that there is often a very significant amount of setup required to change over from one product type to another. Yet another difficulty is that a single type of wafer can become one of several different products based on tested levels of performance, for example, speed. This process is called binning. Finally, batch machines are also often present in assembly factories.

Test operations have several problems that are difficult to treat. First, the sequence of test operations and the test times are not always fixed. These can be changed based on recent product yields or maturity of a product. Second, there are two major types of equipment used in test operations. These are the test system itself, called the tester, and the loading mechanism, called the handler. The tester may have a single or multiple test heads connected to it. The interactions between the tester, the test heads, and the handler can be quite complex. There can be significant sequence-dependent changeover times. The burn-in operations are another example for batch processes in semiconductor manufacturing. In contrast to the diffusion furnaces in wafer fabs, the processing time associated with such a batch is determined by the longest processing time of one of the jobs that form the batch.

In the remainder of this monograph, we will primarily restrict ourselves on the PPC problems found in a single wafer fab. However, in a few situations, we will also discuss PPC problems related to the back-end stage. When we discuss planning approaches in Chap. 7, we consider also the case of simultaneous planning approaches for several wafer fabs or back-end facilities, i.e., we work also on the enterprise level.

2.3 Production Planning and Control Hierarchy

In this section, we discuss the PS and the CS of wafer fabs and also make some comments on the PP and the CP (cf. Sect. 2.2.1 for this notation). The resulting hierarchy forms the starting point for the remaining chapters of this monograph and determines their sequence to a certain degree.

We start by describing the typical decisions that have to be made by the PS and the CS. Planning is performed with a time horizon ranging from months to years. Anticipated demand is an important input for any production planning approach. Planning usually assumes that the time horizon is divided into time buckets with a length of a week or a month. All the planning decisions are related to these time buckets. The results of a typical planning decision are the quantities that have to be released or completed within a certain bucket in such a way that certain performance measures are optimized and the finite capacity of the manufacturing system at an aggregated level is taken into account. Typically, the revenue is considered as a performance measure on the planning level. Certain cycle time assumptions are also the basis for planning decisions. In semiconductor manufacturing, we differentiate between long-term capacity planning that is more strategic and master planning, also called supply network planning, that is more operational. While

capacity planning usually determines the quantities and the product mix for the next years on the enterprise level, master planning has a horizon of several months and assigns quantities to time buckets and also to specific facilities (cf. Vieira [313]).

The PP can be performed in a time- or event-driven manner. Hybrids between these two extrema are also possible. In the time-driven situation, plans are basically determined in a rolling horizon setting. Each planning decision is made for $h := \tau_\Delta + \tau_{ah}$ time buckets. But the planning decisions will be implemented only for the next τ_Δ time buckets in the BS and the BP. After τ_Δ time buckets, a new plan with time horizon h will be determined taking the current state of the BS and the BP into account. The quantity τ_Δ is called the planning interval, whereas τ_{ah} is the additional planning horizon. The event-driven approach initiates the determination of new plans with time horizon h as a consequence of certain changes of the BS or events within the BP.

Order release is at the interface between the planning and the control level. It refines the decisions made on the planning level by disaggregating the quantities in time and space. A typical order release decision results in a set of jobs that have to be launched into a wafer fab at a certain point in time. Weekly or biweekly order release schemes are very common in semiconductor manufacturing. The order release scheme definitely affects the load and consequently the cycle times in wafer fabs. Therefore, order release also influences the planning decisions.

Scheduling is defined as the process of allocation of scarce resources over time [34, 240]. The goal of scheduling is to optimize one or more objectives in a decision-making process. Scheduling can be performed for jobs on single machines, work centers, work areas, and finally for all machines of the shop floor, i.e., for the BS. At the same time, scheduling decisions are also made for the vehicles in the MS. The result is a schedule, i.e., an assignment for each job to at least one time interval on the different resources, i.e., machines or vehicles. Scheduling is usually done with a horizon of one shift or one day. Scheduling is only performed for jobs that have been already released into the BS. As in case of the PP, the CP is basically given by the used rescheduling policy. Similar to the planning case, we can differentiate between time- and event-driven rescheduling schemes (Vieira et al. [314]).

Dispatching is the activity to assign the next job to be processed from a set of jobs awaiting service on an available machine in the JS or a free vehicle in the MS [29, 116, 274]. To select the next job, each job is assigned a priority. These priorities can be determined using a schedule. When there is no feasible schedule available, the priorities can be determined by dispatching rules. The priority for each waiting job is calculated by taking different job and resource attributes into account. Dispatching is on the lowest level in the PPC hierarchy. It is performed in a minute-by-minute manner. We show the described PPC hierarchy in Fig. 2.5.

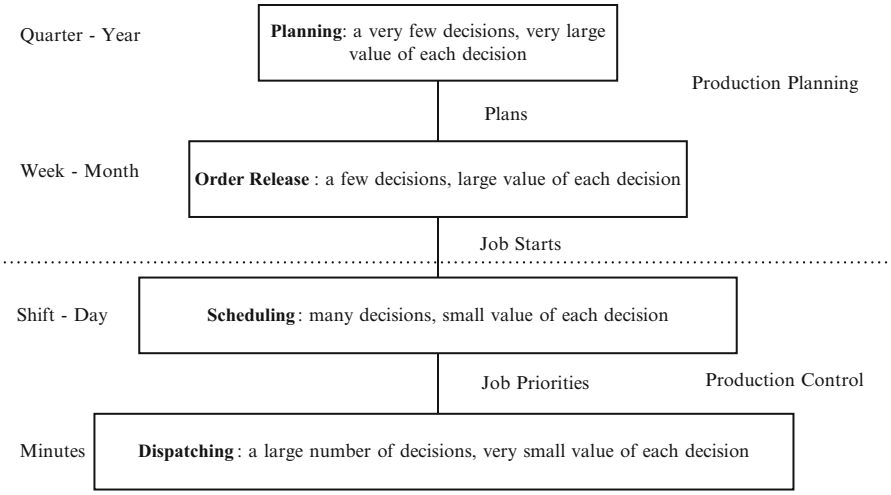

Figure 2.5: Production planning and control hierarchy

In Fig. 2.5, we do not depict the interactions between the different levels in detail. As described in Sect. 2.2, the automated parts of the PS and the CS consist of software that is used to implement the decision-making algorithms and hardware. For the sake of completeness, we will briefly mention the different application systems that are related to PPC decisions in semiconductor manufacturing.

The OS is given by transactional systems like the ERP system or the MES. They are used to gather data from the BS and BP. The ERP system is sometimes also used to support planning decisions. Because of the known shortfalls of ERP systems with respect to planning, often APS or other specialized software systems are in place to support planning decisions of managers. The MES typically contains more fine-grained data. Hence, it can be used to support production control decisions related to the JS. Besides the MES, MCSs are used to control the AMHS and support MS-related decisions. Again, because of the shortfalls of MESs, often more specialized software solutions are applied in wafer fabs [234].

Chapter 3
Modeling and Analysis Tools

Modeling and analyzing problems from semiconductor manufacturing always require a solid knowledge of appropriate decision methods. Models are used within the production planning and control process for representing the BS and BP and for decision-making. Decision methods usually come from the areas of operations research (OR), artificial intelligence (AI), and computer science (CS). They are important prerequisites to solving decision problems.

In this chapter, we begin with a brief discussion of general systems and models. We will then describe several types of models that are used in the remainder of this monograph. Models are important to identify appropriate decision methods. We will discuss very briefly linear programming and mixed integer programming (MIP), stochastic programming, branch-and-bound techniques, dynamic programming, metaheuristics, queueing theory, and discrete-event simulation. The main ingredients of discrete-event simulation models of wafer fabs are presented in some detail.

After the development of a decision method, the question is raised of how good is the method in various situations. Therefore, we also deal with basic questions of performance assessment. We start by introducing important performance measures used in the remaining chapters of this monograph. A simulation-based method to assess the performance of a production planning and control system within a dynamic and stochastic environment is described. The content of this chapter cannot compensate for a deeper study of more specialized textbooks in OR, AI, and CS. However, in order to be as self-contained as possible, we summarize the main ideas of modeling and decision-making in this chapter.

3.1 Systems and Models

In this section, we describe how systems can be represented by models. Furthermore, we discuss different types of models.

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5_3,

3.1.1 Representation of Systems by Models

In Chap. 2, we discussed which systems and processes are related to semiconductor manufacturing. In this section, we generalize the notion of systems slightly in order to introduce the notion of models.

A system S is given by a set of components V. The components and their associations provide the structure of S. The behavior of a system is described by the interaction of the system components. An interaction is given by an information exchange or an exchange of material or energy. We call S open when it interacts with its environment. When such an interaction does not occur, the system is a closed one. An input–output system is a system that totally hides the inside view on the system components. Input–output systems are also called black-box systems (cf. Mesarović and Takahara [181]).

A real system is a certain part of the real world. The components of real systems are physical. The BS of a wafer fab described in Chap. 2 is an example of a real system. Often, we are interested only in certain aspects of a real system, and then it makes sense to work with a representation of the original system. These representations are called models.

Now, we will continue with a more abstract view of models. A model is defined formally as a triplet

$$M := (S_O, S_M, f), \tag{3.1}$$

where we denote the original system by S_O and the model system by S_M. The set of system components of S_O is denoted by V_O. The notation V_M is used for the system components of S_M. The function

$$f : V_O \to V_M \tag{3.2}$$

is called the model mapping. We are interested in models that have a high fidelity related to structure and behavior. Very often, it is not possible and even not necessary to describe f explicitly. The described situation is shown in Fig. 3.1.

There are goals that have to be achieved when creating a model. These goals are used to identify the parts of the real world that have to be modeled and interpreted in an appropriate way. The level of detail in modeling is determined by the goals. Usually, a set of model parameters have to be selected to describe the components of a model and their interactions. In doing so, it is important to remember the famous words of George Box, “All models are wrong, some are useful.” [31].

In case of a wafer fab, the type and number of machines, their characteristics, the structure of the process flows, and the job release rate are parameters of a model of the wafer fab.

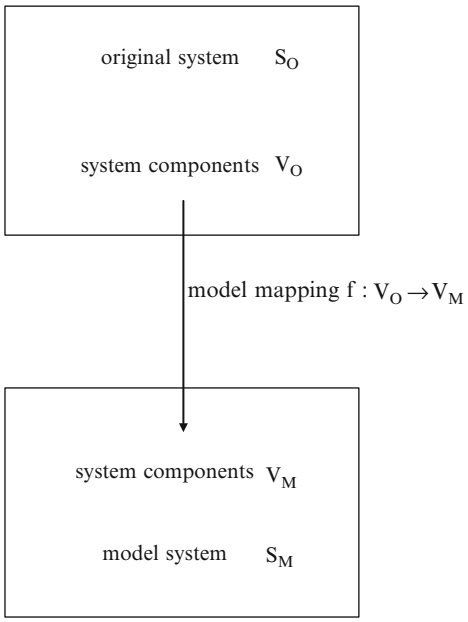

Figure 3.1: Relationship between original and model

3.1.2 Types of Models

There are different kinds of models (cf. Turban et al. [299]). Descriptive models are used to depict the components of a system and their relationships. They describe how a system behaves, but they do not explain the behavior of a system or allow for prognoses of real activities.

Prescriptive models select one or more actions among a set of alternatives. This decision is based on specified criteria. Optimization models are a typical representation of prescriptive models. In the case of these models, optimality criteria are used to select the best alternative. An optimization model consists of objective functions to be optimized and constraints that have to be satisfied. A solution of an optimization model is called feasible if it satisfies all constraints.

We also differentiate between static and dynamic models. Static models are related to a certain snapshot in time of a specific situation. Dynamic models are time-dependent. They represent scenarios that change over time. Dynamic models have the advantage that they represent the development of the system over time.

Furthermore, in the rest of this monograph, we develop deterministic and stochastic models. All model parameters are assumed to be known with certainty in a deterministic model. In contrast, a stochastic model contains certain model parameters that are described by probability distributions.

Because of the efforts to build a model and to maintain it, the simplest model that answers the question is the best. While it is generally true that prescriptive, dynamic, and stochastic models are more complicated than descriptive, static, and deterministic models, we prefer models with the latter characteristics if they are sufficient to answer the given questions. For example, a descriptive, static, and deterministic model implemented in a spreadsheet is often good enough for rough cut capacity planning.

In the next section, we discuss various decision methods that can be used in prescriptive models to select actions among alternatives. We also will describe simulation models as a main ingredient for simulation-based decision-making.

3.2 Decision Methods and Descriptive Models

In this section, we discuss various optimal and heuristic approaches that will be applied in the remaining chapters of this monograph to make decisions. Furthermore, we present some descriptive models that are useful.

3.2.1 Optimal Approaches vs. Heuristics

A decision problem consists of a set of feasible actions or sequences of actions where we have to select a particular action or sequence of actions that achieves certain objectives in the best possible way according to specified criteria. We call an action or a sequence of actions feasible when it fulfills all the requirements. The formal representation of a decision problem is called a decision model or an optimization model. In its simplest form, a decision model contains a set of alternative feasible actions and an objective function to assess them.

The notion of decision methods is closely related to decision models. Decision methods can be used to determine feasible or even optimal solutions for a certain decision problem. We introduce several decision methods in the remainder of this section in a rather generic way. Various concrete decision problems related to semiconductor manufacturing will be discussed in the remaining chapters of this monograph.

We are interested to find efficient algorithms to solve our decision problems (cf. Kleinberg and Tardos [144]). We denote by $G(n)$ an upper bound of the running time of an algorithm, where we denote by n the size of the input of the algorithm. The function $G(n)$ grows on the order $O(g(n))$ if

$$\lim_{n\to\infty} \frac{G(n)}{g(n)} = c \tag{3.3}$$

holds, where c is a positive constant and $g(n)$ is a given function. We are interested in determining $O(n^k)$ algorithms, where k is a fixed non-negative number. Algorithms of this class are called polynomial-time algorithms. If we cannot find a polynomial-time algorithm, then it is useful to check whether

it can be proved that the problem is NP-hard or not. Roughly speaking, NP-hardness means that it is unlikely that an efficient algorithm exists that will be guaranteed to solve the problem to optimality. Very often, this can be done by showing that special cases of the problem can be transformed with polynomial effort into a known NP-hard problem (see Garey and Johnson [94]).

We differentiate between optimal decision methods and heuristics. Heuristics are used to determine good solutions for NP-hard problems, but do not necessarily provide optimal solutions. Heuristics are approximation algorithms that are used to solve NP-hard problems. The majority of the decision problems discussed in this monograph are NP-hard. Therefore, we mainly focus on efficient heuristics. However, it is generally useful to know optimal decision methods, because we can exploit them to assess the performance of heuristics for small-size problem instances. On the other hand, often very efficient heuristics can be derived from combining optimal decision methods with heuristics.

In the remainder of this section, we discuss several optimal decision methods and heuristic algorithms that will later be used to tackle production planning and control problems in semiconductor manufacturing.

3.2.2 Branch-and-Bound Algorithms

A branch-and-bound algorithm is an enumerative procedure for solving discrete optimization problems optimally (see Brucker and Knust [35]). Let us consider for the sake of simplicity the following maximization problem, which can serve as a model problem.

(P) Find a feasible solution $s^* \in S$ with

$$f(s) \leq f(s^*) \tag{3.4}$$

for all $s \in S$, where we denote by S a finite set of feasible solutions and f is a real-valued objective function. Note that we can focus on maximization problems without loss of generality because we can tackle minimization problems by the same approach by simply maximizing $-f$.

The notion of subproblems is important for branch-and-bound schemes. A subproblem is a subset $S' \subseteq S$. We explain the three main ingredients of a branch-and-bound scheme as follows:

- Branching: The problem S is replaced by a set of subproblems $S_i \subseteq S$, $i = 1, \ldots, r$ with the property $\bigcup S_i = S$. This decomposition process of subproblems is called branching. The branching procedure is recursive, i.e., each subset S' can be decomposed in a similar way. We obtain a branching tree with root S and children S_i. An example for a branching tree is depicted in Fig. 3.2.
- Upper bounding: An upper bounding scheme is responsible for calculating an upper bound $UB(S')$ for the objective function value of a subproblem S'.

- Lower bounding: The objective function value of an arbitrary feasible solution $s \in S$ provides a lower bound L for problem (P). When for a subset S' the relation $UB(S') \leq L$ holds, then S' cannot provide a better solution for problem (P). Therefore, we do not need to continue the branching process from the corresponding node of the branching tree. The value of L has to be as large as possible to avoid a large number of branching steps. After some branching steps, we may reach a situation where a subproblem S'' contains only one feasible solution s. We obtain $UB(S'') = f(s)$. When $UB(S'') > L$ is valid, we replace L by $UB(S'')$.

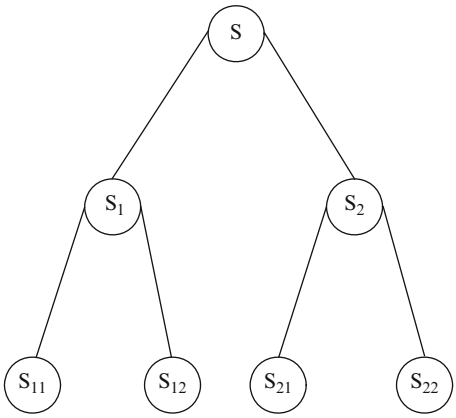

Figure 3.2: Branching tree

Branch-and-bound algorithms are important to solve small-size discrete optimization problems optimally. These known optimum values can be used to assess the performance of heuristics. When the branching tree is truncated, we obtain beam search heuristics (cf. Pinedo [240]). The truncation of the branching tree helps to reduce the computational effort.

3.2.3 Mixed Integer Programming

We introduce MIP formulations as a generalization of linear programming approaches [25, 214, 243]. A linear MIP model is an optimization model with a linear objective function that contains real- and integer-valued decision variables and linear constraints.

Each MIP model can be written in the following form:

$$Z(X) := \min_{(x,y)} \{cx + fy | (x,y) \in X\}, \tag{3.5}$$

where we denote by X the set of feasible solutions. The set of feasible solutions is described by m linear constraints, by nonnegativity constraints for

x, y, and by integer constraints for the decision variables y. Using a matrix representation, we can write the set X in the following form:

$$X := \left\{(x,y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+ | Ax + By \geq b\right\}, \tag{3.6}$$

where

- $Z(X)$ is the optimal objective function value that is obtained by optimizing $cx + fy$ over X.
- x denotes an n dimensional column vector that contains non-negative real-valued entries $x_i, i = 1, \ldots, n$. y is a p-dimensional column vector containing integer-valued components $y_i, 1 = 1, \ldots, p$.
- $c \in \mathbb{R}^n$ and $f \in \mathbb{R}^p$ are row vectors that are given by the coefficients of the objective function (3.5).
- $b \in \mathbb{R}^m$ is the column vector of the right-hand side of the m constraints.
- A and B are $\mathbb{R}^{m \times n}$ and $\mathbb{R}^{m \times p}$ matrices, respectively.

Note that mixed binary optimization models are fairly common. In this case, $y_i \in \{0,1\}, i = 1, \ldots, p$ is valid. When $B = 0$ and $f = 0$ is true, then we have a linear optimization model that can be solved efficiently by the simplex algorithm (see Bertsimas and Tsitsiklis [25]). In the case of MIP models, branch-and-bound techniques (cf. Sect. 3.2.2) will often be applied. The main idea of these techniques for tackling MIPs consists of solving a set of linear optimization problems instead of the MIP model. When $A = 0$ and $c = 0$ are valid, we call the resulting model an integer programming (IP) model. Common software packages to solve MIPs are sophisticated and complex. The software uses a subroutine that solves linear optimization problems in the discussed branch-and-bound algorithms.

We will see in the remainder of this monograph that MIP formulations will usually be used to solve small-size problem instances for scheduling problems optimally (see Nemhauser and Wolsey [214]), whereas linear programming (LP) formulations usually can be applied for planning problems where the required level of detail is not as great.

3.2.4 Stochastic Programming

So far, we have assumed that c, f, A, B, and b in Eqs. (3.5) and (3.6) are deterministic. This assumption is often not realistic in real-world applications. In order to deal with these situations, we introduce stochastic programming as a generalization of linear and MIP (cf. Birge and Louveaux [27]). We assume that the decision model makes some decision in a first stage. Then some random events occur that affect the outcome of the first-stage decision. A recourse decision can be made in a second stage to compensate for any undesirable effects that might have been experienced as a result of the first-stage decision.

In the following, we assume for the sake of simplicity that we consider for now only linear programs, i.e., $B = 0$ and $f = 0$ in Eqs. (3.5) and (3.6),

respectively. We have to make first-stage decisions without full information on some random events. Later, full information is received on the realization of some random vector.

A two-stage stochastic linear program with fixed recourse can be written in the form

$$Z(X) := \min_x \{cx + Q(x) | x \in X\}, \tag{3.7}$$

where the set of feasible solutions is denoted by X. The set of feasible solutions is determined by m linear constraints and by nonnegativity constraints for the decision variables x. Using a matrix representation, we can write the set X in the following form:

$$X := \left\{x \in \mathbb{R}^n_+ | Ax = b\right\}, \tag{3.8}$$

where

- $Z(X)$ is the optimal objective function value that is obtained by optimizing the expression $cx + Q(x)$ over X.
- x denotes an n dimensional column vector that contains non-negative real-valued entries $x_i, i = 1, \ldots, n$.
- $c \in \mathbb{R}^n$ is a row vector that is given by the coefficients of the first term in the objective function (3.7). The function $Q(x)$ is the expected second-stage value function and will be discussed later in more detail.
- $b \in \mathbb{R}^m$ is the column vector of the right-hand side of the m constraints.
- A is an $\mathbb{R}^{m \times n}$ matrix.

The expected second-stage value function $Q(x)$ is the mathematical expectation of the second-stage value function with respect to the random vector ξ, i.e.,

$$Q(x) := E_\xi \tilde{Q}(x, \xi(\omega)), \tag{3.9}$$

where the second-stage objective function, also called the recourse function, is defined as follows:

$$\tilde{Q}(x, \xi(\omega)) := \min_y \{q(\omega) y | y \in Y\}. \tag{3.10}$$

$\tilde{Q}(x, \xi(\omega))$ is the objective function value of a second linear program with the set of feasible solutions

$$Y := \left\{y \in \mathbb{R}^p_+ | Wy = h(\omega) - T(\omega)x\right\}. \tag{3.11}$$

The quantity ω is a realization of a random variable on a probability space Ω. For a given ω, the quantities $q(\omega)$, $h(\omega)$, and $T(\omega)$ are known, where $q(\omega) \in \mathbb{R}^p$ is a row vector that corresponds to the coefficients in the objective function (3.10), $h(\omega) \in \mathbb{R}^s$ is a column vector that is the part of the right-hand side of the s constraints that do not depend on the first-stage decision variable x, and finally $T(\omega) \in \mathbb{R}^{s \times n}$ is the so-called technology matrix. $T(\omega)x \in \mathbb{R}^s$ is the part of the right-hand side of the s constraints of the set of constraints

(3.11) that depends on the first-stage decision x. Finally, $W \in \mathbb{R}^{s \times p}$ is called the recourse matrix. Putting together all the stochastic components of the second-stage data, we obtain the random row vector

$$\xi(\omega) := \left(q(\omega), h(\omega)^T, T_1(\omega)^T, \ldots, T_n(\omega)^T\right) \in \mathbb{R}^{p+s+ns}, \tag{3.12}$$

used in expression (3.9). We denote by $T_i(\omega), i = 1, \ldots, n$ the ith column of the matrix $T(\omega)$. Note that the second-stage decisions y typically are different for different realizations ω.

The representation (3.7)–(3.11) demonstrates the flow of decision-making in a two-stage stochastic program. It starts with taking first-stage decisions x in the presence of uncertainty about future realizations of ξ. Then, in a second stage, the current values of the components of the realization of ξ are known, and the recourse decision y can be made. The first-stage decisions are made in such a way that their future effects are taken into account by considering the recourse function $Q(x)$ that is the expected value of taking decision x. Note also that stochastic MIPs are useful in some situations (see Birge and Louveaux [27] for details on such optimization problems).

The following situation is important in real-world applications. We assume that there are K possible scenarios. We consider the different scenarios as realizations ω. Therefore, we have simply $\omega = 1, \ldots, K$. The probability of scenario ω is denoted by $p(\omega)$. In this case, we can calculate the second-stage value function $Q(x)$ easily. We obtain

$$Q(x) := \sum_{\omega=1}^{K} p(\omega) \left(\sum_{i=1}^{p} q(\omega)_i y_{\omega i} \right) = \sum_{\omega=1}^{K} \sum_{i=1}^{p} p(\omega) q(\omega)_i y_{\omega i}, \tag{3.13}$$

where we denote by y_ω a solution of the second-stage linear program for scenario ω, i.e., we calculate the expected value of the recourse function over all scenarios. As a result, we obtain a large linear program with a specific structure that can be solved efficiently by decomposition approaches (cf. Bertsimas and Tsitsiklis [25] for more details on such methods).

3.2.5 Dynamic Programming

Dynamic programming is another general technique to find the optimal solution of some optimization problems [35, 144]. It can be applied when the optimal solution can be determined recursively from optimal solutions of smaller subproblems. This leads to recursive formulations where the recursions are organized into stages. A dynamic programming approach generally starts with the smallest subproblems. In contrast to pure recursive algorithms, intermediate results are stored in order to avoid a repeated calculation of them.

Dynamic programming formulations are useful to obtain optimal solutions for scheduling problems, especially for single machine scheduling problems. We will see some of these applications in Chap. 5. Because of the large

computational effort and storage requirements of many optimal dynamic programming approaches, dynamic programming is often used within decomposition approaches. Based on some insights into the problem structure, the entire problem is decomposed into subproblems. Dynamic programming can be used to solve some of the subproblems.

We consider an example to illustrate this feature. A single machine batch scheduling problem in semiconductor manufacturing consists of forming batches and sequencing them. The problem can be decomposed into the subproblem of sequencing the jobs and then forming batches based on the sequence of the jobs. Dynamic programming can be used to solve the batching subproblem based on the fixed job sequence. This problem is much easier than simultaneously sequencing and batching the jobs.

Dynamic programming formulations are also often used in stochastic decision processes, especially for Markov decision processes (see Pinedo [240]).

3.2.6 Neighborhood Search Techniques and Genetic Algorithms

Next, we consider different neighborhood search techniques and genetic algorithms (GA) as examples for modern metaheuristics. A metaheuristic is a set of generic, i.e., not problem-specific, principles and schemes used to construct heuristics. Neighborhood search techniques operate on a single solution of problem (P) defined in Sect. 3.2.2 and transfer it into a new solution, while GAs maintain a population of solutions.

We start by introducing the notion of neighborhood structures. The mapping

$$N : S \to 2^S \tag{3.14}$$

is called a neighborhood structure, where we denote by 2^S the set of all subsets of S.

We introduce the notion of moves. A transformation that changes a solution s of (P) into a solution s' is called a move. The definition of a neighborhood is based on this notion of a move. We call a solution s' of problem (P) a neighbor of a solution s when s' can be obtained from s by a single move. The set of all neighbors of a given solution s forms the neighborhood of s. The following notation for the neighborhood will be used:

$$N(s) := \{s' | s' \text{ neighbor of } s\}. \tag{3.15}$$

The solution s is called the center point of the neighborhood $N(s)$. The cardinality of the set of neighbors of s is denoted by $|N(s)|$. Note that this definition of a neighborhood is clearly covered by the notion of a neighborhood structure. We can enumerate the set of neighbors by

$$N(s) := \left\{n_1(s), \ldots, n_{|N(s)|}(s)\right\}. \tag{3.16}$$

A single move

$$\mu(s, n(s)) \in N^{\mu}(s) \tag{3.17}$$

can be assigned to each neighbor $n(s)$ where we denote by $N^{\mu}(s)$ the set of possible moves. It is obvious that a neighborhood can be described by the set of moves as follows:

$$N^{\mu}(s) := \left\{ \mu_1 := \mu(s, n_1(s)), \ldots, \mu_{|N(s)|} := \mu(s, n_{|N(s)|}(s)) \right\}, \tag{3.18}$$

because we consider discrete optimization problems. Often, swapping and insertion moves are used to define neighborhoods. A swapping move exchanges two different solution elements, for example, the position of two jobs in a sequence. An insertion move removes a solution element and places it somewhere else in the solution. For example, we can remove a job from the second position in a sequence and insert it after the job that is in the fourth position.

The simplest way to design a local search algorithm is a steepest ascent-type iterative improvement algorithm for maximization problems. We start from an initial solution $s \in S$. As long as solutions $s' \in N(s)$ with $f(s) < f(s')$ exist, choose the best solution $s' \in N(s)$, set $s := s'$, and repeat this step. This algorithm terminates with some solution s^*. Generally, s^* is only a local maximum with respect to $N(s)$ because we accept only improvements of the incumbent solution.

There are different possibilities to avoid this drawback. One possibility is to restart the iterative improvement algorithm with different initial solutions. A second possibility is to explicitly take deteriorations of the objective function value into account during the iterations. When such non-improvement solutions are accepted as the incumbent solution, then it is possible to visit the same solution several times during the search process. Therefore, our neighborhood search methods may have a cyclic behavior. Consequently, cycling avoidance strategies have to be incorporated.

We discuss simulated annealing (SA) proposed by Kirkpatrick et al. [142] as a neighborhood search strategy for problem (P) that avoids cycling by selecting the current solution s' in a randomized manner. It accepts this random solution s' in iteration i only with probability

$$P(s'|s) = \begin{cases} \exp\left(-\frac{f(s)-f(s')}{t_i}\right), & \text{if } f(s) - f(s') > 0 \\ 1, & \text{otherwise} \end{cases}. \tag{3.19}$$

The numbers t_i are positive with $\lim_{i\to\infty} t_i = 0$. Often, the t_i are of the form $t_{i+1} := qt_i$, where $0 < q < 1$ is valid. When $f(s) - f(s') \leq 0$, then the move will be accepted. Because of the decreasing t_i, the probability to accept a non-improving move will be decreasing. Therefore, in later iterations, the acceptance rate for non-improvement steps will be rather small. This leads to a certain chance to get stuck in a local optimum, but the probability for this is rather small. A cycling of the solutions during the search can also be

avoided by the introduction of a tabu list. A tabu list contains in its simplest form solutions that have already been visited recently during the search. New neighbors are only accepted as an incumbent solution when they are not included in the tabu list. This type of algorithm is called a tabu search method. Usually, a solution is characterized by certain attributes that can be stored in the tabu list instead of the full representation of the solution. Typically, it is not possible to store all the visited solutions due to memory restrictions and performance issues. Therefore, we consider a list that stores only the last l visited solutions. When l is chosen big enough, then there is only a small probability of cycling. There are various tabu search variants. For a more detailed description of tabu search, the reader is referred to Glover and Laguna [102].

Variable neighborhood search (VNS) is a local-search-based metaheuristic (cf. Hansen and Mladenović [115, 188]). The main idea is to enrich a simple local-search method in order to enable it to escape local optima. This is done by restarting the local search from a randomly chosen neighbor of the incumbent solution. This restarting step is called shaking. It is performed using different neighborhood structures of increasing size. There are a couple of different VNS variants. In the following, we will briefly discuss basic VNS. For the remaining variants, we refer the reader to Hansen and Mladenović [115].

Therefore, we consider a set of neighborhood structures $N_k, k = 1, \ldots, k_{\max}$ and an initial solution $s \in S$. We choose randomly a solution s' from $N_k(s)$. Initially, we use $k = 1$. Based on s' as an initial solution, we receive a local optimum s'' by local search. When

$$f(s'') > f(s) \tag{3.20}$$

is true, then we move there, i.e., we set $s := s''$ and repeat the entire process starting from N_1. When there is no improvement obtained by s'' compared to the incumbent solution s, then we consider the next neighborhood $N_{k+1}(s)$ and repeat the shaking and the local-search step until a certain stopping criterion is meet.

VNS has the advantage that, compared to many other metaheuristics, the number of parameters to be selected is small. We only have to find a set of neighborhood structures and apply them in an appropriate sequence.

A GA maintains a set of feasible solutions called the population. GAs are motivated by principles of evolution and survival of the fittest [103, 183]. Variation operators, i.e crossover and mutation, and selection operators are used to modify the elements of the population. GAs are often appropriate for optimization problems that have the property that a combination of good partial solutions frequently leads to a good solution with respect to the objective function. A solution s is typically encoded into a sequence of symbols called a chromosome. The encoding scheme is called a representation. Each chromosome has to be evaluated by a fitness function f. The fitness function

is often based on the objective function for the optimization problem to be solved. It is well known that the performance of a GA depends strongly on the encoding scheme used (see Rothlauf [270]).

We start from an initial population of chromosomes. Each chromosome is evaluated using a problem-specific fitness function. Next, parents are selected based on their fitness. Offspring, called child chromosomes, are created from a set of parent chromosomes applying crossover operators to the parents. Crossover operators usually combine certain parts of the sequences that form the two parent chromosomes into a new child. Mutation operators are used to disorder the child chromosome with a certain probability. The child chromosomes are added to the original population. Based on the fitness that is associated with each chromosome, some chromosomes are removed based on selection operators to make sure that the size of the population is the same over all generations. The entire cycle of evaluation of the parents, offspring generation, and selection of the new population is called a generation.

We will see in Chap. 5 that GAs are useful to solve scheduling problems for single and parallel machines. But they can also be applied to solve planning problems for semiconductor supply networks or to find an appropriate mix of dispatching rules to control wafer fabs as shown in Chaps. 7 and 4, respectively.

Except for the description of stochastic programming methods in Sect. 3.2.4, so far only decision methods for deterministic problems are studied. Next, we introduce two additional methods that are related to decision-making for stochastic problems, i.e., queueing theory and discrete-event simulation.

3.2.7 Queueing Theory

The first method related to decision-making for stochastic problems is queueing theory. A simple queueing system can be characterized as an input–output system consisting of a waiting line, also called a queue, and a single server. However, there are also queueing systems that are formed by service centers and interconnecting queues. A service center consists of a number of servers working in parallel. Customers from a calling population arrive from time to time and join the queue. In some cases, the number of customers that can be waiting in the queue or in the system is limited by the capacity of the system. The customers in queue are eventually served, and after service, they leave the system. A simple queueing system is shown in Fig. 3.3.

In manufacturing, the customers of the queueing system generally correspond to jobs, and the servers are the machines. Often, the number of job arrivals to and departures from the system are of interest. Based on this information, the time that a job spends on average within the manufacturing system can be estimated. This quantity is called the cycle time (CT). Furthermore, the number of jobs within the manufacturing system that are either undergoing processing or waiting in a queue for processing can also

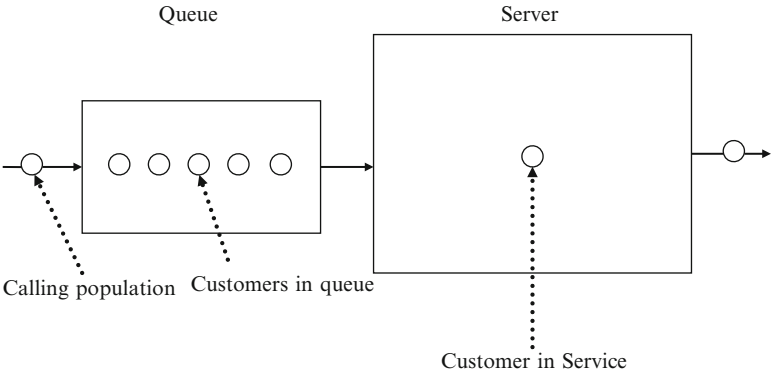

Figure 3.3: Components of a simple queueing system

be determined based on the arrival and departure information. This number of jobs is called work-in-process (WIP). The arrival process of the calling population is usually characterized in terms of inter-arrival times of successive customers. The queueing discipline determines which customer will be selected for service when a server becomes available. Common queueing disciplines include the first-in-first-out (FIFO) or last-in-first-out (LIFO) rules. Service time can be a constant or can have a random duration. Queueing theory is a mathematical approach to analyze queueing systems.

WIP and CT usually vary over time. Because it is difficult to treat this time dependency in analytic models, we are interested in time-averaged values. A queueing system is called steady state when the probability that the system is in a given state is not time-dependent. The steady-state values for WIP and CT can be considered as time-averaged values as the time becomes very large. Because we consider steady-state systems for a long time horizon, the values of WIP and CT do not depend on the initial conditions of the system. The following equation

$$\mathrm{WIP} = \lambda \mathrm{CT} \tag{3.21}$$

holds for a manufacturing system that satisfies steady-state conditions (see Little [165]). The quantity λ is the long-run input rate of the jobs to the server. Equation (3.21) is called Little's law. We consider a single server system with exponentially distributed inter-arrival times with mean rate λ and exponentially distributed service times with mean rate μ. It can be shown that the steady-state probability that n jobs are in the system, denoted by $P(N = n)$, is given by

$$P(N = n) = \left(1 - \frac{\lambda}{\mu}\right)\left(\frac{\lambda}{\mu}\right)^n \tag{3.22}$$

for $n = 0, 1, 2, \ldots$, where we denote by N the long-run number of jobs in this system (cf. Curry and Feldmann [55]). We use $p_n := P(N = n)$ for abbreviation. The expected WIP of the system is the expected value of a random variable that is distributed according to the discrete probability distribution (3.22). We obtain

$$\text{WIP} = E(N) = \sum_{n=1}^{\infty} n p_n = \frac{\lambda}{\mu - \lambda}. \tag{3.23}$$

It is clear that $\lambda < \mu$ has to be fulfilled to ensure a steady state for this system. Based on Little's law, we can compute the long-run CT as

$$\text{CT} = \frac{1}{\lambda - \mu} \tag{3.24}$$

because exponentially distributed inter-arrival times with mean rate λ are induced by a Poisson arrival process with mean rate λ.

When we relax the assumption of exponentially distributed inter-arrival and service times, the following approximation formula is still valid for the expected value of the time that a job spends in a queue. The corresponding random variable is denoted by T_q. The service time is modeled by the random variable T_s. We obtain for $CT_q := E(T_q)$

$$CT_q \approx \left(\frac{C_a^2 + C_s^2}{2} \right) \left(\frac{u}{1 - u} \right) E(T_s), \tag{3.25}$$

where we denote by C_a^2 and C_s^2 the squared coefficient of variation of the inter-arrival time and the squared coefficient of variation of service times, and we set $u = \lambda / \mu$ for abbreviation. The squared coefficient of variation for a random variable T is defined as $C^2(T) := \text{Var}(T)/E(T)^2$. The approximation (3.25) is called the Kingman diffusion approximation (see Hopp and Spearman [119]).

There is a notation introduced by Kendall [138] for queues. It is a five-field notation, where two consecutive entries are separated by a slash. The first entry specifies the inter-arrival time distribution, while the second entry provides information regarding the service time distribution. The third entry specifies the number of parallel servers. The maximum number of jobs allowed in the queueing system at one time is given by the fourth entry. Finally, the optional fifth entry describes the queueing discipline used. For example, M/M/1/∞/FIFO refers to a system with exponentially distributed inter-arrival and service times, a single server, an infinite capacity queue, and FIFO queueing discipline. When the fourth parameter is infinite, it is often omitted. General inter-arrival or service times are denoted by the symbol G, while the symbol M (Markovian) is used for exponentially distributed inter-arrival or service times.

In addition to the single-stage queueing model described above, it is possible to analyze the performance of queueing networks. For a more detailed introduction into queueing theory (including queueing networks) with appli-

cations to manufacturing, we refer to text books like [55, 119]. A survey that also includes a discussion of some limitations of classical queueing theory in semiconductor manufacturing can be found in Shanthikumar et al. [281].

3.2.8 Discrete-Event Simulation Techniques

Discrete-event simulation is the second method that is able to deal with stochastic decision problems. Queueing theory relies heavily on specific distributional assumptions of the underlying stochastic processes. In many real-world situations in semiconductor manufacturing, these assumptions are not fulfilled [281]. Therefore, we introduce discrete-event simulation as another important tool for decision-making in manufacturing that takes the stochastic and dynamics of the BS and the BP implicitly into account, but one that is based on less restrictive assumptions than queueing theory.

Simulation models of wafer fabs are of specific interest in the remainder of this monograph. Simulation is used to describe a process in a time-dependent manner (cf. Law [150] and Banks et al. [21]). Depending on the time progress or method, we differentiate between discrete-event and continuous simulation. In the first case, the timing of future events is determined, and the simulation jumps to the next future event. In some cases, we consider an equidistant time progression. In continuous simulation, infinitesimal small time steps are made. Continuous simulation is basically the numerical treatment of differential equations where difference equations are used to describe the changes in system variables over discrete time steps.

In this monograph, we mainly refer to discrete-event simulation. This type of simulation is based on the metaphor that entities flow through the simulation model and occupy scarce capacity that is offered by servers. There is some competition of the entities for the servers, i.e., they have to wait before they are served. In simulation models of wafer fabs, the servers are represented by resources and the moving entities by jobs.

Next, we briefly describe the main ingredients of a simulation model for wafer fabs. Depending on the goal of a simulation study, models of different complexity can be used ranging from very detailed and close to the real wafer fab to coarse and abstract. For industrial studies, very detailed models are generally used. In academic studies, both detailed and simple models are applied. A simulation model of a wafer fab can be considered as a model system S_M. Typical model components for complete wafer fab models are as follows:

- Equipment, i.e., the set of machines
- Operators and secondary resources
- Components related to material handling
- Process flows

We start by describing the JS-related (see Sect. 2.2.1) modeling issues. Apart from cluster tools, there is generally no need to model the internal behavior

of a machine, i.e., to have a detailed mechanical model or a model of the controllers that work inside a machine. Such models are used by equipment manufacturers but do not often lead to additional insight while analyzing and controlling a wafer fab. Thus, the model of the JS for wafer fabs can be kept rather simple. The typical machine-related parameters are as follows:

- Name of the machine group
- Number of machines in the machine group
- Name of the machine
- Batch-size information
- Batch formation criterion
- Setup time
- Machine qualification and dedication requirements
- Preventive maintenance cycles
- Breakdown-repair cycles

A batch machine is able to process more than one job at the same time. This number has to be specified as the number of jobs or wafers that can be batched together, and it describes the capacity of the batch machine. The batch formation criterion provides information on which jobs can be batched together. Often, we have to deal with incompatible job families, i.e., only jobs from one family can be batched together. Incompatible families are formed due to the different chemical nature (or processing time) of the different process steps on the batch machines. Sometimes, only jobs that refer to the same process step can be used to form a batch.

The setup time is the time that is required to set up a machine before processing. For a given setup state, it can be constant or depend upon the current setup state. In the latter case, we have sequence-dependent setups, and all setup times of a particular machine are listed in a setup time matrix.

Semiconductor manufacturing equipment is complex, and considerable effort is spent to keep the equipment running properly. Preventive maintenance activities are included in most wafer fab level simulation models. Often, weekly, monthly, quarterly, and yearly schedules are included. Despite efforts to prevent failures of the equipment, failures are still quite significant for many types of equipment.

One or more machine breakdown-repair cycles may exist for a machine or the entire machine group. Each cycle definition consists of a time-to-failure (TTF) and a time-to-repair (TTR) probability distribution function. In some cases, the time to failure is not counted continuously but only when the machine is busy. In other cases, failures depend upon the number of jobs or wafers processed and not upon the time-in-process state or the simulation time in general. In these latter cases, the TTF has to be given in multiples of the processing time. When there is more than one cycle for a machine, it has to be specified how to deal with parallel failures. Often, the beginning of a failure that happens when the machine is already down is postponed to the end of the current failure. Another issue concerns parallel failures on different

machines of a machine group. In particular, if the failures are used to model maintenance actions, they will generally not be performed in parallel to avoid unnecessary waiting times at the machine group buffer. The failure model has to reflect this accordingly. Problems related to the modeling of machine breakdowns in semiconductor manufacturing are described by Schömig and Rose [277].

Next, we consider the representation of operators as human decision makers in simulation models of wafer fabs. In general, due to the lack of accurate models, only simple models are applied for the humans participating in the wafer production process. In particular, interaction of operators is not modeled, i.e., the social or communication component of the human workforce is ignored. In addition to the parameters given below, operator control has to be implemented, i.e., what happens if more operators are required than are available. Usually, dispatching rules are applied in this case. The following information is required to model operators:

- Name of the operator group
- Number of operators in the group
- Skills of the operator or the operator group
- Staffing information
- Operator break cycles

The modeling of skills is important because most operators in a wafer fab are certified to run one or two machines because of the complexity of the equipment and the training costs. There are, however, a few operators who are cross-trained for multiple types of equipment. Similar to machines, operators have break cycles. There are regular breaks like lunches and random breaks like going to the bathroom. In addition, it is sometimes modeled whether operators are allowed to have breaks together or whether the breaks have to be staggered. Staffing information is required when the number of operators changes from shift to shift. It is also possible to model holidays. We refer to Mosley et al. [210] for an example of modeling operators in simulation models of wafer fabs. But often, operators are not included in simulation models of wafer fabs. This coincides with the changing role of operators in most highly automated wafer fabs (cf. the comments in Sects. 2.2.2 and 2.2.3).

Besides operators, sometimes, the detailed modeling of other auxiliary resources is necessary, especially, when these resources are scarce. Reticles in the photolithography area are an important example. The modeling of reticles within a simulation model is described, for example, in [41, 201, 228].

The main components of the MS that have to be part of the simulation model are as follows:

- Carriers
- Stockers and their assignment of bays or certain machines
- Transportation system

Note that often specialized simulation packages are used to model the JS and the MS of wafer fabs. For example, the commercial simulation packages

AutoSched AP and AutoMod are used to simulate wafer fabs. AutoSched AP simulates the JS of a wafer fab, whereas AutoMod is responsible for the simulation of the corresponding MS. The two simulation engines can be coupled by the model communication software of Brooks automation. This approach is used, for example, by Schulz et al. [278] and Pillai et al. [239] for 300-mm full-factory simulations. More simulation studies related to the MS of wafer fabs can be found, for example, in [130, 282, 316].

We continue by describing the representation of the BP within a simulation model. A process flow is required for each product manufactured in a wafer fab (see Sect. 2.2). It lists all process steps required to finish the product. In general, process flows are deterministic, but, depending on the product, it may contain alternative subprocess flows or rework loops. For each process flow, the following parameters are supplied for each process step:

- Name of the process step
- Name of the machine or machine group where the process step has to take place
- Operator requirements, i.e., the required qualification of the operators, the number of operators, and whether they have to be present during the whole period of loading, processing, or unloading or only during portions of these operations
- Auxiliary resources required
- Processing time
- Load and unload times
- Required setup state of the machine
- Amount of scrapped material
- Rework loops
- Alternative flows

The processing time is given per process step and consists of several components. Often, a machine can be loaded with more wafers than can be processed; processing of these wafers takes place in several portions. Therefore, the processing time often depends on the number of wafers or on the number of jobs. As a consequence, the processing times of jobs of the same product at a certain process step are not necessarily the same, for example, due to scrapped wafers. Note that some machines, like steppers in the photolithography work area or pipeline tools, require more sophisticated approaches to determine the processing time (see Mönch et al. [201] for the stepper case where the processing time depends on the product, on the mask level, and also on the number of ICs on a single wafer in addition to the number of wafers). The processing time for batches on diffusion furnaces in wafer fabs depends on the family of the job. The processing time associated with a batch on burn-in ovens at the back-end stage is determined by the longest processing time of one of the jobs that form the batch.

Note that in the case of cluster tools, the situation is even more complicated because the processing time depends on the sequence of processed jobs (see

Sect. 2.2 for more details on cluster tools). In principle, we have to model the internal behavior of a cluster tool to determine these processing times. Therefore, usually only simple processing time models are used in simulation models of full-size wafer fabs (see Shikalgar et al. [282]).

The load time t_{load} and unload time t_{unload} are defined as the time that is required to move a job to or from a buffer or other material handling system device before or after processing. Of course, t_{load} and t_{unload} have to add to the processing time.

At certain process steps, sometimes jobs, wafers, or dies are processed in a way that they become useless for further processing. In this case, a percentage of units scrapped is provided to reflect this behavior. This percentage is called the amount of scrapped material.

Rework is closely related to scrapped material. A percentage is given that a rework loop has to be entered either for the whole job or for several wafers. If a rework loop occurs, the whole job is processed, or a child job with rework wafers is built. The parent job may either wait until the rework loop is successfully finished and rejoin with the child job, or it may proceed immediately. At the end of the rework loop, a decision is made whether the job is allowed to continue, whether the job has to repeat the rework loop, or whether it is scrapped.

In some situations, several subflows are possible to produce different ICs that come from the same technology, i.e., a process flow consists of a sequence of subflows. For some positions in this sequence, several subflows are possible. A certain percentage is given for each of the possible subflows in this situation. At the end of the subflows, they merge again into a single process flow. It is possible that the subflow consists of a single process step. In this case, the term alternative process step is used. The concept of alternative process steps is important to model heterogeneous machines correctly.

We consider a simulation model of small complexity suggested by researchers from Intel Corporation and described by Spier and Kempf [291] as an example of a simulation model that contains typical features of a wafer fab with respect to BS and BP. It contains only three machine groups and two process flows with six process steps. The process flow is organized in two layers. Among the machine groups, there is a batch-processing machine group and a machine group with sequence-dependent setup times. The model mimics some important features of wafer fabs. We show the process flow in Fig. 3.4. We call this simulation model the MiniFab model.

The processing times of the different process steps in minutes and the maximum batch sizes in jobs are shown in Table 3.1.

Simulation studies with simulation models of full-size wafer fabs tend to be very time-consuming. This is partially caused by modeling a lot of details that are not always necessary. That is why some researchers and simulation practitioners started to consider reduced simulation models [121, 231, 263, 265, 269]. The reduction is often achieved by modeling only process steps that are related to bottleneck machines. The remaining process steps are

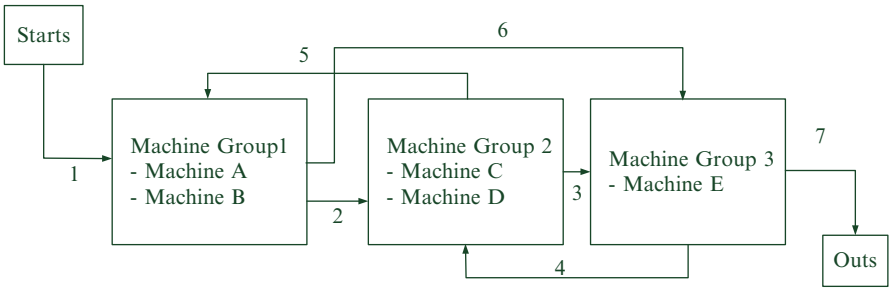

Figure 3.4: Process flow of the MiniFab model

Table 3.1: Process flows and maximum batch sizes of the MiniFab model

Process step	Machine group	Processing time	Maximum batch size
1	1	225	3
2	2	30	1
3	3	55	1
4	2	50	1
5	1	255	3
6	3	10	1

replaced by fixed time delays instead of modeling the processing of single process steps on non-bottleneck machines. However, it is far away from being trivial to find appropriate delays. Often, a large amount of simulation runs with full-size simulation models are also required to assess the quality of a reduced simulation model.

Each simulation model requires checking the logic of the model. This process is called verification. At the same time, it is also necessary to compare the results of the simulation model to reality. This step is called validation. It is usually an iterative process [21, 150] that tends to be time-consuming in the case of wafer fabs.

Therefore, reference simulation models are available for the research community to reduce the effort needed to create simulation models and to validate and verify them. Such simulation models are available in the testbed hosted at the modeling and analysis of semiconductor manufacturing (MASM) laboratory as a result of the measurement and improvement of manufacturing capacity (MIMAC) project (cf. Fowler and Robinson [83] for a description of these models that are part of the MASM Lab testbed). The MIMAC 1 model is also briefly described in Sect. 4.4.

It appears that, while simulation models are well established for single wafer fabs, it is not true for semiconductor supply networks. Only some preliminary work has been published for simulation modeling of an entire supply network (cf. Duarte et al. [74] for some recent work).

Usually, we use simulation when the analytic methods described in Sects. 3.2.2–3.2.7 do not adequately represent the system being studied. For example, in some situations, the objective function is nonlinear, some of the restrictions cannot be modeled by a set of linear constraints, or the optimization problem is simply too large to apply branch-and-bound techniques. While discrete-event simulation offers some advantage with respect to modeling capabilities, it takes considerable effort to build a simulation model, mainly because of data gathering and modeling difficulties. The main data needed to create a simulation model was described in this subsection. Experience in statistics is necessary to perform a meaningful interpretation of the simulation output.

We start by describing simulation techniques for the support of production planning and control decisions. Discrete-event simulation can be used to represent the BS and the BP. In this case, they are identified by the discrete-event simulation model. We will see that this approach is useful to assess the performance of planning and control algorithms in Sect. 3.3. In the simplest form of this approach, simulation can be used to decide the rule for which job should be processed next on an available machine. Therefore, discrete-event simulation is a tool to assess the performance of dispatching rules.

When simulation is used as a decision-making tool and not for identifying the BS and the BP, then either performance measure values as a result of planning or control instructions for BS and BP are estimated by simulation, or certain parameters that describe the behavior of the BS of interest are determined using simulation. Therefore, we differentiate between simulation-based optimization, iterative simulation, and finally the determination of immediate control instructions, i.e., which job has to be processed next on a given machine by simulation.

Simulation-based optimization starts from the idea that it is in some situations difficult to describe and evaluate an objective function (cf. Fu et al. [91] and Fu [92] for a more detailed description of the main concepts). In this situation, the objective function can be implicitly represented by a corresponding simulation model. Simulation-based optimization therefore also requires an appropriate simulation model. Optimization is typically performed using metaheuristics such as SA, GA, tabu search, or VNS. These methods tend to be computationally expensive. Therefore, the level of detail for the simulation model is important in simulation-based optimization applications. Stochastic effects typically are not neglected. They have to be taken into account both for the metaheuristic that is used for optimization and the simulation model itself. The overall scheme for simulation-based optimization applications can be described as follows.

Algorithm Simulation-Based Optimization

1. Start from a given initial solution.
2. Use the simulation model in order to estimate the objective function value by multiple runs in a stochastic setting.

3. Modify the incumbent solution by local changes using a certain metaheuristic.
4. Repeat the algorithm from step 2 onwards until a given stopping criterion is met.

Simulation-based optimization can be used either on the entire BS and BP level or for subsystems and subprocesses of it, respectively.

In contrast to simulation-based optimization, iterative simulation is used to find appropriate values for certain parameters of a decision model for planning or control problems. In production planning models, often the CT is such a parameter. Then, the production planning or control problem is solved using the current values for the parameters of interest. The resulting solution provides input for the discrete-event simulation model. For example, the quantity of jobs to be released is determined by a production planning model, then the parameter of interest is estimated using simulation. The current parameter value is updated using the simulation results. The updated value is used again as an input of the production planning or control model. The overall scheme can be summarized as follows.

Algorithm Iterative Simulation

1. Use an appropriate initial value $\rho_{\text{curr}} := \rho_{\text{init}}$ for the unknown parameter ρ. Solve the production planning or control problem using ρ_{init}.
2. Perform a simulation using the solution of the production planning and control problem as input for the simulation.
3. Determine the real parameter value ρ_{sim} as a result of the simulation.
4. Determine a new current parameter value ρ_{curr} based on the actual values for ρ_{curr} and on ρ_{sim} using, for example, exponential smoothing.
5. Solve the production planning or control problem again using the updated value for ρ_{curr}.
6. Repeat the scheme starting in step 2 until a given termination criterion is met.

Often, only a few iterations are necessary to achieve the desired maximum difference between the parameter values in two consecutive iterations. Furthermore, the initial value of the parameter of interest is often not a crucial factor. Iterative simulation in a supply chain context is discussed, for example, in Almeder et al. [7]. In this monograph, we will also see applications for setting parameters of dispatching rules due to Vepsalainen and Morton [312] in Chap. 4 and for production planning applications in Chap. 7.

Finally, discrete-event simulation can be used to determine immediate control instructions. Therefore, a simulation model that represents the current state of the BS and the BP, including a certain set of dispatching rules to decide which job has to be processed next, is necessary. An assignment of jobs to machines and sequences of jobs on single machines are determined using the dispatching rules. When different dispatching rules are used, multiple assignments and sequences are the result. Based on the preferences of a human decision maker, one specific assignment and the corresponding sequences are

chosen and then are used to make the decisions in the production control system. This concept is also known as simulation-based scheduling and will be discussed in more detail in Sect. 5.2.

3.2.9 Response Surface Methodology

Running simulation experiments is time-consuming, especially when simulation is used to find an appropriate setting of several parameter values for the BS and the BP with respect to a certain performance measure. An exhaustive search by simulation is not possible from a computational burden point of view. In this case, meta-models are an appropriate way to treat this kind of optimization problem. To model this class of problems, we assume that the performance measure value, called the response, can be expressed in the following form:

$$y := f(\xi_1, \ldots, \xi_k) + \varepsilon, \tag{3.26}$$

where the form of the true response function f is unknown and ε represents the variability. The controllable input variables are denoted by $\xi_i, i = 1, \ldots, k$. They represent the parameter values of the BS and BP that have to be varied. The term ε is treated as a statistical error. We assume that it has a normal distribution with mean zero and variance σ^2, i.e., $\varepsilon \sim N(0, \sigma^2)$. The variables ξ_i are called natural variables. However, it is often more convenient to work with dimensionless variables that are normalized, i.e., from $[-1, 1]$. These variables are called coded variables and can be achieved by some transformation of the natural variables. The true response function can be represented in the form $\eta := f(x_1, \ldots, x_k)$. Because f is unknown and often complicated, we approximate it by a low-degree, usually first- or second-order polynomial. Second-order meta-models are of the general form

$$y := \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{j} \sum_{i \neq j} \beta_{ij} x_i x_j + \varepsilon, \tag{3.27}$$

where the unknown coefficients β_0 and $\beta_i, \beta_{ij}, i = 1, \ldots, k, j = 1, \ldots, k$ are called the parameters of the model that have to be fitted from results of simulation runs. Note that we obtain a first-order model out of the second-order model (3.27) when we select $\beta_{ii} = 0$ and $\beta_{ij} = 0$. While least square analysis is often used to estimate the significant parameters for the meta-model, analysis of variance techniques is used to determine which parameters are statistically significant.

We assume that the discrete-event simulation model is a reasonable representation of the BS and the BP. If the meta-model is an adequate approximation of the corresponding simulation model, then the optimization of this model will be approximately equivalent to the optimization of the BS and BP. Therefore, after we have determined the second-order meta-model, we can use it instead of the simulation model to optimize the response by

determining appropriate values for the coded variables. By using standard optimization techniques, i.e., steepest descent for the first-order model and conjugate gradient methods for the second-order meta-model, the corresponding optimization problems can be solved using standard optimization software.

For a more detailed introduction into the response surface methodology, we refer to Myers et al. [212]. Some applications of meta-modeling techniques in semiconductor manufacturing are shown by McAllister et al. [177]. Chapter 4 contains applications related to the design of blended dispatching rules in semiconductor manufacturing.

3.2.10 Learning Approaches

Following Russell and Norvig [273], a basic learning system contains a performance element and a learning element. The performance element decides what actions are to be taken, whereas the learning element modifies the performance element to allow it to make better decisions. The learning element takes feedback from the environment into account during learning. A reasoning mechanism is responsible for the learning.

In a certain sense, the regression analysis described in Sect. 3.2.9 might be considered to be learning a continuous function from examples of its input and output. The resulting meta-model is the performance element, while least square analysis is used as the reasoning mechanism.

Next, we continue with a discussion of neural networks as a widely used learning approach. A neural network consists of a set of nodes, called neurons, and the neurons are connected in different ways. Each neuron is used to implement an activation function T:

$$y_i := T\left(\sum_{j=1}^{n} w_{ji} u_j\right). \tag{3.28}$$

This function determines output values as a result on given input values. Input values are denoted by $u_j, j = 1,\ldots,n$. The output values are denoted by $y_i, i = 1,\ldots,m$. The quantities $w_{ji}, j = 1,\ldots,n, i = 1,\ldots,m$ are the corresponding weights. The neurons are organized in layers. Besides input and output layers, a neural network also has internal layers. Input values are transformed into output values using the weighted input values.

Applying neural networks to make a certain decision requires determining appropriate input values to characterize the problem and to represent the solution of the problem by the output values. Training data, i.e., a certain set of problem instances with known high-quality solutions, also called examples, is used to discover how the output values depend on the input values. It is clear that the neural network as a meta-model is the performance element, while the learning approach for the weights is the learning element.

The main advantage of neural networks is their ability to learn by adapting the weights. Furthermore, neural networks are able to deal with noisy

data. Nonlinear functions can be represented by neural networks. The basic limitation of the neural network approach is that a new network has to be constructed when the underlying situation has changed. This often requires a large training effort. At the same time, training data is often not available in case of a dynamic and stochastic BS and BP. In Chap. 5, we will mention applications of neural networks in scheduling.

Inductive decision trees are another learning approach. A decision tree takes, as input, a situation described by a set of attributes and returns the predicted output value for the input. A decision tree makes decisions by performing a sequence of tests. The tree consists of internal nodes and leaf nodes. The leaf nodes represent the output value that is the result when this leaf is reached. The internal nodes test the value of one of the properties. The learning is performed by using a set of examples, i.e., a mapping between situations and output values to determine the decision tree (see Russel and Norvig [273]).

3.2.11 Summary of Decision Methods and Descriptive Models

Finally, we would like to combine the different type of models presented in Sect. 3.1.2 and the various methods from Sect. 3.2. The resulting scheme is shown in Fig. 3.5. We can clearly see that while discrete-event simulation is both descriptive and prescriptive at the same time, it is always dynamic. Only dynamic programming and discrete-event simulation are deterministic and stochastic. Queueing theory is descriptive and contains elements of both static and dynamic models, but it is stochastic. The response surface methodology is descriptive and can be also prescriptive.

It is not reasonable to assign the learning approaches discussed in Sect. 3.2.10 to any of the model types from Sect. 3.1.2 because learning models are often used to support the parameter selection process for decision methods. Therefore, we avoid a specific assignment.

It is interesting to remark that, as already stated in Sect. 3.1.2, dynamic and stochastic models are difficult to analyze. We can see from Fig. 3.5 that only discrete-event simulation is able to deal with these kinds of models. Based on this insight, we can conclude that discrete-event simulation is of particular importance.

3.3 Performance Assessment

In this section, we present a performance assessment methodology. In addition, we discuss an architecture that allows for simulation-based performance assessment of production planning and control approaches.

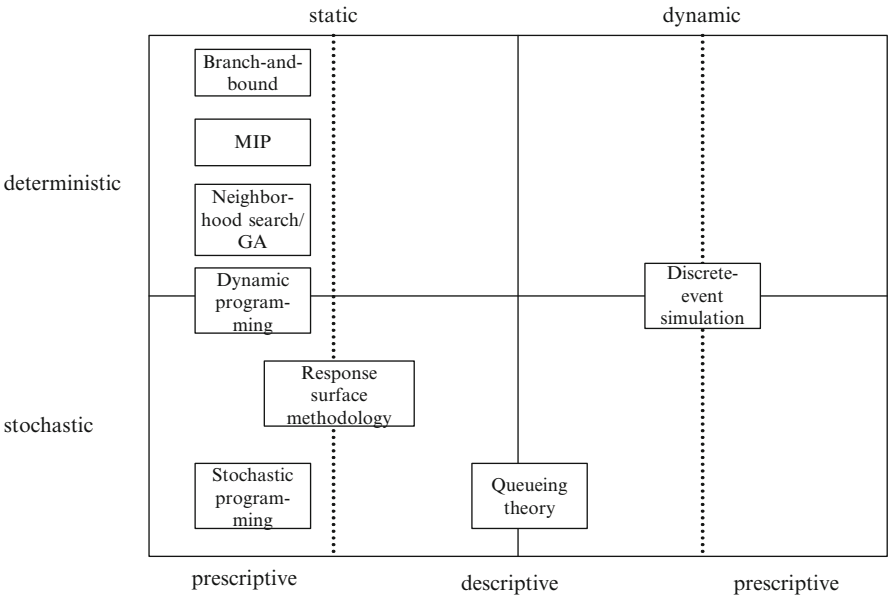

Figure 3.5: Scheme for decision methods and descriptive models

3.3.1 Performance Assessment Methodology

When a new decision-making algorithm is proposed, in the beginning, its performance is unknown. Therefore, it is necessary to assess its likely performance before it can be applied in a wafer fab. There are, in principle, two possibilities. The first one takes a snapshot of the state of the BS and the BP and then determines the values of certain performance measures. This type of performance assessment is described in some detail in Rardin and Uzsoy [257]. The second one repeats running the decision-making algorithm in a time-based or event-based manner taking the current state of the BS and the BP into account. We refer to the first approach as a single instance-based performance assessment, whereas the second one is called a rolling horizon-type performance assessment scheme. Note that the single instance-based approach is rather static, while the latter one is much more dynamic and takes appropriate feedback of the BP and BS into account.

In a next step, we describe a performance assessment methodology that can be applied in both situations. In this methodology, a fixed production planning or control approach with unknown performance is considered. In order to assess its performance, the following scheme is suggested:

- Determination and specification of the appropriate performance measures
- Determination of production planning and control approaches used for comparison with the new approach
- Description of different problem instances and simulation scenarios in the first and second cases, respectively, and specification of designed experiments
- Specification of the performance assessment strategy
- Description of the hardware and software environment for the performance assessment
- Running the new planning and control approach for each problem instance or, in a rolling horizon manner, by discrete-event simulation for each simulation scenario in the first and second cases, respectively, and discussion and interpretation of the results

We start by describing the difference between direct and indirect performance measures. This distinction corresponds to the planning and control setting presented in Fig. 2.2. Direct performance measures consider quantities that have direct influence on the performance of the manufacturing system related to the BS and the BP. Throughput (TP), CT (cf. Sect. 3.2.7), and total weighted tardiness (TWT) are examples of direct performance measures. TP is defined as the number of completed jobs leaving a system within a certain period of time. The TP rate is consequently the number of completed jobs per unit of time. In the MS, the number of moves of the vehicles is a TP-related measure. Closely related to TP is the makespan $C_{\max}$ of a set of n jobs. It is defined by the expression

$$C_{\max} = \max\left\{C_j | j = 1, \ldots, n\right\}, \tag{3.29}$$

where we denote by C_j the completion time of job j. The measure CT for job j is defined as

$$\mathrm{CT}_j := C_j - r_j, \tag{3.30}$$

where r_j denotes the ready (release) time of j. The average CT (ACT) of n jobs is given by

$$\mathrm{ACT} := \frac{1}{n} \sum_{j=1}^{n} (C_j - r_j). \tag{3.31}$$

ACT is an important measure in semiconductor manufacturing because a small ACT value may lead to large yield because the probability of contamination on the shop floor is smaller in case of a small CT (see Atherton and Atherton [14]). The carrier delivery time (CDT) is the counterpart of the BS-related CT within the MS. In some situations, the variance of CT is of interest. It is defined by

$$\mathrm{Var}(\mathrm{CT}) := E((\mathrm{CT} - E(\mathrm{CT}))^2) \approx \frac{1}{n-1} \sum_{j=1}^{n} (\mathrm{CT}_j - \mathrm{ACT})^2. \tag{3.32}$$

Closely related to CT is the measure total flow time or total completion time (TC). It is defined as

$$\mathrm{TC} := \sum_{j=1}^{n} C_j = n\mathrm{ACT} + \sum_{j=1}^{n} r_j, \tag{3.33}$$

i.e., minimizing ACT is equivalent to minimizing TC. Its weighted counterpart is the total weighted completion time:

$$\mathrm{TWC} := \sum_{j=1}^{n} w_j C_j, \tag{3.34}$$

where we denote by w_j the weight of job j. On-time delivery performance-related measures are also important because of possible customer satisfaction and hence advantage in the fierce competitive market. The tardiness T_j of a job j is given by

$$T_j := \max(C_j - d_j, 0), \tag{3.35}$$

whereas the performance measure TWT is defined by

$$\mathrm{TWT} := \sum_{j=1}^{n} w_j T_j, \tag{3.36}$$

where we denote by d_j the due date of job j. The average weighted tardiness (AWT) is defined as

$$\mathrm{AWT} := \frac{1}{n}\mathrm{TWT}. \tag{3.37}$$

When we have $w_j \equiv 1$, then we are talking about total tardiness (TT). We use the notation AT for the average tardiness.

Somewhat related to TT is the maximum lateness. This performance measure is defined as follows:

$$L_{\max} := \max\{L_j | j = 1, \ldots, n\}, \tag{3.38}$$

where we denote by $L_j := C_j - d_j$ the lateness of j. Similar to Var(CT), the variance of the lateness $\mathrm{Var}(L)$ is defined by

$$\mathrm{Var}(L) := E((L - E(L))^2) \approx \frac{1}{n-1}\sum_{j=1}^{n}(L_j - \mathrm{AL})^2, \tag{3.39}$$

where the average lateness (AL) is given by $\mathrm{AL} := 1/n\sum_{j=1}^{n} L_j$.

Finally, in some situations, the number of tardy jobs (NTJ) is of interest. This performance measure is defined as follows:

$$\mathrm{NTJ} := \sum_{j=1}^{n} U_j, \tag{3.40}$$

where we have

$$U_j := \begin{cases} 1, \text{ if } 0 < C_j - d_j \text{ for job } j \\ 0, \text{ otherwise.} \end{cases} \tag{3.41}$$

The weighted counterpart of NTJ, the weighted number of tardy jobs (WNTJ)

$$\text{WNTJ} := \sum_{j=1}^{n} w_j U_j, \tag{3.42}$$

is also considered. TT, TWT, $L_{\max}$, NTJ, and WNTJ are important measures in terms of on-time delivery performance. In Table 3.2, more examples for direct performance measures are shown.

Table 3.2: Examples for direct performance measures

Class	Example
due date-oriented	maximum lateness $L_{max} := \max\{C_j - d_j \mid j = 1, \ldots, n\}$
throughput-oriented	TP of a certain work area
cycle time-oriented	ACT of the jobs of a certain product family
load-oriented	WIP

Note that in many practical situations, we have to deal with multiple criteria that are sometimes in conflict to each other. The desirability function approach in optimizing multiple criteria of interest was originally suggested by Derringer and Suich [66]. The approach transforms each objective value into a value between 0 and 1. Thus, each criterion y_i is converted into an individual desirability function d_i that varies over the range zero to one. If y_i is outside the acceptable range defined by the user, then $d_i = 0$. However, if y_i meets the goal, then $d_i = 1$. Let U_i be the maximum allowable value for the response y_i, and let G_i be the goal value for y_i. We define d_i as follows:

$$d_i := \begin{cases} 1, & \text{if } \; y_i < G_i \\ \left((U_i - y_i)/(U_i - G_i)\right)^{z_i}, & \text{if } \; G_i \le y_i \le U_i, \\ 0, & \text{otherwise} \end{cases} \tag{3.43}$$

where $z_i > 0$ is a real number known as the weight on the desirability function. When $z_i = 1$ for each objective i, the desirability function is linear. Choosing $z_i > 1$ places more emphasis on being close to the goal value, while setting $0 < z_i < 1$ decreases importance on proximity to the goal value. Once the individual desirabilities have been calculated, the combined desirability D that is to be maximized is computed as the geometric mean of the individual desirabilities. We obtain for the case of m desirabilities:

$$D := \left(\prod_{i=1}^{m} d_i \right)^{1/m}. \tag{3.44}$$

We will see applications of the desirability function approach related to dispatching and scheduling in Chaps. 4 and 5, respectively.

In contrast to direct performance measures, indirect performance measures are not directly related to the performance of the manufacturing system. They are used to evaluate properties of certain production planning and control algorithms. Consequently, they are related to the PS, PP, CS, and finally the CP. Robustness and stability measures are examples for this class of performance measures.

Performance measures related to robustness measure the change in the objective value of a plan or schedule after plan or schedule revisions. Stability of a plan or schedule is defined as the deviation of the final plan or schedule related to the original plan or schedule (cf. [106, 141, 233]). The deviation of two schedules can be measured, for example, as the difference of the completion times of jobs. Stability measures the difference of initial and of executed plans and schedules under the influence of disruptions. More indirect performance measures are shown in Table 3.3.

Table 3.3: Examples for indirect performance measures

Class	Example
re-planning effort-oriented	number of required re-planning activities
run time-oriented	run time of a certain production planning algorithm
agility	time needed to obtain the original WIP after the breakdown of a major bottleneck machine
stability	deviation of the final plan or schedule from the original one

Next, we have to discuss production planning and control approaches for comparison with the new approach. Decentralized and centralized production control approaches can be distinguished. Hierarchical approaches are somewhere between these approaches. Decentralized approaches work on local data, and coordination and cooperation issues become important in order to avoid the limitation of the myopic view of decentralized approaches. Dispatching rules in semiconductor manufacturing are an example of this type of approach (cf. Chap. 4 for more details on dispatching rules). Centralized approaches are given, for example, by LP, MIP, and by various kinds of neighborhood search techniques.

Often, it makes sense to use exact procedures like branch-and-bound or MIP described in Sect. 3.2 for small-size problem instances to ensure the correctness of implementations and to get a sense of the performance of the new approach. For large-size problem instances, heuristics are used for comparison. In some situations, it is possible to compare the performance of the new approach with the approach that is used in the company by considering problem instances or simulation models and the corresponding production planning and control instructions from the company.

We describe now the determination of problem instances and simulation scenarios. We start with the generation of synthetic problem instances. In the case of individual problem instances, we determine a set of factors that we expect to have an impact on the performance of the new algorithm. We consider levels for each factor, i.e., we vary the values of the factors in a controlled way. Often, the levels are selected as realizations of a random variable that follows a prescribed probability distribution. In the case of full factorial experiments, problem instances are generated for each factor combination. Usually, a certain number of stochastically independent problem instances are generated for each factor combination. Methods from the theory of designed experiments can be used to study the impact and the effect of different factors (cf. Montgomery [208]). One of the primary approaches to designing an experiment is to select the location of design points to optimize some criterion function; this is often called optimal design of experiments (cf. Pukelsheim [249]). Typically, this criterion is related to the variance/covariance matrix of the parameters in the model. The most popular optimization criterion is D-optimality, which seeks to minimize the volume of the joint confidence region of all the parameters in the model.

It is also possible to avoid the generation of problem instances by taking problem instances directly from the BP. Alternatively, for certain classes of problems, benchmark instances are publicly available (cf. the OR-Library [220] for a collection of such problem instances available over the web).

Performance assessment experiments for rolling horizon approaches are organized as different simulation scenarios. In contrast to the problem instance case, usually the number of scenarios is fairly small. A single simulation scenario is given by a set of independent parameters, a range of variation for the parameters, and a set of dependent variables, i.e., basically the performance measure values. Each scenario is represented by a specific simulation model. By using designed experiments (see Montgomery [208]) and meta-modeling techniques like response surface methodology (see Myers et al. [212] and Sect. 3.2.9), it is possible to reduce the number of required simulation runs by constructing an appropriate meta-model, often a regression model, and optimizing the performance measure values using the simpler meta-model instead of the simulation model. Furthermore, modern variance reduction techniques can also help to decrease the number of necessary simulation runs (cf. McAllister et al. [177]). Often, the simulation scenarios are based on data that is collected in a wafer fab. But at the same time, the scenarios are often also based on reference simulation models like the MASM Lab testbed (see also the description in Sect. 3.2.8). The usage of such public benchmark models offers some additional advantage as the results are now comparable because they are company-independent.

The choice of the performance assessment strategy is important because of the stochastic behavior of manufacturing systems. There are two main possibilities to assess the performance of a system in simulation modeling.

In the first case, the stationary behavior of a complex manufacturing system is of interest. After the warm-up period, the performance indicators of interest are measured. Long runs have to be performed. However, because of the stationary behavior, a relatively small number of independent runs is often enough (see Law [150]). The performance measure values are taken as the average of the performance measure values for the single runs. Besides average values that estimate the mean, confidence intervals are provided to estimate the range of values, which is likely to include the unknown performance measure value.

Looking at the transient behavior of a manufacturing system is the second possibility. Transient behavior appears during the transition phase from one system state to another. Product mix changes, ramp-up, and introduction of new machines lead to transient behavior of the manufacturing system. Studying a manufacturing system in the transition phase requires the measurement of the performance measures of interest in a time-dependent manner. As opposed to the stationary case, a considerably larger number of simulation runs is required in order to obtain statistically significant results for the performance measures. The length of a single run is determined by the length of the transition phase. The performance measure values are calculated as the average of the performance measure values obtained at a single point of time.

A performance assessment strategy is basically given by the decision about whether to investigate a stationary manufacturing system or a manufacturing system in the transient phase from a given stationary to another stationary behavior or not. Furthermore, the number of replications and the run length are also part of the performance assessment strategy. In the case of single problem instances, the maximum amount of computing time for each problem instance or the number of replications of computations are also elements of a performance assessment strategy.

