Ordonnancement multi-objectif d'ateliers complexes de type job-shop : application à la fabrication de semiconducteurs

Karim Tamssaouet

N° d'ordre NNT : 2019LYSEM016

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée au sein de

l'École des Mines de Saint-Étienne

École Doctorale N° 488

Sciences, Ingénierie, Santé

Spécialité de doctorat : Génie Industriel

Soutenue publiquement le 17/07/2019, par

Karim Tamssaouet

Multiobjective Complex Job-Shop Scheduling: Application to Semiconductor Manufacturing

---

Devant le jury composé de :

Safia Kedad-Sidhoum, Professeur , CNAM, Paris	Présidente
Christelle Jussien-Guéret, Professeur, Université d'Angers	Rapporteuse
Farouk Yalaoui, Professeur, Université de Technologie de Troyes	Rapporteur
Margaux Nattaf, Maître de Conférence, INP Grenoble	Examinatrice
Reha Uzsoy, Professeur, North Carolina State University	Examinateur
Stéphane Dauzère-Pérès, Professeur, EMSE, Gardanne	Directeur de thèse
Claude Yugma, Chargé de Recherche, EMSE, Gardanne	Co-directeur
Jacques Pinaton, Ingénieur, STMicroelectronics, Rousset	Encadrant industriel
Emmanuel Troncet, Ingénieur, STMicroelectronics, Rousset	Encadrant industriel
Leon Vermarien, Ingénieur, STMicroelectronics, Rousset	Invité

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	MATHEMATIQUES APPLIQUEES	O. Roustant, Maître-assistant
MECANIQUE ET INGENIERIE	S. Drapier, professeur	INFORMATIQUE	O. Boissier, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	IMAGE, VISION, SIGNAL	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	X. Delorme, Maître assistant
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

ABSI	Nabil	CR	Génie industriel	CMP
AUGUSTO	Vincent	CR	Image, Vision, Signal	CIS
AVRIL	Stéphane	PR2	Mécanique et ingénierie	CIS
BADEL	Pierre	MA(MDC)	Mécanique et ingénierie	CIS
BALBO	Flavien	PR2	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR2	Sciences et génie de l'environnement	FAYOL
BEIGBEDER	Michel	MA(MDC)	Informatique	FAYOL
BLAYAC	Sylvain	MA(MDC)	Microélectronique	CMP
BOISSIER	Olivier	PR1	Informatique	FAYOL
BONNEFOY	Olivier	MA(MDC)	Génie des Procédés	SPIN
BORBELY	Andras	MR(DR2)	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR2	Génie Industriel	FAYOL
BRODHAG	Christian	DR	Sciences et génie de l'environnement	FAYOL
BRUCHON	Julien	MA(MDC)	Mécanique et ingénierie	SMS
BURLAT	Patrick	PR1	Génie Industriel	FAYOL
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR1	Génie Industriel	CMP
DEBAYLE	Johan	CR	Image Vision Signal	CIS
DELAFOSSE	David	PR0	Sciences et génie des matériaux	SMS
DELORME	Xavier	MA(MDC)	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR1	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
DOUCE	Sandrine	PR2	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR1	Mécanique et ingénierie	SMS
FAVERGEON	Loïc	CR	Génie des Procédés	SPIN
FEILLET	Dominique	PR1	Génie Industriel	CMP
FOREST	Valérie	MA(MDC)	Génie des Procédés	CIS
FOURNIER	Jacques	Ingénieur chercheur CEA	Microélectronique	CMP
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GARCIA	Daniel	MR(DR2)	Génie des Procédés	SPIN
GAVET	Yann	MA(MDC)	Image Vision Signal	CIS
GERINGER	Jean	MA(MDC)	Sciences et génie des matériaux	CIS
GOEURIOT	Dominique	DR	Sciences et génie des matériaux	SMS
GONDRAN	Natacha	MA(MDC)	Sciences et génie de l'environnement	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR1	Génie des Procédés	SPIN
GUY	Bernard	DR	Sciences de la Terre	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR1	Génie des Procédés	SPIN
KERMOUCHE	Guillaume	PR2	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	MR(DR2)	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	CR	Mécanique et ingénierie	FAYOL
MALLIARAS	Georges	PR1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et ingénierie	CIS
MOUTTE	Jacques	CR	Génie des Procédés	SPIN
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
NORTIER	Patrice	PR1		SPIN
OWENS	Rosin	MA(MDC)	Microélectronique	CMP
PERES	Véronique	MR	Génie des Procédés	SPIN
PICARD	Gauthier	MA(MDC)	Informatique	FAYOL
PIJOLAT	Christophe	PR0	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR1	Génie des Procédés	SPIN
PINOLI	Jean Charles	PR0	Image Vision Signal	CIS
POURCHEZ	Jérémy	MR	Génie des Procédés	CIS
ROBISSON	Bruno	Ingénieur de recherche	Microélectronique	CMP
ROUSSY	Agnès	MA(MDC)	Génie industriel	CMP
ROUSTANT	Olivier	MA(MDC)	Mathématiques appliquées	FAYOL
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
TRIA	Assia	Ingénieur de recherche	Microélectronique	CMP
VALDIVIESO	François	PR2	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR1	Génie industriel	CIS
YUGMA	Gallian	CR	Génie industriel	CMP

Mise à jour : 01/02/2016

Acknowledgements

This research project would not have been possible without the help and support of many people. Firstly, I would like to express my gratitude to my thesis advisor Stéphane Dauzère-Pérès for patiently guiding me throughout all these years. This work owes a great deal to his vast expertise and insightful comments. I will always remember his exceptional working capacity and contagious cheerfulness. My sincere thanks also go to Claude Yugma for his support, his availability, and his kindness. I would be remiss if I did not express my gratitude to Sebastian Knopp, who kindly offered his help and his support on several occasions.

I would like to thank Christelle Jussien-Guéret and Farouk Yalaoui for accepting the invitation to be part of my committee and reviewing the present manuscript. Warm words of thanks go to Safia Kedad-Sidhoum and Margaux Nattaf for their constructive comments and suggestions. I would also like to thank Reha Uzsoy for accepting the invitation, and I appreciate his extensive and extremely detailed examination of this work.

This industrial thesis would not have been possible without the contribution of several teams of STMicroelectronics Rousset. A special thank goes to Eric Tartière, Véronique Lemaire and Benjamin Coup who showed a continuous commitment for the project. Many thanks to Jacques Pinaton, who provided me an opportunity to join the Process Control team and benefit from his long experience in semiconductor manufacturing. I would like to thank Emmanuel Troncet and all the Industrial Engineering team for their support and all the invaluable information they provided to make the proposition of this thesis realistic. My gratitude also goes to all operators, engineers, and managers of the cleaning and diffusion area who gave their time and effort to shed light on the complexity of the industrial context.

I want to thank all current and former members of the SFL department for the stimulating discussions and all the fun we have had in the last years. A big and heartfelt thank you to Abdel, Xavier, Margaux, and Wei-Ting for their friendship and the relaxing talks. I want to thank all the friends who made this journey enjoyable and extremely rewarding. Thanks to Sophia, Wilson, Taki and Abdel for the joy and laughter they brought. Last but not least, I would like to say *tanemirt* to my parents and my brothers for supporting me spiritually throughout the writing of this thesis and my life in general.

Karim Tamssaouet
26/09/2019

Contents

List of Figures

List of Tables

General Introduction

This thesis deals with real-life complex scheduling problems, stemming in particular from semiconductor manufacturing systems where dispatching rules are still popular. Optimization algorithms are a promising alternative to dispatching rules, provided that the solved problem effectively addresses the rich set of constraints and criteria. Scheduling lots in the diffusion work area of the semiconductor manufacturing facility of STMicroelectronics in Rousset (France), requires that many constraints are considered within a flexible job-shop environment such as release dates, setup times, time lags and unavailability periods. All the machines are capable of processing several jobs in parallel, and a subset of them are complex, in the sense that the processing time of operations depends on the loading sequences. Different criteria have to be considered to optimize the various operational performance measures of the work area such as throughput, machine utilization and cycle time. Though a specific work area is studied, the investigated model captures a large number of practical features. Hence, this work can be applied in many other industrial contexts and contributes to narrow the gap between theory and practice that is acknowledged in the scheduling literature.

In Chapter 1, a broad overview of the industrial and scientific contexts for this work is given. After briefly describing the semiconductor manufacturing process and highlighting the various aspects that make it a complex industry, scheduling decisions are positioned within the diverse and interacting set of decisions that operation managers of a wafer fabrication facility have to take. This chapter motivates the different assumptions and choices made to model the considered scheduling problem and lists some of the potential advantages of optimization algorithms over dispatching rules. Finally, this chapter reviews some works related to the different features of the considered scheduling problems and solution approaches.

Chapter 2 presents a comprehensive description of the diffusion work area and all aspects that should be taken into account by the developed approach so that its practical implementation becomes possible. This chapter is the outcome of the immersive training within the work area and many discussions with operators, engineers, and managers. Each feature of the scheduling problem is described, and its inclusion in the problem definition is motivated. The detailed understanding of the industrial problem calls for new optimization criteria to be defined. Two new criteria are introduced to respectively increase the consistency between local scheduling decisions and the global production plan at the facility level and to better measure the throughput of the work area within a rolling horizon framework. Beyond being a synthesis of our understanding of the industrial problem, the content of this chapter is the foundation for the problem formulation in Chapter 4 and the design of a data management module permitting the extraction of the necessary data from the information systems of the company.

Before formulating in Chapter 4 the problem described in Chapter 2, Chapter 3 recalls the *batch-oblivious* approach proposed by Knopp (2016), which serves as a basis for our approach. The objective of this thesis is to extend this approach to deal with more constraints and more criteria and to improve its efficiency. After recalling the different components of the batch-oblivious approach, Chapter 3 suggests new algorithms that allow improving solutions during the start time computation with a low computational cost and which lead to solutions with a more relevant structure from an industrial perspective.

Chapter 4 formalizes all the constraints and criteria described in Chapter 2. The batch-oblivious approach is extended by integrating new constraints: Minimum time lags, minimum batch size, availability constraints, and production targets. A significant part of this chapter is devoted to the extension of the batch-oblivious approach in order to model the internal structure of complex batching machines so that the conjunctive graph can naturally capture their complex behavior. Different considerations that make the proposed modeling suitable for a stable industrial application are given.

In the two preceding chapters, only a single criterion is considered while up to six criteria must be considered in the industrial scheduling problem. Chapter 5 deals with the multicriteria aspect of the investigated problem and proposes different approaches. In the two first approaches, the decision maker is given a flexible modeling of his preferences depending on whether a trade-off is permitted between any pair of the considered criteria. The two approaches use these preferences differently during the search process and store the set of nondominated solutions in a passive archive. These two approaches are compared to a third approach from the literature that uses the dominance status between the current solution and the set of nondominated solutions stored in an active archive. The comparison is performed based on the given preferences and known quality indicators and demonstrates that each approach can be preferable depending on the context. This chapter ends with numerical experiments that attest the significant improvement that can be brought by the proposed approach.

Chapter 1

Industrial and Scientific Context

This chapter gives a general overview of the industrial and scientific context of this thesis. Section 1.1 starts the chapter with a brief description of the semiconductor manufacturing process and highlights the different aspects that make it a complex industry. Then, Section 1.2 provides an overview of the operations management of a wafer fabrication facility and positions the scheduling decisions within the rich and interacting set of decisions. Scheduling, as one of the approaches for production control, is compared to a widely used dispatching approach in Section 1.3. This section also motivates the different assumptions and choices that are made for modeling the studied scheduling problem. The scientific context is discussed in Section 1.4, where the different features of the considered scheduling problems and the solution approaches are defined and the related works reviewed.

1.1 An Overview of Semiconductor Manufacturing: Process and Complexity

The process in semiconductor manufacturing is used to create integrated circuits (also known as chips, or die) that are present in everyday electrical and electronic devices. Integrated circuits, consisting of thousands of components, are gradually created on a wafer made of pure semiconducting material, Silicon most of the time. A wafer is a thin disc that is sliced from a single crystal ingot. Depending on the size of the chips and of the wafer, hundreds or thousands of chips can be obtained from a single wafer. The process by which integrated circuits are created from a raw wafer includes four phases: Wafer fabrication, wafer probe, assembly or packaging, and final testing. The first two phases are grouped into the "Front-end" process, and the last two in the "Back-end" process. Wafer fabrication, the most technologically complex and capital intensive phase, involves processing wafers to build up the necessary layers and patterns of conductors, semiconductors, and insulators. Wafer probe involves electrically testing the individual circuits on each wafer to verify their functioning and to discard defective ones. The circuits that pass this test are sent to assembly, where they are placed in plastic or ceramic packages that protect them from the environment. The final testing ensures that customers receive a defect-free product. A detailed description of semiconductor manufacturing processes is given in Ovacik and Uzsoy (2012) and Mönch et al. (2012).