In order to compare different production planning and control algorithms from a run time behavior point of view, it is required to fix a hardware and software environment for the performance assessment. It is furthermore required to describe the used load balancing strategy of the computer network in the case of distributed production planning and control algorithms. In the final step of the performance assessment scheme, it is required to run the experiments for all problem instances or all simulation scenarios. Then, the results have to be analyzed and interpreted. Based on the results, often several refinements of the design of experiments are necessary.

We note that additional performance measures that are related to the performance of the BS and the BP of wafer fabs can be found in Leachman and Hodges [155].

3.3.2 Architecture for Simulation-Based Performance Assessment

We continue by describing a simulation-based architecture that allows for simulation-based performance assessment of production planning and control approaches. There are two principle possibilities to incorporate production planning and control algorithms into a discrete-event simulation tool. In the first possibility, the production planning and control algorithm basically uses the information and data on the level of the simulation tool. As a result of this strategy, proprietary source code is obtained that is difficult to understand and to maintain. It is rather complicated to implement different production planning and control approaches. However, for comparison purposes, the ability to plug in different production planning and control approaches is highly desirable.

The second strategy is much more flexible. A blackboard-type data layer (see Mönch et al. [202]) that acts as a mirror of the base process BP emulated by the simulation model is used. The simulation tool is responsible for an update of the corresponding objects of the data layer in the case of the occurrence of well-defined events. When, for example, a job is released to the shop floor, i.e., it is started in the simulation, then a corresponding job object is created in the data layer. Other events of importance are, for example, the start of a setup operation of a certain machine or the completion of a certain process step, i.e., the job leaves a machine. In the latter case, the state of the job object is changed in the data layer, whereas in the first case, the state of the machine object is changed. Besides business objects like machines, jobs, and products, the data layer contains objects that represent the production planning instructions *mp* and production control instructions *mc*. Furthermore, it contains objects that are used to store statistics. The simulation engine implements the production control instructions *mc* in a dispatching manner.

The data layer is located in the memory of the computer; hence, fast access is possible. The incremental, event-driven update of the business objects avoids time-consuming queries from databases. The object model of the data layer is much easier to understand and to maintain than the proprietary data structures of a certain simulation tool. When the BS is segmented into different work areas, a separate blackboard-type data layer can be assigned to each segment of the manufacturing system. The architecture has to contain a persistency mechanism that supports object state and performance measure tracing. Using object-oriented databases seems to be appropriate for that purpose because of the highly nested objects from the data layer. The objects of the data layer are stored in a periodic manner.

The built-in mechanism for job selection and machine load of the discrete-event simulation tool acts as a dispatcher in a natural way. The dispatcher uses sorted lists with jobs, so-called dispatching lists, calculated using production control algorithms. When a machine becomes available in the simulation, the next job to be processed is determined from the dispatching list

of the machine. Furthermore, the simulation tool also specifies under what circumstances the production control algorithm is applied to calculate new dispatching lists. This feature is denoted as logic for calling of the production control approach.

The architecture is completed by a demand forecast module and by a demand generation module. While the first module is designated to the determination of forecast using historical demand, the second one generates concrete demand that is used as an important input for the planning algorithms of the PS. The PS determines which quantities have to be released in a certain point of time into the BS. This information is used by the CS to calculate the *mc*. The architecture is shown in Fig. 3.6.

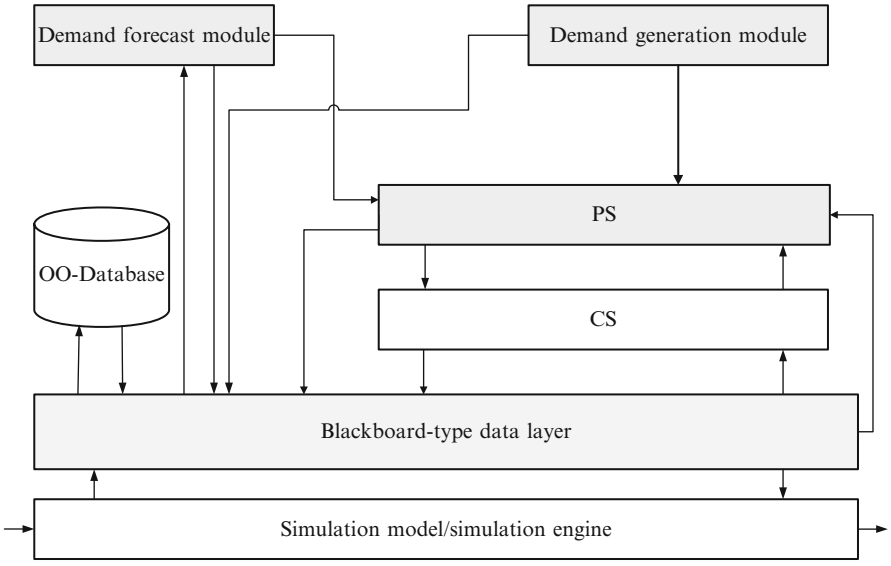

Figure 3.6: Simulation-based architecture for performance assessment

Building production planning and control applications from scratch is not very common because of available commercial and academic optimization and scheduling libraries and of available editors for dispatching rules. Examples are the ILOG solver and scheduler classes (cf. Le Pape [152]), class libraries for GAs (cf. Pain and Reeves [227]), and the dispatching rule editor from the simulation tool AutoSched AP. Therefore, an architecture for benchmarking has to take this fact into account. Because most of the available class libraries are written in the C++ programming language, the degree of freedom for choosing an appropriate implementation language for the architecture is limited. The suggested architecture allows for a plug-in of different production planning and control approaches.

In the described architecture, we only model the JS. However, an extension of this architecture that also includes the MS of a wafer fab is described by Driessel and Mönch [71]. In this situation, we have to use one simulation model for the JS and a second one for the MS. Note that we will provide certain examples for the usage of the simulation-based architecture for performance assessment of scheduling and production planning approaches in the remaining chapters of this monograph.

Chapter 4
Dispatching Approaches

In this chapter, we discuss dispatching approaches. Dispatching is on the lowest level of the PPC hierarchy described in Chap. 2. We start with a taxonomy of dispatching rules. A dispatching rule assigns a certain index to each job waiting in a queue to be processed on a machine or transported by a vehicle. The job with either the largest or the smallest index is selected to be processed or transported next. Typical attributes are the ready time, the processing time, or the due date of a certain operation or of the job itself. We describe simple dispatching rules that are characterized by an index that is based only on a small number of attributes of a job or the BS and the BP. We continue with the discussion of composite dispatching rules. Composite dispatching rules have an index that is formed by combining indices of simple dispatching rules.

Batching rules are an extension of dispatching rules. A batching rule helps to make the batch formation decision, i.e., which jobs are part of the batch, and also to decide which batch has to be processed next on an available machine. Some dispatching and batching rules use only information related to the jobs that wait in front of a machine or machine group, while other rules may use information regarding machines different from this machine or machine group. The second type of rules are called time-based look-ahead rules.

More sophisticated approaches like rule-based systems and several other approaches to find parameters in batching and dispatching rules are discussed. We present methods to weight indices so that a rule simultaneously works towards multiple objectives. Finally, we also describe an approach to discover appropriate dispatching rules using GAs and discrete-event simulation.

4.1 Motivation and Taxonomy of Dispatching Rules

Dispatching is on the lowest level of the PPC hierarchy described in Chap. 2. Dispatching rules are used to choose the next job that is processed or

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5_4,

transported by a resource. When the resource is a machine, a dispatching rule selects the next job to be processed among the jobs that are waiting in front of a machine group [29, 116, 240]. When the resource is given by a vehicle, the job is selected among the jobs that wait to be transported by the available vehicle. Note that this is a resource-based point of view, i.e., the resulting dispatching rules are resource-initiated. On the other hand, job-initiated dispatching rules are also possible. In this situation, jobs seek resources with free capacity. This might be important when no job is queueing in front of several available resources and a new job arrives. In this chapter, we focus on resource-initiated dispatching rules because they are more important during heavy material flow.

Dispatching rules are generally myopic in time and space, and it may be difficult to know how and when to adapt them to different situations on the shop floor. However, their decision logic is easy to understand, and they can be implemented with less effort on the shop floor of a wafer fab. Dispatching rules are still the main production control ingredients in many wafer fabs as indicated by Pfund et al. [234] and Sarin et al. [274].

A dispatching rule ranks all the waiting jobs according to an index

$$I_j(a_1,\ldots,a_k) := f(a_1,\ldots,a_k), \tag{4.1}$$

where j denotes the job and $a_i, i = 1,\ldots,k$ are attributes that determine the priority of j. They can be related to j or to properties of the BS or the BP. The right-hand side of expression (4.1) is a function $f : \mathbb{R}^k \to \mathbb{R}$. Often, the indication of all attributes will be suppressed, and only some of the attributes appear on the left-hand side of the priority index. We always assume that the job with either the largest or the smallest value of $I_j(a_1,\ldots,a_k)$ is selected as the job that will be processed or transported next. Based on the number and the nature of the attributes and the form of f, we can develop a taxonomy of dispatching rules.

Simple rules use only one to three attributes to determine the value of the priority index, i.e., $k \leq 3$. Note that the value $k \leq 3$ is somewhat arbitrary as the division between simple and composite dispatching rules. The function f often has the form $f(a_1) := a_1$, $f(a_1) := 1/a_1$, $f(a_1,a_2) := a_1/a_2$, or $f(a_1,a_2) := a_1 - a_2$. Composite dispatching rules are based on more than three attributes, i.e., $k > 3$. At the same time, the form of f is usually more complicated in case of composite dispatching rules; exponentiation, summation, or multiplication are often used. Composite dispatching rules combine several simple dispatching rules together.

A second way of classifying dispatching rules is according to the information on which they are based. A local dispatching rule uses only information that is related to the resource for which the jobs are in queue. On the other hand, a global rule is based on information regarding other resources. Somewhat related to global rules are look-ahead rules because they take future job arrivals from upstream machine groups into account.

A third way to classify dispatching rules is to make a distinction on whether the value of the priority index is time-dependent or not. Dynamic rules are time-dependent, i.e., the current time is an attribute in the priority index, while static rules are not time-dependent.

Besides the types of dispatching rules above, more complex dispatching rules can be constructed using truncation, conditioning, and multilevel approaches. Truncating a dispatching rule refers to the situation where jobs are selected according to a certain dispatching rule, excepting the case where a certain condition is not fulfilled any longer for the jobs. For example, a dispatching rule is used to choose the job to be processed next until at least one job has waited no longer than a specified time. Conditioning implies changing rules according to the BS and the BP state. One might switch back and forth, for example, between two different dispatching rules based on some measure of BS and BP congestion. Multilevel rules may be used to apply tie-breaking or secondary criteria; for example, one might first select the jobs with a small value according to a certain criterion and then use a second dispatching rule to select the job to be processed next among this subgroup of jobs.

In this chapter, we differentiate between simple and composite dispatching rules. We will discuss batching rules because of their practical relevance in wafer fabs. Look-ahead rules are discussed, due to the fact that they are important because of batching- and setup-related decisions.

Usually, discrete-event simulation is used to predict/assess the performance of dispatching approaches in semiconductor manufacturing. Each simulation tool owns a number of built-in dispatching rules. Often, new dispatching rules can be added by means of customizing the simulation tool. It is pointed out by Fowler et al. [87] that comprehensive testing of previously developed flow control approaches in realistic settings is needed. Finally, it is stated in [51, 307] that contradictory results with respect to the performance of dispatching rules are common in complex job shops. It is pointed out by Geiger et al. [96] that the only general conclusion from many years of research on dispatching rules is that there is no dispatching rule that outperforms consistently all the other rules under a variety of BS and BP conditions and performance measures.

We will see in Chap. 5 that dispatching rules can also be used to make scheduling decisions in the list scheduling framework. For some specific situations, even the application of simple dispatching rules within a list scheduling approach leads to the optimal solution. The main idea of this framework consists in applying dispatching in a repeated manner. Hence, a solid knowledge of dispatching rules is also beneficial for scheduling.

There is some relationship between dispatching and order release schemes. We will see in Chap. 6 that many papers support the thesis that when order release approaches become more effective, dispatching decisions will have a diminishing effect on the BS performance.

4.2 Simple Dispatching Rules

In this section, we differentiate between dispatching rules for selecting the next job to be processed on an available machine and to find the next job to be transported by an available vehicle.

4.2.1 JS-Related Dispatching Rules

The FIFO dispatching rule is a popular example of a simple dispatching rule. The corresponding priority index is given by

$$I_j := r_j. \tag{4.2}$$

The job with the smallest index is selected next. The FIFO rule is included as a default rule in virtually all discrete-event simulation packages. Another important example is given by the earliest due date (EDD) dispatching rule. Its priority index is as follows:

$$I_j := d_j. \tag{4.3}$$

EDD selects the job that has the smallest due date. The intention of the EDD rule is to ensure a high on-time delivery performance. Similar to FIFO, EDD is included in most simulation packages. A variant of the EDD rule is the operational due date (ODD) dispatching rule, where the due date of the job is simply replaced by a process step-specific due date d_{jk} for each process step k of job j. These local due dates for the process steps can be obtained from the due dates of the corresponding job by:

$$d_{jk} := d_j - \mathrm{FF} \sum_{h=k+1}^{n_j} p_{jh}, \tag{4.4}$$

where $\mathrm{FF} \geq 1$ is a constant that is called flow factor, p_{jh} is the processing time of operation h of j, and n_j denotes the number of process steps of job j.

Another important rule is the shortest processing time (SPT) dispatching rule. The corresponding index is

$$I_j := 1/p_j, \tag{4.5}$$

where p_j is the processing time of the current process step of job j. This dispatching rule was originally proposed for operating systems where the goal is to keep the number of jobs waiting for a processor as small as possible. It is well known that the SPT rule leads to small CT values in certain types of manufacturing systems because jobs with a small processing time will always be selected first. The application of this rule, a truncation variant of it, and a conditioning variant of SPT for wafer fabs is studied by Rose [267]. It turns out that SPT-type rules do not regularly reduce the ACT value, i.e., for some products this value increases, for some it decreases, and for most of

the products there is no effect. This behavior is caused by the fact that the coefficient of variation of the processing time of the operations is smaller than one for most of the machine groups of a wafer fab, i.e., the processing times of the operations are very similar. Hence, it is hard to predict changes in the CT values in wafer fabs using SPT. Note that the longest processing time (LPT) dispatching rule defined by the index

$$I_j := p_j \tag{4.6}$$

is of course the opposite to the SPT dispatching rule, i.e., the job with the largest processing time is selected first.

Jobs typically are classified in a wafer fab into regular jobs and hot jobs (cf. Sect. 2.2.3). As a result of this classification, different weights can be assigned to the jobs. The highest value first (HVF) dispatching rule selects a job with the highest weight first. The corresponding index of job j is given by

$$I_j := w_j. \tag{4.7}$$

When $w_j = 1$ for all regular jobs, the FIFO dispatching rule is often used as a tie breaker.

A generalization of the SPT rule that incorporates the weights w_j of the jobs is the weighted shortest processing time (WSPT) dispatching rule with index:

$$I_j := w_j / p_j. \tag{4.8}$$

The shortest remaining processing time (SRPT) dispatching rule works towards the selection of jobs that have only a small number of operations to be completed. Its priority index is given by the expression

$$I_j := \sum_{k=l}^{n_j} p_{jk}. \tag{4.9}$$

Totally, $n_j - l + 1$ process steps are necessary to complete job j, i.e., process step l is the current one.

Next, we discuss an example of a dispatching rule that results in good machine utilization and workload balance between the downstream machine groups. It is the fewest lots in the next queue (FLNQ) dispatching rule. This dispatching rule prioritizes jobs with the objective of balancing the workload on different machines. This is accomplished by prioritizing jobs heading to the next operation with the least number of jobs in its queue. The corresponding priority index is given by

$$I_j(t) := n(j), \tag{4.10}$$

where we assume that k is the current process step of job j. Then we denote by $n(j)$ the number of jobs waiting in front of the machine group that corresponds

to the next process step $k+1$ of j. FLNQ-type dispatching rules typically have poor performance in on-time delivery measures like TWT or flow time-related measures like ACT.

The least setup cost (LSC) dispatching rule selects the job to be processed next whose operation requires the smallest setup time among the jobs queueing in front of the machine group. The corresponding priority index is given by

$$I_j := s_{kj}, \tag{4.11}$$

where we denote by s_{kj} the setup time that is needed to process j given that job k was the most recent job processed on the machine. Therefore, the LSC rule is a setup avoidance rule, i.e., when there are jobs that require the current setup state on the machine, then these jobs are selected among the queued jobs. This is a dispatching rule commonly used by many practitioners for dispatching and scheduling problems with sequence-dependent setup times. Whenever there is a job and a machine available, the LSC rule searches for the machine/job combination that causes the least setup time and selects this job to be processed on that machine. When a job is available and the amount of setup between two or more available machines is the same, one will be selected randomly. However, one could also use other tie breakers. We will later see in Sect. 4.3.2 how this rule is used to construct composite dispatching rules.

The flow control (FC) dispatching rule is somewhat similar to the FLNQ rule. The FC dispatching rule calculates the number of remaining production hours per machine for the next machine group in the product's route. More formally, the corresponding index is given by

$$I_j := \frac{m}{\sum_{l \in J(M)} p_l}, \tag{4.12}$$

where we again assume that k is the current process step of j. The quantity m is the number of machines in the machine group M that corresponds to the next process step $k+1$. Finally, we sum up the processing times p_l of all jobs queueing in front of this machine group. This job set is denoted by $J(M)$.

While the dispatching rules discussed so far are general purpose rules, we now discuss an important class of simple dispatching rules for wafer fabs. As defined by Lu et al. [169], fluctuation smoothing policies are a subclass of the least slack (LS) dispatching rule. Known as minimum slack (MS), these dispatching rules give the highest priority to those jobs where the slack is the smallest. The slack of job j that is in buffer b_i is defined by

$$s(j) := \beta(j) - \gamma_i, \tag{4.13}$$

where $\beta(j)$ is the real number attribute of j that is associated with j when it enters the wafer fab. Furthermore, the buffer b_i is associated with the real number γ_i. Of course, we set again

$$I_j := s(j) \tag{4.14}$$

for the resulting index. We will see that different dispatching rules can be obtained from expression (4.13) using particular choices for $\beta(j)$ and γ_i. Among them, there are fluctuation smoothing policies. These policies are used to reduce the mean and standard deviation of CT. This is important because by reducing ACT, the product can get to market faster and can keep up with the changing environments associated with semiconductor manufacturing. Also by reducing the Var(CT) value (see Sect. 3.3.1), companies can predict the completion time of a product much more accurately.

The first fluctuation smoothing policy is a policy for reducing the Var(L) value. It evaluates the slack of a job and lets the one with the least slack process on the available machine first. In this situation, the slack is defined by setting $\beta(j) := d_j$ and $\gamma_i := \xi_i$, where ξ_i is an estimate for the remaining CT of j at the process step that is associated with b_i. Then the quantity $d_j - \xi_i - t$, where t is the current time, is a measure for the urgency of job j. Because all the jobs queueing in buffer b_i have the current time t as a common term, it can be ignored, and consequently it is enough to consider $d_j - \xi_i$ as slack. This rule is also known as the LS dispatching rule. In this situation, we use the crude estimate $\xi_i := \sum_{k=l}^{n_j} p_{jk}$ for the remaining processing time of job j in the wafer fab to obtain the classical global LS rule. Note that the dispatching rule based on index (4.14) attempts to make each job equally late or equally early. The Var(L) value is small when all jobs are either equally too early or too late. Therefore, the resulting dispatching rule is called the fluctuation smoothing for variance of lateness (FSVL) policy (see Lu et al. [169]).

The second fluctuation smoothing policy is used to reduce the Var(CT) value. Therefore, we call the resulting policy FSVCT. It also evaluates the slack of a job and places the job with the least slack on the machine that is associated with buffer b_i next. The slack for this rule is defined by setting $\beta(j) := r_j$ and again $\gamma_i := \xi_i$. This is of course a CT-related measure. Hence, FSVCT attempts to reduce the Var(CT) by a similar argumentation as for the FSVL policy.

The third fluctuation smoothing policy is intended to reduce the ACT value. The resulting dispatching rule is called fluctuation smoothing for mean cycle time (FSMCT) policy. It is known from queueing theory (cf. the description in Sect. 3.2.7 and the Kingman approximation) that the delay of jobs at a server is caused by the burstiness of the job arrivals and the variations in the service time. We cannot influence the service times. Because of this, following Lu et al. [169], we are interested in simultaneously reducing the burstiness of arrivals to all the buffers of the wafer fab. The resulting policy is therefore appropriate for reducing the ACT value.

We start by describing how we reduce the burstiness in the arrivals to buffer b_{k+1}. The basic idea is to set periodic due dates for jobs to reach b_{k+1}. We denote by λ the mean release rate for the wafer fab. Hence, the mean inter-arrival time between jobs is given by $1/\lambda$. Therefore, we set n/λ as the

due date to reach b_{k+1} of the nth job that is released into the wafer fab. When we are able to reduce the variance of lateness Var(L) for reaching b_{k+1}, we obtain an arrival stream at b_{k+1} that is almost deterministic and therefore not bursty.

In order to reduce Var(L) for reaching b_{k+1}, we consider b_{k+1} as the final sink of the wafer fab and use FSVL with respect to this system. We define by ξ_i^k an estimate for the remaining partial CT to go from b_i to b_{k+1}. In this situation, we define $\beta(j) := n/\lambda$, where n is the nth job released into the wafer fab and $\gamma_i := \xi_i^k$. Of course, we have

$$\xi_i^k = \xi_i - \xi_{k+1}, \tag{4.15}$$

and therefore $\gamma_i = \xi_i - \xi_{k+1}$. But because we consider a fixed b_{k+1}, the summand ξ_{k+1} is common for all jobs at the buffers $b_i, i = 1, \ldots, k$ and can be omitted. Therefore, we use

$$s(j) = n/\lambda - \xi_i. \tag{4.16}$$

The expression (4.16) is so far only defined for jobs in buffers $b_i, i \leq k$. Because we are interested in simultaneously reducing the burstiness of job arrivals at all buffers, we extend expression (4.16) to all buffers.

However, it is important to come up with appropriate ξ_i values in expression (4.16). It is discovered in Lu et al. [169] that iterative simulation is quite effective to solve this problem. As described in Sect. 3.2.8, the initial value $\xi_i^{(0)} \equiv 0$ is used within the first simulation run. We obtain estimates $\hat{\xi}_i^{(0)}$ from the first simulation run for the CT of the remaining process steps that are associated with b_i. The new setting $\xi_i^{(1)} := \hat{\xi}_i^{(0)}$ is used in the second simulation. This procedure is repeated in an iterative manner until the difference between two consecutive $\hat{\xi}_i^{(s)}$ and $\hat{\xi}_i^{(s+1)}$ values is small. We refer to Sect. 4.7.2 where more implementation details for a similar problem are presented.

Finally, we refer to Sarin et al. [274] for a more complete description of various simple dispatching rules used in wafer fabs.

4.2.2 MS-Related Dispatching Rules

In this section, we describe how the next job is selected to be transported by a vehicle. As in the case of JS-related dispatching rules, we specify the corresponding dispatching rules by priority indices. In MS-related dispatching rules, we assume that a given set of transportation jobs is waiting to be transported. For simplicity reasons, we assume that each transportation job consists of one job that is to be processed within a wafer fab, usually a FOUP. Performance measures of interest for AMHS are the number of carrier moves, i.e., TP and CDT (see Sect. 3.3.1).

The nearest job first (NJF) dispatching rule is an intuitive greedy heuristic that is defined as follows. An empty vehicle is dispatched to the job whose waiting point is the nearest to the current location of the empty vehicle c. The following index is used to assess job j:

$$I_j := d(j,c), \tag{4.17}$$

where $d(j,c)$ denotes the length of the shortest path between j and c. The job with the smallest index will be selected. It is clear that NJF can lead to high effective utilization of the AMHS, i.e., loaded travel. However, this rule does not take into account the waiting time of jobs for an empty vehicle. Therefore, there is the possibility that some jobs wait a long time for a transport, i.e., the variance of CDT is large. Another limitation of NJF is that this rule becomes inefficient in situations when there are several vehicles available to transport jobs, but they are being blocked by the first vehicle (cf. Liao and Fu [161]).

The first limitation of the NJF dispatching rule is resolved by the longest waiting time (LWT) dispatching rule. It dispatches an empty vehicle c to the job with the LWT among the jobs that are ready for transportation. The corresponding priority index is given by:

$$I_j := wt_j, \tag{4.18}$$

where we denote by wt_j the waiting time of job j for transportation. The job with the largest index is selected next. It is clear that the LWT rule is similar to the FIFO dispatching rule for the BS. Of course, the LWT rule tends to reduce the variation of CDT at the expense of effective vehicle utilization. The second limitation of the NJF rule is considered within the design of the farthest job first (FJF) dispatching rule. Blocking effects can be avoided by dispatching an empty vehicle to the farthest job away from it. The corresponding index is given by expression (4.17). However, we select the job with the largest value of the index in this situation.

The modified nearest job first (MNJF) dispatching is proposed by Liao and Fu [161] to combine the advantage of the NJF and the LWT rule. The rule works as follows. An empty vehicle is dispatched to the job with the longest waiting time, when this time is longer than a threshold value τ. When the set of those jobs is empty, the NJF dispatching rule is applied to determine the job that is selected next. The MNJF rule is a truncation dispatching rule.

The results of simulation experiments provided in [161] show that NJF performs well with respect to TP and average delivery time. But at the same time, the variation in CDT is large when NJF is used. The performance of the MNJF rule, however, is very close to that of NJF, but the CDT variation is much smaller because of the truncation strategy.

Dispatching rules for AMHS that take hot jobs into account are proposed by Liao and Wang [162]. These rules allow for an almost no-wait transport of these high-priority jobs. Additional job-initiated dispatching rules for the MS

are described by Lin et al. [163]. Note that in addition to location and waiting time-related attributes, attributes that take the situation at the different buffers associated with an AMHS into account can be used.

4.3 Composite Dispatching Rules

In this section, we discuss two classes of JS-related composite dispatching rules and one MS-related composite dispatching rule.

4.3.1 Critical Ratio Dispatching Rules

We start with the critical ratio (CR) dispatching rule. It is defined as the ratio of EDD- and SRPT-type rules. Its priority index is given by the following expression:

$$I_j(t) := \frac{d_j - t}{\sum_{k=l}^{n_j} p_{jk}}, \tag{4.19}$$

where l denotes the current process step of j. The job with the smallest CR index is selected first because in this situation, $d_j - t$ is small relative to the remaining processing time, i.e., the job is already late or has only a small amount of slack. Note that the priority index given by expression (4.19) is negative when $t > d_j$. In this situation, the job is already late, as its due date has passed. When $0 \leq I_j(t) \leq 1$, then the job j will most likely be late. Finally, an on-time job will have a critical ratio not smaller than one. A small value for $\sum_{k=l}^{n_j} p_{jk}$ often means that only a small number of process steps to be performed are left. Of course, the numerator $d_j - t$ in expression (4.19) can be replaced by $d_j - \sum_{k=l}^{n_j} p_{jk} - t$. However, this leads to the constant term 1 in the priority index value of all jobs and can therefore be omitted. Sometimes, the following dispatching rule

$$I_j(t) =: \begin{cases} (d_j - t)/\sum_{k=l}^{n_j} p_{jk}, & \text{if } t \leq d_j \\ 1/((t - d_j)\sum_{k=l}^{n_j} p_{jk}), & \text{otherwise} \end{cases} \tag{4.20}$$

is also referred to as the CR rule (see Rose [268]). This rule is an example of a conditioning dispatching rule. When $d_j \geq t$, then a priority index similar to index (4.19) is used, whereas in the case of $d_j < t$, a large value for $t - d_j$ and a large value for $\sum_{k=l}^{n_j} p_{jk}$, i.e., a job that is already late has to perform many process steps, lead to a small value of the priority index.

Results of simulation experiments with CR-type dispatching rules can be found in Rose [268]. This study shows that an appropriate due date setting has a large impact on the performance of CR-type dispatching rules. It is shown that in the case of rather tight due dates, CR-type rules do not perform well with respect to on-time delivery performance and CT.

4.3.2 ATC-Type Dispatching Rules

The apparent tardiness cost (ATC) dispatching rule is proposed by Vepsalainen and Morton [311]. It attempts to reduce TWT values. It is a composite dispatching rule that blends the WSPT and the LS dispatching rules. Its priority index is given by

$$I_j(t) := \frac{w_j}{p_j} \exp\left\{-\frac{(d_j - p_j - t)^+}{\kappa \bar{p}}\right\}, \tag{4.21}$$

where $\bar{p}$ is the average processing time of the jobs that are queueing in front of the machine group, t is the time where the next machine becomes available, and finally κ is a scaling parameter that weighs the slack against the WSPT priority index. We use the notation $x^+ := \max(x, 0)$ for abbreviation. The κ parameter is called the look-ahead (or scaling) parameter. Note that there is an implicit scaling parameter 1 on the WSPT term. It is well known that the performance of ATC-type dispatching rules strongly depends on an appropriate setting of the κ parameter.

As ATC-type rules are found to be sensitive to the κ value setting, ten different κ values from 0.5 to 5 in increments of 0.5 are used by Mehta and Uzsoy [180]. This approach is called grid search in Pfund et al. [236]. For a particular situation, i.e., a certain number of jobs waiting for processing, the ATC rule is used independently for each κ value, the TWT value is calculated for all queueing jobs, and finally the κ value is chosen that provides the smallest TWT value. The κ value chosen is then fixed for a given job set within ATC-type dispatching rules.

Various rules for the selection of this parameter are discussed in Pinedo [240]. The tightness of the due dates, the range of the due dates of the jobs queueing in front of the machine group, and the number of jobs per machine can be used to determine the look-ahead parameter. A heuristic curve-fitting method is used to determine the equations for calculating proper values of the look-ahead parameters. Neural networks are proposed by Kim et al. [140] to find appropriate look-ahead parameters.

There are generalizations of the ATC dispatching rule with respect to sequence-dependent setup times and unequal ready times of the jobs. We consider first the case of sequence-dependent setup times. Because we often have to deal with sequence-dependent setup times, an extension of the ATC dispatching rule to this situation is proposed by Lee and Pinedo [159]. This dispatching rule combines the WSPT dispatching rule, the LS rule, and the LSC rule into a single priority index. The corresponding priority index of job j at time t when job l is processed is given by

$$I_j(t,l) := \frac{w_j}{p_j} \exp\left\{-\frac{(d_j - p_j - t)^+}{\kappa_1 \bar{p}}\right\} \exp\left\{-\frac{s_{lj}}{\kappa_2 \bar{s}}\right\}, \tag{4.22}$$

where s_{lj} denotes the setup time that is required to process job j immediately after job l. The average setup time of the waiting jobs is denoted by $\bar{s}$. This dispatching rule is called ATC with setups (ATCS) for abbreviation. Different expressions to determine appropriate values for κ_1 and κ_2 based on attribute values of the waiting jobs are proposed by Lee and Pinedo [159]. Neural networks are used for the same purpose in Park et al. [229]. Chen et al. [45] develop a four-phase method to determine a set of scaling parameter values that perform well over a wide range of problem instances, i.e. provide robust performance. In the first phase, factor ranges that characterize the problem instances in each machine group are calculated. In the second phase, a face-centered cube design is used to decide the placement of design points in the factor region. The third phase involves adding an explicit scaling factor for the WSPT term and then using designed experiments to find good scaling parameter values at each design point. In the last phase, the central point of the area in which all of the good scaling parameters lie is identified with the robust scaling parameter.

In the case of unequal ready times, it makes sense to wait for a future job arrival in some situations. The resultant dispatching rule is called ATCR, and the corresponding priority index is given by

$$I_j(t) := \frac{w_j}{p_j} \exp\left\{-\frac{(d_j - p_j - t)^+}{\kappa_1 \bar{p}}\right\} \exp\left\{-\frac{(r_j - t)^+}{\kappa_2 \bar{p}}\right\}, \tag{4.23}$$

where κ_1 and κ_2 denote the look-ahead parameters for the slack- and the ready-time-related terms of the priority index, respectively. The slack-related term can be motivated in a similar way as in the ATC priority index (4.21) without ready times, while the ready-time-related term is responsible for reducing the job priority when a job is not ready at t. Results of computational experiments for an approach based on inductive decision trees to select the two look-ahead parameters are presented by Zimmermann et al. [332].

The index (4.22) is extended by Pfund et al. [236] to the situation where ready times of the jobs occur. The resulting priority index is called ATCSR. It is given by

$$I_j(t,l) := \frac{w_j}{p_j} \exp\left\{-\frac{(d_j - p_j - \max(r_j,t))^+}{\kappa_1 \bar{p}}\right\} \exp\left\{-\frac{s_{lj}}{\kappa_2 \bar{s}} - \frac{(r_j - t)^+}{\kappa_3 \bar{p}}\right\}. \tag{4.24}$$

A grid search approach is used to determine appropriate $(\kappa_1, \kappa_2, \kappa_3)$ triples. Furthermore, regression-based approaches are proposed for the same problem. Note that it is also possible to use more sophisticated approaches to calculate the slack of a job within priority indices of ATC-type rules.

Usually, ATC-type dispatching rules lead to small TWT values. However, there is some effort required to find appropriate look-ahead parameters. The results of extensive simulation experiments with different composite dispatching rules in wafer fabs are presented by Bahaji and Kuhl [16].

4.3.3 Composite Dispatching Rules for the MS

Following Jeong and Randhawa [128], we present a dispatching rule for vehicles that combines several attributes to achieve multiple performance measures simultaneously. We assume that each machine has an input buffer and an output buffer. The corresponding priority index is given as follows:

$$I_j := \alpha_1 D_j + \alpha_2 \mathrm{IQ}_j + \alpha_3 \mathrm{OQ}_j, \tag{4.25}$$

where D_j is the unloaded travel distance of an idle vehicle to job j, IQ_j is the remaining space in the input buffer of a machine that is the destination of j, and OQ_j is the remaining space in the outgoing buffer of a machine that is the source of j. The quantity D_j is defined as follows:

$$D_j := \begin{cases} \frac{\max d_k - d_j}{\max d_k - \min d_k}, & \text{if } \max d_k \neq \min d_k \\ 1, & \text{otherwise} \end{cases}, \tag{4.26}$$

where d_k is the distance of the current location of the available vehicle to the machine where job k is in the output buffer. The quantity IQ_j is given by

$$\mathrm{IQ}_j := 1 - n_j^{iq}/c_j^{iq}, \tag{4.27}$$

where n_j^{iq} is the number of jobs in the current input queue of the machine that is the destination of j. The quantity c_j^{iq} represents the capacity of the incoming buffer for the machine to which job j, which is the first job in the output buffer of a machine, is going to move. Finally, the quantity OQ_j is given by

$$\mathrm{OQ}_j := n_j^{oq}/c_j^{oq}, \tag{4.28}$$

where we denote by n_j^{oq} the current number of jobs in the output buffer of the machine that contains j, and c_j^{oq} is the capacity of this output buffer.

The first part of the rule prioritizes the job that is the closest to the newly available empty vehicle, while the second part tends to prefer jobs that are going to machines that have a low queue size in their input buffers. The third part prefers jobs in the output buffers of machines that have a large queue length of the outgoing buffer. Furthermore, the three parts of dispatching rule (4.25) are weighted by using $\alpha_i \geq 0$ and $\alpha_1 + \alpha_2 + \alpha_3 = 1$. A large value of the first part leads to a high utilization of the AMHS, whereas a large value for the second part leads to a decreased value of carrier waiting time because a destination machine whose input buffer is full is not able to accept another job unless the queue is cleared. The third part reduces the probability of machine blocking.

Note that JS- and MS-related dispatching rules are generally considered separately. There are only a few papers that treat them simultaneously (cf. Tyan et al. [300], for example).

4.4 Simulation Results for Assessing Dispatching Rules

We describe the results of a simulation study due to Bullock et al. [37] using the MIMAC 1 model [83] and the FIFO, EDD, SPT, ATCS, FSVL, FSVCT, and the FSMCT dispatching rules. The ATCS dispatching rule is applied only for the steppers. For the remaining machines, the FIFO dispatching rule is used in this situation. The model contains two products, 83 machine groups, and 32 operators. Reentrant flows and rework are contained in the model. Other characteristics of the MIMAC 1 model are shown in Table 4.1.

Table 4.1: Characteristics of the MIMAC 1 model

Product	Weight	Wafers per job	Release size (in jobs)	Constant time until next release (hours)	Raw processing time (days)
1	1	48	1	3.034	13.1
2	5	48	1	6.048	14.9

We consider the performance measures ACT, Var(CT), Var(L), TP, AT, and finally TWT. The simulation time is three years. The first year is truncated to eliminate the initialization bias. Because due dates are not included in the MIMAC 1 model, we set due dates according to

$$d_j := r_j + \sum_{k=1}^{n_j} p_{jk}. \tag{4.29}$$

Ten independent replications of each simulation run are performed to obtain statistically reasonable results. The results of simulation experiments are shown in Table 4.2.

Table 4.2: Simulation results for different dispatching rules

Rule	ACT (days)	Var(CT)	TP (jobs)	AT (days)	Var(L)	TWT (days)
FIFO	38.09	13.67	5,723	24.41	28.63	452,679.10
EDD	25.66	0.15	5,823	11.98	0.26	238,774.59
SPT	25.62	0.03	5,830	11.94	0.30	253,631.29
ATCS	29.34	3.65	5,813	15.66	11.14	189,516.95
FSVL	26.08	0.19	5,827	12.40	0.34	256,435.18
FSVCT	26.53	0.04	5,832	12.85	0.15	275,690.00
FSMCT	25.74	0.04	5,837	12.06	0.07	250,554.04

It turns out that FIFO is outperformed by the remaining dispatching rules with respect to all performance measures because it does not take into account any of the attributes of the jobs except the ready time. The SPT rule performs best with respect to CT and Var(CT). The different fluctuation smoothing policies perform well with respect to Var(CT) and Var(L). As expected, the ATCS dispatching rule outperforms the other rules with respect to TWT, followed by the EDD rule. It is interesting to note that the ATCS rule does not perform well with respect to AT. This can be explained by the fact that this dispatching rule takes the weights of the jobs into account. Only the FIFO rule performs worse.

Related simulation experiments for larger wafer fabs can be found in Mittler and Schömig [186, 187]. The fluctuation smoothing policies especially show a similar behavior in the case of large-scale models. Simulation results for ATC-type dispatching rules in large-scale wafer fabs are presented by Mönch and Zimmermann [200].

4.5 Batching Rules

In this section, we only consider the case of parallel batching, i.e., several jobs can be processed at the same time on the same machine. We assume that at most B jobs can be batched together, i.e., B is the maximum batch size. We note that it is possible to make batch formation decisions based on the number of wafers instead of the number of jobs, but in this monograph, all decisions are based on the number of jobs. The set of jobs that can be used to form a batch is called a family as described for diffusion furnaces in Sect. 2.2.3. We assume that we have f incompatible job families.

Batching rules can be seen as a generalization of dispatching rules, i.e., we have $B = 1$ in the case of pure dispatching rules. When a batch-processing machine becomes available, the next batch has to be formed and then selected to be processed on this machine. Therefore, batch formation is an additional decision when compared to a pure dispatching rule. We assume in the beginning that a large number of jobs are queueing in front of the batch machine group.

One possible way to solve this problem consists of selecting a job among the queueing jobs using one of the dispatching rules described in Sect. 4.2 or 4.3. Then at most $B-1$ additional jobs are selected according to certain criteria among the jobs that are queueing in front of the batch machine group.

The following algorithm is used to determine the batch to be processed next, when only one priority index I_{ij} is used to assess all the jobs j of family i. We assume that jobs with large I_{ij} are selected next to be processed within a batch.

Batching algorithm (BA)

1. Sort all the job within each family $i = 1, \ldots, f$ in nonincreasing order with respect to I_{ij}.

2. Let us denote the length of the list that corresponds to family i by $l(i)$. Consider the first $\min(B, l(i))$ jobs of each family to form batch $B(i)$ of family i. Denote the number of jobs within $B(i)$ by $|B(i)|$.
3. Select the batch with the largest value

$$I_{B(i)} := \frac{|B(i)|}{B} \sum_{j \in B(i)} I_{ij} \tag{4.30}$$

among the families $i = 1, \ldots, f$ to be processed next.

It is clear from expression (4.30) that we assess each batch by taking the sum of the priority indices of the jobs that form the batch. The factor $|B(i)|/B$ makes sure that full batches are preferred compared to batches that contain only a small number of jobs.

We consider two examples for this approach. We may use the EDD dispatching rule to determine the most important jobs within each family. We have to modify priority index (4.3) to $I_{ij} := -d_{ij}$, where we denote by d_{ij} the due date of job j of family i to align with the algorithm BA. Similarly, we can use the ATC dispatching rule with priority index (4.21). The resulting batching rule is called the batched ATC (BATC) rule.

Next, we study the case where not enough jobs are available to form a full batch. In this situation, a decision has to be made whether to start the batch that is not full or to wait until enough jobs are available. We start with the single product case. Let L be the number of jobs in the queue in front of the batch machine group. The resulting decision rule can be formulated as follows.

Algorithm Minimum Batch Size (MBS)

1. Anytime there are at least S jobs in the queue with $S \leq B$, then a batch can be processed. The quantity S is called the minimum batch size.
2. When there are fewer than S jobs in the queue and a machine is available, the machine will remain idle.

Two special cases of the MBS rule are important, namely MBSG where $S = 1$, called the greedy batch policy, and thereby loading the machine every time a new job appears; and MBSF where $S = B$, called the full batch policy, and consequently not loading the machine until it can be loaded at full capacity.

The algorithm MBS has one advantage that makes it appealing to the fab manager. It is extremely easy to implement on the shop floor. This is because MBS requires minimal computation and real-time information to make a decision. The MBS rule is considered as a theoretical standard that is used to assess the performance of other batching policies. This is discussed by Deb and Serfozo [62] who showed that with Poisson arrivals, the MBS rule is the optimal policy among those that only consider the current status of the BS.

However, MBS also has several disadvantages. The first drawback of MBS is how to determine an appropriate minimum batch size prior to, and

during, implementation. Although Gurnani et al. [111] introduce an algorithm for computing the critical number S, the calculation of this value in practice is often quite difficult considering that product mix and production rates typically change according to business forecasts. As a result, the optimal MBS will change as production changes. Another drawback of the MBS algorithm is that it fails to consider the current state of the BS as well as the impact of its dispatching decisions on the entire system. Thus, the MBS batching decision may result in wasted capacity if the MBS chosen is less than the capacity of the machine or if additional idle time is necessary to form a batch. Ultimately, MBS provides local decisions.

Of course, it is possible to combine BA and MBS. This offers possibilities to extend MBS to the multiproduct situation. In this case, we apply MBS to the families where $|B(i)| < B$. Then, we apply the index (4.30) to assess the batches formed for the families that can offer a batch to be processed next.

It is a weakness of the algorithms BA, MBS, and the resulting hybrids that they consider only jobs that are available at the time when the batch has to be formed. But it is intuitively clear that it is reasonable to exploit knowledge of the expected state of the BS. Therefore, it sometimes makes sense to start a non-full batch or to decide to wait for jobs that arrive in the future to increase the fullness of batches. We will study the corresponding look-ahead rules in Sect. 4.6.

4.6 Look-Ahead Rules

Look-ahead rules are dispatching or batching rules that take information related to future job arrivals into account. Such information is important in the case of sequence-dependent setup times and in the case of batch processing. This kind of real-time information is available from the MES in most wafer fabs. In the first case, information with respect to future job arrivals might avoid expensive setup changes by waiting for a job that requires the same setup state, but it is not available now. In the latter case, in some situations, it is reasonable to wait for future job arrivals to increase the fullness of a certain batch. In the remainder of this section, we discuss a rule that dynamically selects the batch size and a rule that makes decisions based on the next arrival of jobs. Additional look-ahead research is also briefly discussed. Finally, we describe batching rules that are generalizations of the algorithm BA using ATC-type heuristics.

4.6.1 Dynamic Batching Heuristic

One possibility for improving the MBS rule is to make more intelligent decisions on batching and possibly wasting less capacity. This can be accomplished by using real-time information to gain knowledge about future

states of the BS. As a result, Glassey and Weng [101] introduce the dynamic batching heuristic (DBH), a heuristic that incorporates real-time data into the decision process.

DBH dynamically determines the batch size of the batch to be formed and processed next based on the BS status at time t_0 and whether idle time to wait for additional jobs should be inserted to minimize total overall delay at the batch machine. It is shown in [101] that the look-ahead procedure of DBH is beneficial with respect to average delay at the batch machine, i.e., the average waiting time of the jobs in queue.

In order to sketch the main idea of DBH, we introduce the following additional notation:

t_0 : time epoch that the batch machine is idle and the number of jobs in the queue is positive
t_j : arriving epoch of the next jth job after t_0
q : number of jobs in queue at t_0
T : processing time
L : look-ahead number, where we assume that the next L arrival epochs are known with certainty at t_0

DBH is proposed for a single batch machine and a single product. The DBH formulation is based on the following two insights:

1. The time of loading the batch-processing machine is either the time that it becomes idle and there are jobs waiting or, if it has been idle, at the time when some job arrives.
2. The batch machine starts service right away, when $q \geq B$. Waiting will only result in more delay under such circumstances.

Since it becomes difficult to predict the later arrivals in a long planning horizon, the DBH is proposed for operating the batch machine in a planning horizon that is equal to the processing time T of the product. This situation is shown in Fig. 4.1. The overall heuristic can be summarized as follows.
Algorithm DBH

1. If the batch machine becomes idle, we have to differentiate two cases. In the first case, i.e., when jobs are in the queue, go to step 2. Otherwise, if the queue is empty, go to step 3.
2. Let t_0 be the time epoch that the batch machine becomes idle. Start the decision heuristic described below.
3. Wait until a job arrives. Let t_0 be its arrival epoch. Start the decision heuristic described below.

We now describe the decision heuristic used in the DBH algorithm. We assume $q < B$ because otherwise we will always start processing a full batch. When the q jobs available at time t_0 are processed immediately at time t_0, the other jobs arriving at t_j, where $t_j < t_0 + T$, will stay in queue at least $T + t_0 - t_j$. On the other hand, when we wait for j job arrivals and then load $q + j$ jobs at time epoch t_j, then each queued job has to wait $t_j - t_0$ time units. Therefore,

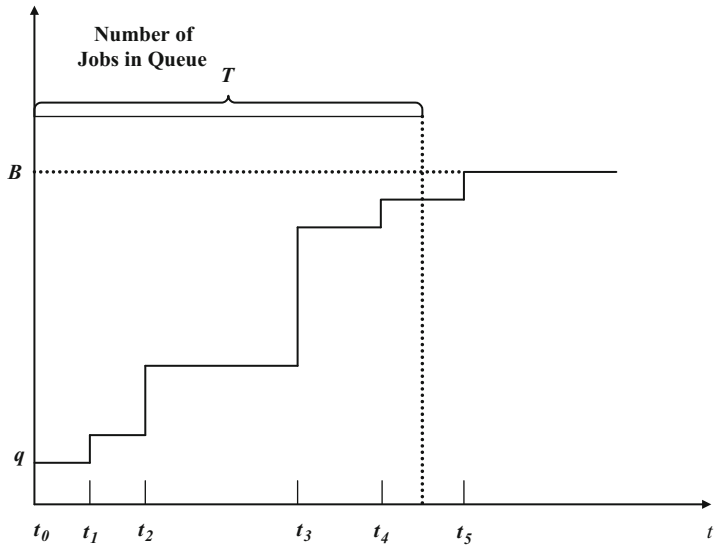

Figure 4.1: Arrival of jobs in the queue of a batch machine over time

the total delay will increase by $(t_j - t_0)q$. The job that arrives at t_j will be processed immediately. Therefore, the delay will be decreased by $(T + t_0 - t_j)j$ because the j jobs that arrived after t_0 will be processed and not have to wait longer. Therefore, the determined net saving is

$$\text{Net}(t_j) = (T + t_0 - t_j)j - (t_j - t_0)q. \tag{4.31}$$

When $\text{Net}(t_j) > 0$, then the delay can be reduced by waiting until t_j to start the batch. In contrast, when $\text{Net}(t_j) < 0$, then the corresponding delay is increased. In case of $\text{Net}(t_j) = 0$, there is no gain or loss. Therefore, it is reasonable to wait until t_i, where i is given by

$$i = arg\max_j \left\{ (T + t_0 - t_j)\, j - (t_j - t_0)\, q \mid 0 \le j \le j_{\max} \right\}. \tag{4.32}$$

We have

$$j^* := arg\max_j \{ t_j \mid t_j \le t_0 + T \}, \tag{4.33}$$

i.e., j^* denotes the maximum number of arrivals within T, and we define $j_{\max} := \min\{j^*, B - q, L\}$. Therefore, $t_{j_{\max}}$ is the last possible loading epoch. It is determined by the look-ahead number L and T. Of course, the setting $L \le B - 1$ makes sense because of the maximum batch size of B and $1 \le q$.

Although DBH performs better than MBS in all situations, it requires much more computation than MBS and also requires real-time information

to make dispatching decisions. Likewise, the number of jobs to look ahead is necessary before DBH can be implemented. It is also inflexible with respect to the planning horizon length.

4.6.2 Next Arrival Control Heuristic

Half of the potential benefit in using DBH is gained by looking ahead only to the next arrival (see Glassey and Weng [101]). This is the starting point for the next arrival control heuristic (NACH) proposed in [84]. Fowler et al. [84] note also that the further ahead one looks, the greater the potential impact of the decision on arrivals that occur outside of the time horizon of T time units as suggested by Glassey and Weng [101]. An extension of the original NACH approach for parallel batch machines and multiple products is provided by Fowler et al. [85].

In this monograph, we describe NACH in a slightly generalized context, where the status of critical machines in subsequent downstream processing is taken into account during batch processing decision-making [287]. We describe a methodology that is intended to balance the time a job spends waiting for batching with the time spent in setup at downstream machines. The resulting heuristic is called NACH-setup. We will use the following notation to describe NACH-setup:

q_j : number of jobs in queue for product j
N : total number of products
T_j : processing time of a batch of product j
S_j : downstream setup required by product j given the current status of the BS
W_j : weighted processing time for a batch of product j
TN_j : time until the next arrival of a job of product j
t_{1j} : time of the next arrival of a job of product j

The NACH-setup logic consists of two cases. The first case, a push decision, occurs when a batch machine is idle and a job arrives. At this point, a trade-off similar to the one employed by DBH is made to determine if the machine should begin processing this product now or wait for the next arrival. The second case, a pull decision, occurs when a machine has just finished processing and must choose whether or not to pull jobs and immediately begin processing again. If the decision is to pull jobs, the type of product to process must be determined.

We start by describing the push decision logic. It is similar to DBH with $L = 1$. A batch loading decision is said to occur at epoch t_0. If there are jobs in queue, t_0 corresponds to the time epoch that the machine becomes idle. Otherwise, t_0 corresponds to the arrival epoch of the next job. The potential times the next load begins is specified by the number of future arrivals that will occur before the next load begins.

If $B \leq q$ at time t_0, then a full load is available, i.e., waiting will only result in an increased delay. Therefore, a full load will be dispatched to the machine.

However, if $0 < q < B$, a decision must be made whether to load the machine immediately or to wait for the next arrival. The decision of whether or not to wait is determined by calculating the net decrease in delay, if any, caused by waiting similar to the case of DBH. We obtain for the corresponding net change in the delay for product j:

$$\Delta_{1j} := (T_j + S_j + t_0 - t_{1j}) - (t_{1j} - t_0)q_j. \tag{4.34}$$

Note that the main difference between expression (4.31) and (4.34) is the additional setup time in the first term of expression (4.34). When $\Delta_{1j} > 0$, then the delay is reduced by waiting until t_{1j} to start the batch. We note that in the case of the push logic, we have to consider only the arriving product type in determining whether to make a batch or not. Let this product type be denoted by j. The push decision can be formalized as follows.
Algorithm Push Decision (Push)

1. Increase the inventory of this product by one. Determine the number of idle batch machines. Denote this value by m_{idle}. If $m_{\text{idle}} = 0$, then stop, i.e., no push decision is made at this point of time. If $m_{\text{idle}} > 1$, then go to step 4. If there is a full load of this product, then also go to step 4.
2. Determine the time of the next arrival of this product, i.e., find t_{1j}. Let t_0 be the current time. If $t_{1j} > t_0 + T_j$, then set $t_{1j} := t_0 + T_j$.
3. Calculate Δ_{1j} for the product that enters the queue to determine whether it is worth waiting for the next arrival of this product before making a batch. If $\Delta_{1j} > 0$, then stop, i.e., no push decision is made at this point of time.
4. Start a batch of this product now on an idle batch machine. Decrease the inventory of this product by the size of this batch.

The more complex issue, the influence of one product on another, is embedded into the pull decision logic. Therefore, we continue by describing the pull decision logic. A pull decision is necessary when a specific batch machine becomes idle immediately after completing its previous batch. Again, the benefit to wait can be expressed as Δ_{1j} using expression (4.34). If we have $\Delta_{1j} > 0$ for all j, then the batch machine has to wait. The pull decision can be summarized as follows.
Algorithm Pull Decision (Pull)

1. If no jobs are waiting, then stop, i.e., no pull decision is required.
   Determine the number of idle batch machines, including this batch machine. Denote this number by m_{idle}. Determine the time of the next batch machine completion. This time is denoted by t_C. Set $t_C := \infty$ if no batch machines are busy. Set the selected product indicator j_{prod} to 0.
2. If there is no full load for any product, then go to step 3.
   Determine the product among those that have a full load that will cause the weighted shortest processing time. Set j_{prod} to that product number and go to step 6.

3. Determine the next arrival time t_{1j} for all j. If $m_{\text{idle}} > 1$ or $t_c < t_{1j}$, then calculate the delay due to processing j (see Eq. (4.37)). Set j_{prod} to that product number of the product with the corresponding minimum value and go to step 6.
4. Calculate Δ_{1j} for each j to determine whether it is worth waiting for the next arrival of that product to appear before making a batch.
   If it is worth waiting until the next arrival appears for all products, i.e., if $\Delta_{1j} > 0$ for all products, then stop. In this case, no pull decision has to be made.
   On the other hand, if it is not worth waiting until the next arrival for any of the products, i.e., $\Delta_{1j} \leq 0$, set j_{prod} to the product number of that product that causes no setup time. If no products exist without setup time, set j_{prod} to the product number of the product with the weighted shortest processing time and go to step 6.
5. For each product for which it is worth waiting for the next arrival, i.e., $\Delta_{1j} > 0$, determine the total waiting time incurred by all products by waiting for the next arrival of this product.
   For each product for which it is not worth waiting for the next arrival, determine the total waiting time incurred by all products when a batch of this product is started now.
   Determine the minimum of the above and set j_{prod} to the product number of the corresponding product.
   If the minimum is for a product for which it is worth waiting, then stop. In this case, no pull decision is required.
6. Start a batch of product j_{prod} on the batch machine that just completed. Decrease the inventory of that product by the size of the batch.

If it is determined that all products should start processing now, the algorithm Pull chooses the batch that requires no setup downstream. If no such batch can be formed, the algorithm selects a batch with the weighted shortest processing time. It has been shown that using a WSPT scheme leads to schedules with a minimum mean flow time (cf. Pinedo [240]). The weighted value for each product, W_j, is defined as follows:

$$W_j := (T_j + S_j) \sum_{i=1, i \neq j}^{N} q_i, \quad j = 1, 2, \ldots, N. \tag{4.35}$$

The quantity W_j represents the total delay incurred by all of the other products at the batch machine by starting j immediately. The product with the minimum value is selected in step 2 and step 4.

If it is found in step 5 that some products should wait and some should begin processing, the total delay over all products as a result of waiting, DW_j, or processing, DP_j, is computed as follows:

$$\mathrm{DW}_j := \mathrm{TN}_j \sum_{i=1}^{N} q_i + \sum_{i=1, i\neq j}^{N} \left((T_j + S_j)q_i + (T_j + S_j + \mathrm{TN}_j - \mathrm{TN}_i)^+\right),\ j \in S_1, \quad (4.36)$$

$$\mathrm{DP}_j := (T_j + S_j) \sum_{i=1, i\neq j}^{N} q_i + \sum_{i=1}^{N} (T_j + S_j - \mathrm{TN}_i)^+,\ j \in S_2. \quad (4.37)$$

Recall that we set $x^+ := \max(x, 0)$ for abbreviation. S_1 is the set of products for which it is determined to wait, and S_2 is the set of products for which it is determined to begin processing. The minimum of these values is selected, and the appropriate action is taken. The first term of the right side of Eq. (4.36) determines the additional waiting time incurred by those jobs already in the queue until the next arrival of j. The second term calculates the additional waiting time for all other products if j begins processing at the time of its next arrival. The $(T_j + S_j)q_i$ portion is the delay for those products already in queue, and the $(T_j + S_j + \mathrm{TN}_j - \mathrm{TN}_i)^+$ portion is the delay (if any) for the next arrival of the other products. The first term of the right side of Eq. (4.37) represents the additional waiting time gained by those jobs already in the queue, while the second term corresponds to the additional waiting time gained by the next arrival of each product. The resultant heuristic is called NACH-setup for abbreviation. This heuristic is very similar to the multi-product NACH procedure proposed by Fowler et al. [85]; however, NACH does not take setups into account.

To assess the performance of the NACH-setup heuristic, simulation experiments are performed using the simulation engine Factory Explorer. The first model is a three-machine system with multiple products. This system is used to compare MBSG, MBSF, NACH, and NACH-setup in a simple controllable environment. The three-machine system is comprised of a serial machine, a batch machine, and another serial machine with setups. The first machine (machine 1) is a dummy machine with infinite capacity used only to get products into the system. The batch machine (machine 2) and the next serial machine (machine 3) have limited capacity, only one machine each, and parameters typical of machines used in semiconductor manufacturing. The different products have identical process flows, which are shown in Fig. 4.2.

We continue by describing the design of experiments used. Five factors are included in the design of experiments for the simple system. The factors are listed below:

- Number of products
- Product mix
- Traffic intensity at the batch step (machine 2)
- Traffic intensity at the setup step (machine 3)
- Dispatching policy at the batch machine

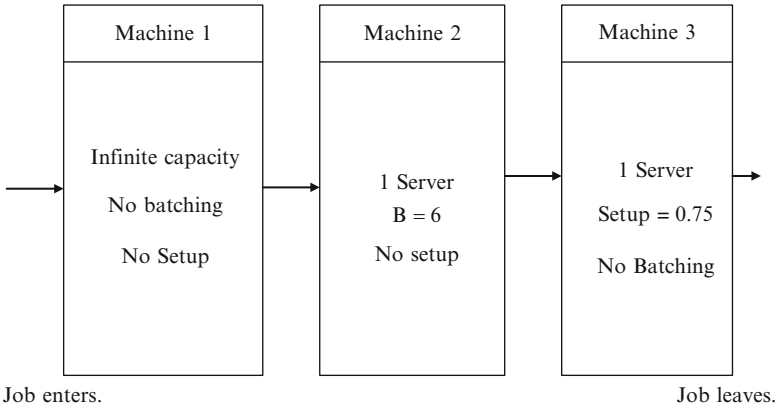

Figure 4.2: Three-machine system with two or four products

The input rate to the system is varied depending upon product mix and batch step traffic intensity, and exponential inter-arrival times are assumed. The processing time at machine 3, the machine with setup, is also varied depending upon product mix, input rate, and traffic intensity at the setup machine. The remainder of the system parameters for this experiment were as follows:

- Number of wafers per job: 48 wafers
- Processing time at machine 1: 2.5 h (deterministic)
- Processing time at machine 2: 2.5 h (deterministic)
- Capacity of machine 2: $B = 6$ jobs
- Setup time between products: $s = 0.75$ h

Altering the input rate to the system controls the traffic intensity at the batch step. Changing the processing time required for each product at the setup step controls the traffic intensity at the setup step. Notice that the experimental range of these values is small, i.e., 0.720–0.734. This is done to examine a range of high utilization at the setup step while avoiding an unstable system. The full experimental design is shown in Table 4.3.

Each data point is replicated three times, each with a run length of 8,640 h and statistics cleared after 1,720 h to take into account for initialization bias. The traffic intensity at the setup step was calculated disregarding any potential setup. Figure 4.3 shows the variation in CT over the six traffic intensity levels at the setup step for the two-product, equal-mix case.

We observe that NACH-setup outperforms all the other policies, except at setup step traffic intensity level 1. At this point, the setup step is not limiting the flow, thus dispatching based on downstream setup will likely not be beneficial. This is supported by the fact that, as the traffic intensity level at the setup step increases, the performance of NACH-setup compared to the other policies is superior.

Table 4.3: Design of experiments

Factor	Level	Count
Number of products	2, 4	2
Product mix	Equal (50%,50%), (25%,25%,25%,25%) Dominant (70%,30%), (40%,40%,10%,10%)	2
Traffic intensity at batch step	0.5, 0.8	2
Traffic intensity at setup step	0.720, 0.723, 0.726 0.729, 0.731, 0.734	6
Dispatching policy	MBSG, MBSF, NACH, NACH-setup	4
	Total factor combinations	192
	Number of replications	3
	Number of simulation runs	576

In Table 4.4, the CT values for the two-product, dominant mix case are shown. With one product dominant over the other, the behavior of the manufacturing system is similar to a system with one product. As expected, because of the dominant-product mix, NACH performs best until traffic intensity level 4. Once this situation is reached, the gain in CT because of the reduced setup downstream becomes critical as the setup step begins to limit the flow of jobs through the system. In the four-product, equal-mix case,

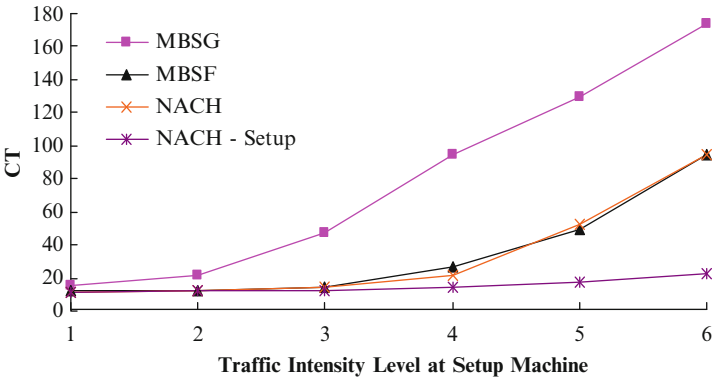

Figure 4.3: CT at the setup step, two products, equal mix

NACH-setup outperforms all the other policies, and, as expected because of the additional setup due to the larger number of products, the difference between them is more substantial. MSBF outperforms NACH because the additional setup time favors larger batches. As in the two-product situation, the differences in CT between the policies are reduced in the four-product, dominant-product mix scenario, and it does not become apparent until traffic

intensity level 4. The detailed results for the four-product case can be found in [287]. Additional experiments using the MIMAC 1 model are also described in [287]. In case of a high input rate, no statistically significant difference between the four policies can be found. There is no significant difference between MBSG, NACH, and NACH-setup in the low- and medium-input rate cases. However, all three policies are superior to MBSF. In addition, NACH-setup performs well for all three input rates, while the relative performance of the others changes with respect to the different input levels.

Table 4.4: Simulation results for two products, dominant mix

Traffic intensity level	MBSG	MBSF	NACH	NACH-setup
1 - 0.720	14.425	12.025	11.050	13.125
2 - 0.723	18.100	12.400	11.500	13.350
3 - 0.726	28.875	12.975	12.375	13.550
4 - 0.729	63.900	34.675	13.700	14.475
5 - 0.731	87.700	53.025	16.375	15.750
6 - 0.734	133.300	75.250	28.225	17.700
Average	57.717	33.558	15.538	14.658

4.6.3 Additional Look-Ahead Research

Weng and Leachman [320] address the same problem as Glassey and Weng [101] and Fowler et al. [84]. However, their minimum cost rate (MCR) heuristic has some noticeable differences from DBH and NACH. MCR seeks to minimize the holding cost per unit time, which is like minimizing the weighted (by cost) number of jobs in queue. Robinson et al. [260] present a heuristic that is essentially a combination of NACH and MCR. The cost rate function used in MCR is incorporated into the rolling horizon scheme used in NACH. We refer to Robinson et al. [261] for a review and a comparison of various real-time control strategies for batch machines in wafer fabs until 2000.

Instead of calculating a threshold number of jobs in queue, Cigolini et al. [52] determine dynamically the length of the time window in a more recent paper. The resulting look-ahead heuristic is called wait no longer than time (WNLTT).

Ham and Fowler [114] propose an extension of NACH. The heuristic, called NACH+, is based on the idea that the incoming inventory into the batch operations is controlled such that unnecessary waiting time does not happen.

4.6.4 BATC-Type Rules

We continue with an extension of the algorithm BA described in Sect. 4.5. Again, we assume that a batch-processing machine is available for processing, and we have to determine the next batch to be processed. In contrast to the algorithm BA, the present extension takes ready times of the jobs into account. This kind of ready time information is typically provided by the MES.

For this purpose, we consider a time window $(t, t+\Delta t)$, where t is the current time. We define the set

$$J(i,t,\Delta t) := \left\{ij | r_{ij} \leq t+\Delta t\right\}, \tag{4.38}$$

where we represent job j of family i by ij. We sort the elements of $J(i,t,\Delta t)$ with respect to the index

$$I_{ij}(t) := \frac{w_{ij}}{p_i} \exp\left\{-\frac{(d_{ij}-p_i-t+(r_{ij}-t)^+)^+}{\kappa \bar{p}}\right\} \tag{4.39}$$

in nonincreasing order, where p_i is the processing time of the jobs of family i and r_{ij} is the ready time of job j of family i. Note that index (4.39) is similar to index (4.24) when no setup times occur. In the next step, we select the first thresh jobs from this list and form all the possible batches. We assess each of these potential batches by using the batch index

$$I_{bi}(t) := \frac{w_{bi}}{p_i} \exp\left\{-\frac{(d_{bi}-p_i-t+(r_{bi}-t)^+)^+}{\kappa \bar{p}}\right\} \frac{n_{bi}}{B}, \tag{4.40}$$

where w_{bi} is the average weight of the n_{bi} jobs that form the batch bi of family i, r_{bi} is the maximum ready time among the jobs that form the batch, and finally d_{bi} is the minimum due date. We summarize the algorithm as follows.
Algorithm Dynamic Batching Dispatching Heuristic (DBDH)

1. Determine the sets $J(i,t,\Delta t)$ for family $i = 1,\ldots,f$. The quantity t is the time where a batching machine is available for processing.
2. Sort all the jobs within each family $i = 1,\ldots,f$ in nonincreasing order with respect to $I_{ij}(t)$ given by expression (4.39).
3. The length of the list that corresponds to family i is denoted by $l(i)$. Consider the first $\min\{\text{thresh}, l(i)\}$ jobs to form potential batches. Assess each of these batches using index $I_{bi}(t)$.
4. Select the batch with the largest $I_{bi}(t)$ value among the families $i = 1,\ldots,f$ to be processed next.

Note that Δt and thresh are parameters of DBDH that have to be selected carefully. Large values of thresh might lead to a huge computational burden, whereas large values for Δt might decrease the quality of the future job arrival information represented by r_{ij}.

We continue with the presentation of the results of some computational experiments. The MIMAC 1 model [83] is used in a slightly modified version. This simulation model consists of two different process flows with more than 200 process steps and over 80 different machine groups.

There are 16 batch machine groups among the machine groups of the MIMAC 1 model. Machine group OXIDE_1 is the bottleneck of the wafer fab. Table 4.5 provides information on this particular batch machine group. In Table 4.5, we denote by $B_{\min}$ the minimum batch size in jobs and by $B_{\max}$ the maximum batch size in jobs. The processing time of the different job families is between 135 min and 1,410 min. The utilization is determined by simulation experiments with the FIFO dispatching rule.

Table 4.5: Bottleneck batching machine group information

Machine group	Number of machines	$B_{\min}$	$B_{\max}$	Utilization (%)
OXIDE_1	3	2	6	84.19

We use a slack-based dispatching rule for the non-batching machines (cf. Sect. 4.2.1). The rule selects the job with the smallest slack for the process step. For the calculation of the slack of the jobs waiting in front of a certain machine group, we simply multiply the processing time by a flow factor. For that purpose, we calculate the difference between the due date of the job and the current time as used in the LS index (4.13). Based on this information, we assign a flow factor to each job. This scheme allows us to determine local due dates for each single process step, i.e., future job arrival information is available at the batch machines. We repeat the calculation of the flow factors every 15 min.

In our experiments, we consider a moderate workload in the system. Machine TTF and TTR (see Sect. 3.2.8) are exponentially distributed. The model is initialized using a WIP distribution of the wafer fab. The length of a single simulation run is 100 days in our experiments. We take five independent replications of each simulation run in order to obtain statistically meaningful results.

We continue by presenting the design of experiments used. The main performance measures are TWT, CT, and TP. Therefore, we set due dates according to the following expression:

$$d_j := r_j + \mathrm{FF} \sum_{k=1}^{n_j} p_{jk}, \tag{4.41}$$

where FF is the flow factor. Furthermore, we define weights of the jobs according to the following two discrete distributions:

$$D_1 := \begin{cases} w_j = 1, \ p_1 = 0.5 \\ w_j = 5, \ p_2 = 0.35 \\ w_j = 10, \ p_3 = 0.15 \end{cases} \tag{4.42}$$

and

$$D_2 := \begin{cases} w_j = 1, \ p_1 = 0.5 \\ w_j = 2, \ p_2 = 0.45. \\ w_j = 10, \ p_3 = 0.05 \end{cases} \tag{4.43}$$

D_1 mimics the situation where a large number of jobs have a large weight, whereas a large number of jobs have a medium weight in D_2 and only a very small portion of the jobs have a large weight in this situation. We summarize the design of experiments in Table 4.6. We denote by $\bar{p}$ the average processing time of the jobs on the batch machines.

Table 4.6: Design of experiments

Factor	Level		Count
FF	1	2 for all the jobs,	2
	2	2 for 50 % of the jobs 1.5 for 50 % of the jobs	
w_j	1	$\sim D_1$	2
	2	$\sim D_2$	
Δt	1	$0.25\bar{p}$	2
	2	$0.5\bar{p}$	
Overall number of experiments			8

The look-ahead parameter κ in DBDH is selected from the grid $\{0.1, 0.2, \ldots, 6.5\}$. The κ value that leads to the smallest TWT value is finally used whenever DBDH makes a decision. Within the experiments, $\mathsf{thresh} = 15$ is chosen.

The corresponding results of the DBDH-based batching strategy are shown in Table 4.7. We use a batching scheme based on the FIFO dispatching rule as a reference. Note that when the FIFO dispatching rule is used for batching, Δt does not have any impact on the decision-making. Therefore, we have to conduct only four simulation experiments in this situation. Consequently, we have to perform a total of 12 different simulation experiments.

We use the notation (level of factor 1—level of factor 2—level of factor 3) in order to indicate the considered factor combinations for DBDH. All the results in Table 4.7 are the ratios of the performance measure values of DBDH and FIFO for the same values of the levels of the first, the second, and finally the third factor. Low values are good for TWT and CT, while we are interested in high values for TP.

Table 4.7: Computational results for DBDH

Factor combination	TWT	CT	TP
1-1-1	0.1031	0.9685	1.0099
1-1-2	0.1191	0.9725	1.0089
1-2-1	0.0918	0.9677	1.0135
1-2-2	0.1475	0.9732	1.0056
2-1-1	0.2913	0.9631	1.0081
2-1-2	0.3035	0.9703	1.0082
2-2-1	0.4230	0.9588	1.0089
2-2-2	0.4363	0.9643	1.0085

From the results shown in Table 4.7, we can see that the algorithm DBDH outperforms the FIFO rule at all factor settings for all performance measures. The TWT values are sensitive to the choice of Δt. In our experimental design, smaller values for Δt lead to slightly better results. Choosing a larger Δt value causes fuller utilized batches and a larger queue size. The machines have to wait longer for jobs that arrive during the given time window. Therefore, this leads to fuller batches. Hence, a careful selection of the Δt values is important for the performance of DBDH.

More computational results can be found in Mönch and Habenicht [194]. It is also shown, by comparison with the algorithm BA, that taking future job arrival information into account can lead to TWT reductions.

4.7 More Sophisticated Approaches

In this section, we discuss rule-based systems, the selection of parameters of dispatching rules using iterative simulation, the construction of appropriate blended dispatching rules, and finally the automated discovery of dispatching rules.

4.7.1 Rule-Based Systems

A rule-based system determines a priority value for each job or batch based on hierarchically structured rule-based criteria systems. In a certain sense, a rule-based dispatch system is a combination of composite, truncation, conditioning, and finally multilevel dispatching rules. Powerful rule-based systems are in use in wafer fabs (see Appleton-Day and Shao [9]).

The main ingredient of a rule-based system is a composite dispatching rule, given by the following priority index for each job j:

$$I_j := \sum_{k=1}^{C} w_k c_k, \tag{4.44}$$

where C is the number of first-order criteria, c_k is the value of the kth first-order criterion, and $w_k \geq 0$ with $\sum_{k=1}^{C} w_k = 1$ are weights to balance the importance of the different first-order criteria. Each c_k can be further refined by appropriate subcriteria, i.e., we consider second-order criteria $c_{kl}, l = 1, \ldots, l_k$, where we denote by l_k the number of second-order criteria for the first-order criterion c_k. Higher-order criteria can be taken into account by following this approach in a recursive manner. The resulting criteria hierarchy is a tree. A set of IF-THEN rules is used to evaluate each leaf of this tree based on certain BS- and BP-related data with a certain attribute value. Typical attribute values are LOW, MEDIUM, LARGE, and HUGE. An attribute value is assigned to each $(l+1)$-order criterion based on the attribute value combination of its l-order subcriteria. By proceeding in a recursive manner, a positive real number can be assigned to each first-order criterion that is necessary to compute the left-hand side I_j in expression (4.44).

Following Thiel et al. [296, 297], we introduce the following example for a rule-based system. The rule-based system consists of the following first-order criteria:

1. On-time delivery performance-related criterion, called on-time urgency criterion
2. Setup-related criterion
3. Load-related criterion

Note that the second criterion deals with setup avoidance issues, while the third one is related to batch formation issues. It is obvious that the second and third criterion are TP-related and therefore in potential conflict with the first criterion.

We show the second- and third-order subcriteria for the on-time urgency in Fig. 4.4. The first second-order criterion measures the slack related to the current process step of the job while the second second-order criterion is concerned with the importance of the due date and is comprised of three third-order criteria. The first third-order criterion considers the progress of processing a job measured in the number of completed process steps. This subcriterion is motivated by the fact that a potential due date violation is less important when the number of already completed process steps is small. The second third-order criterion takes into account whether the job is a regular or a hot job, whereas the third second-order criterion measures the importance of meeting the due date of the job with respect to the type of the customer associated with this job. Specifically, the progress of a job is measured by

$$\text{prog}(j) := l_j / n_j, \tag{4.45}$$

where l_j is the current process step of j, and n_j denotes the number of all process steps of job j. We show the IF-THEN rules with respect to the progress of the job-related subcriterion:

IF $\text{prog}(j) < 1/3$ THEN $c_{121} =$ "LOW"

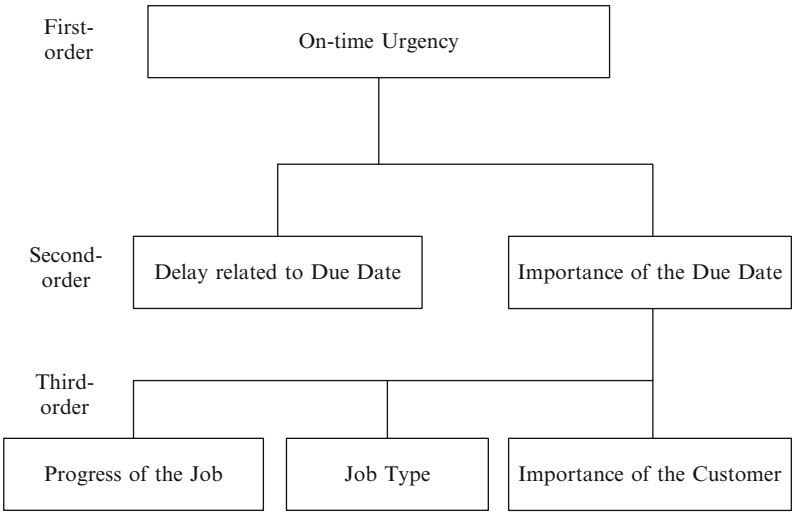

Figure 4.4: Criteria hierarchy related to On-time urgency

IF $1/3 \leq \text{prog}(j) \leq 2/3$ THEN c_{121} = "MEDIUM"
IF $2/3 < \text{prog}(j)$ THEN c_{121} = "LARGE"

It is obvious that the values 1/3 and 2/3 are prescribed values that can be modified to model user preferences. Because these values are arbitrary, extensive simulation-based assessment of rule-based dispatching systems is necessary.