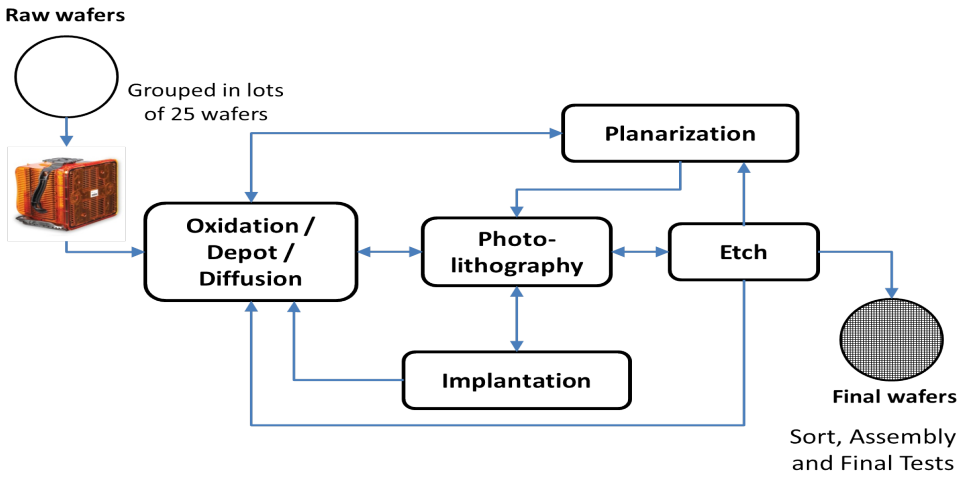


Figure 1.1 – *Processing steps within wafer fabrication (Mönch et al. (2011))*

Each phase of the semiconductor manufacturing process is composed of several stages, each one consisting of a long sequence of operations performed by different machines. As this thesis is conducted within the context of wafer fabrication, we shall focus only on this phase, which encompasses most of the complexity encountered in the three other phases. Recall that the wafer fabrication process builds up layer by layer hundreds or thousands of integrated circuits on each wafer. Most of the operations in this phase have to be performed in a clean-room environment to prevent particulate contamination of wafers. The facility in which wafer fabrication takes place is referred to as a *wafer fab*. Wafer fabs are classified by the diameter of the wafers they are tooled to produce. The diameter has gradually increased to improve throughput and reduce cost. Most of the wafer fabs today process 200mm or 300mm wafers, with the next generation projected to be 450mm. Wafers are grouped in *lots*, each containing up to 25 wafers in a specialized contained, either Front Opening Unified Pod (FOUP) in 300mm fabs or Standard Mechanical InterFace (SMIF) pods for wafers no larger than 200mm. Lots are the elementary entities that move through the fab. Within the context of scheduling, lots are denoted as *jobs* throughout this thesis. From a group of raw wafers to thousands of integrated circuits, a lot has to go through hundreds of processing operations, more than 800 for some technologies. The different processing operations needed to produce a single layer are performed in specialized work areas that consist of machines with similar capabilities. To build all layers, each lot visits the different work areas many times (more than 40 times for the most advances products), leading to *re-entrant* flow. The typical work areas in a wafer fab are shown in Figure 1.1 where the arrows indicate the flow of lots. These work areas are briefly described below:

Cleaning The objective of this operation is the removal of contaminants (particles as well as metallic and organic) from the surface of the wafer. Cleaning is called “wet” when chemical solutions are used and “dry” when gases are used instead.

Diffusion/Oxidation/Deposition A layer of material is grown or deposited on the surface of

a cleaned wafer. *Diffusion* is a common name for the high-temperature process during which dopant atoms are introduced into the semiconductor by diffusion. *Oxidation* aims at growing a dioxide layer on a wafer surface at elevated temperature. *Deposition* deposits dielectric or metal layers. Usually, only one of these three operations is performed per layer.

Photolithography After depositing a photoresistant liquid on the wafer, it is exposed to ultraviolet light through a *mask*, also called *reticle*, which contains the pattern of the circuit. The exposed wafer is then developed by removing polymerized sections of photoresist from the wafer. These operations are the most critical and complex ones, as well as the ones requiring the highest precision. Wafers may pass through this operation more than 40 times. Since both machines and tooling are costly, this area is often a bottleneck.

Etching The *etching* process removes unneeded material from the wafer surface. Patterned etching removes a pattern that was brought onto the wafer during photolithography. Unpatterned etching reduces the thickness and involves the entire area of the wafer. There are two types of etching: Dry etching exposes the wafer surface to a plasma, while wet etching removes material using chemical solutions.

Ion Implantation This process aims at changing the electrical properties of the exposed portion of the layer by introducing dopants in the crystal structure of the semiconductor.

Planarization This is the process of flatting the surface of the wafer, performed each time before adding a new layer.

After these different processing operations, measurement and inspection operations usually have to be conducted. These operations aim at controlling the product quality and the machine performance. Decisions are made according to the inspection results. Good wafers that pass inspection are sent to downstream operations while those that fail inspection are either sent to upstream operations for rework or scrapped. Even when a wafer succeeds in going through all its processing operations and passes all inspections, only a fraction of the chips on the wafer are usually functional. The ratio of the number of wafers which reach the probe test to the original number of wafers at the process starting point is called *process yield*. The ratio of the number of good chips to the total number of chips on a wafer is called *probe yield*.

As a result of the reentrant flows, lots of the same product that are at different stages of completion compete over the production resources. Moreover, different products are produced at the same time in a wafer fab. The level of the diversity of products, called *product mix*, influences the complexity of the operations management. Depending on the product mix, wafer fabs can be classified into *low-mix* and *high-mix* fabs. In low-mix fabs, high quantities of a few product types are manufactured. High-mix fabs usually produce Application Specific Integrated Circuits (ASICs) which are customized chips for specialized

applications. In this case, many different product types are manufactured with potentially small quantities for each product type. In addition to regular production lots, engineering lots need to be included with the production process. These lots are necessary for the development of future products, a necessary condition to survive in a sector with a high velocity of innovation.

While it is possible to dedicate machines to products in low-mix fabs, products must share the same machines in high-mix fabs. Several hundred machines can be found in a single wafer fab. Individual machines cost between 100 thousand and 40 million U.S. dollars. The building blocks of the machinery of a wafer fab are *work centers*, also called *tool groups*. Some work centers consist of a single machine and some of multiple parallel machines that provide similar processing capabilities. Machines in a fab have different capabilities and are subject to different constraints. There are work centers where each machine can process only one wafer at a time (e.g., photolithography machines); work centers where machines can process a subset of wafers of the same lot at a time (e.g., ion implantation) ; or work centers where machines can process all the wafers of a set of lots (a *batch*) at the same time (e.g., diffusion). To maximize quality performance, integrated machines, referred to as *cluster tools*, have been increasingly used in wafer fabs. These complex machines combine several single-wafer processing modules with handling robots in a closed environment. They are capable of processing a certain number of lots in parallel at the cost of a very complex behavior.

Processing durations can vary immensely between production operations, from a few minutes to over 12 hours. Many of the long operations involve *batch* processes, i.e., the joint processing of several jobs on a machine. Due to the complex behavior of cluster tools, the processing times of operations depend on the loading sequence. Though having the same capabilities, machines in the same work center may require different processing times for the same operation. Also, a machine can process a particular product while it is forbidden to perform the same operation on a lot of another product. Some work centers require setup times. For example, in the ion implantation work area, dopants have to be changed frequently, and machines have to be cleaned. The effort to do this change depends on the dopant required by the previous lot.

Additional *auxiliary resources*, are sometimes necessary to process lots on machines. As a typical example, photolithography machines need reticles to transfer the pattern of the integrated circuit on the surface of the wafer. These auxiliary resources are quite expensive, and are usually unique for each layer of each product. Then, high-mix fabs require a high number of reticles while only a minimal number of each type are purchased. Managing this work area is complex because the correct reticle must be available on the tool for the duration of the lot's processing. Recall that, in addition to production operations, lots must go through inspection and measurement operations. These operations are performed on dedicated machines with limited capacity. Hence, only a subset of lots is selected (also called sampled) for inspection. These selection decisions are crucial in order to avoid the late detection of machine failures. For example, a machine producing defective wafers will continue to do so until the wafers processed on this machine are measured. After completing an operation on a machine, a lot

must be moved to the next operation. These handling and transportation operations require resources. In modern 300mm fabs, these operations are performed by Automated Material Handling Systems (*AMHS*). When they are not upgraded by integrating an AMHS, 200mm fabs operate in a semi-automated mode, meaning that operators perform lot transportation and storage while lot loading and unloading and machine settings are automated. Whether human or automated systems, handling resources impose capacity constraints on production rates. When a lot is neither being processed nor being transported, it must be stored. Storage spaces in a fab, called *buffers*, store waiting lots. Buffers can be dedicated to work centers, work areas or a group of work areas. These buffers can be often assumed unlimited for fabs that are mainly run manually. However, in 300mm wafer fabs, specific finite-capacity buffers are used to allow automated transportation of jobs by the AMHS.

The complex, high technology nature of semiconductor manufacturing processes and equipment leads to a significant uncertainty in the production process. Fabs must face unpredictable events that cause disruptions of the production flow by responding as quickly as possible. Production resources are subject to random failures and need random repair times. At various points in the fab process, entire wafers are discarded, either because the wafers failed inspection or because they are broken. If defective wafers are not scrapped, they must be reworked by redoing one or more operations. Sometimes, certain processes will be stopped until the results of some experiments are obtained, or engineering decisions are made. Impacted lots in such case, called *hold* lots, can wait for days before engineering decisions are made. The yields, whether process or probe, can be unpredictable, especially for new processes (Bai and Gershwin (1989)).

The different aspects described above show the complexity of wafer fabrication, a complexity that is rarely found elsewhere. When considering the crucial factors of competitiveness in semiconductor manufacturing and its high cost, the effective management of production operations is critical. As wafer fabrication concentrates most of the capital cost and most of the time of the semiconductor manufacturing process, the effective management of these complex operations is decisive for the competitiveness of a company. First, due to the high capital costs of wafer fabs, it is crucial to maximize the throughput of the fabs and the machine utilization. In the fierce competition of the global market place, reducing production costs is an essential lever for the competitiveness of a company. This reduction can be obtained by effective management of resources, inventories, yield, and labor. In a situation where prices, as well as the state of technology, have settled at a certain level, the capability of improving both quality and delivery time performance, and reducing both the mean and the variance of cycle times has become one of the most decisive factors for success. *Cycle* time is defined in this context as the time between a lot being started in the fab and the completion of the lot.

1.2 Operations Management of a Wafer Fab

Due to the fierce competition and the complexity of wafer fabrication, the operations management faces different complex problems. To better position the encountered problems, it is useful to recall that management decisions are often classified into three categories: *Strategic*, *tactical* and *operational* decisions (Anthony (1965)). These decisions vary on their impact in the company, their scope, and on the horizon they are applied. In the following, different problems are described for the different decision levels, primarily focusing on those concerning a single wafer fab.

Strategic decisions: For a wafer fab, these decisions involve the definition of the product mix to manufacture and the necessary resources to acquire, so that the assigned objectives can be reached. Before the existence of a fab, these decisions include its location, the selection of providers, the design of the plant layout and the structure of its distribution network. These decisions establish the manufacturing capacity, by choosing the type and the number of resources, either human, machines or information systems. Strategic decisions regarding existing fabs impact the manufacturing capacity and product mix. Deciding whether to manufacture a new product or acquire new resources are strategic decisions. An example of a strategic decision is to upgrade a 200mm fab by integrating an AMHS. These decisions are strategic in the sense that they have an enormous impact on the bottom line of the companies, and that the economic and physical resources involved are such that these decisions are taken on a horizon usually measured in years.