4.7.2 Determining Parameters of Dispatching Rules Based on Iterative Simulation

We continue by studying global dispatching rules that take waiting times into account. The waiting times for process steps that have to be performed in the future are unknown. The waiting times depend, for example, on the product mix, on the load of the wafer fab, and on the control strategy used.

Global variants of the ATC dispatching rule are discussed by Vepsalainen and Morton [312]. A solution is proposed that is based on iterative simulation (cf. the discussion in Sect. 3.2.8). The method is called lead time iteration. Based on a crude initial waiting time estimate, successive adjustments of the waiting time are performed by using the measured waiting time from the current simulation run. This method is also used by Ovacik and Uzsoy [223] in order to determine appropriate internal due dates for an ODD-type dispatching rule in the test area of a back-end facility. A lead time iteration scheme is used to estimate waiting time used in the FSMCT dispatching rule in [169] (cf. the discussion in Sect. 4.2.1). We consider a global ATCS rule as

proposed by Vepsalainen and Morton [312]. The index has to be calculated as follows:

$$I_{ji}(t,lk) := \frac{w_j}{p_{ji}} \exp\left\{ -\frac{\left(d_j - p_{ji} - t - \sum_{g=i+1}^{n_j}(wt_{jg} + p_{jg})\right)^+}{\kappa_1 \bar{p}} - \frac{s_{kl,ji}}{\kappa_2 \bar{s}} \right\}, \quad (4.46)$$

where we assume that i is the current process step of job j. The average of the sum of the processing times of the remaining process steps for each job is denoted by $\bar{p}$. The waiting time associated with process step k of job j is denoted by wt_{jk}. The quantity $s_{kl,ji}$ is the setup time that is necessary when process step l of job k is processed before ji. Again, κ_1 and κ_2 are scaling parameters. The resulting dispatching rule is called global ATCS (GATCS). Appropriate values for wt_{jk} are unknown in the beginning because they are a result of the dispatching strategy used. Therefore, we use iterative simulation to determine them. The resulting procedure can be formulated as follows.

Algorithm Lead Time Iteration Procedure (LTIP)

1. Set $l = 1$. Start by an initial waiting time setting using

$$wt_{jk}^{(l)} := (\mathrm{FF} - 1)p_{jk},\ k = i+1,\ldots,n_j, \quad (4.47)$$

   where we denote by $\mathrm{FF} \geq 1$ the flow factor. Initial values for FF can be obtained from a simulation run using the FIFO dispatching rule.
2. Dispatch the wafer fab using the GATCS dispatching rule and the waiting time estimates for the current iteration.
3. Calculate the actual waiting time $q_{jk}^{(l)}$ of each process step jk from simulation run l. In this situation, the waiting time is defined as the time between the completion of process step $j,k-1$ and the start time of process step jk, i.e., the transportation time is included.
4. Update the waiting time estimates as follows:

$$wt_{jk}^{(l+1)} := (1-\alpha)wt_{jk}^{(l)} + \alpha q_{jk}^{(l)},\ k = i+1,\ldots,n_j, \quad (4.48)$$

   where $0 \leq \alpha \leq 1$ denotes a fixed smoothing factor.
5. Terminate the procedure if the stopping condition $\max_{jk} |wt_{jk}^{(l+1)} - w_{jk}^{(l)}| < \varepsilon$ is valid; otherwise, update $l := l+1$ and go to step 2. The quantity ε is a small prescribed value.

Usually, four to eight iterations are enough to fulfill the LTIP stopping condition for reasonable values of ε. The update scheme for the waiting times in step 4 is an exponential smoothing-type approach. It takes the current measured waiting time from the simulation run and the estimated waiting time from the previous iteration into account. Typical α values are 0.7 and 0.9 (see Mönch and Zimmermann [197]). Note that it is also possible to use a more sophisticated update scheme based on double exponential smoothing.

The architecture described in Sect. 3.3.2 is used to implement LTIP. The object-oriented database is used to store the waiting time for each single process step in each iteration. It makes the waiting times persistent because they are required in future iterations.

LTIP allows for large TWT reductions compared to a FIFO dispatching strategy. At the same time, CT decreases and TP increases. Detailed computational results for the MIMAC 1 model can be found in [197]. LTIP is a simple but powerful technique to improve the performance of dispatching rules.

4.7.3 Construction of Blended Dispatching Rules

We consider a blended dispatching rule for the case of l different performance measures. Each performance measure a of interest is represented by a priority index $I^a_{jis}(t) \in [0,1]$ for processing job j of product i at stage s at time t. We obtain for the weighted priority for job j of product i on stage s at time t:

$$P^t_{jis} := \sum_{a=1}^{l} w_a I^a_{jis}(t), \tag{4.49}$$

where $w_a \geq 0$ is the weight of performance measure a and $\sum_{a=1}^{l} w_a = 1$ is valid. Furthermore, it can be achieved by an appropriate transformation of the priority indices of the jobs that the sum of all priority index values for all the jobs of all products queued at machine group m sum up to one as shown in the following expression:

$$\sum_{i=1}^{p} \sum_{s \in J_i(m)} \sum_{j=1}^{n_{si}} I^a_{jis}(t) = 1, \tag{4.50}$$

where p is the number of different products, n_{si} is the number of jobs of product i at stage s, and finally $J_i(m)$ is the set of all stages in the process flow of product i that require machine group m.

We have to determine appropriate values for the weights. Therefore, we express the values of each performance measure as a mapping of the weights. To do this, we look for a mapping:

$$P_a : (w_1, \ldots, w_l) \rightarrow \mathbb{R} \tag{4.51}$$

for each performance measure a. $P_a(w_1, \ldots, w_l)$ is called the response for the weight setting $w_1, \ldots, w_l$. The same combined dispatching weights and rules are used at all machine groups. However, due to the different processing natures of non-batching and batching machines, two varying yet similar approaches are used. For a non-batching machine, the combined criterion is calculated for each job queued in front of machine group m. In the case of a batching machine, the only difference is that jobs are grouped together into

batches based on similar batching requirements, i.e., the same incompatible job family. The combined criterion for the different jobs that form the batch is aggregated into a single index value by adding the weighted criterion for each job in the batch similar to algorithm BA in Sect. 4.5.

Next, we describe how the response for each objective is determined by designed experiments using discrete-event simulation. Model parameters are estimated most effectively when proper experimental designs are used to collect the data (see Montgomery [208]). There are many experimental designs to choose from including factorial, central composite, and D-optimal. However, there are several assumptions that need to be made about the design and the data for the experimental methods to be effective. One critical assumption is the independence of the experimental variables. Least square methods cause erroneous results in the model parameter estimates if this assumption is violated.

In the situation discussed in this section, the experimental variables, i.e., the different w_a, are not independent. The weights identify the proportional contribution of each dispatching rule index in the overall blended priority index. The response variables are a function of the proportions of the different weights. The actual value of a weight is not important, but, rather, it is the relative size when compared with another weight that is important. For example, in case of four criteria, the weights 0.05, 0.05, 0.15, and 0.25 are possible, or they can be 200, 200, 600, and 1,000. The same results are obtained for each of the two weight sets because the ratios of the weights when compared to each other are the same for each set. Both sets of weights can be normalized to 0.1, 0.1, 0.3, and 0.5, i.e., their weight sum is one.

Due to the lack of independence of the weights, standard experimental design strategies have to be abandoned and a mixture design chosen that will accommodate this lack of independence. Mixture experiments address the issue where the components of the experiment have to add to 100 %. There are several mixture designs to choose from depending on the degree of the polynomial that the experimenter is anticipating to best fit the process. For mixture experiments, the experimental design region is a simplex that is a regularly sided figure with q vertices in $q-1$ dimensions (see Montgomery [208]). Let us consider the simplex centroid design with q factors in more detail. This design consists of 2^q-1 design points. This is the number of vectors with q components where k components have the value $1/k$ for $1 \leq k \leq q$ and the remaining components are 0. An example with three different criteria is given in Table 4.8. Note that in Table 4.8 in addition to the 2^3-1 regular design points, three augmented points are added to give more degrees of freedom for error lack of fit and model significance analysis.

Based on the mixture design, a response surface is constructed for each single criterion as described in Sect. 3.2.9. Note that each design point corresponds to a blended dispatching rule with a certain weight setting. Simulation experiments with simulation models of wafer fabs have to be carried out to find the response values. Different analyses of variance have to be

Table 4.8: Simplex centroid design for a mixture experiment

Run number	w_1	w_2	w_3
1	1	0	0
2	0	1	0
3	0	0	1
4	1/2	1/2	0
5	0	1/2	1/2
6	1/2	0	1/2
7	1/3	1/3	1/3
8	2/3	1/6	1/6
9	1/6	2/3	1/6
10	1/6	1/6	2/3

performed for the different responses to determine statistically significant factors. An individual optimum can be determined for each single P_a. However, a global optimum is desired. Therefore, a multiple response optimization using the desirability function approach (cf. Sect. 3.3.1) is performed. The individual meta-models are transferred to a desirability function with values between zero and one, where one is the most desirable. The corresponding desirability function for criterion a is denoted by d_a. Finally, the desirability functions are transformed into a combined objective function using the geometric mean of the individual desirabilities as described in Sect. 3.3.1. Optimization of the combined objective function can be accomplished using different pattern search algorithms such as the algorithm of Hooke and Jeeves [118].

In Dabbas et al. [56, 57, 58], the above approach is used to determine a combined criterion for four objectives that are similar to AT, Var(L), ACT, and finally Var(CT), defined in Sect. 3.3.1. The dispatching rules used are the CR dispatching rule, the throughput (TP) dispatching rule, the line balance (LB) rule, and finally the FC rule.

The CR dispatching rule works, as discussed in Sect. 4.3.1. The TP rule works as follows. The SPT rule is applied to non-batching machines, while for batching machines, the loads with the highest number of wafers per hour get the highest priority because in this situation, the batch fullness can be increased. A more global dispatching rule is the LB rule. Using LB, products at stages with higher deviations from their WIP goal get higher priority. WIP goals determine the average WIP required at each stage of a process flow such that output requirements are met. WIP goals tie the required TP rate of product i to the CT at stage j using Little's law (cf. Sect. 3.2.7):

$$\mathrm{WIP}(L_{ij}) = \lambda_{ij}\mathrm{CT}_{ij}, \tag{4.52}$$

where $\mathrm{WIP}(L_{ij})$ is the WIP goal for product i at stage j, λ_{ij} is the corresponding TP rate, and CT_{ij} is the CT value, i.e., the sum of waiting time and processing time. The quantity λ_{ij} can be calculated by dividing the required

daily output for i by the number of process steps to determine the daily output of i at stage j because then the line is balanced. The goal CT values $\mathbf{CT}_{ij}$ can be derived from simulation studies.

Finally, the FC dispatching rule prioritizes jobs with the objective to balance the workload on the different machines. This rule is described in more detail in Sect. 4.2.1.

The reason for selecting the above dispatching rules is that they each target a different performance measure of interest. CR identifies the quality of dispatching from a point of view of on-time delivery, i.e., AT. The TP rule improves the quality of dispatching from a CT point of view. Its objective is to maximize TP at machines regardless of due date priorities. The LB rule has the objective of minimizing the WIP variation vs. a pre-set goal in an effort to linearize output, while FC takes the workload balancing perspective.

The blended dispatching approach was compared to single dispatching criteria like CR, SPT, or FLNQ in [56, 57] using full wafer fab simulation models. The simulation results indicate that CT, Var(L), and on-time delivery-related performance measures are improved to a large extent. The blended approach shows a behavior similar to the CR dispatching rule with respect to Var(CT), but it outperforms the two remaining dispatching rules. It is also demonstrated by the simulation experiments that the resultant WIP profile is more stable and has a lower average value when using the blended dispatching approach. Lower WIP translates to lower ACT values, whereas less variability in the profile translates to lower Var(CT) values and consequently better overall performance measure values.

4.7.4 Automated Discovery of Dispatching Rules

So far, we assume that we manually select dispatching rules out of a given set of rules. Then simulation experiments have to be carried out to assess the performance of the rules, and finally, an appropriate rule is selected. This approach is in a certain sense rigid. That is why we allow for the selection of appropriate weights to construct blended dispatching rules in Sect. 4.7.3. This approach is less rigid; however, only the weights can be changed and not the structure of the rules.

In this section, we describe work that is related to an automated discovery of dispatching rules. A dispatching rule is represented by an index that assesses all jobs awaiting processing at a given machine at time t when the resource is available (cf. Sect. 4.1 for details). The index takes several attribute values into account. The priority index might be considered as a logical expression. Selecting a dispatching rule means specifying the priority index.

We discuss an approach to construct new dispatching rules based on a given set of primitives. These primitives belong to two subsets (see Geiger et al. [96]):

- A set of relational and conditional functions denoted by F.
- A set of terminals T that is problem-specific and consists of a set of variables and numerical constants. Terminals cannot be broken down into smaller units.

The set of relational and conditional functions is given by unary and binary operators and functions. Examples for unary operators are the four basic arithmetic operators $+, -, /, *$, whereas EXP, ABS, MAX, MIN are examples of functions. Furthermore, the conditional function IF3 is often important. It is the ternary version of the IF-THEN-ELSE expression used in programming languages, i.e., if $a \geq 0$ then b else c, where a, b, and c are expressions. The precedence relationship of the functions is given. It is preserved and cannot be redefined. More examples of functions F to construct dispatching rules can be found in [96, 295].

We continue with examples for terminals. The following terminals are used, and most of them refer to a given job j:

- PT: The processing time p_j is modeled using this terminal.
- DD: This terminal is used to refer to d_j.
- W: The weight w_j is expressed by this terminal.
- CurT: This is the current time t where the dispatching decision is made.
- Con: A constant value, i.e., a number $z \in \mathbb{R}$, is denoted by this terminal.
- AvgPT: This terminal denotes the average processing time of all jobs waiting for processing. It represents the quantity $\bar{p}$.

It becomes clear that the terminals model the attribute values within the priority indices. More examples for terminals used to construct dispatching rules can be found in [95, 96, 237].

The logical expression of the priority index is first transformed into an intermediate representation using prefix notation. In prefix notation, the function is written before the arguments it operates on (see Preiss [248]). Compared to the logical expression, the prefix notation has the advantage that an expression tree can be derived from it automatically by parsing the expression in prefix notation from left to right. Terminals are the leaves of an expression tree, i.e., variables or constants in the logical expression for a certain priority index. Each single function in a logical expression is represented by a node of the expression tree. When a symbol $s \in F$ is detected during the parsing process, then a node is created in the expression tree. When a terminal $\tau \in T$ is found, then a leaf is added to the tree. When a subtree cannot have more leaves, then a new subtree is started when the next symbol τ is detected.

On the other hand, each expression tree can be transformed automatically into an expression in prefix notation by traversing the tree recursively using the following procedure.

Algorithm Traverse Expression Tree

1. When a visited node is a terminal, then it is written at the end of the partial expression in prefix notation.

2. When a nonterminal symbol is found, then a left parenthesis is written. The symbol is written into the expression after the left parenthesis.
3. The left subtree is traversed using step 1 and step 2.
4. The right subtree is traversed, if any, using step 1 and step 2, and finally, a right parenthesis is written.

Let us consider the ATC dispatching rule given by index (4.21) to illustrate these rather abstract concepts. In this example, the index can be represented as:

$$(*(/W\ \mathrm{PT})(\mathrm{EXP}(-(/(\mathrm{MAX}(-(\mathrm{DD}\ (-\mathrm{PT}\ \mathrm{CurT}))0)(*\mathrm{Const}\ \mathrm{AvgPT}))) \quad (4.53)$$

using the notation for functions and terminals introduced. The resulting tree is shown in Fig. 4.5.

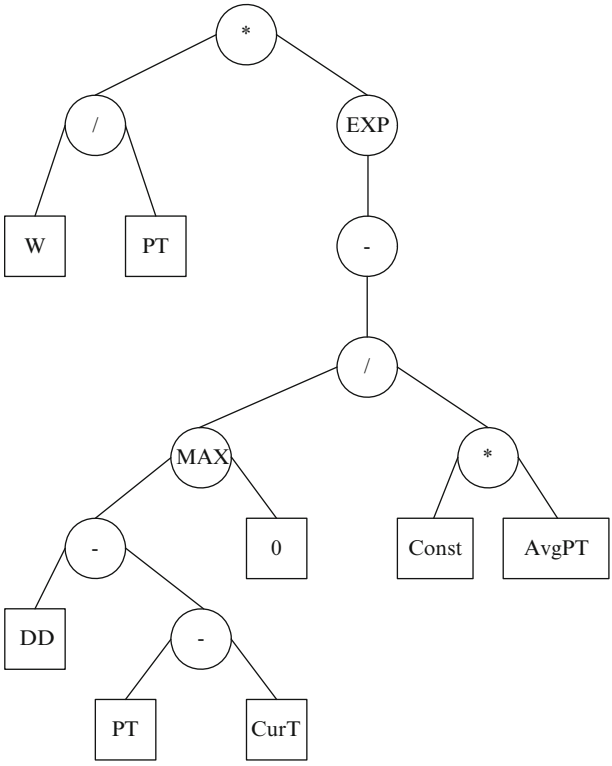

Figure 4.5: Tree representation of the ATC dispatching rule

The basic learning system model described in Sect. 3.2.10 is used to discover new dispatching rules. The corresponding learning element is realized using genetic programming (GP). The performance element is the constructed dispatching rule. Feedback is available from discrete-event simulation. GP is

a special kind of a GA. It uses tree structures of variable lengths to represent solution candidates and can be used to automatically discover logical expressions or even computer programs.

GP starts from a candidate set of dispatching rules called a population (cf. Sect. 3.2.6). These rules are either generated randomly or based on heuristics. The quality of each dispatching rule is assessed using discrete-event simulation and a set of performance measures of interest. The BS and the BP are represented by an appropriate simulation model. Each dispatching rule from the candidate set is equipped after the performance assessment with a set of values for the performance measures.

The reasoning mechanism consists of a selection component and a component that generates new dispatching rules. The selection component uses the performance measure values for each rule of the candidate set to choose a set of high-performing dispatching rules that form the basis for generating a new candidate set. The entire cycle is repeated until a certain stopping criterion is fulfilled.

New dispatching rules will be generated using crossover and mutation operators as in common GAs. The crossover operator works as follows. Two trees are randomly selected from the current set of candidate rules. A subtree is identified in each parent rule randomly. These subtrees are swapped in the next step between the two parent rules. It is clear that because of the entire subtrees used, only feasible offspring are produced. The mutation operator starts by randomly selecting a subtree from a parent rule, then this subtree is replaced by a randomly generated subtree using the sets F and T.

Experiments with the described discovery approach using simulation models of large-scale wafer fabs are described by Pickardt et al. [237]. The discovered rules clearly outperform ATCS-type dispatching rules. The basic discovery approach is extended to automatically learn batching rules for single machine scheduling by Geiger and Uzsoy [95]. Overall, it seems that discovering dispatching rules automatically is a promising direction of future research.

Chapter 5
Deterministic Scheduling Approaches

In this chapter, we describe deterministic scheduling approaches for equipment (i.e., single machines), work centers, and full wafer fab situations. We start with simulation-based scheduling approaches that are somewhere between dispatching and scheduling. Then, we continue by presenting scheduling approaches for single machines. We especially focus on the case of a single batch machine with incompatible job families and the TWT objective. Furthermore, we study the problem of scheduling jobs on a single cluster tool with $C_{\max}$ objective. Then, we present scheduling approaches for work centers, i.e., for parallel machines. We first study scheduling problems for parallel machines with sequence-dependent setup times and then cover parallel batch machines with incompatible job families and ready times of the jobs. After that, we present work that deals with the treatment of secondary resources. Then, we discuss multiple orders per job (MOJ) scheduling problems. We extend these work center approaches to the full wafer fab situation using the concept of disjunctive graphs. We discuss a modified shifting bottleneck heuristic for wafer fabs, and we also describe a distributed variant of the shifting bottleneck heuristic. Next, we extend the shifting bottleneck heuristic to multicriteria situations. We study the performance of these approaches in a rolling horizon setting using simulation models of wafer fabs.

5.1 Motivation and Definitions

Scheduling is defined as the process of allocation of scarce resources over time (Brucker [34]). The goal of scheduling is to optimize one or more objectives in a decision-making process. As already discussed in Sect. 2.3, scheduling is between order release and dispatching in the PPC hierarchy and therefore an important part of the production control layer.

The two major categories in scheduling are deterministic and stochastic scheduling (cf. Pinedo [240]). Deterministic scheduling is characterized by processing times, setup times, due dates, ready times, and weights that are

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5_5,

known in advance. They are assumed not to be influenced by uncertainty. In contrast, stochastic scheduling problems do not assume the existence of deterministic values for processing times, set-up times, or other quantities that are used within the scheduling model. The deterministic values are replaced by corresponding probability distributions. While stochastic scheduling is academically quite interesting, we focus on deterministic scheduling approaches in this monograph.

Deterministic scheduling problems can be further differentiated into static problems where all jobs to be scheduled are available at time $t = 0$ and dynamic scheduling problems that relax this condition. Jobs are ready at different points in time t in the dynamic situation.

In order to classify deterministic scheduling problems, the $\alpha|\beta|\gamma$ notation scheme from the scheduling literature [107, 240] is used within this monograph. The α field describes the machine environment (e.g., single machine or parallel machines), the β field is used for specification of the process restrictions (e.g., ready times or sequence-dependent setup times), and the γ field denotes the performance measure(s) of interest (e.g., $C_{\max}$ or TWT).

The machine environment covered by the α field is determined by the hierarchy provided by the BS (see Sect. 2.2.1). The building blocks of the machinery of a wafer fab are work centers (cf. Sect. 2.2.2). Single machines are the atomic units of a work center. For a single machine, the α field is given by the symbol 1. Single machine scheduling problems are interesting in their own right because they can form subproblems in decomposition schemes for parallel machine or job shop scheduling problems. Often, the machines of a single work center can be modeled as parallel machines. Identical parallel machines will be denoted by Pm, where m specifies the number of machines in the machine group. Because of the different ages of the machines, we often have to deal with unrelated parallel machines, i.e., the situation where the processing time depends on both the job and the machine. For this, we will use the notation Rm. As described in Sect. 2.2.2, each work area consists of several work centers. Therefore, the processing of jobs in a work area can be organized in a flexible flow shop or a flexible job shop manner, and we use the notation FFm and FJm, respectively. A simple job shop, i.e., one machine of each type at each work center, will be denoted by Jm. A wafer fab consists of several work areas. Therefore, we can treat wafer fabs from a scheduling point of view as a complex flexible job shop. A complex job shop as introduced in Sect. 2.2.3 is a job shop that has additional process restrictions that mimic complexities inherent in wafer fabs.

We continue with a discussion of the β field. Important process restrictions are sequence-dependent setup times (s_{jk}) and batching. A batch is defined as a group of jobs that have to be processed jointly (cf. Brucker et al. [36]). The completion time of a batch is determined as the completion time of the last job in the batch. A batch scheduling problem consists in grouping the jobs on each machine into batches and in scheduling these batches. Two types of scheduling problems related to batching are considered. The first type is called a serial batching problem (s-batch). In this situation, the processing time of

a batch is the sum of the processing times of all jobs that form the batch. We differentiate between batching with job availability and batching with batch availability for s-batching. Each job of the batch can be further processed (at the next work center) after its completion time in the first situation. In the case of batch availability, the jobs of the batch can be further processed only when all jobs have completed the batching operation. The second type is parallel batching, for short p-batching (p-batch). In this case, the processing time of the batch is given by the maximum processing time of jobs contained in the batch (cf. Potts and Kovalyov [247] and Mathirajan and Sivakumar [176] for recent surveys related to batching in general and to batching for semiconductor manufacturing, respectively). On the entire wafer fab level, we have to take the reentrant process flows (recrc) into account when making scheduling decisions. Furthermore, in some cases, machine dedications (M_j), also called machine eligibility restrictions, and auxiliary resources (aux) should be modeled. Time constraints (tc) for the process steps also occur as specific process restrictions.

Cluster tools also lead to specific process restrictions. For cluster tools, there exist two types of scheduling problems:

- The scheduling of the wafer movements inside a cluster tool, which is similar to job shop scheduling with transportation and blocking
- The sequencing of the jobs waiting to be processed in front of a cluster tool, which leads to a machine scheduling problem with sequence-dependent processing times caused by different load port recipe combinations for cluster tools

We call the scheduling in the first situation internal scheduling and the one in the latter situation external scheduling. The sequence-dependent processing times are denoted by lrc.

As already described in Sect. 2.2.3, 300-mm manufacturers often have the need and the incentive to group small orders from different customers into one or more FOUPs to form production jobs. These jobs have to be scheduled on the various types of machine groups in the wafer fab and processed together. This class of integrated job formation and scheduling problems are called MOJ scheduling problems.

We start our discussion of the γ field by describing performance measures for the entire wafer fab. The most important among them are TP, CT, and various on-time delivery performance measures. The following measures can be derived for scheduling problems. Large values for TP are a result of minimizing makespan. Minimizing the total flow time measure and its weighted counterpart leads to small values for CT. Typical on-time delivery measures are $L_{\max}$, NTJ, WNTJ, or TWT (cf. Sect. 3.3.1). Performance measures for work areas or even work centers are derived starting from top-level wafer fab-wide performance measures. We summarize the three discussed dimensions for deterministic scheduling problems found in semiconductor manufacturing in Fig. 5.1.

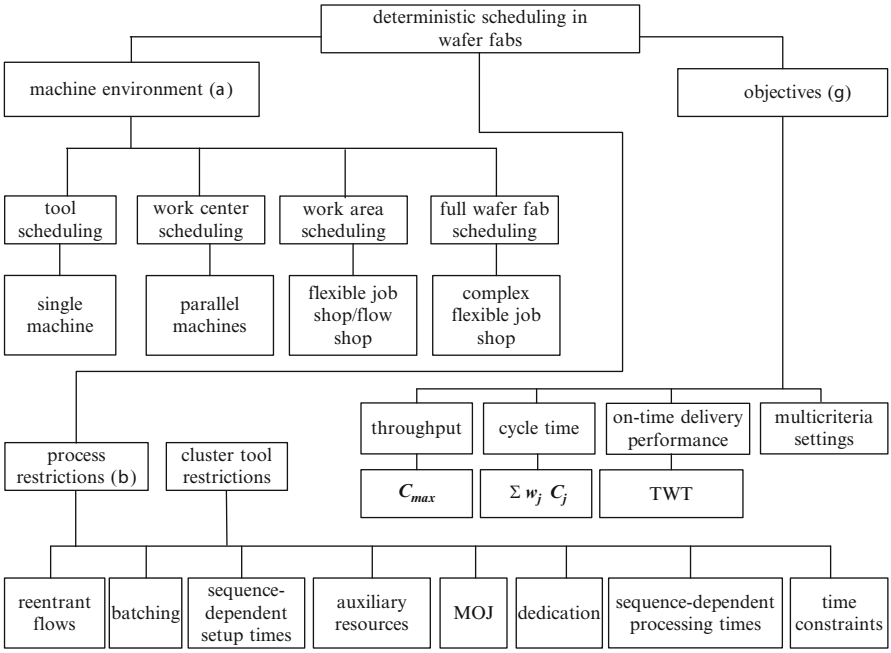

Figure 5.1: Deterministic scheduling in wafer fabs

In today's semiconductor manufacturing, dispatching is typically used in wafer fabs (see Pfund et al. [234]). Scheduling solutions are not very popular. However, with the recent increases in computing efficiency, scheduling methods have become more attractive. Because of the NP-hardness of most of the scheduling problems found in wafer fabs, we typically use heuristics to solve them. Therefore, a large portion of work presented in this chapter is devoted to efficient heuristics. Most of them decompose the overall scheduling problem into smaller, more tractable subproblems.

5.2 Simulation-Based Scheduling

Simulation-based scheduling means that simulation is used as a decision-making tool to determine schedules with a horizon ranging from several hours to a day (see Sect. 3.2.8). Dispatching rules that are already part of the simulation engine are used to allocate jobs to machines. This approach is conceptually similar to list scheduling that is frequently considered in scheduling as a straightforward way to determine schedules (see Pinedo [240]). When a machine becomes available in the simulation, all jobs that are waiting for processing are sorted according to the index of a certain dispatching rule (see Chap. 4). Then the job with the highest or the smallest value for this index is selected to start with processing on this machine. Based on this scheme,

the assignment and the sequencing of jobs observed in the simulation are used to produce a control instruction *mc* in the original CS that is used to influence the BS. Major parts of a simulation-based scheduling system are the following:

1. A simulation engine that contains several dispatching rules for next job selection within an appropriate (up-to-date) simulation model
2. A graphical user interface (GUI) that produces Gantt charts based on the results of a simulation run taken from the event files
3. An interface to the operational systems on the shop floor that develops a dispatch list from the Gantt chart

Simulation-based scheduling relies to a large extent upon the capability to produce simulation models that represent the BP and the BS in a very detailed manner. A snapshot of the BS and BP (WIP and machine status) is taken before the determination of schedules, and a dynamic model is created based on this snapshot. Clearly, automated or at least semi-automated simulation model generation abilities based on data in the OS like the MES are necessary in order to run a simulation-based scheduling system. Usually, all stochastic effects like machine breakdowns are turned off because of the small time horizon. An appropriate model initialization is a nontrivial issue in simulation-based scheduling because estimating the near-time transient performance of the system as it evolves from its current state is challenging (see Davis [60]).

Simulation-based scheduling in its simplest form uses a set of different dispatching rules. Each of the resulting schedules is evaluated by a single performance measure like TP or TWT, and the schedule with the largest or smallest performance measure value, respectively, is selected as a control instruction for the BP. In some situations, even an additional manual inspection of the schedules by the production control staff is possible to check the fulfillment of soft constraints and to include human experience in the schedule selection. Job swaps are possible to manually adjust the schedules. This is a typical Leitstand or MES functionality (see Adelsberger and Kanet [2]).

The selection of a final schedule as a set of production control instructions can also be based on several performance measures. This approach is taken by Sivakumar [284]. This approach is reasonable because it is known that different performance measures can be in conflict and determining a trade-off between them is necessary. We consider the case of l different performance measures. Each performance measure a of interest is represented by a priority $X^t_{ajm} \in [0,1]$ for processing job j on machine m at time t. We obtain for the weighted priority of job j on machine m at time t:

$$P^t_{jm} := \sum_{a=1}^{l} w_a X^t_{ajm}, \tag{5.1}$$

where $w_a \geq 0$ is the weight of performance measure a and $\sum_{a=1}^{l} w_a = 1$ is valid. The weights are selected in [284] based on the importance of the different performance measures, which of course is subjective.

The selection of the w_a can be based on the desirability function approach described in Sect. 4.7.3 to construct blended dispatching rules. In an off-line phase, designed experiments with different weight vectors $(w_1, \ldots, w_l)$ are necessary to find an appropriate meta-model. Based on this meta-model, the desirability function can be optimized and appropriate weights obtained.

There are several simulation-based scheduling systems described in the semiconductor manufacturing literature, most of them applied in the back-end stage. Among them we refer to [213, 246, 284, 285]. The combination of simulation-based scheduling with simulation-based optimization is shown in Werner et al. [321] and Weigert et al. [317]. In these cases, instead of using only a single simulation run, after the generation of an initial schedule based on simulation and dispatching rules, a multiphase algorithm is applied to improve the schedule using metaheuristics and simulation.

5.3 Equipment Scheduling

In this section, we discuss scheduling problems for single machines and parallel machines in a single work center. Parallel machines are the building blocks of wafer fabs. Therefore, we may apply scheduling approaches to schedule jobs in front of a single work center. Scheduling problems for work centers appear rather naturally as a result of decomposition heuristics for flexible job shops. Therefore, it makes sense to deal with scheduling algorithms for single machines and also for work centers.

5.3.1 Scheduling Jobs on a Single Batch Machine

We model diffusion and oxidation operations as batch-processing machines with incompatible job families. The performance measure to be minimized is TWT. The batch problem for a single machine is more complex than the single machine problem $1||\sum w_j T_j$ that is NP-hard by Lawler [151]; therefore, we propose heuristic approaches that lead to good solutions.

The assumptions involved in scheduling a single batch machine with incompatible jobs families to minimize TWT include:

1. Jobs of the same family have the same processing times.
2. All the jobs are available at time $t = 0$.
3. Once a machine is started, it cannot be interrupted, i.e., no preemption is allowed.

The following notation is used throughout this section:

1. Jobs fall into different incompatible families that cannot be processed together. There exist f such families. The family of a batch B_s is given by $F(B_s) \in \{1, \ldots, f\}$.

2. There are n_i jobs of family i to be scheduled. We have $\sum_{i=1}^{f} n_i = n$, i.e., there are n jobs that have to be scheduled.
3. Job j of family i is represented as ij.
4. The priority weight for job j of family i is represented as w_{ij}.
5. The due date of job i of family j is represented as d_{ij}.
6. The processing time of jobs in family i is represented as p_i.
7. The batch machine capacity is B jobs. The number of jobs within a batch B_s is denoted by $|B_s|$. Clearly $|B_s| \leq B$ holds.
8. Batch k of family i is represented by B_{ki}.
9. The completion time of job j of family i is denoted by C_{ij}.
10. The weighted tardiness of job ij is represented as $w_{ij}T_j = w_{ij}(C_{ij} - d_{ij})^+$, where we use the notation $x^+ := \max(x, 0)$.

Using the $\alpha|\beta|\gamma$ notation, this problem can be represented as:

$$1|\text{p-batch}, \text{incompatible}|\text{TWT}, \tag{5.2}$$

where p-batch refers to parallel batching and incompatible to the case of incompatible job families. Note that research for problem (5.2) was initiated by the study of the scheduling problem $1|\text{p-batch}, \text{incompatible}|TT$ by Mehta and Uzsoy [180].

We start by proving some simple structural properties of optimal schedules. The first property is due to Uzsoy [303].

Proposition 5.1 *There exists an optimal schedule for problem (5.2) that does not contain partially full batches except possibly the last batch of each family to be processed in the schedule.*

In a next step, we prove that in an optimal schedule there are precedence relationships between the jobs of the same family that can be characterized by job attributes.

Proposition 5.2 *There is an optimal schedule for problem (5.2), where job ij appears before job ik, if the following two conditions are both satisfied:*

$$w_{ij} \geq w_{ik}, \tag{5.3}$$

$$d_{ij} \leq d_{ik}. \tag{5.4}$$

Furthermore, if jobs ij and ik are both tardy in a schedule S and in a schedule S' that is obtained from S by exchanging ij and ik, then the two jobs have to be scheduled by nonincreasing weight order in S.

Proof. The proof is similar to the proof of Proposition 2 in [180]. We consider an optimal schedule S_1 of the form described in Proposition 5.1 and assume that conditions (5.3) and (5.4) are fulfilled. Let job ij be scheduled in batch B_{si} and job ik in batch B_{ti} that is processed before B_{si}. Now we consider a schedule S_2 where the jobs ij and ik are swapped and the remaining jobs are the same. We denote the completion time of B_{si} in S_1 by C_s and the

corresponding completion time of B_{ti} by C_t. Because all jobs of one family have the same processing time, C_s and C_t will be the same in S_2 and no job different from ij and ik will be affected by the swap operation. We define

$$\Delta \mathrm{WT}(S_1) := w_{ij}(C_s - d_{ij})^+ + w_{ik}(C_t - d_{ik})^+, \tag{5.5}$$

$$\Delta \mathrm{WT}(S_2) := w_{ik}(C_s - d_{ik})^+ + w_{ij}(C_t - d_{ij})^+, \tag{5.6}$$

where the two quantities are the weighted tardiness values of the two jobs before and after the swap, respectively. We will show that $\Delta \mathrm{WT}(S_2) \leq \Delta \mathrm{WT}(S_1)$, i.e., the tardiness of S_2 is not greater than the tardiness of S_1. We use the identity

$$(x+y)^+ - x^+ = y - \min\{y, (-x)^+\} \tag{5.7}$$

that holds for all $x, y \in \mathbb{R}$ and $y \geq 0$. We set $\varepsilon := C_s - C_t > 0$ and can estimate

$$\begin{aligned}
&\Delta \mathrm{WT}(S_1) - \Delta \mathrm{WT}(S_2) \\
&\quad = w_{ik}\left[(C_t - d_{ik})^+ - (C_t + \varepsilon - d_{ik})^+\right] - w_{ij}\left[(C_t - d_{ij})^+ - (C_t + \varepsilon - d_{ij})^+\right] \\
&\quad = w_{ij}\left[\varepsilon - \min\left\{\varepsilon, (d_{ij} - C_t)^+\right\}\right] - w_{ik}\left[\varepsilon - \min\left\{\varepsilon, (d_{ik} - C_t)^+\right\}\right] \\
&\quad \geq w_{ik}\left\{\left[\varepsilon - \min\left\{\varepsilon, (d_{ij} - C_t)^+\right\}\right] - \left[\varepsilon - \min\left\{\varepsilon, (d_{ik} - C_t)^+\right\}\right]\right\} \geq 0
\end{aligned}$$

because of $w_{ik} \leq w_{ij}$, $d_{ik} \geq d_{ij}$, and $\varepsilon > 0$. The schedule S_1 is optimal. Therefore, we have $\Delta \mathrm{WT}(S_1) = \Delta \mathrm{WT}(S_2)$, and we can swap the two jobs without changing the TWT value of the schedule before and after the swap.

Next, we assume that both ij and ik are tardy in S and S' and that ij is scheduled before ik. Let again job ij be scheduled in batch B_{si} and job ik in batch B_{ti} that is processed after B_{si} in S. In this case, we obtain

$$\begin{aligned}
\Delta \mathrm{WT}(S) - \Delta \mathrm{WT}(S') &= w_{ij}(C_s - d_{ij}) + w_{ik}(C_t - d_{ik}) - w_{ik}(C_s - d_{ik}) - w_{ij}(C_t - d_{ij}) \\
&= (w_{ik} - w_{ij})(C_t - C_s).
\end{aligned}$$

Because $C_t > C_s$, $\Delta \mathrm{WT}(S) \geq \Delta \mathrm{WT}(S')$ is only valid when

$$w_{ik} \geq w_{ij} \tag{5.8}$$

holds. Therefore, changing the order of job ij and job ik is only beneficial from a TWT point of view when ik has a larger weight than ij. □

Proposition 5.2 shows that it may be beneficial to change the content of batches of the same family. This property can be used to design efficient neighborhood search approaches. Further swapping rules for jobs across batches of the same family similar to inequality (5.8) in Proposition 5.2 can be found in Devpura et al. [67].

We continue by presenting a MIP formulation for problem (5.2). Because of Proposition 5.1, we know the number of batches in an optimal schedule.

Denote this number by n_b. We use the following indices in the MIP formulation:

$b = 1,\ldots,n_b$: batch index
$s = 1,\ldots,f$: family index
$j = 1,\ldots,n$: job index

The following parameters will be used within the model:

B : maximum batch size
d_j : due date of job j
e_{js} : $\begin{cases} 1, \text{ if job } j \text{ belongs to family } s \\ 0, \text{ otherwise} \end{cases}$
M : large number
p_s : processing time of family s
w_j : weight of job j

The following decision variables are necessary:

C_b : completion time of the bth batch
X_{jb} : $\begin{cases} 1, \text{ if job } j \text{ is assigned to the } b\text{th batch} \\ 0, \text{ otherwise} \end{cases}$
Y_{bs} : $\begin{cases} 1, \text{ if batch } b \text{ belongs to family } s \\ 0, \text{ otherwise} \end{cases}$
T_j : tardiness of job j

The scheduling problem (5.2) may be formulated as follows:

$$\min \sum_{j=1}^{n} w_j T_j \tag{5.9}$$

subject to

$$\sum_{b=1}^{n_b} X_{jb} = 1, \quad j = 1,\ldots,n, \tag{5.10}$$

$$\sum_{j=1}^{n} X_{jb} \le B, \quad b = 1,\ldots,n_b, \tag{5.11}$$

$$\sum_{s=1}^{f} Y_{bs} = 1, \quad b = 1,\ldots,n_b, \tag{5.12}$$

$$e_{js} X_{jb} \le Y_{bs}, \quad j = 1,\ldots,n,\ b = 1,\ldots,n_b, \tag{5.13}$$

$$p_s Y_{1s} \le C_1, \quad s = 1,\ldots,f, \tag{5.14}$$

$$C_{b-1} + \sum_{s=1}^{f} p_s Y_{bs} \le C_b, \quad b = 2,\ldots,n_b, \tag{5.15}$$

$$(C_b - d_j) - M(1 - X_{jb}) \le T_j,\ j = 1,\ldots,n,\ b = 1,\ldots,n_b, \tag{5.16}$$

$$C_b, T_j \ge 0, X_{jb}, Y_{bs} \in \{0,1\}, \quad j = 1,\ldots,n,\ b = 1,\ldots,n_b,\ s = 1,\ldots,f. \tag{5.17}$$

The objective (5.9) intends to minimize the TWT value. Constraints (5.10) ensure that each job is assigned to a batch and constraints (5.11) do not allow

more than B jobs to be assigned to the same batch. With constraints (5.12), we make sure that each batch belongs to a single job family, and constraints (5.13) ensure that the families of the jobs assigned to a batch match the family of the batch. Using constraints (5.14), the completion time of the first batch on the machine is computed, whereas constraints (5.15) ensure the correct completion times for all subsequent batches. Finally, constraints (5.16) express the tardiness for each job, and in Eq. (5.17), the nonnegativity and binary constraints are denoted. A similar formulation for minimizing $C_{\max}$ can be found in Klemmt et al. [145].

Note that the MIP formulation (5.9)–(5.17) is able to solve problem instances up to 20 jobs and two families optimally within 2 h of computing time using CPLEX 10.0. Therefore, we can use the MIP to assess the solution quality of heuristics for small-size problem instances and to test whether the implementation of the heuristics is correct or not.

Because of the NP-hardness of the problem, we decompose the problem into two phases. We form batches in the first phase. After this, we sequence these batches in the second phase. We use the ATC index

$$I_{ij}(t) := \frac{w_{ij}}{p_i} \exp\left\{-\frac{(d_{ij} - p_i - t)^+}{\kappa \bar{p}}\right\} \tag{5.18}$$

introduced in Chap. 4 to sequence the jobs. Note that the processing time p_i depends on the family in expression (5.18). When a batch B_s of family i is formed, it can be assessed using the BATC index from Chap. 4. Therefore, we have

$$I_{B_s}(t) := \sum_{ij \in B_s} I_{ij}(t), \tag{5.19}$$

i.e., we sum up the ATC indices of the jobs that form the batch. For forming and some initial sequencing of the batches, we use the following algorithm.
Algorithm ATC-BATC

1. Sort the jobs within each family according to the ATC dispatching rule, i.e., according to nonincreasing $I_{ij}(0)$ values.
2. For each family, starting from the first job of each sequence obtained in step 1, form full batches as long as this is possible. Let $k_i := \left\lceil \frac{n_i}{B} \right\rceil$ be the number of batches that are formed of family i. Set $l_i = 0, i = 1, \ldots, f$. Initialize $t = 0$.
3. Let the current partial schedule contain l_i batches of family i. Denote the set of unscheduled batches by $\Phi(l_1, \ldots, l_f)$. Note that $\Phi(l_1, \ldots, l_f)$ contains $k_i - l_i$ batches of family i. Calculate the $I_{B_s}(t)$ index for each $B_s \in \Phi(l_1, \ldots, l_f)$. Schedule the batch B_s with the highest BATC index. Let B_s be a batch of family s. Update $l_s := l_s + 1$ and $t := t + p_s$.
4. When all batches are scheduled, then stop. Otherwise, go to step 3.

The κ values used in the indices (5.18) and (5.19) are determined by a grid search approach, i.e., we use $\kappa = 0.1h,\ h = 1, \ldots, 50$ to determine schedules

by the ATC-BATC algorithm. Then we pick the schedule that leads to the smallest TWT value. We denote the corresponding κ value by κ_{best}. For a fixed κ, we use for abbreviation the notation ATC-BATC(κ). Note that a similar heuristic can be based on the EDD dispatching rule (see Perez et al. [232]). In that heuristic, jobs are sequenced in nondecreasing due date order. Then, batches are formed per family starting with the jobs with the smallest due date. After the batch formation, we sequence the batches by taking the batch that contains the job that has the smallest due date. The resulting scheme is called EDD-EDD. However, ATC-BATC outperforms this heuristic.

The initial sequencing of batches obtained by ATC-BATC can be further improved by dynamic programming and a decomposition heuristic. We continue with describing the two heuristics in detail.

We start with the dynamic programming formulation that minimizes TWT given a set of formed batches. Let $N := \sum_{i=1}^{f} \lceil \frac{n_i}{B} \rceil$ be the number of batches formed by the ATC-BATC rule. The set of all the N batches is denoted by Φ. Let $f(J)$ denote the minimum TWT value of a partial schedule containing the batches of the set $J \subseteq \Phi$. We describe the boundary condition, the optimal value that is the TWT value for the optimal sequence, and the recursion relation. The boundary condition is used to initialize the recursion relation. We obtain the following:

Boundary condition: $f(\emptyset) := 0$,
Optimal value: $f(\Phi)$,
Recursive relation:

$$f(J) := \min_{B_s \in J} \left\{ f(J - \{B_s\}) + \sum_{k \in B_s} w_k \left(\sum_{B_r \in J} p_{F(B_r)} - d_k \right)^+ \right\}. \tag{5.20}$$

We use the abbreviation DP for this dynamic programming procedure. The idea behind the DP is straightforward. The optimal sequence for a subset of batches is determined in each iteration assuming that this subset goes first. This is carried out for each subset of batches of size l. Using the recursive relation, this is extended to each subset of size $l+1$. Each of the $l+1$ batches is a candidate to be appended. It is not necessary to know the sequence of the batches for the subsets of size l; knowing the contribution of the l batches to the objective is enough. The optimal sequence of batches can be determined by a simple backtracking procedure after $f(\Phi)$ is determined. Because the DP considers all subsets of Φ, it requires an amount of computation time that is $O(2^N)$. Hence, the DP is appropriate only for a small number of batches.

Note that a similar dynamic programming formulation is presented for the problem $1|\text{p-batch}, \text{incompatible}|TT$ in [180]. However, because of more structural insights into this problem with TT objective, the number of possible batches to be considered at a partial schedule is smaller.

Because of the large computational burden of the DP, we also consider a heuristic that decomposes the sequencing problem for batches into a series

of smaller sequencing problems that can be solved optimally. This type of heuristic was proposed by Chambers et al. [43] for the non-batching case and later applied by Mehta and Uzsoy [180] for solving $1|\text{p-batch}, \text{incompatible}|TT$. The procedure can be described as follows.

Algorithm Decomposition Heuristic (DH)

1. Determine an initial sequence of batches using ATC-BATC(κ_{best}). Denote this schedule by S. Set the values of the parameters λ, α, and iter. The parameter λ is the number of batches that are sequenced optimally. The first α batches for the resulting optimal sequence are fixed, and finally, iter is the total number of iterations.
2. Let $B_{[i]}$ denote the batch in the ith position of S. Furthermore, let $N := \sum_{j=1}^{f} \lceil \frac{n_j}{B} \rceil$ be the number of batches to be scheduled. We initialize $k := \min(\lambda, N)$. Set $\tilde{S} = \emptyset$, where $\tilde{S}$ represents the final schedule to be constructed. Initialize $j = 0$ and $\hat{i} = 1$. Finally, let $P := \{B_{[1]}, \ldots, B_{[k]}\}$ be the set of batches of the initial subproblem.
3. Determine an optimal sequence of the batches of P by using complete enumeration. Denote the batch in the ith position of this sequence by $\beta_{[i]}$.
4. We define $y := \min(|P|, \alpha)$. Fix the batches $\beta_{[1]}, \ldots, \beta_{[y]}$ in positions $j+1, \ldots, j+y$ in $\tilde{S}$. Update $j := j + y$. Set $P := (P - \{\beta_{[1]}, \ldots, \beta_{[y]}\}) \bigcup \{B_{[j+1]}, \ldots, B_{[x]}\}$, where $x := \min(j + \alpha, N)$.
5. When $j < N$, then go to step 3; otherwise, go to step 6.
6. When $\hat{i} < \text{iter}$ and the TWT value of $\tilde{S}$ is smaller than the TWT value of S, then set $S := \tilde{S}$ and go to step 2 and update $\hat{i} = \hat{i} + 1$; otherwise, stop. The last found schedule is the resulting one.

We use the notation DH($\lambda, \alpha, \text{iter}$) for this heuristic. Usually, we use the setting $\lambda = 5$, $\alpha = 2$, and finally $\text{iter} = 15$. In this case, 5! schedules have to be evaluated within each step of DH, which is feasible from a computational point of view. So far, we change entire sequences. But in the spirit of Proposition 5.2, it may also be beneficial to swap jobs across batches of the same job family.

We continue by presenting some results of computational experiments. The experiments are performed with respect to a different number of jobs per family and a different maximum batch size. The processing times of the jobs for each family are given in Table 5.1. Note that this setting mimics the situation in wafer fabs, where process steps on batch machines tend to be long, but at the same time, because jobs on different stages compete for processing, short processing times are possible. We expect that the performance of the heuristics also depends on the due date setting. Therefore, we select due dates according to the following distribution:

$$d_{ij} \sim U\left(\mu\left(1 - \frac{R}{2}\right), \mu\left(1 + \frac{R}{2}\right)\right), \tag{5.21}$$

where R is called the range parameter. The quantity μ within the distribution in expression (5.21) is given by $\mu = (1-T)\widehat{C}_{\max}$, where T is the expected percentage of tardy jobs and $\widehat{C}_{\max}$ is an estimator for $C_{\max}$ given by $\widehat{C}_{\max} := \frac{nE(P)}{B}$. We denote by $E(P)$ the expected processing time. A larger value of T leads to a smaller mean due date and hence to a larger TWT value, and the larger the value of R, the larger the range and variance of the due dates. In total, we consider 320 problem instances.

Table 5.1: Design of experiments for problem (5.2)

Factor	Level	Count
f	4	1
n_i	30, 40, 50, 60	4
B	4, 8	2
p_i	2 with a probability of 0.2 4 with a probability of 0.2 10 with a probability of 0.3 16 with a probability of 0.2 20 with a probability of 0.1	1
w_{ij}	$\sim U(0,1)$	1
d_{ij}	$\sim U\left(\mu\left(1-\frac{R}{2}\right), \mu\left(1+\frac{R}{2}\right)\right)$ $T = 0.3, 0.6,\ R = 0.5, 2.5$	4
	Total factor combinations	32
	Number of problem instances per combination	10
	Total number of problem instances	320

The problem instances are solved using ATC-BATC, ATC-BATC-DH, ATC-BATC-DP, EDD-EDD, and finally EDD-EDD-DH. The first two entries in the notation triplet refer to the scheme that is used to form the batches and determine an initial sequence, whereas the third entry is related to the scheme to improve the initial schedule. Among the different heuristics, we use schedules obtained by ATC-BATC-DH as a reference, because ATC-BATC-DH outperforms the remaining heuristics with respect to the combination of solution quality and computational efforts.

In Table 5.2, we present some computational results. We show the TWT values obtained by the heuristics relative to the TWT values of the ATC-BATC-DH heuristic. Instead of comparing all problem instances individually, the instances are grouped according to factor levels such as the number of jobs per family and maximum batch size. For example, results for a maximum batch size of 4 imply that all other factors have been varied, but B has been kept constant at 4. Due to the fact that sometimes the TWT value is zero, we sum up the TWT values of all the instances for that factor level and divide by the sum of the TWT values obtained by ATC-BATC-DH. In the case where the batches are formed and also sequenced with the EDD dispatching rule, i.e., the EDD-EDD heuristic, the performance is 51 % worse than that of the ATC-BATC-DH heuristic across all the problem instances. When the

Table 5.2: Computational results for problem (5.2)

Compare	EDD-EDD	ATC-BATC	EDD-EDD-DH	ATC-BATC-DP
n_i				
30	1.5035	1.3640	1.1234	0.9912
40	1.5402	1.4332	1.1033	0.9945
50	1.5169	1.4107	1.0923	0.9837
60	1.4784	1.3826	1.0877	0.9777
B				
4	1.4995	1.4133	1.0805	0.9778
8	1.5175	1.3681	1.1300	0.9976
% of tardy jobs				
$T = 30\,\%$	1.4149	1.2838	1.0968	0.9896
$T = 60\,\%$	1.5359	1.4375	1.0965	0.9824
Due date range				
$R = 0.5$	1.5520	1.4397	1.1275	0.9733
$R = 2.5$	1.4405	1.3416	1.0537	0.9994
Overall	1.5053	1.3987	1.0966	0.9842

heuristic EDD-EDD-DH is used, the overall result obtained is just 10 % worse than that of the ATC-BATC-DH. On the other hand, if the method to form batches is a little more sophisticated than just arranging jobs by due dates (i.e., using the ATC rule, but the final sequence is also obtained by the BATC approach), the result is 40 % worse than that of ATC-BATC-DH. The ATC-BATC-DH performance is similar to that of ATC-BATC-DP. In fact, ATC-BATC-DP performs only slightly more than 1 % better than the ATC-BATC-DH on average. In general, the heuristics using the EDD dispatching rule to form batches improve compared to ATC-BATC-DH when n_i increases. However, the performance decreases as B increases. On the other hand, those heuristics forming batches with the ATC rule perform about the same when n_i increases and have mixed performance when B increases. Increasing T does not affect the results much for any of the heuristics except for the EDD-EDD and ATC-BATC heuristics. In general, as expected, all the heuristics perform better when R is large.

Additional computational results for a different number of families can be found in [232]. Note that for $f = 5$ families and 60 jobs per family, the DP heuristic is very time-consuming and cannot be used any longer to improve the initial schedule. A discussion of different scheduling problems with batch machines in semiconductor manufacturing is contained in the survey by Mathirajan and Sivakumar [176].

5.3.2 Scheduling Jobs on a Single Cluster Tool

The external scheduling of cluster tools, i.e., the job sequencing for this type of equipment, is challenging because the CT of wafers—and therefore also the

processing time of the jobs in a cluster tool—depends on the wafer recipes used, cluster tool control and architecture, wafer waiting times, and the sequencing of the jobs (see Dümmler [75]). The following assumptions are made for the scheduling of jobs on a single cluster tool:

1. There are n jobs to be scheduled.
2. All jobs are available at time $t = 0$.
3. The cluster tool has two load locks.
4. Once a cluster tool chamber is started, it cannot be interrupted, i.e., no preemption is allowed.

It is well known that TP maximization can be achieved by $C_{\max}$ minimization. Therefore, we consider the performance measure $C_{\max}$. Using the $\alpha|\beta|\gamma$ notation, the scheduling problem can be described as

$$1|\text{lrc}|C_{\max}. \tag{5.22}$$

Problem (5.22) is NP-hard because the Hamiltonian cycle problem can be reduced to it (see Bianco et al. [26]). This scheduling problem is rarely discussed in the literature except by Oechsner and Rose [218, 219]. Bianco et al. [26] propose several dominance criteria and lower bounding schemes for the more general problem $1|r_j, \text{lrc}|C_{\max}$.

Dümmler [75] considers two main approaches to deal with this complex environment. In the first approach, a detailed deterministic simulation model of the cluster tool is used to evaluate the wafer CT values for job sequences in the scheduling algorithm. A cluster tool simulation engine is described by Dümmler [76]. In a second approach, wafer cycle time approximations are used to construct partial schedules and the detailed model of the cluster tool is only used to determine the $C_{\max}$ value of the final schedules. However, the CT approximations are also found using the simulation model of the cluster tool doing preprocessing before the scheduling decisions are made. In this monograph, we describe only the second approach in detail.

In the following, we present a beam search algorithm proposed by Oechsner and Rose [218, 219] to solve problem (5.22). To introduce the beam search algorithm, we first have to model the scheduling problem by a branching tree as introduced in Sect. 3.2.2. Each node of the tree represents a partial schedule. The root is the sequence with no job scheduled on a distinct position, i.e., (*,*,*,*), with * being a placeholder for a position in the sequence. Each child is a partial schedule with one job scheduled in the first position of the sequence, and for each child this job is different, i.e., we have (1,*,*,*), (2,*,*,*), etc. At each consecutive level, one more job is put into the sequence, until full sequences/schedules are reached. These are the leaves of the branching tree. This means that on level i, one of the jobs still to be scheduled is selected for position i of the schedule. With a finite set of jobs, the number of children per node decreases by one on each level, since fewer jobs remain to be scheduled. We show an example of such a partial branching tree including four jobs in Fig. 5.2.

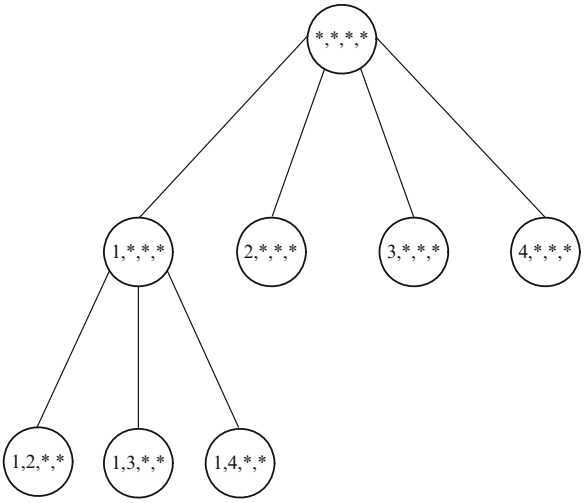

Figure 5.2: Partial branching tree with four different jobs [219]

When we have to schedule jobs belonging to a small number of job types, then the initial number of children per node is given by the number of job types because only the job type is important for the $C_{\max}$ objective. On the other hand, the number of nodes decreases only if all jobs of one job type are scheduled. Each different job type has a specific recipe. A wafer recipe determines the internal flow of the wafers within the cluster tool, i.e., the sequence of chambers to be visited (see the description in Sect. 2.2.3).

Beam search is a heuristic variant of the branch-and-bound approach (see Sect. 3.2.2). Consider a scheduling problem represented as a tree as shown in Fig. 5.2. For a large set of jobs, this branching tree becomes large because of the large number of children of each node on higher levels. Branch-and-bound aims to eliminate some of the children of a node by evaluating each node and comparing the resulting value with a lower bound. If this value is larger, the node and all of its children are discarded. Thus, fewer nodes have to be considered on the next level. However, it is common for many nodes to remain that need to be evaluated. While a branch-and-bound algorithm determines optimal solutions, it can be very time-consuming when the discrete optimization problem is NP-hard. Since problem (5.22) is NP-hard, branch-and-bound is not possible when schedules have to be obtained within a reasonable amount of time.

The aim of a beam search algorithm is to limit the number of nodes that have to be evaluated on each level of the branching tree. With each step, a certain set of nodes is selected based on an evaluation function, and the remaining nodes are discarded. Only the non-discarded nodes are expanded, keeping the size of the branching tree relatively small. The number of these nodes is called the beam width β of the search. When the branching tree

has been expanded fully, this means that on each level except the first, there are at most β nodes. This results in the same number of job sequences that have to be simulated at each level. Thus, the algorithm is faster than branch-and-bound, but the optimality of the resulting schedule can no longer be guaranteed. However, if the selection process for the nodes is appropriate, a near optimal (and sometimes the optimal) solution can still be obtained. On the other hand, if the evaluation is too time-consuming, the speed advantage will be lost.

We continue by describing an appropriate evaluation function used to prune the partial branching tree. To evaluate a certain node of the branching tree, we have to decide how well it fits with the last scheduled job of the partial schedule.

Because we have only two load locks, not every job that is in the partial schedule is of interest when we insert a job after the last job in the partial schedule at time t. For most jobs of the partial schedule, the inequality $C_j < t$ is valid, and these jobs are therefore not of interest at time t. We only need to look at the jobs that are being serviced in the cluster tool at t, because only those jobs will have an effect on the C_j of the other jobs. The situation is shown in Fig. 5.3 adopted from [219].

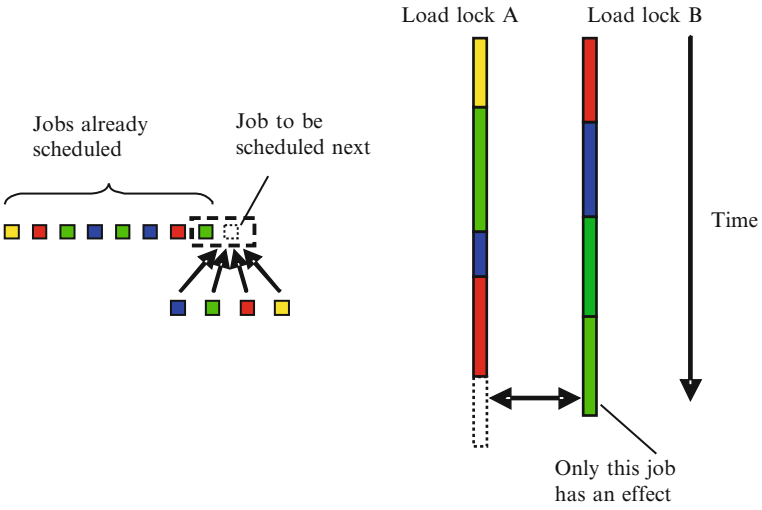

Figure 5.3: Example for including a new job into the partial schedule

We now describe two evaluation functions for the comparison of job combinations. For the first function, the slowdown factor of all possible two-job

combinations is computed. The slowdown factor of two jobs i and j is defined as follows:

$$\mathrm{sdf}(i,j) := \frac{\mathrm{CT}(i,j)}{\mathrm{CT}(i)}, \tag{5.23}$$

where $\mathrm{CT}(i,j)$ is the mean CT of a wafer of job i when it is serviced at the same time as job j. $\mathrm{CT}(i)$ is the mean cycle time of a wafer of job i with only this job being processed in the cluster tool. Since $\mathrm{sdf}(i,j)$ is much higher if two jobs i and j do not fit together well, i.e., when they compete for a resource, it is a good indicator for evaluating job combinations.

The corresponding $\mathrm{CT}(i)$ values are obtained by short off-line simulations completed before starting the scheduling process and the computed sdf values are stored for fast access by the beam search algorithm.

The resulting algorithm can be summarized as follows where we assume for simplicity reasons that the jobs belong to different job types and that $\beta < n$ holds.

Algorithm Beam Search with Two-Job Slowdown

1. Create the root of the branching tree BT. Initialize the level l by $l = 1$. Denote by Φ_k the set of unscheduled jobs associated with node k at the first level and set initially $\Phi_k := \{1,\ldots,n\}$, $k = 1,\ldots,n$.
2. Form a partial schedule for each node by setting $j_{[1]} := k$ for $k = 1,\ldots,n$. Assign the resulting partial schedules to the nodes of level $l = 1$. Set $l := l+1$ and update the corresponding Φ_k accordingly.
3. For each node at level l, consider the job $j_{[l]}$, i.e., the last job of the partial schedule. If for the corresponding set $\Phi_k \neq \emptyset$, consider all the pairs of the form $(j_{[l]}, j_{[l+1]})$, where $j_{[l+1]} \in \Phi_k$, and sort them by nondecreasing $\mathrm{sdf}(j_{[l]}, j_{[l+1]})$ values; otherwise, go to step 4. Use the first $\min(\sum|\Phi_k|, \beta)$ pairs for all nodes of level l to further expand the corresponding nodes of BT. Set $l := l+1$ and update the corresponding Φ_k accordingly. Repeat this step.
4. Simulate the schedules associated with each leaf using the cluster tool simulator to determine $C_{\max}$ and select the schedule with the smallest $C_{\max}$ value.

We abbreviate this algorithm BS-SLD-2. Note that step 4 is necessary because BS-SLD-2 improves the schedules just locally as only pairs of jobs are considered. It is also clear that BS-SLD-2 tends to be computationally expensive when β becomes large.

We refine BS-SLD-2 by taking more than two jobs into account and call it BS-SLD-3. We assume that at least two jobs are part of the partial schedule and call the last two jobs $j-1$ and j, respectively, according to their position in the partial schedule. We consider the case that the two jobs are interleaved and that job j is the last job scheduled, but has a small CT value compared to the CT value of job $j-1$. This means that $C_{j-1} > C_j$ is valid. Of course, the job to be scheduled next has to be compatible with job $j-1$ and not with job j.

In this case, we try to find a job j^* such that $sld(j-1, j^*)$ is as small as possible. To decide which method to use, we look at the CT values of job $j-1$ and job j. If

$$\mathrm{CT}(j-1) \geq \mathrm{thresh}\ \mathrm{CT}(j), \tag{5.24}$$

there is a high probability that $C_j < C_{j-1}$ holds, and we choose the refined method that considers $j-1$ instead of j. When condition (5.24) is not true, we use the method from BS-SLD-2 that computes the sdf value for the last job scheduled. Therefore, we have to change step 3 of BS-SLD-2. The setting $\mathrm{thresh} = 4$ is chosen based on extensive experimentation and might be changed for different problem instances.

Computational results are presented in [218, 219]. The two heuristics provide solutions with a $C_{\max}$ value close to the optimal or best-known $C_{\max}$. However, in case of larger values for n, their performance degrades. As expected, BS-SLD-3 outperforms BS-SLD-2, but the difference is small.

An alternative evaluation function is described in [219]. The method is based on recipe comparison. The basic idea is to select only job pairs with a large number of alternative chambers for each process step to avoid situations where wafers of the two jobs compete for scarce resources. BS-SLD-3 is outperformed by the BS-type algorithm that uses the latter evaluation function.

We also note that Dümmler [75, 76] suggests a GA to tackle a generalization of the problem discussed in this section, namely $\mathrm{Pm}|\mathrm{lrc}|C_{\max}$. The GA simultaneously assigns the jobs to cluster tools and sequences them. All the chromosomes of an iteration are assessed using the cluster tool simulation engine.

5.3.3 Scheduling Jobs on Parallel Machines with Sequence-Dependent Setup Times

We consider parallel machine scheduling problems with sequence-dependent setups in this section as can be found in the ion implantation work area in a wafer fab (cf. Sect. 2.2.3). A job will enter the ion implantation work center when it completes its previous process step. Each implant process step needs a potentially different processing time and carries two types of information: product type and the specie being implanted. The setup time between different product types is often neglected because it is considerably less than the setup time between different species. Boron, difluoroborane, phosphorous, and arsenic are typical species. When the product with a certain specie is completed on the machine, a setup is required if the next job processed in the same machine has a different specie.

The following assumptions are made for this scheduling problem:

1. All the jobs j are available at time $r_j \geq 0$.

2. Once a machine is started, it cannot be interrupted, i.e., no preemption is allowed.
3. The parallel machines are identical.
4. There are sequence-dependent setup times.

Using the $\alpha|\beta|\gamma$ notation, the considered class of scheduling problems can be described as follows:

$$\text{Pm}|r_j, s_{jk}|\text{PM}_i, i = 1,2,3, \tag{5.25}$$

where s_{jk} represents the setup time that occurs when job k is processed next and the current setup state is appropriate for job j. We use the three performance measures $\text{PM}_1 := C_{\max}$, $\text{PM}_2 := \text{TWC}$, and $\text{PM}_3 := \text{TWT}$. These three performance measures represent the indices used for the performance of most manufacturing systems, as discussed in Sect. 3.3.1. Each of the three scheduling problems from Eq. (5.25) is NP-hard, even if $s_{jk} = 0$ is assumed (see Brucker [34]). Therefore, we have to look for efficient heuristics.

We start by presenting a MIP formulation for problem (5.25) with $C_{\max}$ objective. We use the following indices in the MIP formulation:

$j = 0,\ldots,n$: job index.

Note that job $j = 0$ is a dummy job that is used on each machine. We have $p_0 = r_0 = 0$ and $s_{0j} = 0,\ j = 1,\ldots,n$. The following parameters will be used within the model:

r_j : ready time of job j
p_j : processing time of job j
s_{jk} : sequence-dependent setup time when processing job k immediately after job j
M : large number

The following decision variables are necessary:

$$X_{ij} : \begin{cases} 1, \text{ if job } i \text{ immediately precedes job } j \text{ on the same machine} \\ 0, \text{ otherwise} \end{cases}$$
C_j : completion time of job j on machine i
$C_{\max}$: makespan

The scheduling problem can be formulated as follows:

$$\min\ C_{\max} \tag{5.26}$$

subject to:

$$\sum_{i=0}^{n} X_{ij} = 1, \quad j = 1,\ldots,n, \tag{5.27}$$

$$\sum_{j \neq i} X_{ij} \leq 1, \quad i = 1,\ldots,n, \tag{5.28}$$

$$\sum_{k=1}^{n} X_{0k} \leq m, \tag{5.29}$$

$$X_{ij} \leq \sum_{\substack{k=0 \\ k\neq i, k\neq j}}^{n} X_{ki}, \quad i=1,\ldots,n,\ j=1,\ldots,n,\ i\neq j, \tag{5.30}$$

$$C_j + s_{jk} + p_k \leq C_k + M(1-X_{jk}), \quad j=0,\ldots,n,\ k=1,\ldots,n,\ j\neq k, \tag{5.31}$$

$$C_0 = 0, \tag{5.32}$$

$$C_j \geq r_j + \sum_{k=0}^{n} \left(p_j + s_{kj} + (r_k + p_k - r_j)^+\right) X_{kj}, \quad j=1,\ldots,n, \tag{5.33}$$

$$C_j \leq C_{\max}, \quad j=1,\ldots,n, \tag{5.34}$$

$$X_{ij} \in \{0,1\}, \quad i=0,\ldots,n,\ j=1,\ldots,n,\ i\neq j. \tag{5.35}$$

The objective is to minimize the makespan. Constraints (5.27) make sure that each job is assigned to exactly one machine and has exactly one predecessor. The dummy job 0 is used within $X_{0k}, k \in \{1,\ldots,n\}$. We have $r_0 = p_0 = 0$ and $s_{0j} = 0,\ j=1,\ldots,n$. Constraints (5.28) ensure that each job except the dummy job has at most one successor. The number of successors of the dummy jobs is at most m because of constraint (5.29). When a given job i is processed on a machine, then a predecessor has to exist on the same machine. This is modeled by constraints (5.30). When a job k is assigned to a machine immediately after job j, i.e., when $X_{jk} = 1$, its completion time has to be greater than the sum of the completion time of job j, the setup time between j and k, and the processing time of k. This is expressed by constraints (5.31). In case of $X_{jk} = 0$, constraints (5.31) are also fulfilled because of the big M. Constraint (5.32) sets the completion time of the dummy job to zero. Constraints (5.33) indicate that the completion time of a job is always larger than its ready time plus the processing time and the setup time. The makespan is not smaller than any of the completion times of the jobs. This is modeled by constraints (5.34). Finally, constraints (5.35) model the fact that the main decision variables X_{ij} only take binary values. The MIP formulation (5.26)–(5.34) has been used to solve problem instances up to $m = 2$ and $n = 18$ optimally. Therefore, we can use the formulation to assess whether heuristics are correctly implemented or not. Note that similar formulations can be derived for the two remaining performance measures.

Next, we describe some appropriate heuristics. A GA (cf. the description in Sect. 3.2.6) is used for this parallel machine scheduling problem for several reasons. First, a GA is not only flexible in dealing with additional processing restrictions of problems such as w_j, d_j, r_j, or p_j of the jobs, but can also easily handle different objectives without changing the complete evaluation algorithm or technique. We will see that the GAs for the different scheduling problems from expression (5.25) are very similar. Second, a GA can provide feasible solutions in each generation while the feasible solutions of MIP

formulations are greatly dependent on the nature of the problems. Third, a GA produces a population of solutions. Knowing a set of feasible solutions is desirable because this offers flexibility in case of machine breakdowns when the schedule will be implemented in the BS.

A GA hybridized with dispatching rules is used to solve problem (5.25). The GA is used to assign jobs to individual machines, and each machine is scheduled according to a single machine dispatching rule that will be described later. The sequencing of the jobs assigned to an individual machine is necessary because we have to evaluate each chromosome using one of the performance measures $\mathrm{PM}_i, i = 1,2,3$. After all the single machine schedules are determined, the schedule of parallel machines is completed, and the performance measure is returned by the GA and is used in creating the solutions in the next generation. We will use the abbreviation HGA for the hybridized GA throughout the rest of this section. The main architecture of the HGA is shown in Fig. 5.4.

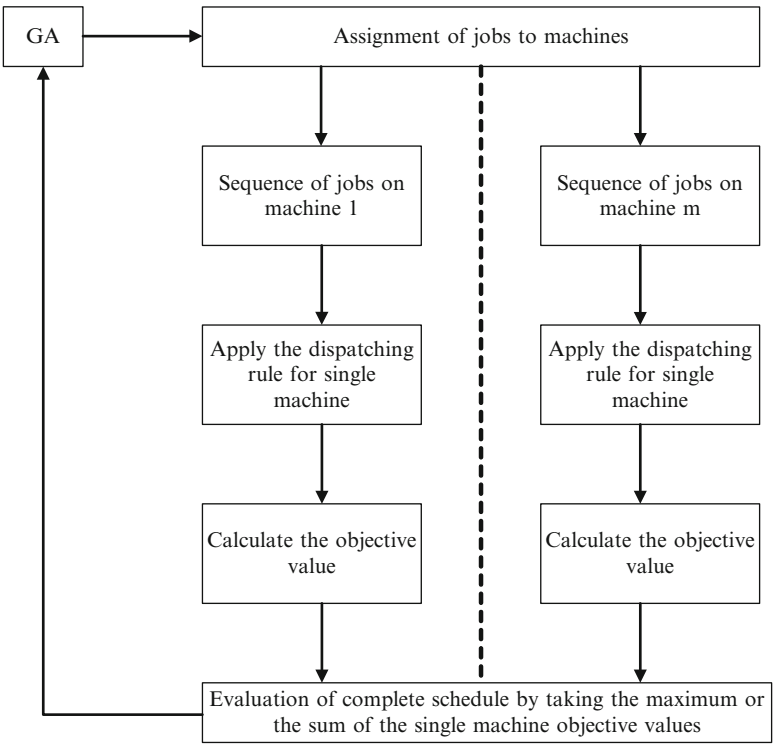

Figure 5.4: Hybridization of a GA to solve parallel machine scheduling problems

The following steps describe how the HGA approach works, i.e., we discuss the encoding scheme, the initialization scheme, selection, crossover, mutation operations, and finally the evaluation and termination scheme.

1. Encoding: We use a job-based representation. In the case of n different jobs, the following representation is used for machine assignment:

$$c := (m_1, m_2, \ldots, m_{n-1}, m_n), \tag{5.36}$$

   where we denote by m_j the machine that is used for processing job j. For example, $c := (1,10,4,4,4,4)$ means that job 1 will be processed on machine 1, job 2 on machine 10, and the remaining jobs on machine 4. We call each of these representations a single chromosome. HGA maintains a population of these chromosomes.
2. Initialization: Each chromosome, represented by the array in expression (5.36), is initialized by randomly assigning an equally likely integer from the set $\{1,\ldots,m\}$ to each of the n jobs.
3. Parents selection: Selection is an operation to select chromosomes according to their performance, for the purpose of generating new offspring. For selection, the roulette wheel technique is used (see Goldberg [103] and Michalewicz [183]). The probability of selection of a certain chromosome is proportional to its fitness. Hence, the chromosomes with better performance are given a better chance to survive in the next generation. The fitness is calculated using linear scaling. The fitness function of the HGA is provided by one of the expressions $1/\mathrm{PM}_i, i = 1,2,3$.
4. Crossover: A one-point crossover (cf. Goldberg [103] or Michalewicz [183]) is utilized in which two parent chromosomes are selected randomly according to a predefined crossover probability p_c from the set obtained in step 3. Furthermore, a crossover point is also randomly chosen to divide each of the two parents. We denote the first chromosome by $c_1 := (m_{11}, m_{12}, \ldots, m_{1,n-1}, m_{1n})$ and the second chromosome by $c_2 := (m_{21}, m_{22}, \ldots, m_{2,n-1}, m_{2n})$. After performing a one-point crossover with a crossover point at position s, we obtain the two resulting chromosomes given by $c_3 := (m_{21}, \ldots, m_{1s}, \ldots, m_{1n})$ and $c_4 := (m_{11}, \ldots, m_{2s}, \ldots, m_{2n})$.
5. Mutation: According to a pre-defined mutation probability p_m for each gene of the resulting offspring, it is decided whether or not to change the machine assignment. A selected gene is randomly changed to a different machine number by a flip operator. For example, the operator might select entry two in the array $(2,1,3,3,1,2)$ and 2 as the randomly generated number to be inserted. Thus, the new string will be $(2,2,3,3,1,2)$. The p_m value is typically one percent of the size of the array (see Levine [160]), because a change of the entry in an array often means a dramatic change of searching direction. A proper value of p_m helps the algorithm maintain diversity while searching. On the other hand, an improper p_m value will either damage the strings if it is too large or lead to premature convergence if it is too small.