Tactical decisions: After designing the manufacturing system and its capacity, there are many decisions related to how to make the best usage of this capacity. Tactical decisions are part of these decisions to be taken for a mid-term horizon, usually several weeks to one year. Whatever the considered time horizon, tactical decisions are taken within the logical framework drawn by strategic decisions and do not seriously modify the production capacities. Typical tactical decisions in manufacturing are *production planning* decisions. Given forecasted demands and a finite manufacturing capacity, the quantities to be achieved within a certain time bucket, a week or a month, should be decided in such a way that certain performance measures of a fab are optimized. In semiconductor manufacturing, these decisions are refined to determine the set of lots to be launched in a wafer fab at the beginning of shorter time buckets, one or more weeks in duration. These decisions are part of what is referred to as *order release*.

To cope with the high diversity of products, especially in high-mix fabs, and the short product life cycles, machines are re-configurable to ensure the flexibility of the system. Each operation requires a *recipe* which can be defined as a pre-planned and reusable set of instructions and settings that specify how an operation is to be performed by a machine. To make a machine capable of performing a the recipe of an operation, a *qualification* in semiconductor manufacturing is needed. A qualification consists in modifying the hardware and software part of a machine and testing that it is functional.

Qualifying as much as possible operations on a machine makes the machine flexible and allows the fab to cope with the diversity of products. Some qualifications can be relatively quick to perform (e.g., equivalent to a production run), but others might be very time-consuming (e.g., equivalent to a production cycle time) and hence costly. Thus, deciding which operations to qualify on which machines is critical and impacts the flexibility and the capacity of the production system. These decisions are part of *qualification management* (see e.g. Johnzén et al. (2011) and Rowshannahad et al. (2015)). Also, being complex, machines in semiconductor manufacturing are unreliable and are subject to frequent failures and unforeseen quality problems. *Preventive maintenance* operations are used to reduce the number and the duration of these failures. However, at the same time, these operations reduce the machine capacity by making the machine unavailable for production operations during maintenance operations. By trading off between planned unproductive downtimes due to maintenance operations and the risk of unscheduled downtimes due to machine failures, *preventive maintenance planning* helps to avoid the unexpected failures without stopping production too often.

Operational decisions: These decisions have short term effects, ranging from minutes to days. Because of the long horizon, strategic and tactical decisions use an aggregate view of the production capacity and the demand. Operational decisions specify the most detailed and precise instructions for executing tasks. Detailed data are needed to capture the status of the production resources and the evolution of the demand. A substantial number of decisions can be classified as operational ones. Each time a machine becomes available, the question of which product is next to be processed should be answered. The answers to these decisions directly impact the utilization of machines and the throughput of the fab, the inventory level and the cycle time of the product. As there are machines that operate better than others from a quality perspective, these decisions can also affect quality measures such as the yield. Production control solutions, such as *dispatching* and *scheduling*, are used to guide or to take these decisions.

The problem of allocation involves other resources, not only production machines. The AMHS, comprising a limited number of vehicles and rails, requires real-time management. Dynamic requests are sent to the system to transport lots between the different machines. These requests should be considered as soon as possible by allocating vehicles. The route of the vehicle in the rail network must also be decided. When the machine that must process the transported lot is not available, the question of where to store the lot meanwhile should be answered. Other operational decisions are related to inspection and measurement machines. Due mainly to their limited capacity, not all lots can be measured. The right set of lots to measure should be selected to ensure that the risk of quality problems on all the production machines is reduced as much as possible. In addition to the dynamic environment, events that disrupt production randomly occur. The different decisions must be quickly reviewed to solve the problems. When a failure or a quality problem occurs on a machine, a quick solution must be found. It may be necessary to reallocate to other machines lots that are already allocated to a

failed machine. Maintenance resources, such as technicians and necessary tools, must be assigned to repair the concerned machine. These resources must be shared by *curative maintenance* and planned preventive maintenance operations. Unexpected events, such as breakdowns, may also concern vehicles of the AMHS. Such events often lead to the blocking of a section of a rail. If the situation is not well handled, the problem may quickly propagate and negatively impacts the production. Reallocation of the vehicles and their rerouting must be decided to recover from such situations.

The decision hierarchy described above does not only have a pedagogical purpose. A large number of decisions differ on their impact in the company, their scope, their timing, their time horizon and the level of required data aggregation. It is not realistic to seek to solve all these decision problems simultaneously. Hence, operations management has been traditionally decomposed by adopting the described decision hierarchy, in which the various decision problems are successively solved. By doing this, the degree to which a decision at a given level, scheduling, for instance, can affect the company's performance, is constrained by decisions taken at higher levels of the hierarchy, such as production planning and qualification management. This does not reduce the importance of decisions at lower levels. It is true that when poor strategic decisions are taken, taking the best tactical decisions is not likely to be of much help. It is also true that, without good decisions at the tactical level, taking the best decisions at the strategic level will not lead to the expected results. The same interaction exists between tactical and operational decisions. If it is not possible to solve all the decision problems at all the decision levels, the performances of a wafer fab can only be better when the different decisions of the same level are addressed in an integrated way. However, in the context of the wafer fabrication, this is also difficult. Many challenges, from industrial and scientific perspectives, still lie ahead for each of the individual problems. Then, when solving a decision problem, to make sure that the interaction is not ignored, other decisions are considered as constraints, and their complexity is simplified.

Within this context, the objective of this thesis is to develop a production control approach. The approach should optimize the different performance measures of a wafer fab through a better allocation of the resources over time. As mentioned earlier, this can be performed through dispatching or scheduling. The proposed approach is a scheduling solution that is meant to replace the widely used dispatching rules. A comparison between these two approaches is detailed in the next section. Again, it is not realistic to try solving the problem of allocating all the machines of a wafer fab at the same time. At the operational level, the model of the production resources and the production flows must be detailed as much as possible. However, an integrated, global model of a wafer fab with this level of details becomes unwieldy, and the allocation decisions of production machines become computationally intractable.