6. Elitism: A steady state genetic algorithm with overlapping populations is used. The worst elements of the preceding population are replaced. This is called an elitist strategy. Rudolph [272] proved that this strategy helps the algorithms converge to the optimum. The number of replaced elements depends on the given replacement probability.
7. Evaluation: The three objectives $C_{\max}$, TWC, and TWT are evaluated separately.
8. Termination: The HGA is controlled by a prescribed number of evaluations, which is equal to the population size times the number of generations.

The HGA requires the calculation of the objective function for each element of the population. Therefore, a sequence of the jobs has to be determined for each single machine. Dispatching rules are used to determine a sequence of the jobs. The FIFO, EDD, SPT, WSPT, LPT, and finally the HVF (highest weight) dispatching rules with priority index (4.7) (cf. Sect. 4.2) are outperformed by two dispatching rules that take sequence-dependent setup times into account.

The first dispatching rule among them is the setup avoidance dispatching rule LSC with priority index (4.11) (cf. Sect. 4.2). A tie between jobs for a given machine will be broken according to one of six secondary dispatching rules mentioned above. In the case of FIFO with multiple jobs ready at the same time, EDD will be used to break the tie.

The second rule is the ATCS dispatching rule. It was proposed by Lee and Pinedo [159] to find schedules with a small TWT value for the scheduling problem $1|s_{jk}|\text{TWT}$. The index of job j at time t when job l has completed its processing on the machine is calculated by

$$I_{lj}(t) := \frac{w_j}{p_j} \exp\left\{-\frac{(d_j - p_j - t)^+}{\kappa_1 \bar{p}}\right\} \exp\left\{-\frac{s_{lj}}{\kappa_2 \bar{s}}\right\}, \tag{5.37}$$

as introduced in Sect. 4.3.2. The resulting schedules will potentially be improved by using a swapping technique. We consider swaps between pairwise adjacent jobs. Because we use only one pass, we use the notation one pass adjacent (ADJ) swap. To have a fair comparison with the LSC dispatching rule, ADJ is also applied to LSC. We also combine the LSC dispatching rule with the SPT rule as a secondary dispatching rule.

The algorithm HGA can be summarized as follows.

Algorithm HGA

1. Create an initial population as described above. Calculate the fitness values of all chromosomes of the initial population by solving the resultant single machine scheduling problems using the LSC and ATCS dispatching rules, respectively. Improve these single machine schedules by ADJ. Set $\text{iter} = 1$.
2. Select appropriate parent chromosomes to find a new population.

3. Perform crossover and mutation operations applied to the chromosomes chosen in step 2 to produce offspring.
4. Calculate the fitness values of all chromosomes of the population by solving the resultant single machine scheduling problems using the LSC and ATCS dispatching rules. Improve these single machine schedules by ADJ. Update the current best chromosome.
5. Select a new generation based on the results in steps 3 and 4, i.e., replace only a certain part of the chromosomes of the current population by offspring. Set $\text{iter} = \text{iter} + 1$.
6. When $\text{iter} > \text{iter}_{\max}$ then stop; otherwise, go to step 2.

A population size of 8, $p_m = 0.01$, and $p_c = 0.6$ is selected for HGA. The number of generations was 300. We run HGA ten times with different seeds to obtain statistically significant results. A grid search was used to select appropriate values for the scaling parameters (κ_1, κ_2) in index (5.37).

Next, we describe the design of experiments used. We expect that the performance of HGA depends on the range of the weights, the range of the due dates, the range of the ready times, and the ratios of average processing times to average setup times. A total of 36 cases with 30 random problem instances for each factor combination are generated, resulting in 1,080 problem instances. The setup time between jobs of different species is a $U(3k, 7k)$-distributed random variable where $k = 2, 6, 10$ to model three setup levels, respectively. The ratio represents the impact of the magnitude of the setup time to the total processing times. The processing times are random variables with a probability distribution $U(-9, 9)$ plus 50, 30, and 10 for each of the three levels. This results in ratios of average processing times to average setup times equal to 5, 1, and 1/5, respectively. The resulting design of experiments is summarized in Table 5.3.

Table 5.3: Design of experiments for problem (5.25)

Factor	Level	Count
w_j	Narrow $\sim U(1,10)$ Wide $\sim U(1,20)$	2
r_j	High load: $r_j \equiv 0$ Moderate load: 50 % $r_j \equiv 0$ and 50 % $r_j \sim U(0,720)$ Low load: $r_j \sim U(0,720)$	3
d_j	Narrow: $r_j + z_j \sum_{j=1}^{n} p_j$, where $z_j \sim U(-1,2)$ Wide: $r_j + z_j \sum_{j=1}^{n} p_j$, where $z_j \sim U(-2,4)$	2
$\bar{p}/\bar{s}$	High: 50/10 with $p-50 \sim U(-9,9)$, $s/2 \sim U(3,7)$ Moderate: 30/30 with $p-30 \sim U(-9,9)$, $s/6 \sim U(3,7)$ Low: 10/50 with $p-10 \sim U(-9,9)$, $s/10 \sim U(3,7)$	3
	Total factor combinations	36
	Number of problem instances per combination	30
	Total number of problem instances	1,080

Because we know that ATC-type dispatching rules do not perform well for wide due dates and widespread ready times (cf. Balasubramanian et al. [18]), HGA is hybridized with LSC in this situation. This setting is also used for the $C_{\max}$ objective. In the remaining cases, HGA is hybridized with the ATCS dispatching rule.

The relative distance (RD) percentage is used to evaluate each heuristic, and it represents how far the performance of the current method is away from the best performance of all heuristics under consideration. ARD is the average RD from all instances of each factor combination. A low ARD value indicates better overall performance, and a heuristic with ARD of 0 % is the best rule for all the problem instances considered. We obtain

$$\mathrm{RD}(H) := (\mathrm{PM}(H) - \mathrm{BP})/\mathrm{BP}, \tag{5.38}$$

where $\mathrm{PM}(H)$ is the performance measure value of heuristic H and BP is the best-known performance measure value over all studied heuristics, i.e., the list scheduling approaches including all the dispatching rules and the HGA. The corresponding computational results are shown in Table 5.4.

Table 5.4: Computational results for problem (5.25)

Compare	$C_{\max}$			TWC			TWT		
	HGA	ATCS	LSC	HGA	ATCS	LSC	HGA	ATCS	LSC
d_j									
Narrow	1.0	14.5	1.3	0.0	19.9	13.4	1.0	160.5	57.1
Wide	0.9	14.4	1.3	0.0	20.6	13.1	0.2	113.2	41.2
r_j									
High load	2.7	12.2	0.2	0.0	28.7	22.5	1.4	38.4	20.7
Moderate load	0.0	9.8	1.5	0.0	18.9	11.4	0.4	106.5	45.1
Low load	0.2	21.4	2.1	0.0	13.0	5.9	0.0	265.5	81.6
$\bar{p}/\bar{s}$									
5	0.2	6.8	0.5	0.0	8.4	18.9	0.0	15.9	32.9
1	0.8	19.3	1.4	0.0	18.6	10.9	0.4	61.7	16.4
1/5	1.9	17.2	1.9	0.0	33.7	10.0	1.4	332.9	98.1

We show the RD values. Instead of comparing all problem instances individually, the instances are grouped according to factor levels. For the performance measures of TWC and TWT, the HGA produces the best solutions across all the problem instances. For the problems of minimizing $C_{\max}$, HGA performs slightly worse than LSC when all the jobs are ready at the beginning. These results indicate that as the setup times become more important in the total processing time and the jobs are all ready at the beginning, LSC takes advantage of grouping the jobs with the least setup times together to reduce the $C_{\max}$ value while the HGA suffers from being trapped in a local

minimum. For the objective of TWC, the smaller the percentage of jobs that are ready at the beginning, the less improvement HGA has in comparison with ATCS and LSC. The HGA performs extremely well for problem instances with a low processing time and setup time ratio and moderate and low load levels when the goal is to minimize TWT. The improvement of at least 100 % indicates that HGA is very practical since jobs will arrive at the work center continuously in many real-world cases. More computational results can be found in [88].

Note that the GA approach taken in HGA is generalized to a four-phase scheduling framework for parallel machines proposed by Mönch [190]. This framework is applied to scheduling jobs on photolithography steppers in [189]. The framework is implemented based on the object-oriented C++ framework GAlib (cf. Wall [315]) that supports the chromosome representations, the genetic operators used, and the scheme of the GA. Furthermore, a multipopulation GA to solve multiobjective scheduling problems for parallel machines is proposed by Cochran et al. [53]. This approach avoids the separate consideration of $\mathrm{PM}_i, i = 1, 2, 3$.

5.3.4 Scheduling Jobs with Ready Times on Parallel Batch Machines

In this section, we extend the single machine batch scheduling problem discussed in Sect. 5.3.1 to a more real-world-like setting. We model diffusion and oxidation operations as scheduling problems for parallel batch-processing machines with incompatible job families. The performance measure to be minimized is TWT.

The assumptions involved in the scheduling of parallel batch-processing machines with incompatible jobs families and unequal ready times of the jobs to minimize TWT include:

1. Jobs of the same family have the same processing times on all machines.
2. The batch-processing machines are unrelated. That means that the machines have a machine-specific maximum batch size.
3. Once a machine is started, it cannot be interrupted, i.e., no preemption is allowed.
4. We assume unequal ready times of the jobs.
5. Dedications/qualifications are used on the batch machines, i.e., only a certain set of incompatible job families is allowed on each machine, mainly because of quality concerns.

We use the same notation as in Sect. 5.3.1. The only additional notation is with respect to ready times, unrelated parallel machines, and machine dedications:

1. The ready time of job j of family i is represented as r_{ij}. When the family information is not important, we use the notation r_j to refer to the ready time of job j of the n jobs.

2. The maximum batch size at machine $i = 1,\ldots,m$ is represented as $B(i)$. Clearly, we have $|B_s| \leq B(i)$ for each batch B_s that is processed on machine i.
3. The family dedication of machine i is denoted by D_i. We have $D_i \subseteq \{1,\ldots,f\}$. Because each job belongs to a family, we can derive from D_i the set of jobs that are allowed to be processed on machine i and also a set of machines that can process a certain job j. The latter set is denoted by M_j to conform with the notation introduced in Sect. 5.1.

Using the $\alpha|\beta|\gamma$ notation, the scheduling problem can be represented as

$$\text{Rm}|r_j, \text{p-batch}, \text{incompatible}, M_j|\text{TWT}, \tag{5.39}$$

where M_j refers to dedications for job j. Note that because of the unequal ready times of the jobs, it is sometimes advantageous to form non-full batches, while in other situations it is a better strategy to wait for future job arrivals to increase the fullness of the batch.

We start by presenting a MIP for problem (5.39). This formulation is similar to the MIP (5.9)–(5.17). The following index sets will be used:

$b = 1,\ldots,b_i$: index for batches on machine i,
$s = 1,\ldots,f$: family index
$j = 1,\ldots,n$: job index
$i = 1,\ldots,m$: machine index
$J_i, i = 1,\ldots,m$: set of all jobs that are allowed on machine i
$M_j, j = 1,\ldots,n$: set of all machines that are allowed to process job j

The following parameters will be used within the model:

$B(i)$: maximum batch size on machine i
d_j : due date of job j
e_{js} : $\begin{cases} 1, \text{if job } j \text{ belongs to family } s \\ 0, \text{otherwise} \end{cases}$
M : large number
p_s : processing time of family s
w_j : weight of job j
r_j : ready time of job j
D_i : set of all families that can be processed on machine i

The following decision variables are necessary:

S_{bi} : starting time of the bth batch on machine i
C_j : completion time of job j
X_{jbi} : $\begin{cases} 1, \text{if job } j \text{ is assigned to the } b\text{th batch on machine } i \\ 0, \text{otherwise} \end{cases}$

$$Y_{bis} : \begin{cases} 1, \text{ if batch } b \text{ on machine } i \text{ belongs to family } s \\ 0, \text{ otherwise} \end{cases}$$
$$T_j \ : \text{tardiness of job } j$$

The scheduling problem (5.39) may be formulated as follows:

$$\min \sum_{j=1}^{n} w_j T_j \tag{5.40}$$

subject to

$$\sum_{i \in M_j} \sum_{b=1}^{b_i} X_{jbi} = 1, \quad j = 1, \ldots, n, \tag{5.41}$$

$$\sum_{j \in J_i} X_{jbi} \leq B(i), b = 1, \ldots, b_i, i = 1, \ldots, m, \tag{5.42}$$

$$\sum_{s \in D_i} Y_{bis} = 1, b = 1, \ldots, b_i, i = 1, \ldots, m, \tag{5.43}$$

$$e_{js} X_{jbi} \leq Y_{bis}, j = 1, \ldots, n, \ i \in M_j, b = 1, \ldots, b_i, \tag{5.44}$$

$$X_{jbi} r_j \leq S_{bi}, j = 1, \ldots, n, \ i \in M_j, b = 1, \ldots, b_i, \tag{5.45}$$

$$S_{bi} + \sum_{s \in D_i} p_s e_{js} X_{jbi} \leq S_{b+1,i}, \ j = 1, \ldots, n, i \in M_j, b = 1, \ldots, b_i - 1, \tag{5.46}$$

$$S_{bi} + \sum_{s \in D_i} p_s e_{js} \leq C_j + M(1 - X_{jbi}),$$
$$j = 1, \ldots, n, \ i \in M_j, \ b = 1, \ldots, b_i, \tag{5.47}$$

$$C_j - d_j \leq T_j, \ j = 1, \ldots, n, \tag{5.48}$$

$$C_j, T_j, S_{bi} \geq 0, X_{jbi}, Y_{bis} \in \{0, 1\}, j = 1, \ldots, n, b = 1, \ldots, b_i, i = 1, \ldots, m. \tag{5.49}$$

The objective (5.40) intends to minimize the TWT value. Constraints (5.41) ensure that each job is assigned to only one batch, and constraints (5.42) do not allow more than $B(i)$ jobs to be assigned to the same batch on machine i. With constraints (5.43), we make sure that each batch belongs to a single job family, and constraints (5.44) ensure that the families of the jobs assigned to a batch match the family of the batch. Using constraints (5.45), the start time of batch b is related to the ready times of the jobs that form batch b, whereas constraints (5.46) ensure the correct start times for all subsequent batches. Constraints (5.47) make sure that the completion time of each job is not smaller than the sum of the start time of its batch and the processing time of the batch. Finally, constraints (5.48) express the tardiness for each job, and expression (5.49) represents nonnegativity and binary constraints. The MIP (5.40)–(5.49) has been solved to optimality for $m = 2$ and up to $n = 18$ in reasonable time. Hence, we have to look for efficient heuristics.

Time window decomposition approaches can be used to solve scheduling problems where not all $r_{ij} = 0$. This approach was originally proposed for the problem $\text{Pm}|r_j, s_{jk}|L_{\max}$ by Ovacik and Uzsoy [222]. We consider at time t

only those jobs that already wait or will be ready within a given time window Δt for processing on the machines of a work center. The set of jobs can be described as follows:

$$J(i,t,\Delta t) := \left\{ij | r_{ij} \leq t + \Delta t\right\}. \tag{5.50}$$

Because $|J(i,t,\Delta t)|$ can be large, we might reduce the set of jobs by using appropriate dispatching rules and consider only thresh jobs with a high priority index value in a second step. We consider the reduced job set

$$\widetilde{J}(i,t,\Delta t,\text{thresh}) := \{ij | ij \in J(i,t,\Delta t) \text{ and } \text{pos}(ij) \leq \text{thresh}\}, \tag{5.51}$$

where we use the ATC index (5.18) for evaluating the jobs of $J(i,t,\Delta t)$ and we denote by $\text{pos}(ij)$ the position of job ij with respect to this index. We consider all job combinations on the set $\widetilde{J}(i,t,\Delta t,\text{thresh})$ to form a batch for machine k. We set $c := |\widetilde{J}(i,t,\Delta t,\text{thresh})|$ for abbreviation. If $c \geq B(k)$, then we have to consider

$$\binom{c}{B(k)} + \binom{c}{B(k)-1} + \ldots + \binom{c}{1} \tag{5.52}$$

different batches for the next batch to be formed. Obviously, the computational effort for evaluating all the job combinations depends strongly on the choice of the parameters Δt and thresh. We will use the following index to assess a certain batch B_s of family i on machine k:

$$I_{B_s}(t) := \sum_{j=1}^{|B_s|} \left(\frac{w_{ij}}{p_i}\right) \exp\left(-\frac{(d_{ij} - p_i - t + (r_{B_s} - t)^+)^+}{\kappa \overline{p}}\right) \frac{|B_s|}{B(k)}, \tag{5.53}$$

where $r_{B_s} := \max_{ij \in B_s}(r_{ij})$ is the maximum ready time of the jobs in the batch. This index is proposed by Mönch et al. [203]. It is called the BATC-II index. Note that the $|B_s|/B(k)$ part of the index in Eq. (5.53) is related to the fullness of batches on machine k. The batch with the highest BATC-II index will be selected next for processing on machine k, so fuller batches are preferable. Finally, we move the time window into the future in a rolling horizon manner.

The time window decomposition heuristic (TWDH) can be described in a more algorithmic manner as follows.

Algorithm BATC-II-TWDH

1. Denote the set of all jobs by J_n. Initialize $i := 1$ to iterate over all families.
2. Determine the next available machine k with availability time $t_a(k)$ and initialize $t := t_a(k)$.
3. Determine the job $ij \in J_n$ with minimal r_{ij}. When $t < r_{ij}$, then set $t := r_{ij}$.
4. Consider a time window of length Δt with left endpoint t. Determine the sets $J(i,t,\Delta t)$ and $\widetilde{J}(i,t,\Delta t,\text{thresh})$.
5. Consider all feasible combinations of jobs to form batches based on jobs from the set $\widetilde{J}(i,t,\Delta t,\text{thresh})$. Find the batch B_s with largest index (5.53) among the batches formed in this step.

6. Update $i := i+1$. When $i > f$, then go to step 7; otherwise, set $t := t_a(k)$ and go to step 3.
7. Select the batch with the largest index found in step 5. This batch is chosen to be selected on machine k. Let B_s be the chosen batch and i the family of this batch. Update $J_n := J_n - \{ij | ij \in B_s\}$ and set $t_a(k) := \max\{t_a(k), r_{B_s}\} + p_i$.
8. When $|J_n| > 0$, then set $i := 1$ and go to step 2, otherwise stop.

Note that BATC-II-TWDH is used for several κ values within the indices (5.18) and (5.53) from a grid over the interval $(0.0, 6.5]$. Neural networks and inductive decision trees from machine learning (cf. Sect. 3.2.10) are proposed by Mönch et al. [205] to automate this step for BATC-II-TWDH. In Klemmt et al. [146], a MIP approach is used to carry out steps 3–7 of BATC-II-TWDH, i.e., solving a problem instance for problem (5.39) leads to solving a sequence of MIPs.

BATC-II-TWDH can be used to find good initial solutions for neighborhood search approaches. We continue by describing a VNS-type heuristic for problem (5.39). VNS is a neighborhood search-based metaheuristic (cf. the description in Sect. 3.2.6).

The VNS algorithm designed for the batching problem operates on the final solution representation, i.e., each job is assigned to a batch and each batch is assigned to a certain position on a machine. The proposed VNS algorithm can be summarized as follows.

Algorithm VNS for Batch Scheduling

Initialization

1. Define K different neighborhood structures N_k.
2. Generate an initial solution x using BATC-II-TWDH.
3. Set $k = 1$.

Loop

1. Repeat until stopping criterion is met.
2. Shaking: Choose randomly $x' \in N_k(x)$, where $N_k(x)$ is a neighborhood of x that is based on the neighborhood structure N_k (see the description in Sect. 3.2.6).
3. Local search: Improve x' by the batch local search (BLS) method (described below).
4. Acceptance decision: If x' is better than x, then $x := x'$ and $k := 1$; otherwise, set $k := (k \bmod K) + 1$.

The proposed BLS algorithm used for the VNS scheme consists of two different phases. During the first phase, the workload of the machines is balanced. If the last batch of the machine with the maximum completion time starts later than the completion time of another machine that is suitable for that batch, the batch is moved to that machine. This step is repeated until no batch

can be moved. During the second phase of the local search, jobs and entire batches are exchanged. The BLS algorithm can be summarized as follows.
Algorithm BLS

1. Phase 1: Balance the workload across machines.
2. Phase 2: Apply the following steps iteratively as long as improvements are obtained:
   (a) Job insert: Remove a job from a batch and insert it into another batch.
   (b) Job swap: Swap two jobs of different batches from the same family.
   (c) Batch swap: Swap two batches of the same family.

The design of the neighborhood structure for the shaking step of the VNS considers manipulations of whole batches across different machines. We define five classes of neighborhood structures:

- *splitBatch*(l): Randomly select a batch and split it into two batches. One remains in the current position, the other one is inserted into the sequence on a different machine. Repeat this step l times.
- *moveBatch*(l): Randomly select a batch from a machine and remove it. Insert it in a random position on a randomly selected machine. Repeat this step l times.
- *moveSeq*(l): Randomly select a position on a machine. Remove a sequence of at most l (l or all remaining batches) and insert this partial sequence on another machine at a randomly selected position.
- *swapBatch*(l): Randomly select two batches from different machines that are both capable of handling those batches and exchange their positions. Repeat this step l times.
- *swapSeq*(l): Randomly select two positions on different machines and exchange the batch sequences starting from that position of at most length l (l or all remaining batches).

Note that in each of these neighborhoods only splits, moves, and swaps are considered that result in a feasible solution, i.e., restrictions of job families and batch sizes are considered. The neighborhoods are applied in the order given above. They are parameterized with different l values. We use $K = 15$ and $l = 2, 3, 5$.

We expect that the solution quality depends on the number of incompatible job families, the number of jobs, the number of machines, and the release and due date settings. The ready times of the jobs are taken as instances of a $U(0, \hat{C}_{\max})$-distributed random variable, where the quantity $\hat{C}_{\max}$ is an estimate of the makespan and is determined as follows:

$$\hat{C}_{\max} := n\bar{p} / \left(0.75\ u\ \sum_{k=1}^{m} B(k) \right). \tag{5.54}$$

The quantity $\bar{p}$ denotes the average processing time and u the utilization of the machines. The factor 0.75 mimics the average fullness of a batch. The corresponding design of experiments is summarized in Table 5.5.

Table 5.5: Design of experiments for problem (5.39)

Factor	Level	Count
f	4, 6, 8	3
m	3, 4, 5	3
n_i	20, 30, 40	3
$B(k)$	$B(1)=3$, $B(2)=4$, $B(3)=6$, $B(4)=4$, $B(5)=2$	1
p_i	2 with $p=0.2$, 4 with $p=0.2$, 10 with $p=0.3$, 16 with $p=0.2$, 20 with $p=0.1$	1
w_{ij}	0.3 with $p=0.75$, 0.6 with $p=0.2$, 1 with $p=0.05$	1
r_{ij}	$\sim U(0,\hat{C}_{\max})$	1
d_{ij}	$\sim U(r_{ij}-4\bar{p}, r_{ij}+4\bar{p})$	1
D_i	$D_1=\{1,\ldots,6\}$, $D_2=\{1,3,7,8\}$, $D_3=\{1,4,6,7\}$, $D_4=\{1,4,5,8\}$, $D_5=\{1,3,4\}$	1
u	0.7, 0.8, 0.9	3
	Total factor combinations	81
	Number of problem instances per combination	2
	Total number of problem instances	162

We compare the performance of the BATC-II-TWDH with the performance of the VNS scheme. The corresponding computational results are shown in Table 5.6. The results in the third row correspond to BATC-II-TWDH. We use a time window of size $\Delta t = \bar{p}/4$. It can be shown that a too small or a too large time window is not beneficial. A similar behavior is observed for the algorithm NACH (cf. Sect. 4.6.2). The computing time for the VNS scheme is 60 s. For each problem instance, three replications with different seeds are performed, and the average TWT value is used for comparison. We show the TWT values for the VNS approach relative to the TWT values found by BATC-II-TWDH. All problem instances are grouped according to different factor levels. For example, $m=3$ means in Table 5.6 that the average TWT value for all problem instances is taken, where $m=3$ is valid. We can see that a larger number of machines and incompatible job families makes the scheduling problem harder to solve. We see from Table 5.6 that VNS clearly outperforms BATC-II-TWDH.

More computational results for problem (5.39) can be found in [146]. GAs similar to HGA from Sect. 5.3.3 are discussed for problem Pm|p-batch, incompatible|TWT by Balasubramanian et al. [18] and for Pm|r_j, p-batch, incompatible|TWT by Mönch et al. [203]. Either jobs are assigned to machines via the GA or batches are formed first by BATC-type dispatching rules and then these batches are assigned to the machines by the corresponding GA. The second approach from [203] is extended to the following bicriteria problem Pm|r_j, p-batch, incompatible|TWT, $C_{\max}$ by Reichelt and Mönch [259].

Table 5.6: Computational results for problem (5.39)

Compare	m			n_i			f		
	3	4	5	20	30	40	4	6	8
TWDH	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
VNS	0.89	0.92	0.93	0.92	0.91	0.93	0.91	0.89	0.94

A different GA for $\text{Pm}|r_j, \text{p-batch}, \text{incompatible}|\text{TWT}$ is proposed by Chiang et al. [48] for the problem studied in [203].

5.3.5 Scheduling Problems for Parallel Machines with Auxiliary Resources

Auxiliary resources in wafer fabs are typically related to steppers in the photolithography work area (cf. Sect. 2.2.2). As ICs are built by repeatedly constructing layers with desired properties on the surface of the wafers, stepper processing depends on both the layer of the wafer that is associated with the current process step of the job and the correct reticle being available at the same time. The auxiliary resource problem arises because every layer of each product can require its own unique reticle, as reticle requirements are typically both product- and layer-dependent. Considering the fact that a reticle must be on the machine for the duration of processing, the relatively small number of reticles present in a wafer fab further complicates the parallel machine scheduling problem associated with steppers.

In this section, we start by discussing a simplified model problem. The assumptions for scheduling jobs on steppers within this model problem are the following:

1. The parallel machines are identical.
2. Once a machine is started, it cannot be interrupted, i.e., no preemption is allowed.
3. We assume unequal ready times of the jobs.
4. An appropriate reticle is necessary to process a job on a machine.

We consider a TWC objective. Note that this measure takes different weights of the jobs into account. In the context of stepper scheduling, it might be preferable to schedule jobs next with many already completed layers. Therefore, individual job weights can be derived based on this information. On the other hand, a small TWC value leads to a small CT value of the jobs. Using the $\alpha|\beta|\gamma$ notation, the scheduling problem can be represented as

$$\text{Pm}|r_j, \text{aux}|\text{TWC}. \tag{5.55}$$

Problem (5.55) is NP-hard because the problem $1|r_j|\mathrm{TC}$ is NP-hard (see Brucker [34]). Hence, we have to look for efficient heuristics.

In the following, we present a MIP formulation for problem (5.55) that is due to Cakici and Mason [42]. It is similar to the formulation given for problem (5.25). We use the following indices and index sets in the corresponding MIP model:

$i = 1,\ldots,m$: machine index
$j = 0,\ldots,n$: job index
$l \in L$: set of layers
J_l : set of jobs that require layer $l \in L$ for processing

Note that the job $j = 0$ is a dummy job that is used on each machine. We have $p_0 = r_0 = 0$. The following parameters will be used within the model:

r_j : ready time of job j
p_j : processing time of job j
w_j : weight of job j
M : large number

The following decision variables are used within the MIP:

$$X_{ij} : \begin{cases} 1, \text{ if job } i \text{ immediately precedes job } j \text{ on the same machine} \\ 0, \text{ otherwise} \end{cases}$$

C_j : completion time of job j

$$e_{ij} : \begin{cases} 1, \text{ if job } i \in J_l \text{ is completed before job } j \in J_l,\ i \neq j \text{ starts its processing} \\ 0, \text{ otherwise} \end{cases}$$

The scheduling problem can be formulated as follows:

$$\min \sum_{j=1}^{n} w_j C_j \tag{5.56}$$

subject to:

$$\sum_{i=0}^{n} X_{ij} = 1, \quad j = 1,\ldots,n, \tag{5.57}$$

$$\sum_{j \neq i} X_{ij} \leq 1, \quad i = 1,\ldots,n, \tag{5.58}$$

$$\sum_{k=1}^{n} X_{0k} \leq m, \tag{5.59}$$

$$X_{ij} \leq \sum_{\substack{k=0 \\ k \neq i, k \neq j}}^{n} X_{ki}, \quad i = 1,\ldots,n,\ j = 1,\ldots,n,\ i \neq j, \tag{5.60}$$

$$C_j + p_k \leq C_k + M(1 - X_{jk}), \quad j = 0,\ldots,n,\ k = 1,\ldots,n,\ j \neq k, \tag{5.61}$$

$$C_0 = 0, \tag{5.62}$$

$$C_j \geq r_j + \sum_{k=0}^{n} (p_j + (r_k + p_k - r_j)^+) X_{kj}, \quad j = 1, \ldots, n, \tag{5.63}$$

$$C_j + p_k e_{jk} \leq C_k + M(1 - e_{jk}),$$
$$j = 1, \ldots, n,\ k = 1, \ldots, n,\ j,k \in J_l,\ l \in L,\ j \neq k, \tag{5.64}$$

$$1 \leq e_{ij} + e_{ji},\ i = 1, \ldots, n,\ j = 1, \ldots, n,\ i,j \in J_l,\ l \in L,\ i \neq j, \tag{5.65}$$

$$X_{ij} \in \{0,1\},\ i = 0, \ldots, n,\ j = 1, \ldots, n,\ i \neq j, \tag{5.66}$$

$$e_{ij} \in \{0,1\},\ i = 1, \ldots, n,\ j = 1, \ldots, n,\ i,j \in J_l,\ l \in L,\ i \neq j. \tag{5.67}$$

The objective is to minimize the TWC performance measure. Constraints (5.57) ensure that each job is assigned to exactly one machine and has exactly one predecessor. The dummy job 0 is used within $X_{0k}, k \in \{1, \ldots, n\}$. Constraints (5.58) model that each job except for the dummy job has at maximum one successor. The number of successors of the dummy jobs is at most m because of constraint (5.59). When a given job i is processed on a machine, then a predecessor must exist on the same machine. This is modeled by constraints (5.60). When a job k is assigned to a machine immediately after job j, i.e., when $X_{jk} = 1$, its completion time has to be greater than the sum of the completion time of j and the processing time of k. This is expressed by constraints (5.61). In case of $X_{jk} = 0$, constraints (5.61) are also fulfilled because of the big M. Constraint (5.62) sets the completion time of the dummy job to zero. Constraints (5.63) ensure that the completion time of a job is always larger than the sum of its ready time and processing time. Constraints (5.64) and (5.65) ensure that if two jobs require the same reticle, one of the jobs has to complete its processing before the other job starts its processing. Finally, constraints (5.66) and constraints (5.67) model the fact that the main decision variables X_{ij} and e_{ij} only take on binary values.

The MIP formulation (5.56)–(5.67) has been used to solve problem instances up to $m = 2$ and $n = 15$ and up two six layers within reasonable time optimally. Therefore, again we can use the MIP formulation (5.56)–(5.67) to assess whether heuristics are correctly implemented and to get some insights into the potential performance of the proposed heuristics.

Next, we describe a heuristic for problem (5.55) that is proposed in [42]. The heuristic consists of two phases. In the first phase, the heuristic determines a feasible solution using dispatching rules within a list scheduling approach that takes the auxiliary resources into account. Then, in a second step, this solution is improved by tabu search. The algorithm of the first phase is summarized as follows.

Algorithm Construction Heuristic (CH)

1. Determine the current values of t and φ, where t is the current time and φ represents a machine that becomes idle at time t. The quantity Φ is the set of unscheduled jobs, and Θ is the set of candidate jobs. Initially, set $t = 0$, $\varphi = 1$, $\Phi := \{1, \ldots, n\}$, and finally $\Theta := \Phi$.
2. For each $j \in \Theta$ determine the index

$$I_j(t) := \begin{cases} 0, \text{ if } t < r_j \\ w_j/p_j, \text{ otherwise} \end{cases}. \tag{5.68}$$

If for all $j \in \Theta$ the condition $t < r_j$ is valid, then use the index

$$I_j(t) := w_j/(p_j + r_j - t) \tag{5.69}$$

instead of index (5.68). The job k with the highest index value will be chosen as a candidate to be assigned to machine φ.

3. When k can be feasibly assigned to φ at $\max(t, r_k)$, i.e., when an appropriate reticle is available, then go to step 4; otherwise, go to step 5.
4. Assign k to φ at time $\max(t, r_k)$. Update $\Phi := \Phi - \{k\}$ and go to step 6.
5. Update $\Theta := \Theta - \{k\}$. When $\Theta = \emptyset$ holds, then go to step 6; otherwise, go to step 2.
6. The value of t has to be updated to the smallest future point of time in which a feasible assignment of a job to a machine is possible, i.e., when a new job is ready for processing or when a machine becomes available. Set $\Theta := \Phi$. If $\Theta \neq \emptyset$, go to step 2; otherwise, stop.

Note that the indices (5.68) and (5.69) are influenced by the fact that the WSPT dispatching rule is optimal for the scheduling problem $1||\text{TWC}$ (cf. Pinedo [240]). The algorithm of the second phase is a tabu search variant. A neighborhood of a feasible solution x of problem (5.55) consists of all pairwise job swaps. After a swap of two jobs, a feasible schedule is obtained by a placement heuristic [42]. The resultant two-phase heuristic is called CH+TS.

Next, we discuss the results from computational experiments. A variety of problem instances are examined to evaluate the efficacy of CH and CH+TS. First, two different levels are considered for the number of layer types, sampling from a discrete uniform distribution $\text{DU}[1, v]$ with $v \in \{3, 6\}$. Experiments are also performed for both two and three machines operating in parallel. For each job, an integer processing time is generated from $\text{DU}[45, 75]$. One-half of the job ready times are generated from $\text{DU}[1, 360]$, while the second half of the jobs have $r_j = 0$. Further, job weights are selected according to $\text{DU}[1, 20]$. The design of experiments is summarized in Table 5.6.

Table 5.6: Design of experiments for problem (5.55)

Factor	Level	Count
m	2, 3	2
n	10, 15	2
p_j	$\sim \text{DU}[45, 75]$	1
w_j	$\sim \text{DU}[1, 20]$	1
r_j	50 % $\sim \text{DU}[1, 360]$ and 50 % $r_j \equiv 0$	1
l	$\sim \text{DU}[1, v]$, $v \in \{2, 3\}$	2
	Total factor combinations	8
	Number of problem instances per combination	10
	Total number of problem instances	80

We compare the TWC values obtained by CH and CH+TS, respectively with the optimal TWC results from the MIP formulation (5.56)–(5.67) by presenting the ratios in Table 5.7.

Table 5.7: Computational results for problem (5.55)

Compare	CH	CH+TS
m		
2	1.012	1.007
3	1.022	1.008
n		
10	1.014	1.004
15	1.021	1.012
l		
1–3	1.021	1.011
1–6	1.014	1.005
Overall	1.017	1.008

We see from Table 5.7 that CH and CH+TS perform well. CH+TS was able to determine the optimal solution for 76 problem instances. CH+TS slightly outperforms CH.

A couple of real-world conditions are not included in problem (5.55). The only additional constraint compared to other parallel machine scheduling problems is the limited number of reticles. In real-world situations, however, many more constraints appear. For example, the steppers often have to be modeled as unrelated parallel machines due to their different generations. Steppers generally have a local reticle stocker that can hold only a certain small number of reticles. Unfortunately, wafer fabs processing a wide variety of products may need to have a large number of different reticles available for use at any given time. Therefore, central stockers located near the stepper work center are sized with sufficient capacity to manage all reticles required in the facility. Because the pattern on each reticle is important in the manufacturing process, moving reticles from one place to another in a wafer fab is not trivial. Loading a reticle onto a machine, called a machine setup, must be done very carefully. In some wafer fabs, every time a reticle is moved, it must be inspected either to ensure that the original pattern is intact or to be sure that there are no unwanted particles of dust or vapor on the surface. Therefore, jobs that require the same reticle are often processed in a consecutive manner on the same machine to avoid frequent reticle changes.

Send-ahead wafers for steppers are also common to perform quality measurements with a single wafer. In this situation, a single wafer out of a certain job is processed instead of the entire job. Because of the required reticle change, a small number of send-ahead wafers is desirable. In order to process a job on a stepper, the job has to be ready, the stepper has to be idle, and finally, the reticle has to be inspected and set up on the idle stepper. In the

remainder of this section, we will briefly discuss approaches from the literature that address some of these constraints.

Akçali and Uzsoy [4] studied the shift-scheduling problem of a photolithography station and created a policy that efficiently distributes the workload across all machines for an entire shift. In their capacity allocation routine (CAR), all waiting jobs are categorized based on the process step they await, and all machines are categorized based on how much work has already been allocated to them. With this information, the CAR incorporates operational and auxiliary resource constraints and uses a greedy heuristic to assign the largest group of waiting jobs to the machine with the highest amount of available capacity and preferably, the required reticle. In the second step, a sequencing problem for jobs on each machine is solved to create a detailed schedule for an entire shift. A similar approach is discussed by Klemmt et al. [147].

A network flow model is proposed by Dìaz et al. [68] to efficiently load reticles onto machines based on the jobs that are soon available to be processed. The current location of all reticles and information of jobs that are waiting for processing or that will be ready for processing soon are used as inputs for the model. The objective is to minimize the number of setups and inspections over the next shift.

The assignments of reticles can be used to determine schedules by list scheduling using the following variant of the ATCS dispatching rule:

$$I_{lj}(t) := \frac{w_j}{p_j} \exp\left(-\frac{(d_j - p_j - t)^+}{\kappa_1 \bar{p}}\right) \exp\left(-\frac{s_{lj}}{\kappa_2 \bar{s}} - \frac{\max(r_j - t, a_{lj}, 0)}{\kappa_3 \bar{p}}\right), \quad (5.70)$$

where we denote by a_{lj} the time that is required to make the reticle for job j ready on a machine that has just finished processing job l and κ_3 is a third look-ahead parameter. Index (5.70) has the advantage compared to index (5.68) that the availability of the reticle and the required setups are more directly taken into account. Computational results using a simulation environment similar to that described in Sect. 3.3.2 can be found in [68].

Operational problems related to send-ahead wafers are discussed by Akçali et al. [5]. In this paper, the main performance measure is CT. A GA for scheduling steppers that takes send-ahead wafers and reticle constraints into account is described by Mönch [189]. A combined objective function is taken that considers TP, TWT, and the number of send-ahead wafers.

Finally, we discuss a different example from the back-end area provided by Kempf et al. [137]. A single burn-in oven is considered. The auxiliary resources are given by load boards (cf. Sect. 2.2.2). A burn-in oven is a batch-processing machine. The oven capacity is given by the number of boards that an oven can carry. The size s of a single job is given by the number of boards that are required to process the job. Because the board type required by

a job depends on the packaging of the circuits, different jobs may require different load boards. Only a limited number of boards are available for each board type. The concept of incompatible job families as already described for scheduling problems (5.2) and (5.39) can be used to model the fact that certain jobs have to be kept in the oven for the same amount of time. Only jobs of the same family can be batched together. Jobs of the same family may use different types of boards. Using the $\alpha|\beta|\gamma$ notation, the scheduling problem can be modeled as

$$1|\text{p-batch}, \text{incompatible}, s, \text{aux}|\text{TC}, \tag{5.71}$$

where s refers to different sizes of the jobs. Because this problem is NP-hard [137], heuristics are suggested that also take the load board constraint into account. Because of the capacity constraints of the oven, the heuristics are inspired either by full-load strategies or by first-fit decreasing (FFD) heuristics for bin packing (see Dowsland and Dowsland [70]). Furthermore, a heuristic that is based on a relaxation of an IP formulation for problem (5.71) is also discussed.

5.3.6 Multiple Orders per Job Scheduling Problems

In this section, we discuss another class of scheduling problems that can be found in 300-mm wafer fabs. A FOUP is the standard unit of job transfer between machines in a wafer fab. FOUPs are expensive. More importantly, a large number of FOUPs can cause a congested AMHS. Therefore, the number of FOUPs is limited and is often a restriction. Because of the combination of decreased line width and increased area per wafer, fewer wafers are required to fill the orders for ICs than was the case before. Therefore, there is quite often a need to group orders of different customers into one FOUP (cf. the description in Sect. 2.2.3). We call the resulting entity a job to conform with scheduling literature. The jobs have to be scheduled on the different machines within a wafer fab when such a grouping decision for orders is made. Consequently, the resulting class of problems are called MOJ scheduling problems.

We consider a set of orders $O := \{1, \ldots, N\}$. Each order $o \in O$ has a size s_o, measured in number of wafers, and a weight w_o that is used to model the importance of order o. Let K denote the capacity of a FOUP measured in wafers. For simplicity, we assume that $s_o \leq K$ for all $o \in O$. There are F FOUPs available.

The assumptions for scheduling MOJ on a single machine are as follows:

1. All the orders are available at time $t = 0$.
2. The orders are of the same product type, i.e., all the orders can be grouped together.
3. Once a machine is started, it cannot be interrupted, i.e., no preemption is allowed.

4. All the orders in a job are processed together. The completion time of the job determines the completion time of all the orders that form the job.
5. There are two different processing modes. In the lot processing mode, the job processing time is just the wafer processing time ρ. The processing time of job j is given as $p_j = \rho \sum_{o \in j} s_o$ in the single item processing mode.

Using the $\alpha|\beta|\gamma$ notation, the two scheduling problems can be represented as follows. In the lot processing case, we obtain

$$1|\text{moj(lot)}|\text{TWC}, \tag{5.72}$$

where we have $\text{TWC} := \sum_{o \in O} w_o C_o$. We denote by C_o the completion time of o. The corresponding scheduling problem in the single item case can be represented as

$$1|\text{moj(item)}|\text{TWC}. \tag{5.73}$$

It is proved by Mason and Chen [171] that problems (5.72) and (5.73) are NP-hard. Hence, we have to look for efficient heuristics.

We present a MIP formulation from Mason et al. [174] that covers both problems (5.72) and (5.73). We use the following indices and index sets in the corresponding MIP model:

$j = 1, \ldots, F$: job index
$o = 1, \ldots, N$: order index
O : set of all orders

The following parameters will be used within the model:

p_j : processing time of job j in the lot processing case
w_o : weight of order o
s_o : size of order o
K : capacity of the FOUP
M : large number

The following decision variables are used within the MIP:

X_{oj} : $\begin{cases} 1, \text{ if order } o \text{ is assigned to job } j \\ 0, \text{ otherwise} \end{cases}$
C_j : completion time of job j
p_j : processing time of job j in the single item processing case
δ_o : completion time of order o

Note that we have $p_j = \rho$ in the lot processing case and $p_j = \rho \sum_{o \in O} s_o X_{oj}$ in the single item case. Therefore, depending on the MOJ environment, p_j is a parameter or a decision variable. Furthermore, without loss of generality, we assume that there is a given sequence of the FOUPs. Now the scheduling problems (5.72) and (5.73) can be formulated as follows:

$$\min \sum_{o \in O} w_o \delta_o \tag{5.74}$$

subject to:

$$\sum_{j=1}^{F} X_{oj} = 1, \quad o = 1,\ldots,N, \tag{5.75}$$

$$\sum_{o\in O} s_o X_{oj} \le K, \quad j = 1,\ldots,F, \tag{5.76}$$

$$C_j - M(1 - X_{oj}) \le \delta_o, \quad j = 1,\ldots,F,\ o = 1,\ldots,N, \tag{5.77}$$

$$p_j \le C_j, \quad j = 1,\ldots,F, \tag{5.78}$$

$$C_j + p_{j+1} \le C_{j+1}, \quad j = 1,\ldots,F-1, \tag{5.79}$$

$$\delta_o \ge 0,\ C_j \ge 0,\ X_{oj} \in \{0,1\}, \quad j = 1,\ldots,F,\ o = 1,\ldots,N. \tag{5.80}$$

The objective is to minimize the TWC performance measure. Constraints (5.75) model that each order is assigned to exactly one job. The capacity restrictions for each FOUP are represented by constraints (5.76). The completion time of o is calculated using constraints (5.77). Constraints (5.78) ensure that each job spends at least its processing time in the system. Constraints (5.79) are used to properly sequence the jobs according to their indices. Finally, constraints (5.80) model the fact that the main decision variables X_{oj} take only binary values and the completion times of orders and jobs are nonnegative. The model is completed by adding constraints

$$p_j = \rho \sum_{o\in O} s_o X_{oj} \tag{5.81}$$

to the model in the single item case. We set $p_j = \rho$ in the lot processing case. Note that problem instances up to 20 orders have been solved to optimality in reasonable time using the MIP model (5.74)–(5.81). Therefore, we continue with the discussion of heuristics to solve large-size problem instances.

We describe a heuristic that is influenced by bin packing algorithms (cf. Dowsland and Dowsland [70] for bin packing-related information). The heuristic consists of a first phase where orders are sequenced by a dispatching rule. Then, the jobs are formed based on a bin packing-type heuristic. The heuristic for the lot processing case can be summarized as follows.

Algorithm MOJ Lot Processing

1. Sort the orders from O with respect to nonincreasing values of the weighted smallest size (WSS) index
$$I_o := w_o / s_o. \tag{5.82}$$
2. Let Φ be the set of orders that are not assigned to a job. Set initially $\Phi := \{1,\ldots,N\}$. Sort Φ according to the order sequence obtained in step 1. The current job is denoted by j_{curr}. Set initially $j_{\text{curr}} := 1$.

3. Form j_{curr} by going through Φ starting with the order with the largest WSS index and taking the capacity constraints of the job into account. Update Φ when an order is placed in j_{curr}.
4. If $j_{\text{curr}} = F$ or $\Phi = \emptyset$, then stop; otherwise, set $j_{\text{curr}} := j_{\text{curr}} + 1$. Go to step 3.

Note that using the WSS rule in step 1 is motivated by the fact that the WSPT dispatching rule leads to an optimal schedule for 1||TWC. The FFD heuristic is used for the job formation phase. The job formation part of the algorithm tries to form jobs that use all of the available capacity of the FOUPs. Because of the WSS sorting, the jobs formed early in the process contain more orders with small s_o, and therefore they are often more fully loaded. Solutions of problem (5.72) with small TWC value tend to use a small number of FOUPs.

We now describe a different algorithm for problem (5.73) because the processing times of the jobs are determined by the actual schedule.

Algorithm MOJ Single Item Processing

1. Sort the orders from O with respect to nonincreasing values of the WSS index.
2. Let Φ be the set of orders that are not assigned to a job. Set initially $\Phi := \{1,\ldots,N\}$. Sort Φ according to the order sequence obtained in step 1. Set initially $F_{\text{curr}} := F$. Let j_{curr} be the current job. Initialize it with $j_{\text{curr}} := F$.
3. Determine the sum of the sizes of the jobs from Φ. Divide this sum by F_{curr}. This gives the expected average job size for each remaining job.
4. Start assigning orders from Φ to j_{curr} starting with the order with smallest WSS index taking the job capacity and the expected average job size from step 3 into account. Update Φ when an order is inserted into j_{curr}. Then, update F_{curr}.
5. When $j_{\text{curr}} = 1$ or $\Phi = \emptyset$, then stop; otherwise, set $j_{\text{curr}} := j_{\text{curr}} - 1$ and go to step 3.

The job formation part of the algorithm tries to balance the sizes of the formed jobs. The job size is calculated as the sum of the sizes of the orders that form the job. Later jobs are being more fully loaded than earlier jobs in order to avoid unnecessary delay times in solutions that have small TWC values. Therefore, we start the job formation with job j_F and assign orders with small WSS index to the later jobs. Solutions of problem (5.73) with small TWC value tend to use as many FOUPs as possible.

The two algorithms decompose the scheduling problems into an order-sequencing subproblem and a job-formation subproblem and solves them in this sequence. Now we take the opposite perspective and decompose the scheduling problems into a job-formation problem and a job-sequencing problem. Based on Proposition 5.3, it turns out that the second subproblem can be solved in a straightforward manner.

Proposition 5.3 *When the jobs are formed in the lot processing environment, then an optimal solution of the job-sequencing subproblem can be obtained by simply sequencing the jobs in nonincreasing order with respect to* $\sum_{o\in j} w_o$. *Moreover, when the job formation decision is made in the item processing mode, then the optimal job sequence can be determined by sorting the jobs with respect to nonincreasing values of* $\sum_{o\in j} w_o / \sum_{o\in j} s_o$.

Proof. The proof is based on the fact that the WSPT dispatching rule leads to an optimal schedule for $1||\text{TWC}$. In the lot processing case, all jobs have the same processing time. Hence, instead of considering the WSPT index given by expression (4.8), it is enough to sort the jobs with respect to $\sum_{o\in j} w_o$. Furthermore, we have $p_j = \rho \sum_{o\in j} s_o$ in the single item processing mode. Therefore, it is sufficient to sort the jobs using $\sum_{o\in j} w_o / \sum_{o\in j} s_o$. □

Grouping GAs (GGAs) (see Falkenauer [78]) are proposed by Sobeyko and Mönch [286] to tackle the job formation phase. A GGA uses an encoding scheme based on genes that represent the groups because groups are the meaningful building blocks of a solution. This means for problem (5.72) and problem (5.73) that jobs are the groups to be formed.

We expect that the performance of the algorithms depend on the size of the orders and the FOUP capacity. Therefore, we consider 40 small-size problem instances for both the lot processing and the single item processing mode. The corresponding design of experiments can be found in Table 5.8.

Table 5.8: Design of experiments for problem (5.72) and (5.73)

Factor	Level	Count
N	10	1
ρ	10	1
w_o	$\sim \text{DU}[1,15]$	1
s_o	$\sim \text{DU}[\frac{\nu-1}{2}, \frac{3\nu+1}{2}]$, $\nu \in \{3,5\}$	2
K	$12\beta+1$, where $\beta \in \{1,2\}$	2
F	$\lceil N\nu/(12\beta)\rceil + 1$	1
	Total factor combinations	4
	Number of problem instances per combination	10
	Total number of problem instances	40

We assess the performance of the two heuristics by solving the problem instances to optimality using the MIP formulation (5.74)–(5.81). The ratio $R(H) := \text{TWC}(H)/\text{TWC}(\text{MIP})$ is considered, where we denote by $\text{TWC}(H)$ the TWC value for a problem instance using one of the heuristics and by $\text{TWC}(\text{MIP})$ the corresponding TWC value obtained by the MIP formulation. It turns out that in the lot processing mode, we obtain $R(H)$ values between 1.005 and 1.020, whereas the corresponding values are between 1.010 and 1.020 in the single item processing mode. Note that the GGA from [286] almost always finds the optimal solution for problem instances with $N = 10$.

The methods for this single machine MOJ scheduling problem without ready times of the orders were extended to single machine MOJ scheduling problems with ready times by Qu and Mason [254], to parallel machine situations by Jia and Mason [129], to flow shops by Laub et al. [149], and finally to job shops by Jampani and Mason [126] and by Jampani et al. [127].

5.4 Full Factory Scheduling

In this section, we consider scheduling approaches for full wafer fabs, which can be modeled from a scheduling point of view as complex job shops (cf. Sect. 2.2.3). We discuss the disjunctive graph representation of complex job shop scheduling problems. The shifting bottleneck heuristic (SBH) is introduced and subproblem solution procedures (SSPs) are discussed. Moreover, a distributed variant of this scheduling heuristic is described. Rolling horizon approaches and their simulation-based performance assessment are discussed. Finally, a multicriteria extension is presented.

5.4.1 Motivation and Problem Statement

The assumption for scheduling jobs in complex job shops are:

1. The job shop consists of groups of unrelated parallel machines.
2. Once a machine is started, it cannot be interrupted, i.e., no preemption is allowed.
3. The jobs j have unequal ready times.
4. Each job j has a process flow $O(j) := \{O_{j1}, \ldots, O_{jn_j}\}$, where we denote by O_{jk} the kth operation of j. Totally, we have n_j operations.
5. Some of the machines have sequence-dependent setup times.
6. There are batch machines in the job shop. Only jobs of the same job family can be batched together.
7. Reentrant process flows are considered.

Using the $\alpha|\beta|\gamma$ notation scheme, the scheduling problem to be solved can be described as follows:

$$\text{FJm}|\text{p-batch}, \text{incompatible}, s_{jk}, r_j, \text{recrc}|\text{TWT}. \tag{5.83}$$

We use the completion time C_j of job j with respect to the entire wafer fab and the corresponding due date d_j to calculate the TWT value. The scheduling problem (5.83) is NP-hard because the single machine scheduling problem $1||\text{TWT}$ that is known to be NP-hard (see Lawler [151]) is a special case of it. Hence, we have to look for efficient heuristics. In the following, we will study mainly decomposition heuristics. These heuristics are based on the concept of disjunctive graphs (cf. Brucker and Knust [35]) that will be described in Sect. 5.4.2.

Until recently, full factory scheduling methods seemed to be too computationally costly in comparison to dispatching methods. However, with the recent dramatic increase in computer efficiency, full fab scheduling methods have become more competitive.

Decomposition heuristics for complex job shops have been studied very intensively over the last ten years starting with the pioneering work of Uzsoy et al. [305] for a test facility. A good documentation of attempts to solve the problem

$$\text{FJm}|\text{p-batch}, s_{jk}, r_j, \text{recrc}|L_{\max} \tag{5.84}$$

for test facilities can be found in Ovacik and Uzsoy [223]. Decomposition approaches for problem (5.83) were studied for the first time by Mason *et al.* [172]. In this section, we will mainly discuss this approach and some extensions of it. We will also show how a simulation-based performance assessment can be carried out for this type of full factory scheduling approach.

5.4.2 Disjunctive Graph Representation for Job Shop Problems

Job shop scheduling problems can be represented by disjunctive graphs (cf. [1, 35, 223]). Many papers have discussed the use of disjunctive graphs for the scheduling problem $\text{Jm}||C_{\max}$. Therefore, we start describing the basic properties for disjunctive graphs with respect to this problem, but later we will add several extensions needed to adequately model wafer fabs. A disjunctive graph is given by the triple

$$G = (V, E_c, E_d). \tag{5.85}$$

We denote by V the set of nodes of the graph. E_c is used for the set of conjunctive arcs, whereas E_d denotes the set of disjunctive arcs. We set $E := E_c \cup E_d$. Furthermore, we define an incidence mapping

$$\omega : E \to V \times V. \tag{5.86}$$

A node $v \in V$ represents an operation of a job on a machine. We use the notation $\langle i, j \rangle$ for a $v \in V$ that represents an operation of job j on one of the machines of machine group i. We may have several operations of a job on machines of one machine group due to reentrant flows of the jobs within a wafer fab. In this case, we use the notation $\langle i, ja \rangle, \langle i, jb \rangle, \ldots$ to distinguish between these operations. Furthermore, the node set contains an artificial starting node s and an artificial end node e. We need also an end node V_j, $j = 1, \ldots, n$ for each job. The quantity n is the number of considered jobs. The end nodes represent the due dates d_j of each job j.

Conjunctive arcs are defined as directed arcs (u, v) from $u \in V$ to $v \in V$. They indicate a direct precedence relationship between these nodes. This relationship is either based on the process flow of the job or on a scheduling

decision. In the first case, u and v belong typically to different machine groups. In the latter case, u and v are associated with the same machine group.

Disjunctive arcs are defined as undirected arcs $e := \{u, v\}$ between nodes $u, v \in V$ associated with operations that are performed on machines of the same machine group. A disjunctive arc e represents the following situations:

1. There is no directed arc between u and v.
2. Alternatively, there exists either the directed arc (u, v) or the directed arc (v, u).

In the first situation, the operations that are associated with u and v are not performed in a consecutive manner on the same machine. There is such a consecutive processing in the second situation. Therefore, disjunctive arcs are used to model scheduling decisions to be made. Some of the disjunctive arcs will be eliminated, and some of them will be changed into conjunctive arcs in the course of the scheduling algorithms. Figure 5.5 shows a disjunctive graph for a job shop that consists of four machines $\{1, 2, 3, 4\}$ and three jobs $\{1, 2, 3\}$. Dashed arcs are used to represent disjunctive arcs. A schedule can be determined by eliminating disjunctive arcs. That means that we have to solve a scheduling problem for each machine group. Then, we replace the disjunctive arcs that are associated with scheduling decisions for each single machine of the machine group by a conjunctive arc chain, and we remove the other arcs.

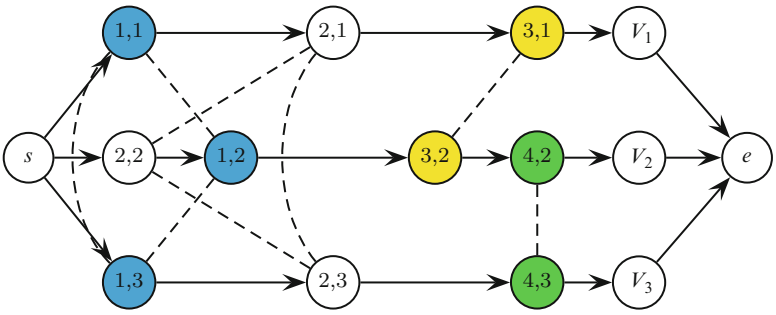

Figure 5.5: Disjunctive graph representation for four jobs on three machines

Such a chain is assigned to each of the machines of the machine groups. An arc chain connects nodes in such a way that the direct precedence relationship among the operations due to the schedule is translated into the graph. Of course, we have to make sure that the resulting graph does not contain any cycle, because otherwise the schedule is not feasible. We show an example of a partial schedule based on the example used for Fig. 5.5 in Fig. 5.6. The schedule for machine 2 is implemented within G. The operation

that corresponds to $\langle 2,1\rangle$ is sequenced before the operation that is associated with $\langle 2,3\rangle$. Similarly, the operation associated with $\langle 2,3\rangle$ is sequenced before the operation belonging to $\langle 2,2\rangle$.

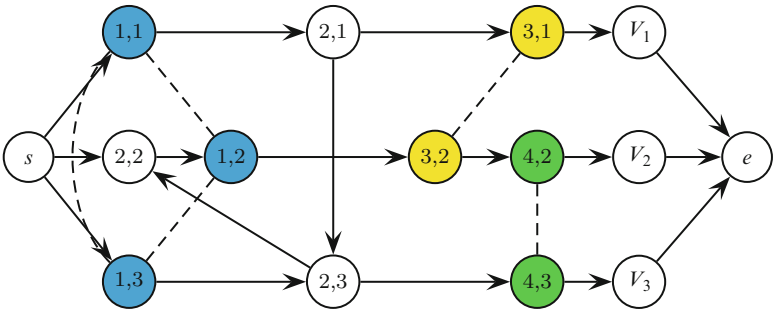

Figure 5.6: Representation of a partial schedule

We complete the description of the main concepts of disjunctive graphs by assigning weights to the arcs. These weights are necessary to represent the time delay caused by the processing of previous operations of the different jobs. We denote the weight of an arc (u,v) as $p(u,v)$. We differentiate between several cases:

- Each conjunctive arc (u,v) with $u \neq s$ and $u \neq V_j$ gets the processing time of the operation that is associated with the node u. Because u is nonartificial, this setting is well defined.
- The ready time of the job that is associated with node u is assigned to arcs of the form (s,u).
- The weight 0 is assigned to arcs of the form (V_j,e).
- Because undirected arcs do not cause any time delay, a weight with value 0 is assigned to them.

Note that all the conjunctive arcs of the form (u,v) with nodes $u \neq s$ have the same weight. Therefore, we simplify the notation and call the corresponding weight $p(u)$.

After introducing weights for the arcs of G we can see the main advantage of modeling job shop scheduling problems by disjunctive graphs. G allows us to evaluate the influence of single scheduling decisions on the entire job shop. In order to do so, we have to determine ready times and due dates for the individual operations of the jobs on the machines of the machine groups by performing longest path calculations. A path in G is defined as a sequence of directed arcs $e_1,\ldots,e_k$ such that a sequence of nodes $v_0,\ldots,v_k$ exists, such that $\omega(e_j) = (v_{j-1},v_j)$. The length of a path is given as the sum of the weights of the arcs that form the path. Shortest and also longest paths can be efficiently calculated using Dijkstra's algorithm (cf. Brucker and Knust [35]). This type of calculation is also used in the critical path method in project scheduling.

The ready time of an operation associated with node u is denoted by $r(u)$. It is the minimum release time of the operation associated with u when we assume that all operations associated with nodes that precede u are started at the ready times that corresponds to these nodes. This is the longest path from s to u. The following recursion is used to set the ready times for a node $v \in V - \{s\}$:

Initial condition:

$$r(s) := 0, \tag{5.87}$$

Recursive relation:

$$r(v) := \max\{r(u) + p(u,v) | v \in V, (u,v) \in E_c\} \tag{5.88}$$

if the operation that is associated with v has to be performed on a machine of an unscheduled machine group. If u is scheduled on a machine with availability time r_a, then we have to set

$$r(v) := \max\{r_a, r(v)\}. \tag{5.89}$$

Based on the ready times, we are able to introduce completion times. The completion time $c(u)$ of an node $u \in V - \{e,s\}$ is defined by

$$c(u) := r(u) + p(u). \tag{5.90}$$

For e, we define furthermore

$$c(e) := r(e). \tag{5.91}$$

The due date of the operation that is associated with node $u \in V - \{e,s\}$ can be determined by the following recursion:

Initial condition:

$$d(e) := r(e), \tag{5.92}$$

Recursive relation:

$$d(u) := \min\{d(v) - p(v) | v \in V, (u,v) \in E_c\}. \tag{5.93}$$

By solving the recursion (5.92), (5.93), it can be shown that $d(u) - p(u)$ for $u \in V - \{s,e\}$ is the difference between $c(e)$ and the length of the longest path from u to e. The length of the longest path represents the sum of the processing times and the waiting times of the remaining operations of the job that is associated with u. Therefore, $d(u) - p(u)$ can be considered as the latest time when the operation associated with node u has to start to avoid an increased value of $C_{\max}$ of the jobs.

Sophisticated production conditions like parallel machines, sequence-dependent setup times, batch machines, and reentrant flows are important

in complex job shops. Therefore, we describe how these characteristics can be modeled by extensions of the disjunctive graph G for $\mathbf{Jm}||C_{\max}$ following [172, 225].

Parallel machines are included in a natural way in G. Only those nodes are connected after a scheduling decision that represent jobs to be processed on a specific machine. We obtain arc chains as already described by the implementation of scheduling decisions within G.

Sequence-dependent setup times can be incorporated into G only after the scheduling decisions for the jobs on a machine group are made. Therefore, we increase the weight of the corresponding arc by the setup time. The initial setup of a machine is included into the weights of the outgoing arcs of the node that corresponds to the first scheduled operation on the machine.

An appropriate modeling of batch machines is more sophisticated [172, 223]. A scheduling decision for a batch machine includes three types of decisions (cf. the description in Sect. 5.3):

1. Batching decisions: which jobs should form a certain batch
2. Assignment decisions: which batch should be assigned to a certain machine
3. Sequencing decisions: in which sequence should batches be processed on a given machine

After solving the scheduling problem for machine groups that contain batch machines, artificial batch nodes are added to G for the formed batches. The predecessors of the artificial batch nodes are those nodes that represent the jobs that form the batch. They are called batched nodes. We denote the set of all batched nodes by V_{b}. For each $v \in V_{\mathrm{b}}$, the corresponding artificial batch node is denoted by v_{b}. The weight 0 is used for the incoming arcs of an artificial batch node because the processing time is represented by the outgoing arcs. The outgoing arcs of the artificial batch node connect the batch node with the successor nodes according to the process flows of the jobs that form the batch. Because we consider batching with incompatible families, all jobs within a batch have the same processing time. Therefore, it is reasonable that the weight of each outgoing arc of an artificial batch node is set to be the processing time of the batch.

In Fig. 5.7, we depict a disjunctive graph where machine 2 is a batch machine. The jobs represented by node $\langle 2,2\rangle$ and node $\langle 2,3\rangle$ form a batch. Rectangles with rounded angles are used to represent batches. Furthermore, we consider a second batch consisting only of one job that corresponds to node $\langle 2,1\rangle$. The second batch is sequenced immediately after the first batch.

The ready time determination scheme is also valid in graphs with artificial batch nodes. The due date determination scheme (5.92), (5.93) can be also applied to calculate the due dates of artificial batch nodes because the weights of the outgoing arcs are defined as in the non-batching case. However, the recursion equation (5.93) has to be modified for predecessors of batched nodes. We obtain

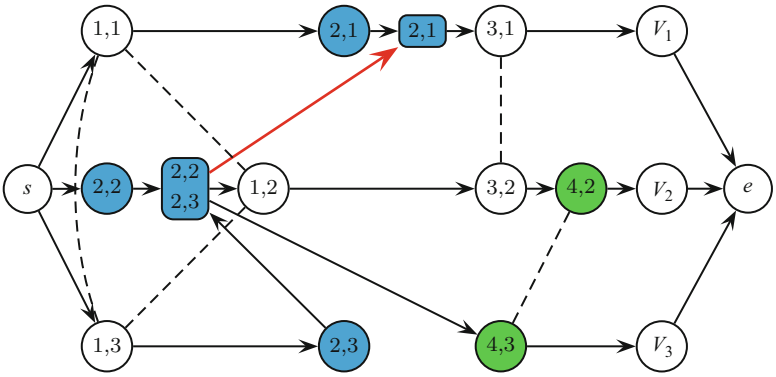

Figure 5.7: Representation of the batches $\{\langle 2,2\rangle, \langle 2,3\rangle\}$ and $\{\langle 2,1\rangle\}$ in G

$$\tilde{d}(u) := \min\{d(v) - p(v) | v \in V - V_b, (u,v) \in E_c\}. \tag{5.94}$$

We can derive $d(u)$ based on $\tilde{d}(u)$ as follows:

$$d(u) := \min\left\{\tilde{d}(u), \min\{d(v) - p(v_b) | v \in V_b, (u,v) \in E_c, \}\right\}. \tag{5.95}$$

This modification is necessary because $p(v) = 0$ for $v \in V_b$. Hence, Eq. (5.94) is not reasonable in this situation.

Finally, we have to consider the modeling of reentrant process flows. They are modeled in such a way that the corresponding nodes are not connected by disjunctive arcs. In this situation, the nodes are already connected by conjunctive arcs according to the process flow of the job.

As in the case of batch machines, new process restrictions can often be modeled by introducing additional nodes and arcs into G. For example, the integrated scheduling of manufacturing and transportation operations for complex job shops with AMHS is tackled in such a way for job shops by Knust [148] and by Driessel and Mönch [73] for complex job shops.

The disjunctive graph model can be used to find MIP formulations for job shop scheduling problems. We illustrate this modeling approach by means of the problem $\text{Jm}||C_{\max}$ (see Brucker and Knust [35]). The following parameters will be used within the model:

p_{ij} : processing time of the operation that is associated with node $\langle i,j\rangle$
M : large number

The following decision variables are used within the MIP:

S_{ij} : starting time of the operation that is associated with node $\langle i,j\rangle$
$y_{\langle i,j\rangle,\langle i,l\rangle}$: $\begin{cases} 1, \text{ if the operation associated with } \langle i,j\rangle \text{ is scheduled before } \langle i,l\rangle \\ 0, \text{ otherwise} \end{cases}$
$C_{\max}$: makespan of the schedule

Let N be the set of nodes that are associated with operations. By defining set A as the pairs of operations constrained by process flow relations and E_i the set of operations to be performed on machine i, the scheduling problem $\text{Jm}||C_{\max}$ can be formulated as follows:

$$\min\ C_{\max} \tag{5.96}$$

subject to:

$$S_{ij} + p_{ij} \leq S_{kj}, \quad (\langle i,j\rangle, \langle k,j\rangle) \in A, \tag{5.97}$$

$$S_{ij} + p_{ij} \leq C_{\max}, \quad \langle i,j\rangle \in N, \tag{5.98}$$

$$S_{ij} + p_{ij} - M\left(1 - y_{\langle i,j\rangle,\langle i,l\rangle}\right) \leq S_{il}, \quad \langle i,j\rangle, \langle i,l\rangle \in E_i,\ i = 1,\ldots,m, \tag{5.99}$$

$$S_{il} + p_{il} - M y_{\langle i,j\rangle,\langle i,l\rangle} \leq S_{ij}, \quad \langle i,j\rangle, \langle i,l\rangle \in E_i,\ i = 1,\ldots,m, \tag{5.100}$$

$$0 \leq S_{ij}, \quad \langle i,j\rangle \in N, \tag{5.101}$$

$$y_{\langle i,j\rangle,\langle i,l\rangle} \in \{0,1\},\ \langle i,j\rangle, \langle i,l\rangle \in E_i,\ i = 1,\ldots,m. \tag{5.102}$$

The objective (5.96) is to minimize the $C_{\max}$ performance measure. Constraints (5.97) model the fact that an operation cannot be started earlier than the sum of the start time of the predecessor operation and the corresponding processing time of this operation. Note that these constraints represent the conjunctive arcs. Constraints (5.98) ensure that $C_{\max}$ is equal to the largest completion time of the jobs. Constraints (5.99) and (5.100) model the fact that only one of the disjunctive arcs can be selected for two operations that have to be processed on the same machine. Finally, the constraints (5.101) lead to non-negative start times of the operations, while constraints (5.102) enforce binary values for the $y_{\langle i,j\rangle,\langle i,l\rangle}$.

An extension of this MIP to problem (5.83) can be found in Mason et al. [175]. In documented computational experiments, only problem instances up to eight jobs have been tackled using the resulting MIP. This is another indication for the need to design appropriate heuristics.

5.4.3 Decomposition Approach

The disjunctive graph modeling approach introduced in Sect. 5.4.2 forms the base for heuristics based on decomposition or on neighborhood search approaches for $\text{Jm}||C_{\max}$ (cf. Brucker and Knust [35] for a description of such techniques). Tabu search approaches are very competitive for $\text{Jm}||C_{\max}$ (see Nowicki and Smutnicki [215]) mainly because neighborhood structures can be designed that allow for fast evaluation of the $C_{\max}$ objective; however, this is not true for complex job shops with the TWT objective. Therefore, we describe decomposition heuristics based on the SBH proposed originally by Adams et al. [1] for $\text{Jm}||C_{\max}$.

The SBH decomposes the overall scheduling problem into a series of smaller scheduling problems: one for each machine group. The resulting scheduling

problems are typically smaller. The corresponding scheduling decisions lead to new conjunctive arcs within G. We have to deal with the following two problems:

1. Subproblem identification step: In this step, we have to identify appropriate constraints for the subproblems to model the interaction between the subproblems.
2. Composition step: We have to determine an appropriate sequence to solve the subproblems identified in the first step. Furthermore, the solutions of the subproblems have to be combined to obtain a solution of the overall scheduling problem.

We start by describing the resulting subproblems. Ready times and due dates for the jobs to be processed on individual machine groups can be determined by using the two basic recursion schemes (5.87), (5.88) and (5.92), (5.93). We use this data to formulate subproblems. Because of the production restrictions in wafer fabs (cf. Sects. 2.2.2 and 2.2.3), we have to solve problems of type

$$\text{Pm}|\text{p-batch}, \text{incompatible}, s_{jk}, r_j|\text{TWT}. \tag{5.103}$$

It is well known that the subproblem formulation based on ready times and due dates is not sufficient to determine schedules that can be combined to form feasible schedules for the overall scheduling problem (see Dauzère-Pérès and Lassere [59]). This is caused by the fact that scheduling decisions on other machines require a certain amount of time to elapse between the start times of operations on a given machine. Therefore, the resulting constraints are called delayed precedence constraints.

To illustrate this concept, we consider a disjunctive graph for three jobs that have to be processed on two machines similar to Ovacik and Uzsoy [223]. The data of this problem instance are shown in Table 5.10.

Table 5.10: Problem instance for a disjunctive graph with cycle

Job	Process flow (machines)	Processing time
1	1	$p_{11} = 5$
2	1–2	$p_{12} = 1, p_{22} = 1$
3	2–1	$p_{13} = 1, p_{23} = 1$

Figure 5.8 depicts the disjunctive graph where machine 2 is scheduled. The operation that is associated with node $\langle 2,3\rangle$ is processed after the operation that belongs to node $\langle 2,2\rangle$. Using longest path calculations, we determine that $r_{11} = 0$, $r_{12} = 0$, and finally $r_{13} = 3$. Taking only these ready times into account, it is clear that it is possible to schedule the operations associated with the nodes $\langle 1,1\rangle$, $\langle 1,3\rangle$, and $\langle 1,2\rangle$ in this sequence on machine 1. The corresponding

start times of the operations are 0, 5, and 6. The disjunctive graph now contains a cycle that consists of the nodes $\langle 1,2\rangle$, $\langle 2,2\rangle$, $\langle 2,3\rangle$, and finally $\langle 1,3\rangle$.

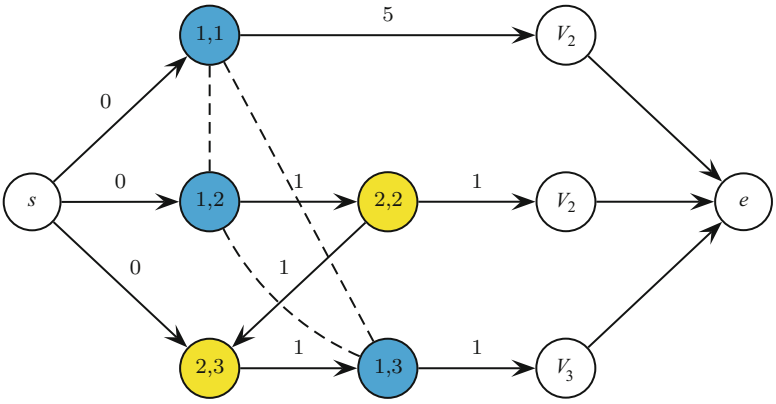

Figure 5.8: Disjunctive graph for three jobs and two machines

We can see from Fig. 5.9 that there is a path from $\langle 1,2\rangle$, $\langle 2,2\rangle$ and $\langle 2,3\rangle$ to $\langle 1,3\rangle$. Therefore, we conclude that the operation associated with $\langle 1,2\rangle$ has to be performed before the operation that corresponds to $\langle 1,3\rangle$. This precedence relation is a consequence of the scheduling decision for machine 2. When we formulate the subproblem for machine 1, we take only precedence relations into account that are followed from the process flows of the jobs. Because we do not respect the precedence relation between $\langle 1,2\rangle$ and $\langle 1,3\rangle$, we determine a schedule that is not feasible. The operation associated with $\langle 1,3\rangle$ can only start when the operations that belong to $\langle 1,2\rangle$, $\langle 2,2\rangle$, and $\langle 2,3\rangle$ are completed. It follows that the earliest start time for the operation associated with $\langle 1,3\rangle$ is two time units after the completion time of the operation corresponding to $\langle 1,2\rangle$.

We denote the precedence relations among the operations of the jobs that have to be taken into account during scheduling decisions by prec. We have to respect such delayed precedence relations to make sure that no cycles are created in the disjunctive graph during the implementation of subproblem-related schedules within the disjunctive graph. We have to replace the subproblem (5.103) by

$$\text{Pm}|\text{p-batch}, \text{incompatible}, s_{jk}, r_j, \text{prec}|\text{TWT}. \tag{5.104}$$

Delayed precedence relations are determined by a topological sorting of the nodes of G using an efficient depth-first search. We refer to Sect. 5.4.4 for details on this approach and its implementation. The algorithm of Pabst [225] is used within our SBH implementation.

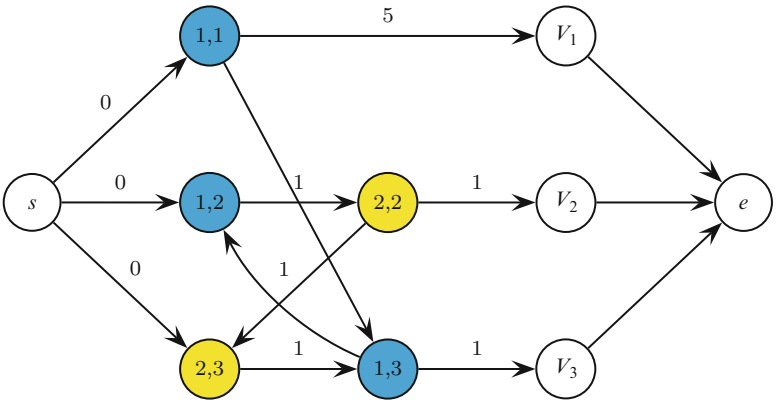

Figure 5.9: Disjunctive graph containing a cycle

The performance of decomposition approaches is influenced by the sequence in which the subproblems are solved and also by the solution approaches called SSPs for the scheduling problem found in Eq. (5.104).

Now we are able to present the SBH and its modification for scheduling entire wafer fabs. The SBH can be formulated as follows [1, 172, 223].

Algorithm SBH

1. We denote by M the set of machine groups that have to be scheduled. Furthermore, we choose the notation M_0 for the set of machine groups that are already scheduled. Initially, set $M_0 := \emptyset$. Generate an initial disjunctive graph G based on the process flows of the jobs. Determine ready times and due dates for all operations associated with nodes in G based on longest path calculations using the two recursions (5.87), (5.88) and (5.92), (5.93).
2. Identify and solve subproblems for each machine group $i \in M - M_0$ taking delayed precedence constraints into account.
3. Determine the most critical machine group $k \in M - M_0$ with respect to a certain criticality measure.
4. Implement the schedule determined in step 2 for the machine group $k \in M - M_0$ into the disjunctive graph, i.e., update G. Set $M_0 := M_0 \cup \{k\}$.
5. (Optional) Reoptimize the determined schedule for each machine group $m \in M_0 - \{k\}$ by considering the newly added disjunctive arcs from step 4 for machine group k.
6. The algorithm terminates if $M = M_0$. Otherwise, determine ready times and due dates for operations based on the nodes of G and go to step 2.

Note that the criticality measure used in step 3 determines in which sequence the machine group schedules are incorporated within G. For determining the most critical machine group, different approaches are possible. One option is

to consider the machine group that is associated with the subproblem that has the largest TWT value within an iteration of the SBH as the most critical machine group.

While this simple approach is intuitively appealing, it turns out that more sophisticated methods are more effective. It is suggested by Pinedo and Chao [241] to use the change of the completion times after the implementation of the schedule for a machine group relative to the graph in the previous iteration as a measure for machine criticality. The criticality measure CM for machine group i is given by

$$\mathrm{CM}(i) := \sum_{j=1}^{n} w_j (C_j'' - C_j') \exp\left\{-(d_j - C_j'')^+ / K\right\}, \tag{5.105}$$

where C_j' denotes the completion time of job j determined during the previous iteration of the heuristic. C_j'' is the new completion time of j, assuming that the schedule proposed in the current iteration for machine group i is implemented within G. Finally, K is a scaling parameter. Obviously, it holds $C_j' \leq C_j''$, as the inclusion of additional precedence constraints never decreases the completion times of the jobs. Therefore, $\mathrm{CM}(i)$ is a measure of how much the proposed schedule for machine group i will potentially increase the TWT value of all jobs in the job shop.

Furthermore, it is possible to exploit more global information like, for example, knowledge of planned and dynamic bottlenecks during decision-making (see Uzsoy and Wang [304]). Various criticality measures like the total machine load (TML) and the average remaining operations to completion (AROC) are studied in Aytuk et al. [15]. A blended index based on these criticality indices to determine the most critical machine group within each iteration of the SBH is proposed by Mönch and Zimmermann [198]. Finally, GAs (see Dorndorf and Pesch [69]) and inductive decision trees (see Osisek and Aytuk [221]) have been used to find an appropriate sequence in which the solutions of subproblems are implemented within G; however, these techniques tend to cause a very large computational burden.

Next, we describe the reoptimization carried out in step 5 in more detail. The reoptimization procedure reschedules the machine groups scheduled earlier taking G into account. G is the disjunctive graph that already contains the schedule of the last bottleneck machine group. Then, based on the TWT value of the new and the old schedule, it is decided whether the old schedule for a machine group is replaced by the new one. Even a small number of reoptimization steps increases the quality of the schedules considerably. The reoptimization algorithm can be described in a more formal way as follows.
Algorithm SBH Reoptimization

1. Set $r := 0$, where r is the number of the completed reoptimization steps. Let $r_{\max}$ be the maximum number of reoptimization steps. Calculate the TWT value that is associated with G. Assume that M_0 machine groups are already scheduled. Let k be the machine group that is scheduled

in the current iteration of the SBH. Set imp := true where imp states whether an improvement with respect to TWT was found or not within one reoptimization step.
2. Repeat steps 3 to 9 until $r_{\max}$ is reached or imp = false.
3. Set imp := false.
4. For all $i \in M_0 - \{k\}$, repeat the following steps.
5. Store the schedule for machine group i and remove the schedule for i from G, i.e., insert disjunctive arcs for the corresponding conjunctive arcs.
6. Determine ready times and due dates for all operations associated with nodes in G based on longest path calculations using the two recursions (5.87), (5.88) and (5.92), (5.93). Identify and solve the subproblem for machine group i taking delayed precedence constraints into account.
7. Implement the schedule determined in step 6 within G. Determine ready times for all operations associated with nodes in G. Calculate completion times based on these ready times and determine the TWT value that corresponds to G.
8. If the new TWT value is smaller than the old one, then update the smallest TWT value found so far. In this case, update imp := true; otherwise, replace the new schedule for machine group i by the old one stored in step 5. Go to step 4.
9. Update $r := r + 1$ and go to step 2.

Usually, two or three reoptimization steps are sufficient. The machine groups can be considered in the order in which they are introduced into the solution. There are SBH variants possible where reoptimization is only carried out after Step 6 in the SBH algorithm, i.e., when all machine groups are scheduled (cf. Ovacik and Uzsoy [223] for refined reoptimization strategies). Real options analysis is proposed by Yeung and Mason [327] to value reoptimization options in the SBH.

So far we assumed that G contains all the operations that have to be performed on all the machine groups. Because this might cause a large computational burden, several attempts were made to reduce the number of nodes in the scheduling graph by considering only specific heavily loaded machine groups explicitly. This results in reduced process flows that contain only operations that belong to these heavily loaded machine groups. The lightly loaded machine groups are represented by an infinite capacity resource that is scheduled using a due-date-oriented dispatching rule like EDD. Experiments with this approach for job shops that also include machine breakdowns are presented by Upasani and Uzsoy [301] and by Upasani et al. [302].

5.4.4 Subproblem Solution Procedures

We next describe in more detail how we can ensure that the delayed precedence relations are respected at the SSP level. The following availability condition from Pabst [225] for each operation associated with a node at the time

of its positioning in the schedule is necessary and sufficient for the avoidance of cycles in G: Either all operations that are associated with predecessors of the node are already scheduled or every operation associated with an unscheduled predecessor will not be scheduled on the same machine as the operation that belongs to the considered node. This statement can be proven by induction over the number of scheduled machine groups (see Pabst [225]).

For the design of appropriate SSPs, the following statement is important. If the previous availability condition holds for all nodes at the scheduling time of the associated operations, then for each node of an SSP it holds that either all direct predecessors are already scheduled or every operation associated with an unscheduled predecessor will not be scheduled later on the same machine like the operation associated with the considered node.

Moreover, as proved by Pabst [225], the next condition is necessary to avoid errors in the calculation of the availability times of the nodes and sufficient for the avoidance of cycles in G. For each node at the time of the positioning of the associated operation in the schedule, it holds that operations associated with direct predecessors are already scheduled and completed. Following Pabst [225], we define a direct precedence relation between two nodes belonging to the same subproblem if each path connecting the related nodes passes only nodes that belong to other subproblems.

In order to provide the required information for the solution of the subproblems, it is necessary to use an algorithm that determines all direct predecessors for a given set of jobs. A coupling with the calculation of the ready times of the operations of the jobs is useful. The depth-first search algorithm already mentioned in Sect. 5.4.3 is used.

SSPs are often provided by list scheduling algorithms that allow for taking the delayed precedence constraints into account. We briefly sketch an SSP for problem (5.104). Because the algorithm is a generalization of the BATC-II-TWDH algorithm described in Sect. 5.3.4, we describe only the main differences that are caused by the delayed precedence relations. Steps 1–4 are the same with the exception that the jobs to be scheduled on the machine group are determined using G. In step 5, only such job combinations are considered that do not contain jobs that are related by a precedence constraint to each other. Furthermore, only jobs with predecessors that are already included in a batch are considered. We denote the time, where all predecessors are scheduled and completed with $r_{\text{succ},i}$ for batch B_s of family i selected in step 7. In contrast to BATC-II-TWDH, in the presence of the delayed precedence relations, we have to advance the availability time of the machine k to $t_a(k) := \max(t_a(k), r_{B_s}, r_{\text{succ},i}) + p_i + s_{B_s}$, where s_{B_s} is the setup time that occurs before processing of batch B_s. We denote this SSP by SSP(BATC-II-TWDH). Note that this SSP can also be used to solve the scheduling problem

$$\text{Pm}|r_j, \text{prec}|\text{TWT}. \tag{5.106}$$

In this case, we set $\text{thresh} := \infty$, Δt can be large, and the assessment of job combinations is obsolete.

An SSP based on GAs is studied by Mönch et al. [206]. Note that SSP(BATC-II-TWDH) is outperformed by the GA-based SSP in many situations. Furthermore, an SSP based on list scheduling approaches using variants of the ATCS dispatching rule (cf. Sect. 4.3) and VNS for the problem

$$\text{Pm}|r_j, s_{jk}, \text{prec}|\text{TWT} \tag{5.107}$$

is proposed by Driessel and Mönch [72]. Note that in principle, a time window approach as for SSP(BATC-II-TWDH) also can be used to solve this problem.

5.4.5 Simulation-Based Performance Assessment

Several assessment efforts for static problem instances are described in the literature (cf. [64, 65, 223, 242] among others). But because we apply a deterministic scheduling approach to a stochastic BS and BP, there is a need to use the SBH in a rolling horizon setting. This approach allows one to take current information of the BS and the BP into account.

We use the software architecture described in Sect. 3.3.2 to carry out experiments in this section. The center point of this architecture is a data layer in the memory of the computer that contains all the information to construct the disjunctive graph and make the scheduling decisions. The data layer is between a simulation model that emulates the manufacturing process of interest and the scheduling application for the SBH. It acts as a mirror of the manufacturing process. It contains job information, process flows, and machines.

Calculated schedules are submitted to the simulation engine AutoSched AP in order to use the information of the schedules in a dispatching-based manner. The architecture allows for rolling horizon-type scheduling as well as for event-driven rescheduling activities. The different products are represented by separate ASCII files of the simulation model. An object-oriented database is used to store results and the disjunctive graph in order to reduce initialization efforts. The architecture is depicted in Fig. 5.10. Note that in contrast to the situation in Fig. 3.6, there is no need for a forecast module, a demand generation module, or a PS. The most important part of the CS is the SBH. The dispatcher is given by a dispatching rule that is used to select the next job to be processed on a machine with respect to the schedule determined by the SBH.

Next, we describe the design of experiments. We consider the full MIMAC 1 model (see Fowler and Robinson [83]). It contains over 200 machines that are organized into over 80 machine groups (cf. also Sect. 4.4). The model contains two technologies with 210 and 245 steps, respectively. We denote these two technologies by P1 and P2. In order to generate random process flows, process flows P1 and P2 are divided into 16 subprocess flows. The

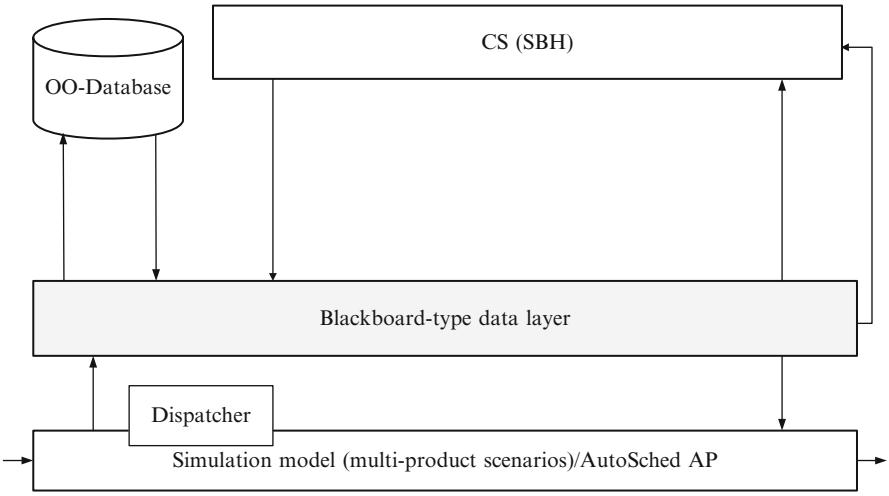

Figure 5.10: Simulation architecture for a performance assessment of the SBH

segmentation into subprocess flows takes the layer structure of wafers into account, i.e., performing the process steps of one subprocess flow leads to the manufacturing of a certain number of layers. We cover approximately the same number of layers within the corresponding segments of process flows P1 and P2. For each segment of the new process flows, we randomly choose one subprocess flow either from P1 or from P2. Following this approach, we are able to create a large number of different process flows. We consider 2, 4, 8, and 16 different products in our simulation experiments. Jobs representing different products form different families and cannot be batched together. Therefore, they influence the dynamics of the wafer fab to a large extent. We set the due date of job j by

$$d_j := r_j + \mathrm{FF} \sum_{k=1}^{n_j} p_{jk}, \tag{5.108}$$

where the abbreviation FF is used for the flow factor. The allocated amount of waiting time that can be spent by job j is therefore given by $(\mathrm{FF}-1)\sum_{k=1}^{n_j} p_{jk}$. We use $\mathrm{FF} = 1.4$, i.e., tight due dates, in our simulation experiments. The weights of the jobs are selected according to the discrete distribution

$$D := \begin{cases} w_j = 1,\ p_1 = 0.5 \\ w_j = 5,\ p_2 = 0.35. \\ w_j = 10,\ p_3 = 0.15 \end{cases} \tag{5.109}$$

Furthermore, we use a high load of the wafer fab. To reduce the simulation time, we start each simulation with a WIP distribution of the jobs. We perform deterministic simulation runs for 50 days. We apply the SBH every 2 h with a scheduling horizon of 2 h, i.e., we have $\tau_\Delta = 2h$ and $\tau_{ah} = 0h$ (cf. Sect. 2.3 for this notation). The current state of the BS and BP is taken into account every 2 h for determining a new schedule.

We are interested in the TP, the CT, and the AWT value. We use AWT instead of TWT because AWT takes also the number of completed jobs into account. Therefore, AWT is a fairer measure for comparison in situations where the TP value, expressed as the number of completed jobs, is different for the SBH and a FIFO-based dispatching scheme. In order to avoid difficulties with absolute values, we use relative values denoted by REL. The value for REL is defined as the ratio of the performance measure value obtained by the SBH or the HVF-EDD dispatching rule and the corresponding performance measure value for the wafer fab that is controlled using the FIFO dispatching rule. The HVF-EDD dispatching rule uses first HVF (highest weight) (cf. Sect. 4.2.1) and breaks ties by applying EDD. This dispatching rule is an example of a multilevel rule (cf. Sect. 4.1). It is selected for comparison with the SBH because it outperforms several dispatching rules in the present situation (cf. Mönch and Zimmermann [200] for those dispatching rules). The simulation results are presented in Table 5.11.

Table 5.11: Computational results for the SBH and HVF-EDD

Number of products	Rel(AWT) SBH	Rel(AWT) HVF-EDD	Rel(TP) SBH	Rel(TP) HVF-EDD	Rel(CT) SBH	Rel(CT) HVF-EDD
2	0.4588	0.4734	0.9384	0.9788	0.9994	0.9165
4	0.2499	0.1373	0.9973	0.9892	0.9185	0.8297
8	0.1673	0.2113	1.0567	0.9845	0.8412	0.8004
16	0.2276	0.7131	1.0068	0.9002	0.9625	1.0168

We can conclude from Table 5.11 that we obtain decreasing AWT values for an increasing number of products compared to the FIFO strategy. At the same time, we can observe that the relative CT value is decreased with an increasing number of products in some situations. We find some TP losses for two and four products. TP reductions are the price for AWT reductions in the case of large-scale wafer fabs, at least for a small number of products. The SBH clearly outperforms the HVF-EDD dispatching rule in many situations with respect to the three measures.

Many more computational results can be found for a small number of products in Mönch et al. [206] for the TWT measure, in Sourirajan and Uzsoy [288] for the $L_{\max}$ measure, and for the multiproduct case with TWT as performance measure in Mönch and Zimmermann [200].

We have to deal with the question of how to take the feedback from the BS and from the BP into account in a rolling horizon setting. In our experiments, we periodically determine a schedule from scratch. Dispatching rules are used to implement this schedule, i.e., when a machine becomes free, the job or batch is selected based on priorities obtained from the schedule rather than implementing the schedule rigidly. A similar approach is taken by Barua et al. [23].

It is also possible to repair an existing schedule when machine breakdowns occur. A repair can also be performed periodically. Different rescheduling strategies for a SBH for scheduling wafer fabs are proposed by Mason et al. [173].

5.4.6 Distributed Shifting Bottleneck Heuristic

Even computationally efficient implementations of the SBH (see Blazewicz et al. [30]) for complex job shops are time-consuming for moderate scheduling horizons. In the situation of a larger scheduling horizon, for example, $h = 2$ days, the number of nodes of G to be considered grows tremendously; hence, the solution of the scheduling problem requires very large computational efforts in terms of memory and computation time. On the other hand, considering a very small scheduling horizon leads to the problem of a proper internal due date setting that is often difficult and can lead to performance deterioration.

A conceptual framework for solving scheduling problems via decomposition into subproblems that are larger than the machine groups in the original shifting bottleneck approach is described by Brandimarte et al. [32]. But a concrete decomposition and solution scheme and results of computational experiments are not presented. Based on an appropriate physical decomposition of the BS into work areas (cf. the description in Sect. 2.2.2), we present a two-level hierarchical approach following Mönch and Driessel [193]. We use a simple job planning approach in order to assign internal ready times $r_j^{(k)}$ and internal due dates $d_j^{(k)}$ to each job j for each work area k on the top level. Based on these internal ready times and due dates, we apply the SBH to each work area in order to come up with detailed schedules for the jobs within the work areas on the base level. The overall scheme is called distributed SBH (DSBH).

For describing the DSBH, we start at the top level. This level is based on an aggregated model. As proposed by Habenicht and Mönch [112], we aggregate consecutive process steps of a process flow into macro operations. We denote the macro operation l of job j by mo_{jl}. Assume, that mo_{jl} is formed by the $s+1$ consecutive process steps $O_{jr}, O_{j,r+1}, \ldots, O_{j,r+s}$, where the rth process step is the generating one. In an analogous way to the processing time for a

process step, we define the processing time of the macro operation mo_{jl} by

$$p(\mathrm{mo}_{jl}) := p_{jr} + p_{j,r+1} + \ldots + p_{j,r+s}. \tag{5.110}$$

Each macro operation is related to a specific work area. We assume that we can determine a unique process step among the macro operation forming process steps that has to be performed on a machine group with a high utilization. Hence, the behavior of a macro operation is basically influenced by the machine group of this process step. We consider aggregated capacities for the machines of a certain machine group that take batching characteristics and machine breakdowns into account. We assign start dates and due dates to each single macro operation based on an infinite capacity approach (ICA). The ICA algorithm can be summarized as follows.

Algorithm ICA

1. Adjust d_j for each job j in such a way that $\sum p(\mathrm{mo}_{jl}) \leq d_j - t$ is valid, where we denote by t the current time.
2. Determine the quantity $h_j := (d_j - t)/\sum p(\mathrm{mo}_{jl})$ at time t. If $h_j \geq 1$, the remaining time until d_j is distributed equally among the remaining macro operations of job j. The sum of the possible waiting times for the operations that form the macro operation mo_{jl} is $(h_j - 1)p(\mathrm{mo}_{jl})$ to meet the due date d_j.
3. Calculate ready times $r(\mathrm{mo}_{jl})$ for operation mo_{jl} in a recursive manner via the expression $r(\mathrm{mo}_{j,l+1}) := r(\mathrm{mo}_{jl}) + h_j p(\mathrm{mo}_{jl})$ for $l \geq l_0 + 1$, where we assume that the macro operation mo_{jl_0} is the current one with a given $r(\mathrm{mo}_{jl_0})$.
4. The due date $d(\mathrm{mo}_{j,l-1})$ of macro operation $\mathrm{mo}_{j,l-1}$ is given by $r(\mathrm{mo}_{jl})$ for $l \geq l_0 + 1$.

We apply the ICA approach in a rolling horizon manner every $p_{\max}\tau_\Delta$ time units taking the current state of the BS and the BP into account. We call τ_Δ the scheduling interval of the base level, and $p_{\max}$ is a positive integer. ICA provides the internal ready times $r_j^{(k)}$ and due dates $d_j^{(k)}$ with respect to work area k.

The naive DSBH (NDSBH) uses the internal ready times and due dates determined by ICA to calculate detailed schedules for the jobs that have to be processed on the machine groups of a single work area within a certain scheduling horizon. The heuristic can be summarized as follows.

Algorithm NDSBH

1. Determine $r_j^{(k)}$ and $d_j^{(k)}$ of the jobs for each work area k using ICA.
2. Determine schedules for each single work area by using the algorithm SBH for $r_j^{(k)}$ and $d_j^{(k)}$ from step 1.

In the case that more than one consecutive macro operation is expected to be performed within the scheduling horizon h, we have to consider the due date of the last macro operation as the due date for the calculation of the TWT.

Furthermore, we have to modify the disjunctive graph G to make sure that the reentrant behavior of the process flows is correctly modeled. Therefore, we add additional disjunctive arcs between nodes that represent operations of the same job belonging to consecutive macro operations. The approach is similar to the treatment of reentrant flows in the SBH. The main differences are the multiple internal ready times and multiple internal due dates with respect to work area k. We show the resulting G for two jobs in Fig. 5.11. Job 1 contains one macro operation, represented by the nodes $\langle 1,1\rangle$, $\langle 2,1\rangle$, and $\langle 3,1\rangle$. Job 2 has two macro operations: one consists of the operations that belong to the nodes $\langle 1,2a\rangle$, $\langle 2,2a\rangle$, and $\langle 3,2a\rangle$, whereas the second one is given by $\langle 1,2b\rangle$, $\langle 2,2b\rangle$, and finally $\langle 3,2b\rangle$. Additional conjunctive arcs are included between nodes $\langle 1,2a\rangle$ and $\langle 1,2b\rangle$, $\langle 2,2a\rangle$ and $\langle 2,2b\rangle$, and $\langle 3,2a\rangle$ and $\langle 3,2b\rangle$. Note that the artificial end nodes V_j contain the $d_j^{(k)}$ of the related macro operations. The ready times of the operations associated with node $\langle 1,1\rangle$, $\langle 1,2a\rangle$, and $\langle 1,2b\rangle$ are also derived from the top level of the hierarchy via the ICA approach.

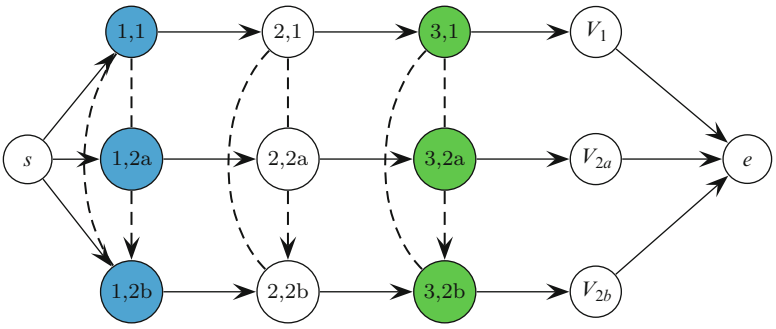

Figure 5.11: Modified graph for macro operations

NDSBH has the drawback that a complete decoupling of the work areas can take place that might result in suboptimal or even infeasible schedules with respect to the overall scheduling problem for the entire job shop. On the other hand, if ICA of the top level determines accurate start dates and end dates with respect to available capacity, it might be sufficient to consider NDSBH.

A second approach considers modified start dates and end dates iteratively. By using this approach, we obtain better schedules for the overall scheduling problem in a step-by-step manner. The suggested procedure is called iterative DSBH (IDSBH). The basic steps can be summarized as follows.

Algorithm IDSBH

1. Let $\mathbf{MB}$ denote the set of all work areas. Denote by $\mathbf{MB}_0$ the set of work areas for which a schedule is already calculated. Initially, set $\mathbf{MB}_0 := \emptyset$.

Determine initial ready dates $r_j^{(k)}$ and due dates $d_j^{(k)}$ of the jobs with respect to each work area k using ICA.

2. Determine schedules for all work areas $\mathbf{MB} - \mathbf{MB}_0$ by using the SBH and the current $r_j^{(k)}$ and $d_j^{(k)}$ calculated in step 1.
3. Determine the most critical work area from $\mathbf{MB} - \mathbf{MB}_0$ by calculating the TWT value of the jobs with respect to the corresponding work area. Denote the most critical work area by k. Update $\mathbf{MB}_0 := \mathbf{MB}_0 \cup \{k\}$.
4. Use the schedule for k in order to determine modified ready times $\tilde{r}_j^{(k)}$ and end due dates $\tilde{d}_j^{(k)}$ for the remaining work areas, i.e., for macro operations of jobs in work areas $\mathbf{MB} - \mathbf{MB}_0$, based on the start and completion times of the jobs in k. Send the modified start dates and end dates to the remaining work areas. Update the current ready times and due dates.
5. Determine schedules for the work areas $m \in \mathbf{MB} - \mathbf{MB}_0$ that take the new ready times and due dates into account.
6. If $\mathbf{MB} = \mathbf{MB}_0$, then stop. Otherwise, go to step 3.

IDSBH eliminates most of the drawbacks of NDSBH. However, if the scheduling horizon is large, scheduling decisions based on modified start dates and end dates might negatively affect the previous scheduling decisions for critical work areas, especially because of multiple macro operations of jobs within one work area. Hence, an adaptation of the schedules, similar to the reoptimization cycles of the shifting bottleneck heuristic is necessary, and this tends to be computationally expensive. A truncated variant of IDSBH may perform only a certain number of iterations instead of doing all iterations in order to reduce the computational burden.

Results of extensive simulation experiments with a similar setting as described in Sect. 5.4.5 can be found in [193]. It turns out that IDSBH outperforms NDSBH with respect to TWT and that CT and TP are similar. The SBH as described in Sect. 5.4.3 performs only slightly better. It is also shown by Mönch et al. [204] that the DSBH-type algorithms lead to smaller computational burden when different computers are used to solve the work-area-related scheduling problems after the decoupling by ICA.

5.4.7 Multicriteria Approach to Solve Large-Scale Job Shop Scheduling Problems

In this section, we extend problem (5.83) to the case of more than one performance measure. This approach is reasonable because usually several performance measures are of interest in wafer fabs. It is demonstrated by Demirkol and Uzsoy [63] that solving problem (5.84) leads to schedules that perform well even for performance measures other than $L_{\max}$. Multicriteria approaches are not often studied for complex job shop scheduling problems, with the notable exception of Pfund et al. [235].

In the following, we assume a scheduling problem with performance measures γ_1 and γ_2 to be minimized. Let S be a feasible schedule and let $\gamma_1(S)$ and $\gamma_2(S)$ be the objective function values for S with respect to the two measures. While our discussion is on bicriteria problems, the ideas can be easily extended to problems with more criteria.

Sometimes the decision maker has a priori information regarding the nature of optimization to be performed. For instance, criteria γ_1 may be of primary importance and γ_2 of secondary interest. In other words, a solution best in γ_2 may be desired among all solutions that are best in γ_1. This is called hierarchical or lexicographic optimization. In other cases, the decision maker may have a composite linear function of the form $F(\gamma_1, \gamma_2) := \alpha\gamma_1 + (1-\alpha)\gamma_2$, $0 \leq \alpha \leq 1$ in mind that needs to be minimized. The weighted sum translates multiple objectives into a single objective value for a proposed schedule.

S is called Pareto optimal or nondominated if there exists no other schedule S' such that $\gamma_1(S') \leq \gamma_1(S)$ and $\gamma_2(S') \leq \gamma_2(S)$ where at least one of the two inequalities is strict. If all Pareto optimal solutions are known, the decision maker can choose the schedule that is most preferred from this set. This approach is called an a posteriori approach and is generally the most difficult and the most computationally expensive. For a comprehensive survey on multicriteria scheduling, we refer the reader to T'kindt and Billaut [298].

We consider a multicriteria scheduling problem in which we combine $C_{\max}$, TC, and TWT into a single aggregation function. Our aggregation function, however, is different from the linear combination of objectives described earlier. We use, instead, a desirability function (cf. the description in Sect. 3.3.1) to aggregate the objectives. We assume the decision maker has a prefixed goal/target value and an upper/worst-case value for every criterion. We also assume the decision maker, just as in the linear combination case, is aware of the priorities on the objectives. We use the desirability function approach already discussed in Sect. 3.3.1. We use $m = 3$, as we represent the desirabilities of $C_{\max}$, TC, and TWT as d_1, d_2, and d_3, respectively. Each desirability value d_i in our experiments will have goal value G_i and maximum (upper) limit U_i.

It is important to link the properties of a solution that is optimal with respect to a desirability function and its connection to the classical multicriteria idea of a Pareto optimal or nondominated solution. From the definition and discussion of the desirability function, it is clear that the optimal solution depends upon the G_i values and U_i values assumed for each criterion i. We have listed below two specific cases with regard to this. While these cases highlight situations where optimality with respect to the desirability function and Pareto optimality may not always match with each other, they also show that these differences are more due to the framing of the problem with respect to upper and lower limits than any inherent issues in the desirability approach.

1. The way the desirability function is structured, any solutions that are over the prespecified upper limit for a minimization problem for any of the criteria are assigned a value of 0. If the upper limit is fairly strict,

i.e., aggressive, this can exclude nondominated solutions that are at the ends of the efficient frontier. But in practice, this simply means that these solutions are not acceptable because they perform poorly for at least one of the criteria. Thus a nondominated solution may not be optimal from a desirability point of view as certain constraints on individual criteria have to be met.

2. We consider now the set of feasible solutions $\bar{S}$ with non-zero desirability values. Thus, for each solution in $\bar{S}$, the criteria values will be strictly less than the upper limits. Let x^* be a dominated solution in $\bar{S}$. Let ND_{x^*} be the set of nondominated solutions that dominate x^*. Also, let $x(i)^*$ represent the value of criterion i for solution x^* and G_i represent the goal value for criterion i. Then, for a given set of desirability weights, x^* is an optimal solution with respect to the desirability function D if there exists no criterion i and no x^{**} in ND_{x^*} such that $x(i)^{**} \leq G_i \leq x(i)^*$. This situation arises because D does not distinguish between solutions that are below G_i for criterion i because it assigns them a value of 1 even though there are differences in criteria values. This reflects the idea that a satisfaction level for that criterion has been met. We note that while x^* is optimal for the desirability function despite being dominated, there exists a solution(s) in ND_x which is (are) nondominated and also alternately optimal. Also, as a special case, if we were to set G_i to 0 or to a really low value, x^* would no longer be optimal for the desirability function. This leads to our decision for the choice of G_i values in our approach.

The assumptions of the discussed scheduling problem are the same as in Sect. 5.4.1. However, in contrast to the problem discussed there, we have to deal with three different criteria. Using the $\alpha|\beta|\gamma$ notation, the multicriteria scheduling problem to be solved can be represented as

$$\mathrm{FJm}|\text{p-batch}, \text{incompatible}, s_{jk}, r_j, \text{recrc}|C_{\max}, \mathrm{TC}, \mathrm{TWT}. \tag{5.111}$$

Note that the problem (5.111) is NP-hard because when any of the three performance measures are considered separately, the resultant scheduling problem is NP-hard. Therefore, we will use an appropriate modification of the algorithm SBH described in Sect. 5.4.3. At step 2 and step 3 of algorithm SBH, we propose to use the desirability function approach. SSPs are identified and solved in step 2, while the most critical machine group is determined in step 3 using a machine criticality measure (MCM). We will use the abbreviation SSP level and MCM level in the remainder of this section.

At the SSP level, we use the ATCSR dispatching rule (see Sect. 4.3.2 for the corresponding index) to determine schedules for the SSPs. The priority index of the ATCSR dispatching rule for job j is given by

$$I_j(t,l) := \frac{w_j}{p_j} \exp\left(-\frac{(d_j - p_j - \max(r_j, t))^+}{\kappa_1 \bar{p}}\right) \exp\left(-\frac{s_{lj}}{\kappa_2 \bar{s}} - \frac{(r_j - t)^+}{\kappa_3 \bar{p}}\right), \tag{5.112}$$

where $\bar{s}$ is the average setup time and κ_2 and κ_3 are look-ahead parameters for the setup and ready time terms, respectively. The three look-ahead parameters serve as scaling parameters. Varying the look-ahead parameters κ_i and therefore varying the relative importance of the terms in index (5.112) can provide high-quality schedules for the subproblems. It can be easily shown by experimentation that the solutions that will be picked at the SSP level using the desirability function will be the nondominated solutions among the schedules that are generated by varying the look-ahead parameters of the ATCSR dispatching rule. Using ATCSR-based list scheduling in this manner for multicriteria purposes is based on work by Balasubramanian et al. [19] that empirically explores the performance of a composite dispatching rule similar to the ATCSR rule for single machine bicriteria scheduling.

At the SSP level, the look-ahead parameters are varied using a grid search approach in order to generate a wide range of schedules for subsequent consideration by D for the three objectives of interest. We use five different values for each scaling parameter and thus test 125 different combinations. Parameter κ_1 is incremented from 0.1 to 2.1 in steps of 0.4; κ_2 is incremented from 0.1 to 1.1 in steps of 0.2, while κ_3 is incremented from 0.001 to 0.011 in steps of 0.002. A schedule is determined for each combination of the different look-ahead values, i.e., for each triple $(\kappa_1, \kappa_2, \kappa_3)$ from the grid. The values for U_i and G_i are thus set to the worst or best value, respectively, observed for each objective over the different schedules considered at the SSP level. Clearly, the values for U_i and G_i can be fixed, predetermined values, as knowledge of a particular machine group and its performance may be known a priori in a real-world setting, thereby making it easier to decide upon appropriate values for U_i and G_i. In a next step, using these values for U_i and G_i, we calculate the combined desirability D for each schedule from the grid.

Considering the fact that D is the geometric mean of all d_i, we mandate that the schedule with the highest D value over all schedules from the grid will not result in a d_i of zero. Clearly, the schedule that performs worst for one objective may not necessarily perform poorly for the other objectives of interest. Therefore, we assign a desirability value of 0.0001. Letting any $d_i = 0$ for a schedule causes the D value of this schedule to equal 0, which disqualifies this schedule for selection even when its performance for other objectives may be quite good.

We will use the TWT value of an SSP associated with a machine group as the MCM because it turns out that a criticality measure based on index (5.105) does not lend itself easily to the inclusion of multiple objectives using the desirability approach. This is mainly because the index contains completion time differences. Furthermore, the value of the scaling parameter K can dramatically affect which machine group will be selected as the most critical one. We call this approach for determining the most critical machine group MCM-TWT.

Before the desirability function can be used at the MCM level, the U_i and G_i values for each of the three objectives of interest must be determined.

We employ a procedure similar to the one used at the SSP level. The only difference is that at the MCM level, we are interested in identifying and scheduling the machine group with the lowest D value. Since the D value aggregates three different criteria, a low desirability value for a given machine group indicates that it is the most critical with respect to all the criteria under consideration and therefore should be scheduled first.

We consider two different approaches for identifying the critical machine group at the MCM level. First, the impact of each schedule for a machine group on $C_{\max}$, TC, and TWT of the entire complex job shop is assessed when identifying the critical machine group by inserting the schedule of the machine group into G. This approach is abbreviated as global MCM (GMCM). Alternatively, the critical machine group can be identified using only SSP-level performance metrics, i.e., we do not consider the impact of the machine group on the rest of the complex job shop. This approach is called local MCM (LMCM). The goal in the proposition of these two approaches is to test whether a difference is noticeable in the global and local approaches. In a practical setting, if the SBH procedure were to be used for scheduling, the computation time for the LMCM approach for large problem sizes would be less than the time for the GMCM approach. But intuitively, it seems that the GMCM approach reflects the critical machine group more accurately, since it takes into account conflicts between jobs in the entire wafer fab.

Next, we discuss some computational results for the MiniFab model, described in Sect. 3.2.8 and shown in Fig. 3.4. In contrast to Sect. 5.4.5, we consider only static problem instances of the MiniFab model. Each of the problem instances contains 20 jobs. Two products are taken into account, and there are ten jobs of each product. The weights of the jobs are selected as $w_j \sim \mathrm{DU}[1,100]$, and the ready times are zero. The due dates of the jobs j are chosen according to

$$d_j := \mathrm{FF}_j \sum_{k=1}^{n_j} p_{jk}, \tag{5.113}$$

where FF_j is the FF value for job j. The FF values are generated according to $U(\mathrm{FF}_{\min}, \mathrm{FF}_{\min} + \mathrm{FF}_{\max}/2)$, where $\mathrm{FF}_{\min} := \min_j \mathrm{FF}_j$ and $\mathrm{FF}_{\max} := \max_j \mathrm{FF}_j$ and the FF_j are determined by solving a single problem instance using FIFO dispatching at all machine groups.

We generate 20 different problem instances based on the MiniFab model, each with its own unique FF_j and w_j values. We focus on optimizing $C_{\max}$ and TWT, disregarding TC in an attempt to illustrate the difference between the performance of the different approaches. The impact of using desirability functions at the MCM level is small, because the MiniFab model contains only three machine groups. However, using desirability functions at the SSP level produces significantly better results for the MiniFab model-based problem instances. As the SSP-level results are independent of the approach used at the MCM level, the approaches to be compared reduce to using SSPs for pure TWT minimization as described in Sect. 5.4.4 and for SSPs that are based

on the ATCSR dispatching rule and grid search to find schedules with large values for D. The former SSP is called SSP-TWT, while the latter one is called SSP-Des for abbreviation.

We compare the results obtained by SBH with results taken from using a deterministic forward simulation with EDD and CR dispatching rules (cf. Sects. 4.2 and 4.3 for a definition of the corresponding priority indices). We show objective space plots for two representative problem instances based on the MiniFab model in Figs. 5.12 and 5.13.

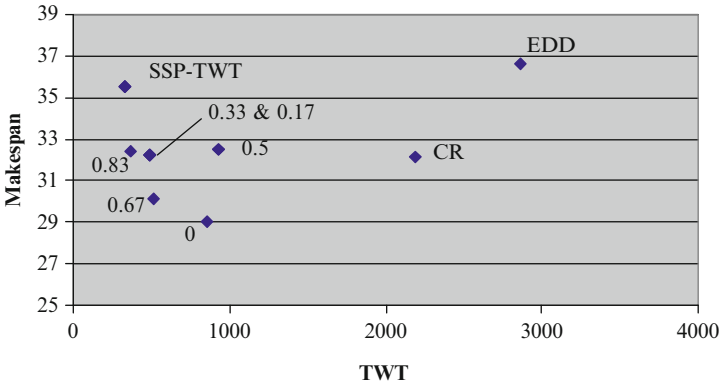

Figure 5.12: Solutions in the objective space: trade-off situation

The numbers adjacent to each point indicate the weight z_3 of the TWT criterion in D. Therefore, all solution points that have been labeled by a fraction between 0 and 1 are those obtained by using the different weight combinations at the SSP level. The weight of $C_{\max}$ is given by $1 - z_3$. In Fig. 5.12, it is possible to see the trade-off between TWT and $C_{\max}$. While the SSP-TWT solution produces the smallest TWT value, its $C_{\max}$ value is not small. The solutions generated by the different weight combinations generate a variety of solutions, which produce slightly worse TWT values, but the solutions are still significantly better than solutions obtained by the CR or EDD rules, and they improve on $C_{\max}$. Figure 5.13 shows a graph where using the desirability approach generates a solution better in both TWT and $C_{\max}$ than the solution found by SSP-TWT.

These two different problem instances shown in Figs. 5.12 and 5.13 roughly classify the nature of solutions for all 20 of the problem instances, i.e., either a trade-off exists between the two objective values or SSP-Des generates a solution better in both objective values.

More results of computational experiments also using the SBH in a rolling horizon setting for the MIMAC 1 model can be found in [235] taking a design

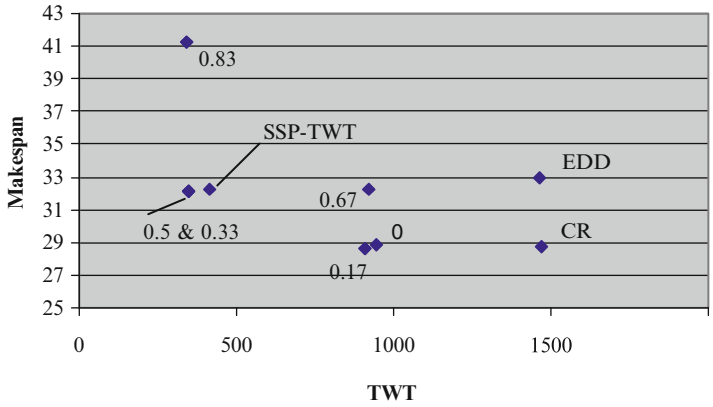

Figure 5.13: SSP-Des solution is better in both objective values

of experiments similar to those in Sect. 5.4.5. The three different performance measures for problem (5.83) are considered. Experimental results suggest that SSP-Des on the SSP level and MCM-TWT on the MCM level is the setting that performs well most often for a wide range of decision-maker priorities, followed closely by the combinations SSP-Des/LMCM and SSP-Des/GMCM. An important conclusion is that the use of the desirability function approach on the SSP level consistently produces superior results.

Chapter 6
Order Release Approaches

In this chapter, we provide an overview of order release approaches for semiconductor manufacturing. Order release is between production planning and scheduling in the PPC hierarchy. The fundamental concepts of push-based order release and pull-based order release are presented, along with a comparison of these two key methods and their implementation in a variety of production environments. Next, we present two seminal wafer fab-specific order release approaches, namely starvation avoidance by Glassey and Resende [100] and workload regulation by Wein [318].

After discussing subsequent order release methods that followed these first two key approaches, we demonstrate the interaction between order release and scheduling. A DSBH-type heuristic (cf. Sect. 5.4.6) is used to compare three different order release methods from the literature under a variety of wafer fab operating conditions.

Next, we present the findings of a large-scale order release study that was conducted at a large, global semiconductor manufacturer in order to answer important questions relating to both the timing and quantity of order release into their wafer fab.

Finally, we present a MIP model for optimizing order release into wafer fabs that seeks to improve the utilization of machines in the constraint machine group. This optimization model determines both the timing and quantity of order releases into a wafer fab on a weekly basis.

6.1 Push Versus Pull Approaches

In this section, we start by discussing push and pull approaches for order release. Moreover, we compare these two important classes of order release schemes.

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5_6,

6.1.1 Push Approaches for Order Release

From the early years of material requirements planning (MRP) in the 1960s through the early to mid-1980s, many followers of MRP/MRP II methods simply released all orders for which material and planning were available according to the calculated order release date (see Wight [322]). This push approach amounted to material planners starting into their respective systems what they hoped to get out without any recognition or understanding of the BS capacity and/or potential BS congestion. Such push approaches quite often led to large WIP levels and high CT values in the presence of capacity constraints. It is not surprising therefore that dispatching rules were the primary means of production control and the subject of much research in the 1970s and 1980s as there was much WIP to control and dispatch.

Faced with excessive amounts of WIP, many companies responded by limiting the daily release of material to some fixed levels that were based on production goals. However, this release-limiting approach did not properly comprehend BS capacity and congestion, i.e., it was still a push philosophy. Typically, the release-limiting policies only marginally improved WIP levels as compared to the earlier order release methods, as companies initially overestimated their own production capacity. However, as these same companies worked to better understand their own capacity and potential congestion issues, they achieved greater WIP management success because they gradually began to adjust order release rates to appropriate levels.

6.1.2 Pull Approaches for Order Release

A new collection of order release strategies based on the concept of pulling work into a BS that was ready for it versus pushing work into the same BS, regardless of its ability to accept the work, began to emerge in the early 1980s due to the following:

1. Internal recognition that current release practices lacked intelligence
2. The appearance of Japanese management concepts like just-in-time (JIT)
3. The development of bottleneck-based methods like the optimized production technique (OPT) (cf. Jacobs [125])

Initially, some industries attempted to adopt JIT philosophies to reduce WIP by implementing Kanban cards—a signaling mechanism between different points in the manufacturing process that visually indicates when a new order can be released into the BS or when an existing WIP job can be moved to a subsequent/downstream process step—or some other limited or fixed WIP approach.

Concurrent with the growth of interest in JIT and Kanban strategies, bottleneck resource scheduling philosophies gained prominence. These methodologies revolved around first identifying bottleneck machines and/or processes and then using the identified bottlenecks as central points of focus for production control strategies. Both OPT and Goldratt's theory

of constraints [104] are representative. The drum-buffer-rope approach to production control by Goldratt and Fox [105] suggests that:

1. The slowest paced (bottleneck) process provides the pace of a system or production line (drum).
2. The bottleneck should be tied to the entry points of the system (ropes).
3. The bottleneck should be always provided with a time-phased inventory of work (buffer) that guards it from being idle.

In the 1990s and early 2000s, considerable interest grew in the CONWIP methodology (see Spearman et al. [290]) for limiting the inventory levels of a manufacturing system. Although conceptually similar to early input/output control ideas from the 1970s, CONWIP focuses on WIP control rather than TP control and can also be viewed as a generalization of Kanban. CONWIP is a simple, robust control philosophy based on a fundamental understanding of the relationship between the WIP in a BS and its TP. This relationship is visually captured in a production system characteristic curve that can be either developed through a simulation study or approximated analytically. An example for a characteristic curve is shown in Fig. 6.1.

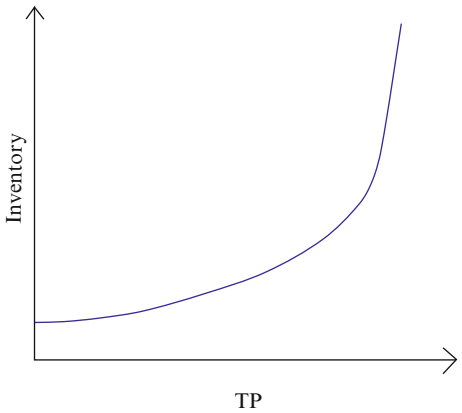

Figure 6.1: Example of manufacturing system characteristic curve

Once the characteristic curve is developed, one selects a target WIP level for a desired TP rate of the BS. Then, efforts are made to keep WIP at or below this target level in the BS and measure the resulting TP to validate the previously developed characteristic curve. WIP targets must be adjusted if the TP of the BS does not meet the desired goals. Clearly, the specification of the order release rate into the wafer fab directly impacts performance according to Little's law (cf. Eq. (3.21)). The combination of setting the value of the TP rate λ and specifying a desired target inventory level, represented by the WIP, leads to an effective estimate of expected CT via this

important governing law of factory physics. Spearman et al. [289] describe the development of a CONWIP-based hierarchical production planning and control system for a circuit board manufacturer's facility.

6.1.3 Comparing Push Versus Pull Approaches

As described in Sects. 6.1.1 and 6.1.2, push order release approaches are motivated by a company wanting to produce some desired quantity of goods, while pull-based methods are based on the knowledge of what actually can possibly be produced in light of existing capacity constraints and/or congestion issues. The superiority of pull systems is well documented in the literature for a variety of production environments employing dispatching policies under varying levels of due date tightness [256, 262].

Unfortunately, research findings do not always make their way into practice, as the presumed need for and protection of large amounts of WIP have not been easily overcome in some industries and companies, even when these same companies purport to follow JIT philosophies.

6.2 Tailored Approaches for Wafer Fabs

Order release and dispatching have received a fair amount of attention from both semiconductor manufacturing researchers and practitioners alike. One of the earliest papers that focused on semiconductor manufacturing workload control is by Dayhoff and Atherton [61]. Although they focused solely on dispatching, their concept of signature analysis is embedded in much of the semiconductor manufacturing-focused research that followed, which concerned the impact of workload control in wafer fabs. Two seminal works of note, the starvation avoidance method of Glassey and Resende [100] and Wein's workload regulation technique [318], were the primary catalysts that launched a flurry of order release methods for wafer fabs. We discuss these two important papers in the next two subsections. Then we focus on more recent order release methods.

6.2.1 Starvation Avoidance

The starvation avoidance (SA) method of Glassey and Resende [100] focuses on a single bottleneck work center and calculates a virtual inventory measured over a lead time in order to regulate order release. This virtual inventory comprises all work in the BS that potentially can reach the bottleneck within the prespecified lead time. The lead time is the time required for jobs to arrive to the bottleneck the first time after order release in a single-product environment. In a multiproduct system, the worst case time among the products to arrive to the bottleneck is used.

An important challenge in the SA method is tracking jobs that are recirculating in the system as is the case in semiconductor manufacturing.

One needs to detect when each recirculating job will move again/next within the lead time window. The virtual inventory calculation also includes a mechanism to account for failed machines at the bottleneck machine group. A target value for the virtual inventory level is determined using inventory concepts and safety stock considerations. The primary idea of SA is to determine an acceptable level of risk of the bottleneck machine group running out of work, i.e., starving.

Glassey and Resende [100] compare SA to four order release methods: random starts, uniform starts, a simple input/output approach based on the idea of constant WIP, and a simplified version of the workload regulation method of Wein [318]. The workload regulation method is described in more detail in Sect. 6.2.2. Glassey and Resende [100] also present a hybrid dispatching method to boost the efficacy of SA that was subsequently enhanced and expanded by Leachman et al. [156]. In addition to the hybrid dispatching method, they use simple FIFO and SRPT dispatching (cf. Sect. 4.2.1 for the corresponding priority indices) in their study.

Unlike Wein [318], the SA study concentrated more on the order release process rather than investigating a large number of dispatching rules. In fact, Glassey and Resende [100] suggest that dispatching decisions seem to have little impact when uniform job releases are used. However, the comparisons of SA both to the simple input/output approach based on constant WIP and to Wein's method for a simple wafer fab with 12-step process flows contain no mention of statistical significance.

One of the issues practitioners can have with the SA method is that it is both conceptually and computationally more complex than other available approaches and it requires global information about the wafer fab inventories. A companion paper by Lozinski and Glassey [168] provides details on performing the necessary calculations and implementing the approach. The superiority of SA over other simpler methods has never been adequately verified. In fact, at least two subsequent attempts [54, 99] suggest the opposite conclusion. However, the concept has a strong intuitive appeal and is inherent in much of the subsequent research and development of workload control and production control software for semiconductor manufacturing.

6.2.2 Workload Regulation

The second seminal paper in semiconductor manufacturing workload control was the Wein [318] work that introduced the concept of workload regulation (WR). The workload-regulating input method for order release is conceptually similar to other bottleneck methods that compute the load destined for the bottleneck machine group and then strive to maintain this target level of loading. The WR method computes machine group load in terms of the number of hours of work in front of the bottleneck machine group

rather than expressing the workload as a number of jobs or wafers. Wein [318] introduces several modified versions of dispatching rules and introduces a new dispatching method called workload balancing.

The bottleneck-oriented order release methodology that Wein created is not that different from the OPT concept. Similarly, expressing bottleneck workload in terms of hours, rather than jobs or items, had been used previously as an input/output control variable. The primary theoretical contributions of the seminal Wein [318] paper pertain to dispatching based on workload balancing. Of potentially even more importance, however, are the following two points:

1. The WR approach taken by Wein focuses on the semiconductor manufacturing environment.
2. The examination of four order release methods, i.e., random starts, uniform starts, a version of input/output control, and a bottleneck approach, combined with a number of dispatching rules is based on a rigorous design of experiments using a significant testbed model.

The simulation study of Wein [318] compares a number of order release strategies using three variations of a realistic semiconductor wafer fab model that was developed using actual wafer fab data. Similar to Glassey and Resende [100], Wein concludes that order release with a 30–40 % change in desired performance is more important than dispatching that leads to less than 10 % change. However, Wein's statistical results reveal an important interaction between order release and dispatching decisions. Finally, both the SA and the WR studies support the conclusion that pull-based order release strategies are preferable to push-based methods.

Most major semiconductor manufacturers are aware of Wein's WR work, have embraced the WR concept philosophically, and have developed control systems around the method. This is most likely due to the fact that the information associated with and the computational requirements of implementing this approach are modest as compared to SA. While most of the information is determined from the jobs being released, one also needs to estimate the relationship between the desired workload target and BS TP. Such an estimate is often determined using a wafer fab simulation model and/or a queueing network approximation of the wafer fab.

It is interesting to note that there is no mention in either Glassey and Resende [100] or Wein [318] of how dispatching decisions at batch processes, such as diffusion ovens and wet sinks in etch, are treated. The dispatching methods cited in these studies do not seem to apply, and given the stated interaction between order release and dispatching, it follows that a similarly strong interaction may exist with batching disciplines as well.

6.2.3 Subsequent Order Release Methods

While both SA and WR were developed for single bottleneck systems, both have been extended to multiple bottleneck environments by a number of authors [17, 54, 99, 156, 166, 167]. When we consider the dispatching aspect of flow control, it follows that as order release approaches become more effective, dispatching decisions will have a diminishing effect on BS performance. However, dispatching can still be a means to assist the flow control process by smoothing input flows to bottlenecks to prevent starvation and clogging. For example, dispatching strategies may prioritize jobs for processing on a given machine that are required at a key downstream step while deprioritizing jobs for those whose downstream steps already have sufficient amounts of WIP in front of key tools. The work of Wein [318] and Leachman et al. [156] are examples of this trend.

Miller [184] describes an IBM wafer fab simulation model and its application to the study of flow control policies for reducing CT. Through effective flow control, WIP was reduced 30 % in concert with a reduction in CT of 25 %. This is even more impressive when one considers that at the same time, TP modestly increased. These reductions were achieved using a very simple closed-loop order release method similar to CONWIP. Miller [184] also concludes that when queues are reduced by better order release practices, dispatching becomes less important. A simulation study of a packaging line at IBM Bromont by Chandra and Gupta [44] uses an order release strategy similar to WR in that the release quantities of different products into the packaging line are determined so that total manufacturing lead time is minimized, subject to satisfying product demands. The release quantities are considered for the bottleneck, which happened to be the last batch station of the line.

A case study by Martin-Vega et al. [170] provides an interesting example of applying the general JIT philosophy to a photolithography area in a wafer fab. Although a mention is given to Kanban, the authors achieve WIP reduction by physically limiting and redesigning buffer spaces and by prioritizing specific operations. Leachman [154] is a suggested reference for readers desiring a discussion of production planning and scheduling practices in the semiconductor industry as well as for additional discussion of workload control implementations.

It should be noted, however, that exceptions to the viewpoint that order release, when done well, is more important than dispatching do exist. Lu et al. [169] introduce fluctuation smoothing policy-type dispatching rules (cf. Sect. 4.2.1). Their dispatching policies compare favorably with WR in Wein's same wafer fab setting in that they produce more than 10 % reductions in both the ACT and Var(CT). Given the variety of wafer fab environments, order release strategies, and dispatching approaches, the only thing that is clear is that no one specific approach or method exists that is best for all semiconductor manufacturing environments or conditions.

In order to overcome some of the performance problems associated with CONWIP and workload regulation-based rules during product mix changes, Rose [264] introduces the constant load rule, CONLOAD. It is claimed that while pull-based approaches are capable of maintaining appropriate inventory levels in a wafer fab based on the current BS status, they unfortunately can suffer from not comprehending the wafer fab's current and/or desired product mix. By taking into account the associated additional load that is placed on a single machine or a group of machines due to a pending order release decision, more informed order release decisions can be made based on a desired bottleneck machine group loading threshold. This threshold is calculated as the product of the desired bottleneck machine group's utilization and the number of machines in the bottleneck machine group. A simulation-based study concludes that CONLOAD outperforms CONWIP, a workload regulation-based approach called CONWORK, and a simple push methodology in terms of producing and maintaining a desired level of bottleneck machine group utilization while providing a smooth evolution of fab WIP over time. An additional study by Rose [266] reveals that CONWIP-based order release methods can help to reduce the variability in both WIP and CT. However, it is confirmed that this reduced variability may come at the price of increased mean values of both WIP and CT.

Later, Bahaji and Kuhl [16] present multiobjective composite dispatching rules for both an application-specific integrated circuit (ASIC) wafer fab and a low-mix, high-volume wafer fab. The proposed composite dispatching rules utilize a combination of values based on current BS and individual job status, such as the processing time of a job, the job's arrival time at the current process step, the amount of work present in queue at the job's next process step, and a job's accumulated CT as compared to its theoretical processing time, i.e., the job's current flow factor. The authors conduct a rigorous statistical analysis of both wafer fab environments using an AutoSched AP simulation model for five different performance measures of interest based on MASM lab testbed dataset 5 (cf. Fowler and Robinson [83] for a description of these models). After analyzing four proposed approaches and ten competing methods from the literature, Bahaji and Kuhl [16] find that their composite dispatching approaches outperform both fixed-interval push order release and a CONWIP policy in terms of producing superior ACT, the lowest amount of variability in CT, and meeting required job due dates.

Finally, Qi et al. [250] examine the impact of production control methodologies and other BS factors on both the ACT and VAR(CT), as well as average lateness, WIP, and wafer fab output, at a Chartered Semiconductor wafer fab. A full factorial design of experiments that examines three order release methodologies in concert with three dispatching rules and three greedy batching policies reveals that the proposed WIPLOAD control (WIPLCtrl) job release methodology nicely balances fab performance across all of the performance measures of interest. In addition, a Markov process-based analysis of the behavior of WIPLCtrl using a model of a transfer line

system is presented in [251]. After defining WIPLOAD as the sum of the remaining processing times of all jobs in the BS, Qi et al. [250] introduce a control policy that only releases new jobs into the wafer fab when some desired reference WIPLOAD level is not being met. This reference level may be prescribed by the wafer fab's manufacturing manager according to some desired level of TP. The AutoSched AP simulation model of the Chartered Semiconductor wafer fab contained many realistic BS factors such as machine breakdowns. Thorough experimentation conducted suggests the efficacy of their WIPLCtrl approach for a variety of wafer fab output levels.

6.3 Interaction of Order Release and Scheduling

In this section, we start by discussing the scheduling heuristic and the order release approaches used. Then we describe the experimental setting and present computational results. Finally, we discuss some conclusions from the interaction study.

6.3.1 Scheduling Approach and Order Release Schemes

Given this background on the evolution and importance of order release in wafer fabs, we now investigate the influence of three order release strategies on the performance of a popular job shop scheduling heuristic. Order release schemes and scheduling are usually treated independently. There is only little known on the interaction of order release schemes and sophisticated scheduling approaches. The interaction of a scheduling approach and an order release scheme is discussed in a sequence-dependent setup situation by Ashby and Uzsoy [11].

As described in Sect. 5.4, the SBH is a decomposition-based heuristic that solves the job shop scheduling problem iteratively by solving a sequence of machine scheduling subproblems and then determines the overall shop schedule via a disjunctive graph. Mason et al. [172] modify the SBH for complex job shops as exemplified by semiconductor wafer fabs. Batch-processing machines and reentrant process flows are modeled by adding additional arcs to the disjunctive graph. In turn, unfortunately, the size of the graph increases significantly with a large scheduling horizon $h := \tau_\Delta + \tau_{ah}$, and as a result, runtime performance can be poor and software application memory requirements can be large.

To effectively investigate the interaction of order release and scheduling, we consider the two-layer, distributed approach for wafer fab scheduling DSBH that is described in Sect. 5.4.6. We use an order pool to collect jobs released for production prior to their release to the BS as a new ingredient. The overall situation is shown in Fig. 6.2.

Within DSBH, the SBH is applied separately for each work area due to the decoupling effect of the top ICA layer. Clearly, the performance of the DSBH

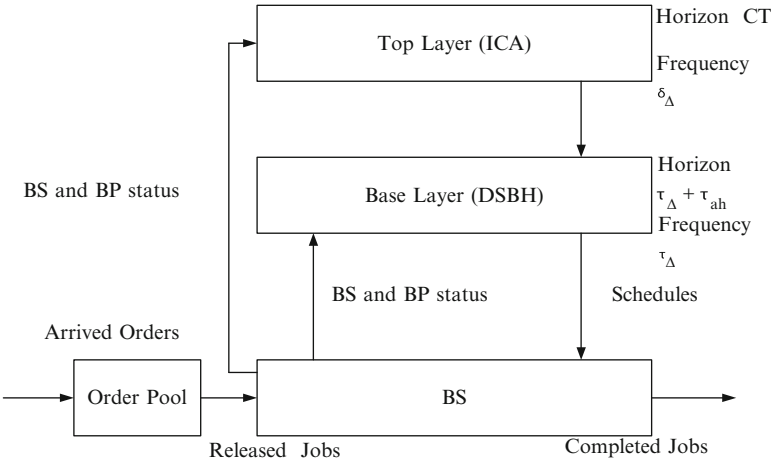

Figure 6.2: Interaction of DSBH and order release

can be improved if a schedule for one or more work areas already exists. This leads to the IDSBH scheme as described in Sect. 5.4.6. Given this background, we use the DSBH approach to investigate the interaction between the push, CONWIP, and CONLOAD order release strategies and scheduling.

The push strategy releases jobs into the BS as required by customer due dates. Only simple capacity considerations are taken into account during job release, and the release time r_j for job j is calculated by a simple backward calculation based on some desired flow factor $\mathrm{FF} \geq 1$:

$$r_j := d_j - \mathrm{FF} \sum_{i=1}^{n_j} p_{ji}. \tag{6.1}$$

The CONWIP order release strategy requires a characteristic curve of the wafer fab that provides the relationship between WIP and the production rate of the wafer fab, i.e., number of jobs produced/output per day. Once the WIP level corresponding to the desired production rate is determined, this amount of WIP is set as the CONWIP quantity. Then, a new job is released into the fab each time a job completes its processing such that the target WIP level is achieved. Finally, the CONLOAD strategy also requires the use of a characteristic curve. The workload of the wafer fab is measured as the sum of the processing times at each remaining process step for all released jobs. We obtain

$$\mathrm{WL} := \frac{1}{n\ \mathrm{CT}} \sum_{j=1}^{n} \sum_{i=k_j+1}^{n_j} p_{ji}, \tag{6.2}$$

where we assume that job j has completed all of its processing through process step k_j and that target cycle time is given by CT. In addition, the total number of jobs released into the fab that have not yet completed their processing is denoted as n. It is easy to see that Eq. (6.2) reveals that $\mathrm{WL} \in [0,1]$ when $\mathrm{CT} := \mathrm{FF}\sum_{i=1}^{n_j} p_{ji}$ is used.

6.3.2 Experimental Setting and Computational Results

We use the simulation framework described in Sect. 3.3.2 to analyze the interaction of order release and scheduling for a simulation model that is derived from the MiniFab model (cf. the description in Fig. 3.4). The new model contains three work areas. Each of them contains the machinery of the MiniFab model. The process flows are organized into two mask layers.

We focus on different performance measures of interest. The ACT and AWT measures are considered. In addition, we use TP for the wafer fab within the simulation horizon T that is defined in this situation as follows:

$$TP := |\{j|0 \le r_j, C_j < T\}|. \tag{6.3}$$

We also consider the average WIP in jobs during the simulation horizon as a performance measure. In addition to the three order release strategies described, we also vary the loading of the BS, the distribution of job weights, and the desired wafer fab FF used in setting job due dates. We compare the performance of the DSBH to FIFO dispatching using two different weight distributions for jobs in terms of the probability that a given job will have a specific weight value. We have

$$D_1 := \begin{cases} w_j = 1,\ p_1 = 0.5 \\ w_j = 5,\ p_2 = 0.35 \\ w_j = 10,\ p_3 = 0.15 \end{cases} \tag{6.4}$$

and

$$D_2 := \begin{cases} w_j = 1,\ p_1 = 0.5 \\ w_j = 2,\ p_2 = 0.45. \\ w_j = 10,\ p_3 = 0.05 \end{cases} \tag{6.5}$$

The two weight distributions differ in that D_1 has a small number of jobs that have a high weight and a large number of jobs that have a medium weight as compared to D_2. Distribution D_2 represents a wafer fab in which a very small portion of the jobs have a high weight and the remaining jobs have a small weight.

Figure 6.3 shows the relationship between WIP and wafer fab TP for our simulation model of interest. From our initial simulation runs, we see that the DSBH leads to a higher WIP level for a fixed TP value than pure FIFO dispatching does. Based on the relationships displayed in Fig. 6.3, we define specific TP levels of interest. For example, we use $\lambda_1 = 14$ jobs per day in our

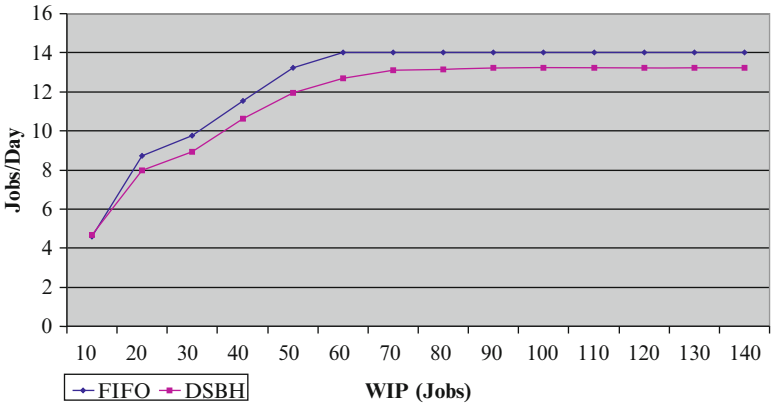

Figure 6.3: Simulated characteristic curve

model as the TP rate obtained by a WIP of 80 jobs. We refer to this situation as high wafer fab loading. Furthermore, a WIP of 60 jobs leads to a TP rate of 13.3 jobs per day in a moderately loaded wafer fab, while the low load case, i.e., 40 jobs in WIP, leads to a TP rate of 11.5 jobs per day. Additionally, we obtain a very highly loaded BS by increasing the job release rate that leads to a highly loaded BS. We use $\mathrm{WL} = 0.76$ for the highly loaded case and $\mathrm{WL} = 0.78$ for the very highly loaded case for the CONLOAD strategy. In this situation, we simply set the target CT as the raw processing time, i.e., the sum of the processing time of all process steps of a job. Finally, we use these desired wafer fab TP rates as the job release rates for the push order release strategy.

For each performance measure of interest, we compute the performance ratio of the DSBH-obtained result to the result derived by pure FIFO dispatching. In this way, any performance ratio greater (less) than one denotes superior DSBH performance for objectives that we wish to maximize (minimize). Of the four performance measures of interest, the only one that we wish to maximize is TP. Otherwise, we seek to minimize AWT, ACT, and WIP.

In all experiments, we simulate 180 days of wafer fab operations once an appropriate amount of warm-up time has elapsed to initialize the wafer fab. We do not consider any machine failures in our experimentation and employ a scheduling time horizon of $h = 2$ and $\tau_{ah} = 0\,\mathrm{h}$. Tables 6.1 and 6.2 present the results for the model comparison of DSBH scheduling and FIFO dispatching under all three order release strategies for the high and very high load cases.

We use P, CW, and CL for abbreviation for the push, CONWIP, and CONLOAD order release schemes, respectively. In the case of a highly loaded wafer fab, the FIFO dispatched system is stable, while the DSBH results suggest increasing WIP levels, which consequently produce large CT values.

Table 6.1: Computational results for AWT and ACT

			AWT			ACT		
Load	Weight	FF	P	CW	CL	P	CW	CL
High	D_1	1.3	0.98	0.65	0.70	1.56	1.12	1.18
		1.5	1.11	0.59	0.57	1.61	1.14	1.10
		1.7	1.82	0.62	0.43	1.82	1.15	1.04
	D_2	1.3	1.00	0.83	1.02	1.51	1.11	1.21
		1.5	1.61	1.41	0.87	1.47	1.11	1.11
		1.7	2.35	0.97	0.71	1.52	1.12	1.04
Very high	D_1	1.3	0.39	0.44	0.62	1.06	1.02	1.19
		1.5	0.33	0.37	0.48	1.06	1.02	1.12
		1.7	0.35	0.32	0.46	1.14	1.02	1.12
	D_2	1.3	0.58	0.60	0.85	1.04	1.01	1.18
		1.5	0.58	0.58	0.70	1.06	1.02	1.08
		1.7	0.64	0.54	0.62	1.12	1.01	1.04

Table 6.2: Computational results for TP and WIP

			TP			WIP		
Load	Weight	FF	P	CW	CL	P	CW	CL
High	D_1	1.3	0.98	0.97	0.96	1.60	1.06	1.07
		1.5	0.97	0.96	0.95	1.81	1.06	0.93
		1.7	0.96	0.95	0.94	2.08	1.08	0.87
	D_2	1.3	0.98	0.98	0.98	1.50	1.04	1.14
		1.5	0.98	0.98	0.97	1.44	1.01	1.05
		1.7	0.98	0.97	0.96	1.63	1.03	1.07
Very high	D_1	1.3	0.98	0.98	0.97	1.14	0.99	1.12
		1.5	0.98	0.96	0.96	0.99	0.98	1.14
		1.7	0.96	0.97	0.95	1.07	0.97	0.85
	D_2	1.3	0.98	0.99	0.97	0.93	0.99	1.12
		1.5	0.98	0.98	0.98	0.93	0.98	1.02
		1.7	0.98	0.98	0.97	0.93	0.98	1.00

The best improvement occurs in the cases of tight, i.e., FF = 1.3, and moderate due dates, i.e., FF = 1.5, for the push scheme. Furthermore, we find no significant difference between the two job-weighting distribution schemes. Therefore, it appears that only in a very congested wafer fab would the use of DSBH be warranted under a push order release strategy; otherwise, FIFO dispatching is advisable.

However, in a highly loaded system, i.e., 80 jobs in WIP, the use of the DSBH in combination with CONWIP order release can lead to an AWT reduction of 30 % or more as compared to the FIFO dispatching scheme. Finally, Tables 6.1 and 6.2 suggest that the DSBH method with CONLOAD outperforms FIFO dispatching with respect to AWT in almost all situations.

It follows that it is useful to combine DSBH scheduling with a CONLOAD order release strategy in highly loaded wafer fabs. For additional experimentation and results, we refer to Mönch [191].

6.3.3 Conclusions from the Interaction Study

Our experimentation does not reveal any significant difference between the three order release strategies in the low, and moderate-loaded cases. In every case, pure FIFO dispatching outperforms the DSBH scheduling method for the experimental wafer fab model under study (see Mönch [191]). For all experimental levels of FF and job weight distribution, we find that ACT increases and TP decreases when DSBH is used. Further examination of the results confirms that this behavior is caused by low-quality scheduling decisions being produced in the DSBH subproblems.

For a highly loaded BS, the use of either CONWIP- or CONLOAD-type order release strategies in combination with DSBH scheduling appears to be quite useful for reducing AWT. Finally, the push order release strategy can be applied in conjunction with the DSBH in a very highly loaded BS to produce reductions in AWT. However, CONWIP performance is superior to that of both CONLOAD and push in the most congested wafer fab case, while CONLOAD outperforms push. Note that we only consider the case of continuous job arrivals in these experiments, i.e., newly arrived jobs are released from the order pool on a regular, fixed time interval basis such as every two, three, or four hours. However, one can investigate additional job release schemes characterized by daily or weekly release frequencies. In this situation, we expect reduced due date-based performance for both CONWIP and CONLOAD order release strategies, as the time that a job spends waiting in the BS will be shifted to waiting time in the order pool prior to being released into the fab.

In the future, it is important to investigate the connection between order release decisions and the anticipated scheduling decisions of the DSBH to allow for release of new jobs into the wafer fab based on the anticipated load at bottleneck machines caused by both newly released and current WIP jobs. The next section describes an order release case study at an actual wafer fab that further investigates the frequency and size of order releases into the wafer fab.

6.4 A Large-Scale Order Release Study

In this section, we start by describing the overall situation. We discuss the results of the release timing study. Finally, the findings of the release quantity study are presented.

6.4.1 Overall Situation

A global semiconductor manufacturer commonly releases new wafer jobs into one of its wafer fabs both during morning and evening shifts, but never during their night shift. The jobs are released one at a time, with the job releases being carefully spread out across the time period spanning the morning and evening shifts. Upper-level management at this company commissioned a simulation-based case study to examine how different job release policies could potentially impact wafer fab performance. Simulation is a popular method for conducting such case studies as a high-fidelity model can mimic wafer fab operations quite effectively without ever having any impact on current wafer fab operations and output. The simulation-based order release study investigated two specific questions. First, the study examined the impact of releasing wafer jobs into the wafer fab around the clock, i.e., during all three production shifts, as compared to the current two-shift release policy. We refer to this question as the release timing case study. Next, management was interested in understanding the impact of releasing similar products as groups, called trains, of jobs into the wafer fab rather than spreading out individual job releases over time. We refer to this question as the release quantity case study.

6.4.2 Release Timing Case Study

Consider five different job release plans, denoted as Case 1 through Case 5, that each release an equal fraction of a given week's job starting from Sunday through Saturday. The cases differ in terms of at what time(s) during each day jobs are released.

Figure 6.4 portrays the job release distribution for Case 1 along with the proportion of each day's job releases that enter the factory during 2-h time intervals.

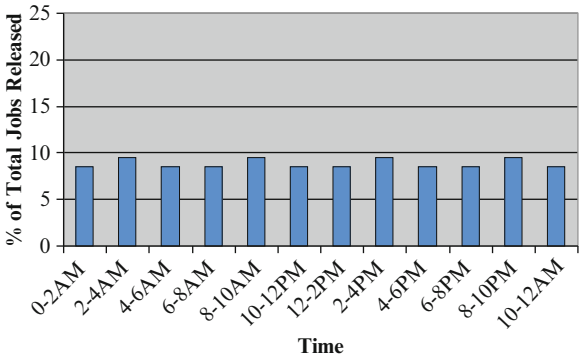

Figure 6.4: Order release distribution for Case 1

It is important to note that jobs are released individually at evenly spaced time intervals within each 2-h time block in Fig. 6.4 according to the total number of job releases planned for the time block.

In Case 2, job release only occurs during two time blocks per day. The situation is depicted in Fig. 6.5. All jobs originally released between midnight and noon in Case 1 are now scheduled for release after the morning shift change, i.e., between 6:00 and 7:00 am. Furthermore, all jobs originally scheduled for release between noon and midnight in Case 1 are rescheduled for release after the evening shift change, i.e., between 2:00 and 3:00 pm. Within each of the two job release time blocks in Fig. 6.5, individual jobs are released uniformly over time.

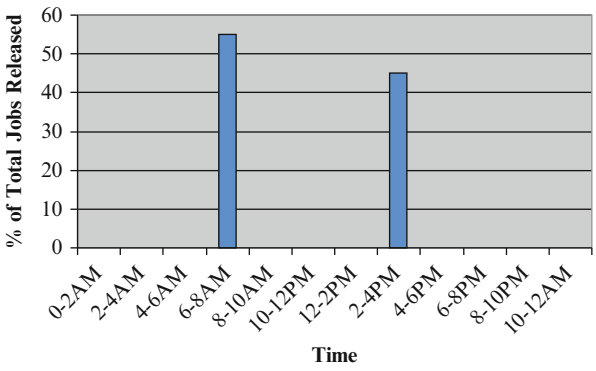

Figure 6.5: Order release distribution for Case 2

Case 3 redistributes daily job releases into equal numbers of jobs every 2 h for each day. This is shown in Fig. 6.6. Jobs are released at each even-numbered hour throughout the 24-h day, i.e., 12 times per day. As the number of jobs scheduled each day is not necessarily evenly divisible by 12, the number of jobs released at 8:00 pm and 10:00 pm will be potentially less than the other job releases to account for beginning- and end-of-day effects.

Case 4 job releases follow the semiconductor manufacturer's current order release policy in that job release occurs only during the morning and evening shifts. The policy is shown in Fig. 6.7.

Individual job releases are distributed evenly for each of these two shifts, with all jobs released during the first half of the day, i.e., between midnight and noon in Case 1, being scheduled for release at evenly spaced time intervals during the morning shift, i.e., between 6:00 am and 2:00 pm. All jobs originally scheduled for release during the second half of the day, i.e., between

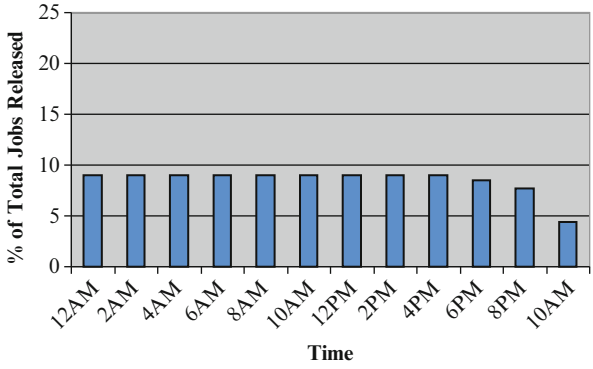

Figure 6.6: Order release distribution for Case 3

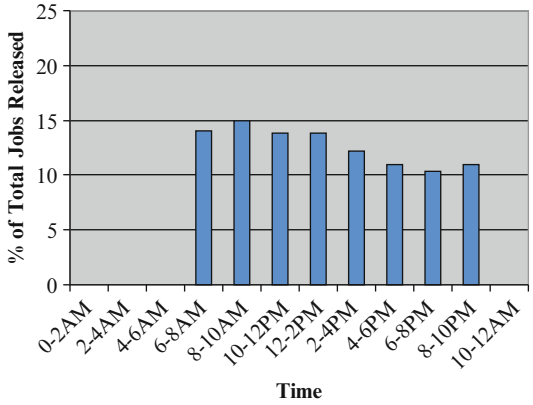

Figure 6.7: Order release distribution for Case 4

noon and midnight in Case 1, are rescheduled for release at evenly spaced time intervals during the evening shift, i.e., between 2:00 and 10:00 pm in Case 4.