A practical and realistic approach is to decompose this complex problem into smaller and more tractable subproblems. A quite natural decomposition approach consists in separately managing the different areas of the fab. As highlighted before, machines of a specific work area share the same capabilities. For example, while it is important to minimize setup times in the photolithography or ion implantation work areas, it is important to utilize as

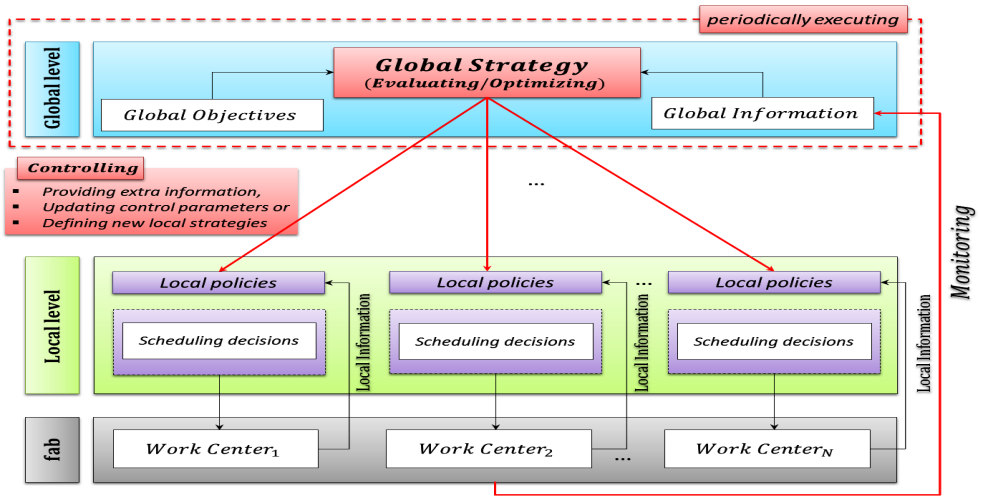


Figure 1.2 – *Overall structure of the scheduling system of a wafer fab (Sadeghi (2017))*

much as possible the batching capacities of machines in the diffusion work area. Constraints which come from process considerations are different from one work area to another. This decomposition supports the development of models for individual work areas and probably gaining computational advantages by exploiting the special structure of the encountered decision problems. However, solving tractable problems and gaining computational advantages are not sufficient arguments for a company to adopt an optimization approach. Even if the problem at a wafer fab is decomposed, only the global performances of the wafer fab are important. When optimizing the production control of a work area, its interactions with other work areas and its impact on the overall performances of the wafer fab must be taken into consideration. Figure 1.2 shows the framework that is proposed by Sadeghi (2017). It is proposed to guide the local scheduling solutions at work area level by providing global information and targets. Targets are determined by a control approach that models the whole fab and optimizes its overall performances. In this way, it is made sure that optimizing an individual area is not done at the expense of the overall performance of the wafer fab.

In this thesis, the proposed scheduling approach is applied to the integrated control problem in the cleaning and diffusion areas. The motivations for such integration are given in Section 2.1. The description of the studied work area with the different considered aspects is detailed in Chapter 2. Due to the large number of constraints and criteria that are handled, the proposed approach can be applied to different other areas such as the ion implantation and the photolithography areas. In the remainder of this work, the industrial context is used to illustrate the different features of the very general scheduling problem under study. In Section 1.3, the improvements that can be brought by an optimized scheduling solution, compared to the widely used dispatching solutions, are highlighted. Some modeling choices are motivated, and some challenges are recalled. Section 1.4 provides a brief literature review regarding the different considered features of the scheduling problems and the considered solution approaches.

1.3 Production Control in Wafer Fabrication

In today's wafer fabs, scheduling solutions are still not widely applied (Mönch et al. (2012)). Dispatching rules are often used for shop floor control. A dispatching rule selects the next lot to be processed among the lots that are waiting in front of a machine group. This selection is based on the priorities that are assigned to lots based on different lot and machine attributes. For some sophisticated rules, upstream and downstream information can also be considered. The persistence of their use in the industrial environment can be explained by the relative easiness of their implementation and the "explainability" of the proposed decisions, i.e., the easiness of explaining and understanding their decision logic. More than the easiness of understanding, dispatching rules can be enriched by shop floor managers based on their insights and experience.

Besides these strengths, weaknesses of dispatching rules can be highlighted. First, they are generally myopic in space and time. They are myopic in space as they are designed to select a lot to process among the queue of lots waiting in front of a particular machine group. Even if some upstream and downstream information is taken into consideration, machine groups in the same shop floor are managed independently. While dispatching rules are adapted to the dynamic environment of a fab, they are myopic in time as decisions are made without looking far in the future. A second reason lies in the fact that it may be challenging to adapt dispatching rules to different situations in the shop floor. In a fast-changing industry like the semiconductor industry, changing situations on the shop floor that result from changes in the product mix is rather frequent. Finally, a third reason can be mentioned as a consequence of the second one. Facing new situations, shop floor managers tend to increase the complexity of the associated dispatching rules to handle new situations or exceptions. By doing this, dispatching rules may quickly lose their advantage of being understandable.

These weaknesses explain why sequences of lots proposed by dispatching rules are considered suggestive and not prescriptive. Operators and shop floor managers usually take the final decisions. When the propositions are judged relevant, the associated lots or batches are loaded into machines in the given sequence. The myopic character of dispatching rules in space are corrected in two ways in practice. A unique shop floor manager may communicate to operators a modified solution to make sure that the overall performance of the whole area is primarily optimized or to ensure the performance of a particular machine group. Operators and managers may also communicate with upstream areas to ask for lots that allow their area performance to be optimized. The propositions of dispatching rules may also be modified to answer the request of a downstream area. These practices also allow the myopic character of dispatching rules to be dynamically corrected.

Instead of using dispatching systems, optimized scheduling solutions are a promising alternative for shop floor control (Mönch et al. (2011)). The considerable amount of data that such systems require is no longer an obstacle thanks to the high degree of automation and the high storage capacity of databases allowing real-time data collection. Also, the increasing computation power of modern computers makes it possible to propose high-quality schedules in a short computational time, which opens up the opportunity of using the op-

timized scheduling solutions in a dynamic environment. Before detailing how scheduling systems can overcome the weaknesses of dispatching systems, let us highlight in Figure 1.3 the flexibility that scheduling systems bring, giving the different uses that can be made of control systems. The horizontal axis represents a simplified view of the degree of freedom of a human scheduler to deviate from the decisions proposed by the system. The vertical axis represents the decisions that are expected from these solutions, from only batching decisions to complete scheduling decisions. Note that this graph can be adapted to represent other decision hierarchies for other areas than the diffusion area and that each level of decisions on the vertical axis induces all the previous levels. In the studied industrial context, the current dispatching system is used to propose batching and sequencing decisions of the batches on machine groups, and it is not mandatory to follow these propositions. To obtain more refined decisions, the dispatching system must be reviewed and improved. It is difficult to consider the suggestions of such a system as firm decisions to be followed by operators due to the weaknesses described above. Regarding optimized scheduling systems, using a "simple" post-processing, the proposed schedules can be modified to be adapted to the required decisions types. Also, by overcoming the weaknesses of dispatching systems, scheduling systems can prescribe solutions to implement.