Finally, Case 5 job releases occur only during the morning and evening shifts. The jobs released into the wafer fab in Case 5 are released in groups only at specific even-numbered hours during these two production shifts. This is depicted in Fig. 6.8.

The semiconductor manufacturer's validated AutoSched AP simulation model was used with representative job starts data to examine the five order release cases previously discussed. Each simulation replication was run for a period of three years, with the first year of results being discarded to mitigate any potential for initialization bias. Given the complexity inherent in the company's simulation model, each simulation run required approximately 12 h

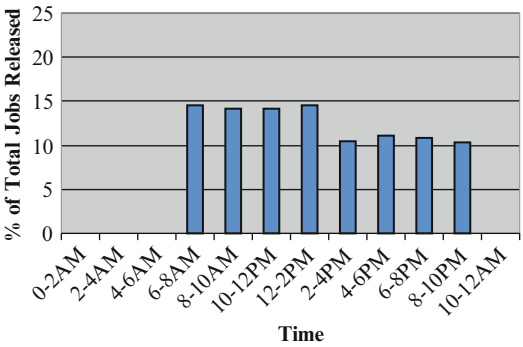

Figure 6.8: Order release distribution for Case 5

of wall clock time. In order to measure the potential for on-time delivery compliance, a due date offset equivalent to three times of each job's raw processing time was used.

A summary of all simulation replications made for the five job release cases is given in Table 6.3 in terms of TP, expressed as number of jobs completed per day; ACT, expressed as a multiple of the raw processing time, i.e., as flow factor FF; and the percentage of jobs completed on or before their due date, denoted by OTD(%). As stated previously, Case 4 is the most accurate characterization of the company's current job release policy. Case 3 was identified by the semiconductor manufacturer as the most appropriate alternative approach for comparison purposes with Case 4. Finally, Case 5 is quite similar to Case 4, except that instead of releasing jobs every m minutes during the morning and evening shifts, it only releases jobs into the wafer fab every 2 h. For these reasons, it was decided to focus the detailed results analysis on these three cases rather than on all five options.

Table 6.3: Simulation results for release timing case study

Compare	TP (Jobs)	ACT (FF theoretical)	OTD(%)
Case 1	34.808	3.180	82.400
Case 2	34.848	3.150	86.900
Case 3	34.853	3.080	92.000
Case 4	34.850	3.110	90.000
Case 5	34.854	3.130	89.100

A paired t-test analysis of Cases 3 and 4 revealed the following with 95 % confidence:

1. The ACT value of jobs in Case 3 is shorter than the ACT value of Case 4 jobs.

2. The on-time delivery performance of Case 3 is superior to that of Case 4.

Furthermore, no significant statistical difference was found between the TP of Cases 3 and 4. As Case 3 was determined to be statistically superior to Case 5 in all three performance measures of interest, the semiconductor manufacturer's study found compelling evidence to try an alternative job release strategy that was determined to have the potential to improve operational performance of its wafer fab.

6.4.3 Release Quantity Case Study

In the previous release timing case study, the jobs to be individually released into the wafer fab were furnished by the semiconductor manufacturer in a particular desired order without any regard being given to the product type of each job. In the release quantity case study, both Cases 3 and 4 are examined further by arranging the list of jobs to be released into the BS into groups, i.e., trains of multiple jobs less frequently, according to product type. A potential benefit of the train approach is that early batch process steps will be able to make fuller batches as sufficient quantities of production jobs will be available within a shorter time horizon.

Figures 6.9 and 6.10 present the release time distribution for Case 3 trains of jobs and Case 4 in the release quantity case study, respectively. The release time distributions for the trains of jobs (TofJ) of the two cases are denoted for abbreviation as TofJ3 and TofJ4, respectively.

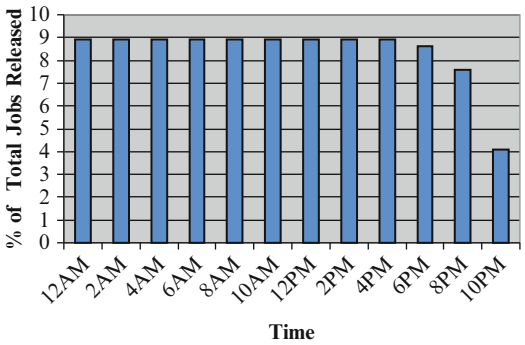

Figure 6.9: Order release distribution for Case 3 trains jobs

An initial analysis of the job trains contained in both the TofJ3 and TofJ4 simulation model inputs led the semiconductor manufacturer to suggest the establishment of an upper bound on the number of jobs that can be present

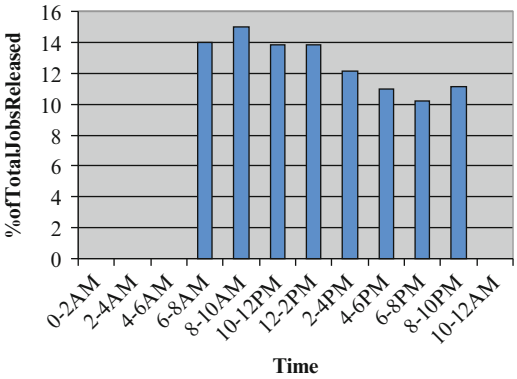

Figure 6.10: Order release distribution for Case 4 trains of jobs

in a single train. The limit of no greater than ten jobs per train was established for both cases and with the idea that a train of ten jobs or less would not unnecessarily overload any fab machine group. We use the notation TofJ3_NGT10 and TofJ4_NGT10, respectively. Finally, the job trains established for Case 4 were examined for two additional upper-bound limits on train size. The TofJ4_NGT_OvenBatch case was created to restrict Case 4's job train size to the maximum load size, measured in jobs, of the first diffusion oven process step contained in each product's process flow. Similarly, the TofJ4_NGT6 case restricts the length of the Case 4's jobs to six jobs.

As was the case in the previous case study, the semiconductor manufacturer's AutoSched AP simulation model was used, and each simulation replication was run for a period of 3 years, with the first year of results being discarded to mitigate any potential for initialization bias. Table 6.4 shows the results of all simulated cases. We note that each TofJ case has a higher ACT value and a lower OTD(%) value than the original Cases 3 and 4 except for TofJ4_NGT_OvenBatch. Although releasing jobs in trains can significantly increase CT values and reduce OTD(%) compared to current practice, the resulting performance appears to depend on the size of the trains formed. In some cases, the semiconductor manufacturer was presented with the opportunity to actually improve performance, should they choose to use oven batch-sized trains.

6.5 Optimization-Based Order Release

In addition to simulation-based methods for order release analysis/planning, mathematical optimization-based approaches can be used. Such an approach is taken, for example, by Missbauer [185]. In this section, we discuss an optimization-based approach for order release in semiconductor manufacturing.

Table 6.4: Simulation results for release quantity case study

	TP (jobs)	ACT (FF theoretical)	OTD(%)
Case 3	34.853	3.080	92.000
Case 4	34.850	3.110	90.000
TofJ3	34.858	3.240	78.200
TofJ4	34.857	3.250	76.800
TofJ3_NGT10	34.846	3.170	85.800
TofJ4_NGT10	34.854	3.180	84.700
TofJ4_NGT_OvenBatch	34.855	3.100	93.100
TofJ4_NGT6	34.855	3.250	76.500

Consider a wafer fab that is interested in planning starts for some period of time, for example, the next week. At the current time, WIP exists in the wafer fab at a variety of locations, i.e., at different process steps, with each job in WIP containing some number of wafers of a predefined technology, process, and device type. The process step at which each job is located corresponds to some process flow that the job is required to follow. During this process flow, we focus on the photolithography process steps and the loading of the potential bottleneck machines of the wafer fab, the photolithography steppers (cf. the description in Sect. 2.2.3). New job starts into the wafer fab are released to meet customer demands for a specific product. They are characterized in terms of their technology, process, and device. However, the wafer fab has potentially more than one option, i.e., job type designation, that it can make on new job starts that directly corresponds to the specific steppers that will be visited during critical photolithography layers.

The option designation for all new job starts directly impacts the wafer fab loading, as the choice of any device d's option 1, for example, can require the job to visit the first stepper two times and the second stepper six times at the eight critical layers of the process flow. However, designating device d's job release as option 2 alternately can result in five visits to the first stepper, two visits to the second stepper, and one visit to the third stepper for the eight critical layers. Simulation-based optimization is used by Mönch et al. [201] to solve a similar load-balancing problem for steppers in an ASIC wafer fab.

In this way, effective release job option designations are an important way in which the photolithography capacity can be utilized most effectively. The problem is further complicated when one considers that jobs can be released today, tomorrow, or on any other day within the desired job release horizon. However, management may dictate that specific jobs and/or specific job option designations must be started on a specific day.

We now provide a MIP formulation for the wafer release optimization problem. The only wafer fab capacity constraints being represented in this model are photolithography steppers. Without loss of generality, we use MES data to collect information on the expected CT values of all other process

steps for each process flow and model job travel through the BS in terms of daily, i.e., 24-h, movements from a given process step to the expected location of the job after 24 h of wafer fab operations.

The following indices and index sets will be used within the model:

t : technology index
p : index of process flows
d : index of devices
o : index of job type designation options for new job starts
s : index of process steps in process flow p of technology t
l : index of existing jobs in WIP that are following a specific process flow
e : index of photolithography machines
h : index of production days in the planning horizon
T : technology set
$P(t)$: set of process flows of technology t
$D(t)$: set of devices of technology t
$O(t)$: set of job type designation options for new job starts of technology t
$S(t,p)$: set of process steps in process flow p of technology t
$L(t,p)$: set of all existing jobs in WIP that are following process flow p of technology t
E : set of all steppers
H : set of all production days in the planning horizon

In addition, for ease of reading, we define $K(t,p) := \{t\} \times P(t) \times S(t,p) \times L(t,p)$, $R(t,p) := K(t,p) \times H$, and $J(t) := \{t\} \times P(t) \times D(t) \times O(t) \times H$ for abbreviation. The following parameters will be used within the model:

α : first day job starts can be made
β : last day job starts can be made
σ_{tps} : number of step s in process flow p of technology t
π_{tps} : $\begin{cases} 1, & \text{if process step } s \text{ in process flow } p \text{ of technology } t \text{ is a photolithography step} \\ 0, & \text{otherwise} \end{cases}$
ω_{tpsl} : processing rate (in wafers per hour) for fab job l at photolithography step s in process flow p of technology t
ξ_{tpsd} : processing rate (in wafers per hour) for any job of device type d at photolithography step s in process flow p of technology t
ϕ_e : number of available processing hours per day for stepper e
τ_{tpl} : number of wafers of job l in initial WIP that follow the process flow p of technology t
η_{tp} : number of the last step in process flow p of technology t
δ_{tpl} : number of the process step in process flow p of technology t at which job l is initially located

$\mu_{tp,\delta_{tpl},1}$: number of the process step in process flow p of technology t at which job l will move to 24 h after being at its initial location δ_{tpl}

γ_{tps} : number of the process step in process flow p of technology t at which any job will move to 24 h after being at process step s

$$\kappa_{tps} : \begin{cases} 1, & \text{if process step } s \text{ in process flow } p \text{ of technology } t \text{ can have WIP present during the daily job location assessment} \\ 0, & \text{otherwise} \end{cases}$$

$$\psi_{tpss'} : \begin{cases} 1, & \text{if step number } s' < \gamma_{tps} \\ 0, & \text{otherwise} \end{cases}$$

$$\varepsilon_{tplse} : \begin{cases} 1, & \text{if stepper } e \text{ is qualified to process existing job } l \text{ at process step } s \text{ in process flow } p \text{ of technology } t \\ 0, & \text{otherwise} \end{cases}$$

$$\upsilon_{tpdose} : \begin{cases} 1, & \text{if stepper } e \text{ is qualified to process job releases of device } d\text{'s option } o \text{ at process step } s \text{ in process flow } p \text{ of technology } t \\ 0, & \text{otherwise} \end{cases}$$

ρ_{tph} : number of wafers for process flow p in technology t that must be released on day h

θ_{tp} : number of wafers for process flow p of technology t to be released during some days $\alpha \leq h \leq \beta$

$$\lambda_{tpdo} : \begin{cases} 1, & \text{if jobs of device } d\text{'s option } o \text{ are qualified for release in process flow } p \text{ of technology } t \\ 0, & \text{otherwise} \end{cases}$$

The following decision variables are required in the model:

I_{tpsl} : initial WIP (in wafers) for job l at its initial step s in process flow p of technology t

M_{tpdoh} : number of wafers in WIP from a new release of device d's option o in process flow p of technology t on day h

N_{tpdsoh} : number of wafers in WIP at process step s on day h in process flow p of technology t from a new release of device d's option o

V_{tpslh} : WIP (in wafers) for job l at process step s in process flow p of technology t at the end of day h

Q_{tpselh} : total hours of workload associated with existing job l, which follows process flow p of technology t that is assigned to the stepper e at process step s on day h

$R_{tpdseoh}$: total hours of workload associated with new job releases of option o of device d, which follows process flow p of technology t that is assigned to stepper e at process step s on day h

U_{eh} : loading (utilization) of stepper e on day h
W_{tpdoh} : integer number of 25 wafer jobs of device d's option o released in process flow p of technology t on day h
X_{tpdoh} : integer number of wafers of device d's option o released in process flow p of technology t on day h
Y_h : maximum projected daily loading of any stepper on day h
Z : maximum number of job starts on any day during the starts horizon

The order release optimization model can be formulated as follows:

$$\min w_1 Z + w_2 \sum_{h \in H} Y_h \tag{6.6}$$

subject to

$$I_{tpsl} = \tau_{tpl}, \quad \left\{(t,p,s,l) \in K(t,p) | \sigma_{tps} = \delta_{tpl}\right\}, \tag{6.7}$$

$$V_{tpsl1} = I_{tp,\delta_{tpl},l}, \ \left\{(t,p,s,l) \in K(t,p) | \sigma_{tps} = \mu_{tp,\delta_{tpl},1}\right\}, \tag{6.8}$$

$$V_{tpslh} = V_{tp,\mu_{tp,\delta_{tpl},h-1},l,h-1}, \left\{(t,p,s,l) \in K(t,p), h \in H | h>1, \sigma_{tps}=\mu_{tp,\delta_{tpl},h}\right\}, \tag{6.9}$$

$$\sum_{\{e \in E, r \in S(t,p) | \pi_{tpr}=1, \ \varepsilon_{tplre}=1, \ \delta_{tpl} \le \sigma_{tpr} < \mu_{tp,\delta_{tpl},1}\}} \omega_{tprl} Q_{tprel1} \ge \sum_{\{a \in S(t,p) | \pi_{tpa}=1, \ \delta_{tpl} \le \sigma_{tpa} < \mu_{tp,\delta_{tpl},1}\}} \sum_{b=\sigma_{tpa}} \psi_{tpab} I_{tp,\delta_{tpl},l}, \tag{6.10}$$

$$\left\{(t,p,s,l) \in K(t,p), \sigma' \in \{1,\ldots,\eta_{tp}\} | \pi_{tps}=1, \sigma_{tps}=\sigma', \delta_{tpl} \le \sigma_{tps} < \mu_{tp,\delta_{tpl},1}\right\},$$
$$\sum_{\{e \in E, r \in S(t,p) | \pi_{tpr}=1, \ \mu_{tp,\delta_{tpl},h-1} \le \sigma_{tpr} < \mu_{tp,\delta_{tpl},h}, \ \varepsilon_{tplre}=1\}} \omega_{tprl} Q_{tprelh} \ge \sum_{\{a \in S(t,p) | \pi_{tpa}=1, \ \mu_{tp,\delta_{tpl},h-1} \le \sigma_{tpa} < \mu_{tp(\delta_{tpl})h}\}} \sum_{b=\sigma_{tpsa}} \psi_{tpab} V_{tp,\mu_{tp,\delta_{tpl},h-1},l,h-1}, \tag{6.11}$$

$$\left\{(t,p,s,l,h) \in R(t,p), \sigma' | h>1, \pi_{tps}=1, \sigma_{tps}=\sigma' \le \eta_{tp}, \mu_{tp,\delta_{tpl},h-1} \le \sigma_{tps} < \mu_{tp,\delta_{tpl},h}\right\},$$

$$M_{tpdoh} = 25 W_{tpdoh}, \ \left\{(t,p,d,o,h) \in J(t) | t \in T, h \le \beta, \lambda_{tpdo}=1\right\}, \tag{6.12}$$

$$N_{tpdsoh} = M_{tpdoh}, \left\{(t,p,d,o,h) \in J(t), s \in S(t,p) | \sigma_{tps}=1, h \le \beta, \lambda_{tpdo}=1\right\}, \tag{6.13}$$

$$N_{tpdsoh} = \sum_{\{r \in S(t,p) | \kappa_{tp,\sigma_{tpr}}=1, \sigma_{tpr} \le \gamma_{pr}\}} N_{tpdro,h-1},$$

$$\{(t,p,d,o,t) \in J(t), s \in S(t,p) | t \in T, \kappa_{tp,\sigma_{tps}} = 1, \sigma_{tps} > 1, \lambda_{tpdo} = 1\}, \quad (6.14)$$

$$\sum_{\{e \in E | v_{tpdose}=1\}} \xi_{tpsd}\ R_{tpdseoh} \ge$$

$$\sum_{\{a \in S(t,p) | \pi_{tpa}=1,\ \sigma_{tpa} < \gamma_{tps1},\ \kappa_{tp,\sigma_{tpa}}=1\}} \{b | b=\sigma_{tpa},\ \sigma_{tpa} < \gamma_{tps},\ \sigma_{tps} < \gamma_{tpa}\} \psi_{tpab}\ N_{tpdao,h-1},$$

$$\{(t,p,d,o,h) \in J(t), s \in S(t,p) | t \in T, \pi_{tps} = 1, \lambda_{tpdo} = 1\}, \quad (6.15)$$

$$\phi_e U_{eh} \ge \sum_{t \in T} \sum_{p \in P(t)} \sum_{\{s \in S(t,p) | \pi_{tps}=1\}} \{l \in L(t,p) | \varepsilon_{tplse}=1,\ \mu_{tp,\delta_{tpl},h-1} \le \sigma_{tps} \le \mu_{tp,\delta_{tpl},h}\} Q_{tpselh}$$

$$+ \sum_{t \in T} \sum_{p \in P(t)} \sum_{d \in D(t)} \sum_{\{s \in S(t,p) | \pi_{tps}=1\}} \sum_{\{o \in O(t) | \lambda_{tpdo}=1,\ v_{tpdose}=1\}} R_{tpdseoh},$$

$$\{e \in E, h \in H\}, \quad (6.16)$$

$$\sum_{p \in P(t)} \sum_{\{o \in O(t) | \lambda_{tpdo}=1\}} X_{tpdoh} \ge \rho_{tdh},$$

$$\{t \in T,\ d \in D(t),\ h \in H | \alpha \le h \le \beta\}, \quad (6.17)$$

$$\sum_{p \in P(t)} \sum_{\{o \in O(t) | \lambda_{tpdo}=1\}} \sum_{\{h \in H | \alpha \le h \le \beta\}} X_{tpdoh} = \sum_{\{h' \in H | \alpha \le h \le \beta\}} \rho_{tdh'} + \theta_{td},$$

$$\{t \in T, d \in D(t)\}, \quad (6.18)$$

$$25 W_{tpdoh} \ge X_{tpdoh},\ \{(t,p,d,o,h) \in J(t) | t \in T, h \le \beta, \lambda_{tpdo} = 1\}, \quad (6.19)$$

$$Z \ge \sum_{t \in T} \sum_{p \in P(t)} \sum_{d \in D(t)} \sum_{\{o \in O(t) | \lambda_{tpdo}=1\}} W_{tpdoh}, \quad \{h \in H | \alpha \le h \le \beta\}, \quad (6.20)$$

$$Y_h \ge U_{eh}, \qquad \{e \in E, h \in H\}, \quad (6.21)$$

$$I_{tpsl} \ge 0, M_{tpdoh} \ge 0, N_{tpdsoh} \ge 0, V_{tpslh} \ge 0, Q_{tpselh} \ge 0, U_{eh} \ge 0, Y_h \ge 0, Z \ge 0,$$
$$\{t \in T, p \in P(t), d \in D(t), s \in S(t,p), o \in O(t), e \in E, l \in L(t,p), h \in H\}, \quad (6.22)$$

$$W_{tpdoh},\ X_{tpdoh} \in \mathbb{Z}_+, \quad \{(t,p,d,o,h) \in J(t) | t \in T\}. \quad (6.23)$$

We seek to minimize the weighted sum of the maximum daily starts on any single day and the sum of the maximum daily loading on each stepper over the entire planning horizon. Clearly, weights are required to properly balance the two objective function components in terms of their dimensionality, i.e., units of measure, along with the desired importance the user prefers to specify for each individual objective function component. The objective function that combines these two key performance measures is given by expression (6.6). In this objective function, both $w_1 \in \mathbb{R}_+$ and $w_2 \in \mathbb{R}_+$ are weights that can be specified according to the user's preference regarding the importance of each objective function component in relation to each other.

We assume that a certain number of jobs exist currently in the wafer fab. Without loss of generality, we assume that the MES or other database can be queried to ascertain the current location, i.e., process step, of each existing job in WIP as well as other job-specific attributes such as its associated technology, process, and the number of wafers in the job. Constraints (6.7) establish the value of I_{tpsl} based on the initial MES information.

Constraints (6.8) and (6.9) model the movement of the jobs existing in the initial WIP through each job's own respective process flow. Using MES data, individual process step CT values are obtained and then aggregated to establish the expected location of each job 24 h from the current time. While constraints (6.8) make this calculation based on each initial step number δ_{tpl} of each job, constraints (6.9) recursively project the rest of each job's 24 h daily movements using the same MES data based on the idea of a daily job location assessment.

Constraints (6.9) establish the daily wafer fab location in terms of the process step for each existing job during the planning horizon, while constraints (6.10) and (6.11) determine the assignment of the processing item associated with each photolithography process step to each qualified stepper by considering the wafer processing rate of each stepper at the process step. While constraints (6.10) perform this computation for the first day of the planning horizon, constraints (6.11) recursively compute this quantity for all subsequent days of the planning horizon. In this way, the total hours required to complete each existing job at each photolithography process step are completely allocated to one or more steppers. Therefore, rather than assigning a specific stepper to process a specific existing job at a given process step, we ensure that the total workload associated with the process step is allocated to one or more steppers. This approach allows for reducing the complexity of the model as typical binary assignment decision variables are not required.

Constraints (6.12) use the primary decision variable W_{tpdoh} for the number of jobs released into the wafer fab of a given type on a given day to establish the number of new wafer starts by a qualified technology-process-device option combination each day over the starts horizon for all new jobs released into the fab. The number of wafers released found in constraints (6.12) is subsequently used to establish the initial WIP at the first process step of each valid process flow in constraints (6.13).

Next, the initial WIP at the first process step of each valid process flow as determined in constraints (6.13) for the new wafer starts is recursively projected to where it is expected to move based on CT data from the MES that is mapped into a parameter for every day in the planning horizon, i.e., daily job location assessment, in constraints (6.14).

Constraints (6.14) establish the daily fab location in terms of the process step for each new job release during the planning horizon, and constraints (6.15) determine the assignment of the processing time associated with each photolithography step to each qualified stepper by considering the device-specific processing rate of each stepper at the process step. This calculation is analogous to the one in constraints (6.11), which focuses on existing jobs in the wafer fab.

With constraints (6.10), (6.11), and (6.15), all qualified steppers have some amount of assigned workload for each photolithography process step on a day of the planning horizon. Constraints (6.16) sum up all of these workload requirements and compute individual daily stepper loading percentages based on the available hours per day for each stepper.

If it is specified that some desired number of wafer starts must be started on a specific day during the starts horizon, constraints (6.17) ensure that at least this desired number of wafers is started on that day.

Finally, constraints (6.18) ensure that all starts demand is satisfied. Constraints (6.19) compute the number of 25 wafer jobs to be started over the starts horizon for each valid type of starts designation from the individual wafer starts decision variable.

Constraints (6.20) are used to compute the maximum number of job starts on any single day during the horizon in which new job starts can be made. These constraints are necessary to establish the value of one of the two objective function variables. Next, constraints (6.21) set the value of the second objective function variable, the maximum daily loading of a stepper during any day of the planning horizon.

Finally, constraints (6.22) describe the nonnegativity requirements for each decision variable, while constraints (6.23) require positive integer values for the two decision variables relating to wafer and job starts.

We now consider the case of a wafer fab's starts planner who is interested in making job release plans for the next work week. The current BS status in terms of stepper quantities, the current location of all jobs, and the anticipated wafer releases for the next week in terms of quantities and device types based on customer demand forecasts for the quarter are input quantities of the optimization model. While some of this demand is for a specific quantity of wafer releases on a specific day for one or more devices, much of the demand is general demand in the form of 100 wafers of device type D_1 that is of process P_1 of technology T_1 and should be started the next week.

Furthermore, the starts planner knows that due to an update from the photolithography process engineers, available option designations for these 100 wafers are options O_2 and O_3. All other such information is also available

and properly input into the model to develop a starts plan for the week that seeks to both:

1. Minimize the maximum daily loading of a photolithography stepper during any day of the planning horizon
2. Minimize the maximum total number of job starts that are made on any day during the next week

Table 6.5 presents example optimization model results detailing the job release plan for the upcoming week at the wafer fab.

Table 6.5: Example of weekly job starts recommended by Model (6.6)–(6.23)

Technology	Process	Device	Option	Day number	Jobs to start
AA	AAQ	7C65	O73	6	1
AA	AAQ	7C69	O73	1	1
AA	L8C	7AA6	O83	5	1
ALP11	S8DI	8C24	O72	1	1
ALP11	S8DI	8C25	O81	5	1
ALP11	S8DI	8C25	O83	3	4
ALP11	S8DIN	8F26	O82	7	1
ALP11	S8Q	8C38	O72	4	1
ALP11	S8Q	8C39	O83	3	1
ALP11	S8TMC	8C27	O83	3	2

Based on these recommended starts prescribed by the model, Table 6.6 indicates the expected stepper loading for the upcoming two weeks as well. It is important to note that while job starts are being made over some planning horizon such as a week, the overall planning horizon of the model must be at least the length of the longest expected CT of any device's process flow, as it is important to properly model the transitions of both existing jobs in the wafer fab as they work their way through the wafer fab and new job releases.

Finally, as this model will be run potentially on a weekly basis to plan weekly job releases, a new BS status is imported into the model each week to ensure that any unplanned events/changes in the wafer fab over the previous week are properly accounted for in the latest model, whether it be new technologies/process flows or new stepper processing rates and/or availabilities.

The order release optimization model (6.6)–(6.23) can be expanded and customized for the needs of a specific wafer fab. For example, Cypress Semiconductor, a global semiconductor manufacturer that designs, develops, manufactures, and markets high-performance, mixed-signal, programmable solutions for a wide variety of customers, operates an 80,000-square-foot wafer fab in Bloomington, Minnesota, called Fab 4. Fab 4 uses an

Table 6.6: Example photolithography stepper loadings by day

Day	iLine1	iLine2	iLine4	jLine1	jLine2	jLine3	jLine4	Alpha1	Alpha4
1	91	74	87	84	82	87	85	78	64
2	51	72	78	75	78	60	86	59	88
3	78	89	67	71	52	51	53	60	53
4	87	56	86	69	63	78	55	54	70
5	80	67	53	54	58	70	82	68	89
6	83	88	92	55	78	63	56	89	79
7	69	81	69	77	62	90	55	79	73
8	75	88	79	67	71	85	73	85	56
9	77	86	62	82	89	53	77	68	50
10	78	52	89	58	88	60	72	77	77
11	86	78	60	78	83	80	56	72	78
12	73	85	67	79	88	76	71	88	57
13	85	86	83	86	85	82	84	52	85
14	54	51	71	70	65	60	51	58	91

optimization-based approach for planning weekly order release. Fab 4's order release model, in addition to similar functionality to this base model, contains Cypress's own proprietary, company-specific constraints and additional objective functions that allow Fab 4 to effectively load its machines under a wide array of product mix scenarios. Currently, a practical-sized instance of the MIP model (6.6)–(6.23) can be solved to within 1 % of the optimal solution in less than two minutes on a desktop computer using commercially available optimization software.

Chapter 7
Production Planning Approaches

In this chapter, we discuss production planning approaches for semiconductor manufacturing. Planning is on the highest level of the PPC hierarchy. Planning approaches provide important input for the order release schemes discussed in Chap. 6. We start by describing short-term planning approaches. Spreadsheet modeling and simulation are used in this situation.

Then, we continue by describing master planning approaches in semiconductor manufacturing. They are used to assign production quantities to different facilities in different periods of time for a horizon of several months. Weekly time periods are considered. Simulation-based performance assessment of master planning approaches is briefly discussed. Next, we discuss capacity planning approaches. In contrast to master planning, these approaches deal with a longer planning horizon and monthly time periods. We discuss only deterministic planning approaches for master and capacity planning. Then, we present enterprise-wide planning approaches. In this situation, we consider a planning horizon of several years and quarters as periods. We also deal with the question of whether or not it is beneficial to open new facilities. Deterministic and stochastic settings are described for enterprise-wide planning problems.

One typical assumption in planning approaches is a fixed CT; however, the CT is load-dependent. Therefore, we discuss different possibilities to model load-dependent CT within planning approaches. We consider CT-TP curves, iterative simulation, and finally clearing functions.

7.1 Short-Term Capacity Planning

In this section, we start by discussing the motivation of spreadsheet-based and simulation-based short-term capacity planning. We then make the first approach more concrete for wafer fabs. Spreadsheet-based short-term capacity planning approaches are discussed for back-end facilities. Finally, short-term capacity planning based on discrete-event simulation is described.

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5_7,

7.1.1 Motivation

Spreadsheet-based capacity planning models are ubiquitous. From the early days of Lotus 123 and Quattro Pro to today's Microsoft Excel-based tools, many planners and other wafer fab personnel have developed their own spreadsheet capacity model to make important decisions with regard to near-term capacity needs in the wafer fab (see Occhino [216] and Ozturk et al. [224]). While they can and do vary in size, complexity, level of detail, focus area, and accuracy or validity, spreadsheet-based tools are widely accepted methods for short-term capacity analysis in both wafer fabs and assembly and test facilities.

The typical goal of any short-term capacity planning spreadsheet model is to calculate the expected utilization of one or more machines or machine groups under some amount of demand or loading. While this utilization calculation is often needed to assess the feasibility of a proposed machine loading scenario or to justify the need for additional wafer fab equipment, the underlying mathematics take a variety of forms. The ways in which utilization is defined or calculated and then reported by wafer fab personnel often differ due to one or more modeling assumptions and/or the contingency factors used by the analyst. While spreadsheet-based short-term capacity analyses are predominantly used throughout the front-end and back-end facilities worldwide, the models quite often contain static, deterministic data inputs that are updated on some sort of periodic or as-needed basis. While these updates can be automatically made using SQL queries into corporate data sources (cf. Witte [323] for such an approach), even the most up-to-date information being included in the model will still produce only a static, deterministic estimate of machine group capacity utilization.

Unfortunately, in the absence of some fairly sophisticated queueing network analysis (which is rare in the short-term capacity planning models), spreadsheet-based capacity analysis is unable to accurately model and predict dynamic performance measures associated with the planned capacity levels, such as ACT, WIP levels, and CT variability. While this may not always be of interest to managers conducting strategic capacity analyses such as yearly or five-year plans, short-term capacity analysis often is interested in expected out dates for products/jobs currently in the manufacturing line both in the front-end and the back-end. Discrete-event simulation can help to provide a dynamic perspective for short-term capacity planning.

7.1.2 Spreadsheet-Based Approaches for Wafer Fabs

The basic approach for short-term capacity planning in wafer fabs typically requires some of the following set of machine-specific input data for each machine group w that we desire to analyze:

- Number of machines contained within machine group w, denoted by $Q(w)$.
- Percentage of time that machines in machine group w are available on average for processing wafers, denoted by $\mathrm{Av}(w)$.

- Percentage of average available time each time period that machines in machine group w are processing wafers, denoted by $\text{Eff}(w)$.
- Total number of hours per time period that machines in machine group w are scheduled for production, denoted by $\text{SH}(w)$.

Multiple, different efficiency values can also be used in place of a single parameter estimate. For example, some wafer fabs track operator efficiency, machine loading efficiency, and other measures. In this case, the collection of efficiency measures, all of which are defined from 0 % to 100 %, would be multiplied together to compute an overall efficiency measure for the machine in question. With these machine-specific inputs, the total number of expected productive hours $\text{PH}(w)$ can be computed for each machine group w as follows:

$$\text{PH}(w) := Q(w)\text{Av}(w)\text{Eff}(w)\text{SH}(w). \tag{7.1}$$

For example, an etch machine group containing eight machines, each of which is scheduled 24 h per day, seven days per week, has a historical availability due to both scheduled and unscheduled downtime events of 92 %. In addition, the corporate policy is to assume an 85 % productivity efficiency, which relates to the company's desired minimum amount of idle time on the machine, and a 90 % load efficiency, which pertains to how fully the machine is typically loaded with regard to maximum load size. In this case, we obtain:

$$\text{Eff}(w) = (0.85)(0.9) = 0.765. \tag{7.2}$$

Applying Eq. (7.1) results in the expected number of productive hours per week given by

$$\text{PH}(w) = (8)(0.92)(0.765)(168) = 945.9. \tag{7.3}$$

Once the available machine group productive hours are known, the next step is to characterize how the machine group is impacted, i.e., visited, by demand for a specific product that is made according to some specified process flow or route i, i.e., route-specific information.

Given the reentrant nature of front-end wafer fabrication processes, it is important to capture a number of inputs to properly characterize a machine group's route-specific information. These inputs should include recipe-based parameters as different recipes are visited various numbers of times in a typical manufacturing route i. In addition, since the speed or processing rate of a machine can be recipe-dependent, this too should be taken into account.

With this in mind, route-specific inputs often contain some of the following inputs for recipe r:

- Number of times the current recipe r is visited for route i, denoted by NV_{ir}.
- Rate at which recipe r processes wafers, expressed in wafers per hour, on route i, denoted by UPH_{ir}.
- Percentage of recipe r wafers that must be reworked on route i, denoted by RWP_{ir}

In the case where multiple recipes are specified for a given route, typical capacity analyses aggregate the inputs to calculate a total number of visits, as well as average UPH and RWP values. Average UPH is calculated using a visits-weighted harmonic mean. A harmonic mean is required when any time rates, i.e., some quantity per unit time, are to be averaged, such as units per hour. The weighted harmonic mean of j positive real numbers $n_1,\ldots,n_j$ associated with weights $w_1,\ldots,w_j$ is defined as follows:

$$H = \sum_{k=1}^{j} w_k \Big/ \sum_{k=1}^{j} \frac{w_k}{n_k}. \tag{7.4}$$

Assume route i has j different recipes that machine group w encounters and that recipe $l, l = 1,\ldots,j$ is visited NV_{il} times by machine group w. A visits-weighted harmonic mean is used to calculate the average UPH, denoted by AUPH, for machine group w on route i with $\sum_{l=1}^{j} \mathrm{NV}_{il}$ total visits as follows:

$$\mathrm{AUPH}_i(w) := \sum_{l=1}^{j} \mathrm{NV}_{il} \Big/ \sum_{m=1}^{j} \frac{\mathrm{NV}_{im}}{\mathrm{UPH}_{im}(w)}. \tag{7.5}$$

The average RWP, denoted by ARWP, is a visits-weighted arithmetic mean that is given by

$$\mathrm{ARWP}_i(w) := \sum_{l=1}^{j} \left(\mathrm{NV}_{il}\ \mathrm{RWP}_{il}(w)\right) \Big/ \sum_{m=1}^{j} \mathrm{NV}_{im}. \tag{7.6}$$

Given the machine- and route-specific inputs, a short-term capacity analysis can be performed to determine the maximum number of wafers that machine group w can feasibly process in some desired period of time for route i with $\sum_{l=1}^{j} \mathrm{NV}_{il}$ total visits by machine group w on route i. This is called the maximum number of wafer starts per time period (MaxWSPT) and is determined as follows:

$$\mathrm{MaxWSPT}_i(w) = \frac{\mathrm{PH}(w)\mathrm{AUPH}_i(w)(1-\mathrm{ARWP}_i(w))}{\sum_{l=1}^{j} \mathrm{NV}_{il}}. \tag{7.7}$$

It follows that a similar analysis across the different routes to which machine group w is assigned will provide a range of $\mathrm{MaxWSPT}_i(w)$ values for the different routes. From this point, performing a short-term analysis across all machine groups used on a given route i will reveal the true maximum number of wafer starts per week that each route i can feasibly accommodate in terms of available capacity. This is equal to the minimum $\mathrm{MaxWSPT}_i(w)$ value for all machine groups visited on route i.

Finally, now that machine group- and route-specific parameters are known, the demand placed on the machine, i.e., demand-specific information, is the last piece of information required to compute the machine group's utilization. This information is typically specified in terms of the following inputs:

- The number of wafer starts planned for route i in the period, denoted by PS_i.
- The number of days in the period for which the analysis is being conducted, denoted by ND

Clearly, some unit conversion may be required to convert PS_i to a weekly quantity based on ND. Once this is reconciled, machine group w's capacity loading can be calculated as follows:

$$\mathrm{CLP}(w) = \sum_{i \in \mathrm{Routes}} \frac{\mathrm{PS}_i}{\mathrm{MaxWSPW}_i(w)}. \tag{7.8}$$

It is possible, even desirable, to maximize $\mathrm{CLP}(w)$ for each machine group w to 100 %, as this quantity is related to utilizing productive hours rather than total hours. Recall that we previously used various efficiency and availability factors to derate total hours down to the expected available number of $\mathrm{PH}(w)$. Therefore, a 100 % value for $\mathrm{CLP}(w)$ does not mean the machine group is always busy, i.e., 100 % utilized, but rather that the machine group is completely utilizing all planned available productive hours. It follows that because not all machine groups are required for processing wafers on all routes, wafer fab personnel are able to quickly analyze a variety of starts scenarios in such a spreadsheet-based capacity planning tool. After analyzing corporate planning's demand statement, the capacity planning tool reports the expected machine utilization levels if such a plan were implemented. Once the user appropriately adjusts the starts plan to make it capacity-feasible, subsequent discussions are typically had with sales and marketing to see if any additional products for which capacity is available to start can be sold. If so, additional starts are analyzed within the capacity planning tool with the goal of maximizing the number of machine groups for which $\mathrm{CLP}(w)$ attains its maximum 100 % value. In this way, the corporation's overall goal of maximizing profits is pursued via the appropriate, feasible utilization of available production capacity.

The above described methodology for front-end short-term capacity planning was used at Micrel Semiconductor to provide greater visibility into the hidden factory associated with Micrel's front-end wafer fabs. This hidden factory refers to the incremental manufacturing capacity that exists within a given wafer fab that is not being realized due to a combination of misleading machine performance assumptions and suboptimal wafer starts plans. By specifying accurate, up-to-date machine performance inputs to the capacity model, Micrel's wafer fabs provide both wafer fab and corporate planners a clear view of the amount of route-specific wafer starts that can be accommodated in their wafer fab. Similarly, by comprehending current market demands and forecasted orders, Micrel's planners provide Micrel's wafer fabs with capacity-feasible wafer starts plans that maximize the manufacturing capacity within each Micrel wafer fab.

7.1.3 Spreadsheet-Based Approaches for Back-End

In contrast to front-end processes, back-end processes are characterized by linear, rather than reentrant, process flows. CT in back-end facilities is generally measured in days rather than weeks. Although back-end machine groups are typically visited only a single time during a process flow, additional complexities exist in the back-end, such as the need for auxiliary handler equipment in final test processes (cf. the description in Sect. 2.2.3) and the fact that device outs, i.e., shipments to customers, are the typical demand-specific inputs, rather than wafer starts, that make short-term capacity planning for back-end facilities non-trivial.

Front-end wafers are sent to a sorting process that evaluates each individual die's functionality and marks defective dies in an electronic wafer map. This map electronically records the good and bad dies on the entire wafer. The map travels electronically to the assembly area with the wafer, and the assembly equipment reads the map such that it knows what good dies to assemble and then send on to final test. Back-end short-term capacity planning is complicated by the fact that hundreds of wafers from the front-end turn into hundreds of thousands of individual dies that are to be packaged as functional ICs, memory products, communications modules, or other products.

Spreadsheet-based tools are prevalent across back-end facilities. Similar to the front-end discussion in Sect. 7.1.2, back-end capacity planning requires machine-specific, route-specific, and demand-specific inputs. While the machine-specific input parameters are quite similar, back-end processes for electrical testing of individual dies, for example, require slightly different route-specific information. Furthermore, back-end demand-specific information may be specified in a variety of units of measure, such as wafers for the sort process or thousands of dies for the assembly and test processes. However, the same approach can be taken in order to estimate equipment utilization and/or capacity loading.

Consider the final electrical testing phase of the back-end process, a common bottleneck operation. The equipment associated with this step includes not only the tester but an accompanying handler at a minimum and potentially a load board (cf. Sect. 2.2.3). Route-specific inputs for a short-term capacity analysis for the electrical test of die d may include the following:

- The amount of time (in seconds) required for the tester to locate/navigate to the die being tested, denoted by $\mathrm{IT}(d)$
- The amount of time (in seconds) required to test an individual die, denoted by $\mathrm{TT}(d)$
- The number of test programs that an individual die must undergo, for example, room temperature test, elevated temperature test, etc., denoted by $I(d)$

- The number of locations on the die to be tested, denoted by $S(d)$
- The probability that a die being tested functions properly, i.e., yield, denoted by $Y(d)$

Consider device d that is to be electrically tested. If we assume $O(d)$ individual die outs are required by customers for device d, then the total number of test insertions required is calculated as

$$\mathrm{TI}(d) := O(d)I(d)/Y(d). \tag{7.9}$$

Now that the total number of test insertions is determined, the total amount of tester and handler time required to electrically test device d, denoted by total test hours (TTH), is calculated as

$$\mathrm{TTH}(d) := \mathrm{TI}(d)\,(\mathrm{IT}(d) + \mathrm{TT}(d))\,S(d)/3600. \tag{7.10}$$

In Eq. (7.10), the 3,600 value in the denominator is used to convert seconds into hours. This total number of hours required for final testing to produce $O(d)$ good customer units out of device d would then be summed up with all other devices that require similar back-end equipment in order to compute the capacity loading for each machine group and handler group using the previously defined machine- and handler-specific inputs.

An interesting reality in short-term capacity planning for back-end facilities is the comprehension of both tester and handler capacity requirements. Consider the following resource requirements resulting from a capacity analysis:

- Device LM001 requires 27.5 h of tester T_1 and handler H_1 time.
- Device JWF223 requires 42.5 h of tester T_1 and handler H_2 time.
- Device SJM11 requires 30.0 h of tester T_2 and handler H_1 time.

Clearly, this product mix results in a total of $27.5 + 42.5 = 70.0$ h of required tester T_1 time and 30.0 h of tester T_2 time. However, in terms of the handling resources, a total of $27.5 + 30.0 = 57.5$ h of handler H_1 time is required, in addition to 42.5 h of handler H_2 time. In this case, assuming a 24-h work day and an 85 % efficiency factor, the required number of resources needed to produce these desired device outs in a single day would be calculated as follows:

- Number of required testers T_1: $\lceil \frac{70}{24(0.85)} \rceil = 4$.
- Number of required testers T_2: $\lceil \frac{30}{24(0.85)} \rceil = 2$.
- Number of required handlers H_1: $\lceil \frac{57.5}{24(0.85)} \rceil = 3$.
- Number of required handlers H_2: $\lceil \frac{42.5}{24(0.85)} \rceil = 3$.

Therefore, the utilization of the tester and handler resources must be carefully computed as above so that accurate short-term capacity planning is performed that produces effective estimates of capacity loading and/or resource utilization. This is necessary when one considers the fact that an insufficient

amount of tester or handler resources can limit back-end test capacity. Although typically not the case, additional auxiliary resources, such as load boards and/or operators, can also limit capacity. If this is potentially the case in the back-end facility being analyzed, then a similar analysis should be performed on those resources as well, as they also can be modeled in terms of machine-, route-, and demand-specific inputs. We note that all the calculations described in this section can be performed by spreadsheets.

7.1.4 An Integrated Approach Using Simulation

Often, a result of short-term capacity analysis is the expected out dates for products/jobs currently in the BS both in the front-end and back-end. Discrete-event simulation can help to provide a dynamic perspective for short-term capacity planning. While calculating such a WIP Flush projection can be done in a spreadsheet using historical estimates for expected process step CT, many analysts have turned to discrete-event simulation methods as they are designed to accommodate many of the uncertain realities present in wafer fabs that spreadsheet models do not readily model, such as machine failures (cf. Sect. 3.2.8).

While spreadsheet-based models do include machine availability assumptions, stating that a machine is available 92 % of the time, for example, is simply a high-level (but necessary) assumption. This is different from the capability provided in simulation models to specify both machine TTF and TTR distributions. The same 92 % availability can be modeled as mean TTF of 100 h and mean TTR of 8 h, for example, in a simulation model when TTF and TTR are assumed to be exponentially distributed with rate parameter $\lambda = 0.010$ and $\lambda = 0.125$, respectively.

A validated simulation model of the wafer fab can be automatically populated with the current WIP at each process step and the status of each machine group such that a short-term WIP Flush analysis can be conducted to estimate the day and time one or more jobs of interest are expected to exit the BS. This can be especially useful when customers are calling to ask when their requested products will be available to them. In addition, wafer fabs sometimes perform WIP Flush runs to estimate if they will be able to make their quarterly shipment goals to their back-end facilities.

It is important to note that many of the inputs required to build a valid simulation model can also be found in spreadsheet-based capacity analysis models, and as such, one powerful technique for performing short-term capacity analysis studies is an integrated approach that utilizes the strengths of both methods. First, the spreadsheet model can be used to accurately determine resource levels in terms of number of machines, operators, and/or other capacitated resources that are required to make some desired quantity of goods. The resulting resource levels can then be fed into the simulation model, along with process flow, equipment, and demand information, and this proposed BS and BP configuration can be simulated to ascertain the

resulting dynamic performance of the wafer fab in terms of CT and WIP levels.

While this integrated approach has proven quite valuable, an additional level of model utility can be achieved when the results of the simulation runs are used to change some of the underlying assumptions and/or inputs contained in the spreadsheet capacity analysis model. The analyst can fine-tune the desired performance of the wafer fab under study by using both modeling approaches in this interactive fashion. This can be especially important considering the ability of a spreadsheet model to compute required investment levels regarding new equipment acquisition and personnel hiring decisions. By using both spreadsheet- and simulation-based short-term capacity analysis methods, a greater level of insight and understanding may be afforded to the analyst conducting the study.

7.2 Master Planning

Master planning (MP) is somewhere between short-term capacity planning and more strategic capacity planning. It deals with determining appropriate wafer quantities for several products, several production sites, and several periods of time.

A master plan typically has a horizon of six months divided into weekly time buckets. Since market demand is not entirely known when planning a couple of weeks or months ahead, we have to distinguish between firm customer orders and additional forecasts. Explicit customer requirements are confirmed, postponed, or reduced by the order management process based on available supply. On the other hand, the demand planning process performed every month by sales and marketing departments tries to foresee the rest of the market needs. Both are main inputs of MP (see Vieira [313]).

In the following, we describe a model for master planning as proposed by Ponsignon and Mönch [245]. The resultant model is called MPSC for abbreviation. We start by presenting the related index information:

$p = 1, \ldots, P$: product index
$t = 1, \ldots, T$: time index
$k = 0, \ldots, k_{\max}$: index for measuring capacity consumption
$m = 1, \ldots, m_{\max}$: facility index
$b = 1, \ldots, b_{m,\max}$: bottleneck index for facility m

We assume that P products can be processed in $m_{\max}$ facilities consisting of $\mathrm{ih}_{\max}$ in-house locations and $\mathrm{sc}_{\max}$ subcontractor sites. The total number of bottleneck work centers associated with all facilities is represented by $b_{\max}$. We assume that each bottleneck is assigned to exactly one facility and that each facility has at least one bottleneck. This assumption is reasonable because planned bottlenecks, caused by very expensive machines, exist in all wafer fabs. Clearly, $\sum_{m=1}^{m_{\max}} b_{m,\max} = b_{\max}$ holds. In case of subcontractors, we model only one bottleneck, i.e., we set $b_{m,\max} = 1$. The quantity T stands for the planning horizon measured in periods. We use one week as the length of

a single time bucket. We assume for simplicity reasons that all products have the same cycle time of $k_{\max}+1$ weeks.

The following parameters are part of MPSC:

B_{p0} : initial backlog of product p at the beginning of the first period
$C_{mbt}^{\min}$: minimum utilization of bottleneck b in facility m in period t (in hours or pieces)
$C_{mbt}^{\max}$: maximum available capacity of bottleneck b in facility m in period t (in hours or pieces)
cc_{pmbk} : capacity consumption of one wafer of product p when this product is processed in facility m at bottleneck b and the completion period is k periods ahead
$d_{pt}^{(fc)}$: additional forecast demands for product p at the end of period t
$d_{pt}^{(o)}$: confirmed orders for product p at the end of period t
hc_{pt} : inventory cost for holding one wafer of product p during period t
I_{p0} : initial inventory level of product p at the beginning of the first period
lc_{pmt} : location cost when product p is processed in facility m in period t, i.e., fixed costs
mc_{pmt} : cost to produce one wafer of product p in facility m in period t, i.e., variable costs
rev_{pt} : expected revenue per wafer for satisfying additional demands of product p in period t
udc_{pt} : cost due to unmet confirmed orders for one wafer of product p postponed from period t to period $t+1$
$x_{pmt}^{(i)}$: initial number of wafers of product p to be completed at the end of period t in facility m, i.e., WIP started before the first period of the model
α : large number

The following decision variables are used within the model:

x_{pmt} : number of wafers of product p to be completed at the end of period t in facility m
$s_{pt}^{(fc)}$: sales quantity of additional forecast demand of product p in period t
$s_{pt}^{(o)}$: sales quantity of confirmed orders of product p in period t
B_{pt} : backlog of confirmed orders of product p at the end of period t
I_{pt} : inventory level of product p at the end of period t
u_{pmt} : binary indicator variable for occurrence of fixed production costs of product p in facility m in period t

The model can be formulated as follows:

$$\max \sum_{p=1}^{P} \sum_{t=1}^{T} \left\{ rev_{pt} s_{pt}^{(fc)} - hc_{pt} I_{pt} - udc_{pt} B_{pt} - \sum_{m=1}^{m_{\max}} (mc_{pmt} x_{pmt} + lc_{pmt} u_{pmt}) \right\} \tag{7.11}$$

subject to:

$$I_{p,t-1} - s_{pt}^{(o)} - s_{pt}^{(fc)} + \sum_{m=1}^{m_{\max}} \left(x_{pmt} + x_{pmt}^{(i)} \right) = I_{pt},\ p = 1,\ldots,P,\ t = 1,\ldots,T, \quad (7.12)$$

$$s_{pt}^{(o)} + B_{pt} = d_{pt}^{(o)} + B_{p,t-1},\ p = 1,\ldots,P,\ t = 1,\ldots,T, \quad (7.13)$$

$$s_{pt}^{(fc)} \le d_{pt}^{(fc)},\ p = 1,\ldots,P,\ t = 1,\ldots,T, \quad (7.14)$$

$$C_{mbt}^{\min} \le \sum_{p=1}^{P} \sum_{k=0}^{\min(k_{\max},T-t)} cc_{pmbk} \left(x_{pm,t+k} + x_{pm,t+k}^{(i)} \right) \le C_{mbt}^{\max},$$

$$m = 1,\ldots,m_{\max},\ b = 1,\ldots,b_{m,\max},\ t = 1,\ldots,T, \quad (7.15)$$

$$x_{pmt} \le \alpha u_{pmt},\ p = 1,\ldots,P,\ m = 1,\ldots,m_{\max},\ t = 1,\ldots,T, \quad (7.16)$$

$$x_{pmt} \ge 0, s_{pt}^{(o)} \ge 0, s_{pt}^{(fc)} \ge 0, I_{pt} \ge 0, B_{pt} \ge 0,\ p = 1,\ldots,P,$$

$$m = 1,\ldots,m_{\max},\ t = 1,\ldots,T, \quad (7.17)$$

$$u_{pmt} \in \{0,1\},\ p = 1,\ldots,P,\ m = 1,\ldots,m_{\max},\ t = 1,\ldots,T. \quad (7.18)$$

The objective is to maximize the overall difference between the revenues and the sum of costs. The first term in the objective function (7.11) models the revenues for fulfilling additional forecast demands. The costs for holding inventory are modeled by the second term. The third term refers to penalty costs for backlogged customer orders. The fourth and fifth terms represent variable and fixed production costs, respectively.

Constraint (7.12) represents the flow balance in every period and for every product. The inflows are the initial inventory, the production quantities, and the WIP inventory; the outflows are the sales quantities related to confirmed orders and forecasts and the ending inventory. Constraints (7.13) and (7.14) relate sales quantities to market demand. Backlog is allowed only for customer orders. In case of additional forecasts, we only consider a maximum bound. The capacity restrictions for every bottleneck in each period are defined in constraints (7.15) with minimum and maximum utilization limits. The overall loading is calculated by taking production quantities and WIP inventory of all products into account. We assume $\sum_{p=1}^{P} cc_{pmb0} > 0$ to ensure that there is at least one product p such that $\sum_{k=1}^{\min(k_{\max},T-t)} cc_{pmbk} > 0$ for all $t = 1,\ldots,T$, b, and m. Inequalities (7.16) set the binary variable u_{pmt} to 1 whenever there is a positive production for the considered product, location, and time period. On the other hand, $u_{pmt} = 0$ leads to $x_{pmt} = 0$. It makes sure that an additional facility is used only when it is necessary. Nonnegativity and binary conditions are defined by constraints (7.17) and (7.18).

It is shown in [245] that this problem is NP-hard because a knapsack problem can be reduced to a special case of it. Therefore, efficient heuristics are proposed in [245]. A product-based decomposition heuristic and a GA are described. The product-based decomposition procedure is similar to fix-and-optimize approaches in lot sizing. It can be summarized as follows.

Product-based Decomposition (PD)

1. Initialize the objective function value by $f_{\text{curr}} := 0$.
2. Sort the products with respect to the index I_p in descending order, where we define
$$I_p := \sum_{t=1}^{T} udc_{pt} d_{pt}^{(o)}, \quad p = 1, \ldots, P. \tag{7.19}$$
3. Decompose the set of all products into n disjoint subsets $P_1, \ldots, P_n$ of equal size, only the last subset might have a different size, such that products with similar I_p values are part of the same subset or in consecutive subsets.
4. Solve MPSC given by objective function (7.11) and constraints (7.12)–(7.18) for the current product subset P_i by taking the actual maximum capacity limits into account and by setting the minimum capacity bounds to zero. Increment f_{curr} with the objective value of the current subproblem.
5. Decrease the maximum capacity limits as follows:
$$C_{mbt}^{\max} := C_{mbt}^{\max} - \sum_{p \in P_i} \sum_{k=0}^{\min(k_{\max}, T-t)} cc_{pmbk} \left(x_{pm,t+k} + x_{pm,t+k}^{(i)} \right) \tag{7.20}$$
for each $m = 1, \ldots, m_{\max}$, $b = 1, \ldots, b_{m,\max}$, and $t = 1, \ldots, T$.
6. As long as any product subset has not been considered, increment the index i of the current product subset P_i and go to step 4, else return f_{curr}.

In step 3, the quantity n is determined by some preliminary computational experiments in such a way that the subproblems in step 4 can be solved to optimality by a MIP solver.

We see that the minimum capacity limit is ignored in the PD algorithm. Consider that a minimum utilization threshold leads to an artificial increase of production quantities for products of the first subset. As a result, the remaining capacity may not be sufficient for other subsets. That is why an a posteriori repair scheme where the bottleneck usage in each time period is checked and increased in the case that the minimum bound is not met is proposed by Ponsignon and Mönch [245].

Some computational results for $P \in \{50, 100, 200\}$ and $m_{\max} \in \{8, 12\}$ are shown in Table 7.1. We provide the ratio of the objective function values obtained by PD and by the MIP. The total number of considered problem instances is 120. The MIP is solved using the commercial solver CPLEX. The number of products within each subproblem is four, i.e., we have $n := \lfloor P/4 \rfloor$. The maximum computing time for the MIP is 30 min per problem instance, while the average computing time of PD for $P = 50$, $P = 100$, and $P = 200$ is 10, 15, and finally 30 min, respectively.

We can see from Table 7.1 that the MIP gap increases quickly when the number of products gets larger. Up to 100 products, PD behaves similar to the MIP, but PD clearly outperforms the MIP for $P = 200$.

Table 7.1: Computational results for MPSC

P	ih_{max}	sc_{max}	PD/MIP ratio	Average MIP gap
50	6	2	0.9775	0.0318
50	8	4	0.9753	0.0185
100	6	2	0.9824	0.1441
100	8	4	0.9762	0.0786
200	6	2	1.1343	0.7361
200	8	4	1.1090	0.4008

More computational results, including results for the proposed GA, can be found in [245]. Note that the GA is faster than PD, especially for large-scale problem instances. However, PD usually performs better from a solution quality point of view. Some computational results using heuristics for master planning in a rolling horizon setting can be found in Ponsignon and Mönch [244]. An architecture similar to those described in Sect. 3.3.2 is used. Feedback from the BS and the BP is taken into account with respect to backlog, inventories, and capacity each time an MPSC instance is solved.

7.3 Capacity Planning

In contrast to master planning, the planning horizon for capacity is usually one to three years. Capacity planning is therefore mid-term or long-term. Instead of weeks, usually months or even quarters are used as periods. Continuous decision variables are generally appropriate for production quantities. However, integer-valued decision variables come into play, when capacity expansion decisions are considered by purchasing new machines.

In the following, we present a multi-period capacity planning formulation that is due to Barahona et al. [22]. We start by introducing the following indices and sets that are used within the model:

$j = 1,\ldots,J$: operation index
$i = 1\ldots,I$: machine group index
$I(j)$: set of all machine groups that can perform operation j
$J(i)$: set of all operations that can be performed on machine group i
PT : set of primary machine groups
ST : set of secondary machine groups
$t = 1,\ldots,T$: period index
p : product index
P : set of all products

The following parameters are used within the model:

γ_{pt} : expected number of wafers completed per wafer started for product p in period t
d_{pt} : demand in wafers per day for product p in period t

b_{jpt} : number of passes, adjusted for yield, of operation j on product p in period t
μ_{it} : initial capacity for machine group i in hours/day in period t
c_{it} : unit capacity for machine group i in hours/day in period t
h_{ijt} : number of hours to process one wafer through operation j on machine group i in period t
m_{it} : cost of purchasing a new machine group i in period t
β_t : total budget available for buying new machines in period t
α_{pt} : upper bound for the unmet demand in wafers per day for product p in period t
q_1 : penalty for buying a primary tool
q_2 : penalty for buying a secondary tool

The following decision variables are used in the model:

U_{pt} : unmet demand for product p in wafers per day in period t
W_{pt} : number of wafers per day for product p that enter the wafer fab in period t
O_{jit} : number of wafers per day that require operation j on machine group i in period t
N_{it} : number of new machines bought for machine group i in period t

The capacity planning model can be formulated as follows:

$$\min \sum_{t=1}^{T} \sum_{p \in P} U_{pt} + \sum_{t=1}^{T} \left(q_1 \sum_{i \in PT} N_{it} + q_2 \sum_{i \in ST} N_{it} \right) \tag{7.21}$$

subject to:

$$\gamma_{pt} W_{pt} + U_{pt} = d_{pt}, \quad t = 1, \ldots, T, \ p \in P, \tag{7.22}$$

$$\sum_{p \in P} b_{jpt} W_{pt} = \sum_{i \in I(j)} O_{jit}, \quad j = 1, \ldots, J, \ t = 1, \ldots, T, \tag{7.23}$$

$$\sum_{j \in J(i)} h_{ijt} O_{jit} \leq \mu_{it} + c_{it} \sum_{\tau=1}^{t} N_{i\tau}, \quad t = 1, \ldots, T, \ i = 1, \ldots, I, \tag{7.24}$$

$$\sum_{i=1}^{I} m_{it} N_{it} \leq \beta_t, \quad t = 1, \ldots, T, \tag{7.25}$$

$$U_{pt} \leq \alpha_{pt}, \quad p \in P, \ t = 1, \ldots, T, \tag{7.26}$$

$$U_{pt} \geq 0, W_{pt} \geq 0, O_{jit} \geq 0, \ t = 1, \ldots, T, \ p \in P, \ i = 1, \ldots, I, \tag{7.27}$$

$$N_{it} \in \mathbb{N}, i = 1, \ldots, I, \quad t = 1, \ldots, T. \tag{7.28}$$

The objective (7.21) of the model is minimizing the sum of the total unmet demand and two penalty terms that discourage purchasing primary and secondary machines, respectively. Constraints (7.22) relate the demand for each product to unmet demand and production quantities. It is assumed that the

demand of all periods is satisfied with production from the same period. This assumption is reasonable because inventory is generally kept very low in semiconductor manufacturing. Note that $0 \leq \gamma_{pt} < 1$ models the occurrence of yield in semiconductor manufacturing. Constraints (7.23) determine the total number of wafers that require a specific operation distributed over all possible machines as the sum of the corresponding production levels. Constraints (7.24) ensure that the total production load on machine group i is smaller than the available capacity for this machine group measured in hours per day of production in a specific period. Budget constraints for purchasing new machines are given by constraints (7.25). Upper bounds for the unmet demand are set by constraints (7.26). Finally, constraints (7.27) and (7.28) express the fact that all decision variables are non-negative and that N_{it} is an integer for each machine group and each period.

Note that a two-stage stochastic programming formulation for a situation similar to that covered in model (7.21)–(7.28) is provided by Hood et al. [117] and Barahona et al. [22]. The first stage deals with capacity expansion decisions. The second stage is related to production decisions that can be made when the demand profile is known with certainty. Several demand scenarios with associated probabilities are provided to tackle the two-stage model similar to that described in Sect. 3.2.4. However, additional difficulties have to be resolved that are imposed by the integrality requirements for variable N_{it}.

We continue by briefly discussing the capacity optimization planning system (CAPS). CAPS is a decision-support system used by IBM for strategic planning of its semiconductor capacity (see Bermon and Hood [24]). It is based on linear programming. CAPS determines the product mix that maximizes profit given the existing machine capacity. At the same time, it is also able to determine the necessary capacity taking a given product mix into account. The unrelated parallel machines that are typical for wafer fabs are modeled in detail in the LP to determine a preferential order in which these machine groups are used.

While capacity planning for a single wafer fab is addressed in model (7.21)–(7.28) and by the CAPS model, a multi-facility situation is covered by the model proposed by Habla and Mönch [113]. The model is somewhat similar to the master planning model described in Sect. 7.2; however, assignment decisions to single wafer fabs are not taken. Consequently, integer-valued decision variables are not necessary. The objective is to maximize revenue for forecasted orders and minimize at the same time production costs, inventory holding costs, and costs for unmet committed orders. A quite general product structure is assumed to allow for modeling make-to-stock, assemble-to-order, and make-to-order production.

A detailed survey of strategic capacity planning approaches in semiconductor manufacturing is presented by Geng and Jiang [97]. A stochastic programming model for capacity planning in wafer fabs with uncertain demand and capacity is described by Geng et al. [98].

7.4 Enterprise-Wide Planning

Enterprise-wide semiconductor planning considers the allocation of products to wafer fabs and then routing the wafers with the ICs for testing. The tested wafers are routed to where they can be cut into individual chips and put in a package. The packages are then sent to final test facilities for testing and classification. The products are classified, i.e., binned, according to performance, and shipped to final inventory warehouses, or demand centers, for selling. Planning when to increase or decrease capacity at the production facilities as well as planning when and whether to build new facilities are some of the possibilities for these operations, for example, purchasing a new machine for one of the bottleneck machine groups in a wafer fab, building a new test facility in a new region, or subcontracting to a foundry. In the following, we present a MIP that is due to Stray et al. [293]. The model can be used to answer the following questions:

- What facilities should be built?
- What machines should be purchased?
- What products should be manufactured in which facilities?
- What demand should be met by subcontracting, and what demand should be left unmet in order to maximize profit?

The model is focused at a strategic level, and a typical instance of the problem covers a few years in several segments of perhaps three months per segment, i.e., quarters. The level of detail is deep enough to support decisions such as quarterly production amounts of each product in a company's product portfolio, including the routing of the product between facilities. The model does not attempt to schedule individual jobs of products within facilities. The model is presented below.

We use the following sets and indices in the model formulation:

$\mathbf{FAM}$: set of product families
$\mathbf{PKG}_p$: set of packages, where one set $\mathbf{PKG}_p$ is for each p in $\mathbf{FAM}$
$\mathbf{BIN}_{pq}$: set of bins for each product package q and family p
$\mathbf{BET}_b$: set of bins that can be sold as product with bin b characteristics
L : set of all location sets
L_F : wafer fab set
L_S : sort location set
L_M : assembly set
L_T : test set
L_D : demand center set
$\mathbf{MG}_l$: set of all machine groups in location l
p : index for product families
q : index for packages
b : index for bins

f, l : index for locations
i : index for machine types
t : index for time periods

In the remainder of this section, we use F, S, M, T, D as abbreviations for fab, sort, assembly, test, and demand center, respectively.
The following parameters are part of the model:

PBC_{lt} : cost of building facility l in period t
POC_{lt} : cost of operating facility l in period t
PRC_{lt} : cost of removing facility l in period t
MPC_{ilt} : cost of purchasing a single machine i in facility l in period t
MOC_{ilt} : cost of operating machine i in facility l through period t
SC_{pt} : cost for subcontracting one job of wafers of family p in period t
m_{il} : number of machines initially installed in machine group i and facility l
MAX_{il}^{S} : maximum number of machines allowed in machine group i in facility l
MAX_{l}^{T} : total number of machines allowed in all machine groups in facility l
α_{il} : maximum machine utilization for machine group i in location l
S_{il} : average downtime of machine group i in hours in location l over a period of length TPL
TPL : length of one period in hours
T : number of periods in the model
C_{plt} : fraction of product p in location l started in period t that finishes in period $t+1$
C_{pqlt} : fraction of product p and package q in location l started in period t that finishes in period $t+1$
Q_{plt} : yield of product p in location l in period t
Q_{fpqlt} : yield of the product p and package q in location l in period t, where f is the wafer fab in which the original wafer was manufactured
Q_{fpqblt} : resulting bins b of a product, depending on origin wafer fab f, family p, package q, location l, and time period t
G_{pq} : number of chips per wafer for family p and package q
T_{ipl} : total time product p takes to complete on machine group i in location l
D_{pqblt} : demand of a product p in package q and bin b at location l and period t
PC_{plt} : cost of starting product p at location l in period t
TC_{ldt} : transportation cost from l to d in period t
IC_{plt} : inventory cost for product p in location l and period t
PV_{pqblt} : sales price for product p in package q and bin b at demand center l in period t

PEN_{pqbt} : penalty for not meeting demand for product p in package q and bin b in period t
WLS_l : number of wafers in a job at wafer fabs and wafer sorts, by location l
CLS_l : number of chips in a job at assembly, test, and demand centers by location l
LBT_l : building time for location l
N : large number

All periods are of the same length in the model. The complementary fractions of C_{plt} and C_{pqlt} finish in period t. In order to incorporate yield at the test operations, the quantity Q_{fpqblt} is summed over q for all p, and this number has to be less than or equal to one.

The following decision variables are used within the model:

$X_{plt}^{S_1}$: number of jobs of product p to start in facility l in period t, $S_1 \in \{F,S,M,T\}$
$X_{fplt}^{S_1}$: number of jobs of product p produced in fab f to start in facility l in period t, $S_1 \in \{S,M,T\}$
X_{fpqlt}^{M} : number of jobs of product p, package q, produced in fab f to start in assembly facility l in period t
$W_{plt}^{S_1,S_2}$: number of jobs of product p to put in inventory before (B) or after (A) location l in period t, $S_1 \in \{A,B\}$, $S_2 \in \{F,S,M,T,D\}$
$W_{fplt}^{S_1,S_2}$: number of jobs of product p produced in fab f to put in inventory before (B) or after (A) location l in period t, $S_1 \in \{A,B\}$, $S_2 \in \{F,S,M,T,D\}$
$W_{fpqlt}^{S_1,S_2}$: number of jobs of product p, package q, produced in fab f to put in inventory before (B) or after (A) location l in period t, $S_1 \in \{A,B\}$, $S_2 \in \{F,S,M,T,D\}$
$W_{fpqblt}^{S_1,S_2}$: number of jobs of product p, package q, bin b, produced in fab f to put in inventory before (B) or after (A) location l in period t, $S_1 \in \{A,B\}$, $S_2 \in \{F,S,M,T,D\}$
$Y_{pldt}^{S_1,S_2}$: number of jobs of product p shipped between two locations l and d in period t, $S_1 \in L$, $S_2 \in L$
$Y_{fpldt}^{S_1,S_2}$: number of jobs of product p produced in fab f shipped between two locations l and d in period t, $S_1 \in L$, $S_2 \in L$
$Y_{fpqldt}^{S_1,S_2}$: number of jobs of product p, package q, produced in fab f shipped between two locations l and d in period t, $S_1 \in L$, $S_2 \in L$
$Y_{fpqbldt}^{S_1,S_2}$: number of jobs of product p, package q, bin b, produced in fab f shipped between two locations l and d in period t, $S_1 \in L$, $S_2 \in L$
Z_{pqbdt} : number of jobs of product p, package q, bin b, demand center d, sold in period t
ς_{pqbdt} : number of jobs of product p, package q, bin b, available at demand center d in period t

M^A_{ilt} : number of machines added to machine group i, location l, and period t
M^R_{ilt} : number of machines removed from machine group i, location l, and period t
Ω^A_{lt} : binary indicator variable for adding plant l in period t
Ω^R_{lt} : binary indicator variable for removing plant l in period t
S_{plt} : number of wafers subcontracted of each family p to each assembly operation l in each period t
M_{ilt} : number of machines in machine group i, location l, and time period t
Ω_{lt} : binary indicator variable for plant existence for location l in time period t

The objective function and the constraints of the model can be formulated as follows:

$$\begin{aligned}
&\max \sum_{t,p,q,b,d\in L_D} \mathrm{PV}_{pqbt}\ Z_{pqbdt} - \sum_{t,p,q,b,d\in L_D} \mathrm{PEN}_{pqbt}\left(D_{pqblt} - Z_{pqbdt}\right) \\
&- \sum_{t,p,l\in L_F} \left(\mathrm{PC}_{plt}\ X^F_{plt} + \mathrm{IC}_{plt}\ W^{AF}_{plt}\right) - \sum_{t,p,l\in L_F,d\in L_S} \mathrm{TC}_{ldt}\ Y^{FS}_{pldt} \\
&- \sum_{t,p,f\in L_F,l\in L_S} \mathrm{IC}_{plt}\ W^{BS}_{fplt} - \sum_{t,p,f\in L_F,l\in L_S} \left(\mathrm{PC}_{plt}\ X^S_{fplt} + \mathrm{IC}_{plt}\ W^{AS}_{fplt}\right) \\
&- \sum_{t,p,f\in L_F,l\in L_S,d\in L_M} \mathrm{TC}_{ldt}\ Y^{SM}_{fpldt} - \sum_{t,p,f\in L_F,l\in L_M} \mathrm{IC}_{plt}\ W^{BM}_{fplt} \\
&- \sum_{t,p,q,f\in L_F,l\in L_M} \left(\mathrm{PC}_{plt}\ X^M_{fpqlt} + \mathrm{IC}_{plt}\ W^{AM}_{fpqlt}\right) \\
&- \sum_{t,p,q,f\in L_F,l\in L_M,d\in L_T} \mathrm{TC}_{ldt}\ Y^{MT}_{fpqldt} - \sum_{t,p,q,f\in L_F,l\in L_T} \mathrm{IC}_{plt}\ W^{BT}_{fpqlt} \\
&- \sum_{t,p,q,f\in L_F,l\in L_T} \mathrm{PC}_{plt}\ X^T_{fpqlt} - \sum_{t,p,q,b,f\in L_F,l\in L_T} \mathrm{IC}_{plt}\ W^{AT}_{fpqblt} \\
&- \sum_{t,p,q,b,f\in L_F,l\in L_T,d\in L_D} \mathrm{TC}_{ldt}\ Y^{TD}_{fpqbldt} - \sum_{t,p,q,b,f\in L_F,l\in L_D} \mathrm{IC}_{plt} W^{BD}_{fpqblt} \\
&- \sum_{t,l\in L} \mathrm{PBC}_{lt}\Omega^A_{lt} - \sum_{t,l\in L} \mathrm{POC}_{lt}\Omega_{lt} - \sum_{t,l\in L} \mathrm{PRC}_{lt}\Omega^R_{lt} \\
&- \sum_{t,l\in L,i\in \mathrm{MG}_l} \mathrm{MPC}_{ilt}M^A_{ilt} - \sum_{t,l\in L,i\in \mathrm{MG}_l} \mathrm{MOC}_{ilt}M_{ilt} - \sum_{p,l\in L_A,t} \mathrm{SC}_{plt}S_{plt} \qquad (7.29)
\end{aligned}$$

subject to:

$$\sum_{t=1}^{r}\left\{\left(1-C_{plt}\right)Q_{plt}X^F_{plt} + C_{pl,t-1}Q_{pl,t-1}X^F_{pl,t-1} - \sum_{d\in L_S} Y^{FS}_{pldt}\right\} = W^{AF}_{plr},$$
$$p\in \mathrm{FAM},\ l\in L_F,\ r=1,\ldots,T, \qquad (7.30)$$

$$\sum_{t=1}^{r}\left(Y^{FS}_{pldt} - X^S_{lpdt}\right) = W^{BS}_{lpdr},\ l\in L_F,\ p\in \mathrm{FAM},\ d\in L_S,\ r=1,\ldots,T, \qquad (7.31)$$

$$\sum_{t=1}^{r}\left\{(1-C_{pst})Q_{fpst}X^{S}_{fpst}+C_{ps,t-1}Q_{fps,t-1}X^{S}_{fps,t-1}-\sum_{d\in L_M}Y^{SM}_{fpsdt}\right\}=W^{AS}_{fpsr},$$
$$f\in L_F,\ p\in \mathrm{FAM},\ s\in L_S,\ r=1,\ldots,T, \tag{7.32}$$

$$\sum_{t=1}^{r}\left\{\sum_{l\in L_S}Y^{SM}_{fplat}-\sum_{q\in \mathrm{PKG}_p}\frac{\mathrm{CLS}_a}{G_{pq}\mathrm{WLS}_f}X^{M}_{fpqat}\right\}=W^{BM}_{fpar},$$
$$p\in \mathrm{FAM},\ a\in L_M,\ f\in L_F,\ r=1,\ldots,T, \tag{7.33}$$

$$\sum_{t=1}^{r}\left\{(1-C_{pqat})Q_{fpqat}X^{M}_{fpqat}+C_{pqa,t-1}Q_{fpqa,t-1}X^{M}_{fpqa,t-1}-\sum_{d\in L_T}Y^{MT}_{fpqadt}\right\}$$
$$=\ W^{AM}_{fpqar},\ f\in L_F,\ p\in \mathrm{FAM},\ q\in \mathrm{PKG}_p,\ a\in L_M,\ r=1,\ldots,T, \tag{7.34}$$

$$\sum_{t=1}^{r}\left\{\sum_{l\in L_M}Y^{MT}_{fpqldt}-X^{T}_{fpqt}\right\}=W^{BT}_{fpqdr},$$
$$f\in L_F,\ p\in \mathrm{FAM},\ q\in \mathrm{PKG}_p,\ d\in L_T,\ r=1,\ldots,T, \tag{7.35}$$

$$\sum_{t=1}^{r}\left\{(1-C_{qlt})Q_{fpqblt}X^{T}_{fpqt}+Q_{fpqbl,t-1}C_{ql,t-1}X^{T}_{fpq,t-1}-\sum_{d\in L_D}Y^{TD}_{fpqbldt}\right\}$$
$$=W^{AT}_{fpqblr},\ f\in L_F,\ p\in \mathrm{FAM},\ q\in \mathrm{PKG}_p,\ b\in \mathrm{BIN}_{pq},\ l\in L_T,\ r=1,\ldots,T, \tag{7.36}$$

$$\sum_{t=1}^{r}\left\{\sum_{l\in L_T}Y^{TD}_{fpqbldt}-\varsigma_{pqbdt}\right\}=W^{BD}_{fpqbdr},$$
$$f\in L_F,\ p\in \mathrm{FAM},\ q\in \mathrm{PKG}_p,\ b\in \mathrm{BIN}_{pq},\ d\in L_D,\ r=1,\ldots,T, \tag{7.37}$$

$$\sum_{\tilde{b}\in \mathrm{BET}_b}\varsigma_{pq\tilde{b}dt}-\sum_{\tilde{b}\in \mathrm{BET}_b}Z_{pq\tilde{b}dt}\geq 0,$$
$$p\in \mathrm{FAM},\ q\in \mathrm{PKG}_p,\ b\in \mathrm{BIN}_{pq},\ d\in L_D,\ t=1,\ldots,T, \tag{7.38}$$

$$Z_{pqbdt}\leq D_{pqbdt},\ p\in \mathrm{FAM},\ q\in \mathrm{PKG}_p,\ b\in \mathrm{BIN}_{pq},\ d\in L_D,\ t=1,\ldots,T, \tag{7.39}$$

$$\sum_{p\in \mathrm{FAM}}\left\{T_{ipl}\left((1-C_{plt})X^{F}_{plt}+C_{pl,t-1}X^{F}_{pl,t-1}\right)\right\}\leq \alpha_{il}M_{ilt}(\mathrm{TPL}-S_{il}),$$
$$l\in L_F,\ i\in \mathrm{MG}_l,\ t=1,\ldots,T, \tag{7.40}$$

$$N\Omega_{lt}\geq X^{F}_{plt},\ p\in \mathrm{FAM},\ l\in L_F,\ t=1,\ldots,T, \tag{7.41}$$

$$\Omega_{lt}=\sum_{r=1}^{\max(t-\mathrm{LBT}_l,0)}\Omega^{A}_{l,r+\mathrm{LBT}_l}-\sum_{r=1}^{t}\Omega^{R}_{lr},\ t=1,\ldots,T,\ l\in L, \tag{7.42}$$

$$M_{ilt}=\sum_{r=1}^{t}(M^{A}_{ilr}-M^{R}_{ilr})+m_{il},\ i\in \mathrm{MG}_l,\ l\in L,\ t=1,\ldots,T, \tag{7.43}$$

$$M_{ilt}\leq \mathrm{MAX}^{S}_{il},\ i\in \mathrm{MG}_l,\ l\in L,\ t=1,\ldots,T, \tag{7.44}$$

$$\sum_{i\in \mathrm{MG}_l}M_{ilt}\leq \mathrm{MAX}^{T}_{l},\ l\in L,\ t=1,\ldots,T, \tag{7.45}$$

$$X^{S_1}_{plt} \geq 0, X^{S_1}_{fplt} \geq 0, X^{M}_{fpqlt} \geq 0, W^{S_1,S_2}_{plt} \geq 0, W^{S_1,S_2}_{fplt} \geq 0, W^{S_1,S_2}_{fpqlt} \geq 0, W^{S_1,S_2}_{fpqblt} \geq 0,$$
$$Y^{S_1,S_2}_{pldt} \geq 0, Y^{S_1,S_2}_{fpldt} \geq 0, Y^{S_1,S_2}_{fpqldt} \geq 0, Y^{S_1,S_2}_{fpqbldt} \geq 0, Z_{pqbdt} \geq 0, \varsigma_{pqbdt} \geq 0, S_{plt} \geq 0,$$
$$f \in L_F,\ l \in L,\ t = 1,\ldots,T,\ p \in \text{FAM},\ q \in \text{PKG}_p,\ b \in \text{BIN}_{pq}, \tag{7.46}$$
$$M_{ilt} \in \mathbb{N},\ M^A_{ilt} \in \mathbb{N},\ M^R_{ilt} \in \mathbb{N},\ t = 1,\ldots,T,\ l \in L,\ i \in \text{MG}_l, \tag{7.47}$$
$$\Omega_{lt}, \Omega^A_{lt}, \Omega^R_{lt} \in \{0,1\},\ t = 1,\ldots,T,\ l \in L. \tag{7.48}$$

The objective function (7.29) includes revenue generated from selling products, the costs of not meeting the demand, the production costs, and the costs for building and operating or removing facilities and machines. The first line of objective function (7.29) represents the revenue generated by meeting demand and the penalty for not meeting the demand. The second line indicates the wafer fab production costs, inventory carrying costs for finished wafers, and the costs for transporting wafers between the wafer fab and sort sites. The third line is related to the inventory carrying costs before sort, the products costs for sort, and the inventory carrying cost before assembly. The fourth through eighth lines represent the costs associated with the assembly and test operations, the transportation between these facilities, the transportation costs to the demand centers, and the inventory carrying costs at each of these facilities. The ninth line indicates the costs for building facilities, operating the facilities, and closing facilities, and the first two terms of the tenth line represent the costs for purchasing machines and operating them. Finally, the last term in the last line of expression (7.29) represents the costs for subcontracting the fabrication of wafers, but does not include any costs for establishing subcontract relationships.

The model contains network flow constraints, capacity constraints, product substitution constraints, demand constraints, production suppressing constraints, facility counting constraints, machine counting constraints, and constraints on the number of machines that can be purchased. The network flow constraints (7.30)–(7.37) enforce the material flow conservation, i.e., total inflow is equal to total outflow. The even-numbered network flow constraints deal with the production of products in a facility and the shipment of products to the next facility, while the odd-numbered network flow constraints deal with the balance of flow between the inflow of materials into a facility and the amount of products started for production.

Constraints (7.38) determine which products will be downgraded in order to meet demand, and the demand by product constraints are given by inequalities (7.39). Wafer fabrication is limited by constraints (7.40), and while they are not shown here, there are similar constraints sets for sort, assembly, and test.

In order to prevent production in a non-existent facility, constraints (7.41) are needed for fab production with similar constraints sets (not shown) for sort, assembly, and test. Constraints (7.42) keep track of the facilities that

are built and shutdown. In a similar way, constraints (7.43) keep track of the number of machines purchased and sold. Constraints (7.44) and (7.45) limit the number of machines in each machine group and place a limit on the total number of machines in a facility, respectively. Finally, constraints (7.46)–(7.48) are nonnegativity and integer restrictions for the decision variables.

Next, we discuss some computational results from Stray et al. [293]. We present a break-even analysis for a situation where demand ranges from very low to very high. Very low demand is related to a situation where the production facilities are running with substantial excess capacity. In the very high demand case, the production facilities are running with too little capacity. In addition, the amount of subcontracting is limited. Practically, this means setting the parameter demand level D_{pqblt} to an even level across all periods and then solving the problem. The solution is then analyzed, and the number of machines bought, wafers subcontracted, and what facilities were built are noted. The model is rerun with a different demand level, and the same characteristics are noted. After running enough problem instances with different demand levels, curves are generated and examined to see what the demand level is that makes the model go from current capacity to subcontracting, from subcontracting to buying new machines, and from buying machines to buying entire wafer fabs, assembly facilities, and test facilities. The network for this analysis is shown in Fig. 7.1.

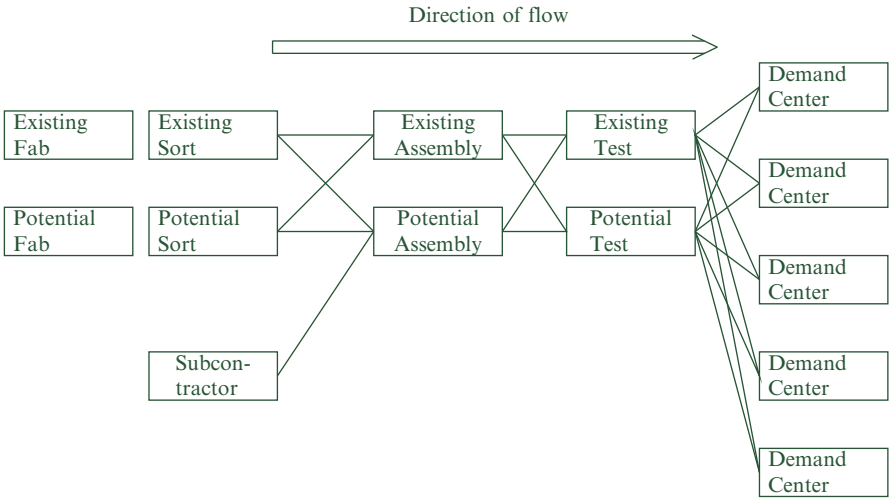

Figure 7.1: Example network

In this scenario, one wafer fab is already up and running, together with one sort, one assembly, and one test facility, and five demand centers. The capacity in this model is balanced so that the maximum capacity of each individual facility matches the maximum capacity of a single facility, preceding it in

the manufacturing supply chain. One sort facility can thus handle the output from one wafer fab, and one assembly facility can handle the output from a single sort facility.

There is also one foundry available with limited capacity. The product from the foundry is ready for the assembly operation. The price of wafers from the foundry is higher than the cost of producing them in-house as long as existing capacity is utilized. However, if a wafer fab has to be built, the cost per unit will increase. When the costs are high enough, subcontracting becomes an interesting option. All existing and potential wafer fabs have the same maximum capacity per period. The maximum capacity is defined as the capacity of a facility when the allowed maximum number of machines has been installed. Demand is varied evenly over the five demand centers.

There are two product families in the model, each divided into two packages. In the binning process, two different qualities result from each package. Wafer fabs are considered to have two bottleneck machine groups, while sort, assembly, and test have one each. The planning problem is NP-hard, because it contains knapsack- and facility location problem-type subproblems. Therefore, we will allow feasible solutions that are provably within 5 % of optimum. Figure 7.2 shows the obtained solutions for 20 different demand settings, varied along the x-axis, with subcontracting of wafers limited. For each demand scenario, the MIP was run using AMPL and CPLEX.

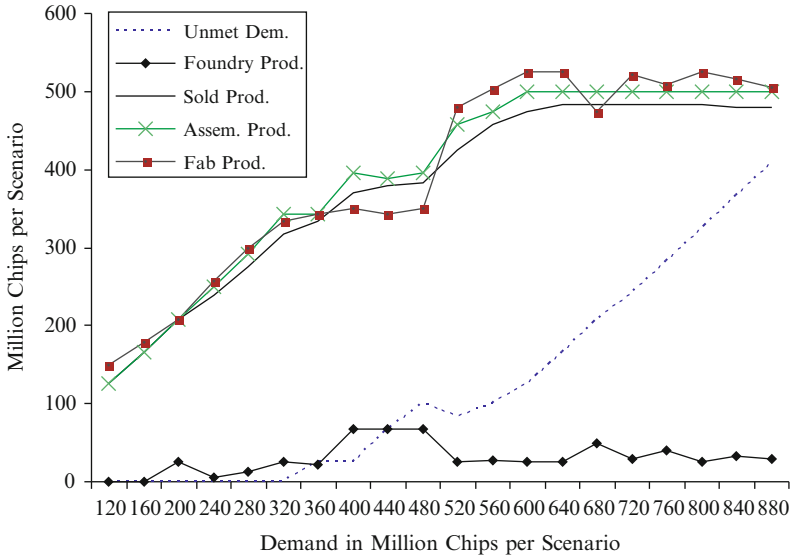

Figure 7.2: Optimal production solutions with limited subcontracting

In this analysis, a wafer fab is built at five hundred twenty million chips, two hundred million more than the maximum capacity of a wafer fab that is

three hundred twenty million chips. After the wafer fab is built, foundry production goes down and stays down comfortably under the allowed maximum. The reason why foundry production is not zero is that it covers for the production capacity that is lost during the building of the wafer fab. The dip in in-house production and increase in subcontracting at the six hundred eighty million chips demand scenario can only be explained by the 5 % allowed MIP gap, i.e., accepting the solution even though it is not optimal. The solution at six hundred forty million chips consisted of shutting down one of the wafer fabs in the last time period, replacing its capacity with subcontracting, saving the operation costs for the facility and its machines, and spending the money on the subcontracted wafers.

Rastogi et al. [258] present a stochastic version of the enterprise model of Stray et al. [293] where the total expected profit is maximized when product demand is uncertain. A two-stage, multiperiod stochastic MIP with recourse was developed to provide solutions that reduce the overall risk in planning (cf. Sect. 3.2.4). The first stage decisions include purchasing of machines at various production facilities, outsourcing production, or even construction of a new production facility depending upon the demand. The second stage (recourse) actions include increasing the internal capacity by purchasing machines at a premium as well as external capacity by subcontracting and cancellation of contracts for outsourcing made in the first stage.

The model provides information regarding the trade-offs between risk and expected short- and long-term returns. It is coded in AMPL and solved using CPLEX. When the uncertainty in demand increases, a more conservative approach is adopted, and the model displays an inherent tendency of no commitment, i.e., the capacity increment is negligible.

In addition to the uncertainty in demand, the effect of correlation between the demands of two products is studied. It is evident from the analysis, and also as stated by Simchi-Levi et al. [283], that positive correlation between the products, for example, increasing market size, involves higher risk compared to negative, for example, introduction of new products, or no correlation.

The usefulness of the model compared to the alternatives available was evaluated. The model was compared to the expected value model of Stray et al. [293] and to the perfect information case, which revealed that as the uncertainty in demand increases, the model improves its performance over the expected value model. However, the gap between the stochastic solution and perfect information solution also increases with the increment in variability of demand. By increasing the number of scenarios to map the uncertainty of demand, the results show that the efficiency of stochastic solutions increases. Adding uncertainty to the deterministic version of the model with multiple scenarios yielded more realistic and robust results, and analysis on correlation between multiple product demands resulted in unintuitive decisions for strategic make/buy problems.

7.5 Modeling of Load-Dependent Cycle Times

CT is load-dependent. It increases nonlinearly with resource utilization as known from queueing theory. This causes some problems in model formulations for production planning because CT information serves as a parameter of the models. At the same time, production planning approaches determine the load of the BS by determining release quantities. In this section, we discuss several methods to tackle this conflict. We study CT-TP curves, iterative simulation schemes, and finally clearing functions. For a detailed review of production planning models with load-dependent CT in manufacturing, we refer to Pahl et al. [226].

7.5.1 Cycle Time Throughput Curves

This section is an abridged version of Ankenman et al. [8]. CT-TP curves are often employed as decision-making tools in manufacturing settings (cf. Brown et al. [33]). A CT-TP curve displays the projected average CT plotted against TP rate, or start rate. These curves are useful for planning at both the strategic and tactical levels.

Decisions regarding the impact on CT of a 2 % increase in start rate can be widely different depending on the shape of the curve and the distance from the knee. For example, if a wafer fab has a curve as illustrated in Fig. 7.3 and is operating at the level of 22,000 wafer starts per month, it will experience only a minor change in average CT by ramping up an additional 500 wafer starts.

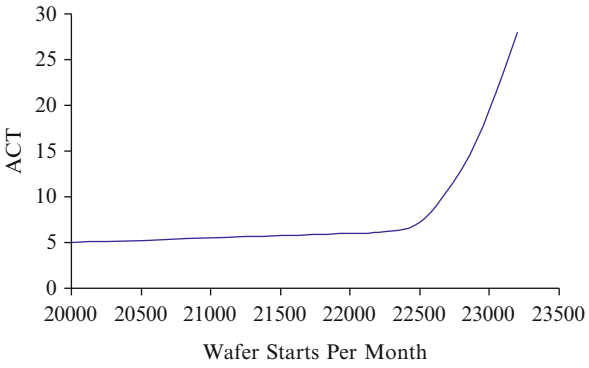

Figure 7.3: Sample CT-TP curve

Alternatively, if the wafer fab is operating on the same curve but is at 22,500 wafer starts per month, a 500-wafer start increase dramatically alters CT. In both cases, we called for a 500-wafer start increase, yet drastically different outcomes resulted from what seemed to be the same action. Man-

agement therefore needs to develop CT-TP curves if they are interested in predicting the impact of start rate changes on average CT.

Unfortunately, the simple collection and analysis of past TP history is insufficient for curve generation. It is unlikely that an operating wafer fab has experienced a sufficient number of changes along the same curve to allow creation of the curve. For example, a wafer fab seldom operates on the flat portion of the curve where equipment utilizations are in the less than 70 % range. It is also unlikely that the wafer fab has carefully ramped up production start rates over the most rapidly changing portion of the curve, so the estimation of the shape in this region becomes problematic. In fact, every time the wafer fab changes its dispatching policy or adds more equipment, it may not just be moving along the curve, it may in fact be shifting to an entirely new curve. The technique of empirical CT-TP curve generation requires the collection of large amounts of representative data. As a result, other than for the simplest of systems, simulation is the preferred method of data generation.

While simulation is the most common technique for generating CT-TP curves, the methods used to select the points to simulate and the effort to allocate to these points vary. Several different design points must be simulated to generate a CT-TP curve. A careful selection of the design points can lead to minimal simulation expense. Various authors have discussed methods for generating a CT-TP curve and how to select these design points (cf. Park et al. [230], Fowler et al. [86], and Yang et al. [325]).

Other authors have presented methods for determining an appropriate allocation of simulation effort to the design points of the CT-TP curve being simulated so as to obtain nearly equal absolute or relative precision (cf. Leach et al. [153]).

The method commonly used by practitioners to generate a CT-TP curve via simulation is to allocate an equal amount of simulation effort to each TP rate being simulated. This situation is shown in Fig. 7.4. As TP rate approaches capacity, the CT and the variance of CT (Var(CT)) increase. Figure 7.4 illustrates that by equally allocating simulation effort to all design points, yielding a CT-TP curve that is less precise as we approach capacity, a clearly undesirable characteristic. We consider single-product CT-TP curves. Note that when we say single product, we could be considering a CT-TP curve of a facility that produces only one product, or we could be focusing on one product out of many provided that the relative mix of the various products, as defined below, remains the same at all levels of the system's TP.

We define the following quantities:

$$\begin{aligned}
\lambda &:= (\lambda_1, \ldots, \lambda_K) && \text{: vector of start rates for } K \text{ products} \\
x &&& \text{: utilization of the bottleneck in the wafer fab, } 0 < x < 1 \\
\alpha &:= (\alpha_1, \ldots, \alpha_K) && \text{: product mix vector where } \alpha_k := \lambda_k / \textstyle\sum_{h=1}^{K} \lambda_h
\end{aligned}$$

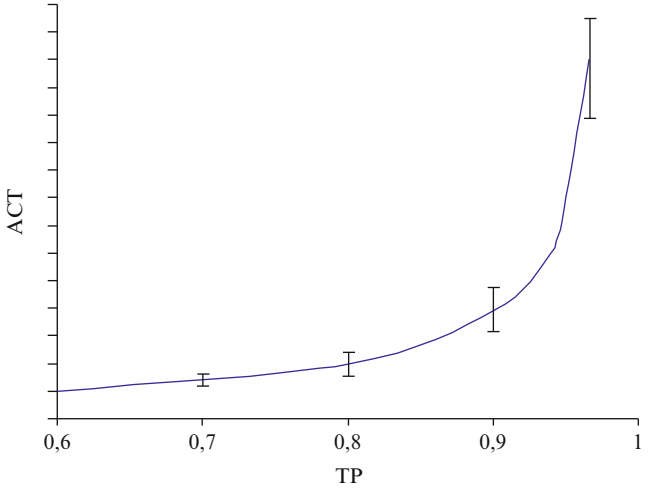

Figure 7.4: CT-TP curve using equal allocation of simulation effort

Without loss of generality, we will only consider the CT of product 1 and denote its steady-state CT as $C(\lambda) = C(x, \alpha)$, a random variable with unknown distribution that depends on the start rates. Notice that if we know the processing capacity of the bottleneck station, then specifying (x, α) is equivalent to specifying λ. We will drop the dependence on α in the single-product case. For the CT random variable to have a limiting distribution steady state, among other things, the system logic and driving inputs must not be changing over time. Generically, let

$$c_r(\lambda) = c_r(x, \alpha) := E(C^r(x, \alpha)),\ r \in \mathbb{N},\ r \geq 1 \tag{7.49}$$

be noncentral moments of the steady-state CT; we drop the subscript r when we refer to the mean, i.e., first moment.

To estimate moments of CT, we will make one or more replications of a typically large number of individual product CT values. Let $C_{ij}(x, \alpha)$ be the jth observed CT from the ith replication for $i = 1, \ldots, m(x, \alpha)$ and $j = 1, \ldots, l(x, \alpha)$. The quantity $m(x, \alpha)$ is the number of replications for given x and α, while $l(x, \alpha)$ denotes the length of a single simulation run for given x and α. Our steady state assumption corresponds to requiring that $C_{ij}(x, \alpha)$ converges in distribution to $C(x, \alpha)$ as $j \to \infty$ for any i.

The most straightforward way to generate a CT-TP curve via simulation is to select a fine grid of TP values, say $0 < x_1 < \cdots < x_d < 1$, and run simulation experiments at each one to estimate $c_r(x)$. We could, equivalently, select a grid of release rates λ that correspond to TP in a steady state. However, later when we fit CT-TP curves to the data, there are a number of advantages to standardizing TP so that system capacity always corresponds to a TP of 1.

Unfortunately, this approach has pitfalls. First, it requires a large number of simulation runs to develop a fine grid. A second repercussion of point-by-point CT-TP curve estimation is that some sort of interpolation is needed to estimate CT properties at TP values x that were not simulated. If the grid points are packed closely enough, then perhaps a simple linear interpolation is adequate. However, as mentioned earlier, exceptionally long runs may be required at the higher levels of TP, which argues against running simulations at a very fine grid.

In a series of papers, Kleijnen and van Beers (cf. Kleijnen and van Beers [143] and van Beers and Kleijnen [308, 309]) describe how the interpolation method of Kriging can be adapted to the output of discrete-event, stochastic simulations in general and queueing simulations in particular. In its simplest form, Kriging estimates $c(x)$ by a weighted average of the estimated ACT values at the grid points $x = (x_1, \ldots, x_d)$. Loosely speaking, the Kriging estimator gives more weight to CT estimates at grid points $x_h, h = 1, \ldots, d$ that are closer to the point x to be interpolated.

Because the Kriging approach is an interpolation method, it favors a finer grid, i.e., more design points x, than the queueing-motivated models we describe below. Furthermore, there is no guarantee that the Kriging estimator will exhibit known properties of the response function, for instance, that $c(x)$ is nondecreasing in x. However, the Kriging approach has the advantages that it is general purpose, it will not be subject to the lack of fit inherent in a poorly chosen meta-model, and it works largely without change for interpolating higher moments than the mean. Further, Kriging extends naturally to a multidimensional independent variable, like the product mix α.

Fowler et al. [86] investigated the use of variance reduction techniques based on common random numbers and antithetic variates (cf. the discussion in Sect. 3.3.1) in efficiently generating CT-TP curves that linearly interpolate a set of (TP, CT) points. In their paper, the term efficient reflects the capability to provide a simulation-based CT-TP curve with an acceptable precision and accuracy by using limited available resources. The goal was to generate CT-TP curves more economically, so the cost of analysis could be reduced, thus allowing companies to make better manufacturing capacity management decisions. The experimentation in their paper included simulating an M/M/1 queueing system (cf. Sect. 3.2.7) and a system with five stations in series, a special case of a Jackson queueing network. The results showed that common random numbers were effective when there was an adequate computing budget, but they introduce too much bias when the computing budget is not large enough. On the other hand, the results showed that antithetic variates were effective for small or large computing budgets.

Park et al. [230] use the D-optimality criterion (cf. Sect. 3.3.1) to choose design points for building the CT-TP curve. Since it concerns the confidence interval of the parameters of a model, D-optimality assumes that a model is specified for the response curve. Park et al. [230] suggest two nonlinear regression models, one of which is bowl shaped and is appropriate when using

batching policies such as the full batch policy MBSF or the minimum batch size policy MBS with a minimum greater than 1 (cf. Sect. 4.5 for a description of these two batching policies) that increase the CT at low levels of TP. The other model, which models the CT as a monotonically increasing function of the TP, is used when there is no batching or a greedy batch policy is employed. The two models are given below. Notice that both models have the CT exploding as the TP, x, nears the capacity β_2. If the TP is normalized to the capacity, then $\beta_2 = 1$. We obtain for the two models:

$$c(x) := \frac{\beta_1 x}{\beta_2 - x} - \beta_3, \tag{7.50}$$

$$c(x) := \frac{\beta_3}{x} + \frac{\beta_1 x}{\beta_2 - x} - \beta_4, \tag{7.51}$$

where $\beta_i \in \mathbb{R}$ are appropriate parameters of the models. Both of these models are generalizations of the CT-TP curve of a G/G/1 queue.

The experimental design for fitting these models is a selection of TP values at which exhaustive simulations are conducted and the steady state ACT value is recorded. Nonlinear regression is used to estimate the parameters, and thus the linear approximation to the variance/covariance matrix is used to approximate the D-criterion. The candidate design points are placed at regular intervals from zero TP to one, where one represents full capacity. The experimental design procedure recommended is a sequential procedure that starts with the minimum number of design points that are required to support the model, i.e., three in the case of model (7.50) and four in the case of model (7.51). The D-criterion can be expressed as a function of the location of the design points. The initial points are selected as the set of three or four in the case of model (7.51) candidate points that maximize the D-criterion. All the other candidate points are then ranked according to the D-criterion for entry into the design if needed. After simulations are conducted at the initial points, the model parameters are estimated. Each additional candidate point is added sequentially in the predefined order until the parameter estimates no longer change by an appreciable amount, i.e., 1 % was used as a stopping criterion. This method was validated through construction of a CT-TP curve for a wafer fab.

Another approach to building CT-TP curves was proposed by Cheng and Kleijnen [46], hereafter called the CK approach, where they generalized the CT-TP curve of the M/M/1 curve as shown below:

$$c(x) := f(x) \sum_{l=0}^{t} \beta_l x^l + \varepsilon(x) = \frac{\sum_{l=0}^{t} \beta_l x^l}{1 - x} + \varepsilon(x), \tag{7.52}$$

where $f(x) := 1/(1-x)$ and it is assumed that TP, represented by variable x, is scaled from zero to one. The quantity $\varepsilon(x)$ is an error term. Again $x = 1$ represents full capacity. The CK approach only deals with the case of no

batching or a greedy batch policy, and thus the model in (7.52) is a more general form of the model (7.50) proposed by Park et al. [230].

To fit model (7.52) to the simulation data, the CK approach develops a linear regression model since the only nonlinear part of the equation, $f(x)$, is known and can be dealt with through a transformation. The variance of the error term in model (7.53) depends on x as

$$\text{Var}[\varepsilon(x)] = [h(x)\sigma]^2, \tag{7.53}$$

where $h(x)$ is assumed known from asymptotic theory or other considerations.

The design of the experiment consists of the location of the design points $x = (x_1, \ldots, x_m)$ and the fraction of a total of N replications assigned to those points $\pi := (\pi_1, \ldots, \pi_m)$. The design is constructed to minimize a criterion called PM, which is a scaled version of the weighted-average variance of the estimated expected response over the TP range of interest.

The CK procedure for fitting the model (7.52) can be summarized as follows. Given $f(x)$, $h(x)$, a maximum value of t, and a fixed budget of N replications, find the optimal design (x, π) by minimizing PM. With the design points x fixed, carry out simulation experiments sequentially and adjust the allocation x. Once the total number of runs has been exhausted, use backward selection to decide the appropriate polynomial order of model (7.52) and obtain the fitted curve.

The CK method leaves open the question of how to specify $f(x)$ and $h(x)$, which affect the design of the experiment and, more importantly, the adequacy of model (7.52) to represent the true CT-TP curve. When these two functions are known, CK is highly effective and efficient, and works within a fixed budget. However, for complicated manufacturing systems, there is not likely to be sufficient information to infer such characteristics. In other words, obtaining good choices for $f(x)$ or $h(x)$, although not impossible, is difficult in practice. Further, we have strong empirical evidence from Allen [6] and Johnson et al. [132] that the $f(x)$ and $h(x)$ used by the CK method can be far from correct in realistic manufacturing simulations. Since model (7.50) used in Park et al. [230] is a specific instance of Eq. (7.52), the same weakness can be attributed to their method as well.

In summary, the procedures by Park et al. [230] and CK are both interesting and useful methods of experimental design for fitting a model such as is given in Eq. (7.52), but there may arise cases in practice where these models are not sufficiently accurate to produce useful CT-TP curves.

A precision-driven design of experiment strategy was proposed in Yang et al. [325] to sequentially build up simulation experiments for the efficient generation of CT-TP curves. It allows the user to specify a precision level and is able to provide a fitted curve with desired precision by running simulation. We summarize the method in the remainder of this section.

The estimation of the CT-TP curve is based on the two statistical regression models (7.54) and (7.55), the forms of which are both motivated by heavy

traffic queueing analysis and supported by extensive investigation of realistic manufacturing systems. One is called the expected CT (ECT) model:

$$c(x) = E[\overline{C}_i(x)] = \frac{\sum_{l=0}^{t} \beta_l x^l}{(1-x)^p}, \quad i = 1,,\ldots,m(x), \tag{7.54}$$

that characterizes the relationship between the expected CT and normalized TP x over a range of interest $[x_L, x_U]$. Unknown parameters are the polynomial coefficients $\boldsymbol{\beta}$, polynomial order t, and the exponent p. As explained earlier, the sample mean CT $\overline{C}_i(x)$ obtained from the ith simulation replication performed at x will be used as the data points to which the CT-TP models are fit. The variance of $\overline{C}_i(x)$ depends on x and is represented by the following variance model:

$$\text{Var}[\overline{C}_i(x)] = \frac{\sigma^2}{(1-x)^{2q}}. \tag{7.55}$$

Both σ^2 and q are unknown parameters. With the sample mean CT data $\{\overline{C}_i(x),\ i = 1,\ldots,m(x)\}$ at different values of x, the sample variance of $\overline{C}_i(x)$ can also be estimated over x, from which the variance model (7.55) can be fitted. With the estimated parameter $\hat{q}$, transforming the response $\overline{C}_i(x)$ by multiplying by $(1-x)^q$ will yield a constant variance and result in a standard nonlinear regression model:

$$c(x)(1-x)^q = E[\overline{C}_i(x)(1-x)^q] = (1-x)^{q-p} \sum_{l=0}^{t} \beta_l x^l = (1-x)^r \sum_{l=0}^{t} \beta_l x^l, \tag{7.56}$$

where $\boldsymbol{\beta}$, t, and the exponent r are unknown parameters. Thus, given a $\{\overline{C}_i(x),\ i = 1,\ldots,m(x)\}$ dataset, the model fitting is performed in two steps:

1. Fit the variance model (7.55) and obtain the q estimate.
2. Use the estimated parameter $\hat{q}$ to stabilize the variance for the original observations $\overline{C}_i(x)$ and then fit model (7.54).

The estimators of the ECT model (7.54) are obtained indirectly by noting that the coefficients $\boldsymbol{\beta}$ in model (7.54) coincide with those in (7.56), and p is estimated by the difference between the q and r estimates.

The goal is to obtain a precisely estimated CT-TP curve that helps manufacturers decide at what TP they should run the system. Thus, Yang et al. [325] evaluate the goodness of the fitting by the relative error achieved on the ECT response estimators. Since the curve fitting is based on the nonlinear regression performed on models (7.55) and (7.56), variance estimates can be obtained on the estimated parameters in Eqs. (7.55) and (7.56). Since model (7.54) is derived indirectly from Eqs. (7.55) and (7.56), a conservative variance estimate can be inferred for $\hat{c}(x)$, the ECT predicted at x under some empirical approximation (see Yang et al. [325]). Yang et al. [325] let the user specify a target precision, say $\gamma\%$, which is defined as the relative error on the ECT estimator:

$$\gamma\% := \frac{\sqrt{\text{Var}[\hat{c}(x)]}}{\hat{c}(x)}. \tag{7.57}$$

Once fitted curves have been obtained, the relative error on the ECT estimate $\hat{c}(x)$ can be approximated for any TP x over $[x_L, x_U]$. The user can choose to check the precision achieved at a TP level of particular interest or at a number of points in $[x_L, x_U]$ before they declare that a fitted curve with desired precision has been generated.

For the efficient estimation of the CT-TP models presented above, design of experiments methodologies is developed to collect simulation data sequentially. The experimental design consists of the location of design points, the TP levels at which simulations will be executed, the allocation of computational effort, and the number of simulation replications assigned to each design point. The best choice of experimental design depends on the true ECT and variance curves, which are unknown at the stage of designing experiments. In light of this, Yang et al. [325] approach the design of experiments problem in a sequential manner. The model curves are estimated ever more precisely as more simulation data are obtained, and further experimentation is guided by the current best estimate of the models. This design and modeling process is continued until the prespecified precision $\gamma\%$ is achieved on the ECT response estimator.

To demonstrate the effectiveness of the Yan procedure, Yang et al. [325] applied it on a number of systems to generate their corresponding CT-TP curves. The systems explored included analytically tractable queueing models and realistic semiconductor manufacturing systems. For simple queueing models such as M/M/1/FIFO, M/M/1/SPT, and M/M/1/LPT, the true CT-TP curves can be derived analytically, and hence the quality of the simulation-based model estimation can be evaluated easily. The real wafer fab considered is provided by the MASM Lab testbed (see Fowler and Robinson [83]). Since the true underlying curve is unknown in this case, nearly true ECT estimates were obtained by running simulations until the standard error of the expected CT estimates were essentially zero. These estimates provide a benchmark to which the ECT estimates obtained from the Yan method are compared. All the computational experiments show that the Yan method is able to generate high-quality CT-TP curves with desired precision. Comparisons were also performed that show that the Yan approach can be more efficient than the procedure proposed by Cheng and Kleijnen [46].

The focus of this section has been on CT as a function of TP, but product mix (PM) can also affect CTs even if the overall system TP is unchanged. However, fitting CT-TH-PM surfaces via simulation is a much more challenging problem and is beyond the scope of this section. More details of this problem are presented in Yang et al. [326].

Note that considerable simulation effort is necessary to determine meaningful CT-TP curves. These curves are valid for all possible TP situations

however, it is not evident to see how these curves can be used in some production planning approaches. In the next two sections, we will describe two more methods. The first method, iterative simulation, tries to determine CT values that are appropriate for certain released quantities. The second method, the clearing function approach, is similar to the CT-TP curve approach; however, it tries to find the relationship between load and CT and also covers the incorporation of the clearing function into production planning approaches.

7.5.2 Iterative Simulation

The first iterative procedure for production planning in semiconductor manufacturing was proposed by Hung and Leachman [120]. An LP model is formulated that requires estimated lead times, i.e., cycle times, F_{pl} for a job of product p to reach process step l after being released into the wafer fab. It is shown by Irdem et al. [124] that an unambiguous convergence of the approach of Hung and Leachman is hard to achieve. This is true even for situations where the demand is constant for all products over the planning horizon. Because of the limitations of the Hung and Leachman approach, we discuss a second formulation that when used within an iterative simulation setting shows a consistent convergence.

We describe an LP model for production planning that is due to Kim and Kim [139]. Recently, this formulation is extended to a production planning situation in semiconductor manufacturing by Irdem et al. [124]. The actual workload profiles on each machine over the planning periods are taken into account. Therefore, the effective loading ratio $e_{pk(g,t)}$ is introduced in [139]. This quantity is defined as the proportion of the start quantity of product p released in a period $g \leq t$ that contributes to the workload at machine group k in period t. Furthermore, the effective utilization u_{kt} of machine group k in period t is taken into account. The quantity u_{kt} is defined as the proportion of the total capacity of machine group k that is available to process the start quantities during period t. The adjusted capacity of machine group k in period t is obtained by simply multiplying the capacity C_{kt} by u_{kt}. It is clear that the effective loading ratios and the effective utilization can be used to model load-dependent cycle times.

Following Irdem et al. [124], the corresponding LP model is formulated. The following indices and index sets are used:

$t = 1, \ldots, T$: period index
p : product index
P : set of all products
$k = 1, \ldots, K$: machine group index
$l = 1, \ldots, l_p$: process step index for wafers of wafer type p

The parameters used within the model are:

a_{plkt} : average machine hours for process step l of a single wafer of wafer type p on machine group k processed in period t
C_{kt} : hours of machine group k available in period t
r_{pt} : unit revenue from product p in period t
c_{pt} : unit incremental production cost of product p in period t
h_{pt} : unit inventory holding cost for product p in period t
b_{pt} : unit backlog cost for product p in period t
d_{pt} : demand for wafer type p in period t
$e_{pk(g,t)}$: effective loading ratio of product p on machine group k in period t due to starts in period g
$e_{pM(g,t)}$: effective loading ratio of product p on the last processing machine in period t because of starts in period g
u_{kt} : effective utilization of machine group k in period t
B_{p0} : initial backlog for wafer type p
I_{p0} : initial inventory for wafer type p

The following decision variables are used in the model:

X_{pt} : release quantity for wafers of type p in period t
Y_{pt} : output quantity for wafers of type p in period t
I_{pt} : units of product p in inventory of finished goods at the end of period t
B_{pt} : units of product p backlogged at the end of period t

The production planning model can be formulated as follows:

$$\max \sum_{p\in P}\sum_{t=1}^{T}(r_{pt}Y_{pt}-c_{pt}X_{pt}-h_{pt}I_{pt}-b_{pt}B_{pt}) \tag{7.58}$$

subject to:

$$\sum_{p\in P}\sum_{g=1}^{t}\sum_{l=1}^{l_p} e_{pk(g,t)}a_{plkt}X_{pg} \le u_{kt}C_{kt}, \quad t=1,\ldots,T,\ k=1,\ldots,K, \tag{7.59}$$

$$Y_{pt}+I_{p,t-1}-B_{p,t-1}+B_{pt}=d_{pt}+I_{pt}, \quad p\in P,\ t=1,\ldots,T, \tag{7.60}$$

$$\sum_{g=1}^{t} e_{pM(g,t)}X_{pg}=Y_{pt}, \quad p\in P,\ t=1,\ldots,T, \tag{7.61}$$

$$X_{pt}\ge 0, I_{pt}\ge 0, B_{pt}\ge 0, \quad t=1,\ldots,T,\ p\in P. \tag{7.62}$$

The objective function (7.58) is related to profit. The objective function value is the difference of revenue and the sum of production, inventory holding, and backlog costs. Constraints (7.59) model the resource capacity, whereas constraints (7.60) are material conservation equations. The release-output relationship is expressed by constraints (7.61). Finally, nonnegativity conditions

for decision variables are taken into account by constraints (7.62). As typical in production planning models, we do not use integer variables.

The iterative procedure proposed by Kim and Kim [139] can be formulated as follows (cf. Sect. 3.2.8 for the general principle of iterative simulation). Note that we use the term KK procedure as an abbreviation.

KK Procedure

1. Initialize the counter for the current iteration $\text{curr} := 1$ and select the maximum number of iterations $\text{iter}_{\max}$. Calculate initial effective loading ratios $e_{pk(g,t),\text{curr}} := e^{(0)}_{pk(g,t)}$ and machine utilizations $u_{kt,\text{curr}} := u^{(0)}_{kt}$ using a steady state simulation. The period demands are used as release quantities within the simulation.
2. Solve the LP (7.58)–(7.62) using $e_{pk(g,t),\text{curr}}$ and $u_{kt,\text{curr}}$ to determine release quantities $X_{pt,\text{curr}}$ and wafer output quantities $Y_{pt,\text{curr}}$.
3. Use a prescribed number of independent replications of a simulation run to obtain updates for $e_{pk(g,t),\text{curr}}$ and $u_{kt,\text{curr}}$, taking the release quantities $X_{pt,\text{curr}}$ of step 2 into account. Take the average for $e_{pk(g,t)}$ and u_{kt} over all simulation runs to determine $e_{pk(g,t),\text{curr}+1}$ and $u_{kt,\text{curr}+1}$. Collect also output quantities $SY_{pt,\text{curr}}$, where S indicates that these quantities are determined from the simulation.
4. If $\text{curr} < \text{iter}_{\max}$, then set $\text{curr} := \text{curr} + 1$ and go to step 2. Otherwise, the iterative scheme terminates.

The mean absolute deviation between $Y_{pt,\text{curr}}$ and $SY_{pt,\text{curr}}$ is used to measure convergence of the KK iterative procedure. A relatively small number of iterations is generally enough. It is shown in [124] by extensive simulation experiments that the KK procedure shows a consistent convergence behavior. Hence, it seems that this procedure has some potential for being applied in practice.

7.5.3 Clearing Functions

Clearing functions are used to model the relationship between the expected output of a manufacturing system and the WIP inventory. These functions have the advantage that they are able to capture the nonlinear relationship between resource utilization, i.e., load, and CT.

Clearing functions were proposed for the first time by Graves [108]. The following linear function f is used in [108]:

$$Y_t = cW_t, \tag{7.63}$$

where $c > 0$ is a constant, called the proportional factor, and Y_t is the output at the end of period t. Finally, W_t is a measure for the WIP at the beginning of period t. An infinite capacity assumption is a consequence of this model,

because the manufacturing system is assumed to be able to complete the amount cW_t even when W_t is very large. The major drawback of this model is that the planned CT is fixed based on Little's law (cf. Eq. (3.21) in Sect. 3.2.7), even when W_t is changing. Therefore, CT values are not appropriate taken into account in model formulations that use this clearing function.

Later, nonlinear clearing functions were proposed that take the finite capacity of the BS into account. The general idea of a clearing function f is introducing a clearing factor $c(W)$ to obtain a clearing function f of the form:

$$f(W) := c(W)W, \ W \geq 0. \tag{7.64}$$

The clearing factor is a nonlinear function of the WIP W. Clearly, we have $f(0) = 0$ for each clearing function of this form.

Kamarkar [136] proposed the following clearing function:

$$f(W) := \frac{C_1 W}{C_2 + W}, \ W \geq 0, \tag{7.65}$$

where C_1 and C_2 are positive constants. This clearing function is a nondecreasing, concave function of WIP. We can see easily that C_1 is the maximal possible output that is obtained for $W \to \infty$. It represents the maximum capacity. The quantity C_2 is a user-specific parameter controlling the curvature of the clearing function. The following clearing function is due to Srinivasan et al. [292]:

$$f(W) := C_1(1 - \exp^{-C_2 W}), \ W \geq 0, \tag{7.66}$$

where again C_1 and C_2 are positive constants. By considering $W \to \infty$, we obtain that C_1 is the maximum possible output. By using the expression $\exp x = \sum_{k=0}^{\infty} x^k/k!$, it is evident that the clearing function (7.66) is of the form (7.64). Note that the two constants C_1 and C_2 can be determined, in principle, by fitting the function to empirical data.

In the remainder of this section, we assume that the clearing function f is concave and $f(0) = 0$ holds. Furthermore, it is a smooth function with the property $\frac{df(W)}{dW} > 0$, i.e., f is monotone increasing.

In the following, we want to derive an LP formulation similar to the model (7.58)–(7.62). Therefore, we have to incorporate the clearing function into the LP model. Following Asmundsson et al. [13], we replace the capacity constraints (7.59) in a single-stage multi-product situation with the product set P and T periods by

$$\sum_{p \in P} \xi_{pt} Y_{pt} \leq f_t \left(\sum_{p \in P} \xi_{pt} W_{pt} \right), \ t = 1, \ldots, T, \tag{7.67}$$

where the following notation is used:

Y_{pt} : total production quantity of product p in period t
ξ_{pt} : amount of resource (machine time) required to produce one unit of product p in period t
W_{pt} : WIP of product p in period t
f_t : clearing function for period t

As stated in [13], there is no link between the mix of WIP available in the period and the corresponding production in the capacity restriction (7.67). To avoid this problem, the overall clearing function is decomposed. We obtain:

$$\xi_{pt} Y_{pt} \leq Z_{pt} f_t \left(\sum_{p \in P} \xi_{pt} W_{pt} \right), \; p \in P, \; t = 1, \ldots, T, \tag{7.68}$$

$$\sum_{p \in P} Z_{pt} = 1, \; t = 1, \ldots, T, \tag{7.69}$$

where the new decision variable $Z_{pt} \geq 0$ represents the allocation of the expected TP represented by the clearing function among the different products.

The capacity constraints (7.68) still have the disadvantage that f_t has the total WIP as argument and not the WIP for a specific product. To solve this problem, it is assumed in [13] that the expected TP between products is proportional to the mix of products represented in the WIP in period t. We obtain

$$\sum_{p \in P} \xi_{pt} W_{pt} = \frac{\xi_{pt} W_{pt}}{Z_{pt}}, \; p \in P, \; t = 1, \ldots, T. \tag{7.70}$$

The quantity $\frac{\xi_{pt} W_{pt}}{Z_{pt}}$ can be interpreted as the extrapolated total WIP in period t. Using Little's law (cf. Eq. (3.21) in Sect. 3.2.7), it is shown in [13] that this extrapolation is exact when all products have the same average CT at the resource. The resultant capacity constraints substituting the right-hand side of Eq. (7.70) into capacity constraints (7.68) are

$$\xi_{pt} Y_{pt} \leq Z_{pt} f_t \left(\frac{\xi_{pt} W_{pt}}{Z_{pt}} \right), \; p \in P, \; t = 1, \ldots, T, \tag{7.71}$$

$$\sum_{p \in P} Z_{pt} = 1, \; t = 1, \ldots, T. \tag{7.72}$$

This is called the allocated clearing function (ACF) formulation. To obtain a tractable LP formulation, we replace the partitioned clearing function (7.71) by a set of linear constraints using outer approximations. Because f is concave, it can be approximated by the convex hull of a set of linear functions $\alpha^c \xi_{pt} W_{pt} + \beta^c$, where $c = 1, \ldots, C$ denotes the individual straight line, i.e., segment, in the approximation. The quantity α^c denotes the slope of the linearized clearing function for segment c, whereas β^c is the intercept

of the linearized clearing function for segment c. We use $\alpha^C = 0$ to model the fact that the maximum throughput is reached. In addition, we have $\alpha^C < \ldots < \alpha^2 < \alpha^1$ and $\beta^1 = 0$. An individual clearing function is assigned to each resource $k = 1, \ldots, K$. The capacity constraint is linear because we have

$$Z_{pt} f_t \left(\frac{\xi_{pt} W_{pt}}{Z_{pt}} \right) = Z_{pt} \min_c \left\{ \alpha^c \frac{\xi_{pt} W_{pt}}{Z_{pt}} + \beta^c \right\} = \min_c \left\{ \alpha^c \xi_{pt} W_{pt} + \beta^c Z_{gp} \right\}. \quad (7.73)$$

Next, we present a corresponding LP formulation following [12, 123] with products $p \in P$ and periods $t = 1, \ldots, T$. Setup times and consequently lot sizing effects are not modeled. For simplicity reasons, we model only a single stage multiproduct system; however, extensions to multistage situations are presented in [12, 123]. The following indices and index sets are used in the resultant LP:

$t = 1, \ldots, T$: period index
p : product index
P : set of all products

The parameters used in the model are:

c_{pt} : unit production cost of product p in period t
ξ_{pt} : amount of resource (machine time) required to produce one unit of product p in period t
h_{pt} : unit inventory holding cost of product p in period t
b_{pt} : unit backlog cost of product p in period t
w_{pt} : unit WIP holding cost of product p in period t
d_{pt} : demand for product p in period t
α^c : slope of the linearized clearing function at segment c
β^c : intercept of the linearized clearing function at segment c
W_{p0} : initial WIP for wafer type p
B_{p0} : initial backlog for wafer type p
I_{p0} : initial inventory for wafer type p

The following decision variables are used within the model:

X_{pt} : release quantity for wafers of type p in period t
Y_{pt} : output quantity for wafers of type p in period t
W_{pt} : WIP quantity for wafers of type p over period t
I_{pt} : units of product p in inventory of finished goods at the end of period t
B_{pt} : units of product p backlogged at the end of period t
Z_{pt} : fraction of capacity that is used by product p in period t

Now, the model can be formulated as follows:

$$\min \sum_{p \in P} \sum_{t=1}^{T} \left\{ c_{pt} Y_{pt} + h_{pt} I_{pt} + b_{pt} B_{pt} + w_{pt} W_{pt} \right\} \quad (7.74)$$

subject to:

$$W_{pt} = W_{p,t-1} - Y_{pt} + X_{pt},\ p \in P,\ t = 1,\ldots,T, \tag{7.75}$$

$$B_{p,t-1} + I_{pt} + d_{pt} = Y_{pt} + I_{p,t-1} + B_{pt},\ p \in P,\ t = 1,\ldots,T, \tag{7.76}$$

$$\xi_{pt} Y_{pt} \leq \alpha^c \xi_{pt} W_{pt} + \beta^c Z_{pt},\quad p \in P,\ t = 1,\ldots,T,\ c = 1,\ldots,C, \tag{7.77}$$

$$\sum_{p \in P} Z_{pt} = 1,\ t = 1,\ldots,T, \tag{7.78}$$

$$X_{pt} \geq 0, Y_{pt} \geq 0, W_{pt} \geq 0,\ I_{pt} \geq 0,\ B_{pt} \geq 0,\ Z_{pt} \geq 0 \quad p \in P,\ t = 1,\ldots,T. \tag{7.79}$$

The objective function (7.74) is based on the production, backorder, WIP, and finished good inventory costs. Constraints (7.75) model the WIP flow. The inventory balance equations are given by constraints (7.76). Constraints (7.77) are related to capacity represented by the linearized clearing function. Constraints (7.78) ensure that the fractions of capacity used by a single product sum up to one. Finally, nonnegativity conditions for decision variables are taken into account by constraints (7.79).

Computational experiments with the linearized ACF multi-stage approach for wafer fabs are described in [12, 123]. Job releases obtained by the ACF formulation are smoother and lead consequently to better overall CT performance compared to production planning approaches based on the fixed CT assumption.

There are several ways to derive clearing functions. The first approach consists in using steady-state or transient queueing models to determine clearing functions analytically [13]. This approach has some limitation in complex manufacturing systems such as wafer fabs. The second approach is estimating clearing functions from empirical data. The empirical data can be collected using discrete-event simulation. The overall procedure from [12, 123] can be summarized as follows.

Estimating a clearing function (ECF)

1. Generate randomly demand realizations that correspond to different bottleneck utilization levels.
2. Determine job releases using a production planning approach that takes the demand realizations from step 1 as input. Alternatively, job releases can be determined based on the demand and some simple backward calculation to obtain starting times for the jobs.
3. For each release plan from step 2, perform repeated simulation runs of the BS and BP using myopic dispatching as FIFO to determine pairs (W_t^k, Y_t^k) for each period t and each machine group k.
4. Determine C_1 for the functional forms (7.65) or (7.66) from the empirical data from step 3. Use a nonlinear least-square fitting technique to find the remaining parameter C_2.

5. Perform a piecewise linearization procedure, i.e., a nonlinear optimization, to find the segments with the corresponding slopes and intercepts (cf. Irdem [123] for details). Three segments are generally appropriate.

It is obvious that the ECF procedure is time-consuming because of the repeated simulation runs and because of running the different optimization procedures.

So far, a clearing function is constructed for each resource separately. The resulting output capacity is allocated to the different products. A different approach is proposed by Kacar and Uzsoy [134]. A clearing function is estimated for each product based on the release quantities and WIP levels of this product and other products in a certain number of periods using multiple regression.

In conclusion, it seems that estimating clearing functions from empirical data is far from being a trivial task. The computational methods and the resultant effort are similar to the case of CT-TP curves.

Overall, it seems that, from a real-world implementation point of view, the iterative simulation approach requires the least effort among the three methodologies discussed in this section.

Chapter 8
State of the Practice and Future Needs for Production Planning and Control Systems

In this chapter, we describe information systems for production planning and control of wafer fabs. We start by discussing the current state-of-the-practice systems. Then we derive requirements for advanced production planning and control systems based on the results of the previous chapters. Next, we describe MES core functionality related to production control. Because the scheduling and dispatching functionality of many MESs is not adequate, we continue by describing specific dispatching systems. We also provide basic principles of a coupling architecture that allows for functionality extensions of MES in a plug-and-play manner. Moreover, we provide details of an agent-based architecture for modern production control systems. Software agents turn out to be an appropriate concept to implement enterprise-wide distributed production planning and control systems. We briefly discuss MCS functionality. Because of the increasing importance of production planning functionality, we finally describe requirements for ERP systems and APSs. These systems are usually provided by commercial software packages. Therefore, we also describe how these systems can interact with internally developed software components.

8.1 Motivation and State of the Art

In the previous chapters, different production planning and control methodologies were introduced. These algorithms have to be embedded into appropriate information systems because human decision makers are often not able to solve the corresponding decision problems since most of the algorithms proposed in the previous chapters require intensive computational effort and power.

Therefore, in this chapter, we mainly discuss the automated parts of the enterprise-wide information system, namely different application systems for production planning and control in wafer fabs or more generally in supply networks for semiconductor manufacturing. Enterprise-wide information systems

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5_8,

for manufacturing contain so-called front-end and back-end application systems. Front-end systems are business-related systems, whereas back-end application systems are formed by shop-floor systems (see Qiu and Zhou [255]). Typical front-end application systems are used for sales management, customer services, and planning, while back-end application systems are used for supportive logistics, machine controls, and MES. It is essential for semiconductor manufacturers to invest in modern APSs and advanced MESs [158, 255] to be competitive. Semiconductor manufacturers can increase their revenues by providing services and sales through efficient channels supported by front-end application systems. At the same time, the manufacturing-related costs can be reduced by efficient back-end application systems.

We start by reconsidering the PPC hierarchy shown in Fig. 2.5 in Chap. 2. In Table 8.1, we assign corresponding front-end and back-end application systems to the different levels of this hierarchy.

Table 8.1: Assignment of application systems to the levels of the PPC hierarchy

Level	Application system
Planning	APS ERP Custom planning system
Order release	ERP/APS/MES
Scheduling	MES Custom scheduling system
Dispatching	MES MCS Custom dispatching system

Note that the individual software systems for production planning and control are typically extensions of commercial software packages that offer a more specialized functionality. In many companies, the packaged software systems provide the necessary data for the tailored software systems.

We start by developing a high-level picture of the architecture of an enterprise-wide information system and will refine the overall picture in subsequent sections. We show the overall architecture for an enterprise-wide information system in a wafer fab in Fig. 8.1.

Application systems of the PS other than the ERP system are often components of APSs. The MES communicates with the immediate control systems for the machines and the AMHS via middleware. Sometimes the notion of an enterprise service bus is used for this kind of middleware. The middleware is responsible for transaction processing and event handling. It also provides recovery functionality. Note that we avoid discussing manufacturing automation details in this monograph. We refer to Lee [158] for background material on this topic. Because the data flow between the planning systems

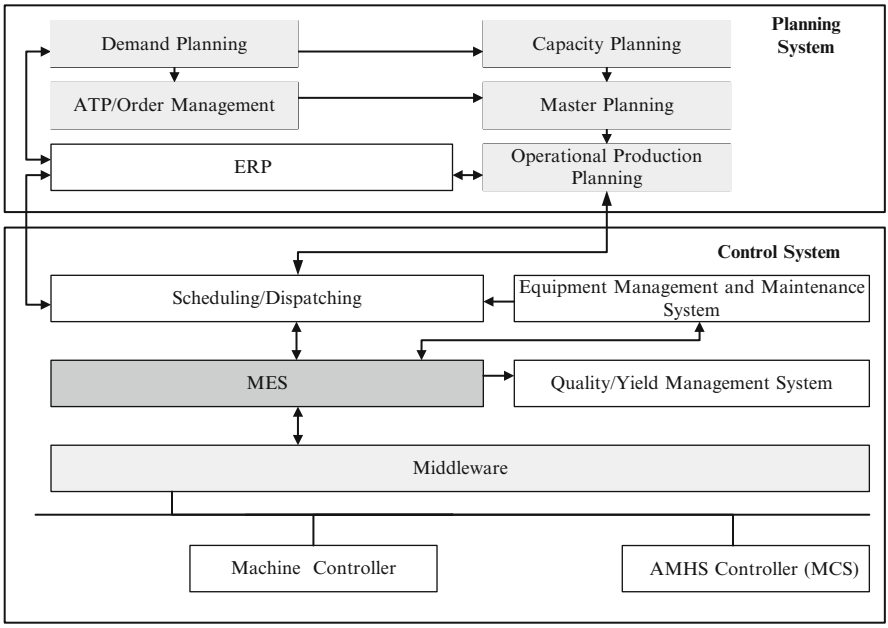

Figure 8.1: Enterprise-wide information system architecture

and the control systems is not extensive, proprietary application programming interfaces or web services are typically used to allow a message transfer between the planning systems and the MES.

When new business requirements appear, the supporting business processes have to change accordingly. Changing the business processes leads to new requirements for the enterprise-wide information system. In the next section, we will discuss some requirements for information systems that allow for these frequent changes.

8.2 Requirements of Production Planning and Control Systems

We derive several requirements for the different components of an enterprise-wide information system in wafer fabs or more generally in semiconductor supply chains. We identify the following requirements according to Qiu and Zhou [255]:

- The application systems should be able to deal with the data volume and the number of transactions that are typical for semiconductor manufacturing. This requirement is important because of the amount of data collected from the machine controllers and AMHS controllers.

- There are requirements related to the responsiveness of the application systems. A close to real-time response is critical for the MES, because the MES has to communicate with the machine controllers and AMHS controllers.
- The application systems should offer standardized connectivity with other application systems inside and outside the wafer fab. A plug-and-play connectivity that is suitable for different computing environments is highly desirable.
- Scalability and reconfigurability are key features of the application systems that are necessary to support business dynamics and growth.
- There is a trend towards multi-product facilities. The application systems have to be able to deal with many products. Furthermore, in some cases a single job is produced in several wafer fabs, i.e., borderless wafer fab scenarios (cf. Qiu and Zhou [255] and Sect. 2.2.2). The different planning systems and the MES have to support these requirements.
- The MES should allow an even tighter integration with automation solutions on the shop floor (see Lee [158]). Furthermore, it is anticipated that the advanced process control (APC) and AMHS operations will in the future be more tightly integrated with scheduling functionality. It is also expected that the importance of scheduling systems will be increased with a higher level of automation in future-generation wafer fabs. The MES architecture has to take these requirements into account.
- Appropriate production planning and control functionality should be offered by the application systems with respect to solution quality and the time needed to compute the solutions. It should be possible to modify and change the algorithms used quite easily to quickly react to changing business needs.
- The application systems should have an open and modular system architecture that is web-based to support the required interoperability for heterogeneous application systems. Furthermore, the target architecture should ensure a certain degree of adaptability for more advanced new technologies. We see also a trend towards collaborative manufacturing [47, 255]. A future enterprise-wide architecture has to take this trend into account.

It seems that service-oriented architectures (SOA) support many of the desired requirements. SOA proposes to define the required business functionality by a set of composable services. When needed because of changes in the business processes, new services can be obtained by service composition to support these business processes. However, it is difficult to assess the benefit of this type of architecture in semiconductor manufacturing.

It seems that SOA principles are used so far only on a preliminary basis in semiconductor manufacturing. An application on the factory automation level is described by Qiu [253]. A service-based application system for virtual manufacturing systems that involves several vendors in the semiconductor industry is discussed by Cherbakov et al. [47].

An interesting discussion of future architectures for enterprise-wide information systems in the pre-SOA era can be also found in Qiu [252]. In this context, E-manufacturing is defined as advanced manufacturing that takes advantage of the Internet and advanced information technologies to integrate the different manufacturing-related applications within a semiconductor supply network.

8.3 Production Control Systems

In this section, we start by discussing MES functionality. Then, we describe dispatching systems and explain their interaction with the MES. We present design and implementation details of a hierarchically organized multi-agent-system for production scheduling. Finally, we briefly discuss MCS functionality.

8.3.1 MES Core Functionality

An MES is an application system that consists of a set of integrated hardware and software components that are used to manage the production from job release until job completion [178, 182]. An MES is between the production PS and the execution of the production; it bridges the BS with the PS. The embedding of an MES into the overall system landscape of a wafer fab according to Chung and Jeng [50] is shown in Fig. 8.2.

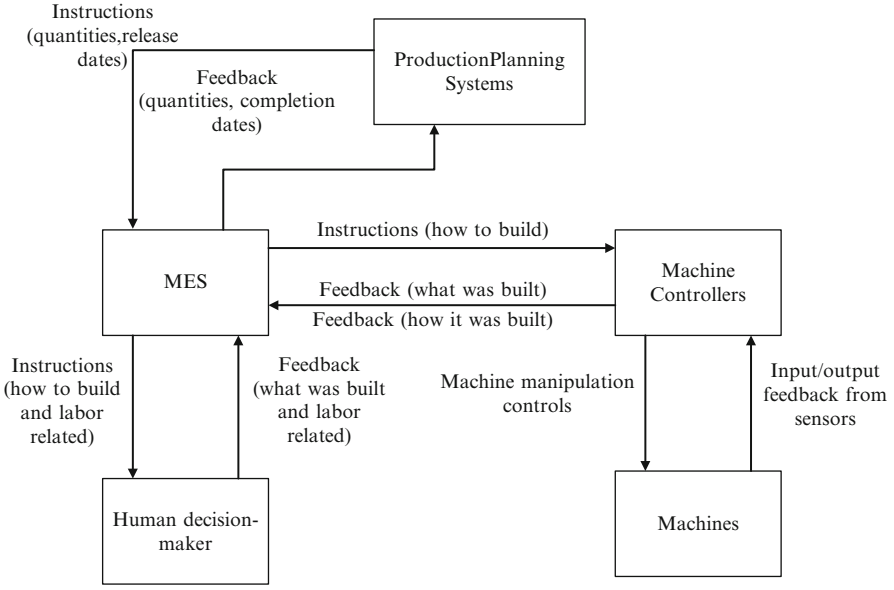

Figure 8.2: Interaction of an MES with other application systems

The main functionality of an MES in a wafer fab includes the following

- Equipment definition
- Product process definition
- Resource status tracking
- WIP status tracking
- Information management, i.e., data collection and acquisition
- Dispatching and scheduling
- Performance analysis

The equipment definition is used to define the processing capabilities of each machine. Usually, the machines are organized in types that are based on the associated processing capabilities.

The product process definition refers to the definition of the process flows. Each product type can potentially be manufactured using different versions. Each process version consists of several subroutes. Each subroute contains several consecutive process steps. A single process step refers to a recipe that is executed on the machine that is associated with the recipe. Each process step can have several versions. They depend on the machine and parameter calibrations. Current and planned production activities are used to define product processes.

Resource status tracking determines what task each single machine is currently performing. Furthermore, the current status of secondary resources is also tracked. Note that the status of the different resources involved in the AMHS is usually tracked in the MCS and not in the MES. The WIP status tracking refers to the situation where the progress of jobs is monitored. The aim is to create a full history for each working object of the BP. The information management functionality deals with monitoring and gathering of data about the different objects within the BS and the BP. Information management is an important prerequisite for implementing the resource and WIP tracking capabilities.

Dispatching and scheduling are among the key functionalities offered by an MES. They are based on the data that is determined by the MES. Some packaged MES products for wafer fabs offer dispatching functionality, but often additional off-the-shelf or homegrown solutions are used (cf. Pfund et al. [234] for the results of a corresponding survey). Therefore, we discuss dispatching and scheduling systems in more detail in subsequent sections.

The performance analysis component of an MES is responsible for comparing performance measure values with the corresponding objectives that are set by the management of the wafer fab or by customers. Graphical and numerical reports are also provided by this component.

Note that quality management and maintenance management are also key capabilities of MES [178, 182]. However, due to their high importance in semiconductor manufacturing, the corresponding functionality is often offered by separate application systems (cf. also Fig. 8.1). Therefore, we will discuss some of these systems briefly in subsequent sections.

Important packaged MES products in semiconductor manufacturing are WorkStream, WorkStream DFS, i.e., FAB300, from Consilium; PROMIS from Brooks Automation, Inc.; and SiView from IBM (see Qiu and Zhou [255]). The development of a homegrown MES is generally supported by the SEMATECH CIM-Framework [279]. However, it seems that packaged MES solutions have become the dominant implementations in wafer fabs (see Pfund et al. [234]).

8.3.2 Dispatching Systems

Dispatching systems are in place in most wafer fabs [234]. In an ideal scenario, a proposed dispatching method's superiority is established through experimental testing using actual semiconductor manufacturing data. In this scenario, the actual BS and BP data obtained in several relational databases including the MES database are extracted from the MES for use in developing and testing dispatching approaches. Therefore, a central repository is used to store this data. The application layer of the dispatching system interacts with the repository and also directly with the MES. This layer contains tools to construct blended, multilevel, conditioning, and truncated dispatching rules (cf. Sect. 4.1) and can be considered as a rule-based system (cf. Sect. 4.7.1). A simulation model of the current BS and BP is built using the data in the repository, and simulation analysis is used to assess the performance of the proposed dispatching rules that are developed using the functionality of the application layer. The dispatching system is completed by a graphical user interface (GUI) and a reporting interface.

Some manufacturers use dispatching systems that communicate directly with their MES in near real time. The dispatching list shown in the GUI of the dispatching system is implemented using functions of the MES.

Applied Materials' real time dispatcher (RTD) product [10] and the FabTime product [77] are commonly used [9, 238]. Once the efficacy of the dispatching approach is confirmed, the final step in this process is the deployment of the dispatching approach in an actual wafer fab.

RTD is a real-time, high-performance dispatching solution that helps manufacturers develop customized rules and improve dispatching analyses and decisions. As a member of Applied Materials' productivity family, RTD directs pre-staging, releases jobs, and adjusts production equipment loads through "what next, where next, and when next" rules. The goal of these rules is to improve the utilization of wafer fab equipment, carriers, and wafer fab personnel. RTD allows planners to define job selection logic in various rules and make production dispatching decisions in real time. This is facilitated by collecting and storing data from multiple sources in a high-speed, temporal repository. In this way, RTD is able to rapidly access both current shop-floor status and historical information over a desired definable time period [10].

FabTime is a web-based digital dashboard system that was created for the original purpose of helping wafer fabs to measure and improve their CT performance. In real time, FabTime provides a comprehensive view of everything a wafer fab manager would need for proactive CT management. The system includes over 120 standard wafer fab management charts focused on metrics such as WIP, moves, job age, WIP turns, per-visit CT, factory CT, machine states, machine overall equipment effectiveness (OEE), scrap, and holds. The FabTime system works by taking a continuous feed of operational transactions from the wafer fab's MES and processing these transactions in real time for storage in a database. Both chart and tabular output can be viewed using a standard web browser across a corporate intranet. In addition to providing real-time alerting so that users can specify conditions under which to be notified (such as a machine down event or the release of a new product), FabTime also includes optional dispatching and static capacity planning modules. As the FabTime system maintains real-time knowledge of current wafer fab status through its database information (which is constantly updated by the MES), FabTime's dispatching module [77] can effectively promote in-fab deployment of the system's recommended job dispatch list for any machine in the wafer fab.

The typical architecture of a dispatching system is shown in Fig. 8.3. It is similar to those described in Sect. 3.3.2 because the simulation model represents the BS and the BP in both situations.

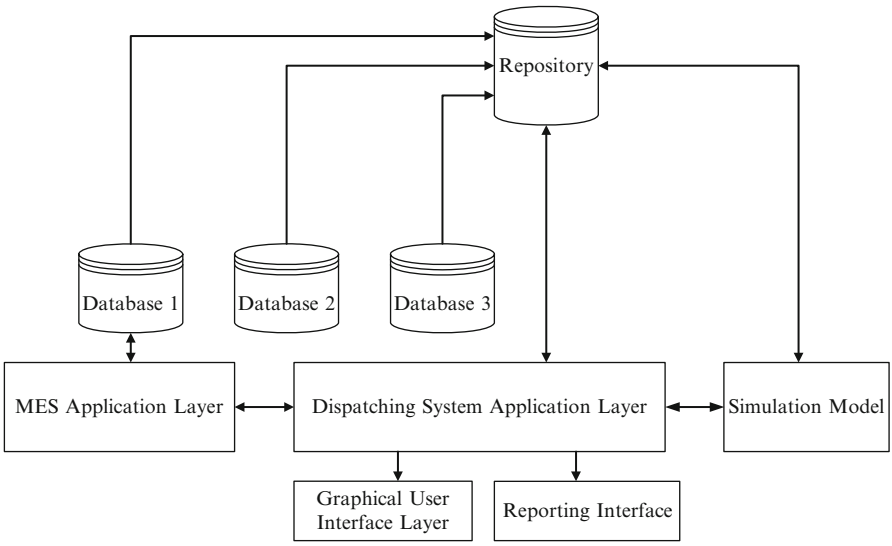

Figure 8.3: Architecture of dispatching systems

8.3.3 Scheduling Systems

Similar to dispatching systems, scheduling systems often are not part of the MES because it is hard to adapt generic scheduling functionality to the situation in a specific wafer fab. Currently, wafer fab-wide scheduling systems are generally not in use (see Mönch et al. [207]). However, scheduling systems for work centers or even work areas are in use in many wafer fabs. Some example systems are described in [28, 147, 329, 330].

We continue by describing the architecture of scheduling systems. A scheduling system typically consists of the following components (see Framinan and Ruiz [90]):

- User interface component
- Schedule generator component
- Business logic component
- Database management component

The main functionality of the different components can be characterized according to Framinan and Ruiz [90] as follows. The user interface component offers the required interfaces between the user of the scheduling system and the system itself. The component allows for the representation of scheduling output. Finally, the user can initiate the scheduling process, including choosing appropriate parameter settings, using the user interface.

The schedule generator component contains all the functionality that is required to determine schedules for the users. It consists of an algorithm library that contains scheduling and rescheduling algorithms. An algorithm generator is responsible for adding new algorithms or obtaining new ones by combining existing ones from the algorithm library. The input data is transformed by a preprocessor into a format that is appropriate for the scheduling algorithm. Finally, a schedule is calculated.

The business logic component is between the user interface component, the schedule generator component, and the database management component. It ensures the required abstraction when data is accessed from the database by the user interface or by the schedule generator. The business logic component is a transformation component. Finally, the database management component is responsible for storing all the data that is obtained from the MES and from other databases similar to the repository in the case of dispatching systems. It provides interfaces to the application systems that contain the data that are necessary for scheduling.

In Sect. 8.3.4, we present some details of an agent-based scheduling system prototype that can be used in wafer fabs. We show how the NDSBH and the IDSBH algorithms from Sect. 5.4.6 can be implemented in a distributed manner.

8.3.4 FABMAS: An Agent-Based Scheduling System

Software agents allow for the implementation of distributed planning and control algorithms. The agents are able to act autonomously; on the other hand, their communication abilities ensure a cooperative behavior and the fulfillment of global system goals. Furthermore, agent-based systems facilitate maintenance and further development tasks of the software (see Weiss [319] and Wooldridge [324]).

We start by the agentification of the scheduling problem for a single wafer fab. Many approaches address the problem of identifying proper agents for a given application domain. We refer, for example, to the Gaia approach described by Zambonelli et al. [331]. The Gaia approach is a generic approach that assigns a set of roles to a given domain. We define a role as a class that determines the normative behavior repertoire of an agent (see Odell et al. [217]). Interactions are identified that take place between the different roles. However, as pointed out by Bussmann [38] and by Bussmann et al. [39, 40], it is necessary to analyze and understand the decisions in the course of the production control process.

We combine the approach of Bussmann [38] with the PROSA reference architecture for holonic manufacturing systems (cf. Van Brussel et al. [310]). PROSA is an abbreviation for *P*roduct, *R*esource, *O*rder, and *S*taff *A*rchitecture. The reference architecture suggests building agent-based production control systems by using these agent (holon) categories. Furthermore, PROSA provides a high-level description of the interaction of instances of these agent categories and a set of examples for using the architecture. A holonic manufacturing system is a system of holons that are able to cooperate to achieve a common objective (see McFarlane and Bussmann [179]). An autonomous and cooperative building block of a manufacturing system for transforming, storing, transporting, and validating information and physical objects is called a holon. Holons can be part of other holons, i.e., there is a recursive structure. Note that for our purposes, the difference between holons and agents is not important (cf. McFarlane and Bussmann [179] for a discussion of related issues). We consider three steps:

1. Analyze the decision-making process.
2. Identify the necessary agents.
3. Choose appropriate interaction protocols.

Based on the proposed hierarchical approach described in Sect. 5.4.6, we distinguish three types of decisions. These decision types are presented in Table 8.2.

The decision-making on the top and middle layer will be performed in a rolling horizon or event-driven manner. The ICA scheme (cf. Sect. 5.4.6) is used on the top layer, while NDSBH or IDSBH (cf. Sect. 5.4.6) makes decisions on the middle layer. The decisions of the base layer are made dependent on the situation of the machine group. If the schedule is infeasible, then

Table 8.2: Decision types in FABMAS

Layer of the hierarchy	Decision	Decision space
Top layer	Start and end dates for each macro operation of a job	Time slots for the start and end dates
Middle layer	Assignment and sequencing decisions for operations of a job	A specific machine among parallel machines, a concrete time slot on that particular machine
Base layer	Whether to follow the schedule or not, assignment and sequencing decisions for operations in the latter case	A specific machine among parallel machines, a concrete time slot on that particular machine

decision-making entities of the jobs make dispatching decisions together with the decision-making entities of the machine groups. A contract net-type allocation algorithm (cf. Weiss [319] for contract nets) is used.

Starting with PROSA, we distinguish between decision-making agents and staff agents. Decision-making agents solve decision problems while the staff agents try to support them in the course of the decision-making process. In PROSA, we find order, product, and resource agents as abstract classes. We identify eight decision-making agent types, i.e., roles, in our application scenario.

1. Each job agent represents a single job.
2. Batch agents are used to control a certain batch, i.e., a collection of jobs that are intended to be processed at the same time on the same machine.
3. A PM agent represents a preventive maintenance order.
4. Work center agents represent machine groups on the shop floor.
5. We aggregate several work center agents into one work area agent.
6. The fab agent consists of all work area agents.
7. Tool agents are used for the representation of auxiliary resources.
8. We consider technology agents that encapsulate the product knowledge, i.e., the routes, according to the product holons of PROSA.

We identify two additional staff agents that encapsulate the scheduling and monitoring functionality for the hierarchical production control scheme.

We summarize the basic functionality of the members of the decision-making and staff agency in Table 8.3. The different agent roles are shown in Fig. 8.4. Each role is described by a set of possible behaviors. A single behavior is given by a set of states and by transition paths from one state to another state. PROSA describes the basic interaction between product, resource, and order agent roles (see Van Brussel et al. [310]). For the purpose of FABMAS, a modeling of the interactions between decision-making and

Table 8.3: Functionality of decision-making and staff agents

Member of the agency	Task description
Fab agent	– Coordinating the work of the fab scheduling agent, the monitoring agent, and the work area agents
Fab scheduling agent	– Preparing to run ICA – Running ICA and providing scheduling information
Work area agent	– Coordinating the work of the corresponding work area scheduling and monitoring agent – Decision-making for choosing the proper machine criticality measure for NDSBH or IDSBH – Providing information services
Work area scheduling agent	– Preparing to run NDSBH or IDSBH – Running the heuristic – Providing scheduling information
Work center agent	– Implementing the work area schedules – Mediating in the case of a contract net-type allocation algorithm
Tool agent	– Implementing the work area schedules
Job agent	– Coordinating the processing of the job that is represented by the job agent – Negotiating with work center agents
Batch agent	– Coordinating the processing of the batch that is represented by the batch agent – Negotiation with work center agents
PM agent	– Coordinating the preventive maintenance step that is associated with the agent
Technology agent	– Providing routing information to job agents, work center agents, and various staff agents – Dynamically changing the assignment of work centers to process steps, e.g., caused by machine breakdowns

staff agents is more important. We identify four different behaviors for staff agents:

- PrepareSolution behavior
- ParameterizeAlgorithm behavior
- SolveOrInterrupt behavior
- CommunicateSolution behavior

A brief description of the different behaviors of staff agents is given in Table 8.4.

For decision-making agents, we identify the behaviors:

- PrepareDecisionMaking
- MakeDecision
- InformDecisionMakingAgents
- StartStaffAgent
- RequestDecisionMakingAgentResults

The different behaviors of decision-making agents are briefly described in Table 8.5.