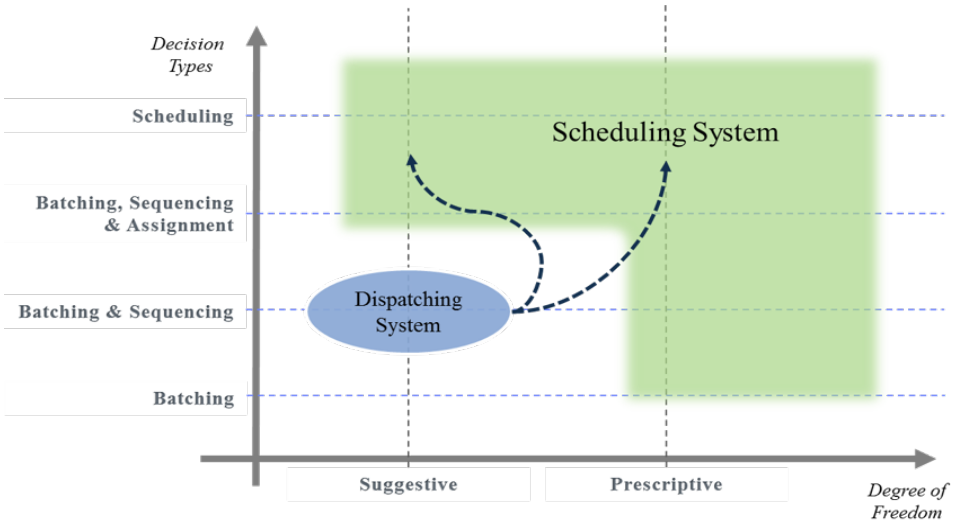


Figure 1.3 – *Use spectrum of scheduling and dispatching systems*

While dispatching means assigning the next job to be processed from a set of waiting jobs, scheduling is the process of allocating scarce resources over time. The conventional approach for scheduling is to solve an optimization model that encompasses all the relevant constraints with the specific objectives to optimize. Since scheduling decisions are time-related, a certain time horizon is inherent to any scheduling process. This ability to project into the future makes optimized scheduling solutions preferable to dispatching systems that are myopic in time. Being objective-oriented instead of rule-oriented, scheduling can also support the scheduling of a whole area, as long as all the relevant constraints are considered, instead of being myopic in space and restricting the decisions to a particular

machine group. Moreover, the objective-oriented paradigm of scheduling optimization algorithms makes them more robust to changing situations in the shop floor. Because of the NP-hardness and very large sizes of most scheduling problems in wafer fabs, heuristics are typically used to solve them. Even when there are radical changes in the shop floor leading to new constraints or new objectives, the modularity of scheduling approaches helps to better deal with these changes than dispatching rules that need to be redesigned.

Regarding the input of scheduling systems, two major categories can be identified: Deterministic and stochastic (Mönch et al. (2012)). In the case where all the characteristics of the decision problem (processing time of each operation, release date, due date, and so forth) are well known and single-valued, the scheduling problem is of deterministic type. In contrast, it is considered of stochastic type if at least one of these characteristics is not known deterministically but only with a probability distribution. While stochastic scheduling problems are academically interesting, they are not often practical (Mönch et al. (2011), Framinan et al. (2014a)). This thesis focuses then on developing algorithms for deterministic scheduling. Another assumption that is made is that all the relevant data of the decision problem are known in advance, i.e., at the point in time where the planning procedure starts. This is what is qualified in the literature as *static* scheduling, in opposition to *dynamic* scheduling where some parameters are unknown in advance, and no a priori knowledge about them is assumed (Blazewicz et al. (2007)).

These assumptions can be questionable in a dynamic environment where lots arrive dynamically, machines are subject to unpredictable breakdowns and rework occurs randomly. Even in these conditions, several arguments can be found in favor of a deterministic and static formulation of the shop floor scheduling problem. Ovacik and Uzsoy (2012) provide several arguments in this direction, and the ones that motivate the choice of a deterministic formulation over a stochastic one are summarized in the following. First, if the uncertainties on the shop floor are extreme, management priorities should focus first on locating and eliminating their sources, without what no scheduling system will work well. Second, the advent of computerized shop floor information systems considerably reduces uncertainties, as an important source of these uncertainties lies in the lack of accurate information on what is taking place in the fab. Finally, adopting a stochastic formulation is not necessarily realistic, and when this modeling is the most adapted one, the study of a deterministic formulation yields essential insights into the solution of the stochastic formulation.

If the arguments given above motivate the choice of the deterministic formulation of the studied scheduling problem, they do not imply that the dynamic aspect of the shop floor is ignored. The scheduling solution needs to be complemented by some mechanism to make sure that unforeseen disruptions on the shop floor or the arrival of new lots are taken into consideration in the proposed schedules. The scheduling algorithm to be proposed in this thesis is used to develop off-line schedules that specify decisions for a certain horizon in the future. These schedules are referred to as *predictive schedules* since they represent the prediction of what will occur on the shop floor under the ideal circumstances of everything going according to the plan. To cope with the dynamic aspect of the shop floor, a schedule is generated for a new horizon periodically. This process is usually referred to as a Rolling

Horizon Procedure. Within this framework, when a scheduling decision is to be made at any point in time, we solve a subproblem consisting of the jobs currently on hand and a subset of the jobs that will arrive shortly. Arrival times are assumed to be known a priori. In addition to this rolling horizon procedure, a strategy that defines how unforeseen events are considered must be selected, so that schedules reflect more accurately the current state of the shop floor. In the literature, a periodic strategy is defined as a strategy where rescheduling occurs regularly with a constant time interval (the rescheduling period), ignoring any event that occurs between consecutive rescheduling times. For example, in our case, a schedule could be generated every 15 minutes on a horizon of 12 hours or more. Under an event-driven strategy, the rescheduling process is also triggered whenever a predefined event occurs.