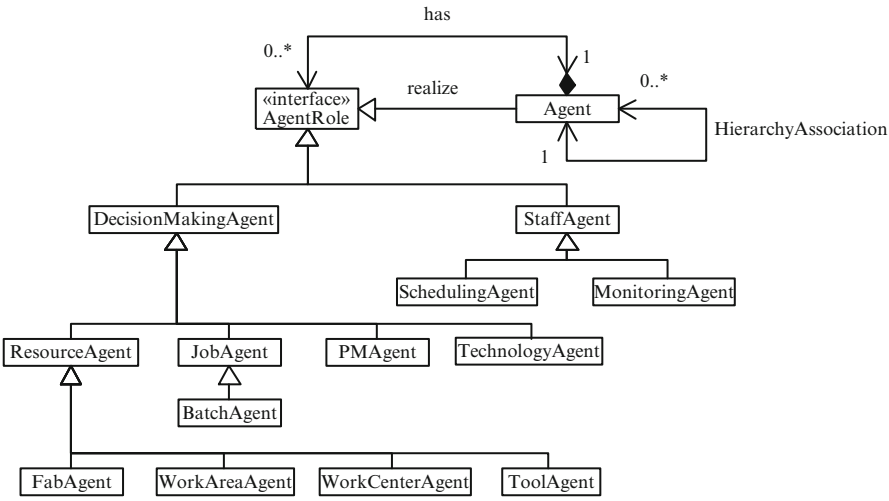

Figure 8.4: Agent roles in FABMAS

Table 8.4: Behaviors of staff agents

Behavior	Description
PrepareSolution	This behavior models the data collection and parameter determination phase of the problem solution process
ParameterizeAlgorithm	Parameters are assigned to the solution algorithm
SolveOrInterrupt	Feasible solutions are determined for the problem under consideration
CommunicateSolution	The behavior informs the decision-making agent associated with the staff agent

Next, we briefly discuss the implementation of the FABMAS prototype [204]. The use of an agent toolkit or framework was avoided because none of the existing tools allows for the implementation of hierarchies and the usage of discrete-event simulation for performance assessment. Furthermore,

problems with respect to computational performance were expected because the majority of toolkits are Java-based. The prototype is implemented using the Microsoft .NET framework for inter computer communication. Within .NET, we implement the prototype in the C# programming language, while some parts of source code are developed in the C++ programming language. According to the foundation for intelligent physical agents (FIPA) [79] Abstract Architecture for agent systems, we develop an agent runtime environment. The runtime environment consists of an agent directory service, an agent management system, an agent container, and an agent communicator. These parts of the system provide services that are used by the agents to get information about other agents and to communicate and interact with them.

Table 8.5: Behaviors of decision-making agents

Behavior	Description
PrepareDecisionMaking	The agent is prepared for a decision-making process, for example, by data collection
MakeDecision	A decision is made by the agent
InformDecisionMakingAgents	Another decision-making agent is informed of a decision of the agent
StartStaffAgent	A service of a staff agent is requested by the decision-making agent
RequestDecisionMakingAgentResults	A certain result obtained by another decision-making agent is requested

For communication purposes, the agent communicator encapsulates communication capabilities. Each communication act between agents is handled by the agent communicator. The agent directory service is the location where agents register their specification as a service entry. Agents are able to ask the local directory service to find information about other agents they want to interact with. If the information is not available, the directory service tries to find the information by contacting other directory services within the whole multi-agent-system. Hence, it is not necessary to establish a global directory service as a centralized information point in a distributed system. Each agent runtime environment requires an agent management system that administers the life cycle of each agent. As a result, the management system is responsible for creating the agents, provides potential mobility services, and removes an agent if it is no longer needed. The last component of the agent runtime environment is the agent container as a collection of all active agents within the environment.

Next, we discuss the communication in FABMAS. Various opportunities for implementing communication abilities are given by choosing the .NET Platform for the development of the agent system. An agent communicator is part of every runtime environment and provides two capabilities for communication. The multicast communication is used for announcement of

new active runtimes and by the agent directory service to keep their agent list up-to-date. The direct communication is used for communication purposes between single agents. Both types of communication are implemented by using the .NET Remoting framework for distributed computing.

The agent communicator hides all the communication capabilities from the agents and can be used as an interface. An agent that is intended to send a message to another agent transfers the message to the agent communicator. The communicator determines the location of the agent, and, if the other agent is on a remote host, .NET Remoting is used for sending the message. If the receiver agent is in the same runtime, the message is directly put into the mailbox of the receiver agent.

The agent communicator is implemented as client and server, simultaneously. The usage of threads makes the agent communicator concurrently executable. According to the FIPA proposal for an agent communication language, we use the content format described by Mönch and Stehli [195]. The format is basically given by a context-free grammar, and it relies heavily on the ontology described by Mönch and Stehli [196].

We continue by discussing the representation of hierarchies within FABMAS. The hierarchy according to the suggested hierarchical approach is modeled by using an agent identifier. The identifier is a pointer to agents. An agent identifier encapsulates the agent name, the address where the agent is located, and the services provided by the agent. Each agent on a higher level stores the agent identifier of its child agents on the next hierarchy layer. The fab agent contains all agent identifiers of the work area agents. Each work area agent knows the identifiers of the agents associated with work centers that are part of the work area. On the other hand, each work center agent knows its work area agent. Thus, a structure exists that allows for communication and cooperation among the decision-making entities at the same hierarchy layer and between adjacent layers.

We use an extension of the architecture described in Sect. 3.3.2 to carry out the performance assessment of FABMAS. The center point of this architecture is a blackboard-type data layer that contains all the information to execute the ICA heuristic (cf. Sect. 5.4.6), construct the disjunctive graphs for the NDSBH and IDSBH schemes (cf. Sect. 5.4.6), and make the scheduling decisions. The data layer is between a simulation model that emulates the manufacturing process of interest and the FABMAS system. The objects of the data layer are updated in an event-driven manner by appropriate simulation events. Calculated schedules are submitted to the simulation engine AutoSched AP in order to use the information of the schedules in a dispatching-based manner. The architecture allows for rolling horizon-type scheduling as well as for event-driven rescheduling activities. We implement an interfacing component in order to connect FABMAS with the blackboard. The .NET system supports the encapsulation process and also registers the component for the Windows Registry. The .NET framework creates a component object model (COM) callable wrapper during runtime. This wrapper offers user-

specific interfaces to the component and also the typical interfaces of COM. The blackboard can call methods of the interface and transfer data via parameters to the FABMAS multi-agent-system. This data is forwarded by the .NET Remoting component to the agent runtime environment. On the other hand, data is transported back from the multi-agent-system to the blackboard via method calls using reference parameters. The communication between the described components is shown in Fig. 8.5.

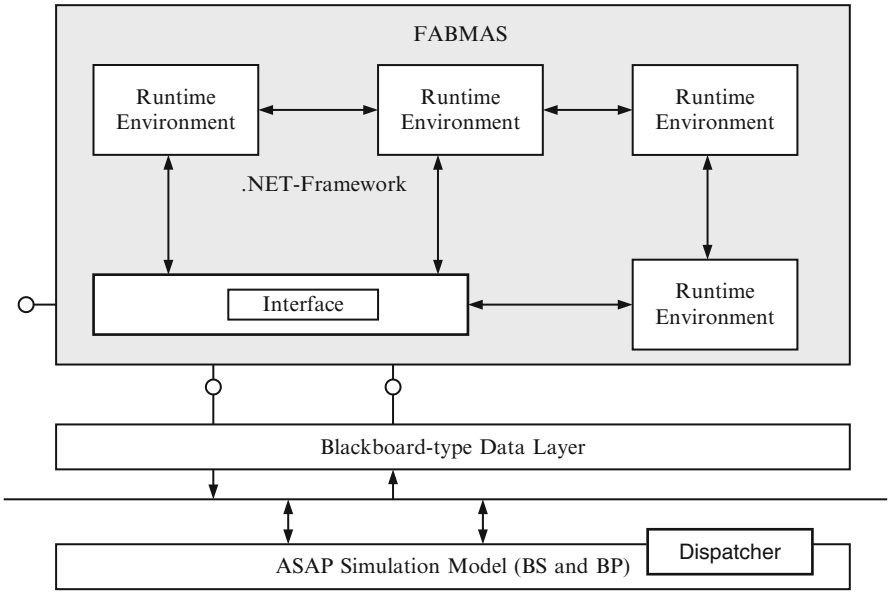

Figure 8.5: Architecture for performance assessment of the FABMAS system

We wish to point out that agent-based approaches have become popular in semiconductor manufacturing over the last decade. In addition to FABMAS, there is an agent-based scheduling system at AMD/GLOBALFOUNDRIES discussed by Pinedo [240]. In addition, Yoon and Shen [328] describe a multi-agent-system for wafer fabs that makes scheduling decisions based on a bidding scheme.

8.3.5 Additional Application Systems as Part of the Control System

Among the different controls, we discuss only the MCS because of its importance in modern wafer fabs. The MCS is responsible for initiating and coordinating movements of carriers for wafers and reticles (see Foster and Pillai [82]). Furthermore, it coordinates interbay and intrabay activities between stockers and machine load ports. Intrabay and intrabay AMHS com-

ponents are monitored by the MCS. Carrier locations within the AMHS are stored in the MCS.

The MCS interfaces with the MES and also with different components of the AMHS. It receives job destination information and also dispatching- and scheduling-related information from the MES. The MES is updated by the MCS when jobs are moved within the wafer fab. The MCS receives the identification numbers of jobs and carriers when they enter the AMHS. The movement of jobs is instructed and coordinated by the MCS. Finally, the MCS collects information with respect to job movements, job locations, and AMHS reliability.

Equipment engineering systems (EES) are used as tools for vendors to monitor process control of machines and tune process parameters remotely (see Lee [158]). It is a physical implementation of equipment engineering capabilities (EEC). The main goal of an EES is to increase the OEE. An EES contains fault detection and classification (FDC) and predictive maintenance. Sometimes, the functionality of a quality/yield management system is also part of an EES. FDC aims to detect anomalies in process control that can cause large quality problems, classify the problems, and report them. Statistical methods and methods from data mining and machine learning are applied to solve this task. Predictive maintenance starts by detecting a failure symptom. Then, the residual life of the equipment is predicted before the failure occurs. A failover process is finally initiated. An equipment management and maintenance system (EMMS) is a software tool that monitors and tracks equipment states and preventive maintenance schedules in real time. It is similar to an EES.

Quality requirements have become stricter over the years in semiconductor manufacturing (see Lee [158]). Therefore, separate quality/yield management systems are installed in most wafer fabs, and they are not generally part of the MES. APC functionality is an integrated part of a quality/yield management system. APC includes run-to-run control, FDC, statistical process control (SPC), and virtual metrology (VM). Run-to-run control is responsible for adapting process control parameters (see Moyne et al. [211]). Therefore, measurements of process sensors are used that are taken on a wafer-to-wafer, job-to-job, or batch-to-batch basis to adjust recipes. Typically, statistical methods and methods from machine learning are used. FDC is typically part of the EES. SPC is used to monitor the physical measurements of the wafers. This measurement is carried out using metrology equipment. VM is based on the insight that metrology is time- and cost-consuming. Therefore, the number of wafers that can be measured is limited. VM uses mathematical modeling to estimate the values of metrology measures on the wafers depending on FDC indicators without physical metrology operations.

While we have described only production control-related application systems so far, in the following section, we discuss in more detail planning-related application systems. Such systems provide instructions that are important for production control-related application systems.

8.4 Production Planning Systems

In this section, we describe the main functionality of ERP systems and APSs. We then briefly discuss the interaction with other systems.

8.4.1 ERP and APS Core Functionality

We start by describing the core functionality of an ERP system. ERP systems are typically transaction-oriented software packages. They offer functionality related to finance, human resources, manufacturing and logistics, and finally sales and distribution (see Hopp and Spearman [119]). ERP systems are operational application systems. They store important BS- and BP-related data, called master data. This includes product data, bills of materials, routes, and resource- and job-related data. ERP systems are often organized in different software modules. Each of these modules supports one of the main ERP functionalities. ERP systems follow an integrated approach that is characterized by the following features [119]:

- Integrated functionality
- Integrated databases
- Consistent user interfaces
- Unified architecture and tools to maintain, improve, and extend the system
- Single vendor/contract and a unified product support

The manufacturing- and logistics-related module typically offers MRP and manufacturing resources planning (MRP II) functionality. However, it is well known that several assumptions of these approaches, like infinite capacity and fixed CT, might lead to fundamental problems and low performance of manufacturing systems.

Because of the simple or even missing bill of materials in many wafer fabs, the production planning functionality of ERP systems is used only to a small extent in wafer fabs. The order management functionality offered by ERP systems is often the most important functionality in this context. The second important functionality is demand planning, i.e., forecasting. However, on the entire enterprise level, ERP systems are often complemented by APSs.

An APS supports decisions within the supply chain management context on a long-term, mid-term, and finally short-term planning level. They can be considered as extensions of ERP systems because these systems are typically not able to solve the entire set of decision-making problems associated with a supply chain. An APS is an application system that supports production planning tasks using OR and AI methods taking the finite capacity of the BS into account. The most important features of an APS are according to Fleischmann et al. [81] as follows:

- Integrated planning along the entire supply chain
- Optimization-based approaches based on mathematical models and algorithms that either provide optimal or heuristic solutions

- Application of a hierarchical planning approach that decomposes the entire planning problem in a series of smaller, less complex subproblems that are assigned to the different layers of the hierarchy

Similar to ERP systems, APSs are usually offered as packaged software systems. They provide functionality related to the strategic design of the manufacturing network, demand planning, supply network planning, external procurement, production planning and scheduling, transportation planning and vehicle scheduling, order fulfillment, and available-to-promise (ATP). In contrast to ERP systems, APSs often use algorithms that are based on data in the memory of the computers. This approach, sometimes called Live-Cache technology, allows for very fast accessing of the data, because accessing relational databases is avoided.

Next, we describe how the different subsystems of the overall PS in Fig. 8.1 are implemented in semiconductor manufacturing. We start with demand planning. Demand planning is related to the task of forecasting the future market demand for the semiconductor products of the companies. Demand planning is more important when the company follows a make-to-stock rather than a make-to-order strategy. Long-term demand forecasts are important for the design of the supply network and for capacity expansion decisions. Mid-term forecasts are essential for the coordination of procurement, manufacturing, and distribution. They form an important input for supply network planning, i.e., master planning. Finally, short-term demand forecasts are essential to ensure high service levels (see Günther [109]).

Based on a survey by Roundy [271], we conclude that the demand planning functionality of ERP systems and APSs is often used in semiconductor manufacturing. However, homegrown solutions are also widespread in this industry. A prototype that offers optimization-based short-term forecast functionality in semiconductor manufacturing is described by Mönch and Zimmermann [199]. It uses web services to interact with different application systems to gather the required data.

The main purpose of an order management component consists in matching customer orders against quantities available in stock or from already planned production orders. The investigation of whether a delivery can be performed or not is called ATP. Usually, one looks for available stock that can be promised for delivery. Many APSs are able to check the available capacity that can be used to place new orders. Furthermore, it is also checked whether or not the size of already planned orders can be increased. This feature is called capable-to-promise (CTP). An order management component is also responsible for order entry and customer service, reporting, order processing, and financial processing.

Capacity planning decisions are described in Sect. 7.3. It seems that often homegrown decision-support systems, including a commercial MIP solver, are used to make these kinds of decisions. Simple spreadsheet-based applications that use the included LP or MIP solvers are also popular. One example for

the first class discussed in the literature is CAPS (see Bermon and Hood [24]) that was run at several IBM wafer fabs to support capacity planning decisions (cf. Sect. 7.3).

Master planning in semiconductor manufacturing is discussed in Sect. 7.2. Master planning algorithms are typically provided by APSs; however, because packaged APSs are often not able to deal appropriately with process restrictions of semiconductor manufacturing, homegrown solutions are also in use. In Kallrath and Maindl [135], it is indicated that heuristics from SAP APO are used for master planning in semiconductor manufacturing.

Operational planning is short-term planning that is discussed in Sect. 7.1. It is often supported by homegrown, spreadsheet-like solutions. In a certain sense, the ICA algorithm, presented in Sect. 5.4.6, is somewhere between operative planning and production control. Usually, this kind of approach is provided by an MES.

A discussion of some trends in planning systems for supply chains in the semiconductor manufacturing industry can be found in Banerjee [20]. Some empirical evidence of APS failures in semiconductor manufacturing can be found in Lin et al. [164]. It is shown that APSs in semiconductor manufacturing often do not perform better than humans that make planning decisions with computer support.

8.4.2 Interaction with Other Systems

We can see from Fig. 8.1 that there is a link between operational planning systems and dispatching and scheduling systems. Furthermore, there can be a link between the ERP system and the MES because the MES needs order information. In addition, when a job is completed, this information has to be sent to the ERP system.

Within the PS, there are several possibilities for the interactions of APS and ERP systems. In the simplest case, there is exactly one APS that is on top of the ERP system. But it is also possible that several APSs are in use together with one centralized ERP system. Furthermore, we might have one centralized APS and several ERP systems. Typically, global supply chains in the semiconductor industry contain multiple APSs and several ERP systems. Little is known about how SOA-type approaches will impact the architecture of the next-generation PS in semiconductor manufacturing.

References

1. Adams, J., Balas, E., Zawack, D.: The shifting bottleneck procedure for job-shop scheduling. Manag. Sci. **34**(3), 391–401 (1988)
2. Adelsberger, H.H., Kanet, J.J.: The Leitstand—a new tool in computer-integrated manufacturing. Prod. Inventory Manag. J. **32**(1), 43–48 (1991)
3. Agrawal, G.K., Heragu, S.S.: A survey of automated material handling systems in 300-mm semiconductor fabs. IEEE Trans. Semicond. Manuf. **19**(1), 112–120 (2006)
4. Akçali, E., Uzsoy, R.: A sequential solution methodology for capacity allocation and lot scheduling problems for photolithography. In: Proceedings of the Twenty-Sixth IEEE/CPMT International Electronics Manufacturing Technology Symposium, pp. 374–381 (2000)
5. Akçali, E., Nemoto, K., Uzsoy, R.: Cycle-time improvements for photolithography process in semiconductor manufacturing. IEEE Trans. Semicond. Manuf. **14**(1), 48–56 (2001)
6. Allen, C.: The impact of network topology on rational-function models of the cycle time-throughput curve. Master's thesis, Northwestern University (2004)
7. Almeder, C., Preusser, M., Hartl, R.F.: Simulation and optimization of supply chains: alternative or complementary approaches? OR Spectrum **31**(1), 95–119 (2009)
8. Ankenman, B.E., Bekki, J.M., Fowler, J.W., Mackulak, G.T., Nelson, B.L.: Simulation in production planning—an overview with emphasis on recent developments in cycle time estimation. In: Kempf, K.G., Keskinozak, P., Uzsoy, R. (eds.) Planning Production and Inventories in the Extended Enterprise, pp. 565–591. Springer, Berlin (2011)
9. Appleton-Day, K., Shao, L.: Real-time dispatch gets real-time results in AMD's Fab 25. In: Proceedings of the 1997 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, pp. 444–447 (1997)
10. Applied-Materials. Applied Real-Time Dispatcher (2011). http://www.appliedmaterials.com/services-software/library/apf-rtd-apf-reporter
11. Ashby, J.R., Uzsoy, R.: Scheduling and order release in a single stage production system. J. Manuf. Syst. **14**(4), 290–305 (1995)
12. Asmundsson, J., Rardin, R.L., Uzsoy, R.: Tractable nonlinear production planning models for semiconductor wafer fabrication facilities. IEEE Trans. Semicond. Manuf. **19**(1), 95–111 (2006)
13. Asmundsson, J., Rardin, R.L., Turkseven, C.H., Uzsoy, R.: Production planning with resources subject to congestion. Nav. Res. Logist. **56**(2), 142–157 (2009)

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5,

14. Atherton, L.F., Atherton, R.W.: Wafer Fabrication: Factory Performance and Analysis. Kluwer Academic Publishers, Norwell (1995)
15. Aytug, H., Kempf, K., Uzsoy, R.: Measures of subproblem criticality in decomposition algorithms for shop scheduling. Int. J. Prod. Res. **41**(5), 865–882 (2003)
16. Bahaji, N., Kuhl, M.E.: A simulation study of new multi-objective composite dispatching rules, CONWIP, and push lot release in semiconductor fabrication. Int. J. Prod. Res. **46**(14), 3801–3824 (2008)
17. Bai, X., Srivatsan, N., Gershwin, S.B.: Hierarchical real-time scheduling of a semiconductor fabrication facility. In: Proceedings of the Ninth IEEE International Electronics Manufacturing Technology Symposium, pp. 312–317 (1990)
18. Balasubramanian, H., Mönch, L., Fowler, J.W., Pfund, M.E.: Genetic algorithm based scheduling of jobs with incompatible families on parallel batch machines. Int. J. Prod. Res. **42**(8), 1621–1638 (2004)
19. Balasubramanian, H., Fowler, J.W., Pfund, M.E.: Single machine bicriteria scheduling using the apparent tardiness cost heuristic. In: Proceedings of the 2006 Industrial Engineering Research Conference (2006)
20. Banerjee, A.: Global trends in supply chain planning in semiconductor industry. SETLabs Briefings **5**(3), 1–11 (2007)
21. Banks, J., Carson, J.S., Nelson, B.L.: Discrete-Event System Simulation, 2nd edn. Prentice Hall, Upper Saddle River (1999)
22. Barahona, F., Bermon, S., Günlük, O., Hood, S.J.: Robust capacity planning in semiconductor manufacturing. Nav. Res. Logist. **52**(5), 459–468 (2005)
23. Barua, A., Raghavan, N., Upasani, A.A., Uzsoy, R.: Implementing global factory schedules in the face of stochastic disruptions. Int. J. Prod. Res. **43**(4), 793–818 (2005)
24. Bermon, S., Hood, S.J.: Capacity optimization planning system (CAPS). Interfaces **29**(5), 31–50 (1999)
25. Bertsimas, D., Tsitsiklis, J.N.: Introduction to Linear Optimization. Athena Scientific, Dynamic Ideas, Belmont (1997)
26. Bianco, L., Ricciardelli, S., Rinaldi, G., Sassano, A.: Scheduling tasks with sequence-dependent processing times. Nav. Res. Logist. **35**(2), 177–184 (1988)
27. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming. Springer, New York (1997)
28. Bixby, R., Burda, R., Miller, D.: Short-interval detailed production scheduling in 300mm semiconductor manufacturing using mixed integer and constraint programming. In: Proceedings of the 2006 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, pp. 148–154 (2006)
29. Blackstone, J.H., Hogg, G.L., Phillips, D.T.: A state-of-the-art survey of dispatching rules for manufacturing job shop operations. Int. J. Prod. Res. **20**(1), 27–45 (1982)
30. Blazewicz, J., Pesch, E., Sterna, M.: The disjunctive graph machine representation of the job shop scheduling problem. Eur. J. Oper. Res. **127**(2), 317–331 (2000)
31. Box, G.E.P.: Robustness in the strategy of model building. In: Launer, R.L., Wilkinson, G.N. (eds.) Robustness in Statistics. Academic Press, New York (1979)
32. Brandimarte, P., Rigodanza, M., Roero, L.: Conceptual modeling of an object-oriented scheduling architecture based on the shifting bottleneck procedure. IIE Trans. **32**(10), 921–929 (2000)
33. Brown, S., Chance, F., Fowler, J.W., Robinson, J.K.: A centralized approach to factory simulation. Future Fab Int. **3**, 83–86 (1997)
34. Brucker, P.: Scheduling Algorithms, 5th edn. Springer, Berlin (2007)
35. Brucker, P., Knust, S.: Complex Scheduling. Springer, Berlin (2006)

36. Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M.Y., Potts, C.N., Tautenhahn, T., van de Velde, S.: Scheduling a batching machine. J. Scheduling **1**(1), 31–54 (1998)
37. Bullock, M., Fowler, J.W., Pfund, M.E.: Evaluation of lot dispatching rules for semiconductor manufacturing. In: Proceedings of the 11th Annual Industrial Engineering Research Conference (2003)
38. Bussmann, S.: An agent-oriented design methodology for production control. PhD thesis, University of Southampton (2003)
39. Bussmann, S., Jennings, N.R., Wooldridge, M.: On the identification of agents in the design of production control systems. In: Proceedings First International Workshop on Agent-Oriented Software Engineering, pp. 141–162 (2001)
40. Bussmann, S., Jennings, N.R., Wooldridge, M.: Re-use of interaction protocols for agent-based control applications. In: Proceedings Third International Workshop on Agent-Oriented Software Engineering, pp. 73–87 (2003)
41. Byrne, P.J.: An analysis of semiconductor reticle management using discrete event simulation. In: Proceedings of the 2007 Summer Computer Simulation Conference, pp. 593–600 (2007)
42. Cakici, E., Mason, S.J.: Parallel machine scheduling subject to auxiliary resource constraints. Prod. Plann. Contr. **18**(3), 217–225 (2007)
43. Chambers, R.J., Carraway, R.L., Lowe, T.J., Morin, T.L.: Dominance and decomposition heuristics for single machine scheduling. Oper. Res. **39**(4), 639–647 (1991)
44. Chandra, P., Gupta, S.: Managing batch processors to reduce lead time in a semiconductor packaging line. Int. J. Prod. Res. **35**(3), 611–633 (1997)
45. Chen, Y., Pfund, M.E., Fowler, J.W., Montgomery, D.C., Callarman, T.E.: Robust scaling parameters for composite dispatching rules. IIE Trans. **42**(11), 842–853 (2010)
46. Cheng, R.C.H., Kleijnen, J.P.C.: Improved design of queueing simulation experiments with highly heteroscedastic responses. Oper. Res. **47**(5), 762–777 (1999)
47. Cherbakov, L., Meissner, G., Osipov, C., Walker, L.: Service Oriented Architecture—SOA: IBM microelectronics "factory in a box" (2007). http://www.ibm.com/developerworks/webservices/library/ws-soa-in-action/
48. Chiang, T.-C., Cheng, H.-C., Fu, L.-C.: A memetic algorithm for minimizing total weighted tardiness on parallel batch machines with incompatible job families and dynamic job arrival. Comput. Oper. Res. **37**(12), 2257–2269 (2010)
49. Chien, C.-F., Dauzère-Pérès, S., Ehm, H., Fowler, J.W., Jiang, Z., Krishnaswamy, S., Mönch, L., Uzsoy, R.: Modelling and analysis of semiconductor manufacturing in a shrinking world: challenges and successes. Eur. J. Ind. Eng. **5**(3), 254–271 (2011)
50. Chung, S.-L., Jeng, M.: An overview of semiconductor fab automation systems. In: Proceedings of the 2003 IEEE International Conference on Robotics and Automation, pp. 1050–1055 (2003)
51. Cigolini, R., Perona, M., Portioli, A.: Comparison of order review and release techniques in a dynamic and uncertain job shop environment. Int. J. Prod. Res. **36**(11), 2931–2951 (1998)
52. Cigolini, R., Perona, M., Portioli, A., Zambelli, T.: A new dynamic look-ahead scheduling procedure for batching machines. J. Scheduling **5**(2), 185–204 (2002)
53. Cochran, J.K., Horng, S.M., Fowler, J.W.: A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. Comput. Oper. Res. **30**(7), 1087–1102 (2003)
54. Cogez, P.: The effect of uncertainty on production release policies. Master's thesis, University of California, Berkeley (1990)
55. Curry, G.L., Feldmann, R.M.: Manufacturing Systems Modeling and Analysis. Springer, Berlin (2009)

56. Dabbas, R.M., Chen, H.-N., Fowler, J.W., Shunk, D.L.: A combined dispatching criteria approach to scheduling semiconductor manufacturing systems. Comput. Ind. Eng. **39**(3-4), 307–324 (2001)
57. Dabbas, R.M., Fowler, J.W.: A new scheduling approach using combined dispatching criteria in wafer fabs. IEEE Trans. Semicond. Manuf. **16**(3), 501–510 (2003)
58. Dabbas, R.M., Fowler, J.W., Rollier, D.A., McCarville, D.: Multiple response optimization using mixture-designed experiments and desirability functions in semiconductor scheduling. Int. J. Prod. Res. **41**(5), 939–961 (2003)
59. Dauzère-Pérès, S., Lassere, J.B.: A modified shifting bottleneck procedure for job-shop scheduling. Int. J. Prod. Res. **31**(4), 923–932 (1993)
60. Davis, W.J.: On-line simulation: need and evolving research requirements. In: Banks, J. (ed.) Handbook of Simulation, pp. 465–516. Wiley, New York (1998)
61. Dayhoff, J.E., Atherton, R.W.: Signature analysis of dispatch schemes in wafer fabrication. IEEE Trans. Compon.Hybrids Manuf. Tech. **9**(4), 518–525 (1986)
62. Deb, R., Serfozo, R.F.: Optimal control of batch service queues. Adv. Appl. Probab. **5**(2), 340–361 (1973)
63. Demirkol, E., Uzsoy, R.: Performance of decomposition methods for complex workshops under multiple criteria. Comput. Ind. Eng. **33**(1/2), 261–264 (1997)
64. Demirkol, E., Uzsoy, R.: Decomposition methods for reentrant flow shops with sequence dependent setup-times. J. Scheduling **3**(3), 155–177 (2000)
65. Demirkol, E., Mehta, S., Uzsoy, R.: A computational study of shifting bottleneck procedures for shop scheduling problems. J. Heuristics **3**(2), 111–137 (1997)
66. Derringer, G., Suich, R.: Simultaneous optimization of several response variables. J. Qual. Tech. **12**(4), 214–219 (1980)
67. Devpura, A., Fowler, J.W., Carlyle, M.W., Perez, I.: Minimizing total weighted tardiness on single batch process machine with incompatible job families. In: Proceedings of the Symposium on Operations Research (OR 2000), pp. 366–371 (2000)
68. Dìaz, S.L., Fowler, J.W., Pfund, M.E., Mackulak, G.T., Hickie, M.: Evaluating the impact of reticle requirements in semiconductor wafer fabrication. IEEE Trans. Semicond. Manuf. **18**(4), 622–632 (2005)
69. Dorndorf, U., Pesch, E.: Evolution based learning in a job shop scheduling environment. Comput. Oper. Res. **22**(1), 25–40 (1995)
70. Dowsland, K., Dowsland, W.: Packing problems. Eur. J. Oper. Res. **56**(1), 2–14 (1992)
71. Driessel, R., Mönch, L.: Simulation framework for complex manufacturing systems with automated material handling. In: Proceedings of the 2007 Winter Simulation Conference, pp. 1713–1721 (2007)
72. Driessel, R., Mönch, L.: Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times. Comput. Ind. Eng. **61**(2), 336–345 (2011)
73. Driessel, R., Mönch, L.: An integrated scheduling and material handling approach for complex job shops: a computational study. Int. J. Prod. Res., accepted for publication (2012)
74. Duarte, B.M., Fowler, J.W., Knutson, K., Gel, E., Shunk, D.L.: A compact abstraction of manufacturing nodes in a supply network. Int. J. Simulat. Process Model. **3**(3), 115–126 (2007)
75. Dümmler, M.: Using simulation and genetic algorithms to improve cluster tool performance. In: Proceedings of the 1999 Winter Simulation Conference, pp. 875–879 (1999)
76. Dümmler, M.: Modeling and optimization of cluster tools in semiconductor manufacturing. PhD thesis, University of Würzburg (2004)
77. FabTime. Fabtime dispatching module (2011). http://www.fabtime.com/dispatch.shtml

78. Falkenauer, E.: Genetic Algorithms and Grouping Problems. Wiley, Chichester (1998)
79. FIPA. Foundation for Intelligent Physical Agents (FIPA) (2011). http://www.fipa.org/
80. Fischmann, C., Böttinger, F., Wertz, R., Kunz, C.: Buffer management for automated material handling systems in semiconductor industries. In: Proceedings 22nd European Conference on Modeling and Simulation (ECMS 2008), pp. 423–427 (2008)
81. Fleischmann, B., Meyr, H., Wagner, M.: Advanced planning. In: Stadtler, H., Kilger, C. (eds.) Supply Chain Management and Advanced Planning, 4th edn., pp. 81–106. Springer, Berlin (2008)
82. Foster, L., Pillai, D.: 300 mm wafer fab logistics and automated material handling systems. In: Doering, R., Nishi, Y. (eds.) Handbook of Semiconductor Manufacturing Technology, pp. 33–1–33–67. CRC Press, USA (2007)
83. Fowler, J.W., Robinson, J.: Measurement and improvement of manufacturing capacity (MIMAC): Final report. Technical Report 95062861A-TR, SEMATECH, Austin, TX (1995)
84. Fowler, J.W., Phillips, D.T., Hogg, G.L.: Real-time control of multiproduct bulk-service semiconductor manufacturing processes. IEEE Trans. Semicond. Manuf. **5**(2), 158–163 (1992)
85. Fowler, J.W., Hogg, G.L., Phillips, D.T.: Control of multiproduct bulk server diffusion/oxidation processes. Part 2: multiple servers. IIE Trans. Scheduling Logistics **32**(2), 167–176 (2000)
86. Fowler, J.W., Park, S., Mackulak, G.T., Shunk, D.L.: Efficient cycle time-throughput curve generation using a fixed sample size procedure. Int. J. Prod. Res. **39**(12), 2595–2613 (2001)
87. Fowler, J.W., Hogg, G.L., Mason, S.J.: Workload control in the semiconductor industry. Prod. Plann. Contr. **13**(7), 568–578 (2002)
88. Fowler, J.W., Horng, S., Cochran, J.K.: A hybridized genetic algorithm to solve parallel machine scheduling problems with sequence dependent setups. Int. J. Ind. Eng. **10**(3), 232–243 (2003)
89. Fowler, J.W., Rose, O.: Grand challenges in modeling and simulation of complex manufacturing systems. SIMULATION—Trans. Soc. Model. Simulat. Int. **80**(9), 469–476 (2004)
90. Framinan, J.M., Ruiz, R.: Architecture of manufacturing scheduling systems: literature review and an integrated proposal. Eur. J. Oper. Res. **205**(2), 237–246 (2010)
91. Fu, M.C., Antradóttir, S., Carson, J.S., Glover, F., Harrell, C.R., Ho, Y.-C., Kelly, J.P., Robinson, S.M.: Integrating optimization and simulation: research and practice. In: Proceedings of the 2000 Winter Simulation Conference, pp. 610–616 (2000)
92. Fu, M.C.: Optimization for simulation: theory vs. practice. INFORMS J. Comput. **14**(3), 192–215 (2002)
93. Gan, B.P., Liow, M., Gupta, A.K., Lendermann, P., Turner, S.J., Wang, X.G.: Analysis of a borderless fab using interoperating AutoSched AP models. Int. J. Prod. Res. **45**(3), 675–697 (2007)
94. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
95. Geiger, C.D., Uzsoy, R.: Learning effective dispatching rules for batch processor scheduling. Int. J. Prod. Res. **46**(6), 1431–1454 (2008)
96. Geiger, C.D., Uzsoy, R., Aytug, H.: Rapid modeling and discovery of priority dispatching rules: an autonomous learning approach. J. Scheduling **9**(1), 7–34 (2006)
97. Geng, N., Jiang, Z.: A review on strategic capacity planning in the semiconductor manufacturing industry. Int. J. Prod. Res. **47**(13), 3639–3655 (2009)

98. Geng, N., Jiang, Z., Chen, F.: Stochastic programming based capacity planning for semiconductor wafer fab with uncertain demand. Eur. J. Oper. Res. **198**, 899–908 (2009)
99. Glassey, P.G.: A comparison of release rules using BLOCS/M simulation. Master's thesis, University of California, Berkeley (1990)
100. Glassey, C.R., Resende, M.G.C.: Closed-loop job release control for VLSI circuit manufacturing. IEEE Trans. Semicond. Manuf. **1**(1), 36–46 (1988)
101. Glassey, C.R., Weng, W.W.: Dynamic batching heuristic for simultaneous processing. IEEE Trans. Semicond. Manuf. **4**(2), 77–82 (1991)
102. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publisher, Norwell, MA (1997)
103. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
104. Goldratt, E.M., Cox, J.I.: The Goal: A Process of Ongoing Improvement. North River Press, Croton-on-Hudson, New York (1986)
105. Goldratt, E.M., Fox, R.: The Race. North River Press, Croton-on-Hudson, New York (1986)
106. Gören, S., Sabuncuoglu, I.: Robustness and stability measures for scheduling: single-machine environment. IIE Trans. **40**(1), 66–83 (2008)
107. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math. **5**, 287–326 (1979)
108. Graves, S.C.: A tactical planning model for a job shop. Oper. Res. **34**(4), 522–533 (1986)
109. Günther, H.-O.: Supply chain management and advanced planning systems: a tutorial. In: Günther, H.-O., Mattfeld, D.C., Suhl, L. (eds.) Supply Chain Management und Logistik: Optimierung, Simulation, Decision Support, pp. 3–40. Physica, Heidelberg (2005)
110. Gupta, J.N.D., Ruiz, R., Fowler, J.W., Mason, S.J.: Operational planning and control of semiconductor wafer production. Prod. Plann. Contr. **17**(7), 639–647 (2006)
111. Gurnani, H., Anupindi, R., Akella, R.: Control of batch processing systems in semiconductor wafer fabrication facilities. IEEE Trans. Semicond. Manuf. **5**(4), 319–328 (1992)
112. Habenicht, I., Mönch, L.: A finite-capacity beam-search-algorithm for production scheduling in semiconductor manufacturing. In: Proceedings of the 2002 Winter Simulation Conference, pp. 1406–1413 (2002)
113. Habla, C., Mönch, L.: Solving volume and capacity planning problems in semiconductor manufacturing: a computational study. In: Proceedings of the 2008 Winter Simulation Conference, pp. 2260–2266 (2008)
114. Ham, M., Fowler, J.W.: Scheduling of wet etch and furnace operations with next arrival control heuristic. Int. J. Adv. Manuf. Technol. **38**, 1006–1017 (2008)
115. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001)
116. Haupt, R.: A survey of priority rule-based scheduling. OR Spektrum **11**(1), 3–16 (1989)
117. Hood, S.J., Bermon, S., Barahona, F.: Capacity planning under demand uncertainty for semiconductor manufacturing. IEEE Trans. Semicond. Manuf. **16**(2), 273–280 (2003)
118. Hooke, R., Jeeves, T.A.: Direct search solution of numerical statistical problems. J ACM **8**(2), 212–229 (1961)
119. Hopp, W.J., Spearman, M.L.: Factory Physics, 2nd edn. Irwin, McGraw-Hill, Boston (2000)

120. Hung, Y.-F., Leachman, R.C.: A production planning methodology for semiconductor manufacturing based on iterative simulation and linear programming calculations. IEEE Trans. Semicond. Manuf. **9**(2), 257–269 (1996)
121. Hung, Y.-F., Leachman, R.C.: Reduced simulation models of wafer fabrication facilities. Int. J. Prod. Res. **37**(12), 2685–2701 (1999)
122. Hutcheson, J.D.: Introduction to semiconductor equipment. In: Nishi, Y., Doering, R. (eds.) Handbook of Semiconductor Manufacturing Technology, pp. 23–33. Marcel Dekker, New York (2000)
123. Irdem, D.F.: Evaluation of clearing functions' fitting methodology and performance of production planning models. Master's thesis, North Carolina State University, Raleigh, North Carolina (2009)
124. Irdem, D.F., Kacar, N.B., Uzsoy, R.: An exploratory analysis of two iterative linear programming—simulation approaches for production planning. IEEE Trans. Semicond. Manuf. **23**(3), 442–455 (2010)
125. Jacobs, F.R.: OPT uncovered: many production and scheduling concepts can be applied with or without the software. J. Ind. Eng. **16**(10), 32–41 (1984)
126. Jampani, J., Mason, S.J.: Column generation heuristic for complex job shop multiple orders per job scheduling. Comput. Ind. Eng. **58**(1), 108–118 (2010)
127. Jampani, J., Pohl, E.A., Mason, S.J., Mönch, L.: Integrated heuristics for scheduling multiple order jobs in a complex job shop. IJMHeur **1**(2), 156–180 (2010)
128. Jeong, B.H., Randhawa, S.U.: A multi-attribute dispatching rule for automated guided vehicle systems. Int. J. Prod. Res. **39**(13), 2817–2832 (2001)
129. Jia, J., Mason, S.J.: Semiconductor manufacturing scheduling of jobs containing multiple orders on identical parallel machines. Int. J. Prod. Res. **47**(10), 2565–2585 (2009)
130. Jimenez, J., Kim, B., Fowler, J.W., Mackulak, G., Choung, Y.I., Kim, D.-J.: Operational modeling and simulation of an inter-bay AMHS in semiconductor wafer fabrication. In: Proceedings of the 2002 Winter Simulation Conference, pp. 1377–1382 (2002)
131. Jimenez, J., Mackulak, G.T., Fowler, J.W.: Levels of capacity and material handling system modeling for factory integration decision making in semiconductor wafer fabs. IEEE Trans. Semicond. Manuf. **21**(4), 600–613 (2008)
132. Johnson, R.T., Yang, F., Ankenman, B.E., Nelson, B.L.: Nonlinear regression fits for simulated cycle time vs. throughput curves for semiconductor manufacturing. In: Proceedings of the 2004 Winter Simulation Conference, pp. 1951–1955 (2004)
133. Johnzén, C.: Modeling and optimizing flexible capacity allocation in semiconductor manufacturing. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne (2009)
134. Kacar, N.B., Uzsoy, R.: Estimating clearing functions from simulation data. In: Proceedings of the 2010 Winter Simulation Conference, pp. 1699–1710 (2010)
135. Kallrath, J., Maindl, T.I.: Real Optimization with SAP APO. Springer, Berlin (2006)
136. Karmarkar, U.S.: Capacitiy loading and release planning with work-in-progress (WIP) and lead-times. J. Manuf. Oper. Manag. **2**, 105–123 (1989)
137. Kempf, K., Uzsoy, R., Wang, C.-S.: Scheduling a single batch processing machine with secondary ressource constraints. J. Manuf. Syst. **17**(1), 37–51 (1998)
138. Kendall, D.G.: Stochastic processes occuring in the theory of queues and their analysis by the method of the imbedded Markov chain. Ann. Math. Stochast. **24**(3), 338–354 (1953)
139. Kim, B., Kim, S.: Extended model for a hybrid production planning approach. Int. J. Prod. Econ. **73**(2), 165–173 (2001)
140. Kim, S.Y., Lee, Y.H., Agnihotri, D.: A hybrid approach to sequencing jobs using heuristic rules and neural networks. Prod. Plann. Contr. **6**(5), 445–454 (1995)

141. Kimms, A.: Stability measures for rolling schedules with applications to capacity expansion planning, master production scheduling and lot sizing. Omega—Int. J. Manag. Sci. **26**(3), 355–366 (1998)
142. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
143. Kleijnen, J.P.C., van Beers, W.C.M.: Application-driven sequential designs for simulation experiments: Kriging metamodeling. J. Oper. Res. Soc. **55**(9), 876–883 (2004)
144. Kleinberg, J., Tardos, E.: Algorithm Design. Pearson/Addison-Wesley, Boston (2006)
145. Klemmt, A., Horn, S., Weigert, G., Hielscher, T.: Simulation-based and solver-based optimization approaches for batch processes in semiconductor manufacturing. In: Proceedings of the 2008 Winter Simulation Conference, pp. 2041–2049 (2008)
146. Klemmt, A., Weigert, G., Almeder, C., Mönch, L.: A comparison of MIP-based decomposition techniques and VNS approaches for batch scheduling problems. In: Proceedings of the 2009 Winter Simulation Conference, pp. 1686–1694 (2009)
147. Klemmt, A., Lange, J., Weigert, G., Lehmann, F., Seyfert, J.: A multistage mathematical programming based scheduling approach for the photolithography area in semiconductor manufacturing. In: Proceedings of the 2010 Winter Simulation Conference, pp. 2474–2485 (2010)
148. Knust, S.: Shop-scheduling problems with transportation. PhD thesis, University of Osnabrück (1999)
149. Laub, J.D., Fowler, J.W., Keha, A.B.: Minimizing makespan with multiple-orders-per-job in a two-machine flowshop. Eur. J. Oper. Res. **182**(1), 63–79 (2007)
150. Law, A.W.: Simulation Modeling and Analysis, 4th edn. McGraw-Hill, Boston (2007)
151. Lawler, E.L.: A "pseudopolynomial" time algorithm for sequencing jobs to minimize total weighted tardiness. Ann. Discrete Math. **1**, 331–342 (1977)
152. Le Pape, C.: Implementation of resource constraints in ILOG SCHEDULE: a library for the development of constraint-based scheduling systems. Intell. Syst. Eng. **3**(2), 55–66 (1994)
153. Leach, S.E., Fowler, J.W., Mackulak, G.T., Nelson, B.L., Marquis, J.L.: Asymptotic variance-based sampling for simulating cycle time—throughput curves. ASU Working Paper Series ASUIE-ORPS-2005-003, Arizona State University, Tempe (2005)
154. Leachman, R.C.: Production planning and scheduling practises across the semiconductor industry. SEMATECH Technology Transfer 94092559A-XFR (1995)
155. Leachman, R.C., Hodges, D.A.: Benchmarking semiconductor manufacturing. IEEE Trans. Semicond. Manuf. **9**(2), 158–169 (1996)
156. Leachman, R.C., Solorzano, M., Glassey, C.R.: A queue management policy for the release of factory work orders. In: Proceedings of the ORSA/TIMS Conference (1989)
157. Lee, T.-E.: A review of scheduling theory and methods for semiconductor manufacturing cluster tools. In: Proceedings of the 2008 Winter Simulation Conference, pp. 2127–2135 (2008)
158. Lee, T.-E.: Semiconductor manufacturing automation. In: Nof, S. (ed.) Handbook on Automation, pp. 911–926. Springer, Berlin (2009)
159. Lee, Y.-H., Pinedo, M.: Scheduling jobs on parallel machines with sequence-dependent setup times. Eur. J. Oper. Res. **100**(3), 464–474 (1997)
160. Levine, D.: Application of a hybrid genetic algorithm to airline crew scheduling. Comput. Oper. Res. **23**(6), 547–558 (1996)

161. Liao, D.-Y., Fu, H.-S.: Speedy delivery—dynamic OHT allocation and dispatching in large-scale, 300-mm AMHS management. IEEE Robot. Autom. Mag. **11**(3), 22–32 (2004)
162. Liao, D.-Y., Wang, C.-N.: Differentiated preemptive dispatching for automatic materials handling services in 300 mm semiconductor foundry. Int. J. Adv. Manuf. Tech. **29**(9-10), 890–896 (2005)
163. Lin, J.T., Wang, F.-K., Yen, P.-Y.: Simulation analysis of dispatching rules for an automated interbay material handling system in wafer fab. Int. J. Prod. Res. **39**(6), 1221–1238 (2001)
164. Lin, C.-H., Hwang, S.-L., Wang, M.-Y.E.: The mythical advanced planning systems in complex manufacturing environment. In: Proceedings of the 12th IFAC Conference on Information Control Problems in Manufacturing, pp. 691–696 (2006)
165. Little, J.D.C.: A proof for the queuing formula $L = \lambda W$. Oper. Res. **9**(3), 383–387 (1961)
166. Lou, S.X., Kager, P.W.: A robust production control policy for VLSI wafer fabrication. IEEE Trans. Semicond. Manuf. **2**(4), 159–164 (1989)
167. Lou, S., Yan, H., Sethi, S., Gardel, A., Deosthali, P.: Hub-centred production control of wafer fabrication. In: Proceedings of the 1990 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, pp. 27–32 (1990)
168. Lozinski, C., Glassey, C.R.: Bottleneck starvation indicators for shop floor control. IEEE Trans. Semicond. Manuf. **1**(4), 147–153 (1988)
169. Lu, S.C.H., Ramaswamy, D., Kumar, P.R.: Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants. IEEE Trans. Semicond. Manuf. **7**(3), 374–388 (1994)
170. Martin-Vega, L.A., Pippin, M., Gerdon, E., Burcham, R.: Applying just-in-time in a wafer fab: a case study. IEEE Trans. Semicond. Manuf. **2**(1), 16–21 (1989)
171. Mason, S.J., Chen, J.-S.: Scheduling multiple orders per job in a single machine to minimize total completion time. Eur. J. Oper. Res. **207**(1), 70–77 (2010)
172. Mason, S.J., Fowler, J.W., Carlyle, M.W.: A modified shifting bottleneck heuristic for minimizing the total weighted tardiness in a semiconductor wafer fab. J. Scheduling **5**(3), 247–262 (2002)
173. Mason, S.J., Jin, S., Wessels, M.C.: Rescheduling strategies for minimizing total weighted tardiness in complex job shops. Int. J. Prod. Res. **42**(3), 613–628 (2004)
174. Mason, S.J., Qu, P., Kutanoglu, E., Fowler, J.W.: The single machine multiple orders per job scheduling problem. Technical Report ASUIE-ORPS-2004-004, Arizona State University, Tempe, AZ (2004)
175. Mason, S.J., Fowler, J.W., Carlyle, M.W., Montgomery, D.C.: Heuristics for minimizing total weighted tardiness in complex job shops. Int. J. Prod. Res. **43**(10), 1943–1963 (2005)
176. Mathirajan, M., Sivakumar, A.I.: A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. Int. J. Adv. Manuf. Tech. **29**(9-10), 990–1001 (2006)
177. McAllister, C.D., Altunas, B., Frank, M., Potoradi, J.: Implementation of response surface methodology using variance reduction techniques in semiconductor manufacturing. In: Proceedings of the 2001 Winter Simulation Conference, pp. 1225–1230 (2001)
178. McClellan, M.: Applying Manufacturing Execution Systems. St. Lucie Press, Boca Raton (1997)
179. McFarlane, D.C., Bussmann, S.: Holonic manufacturing control: rationales, developments and open issues. In: Deen, M.S. (ed.) Agent-Based Manufacturing—Advances in the Holonic Approach, pp. 303–326. Springer, Berlin (2003)
180. Mehta, S.V., Uzsoy, R.: Minimizing total tardiness on a batch processing machine with incompatible job families. IIE Trans. **30**(2), 165–178 (1998)

181. Mesarović, M.D., Takahara, Y.: Abstract Systems Theory. Lecture Notes in Control and Information Sciences. Springer, Berlin (1989)
182. Meyer, H., Fuchs, F., Thiel, K.: Manufacturing Execution Systems: Optimal Design, Planning, and Deployment. McGraw-Hill, New York (2009)
183. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer, Berlin (1996)
184. Miller, D.J.: Simulation of a semiconductor manufacturing line. Comm. ACM **30**(10), 98–108 (1990)
185. Missbauer, H.: Aggregate order release planning for time-varying demand. Int. J. Prod. Res. **40**(3), 699–718 (2002)
186. Mittler, M., Schömig, A.: Comparison of dispatching rules for reducing the mean and variability of cycle times in semiconductor manufacturing. In: Proceedings of the International Symposium on Operations Research, pp. 479–485 (1999)
187. Mittler, M., Schömig, A.: Comparison of dispatching rules for semiconductor manufacturing using large facility models. In: Proceedings of the 1999 Winter Simulation Conference, pp. 709–713 (1999)
188. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
189. Mönch, L.: A genetic algorithm heuristic applied to stepper scheduling. In: Proceedings of the International Conference on Modelling and Analysis of Semiconductor Manufacturing (MASM 2002), pp. 276–281 (2002)
190. Mönch, L.: Scheduling-Framework für Jobs auf parallelen Maschinen in komplexen Produktionssystemen. WIRTSCHAFTSINFORMATIK **46**(6), 470–480 (2004)
191. Mönch, L.: Simulation-based assessment of order release strategies for a distributed shifting bottleneck heuristic. In: Proceedings of the 2005 Winter Simulation Conference, pp. 2186–2193 (2005)
192. Mönch, L.: Simulation-based benchmarking of production control schemes for complex manufacturing systems. Contr. Eng. Pract. **15**(11), 1381–1393 (2007)
193. Mönch, L., Driessel, R.: A distributed shifting bottleneck heuristic for complex job shops. Comput. Ind. Eng. **49**(3), 363–380 (2005)
194. Mönch, L., Habenicht, I.: Simulation-based assessment of batching heuristics in semiconductor manufacturing. In: Proceedings of the 2003 Winter Simulation Conference, pp. 1338–1345 (2003)
195. Mönch, L., Stehli, M.: An ontology for production control of semiconductor manufacturing processes. In: Proceedings First German Conference on Multiagent System Technologies (MATES 2003), LNAI 2831, pp. 156–167 (2003)
196. Mönch, L., Stehli, M.: A content language for a hierarchically organized multi-agent-system for production control. In: Proceedings Multi-Konferenz Wirtschaftsinformatik, Teilkonferenz "Coordination and Agent Technology in Value Networks", pp. 197–212 (2004)
197. Mönch, L., Zimmermann, J.: Improving the performance of dispatching rules in semiconductor manufacturing by iterative simulation. In: Proceedings of the 2004 Winter Simulation Conference, pp. 1881–1887 (2004)
198. Mönch, L., Zimmermann, J.: Simulation-based assessment of machine criticality measures for a shifting bottleneck heuristic in complex manufacturing systems. Comput. Ind. **58**(7), 644–655 (2007)
199. Mönch, L., Zimmermann, J.: Providing production planning and control functionality by web services: state of the art and experiences with prototypes. In: Proceedings of the 5th Annual IEEE Conference on Automation Science and Engineering, pp. 495–500 (2009)
200. Mönch, L., Zimmermann, J.: A computational study of a shifting bottleneck heuristic for multi-product complex job shops. Prod. Plann. Contr. **22**(1), 25–40 (2011)

201. Mönch, L., Prause, M., Schmalfuß, V.: Simulation-based solution of load-balancing problems in the photolithography area of a semiconductor wafer fabrication facility. In: Proceedings of the 2001 Winter Simulation Conference, pp. 1170–1177 (2001)
202. Mönch, L., Rose, O., Sturm, R.: A simulation framework for the performance assessment of shop-floor control systems. SIMULATION—Transactions of the Society of Modeling and Simulation International **79**(3), 163–170 (2003)
203. Mönch, L., Balasubramanian, H., Fowler, J.W., Pfund, M.E.: Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. Comput. Oper. Res. **32**(11), 2731–2750 (2005)
204. Mönch, L., Stehli, M., Zimmermann, J., Habenicht, I.: The FABMAS multi-agent-system prototype for production control of wafer fabs: design, implementation and performance assessment. Prod. Plann. Contr. **17**(7), 701–716 (2006)
205. Mönch, L., Zimmermann, J., Otto, P.: Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines. J. Eng. Appl. Artif. Intell. **19**(3), 235–245 (2006)
206. Mönch, L., Schabacker, R., Pabst, D., Fowler, J.W.: Genetic algorithm-based subproblem solution procedures for a modified shifting bottleneck heuristic for complex job shops. Eur. J. Oper. Res. **177**(3), 2100–2118 (2007)
207. Mönch, L., Fowler, J.W., Mason, S.J., Dauzère-Pérès, S., Rose, O.: A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. J. Scheduling **14**(6), 583–599 (2011)
208. Montgomery, D.C.: Design and Analysis of Experiments, 5th edn. Wiley, New York (2001)
209. Montoya-Torres, J.R.: A literature survey on the design approaches and operational issues of automated wafer-transport systems for wafer fabs. Prod. Plann. Contr. **17**(7), 648–663 (2006)
210. Mosley, S.A., Teyner, T., Uzsoy, R.: Maintenance scheduling and staffing policies in a wafer fabrication facility operating under theory of constraints. IEEE Trans. Semicond. Manuf. **11**(2), 316–323 (1998)
211. Moyne, J., del Castillo, E.D., Hurwitz, A.: Run-to-Run Control in Semiconductor Manufacturing. CRC Press, Chichester (2001)
212. Myers, R.H., Montgomery, D.C., Anderson-Cook, C.M.: Response Surface Methodology: Process and Product Optimization Using Designed Experiments, 3rd edn. Wiley, New Jersey (2009)
213. Najmi, A., Lozinski, C.: Managing factory productivity using object-oriented simulation for setting shift production targets in VLSI manufacturing. In: Proceedings AUTOFACT '89, pp. (3–1)–(3–14) (1989)
214. Nemhauser, G., Wolsey, L.A.: Integer and Combinatorial Optimization, 3rd edn. Wiley, New York (1999)
215. Nowicki, E., Smutnicki, C.: An advanced tabu search algorithm for the job shop problem. J. Scheduling **8**(2), 145–159 (2005)
216. Occhino, T.J.: Capacity planning model: the important inputs, formulas, and benefits. In: Proceedings of the 2000 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, pp. 455–458 (2000)
217. Odell, J.J., Parunak, H.V.D., Fleischer, M.: Modeling agent organizations using roles. Software Syst. Model. **2**(2), 76–81 (2003)
218. Oechsner, S., Rose, O.: A filtered beam search approach to scheduling cluster tools in semiconductor manufacturing. In: Proceedings of the Industrial Engineering Research Conference (2005)
219. Oechsner, S., Rose, O.: Scheduling cluster tools using filtered beam search and recipe comparison. In: Proceedings of the 2005 Winter Simulation Conference, pp. 2203–2210 (2005)

220. OR-Library: OR-Library: a collection of test data sets for a variety of Operations Research (OR) problems (2008). http://people.brunel.ac.uk/~mastjjb/jeb/info.html
221. Osisek, V., Aytug, H.: Discovering subproblem prioritization rules for shifting bottleneck algorithms. J. Intell. Manuf. **15**(1), 55–67 (2003)
222. Ovacik, I.M., Uzsoy, R.: Rolling horizon procedures for dynamic parallel machine scheduling with sequence dependent setup times. Int. J. Prod. Res. **33**(11), 3173–3192 (1995)
223. Ovacik, I.M., Uzsoy, R.: Decomposition Methods for Complex Factory Scheduling Problems. Kluwer, Norwell (1997)
224. Ozturk, O., Coburn, M.B., Kitterman, S.: Conceptualization, design and implementation of a static capacity model. In: Proceedings of the 2003 Winter Simulation Conference, pp. 1373–1376 (2003)
225. Pabst, D.: Handling precedence constraints for the shifting bottleneck heuristic applied in a dynamic semiconductor manufacturing environment. Master's thesis, Institut für Informatik, Universität Würzburg, Würzburg (2003)
226. Pahl, J., Voß, S., Woodruff, D.L.: Production planning with load dependent lead times: an update of research. Ann. Oper. Res. **153**(1), 297–345 (2007)
227. Pain, A.R., Reeves, C.R.: Genetic algorithm optimization software class libraries. In: Voß, S., Woodruff, D.L. (eds.) Optimization Software Class Libraries, pp. 295–329. Springer, New York (2002)
228. Park, S., Fowler, J.W., Carlyle, M.W., Hickie, M.: Assessment of potential gains in productivity due to proactive reticle management using discrete event simulation. In: Proceedings of the 1999 Winter Simulation Conference, pp. 856–864 (1999)
229. Park, Y., Kim, S., Lee, Y.-H.: Scheduling jobs on parallel machines applying neural networks and heuristic rules. Comput. Ind. Eng. **38**(1), 189–202 (2000)
230. Park, S., Fowler, J.W., Mackulak, G.T., Keats, J.B., Carlyle, W.M.: D-optimal sequential experiments for generating a simulation-based cycle time-throughput curve. Oper. Res. **50**(6), 981–990 (2002)
231. Peikert, A., Thoma, J., Brown, S.: A rapid modeling technique for measurable improvements in factory performance. In: Proceedings of the 1998 Winter Simulation Conference, pp. 1011–1016 (1998)
232. Perez, I.C., Fowler, J.W., Carlyle, W.M.: Minimizing total weighted tardiness on a single batch processing machine with incompatible job families. Comput. Oper. Res. **32**(2), 327–341 (2005)
233. Pfeiffer, A., Kadar, B., Monostori, L.: Stability-oriented evaluation of rescheduling strategies by using simulation. Comput. Ind. **58**(7), 630–643 (2007)
234. Pfund, M.E., Mason, S.J., Fowler, J.W.: Semiconductor manufacturing scheduling and dispatching. In: Herrmann, J.W. (ed.) Handbook of Production Scheduling, pp. 213–241. Springer, New York (2006)
235. Pfund, M.E., Balasubramanian, H., Fowler, J.W., Mason, S.J., Rose, O.: A multi-criteria approach for scheduling semiconductor wafer fabrication facilities. J. Scheduling **11**(1), 29–47 (2008)
236. Pfund, M.E., Fowler, J.W., Gadkari, A., Chen, Y.: Scheduling jobs on parallel machines with setup times and ready times. Comput. Ind. Eng. **54**(4), 764–782 (2008)
237. Pickardt, C., Branke, J., Hildebrandt, T., Heger, J., Scholz-Reiter, B.: Generating dispatching rules in semiconductor manufacturing to minimize weighted tardiness. In: Proceedings of the 2010 Winter Simulation Conference, pp. 2504–2515 (2010)
238. Pickerill, J.: Better cycle time and on-time delivery via real-time dispatching. Solid State Tech. **43**(6), 151–154 (2000)

239. Pillai, D.D., Bass, E.L., Dempsey, J.C., Yellig, E.J.: 300-mm full-factory simulations for 90- and 65-nm IC manufacturing. IEEE Trans. Semicond. Manuf. **17**(3), 292–298 (2004)
240. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems, 3rd edn. Springer, New York (2008)
241. Pinedo, M., Chao, X.: Operations Scheduling with Applications in Manufacturing and Services. Irwin McGraw-Hill, New York (1999)
242. Pinedo, M., Singer, M.: A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. Nav. Res. Logist. **46**(1), 1–17 (1999)
243. Pochet, Y., Wolsey, L.A.: Production Planning by Mixed Integer Programming. Springer, New York (2006)
244. Ponsignon, T., Mönch, L.: Architecture for simulation-based performance assessment of planning approaches in semiconductor manufacturing. In: Proceedings of the 2010 Winter Simulation Conference, pp. 3341–3349 (2010)
245. Ponsignon, T., Mönch, L.: Heuristic approaches for master planning in semiconductor manufacturing. Comput. Oper. Res. **39**(3), 479–491 (2012)
246. Potoradi, J., Boon, O.S., Mason, S.J., Fowler, J.W., Pfund, M.E.: Using simulation-based scheduling to maximize demand fulfillment in a semiconductor assembly facility. In: Proceedings of the 2002 Winter Simulation Conference, pp. 1857–1861 (2002)
247. Potts, C.N., Kovalyov, M.Y.: Scheduling with batching: a review. Eur. J. Oper. Res. **120**(2), 228–249 (2000)
248. Preiss, B.R.: Data Structures and Algorithms with Object-Oriented Design Patterns in C++. Wiley, Chichester (1999)
249. Pukelsheim, F.: Optimal Design of Experiments. SIAM, Philadelphia (2006)
250. Qi, C., Sivakumar, A.I., Gershwin, S.B.: Impact of production control and system factors in semiconductor wafer fabrication. IEEE Trans. Semicond. Manuf. **21**(3), 376–389 (2008)
251. Qi, C., Sivakumar, A.I., Gershwin, S.B.: An efficient new job release control methodology. Int. J. Prod. Res. **47**(3), 703–731 (2009)
252. Qiu, R.G.: E-manufacturing: the keystone of a plant-wide real time information system. J. Chin. Inst. Ind. Eng. **20**(3), 266–274 (2003)
253. Qiu, R.G.: A service-oriented integration framework for semiconductor manufacturing systems. Int. J. Manuf. Tech. Manag. **10**(2/3), 177–191 (2007)
254. Qu, P., Mason, S.J.: Metaheuristic scheduling of 300mm lots containing multiple orders. IEEE Trans. Semicond. Manuf. **18**(4), 633–643 (2005)
255. Qiu, R.G., Zhou, M.: Mighty MESs: state-of-the-art and future manufacturing execution systems. IEEE Robot. Autom. Mag. **80**(1), 19–25 (2004)
256. Ragatz, G.L., Mabert, V.A.: An evaluation of order release mechanisms in a job-shop environment. Decis. Sci. **19**(1), 167–189 (1988)
257. Rardin, R.L., Uzsoy, R.: Experimental evaluation of heuristic optimization algorithms: a tutorial. J. Heuristics **7**(3), 261–304 (2001)
258. Rastogi, A.P., Fowler, J.W., Carlyle, W.M., Araz, O.M., Maltz, A., Büke, B.: Supply network capacity planning for semiconductor manufacturing with uncertain demand and correlation in demand considerations. Int. J. Prod. Econ. **134**(2), 322–332 (2011)
259. Reichelt, D., Mönch, L.: Multiobjective scheduling of jobs with incompatible families on parallel batch machines. In: Proceedings EvoCop 2006, LNCS 3906, pp. 209–221 (2006)
260. Robinson, J., Fowler, J.W., Bard, J.F.: The use of upstream and downstream information in scheduling semiconductor batch operations. Int. J. Prod. Res. **33**(7), 1849–1869 (1995)
261. Robinson, J., Fowler, J.W., Bard, J.F.: A review of real-time control strategies for furnace batch sizing in semiconductor manufacturing. Technical report, FabTime Inc. (2000)

262. Roderick, L.M., Phillips, D.T., Hogg, G.L.: A comparison of order release strategies in production control systems. Int. J. Prod. Res. **30**(3), 611–626 (1992)
263. Rose, O.: WIP evolution of a semiconductor factory after a bottleneck workcenter breakdown. In: Proceedings of the 1998 Winter Simulation Conference, pp. 997–1003 (1998)
264. Rose, O.: CONLOAD—a new lot release rule for semiconductor wafer fabs. In: Proceedings of the 1999 Winter Simulation Conference, pp. 850–855 (1999)
265. Rose, O.: Why do simple wafer fab models fail in certain scenarios? In: Proceedings of the 2000 Winter Simulation Conference, pp. 1481–1490 (2000)
266. Rose, O.: CONWIP-like lot release for a wafer fabrication facility with dynamic load changes. In: Proceedings of the SMOMS '01 (ASTC '01), pp. 41–46 (2001)
267. Rose, O.: The shortest processing time first (SPTF) dispatch rule and some variants in semiconductor manufacturing. In: Proceedings of the 2001 Winter Simulation Conference, pp. 1220–1224 (2001)
268. Rose, O.: Some issues of the critical ratio dispatch rule in semiconductor manufacturing. In: Proceedings of the 2002 Winter Simulation Conference, pp. 1401–1405 (2002)
269. Rose, O.: Improved simple simulation models for semiconductor wafer factories. In: Proceedings of the 2007 Winter Simulation Conference, pp. 1708–1712 (2007)
270. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Springer, Berlin (2006)
271. Roundy, R.: Report on practices related to demand forecasting for semiconductor products. Technical report, School of Operations Research and Industrial Engineering, Cornell University (2001)
272. Rudolph, G.: Convergence analysis of canonical genetic algorithms. IEEE Trans. Neural Network **5**(1), 96–101 (1994)
273. Russell, S.J., Norvig, P.: Artificial Intelligence—A Modern Approach, 2nd edn. Prentice Hall, New Jersey (2003)
274. Sarin, S.C., Varadarajan, A., Wang, L.: A survey of dispatching rules for operational control in wafer fabrication. Prod. Plann. Contr. **22**(1), 4–24 (2011)
275. Scholl, W., Domaschke, J.: Implementation of modeling and simulation in semiconductor wafer fabrication with time constraints between wet etch and furnace operations. IEEE Trans. Semicond. Manuf. **13**(3), 273–277 (2000)
276. Schömig, A., Fowler, J.W.: Modeling semiconductor manufacturing operations. In: Proceedings of the 9th ASIM Dedicated Conference Simulation in Production and Logistics, pp. 55–64 (2000)
277. Schömig, A., Rose, O.: On the suitability of the Weibull distribution for the approximation of machine failures. In: Proceedings of the 2003 Industrial Engineering Research Conference (2003)
278. Schulz, M., Stanley, T.D., Renelt, B., Sturm, R., Schwertschlager, O.: Simulation based decision support for future 300mm automated material handling. In: Proceedings of the 2000 Winter Simulation Conference, pp. 1518–1522 (2000)
279. SEMATECH. Computer Integrated Manufacturing (CIM) framework specification 2.0 (1997). http://www.sematech.org/docubase/document/1697jeng.pdf
280. SEMI-E10-0304E. Specification for definition and measurement of equipment reliability, availability, and maintainability (RAM) (2004)
281. Shanthikumar, J.G., Ding, S., Zhang, M.T.: Queueing theory for semiconductor manufacturing systems: a survey and open problems. IEEE Trans. Autom. Sci. Eng. **4**(4), 513–522 (2007)
282. Shikalgar, S.T., Fronckowiak, D., MacNair, E.A.: 300mm wafer fabrication line simulation model. In: Proceedings of the 2002 Winter Simulation Conference, pp. 1365–1368 (2002)
283. Simchi-Levi, D., Kaminsky, P., Simchi-Levi, E.: Designing and Managing the Supply Chain. McGraw-Hill, Boston (2000)

284. Sivakumar, A.I.: Multiobjective dynamic scheduling using discrete event simulation. Int. J. Comput. Integrated Manuf. **14**(2), 154–167 (2001)
285. Sivakumar, A.I., Gupta, A.K.: Online multiobjective Pareto optimal dynamic scheduling of semiconductor back-end using conjunctive simulated scheduling. IEEE Trans. Electron. Packag. Manuf. **29**(2), 99–109 (2006)
286. Sobeyko, O., Mönch, L.: Genetic algorithms to solve a single machine multiple orders per job scheduling problem. In: Proceedings of the 2010 Winter Simulation Conference, pp. 2493–2503 (2010)
287. Solomon, L., Fowler, J.W., Pfund, M.E., Jensen, P.H.: The inclusion of future arrivals and downstream setups into wafer fabrication batch processing decisions. J. Electron. Manuf. **11**(2), 149–159 (2002)
288. Sourirajan, K., Uzsoy, R.: Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication. J. Scheduling **10**(1), 41–65 (2007)
289. Spearman, M.L., Hopp, W.J., Woodruff, D.L.: A hierarchical control architecture for constant work-in-process (CONWIP) production systems. J. Manuf. Oper. Manag. **2**(3), 147–171 (1989)
290. Spearman, M.L., Woodruff, D.L., Hopp, W.J.: CONWIP: a pull alternative to Kanban. Int. J. Prod. Res. **28**(5), 879–894 (1990)
291. Spier, J., Kempf, K.: Simulation of emergent behavior in manufacturing systems. In: Proceedings of the 1995 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, pp. 90–94 (1995)
292. Srinivasan, A., Carey, M., Morton, T.E.: Resource pricing and aggregate scheduling in manufacturing systems. GSIA Working Papers 88-89-58, Carnegie Mellon University, Tepper School of Business, Pittsburgh (1988)
293. Stray, J., Fowler, J.W., Carlyle, W.M., Rastogi, A.P.: Enterprise-wide strategic and logistics planning for semiconductor manufacturing. IEEE Trans. Semicond. Manuf. **19**(2), 259–268 (2006)
294. Sze, S.M.: Semiconductor Devices: Physics and Technology, 2nd edn. Wiley, New York (2001)
295. Tay, J.C., Ho, N.B.: Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput. Ind. Eng. **54**(3), 453–473 (2008)
296. Thiel, M.: Simulationsbasierte Reihenfolgeplanung in der Halbleiterindustrie. PhD thesis, Technical University of Ilmenau (2001)
297. Thiel, M., Schulz, R., Gmilkowsky, P.: Simulation-based production control in the semiconductor industry. In: Proceedings of the 1998 Winter Simulation Conference, pp. 1029–1033 (1998)
298. T'kindt, V., Billaut, J.-C.: Multicriteria Scheduling: Theory, Models and Algorithms, 2nd edn. Springer, Berlin (2006)
299. Turban, E., Aronson, J.E., Liang, T.-P.: Decision Support Systems and Intelligent Systems, 7th edn. Prentice Hall, USA (2005)
300. Tyan, J.C., Du, T.C., Chen, J.C., Chang, I.-H.: Multiple response optimization in an fully automated FAB: an integrated tool and vehicle dispatching strategy. Comput. Ind. Eng. **46**(1), 121–139 (2004)
301. Upasani, A.A., Uzsoy, R.: Integrating a decomposition procedure with problem reduction for factory scheduling with disruptions: a simulation study. Int. J. Prod. Res. **46**(21), 5883–5905 (2008)
302. Upasani, A.A., Uzsoy, R., Sourirajan, K.: A problem reduction approach for scheduling semiconductor wafer fabrication facilities. IEEE Trans. Semicond. Manuf. **19**(2), 216–225 (2006)
303. Uzsoy, R.: Scheduling batch processing machines with incompatible job families. Int. J. Prod. Res. **33**(10), 2685–2708 (1995)

304. Uzsoy, R., Wang, C.-S.: Performance of decomposition procedures for job shop scheduling problems with bottleneck machines. Int. J. Prod. Res. **38**(6), 1271–1286 (2000)
305. Uzsoy, R., Martin-Vega, L.A., Lee, C.Y., Leonard, P.: Production scheduling algorithms for a semiconductor test facility. IEEE Trans. Semicond. Manuf. **4**(4), 270–280 (1991)
306. Uzsoy, R., Lee, C.-Y., Martin-Vega, L.: A review of production planning and scheduling models in the semiconductor industry—Part I: system characteristics, performance evaluation and production planning. IIE Trans. Scheduling Logistics **24**(4), 47–61 (1992)
307. Uzsoy, R., Lee, C.-Y., Martin-Vega, L.: A review of production planning and scheduling models in the semiconductor industry—Part II: shop floor control. IIE Trans. Scheduling Logistics **26**(5), 44–55 (1994)
308. van Beers, W.C.M., Kleijnen, J.P.C.: Kriging for interpolation in random simulation. J. Oper. Res. Soc. **54**, 255–262 (2003)
309. van Beers, W.C.M., Kleijnen, J.P.C.: Kriging in simulation: a survey. In: Proceedings of the 2004 Winter Simulation Conference, pp. 113–121 (2004)
310. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference architecture for holonic manufacturing systems: PROSA. Comput. Ind. **37**(3), 255–274 (1998)
311. Vepsalainen, A.P.J., Morton, T.E.: Priority rules for job shops with weighted tardiness costs. Manag. Sci. **33**(8), 1035–1047 (1987)
312. Vepsalainen, A.P.J., Morton, T.E.: Improving local priority rules with global lead-time estimates: a simulation study. J. Manuf. Oper. Manag. **1**, 102–118 (1988)
313. Vieira, G.E.: A practical view of the complexity in developing master production schedules: fundamentals, examples, and implementation. In: Herrmann, J.W. (ed.) Handbook of Production Scheduling, pp. 149–176. Springer, New York (2006)
314. Vieira, G.E., Herrmann, J.W., Lin, E.: Rescheduling manufacturing systems: a framework of strategies, policies, and methods. J. Scheduling **6**(1), 39–62 (2003)
315. Wall, M.: GAlib: A C++ library of genetic algorithm components (2010) http://lancet.mit.edu/ga/
316. Wang, F.K., Lin, J.T.: Performance evaluation of an automated material handling system for a wafer fab. Robot. Comput. Integrated Manuf. **20**(2), 91–100 (2004)
317. Weigert, G., Klemmt, A., Horn, S.: Design and validation of heuristic algorithms for simulation-based scheduling of a semiconductor backend facility. Int. J. Prod. Res. **47**(8), 2165–2183 (2009)
318. Wein, L.M.: Scheduling semiconductor wafer fabrication. IEEE Trans. Semicond. Manuf. **1**(3), 115–130 (1988)
319. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, Massachusetts (1999)
320. Weng, W.W., Leachman, R.C.: An improved methodology for real-time production decisions at batch-process work stations. IEEE Trans. Semicond. Manuf. **6**(3), 219–225 (1993)
321. Werner, S., Horn, S., Weigert, G., Jähnig, T.: Simulation based scheduling system in a semiconductor backend facility. In: Proceedings of the 2006 Winter Simulation Conference, pp. 1741–1748 (2006)
322. Wight, O.W.: Input/output control: a real handle on lead time. J. Prod. Inventory Manag. **11**(3), 9–31 (1970)
323. Witte, J.D.: Using static capacity modeling techniques in semiconductor manufacturing. In: Proceedings of the 1996 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, pp. 31–35 (1996)

324. Wooldridge, M.: An Introduction to Multiagent Systems, 2nd edn. Wiley, Chichester (2009)
325. Yang, F., Ankenman, B.E., Nelson, B.L.: Efficient generation of cycle time-throughput curves through simulation and metamodeling. Nav. Res. Logist. **54**(1), 78–93 (2007)
326. Yang, F., Liu, J., Nelson, B.L., Ankenman, B.E., Tongarlak, M.: Metamodelling for cycle time-throughput-product mix surfaces using progressive model fitting. Prod. Plann. Contr. **22**(1), 50–68 (2011)
327. Yeung, T.G., Mason, S.J.: Using real options analysis to value reoptimization options in the shifting bottleneck heuristic. Nav. Res. Logist. **53**(4), 285–297 (2006)
328. Yoon, H.J., Shen, W.: A multi-agent based decision making system for semiconductor wafer fabrication with hard temporal constraints. IEEE Trans. Semicond. Manuf. **21**(1), 83–91 (2008)
329. Yugma, C., Dauzère-Pérès, S., Derreumaux, A., Sibille, O.: A batch optimization software for diffusion area scheduling in semiconductor manufacturing. In: Proceedings of the 2008 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, pp. 327–332 (2008)
330. Yurtsever, T., Kutanoglu, E., Johns, J.: Heuristic based scheduling system for diffusion in semiconductor manufacturing. In: Proceedings of the 2009 Winter Simulation Conference, pp. 1677–1685 (2009)
331. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: the Gaia methodology. ACM Trans. Software Eng. Methodol. **12**(3), 317–370 (2003)
332. Zimmermann, J., Mönch, L., Otto, P.: On the parameterization of dispatching rules in manufacturing. In: Proceedings of the 49th International Scientific Colloquium, Technical University of Ilmenau, pp. 74–79 (2004)

Index

L. Mönch et al., *Production Planning and Control for Semiconductor Wafer Fabrication Facilities*, Operations Research/Computer Science Interfaces Series 52, DOI 10.1007/978-1-4614-4472-5,