In real applications, modeling a scheduling problem and developing optimization algorithms to solve is only part of the story. The optimization engine has to be embedded in a system that enables the decision-maker to actually use it. The system has to be integrated into the information system of the enterprise, which can be a formidable task (Pinedo (2016)). In addition to the development or adaptation of the scheduling engine, it is required to design and implement a database management module. This module makes factory's databases and knowledge-bases suitable for input to a scheduling engine. This requires significant efforts. Any additional complexity in these tasks should be counterbalanced by a significant gain in other aspects. Any decision to be taken regarding the development of the scheduling system, should not only consider the efficiency of the scheduling algorithms. The impacts of such decisions on the development or the maintenance of the data management module, for instance, should be taken into consideration.

In this section, some weaknesses of the widespread used dispatching systems are first highlighted. After mentioning some technical conditions that make scheduling systems applicable in industry, the gains that can be brought by using scheduling systems are briefly recalled. The choice for a static and deterministic formulation of the studied scheduling problem is motivated, and a general framework that allows solving deterministic problems within a dynamic and stochastic environment is briefly described. In the remainder of this chapter, the different features of the considered scheduling problems and the solution approaches are defined and the related works reviewed.

1.4 Literature Review

A tremendous amount of research on scheduling was conducted in the last decades. General introductions on scheduling are provided in the textbooks of Blazewicz et al. (2007), Brucker (2007), Framinan et al. (2014b) and Pinedo (2016). To solve scheduling problems, diverse solution approaches are designed in the literature. In Section 1.4.1, a brief overview of these approaches is given. Most of the reviewed works in this thesis use approximation approaches as this thesis proposes a heuristic approach. In Section 1.4.2, a sample of works dealing with one or more features of the studied scheduling problem are reviewed. The classical job-shop scheduling problem, as a basis of the tackled problem, is first reviewed in Section 1.4.2.1. In

Section 1.4.2.2, we briefly describe the different constraints that are considered in this thesis and review some works that consider these constraints within a job-shop environment. The most used criteria in the scheduling literature are then reviewed in Section 1.4.2.3. Some of the specific criteria that are used within the context of semiconductor manufacturing are also recalled.

1.4.1 Solution Approaches

Known and successful algorithms for scheduling problems cover a wide range of algorithms that can be divided into two classes: *Exact* algorithms and *approximation* algorithms. Exact algorithms produce optimal solutions, but their running time cannot be bounded from above by a polynomial in the size of an instance for NP-hard problems if $\mathcal{P} \neq \mathcal{NP}$. In the following, the most commonly used exact algorithms are briefly described. *Branch-and-bound* procedures are search methods by implicit enumeration of the set of candidate solutions, that is thought of as forming a rooted tree with the full set at the root. As its name implies, this method consists of two fundamental procedures: Branching and bounding. Branching is the procedure of partitioning a large problem into two or more sub-problems usually mutually exclusive. Bounding calculates a lower bound on the optimal solution value for each subproblem generated in the branching process. *Dynamic programming* relies on the idea of transforming the resolution of one problem into the resolution of subproblems, related by a recurrence relationship on the value of the objective function. Stored solutions for subproblems are combined in order to obtain solutions of larger subproblems until the original problem is solved. *Constraint Programming* is based on the feasibility and the propagation of constraints. Besides these generic methods, some particular algorithms exploit specific properties of scheduling problems in order to construct a solution which is guaranteed to be optimal. Johnson's and Moore's algorithms (Johnson (1954), Moore (1968)) are examples of such algorithms.

Approximation algorithms (heuristics) are an interesting alternative to exact algorithms for solving NP-hard problems, as heuristics often provide reasonable trade-offs between solution quality and computational effort. They can provide performance guarantee if it is possible to quantify the gap between the best-provided solution and the optimal solution, which is usually not the case. Even if no classification of approximation algorithms can be unambiguous, it is useful to consider one. Here, approximation algorithms are differentiated as *Construction* heuristics and *improvement* heuristics. Construction heuristics are specifically tailored for a particular problem for which they iteratively build a solution. The most commonly used construction heuristics for scheduling problems are list algorithms where the list of operations is sorted according to a decision strategy called a dispatching rule such as SPT (Shortest Processing Time), EDD (Earliest Due Date), FIFO (First In First Out). Most construction heuristics are greedy, meaning that out of the remaining operations/tasks to be included to a so far generated partial schedule, the next operation is chosen to be the one that is contributing best to the objective function under consideration at the time the decision is made. These heuristics may construct solutions that are optimal for some special scheduling

problems. In general, the generated schedules are either active or non-delay (Pinedo (2016)).

Improvement heuristics (including metaheuristics) are high level algorithms for exploring search spaces with relatively few modifications to make them adapted to a specific problem. According to Blum and Roli (2003), metaheuristics can be differentiated between *trajectory* methods and *population* methods. Among trajectory methods, we can mention Simulated Annealing (SA, Kirkpatrick (1984)) , Tabu Search (TS, Glover (1989)), Greedy Randomized Adaptive Search Procedure (GRASP, Feo and Resende (1995)), Variable Neighborhood Search (VNS, Hansen and Mladenović (1997)), Guided Local Search (GLS, Voudouris and Tsang (1999)) and Iterated Local Search (ILS, Lourenço et al. (2003)). Among population metaheuristics, we can mention Genetic Algorithms (GA, Goldberg and Holland (1988)) and Ant Colony Optimization (ACO, Dorigo and Di Caro (1999)).

1.4.2 Complex Job Shop Scheduling

Scheduling deals with the allocation of resources to tasks over given time and its goal is to optimize one or more objectives (Pinedo (2016)). Significant research on scheduling was conducted in the last decades. General introductions on scheduling are provided in the textbooks of Blazewicz et al. (2007), Brucker (2007), Framinan et al. (2014b) and Pinedo (2016). To classify scheduling problems, they provide updated versions of the classification scheme introduced in Graham et al. (1977). The resources and tasks in an organization can take many different forms, depending on the studied sector of activity. In the following, the brief literature review mainly addresses different aspects of manufacturing production scheduling. Most of the reviewed works use approximation approaches as this thesis proposes a heuristic approach. Section 1.4.2.1 discusses the classical job-shop scheduling problem, considered as a basic problem of the studied problem in this thesis. Section 1.4.2.2 briefly defines the different constraints that are found in the studied context and reviews works that integrate them. Section 1.4.2.3 give an overview of the different criteria that are considered in the scheduling literature before listing those that are particularly studied within the context of semiconductor manufacturing.

1.4.2.1 Classical Job-Shop Scheduling

In a job-shop scheduling problem, a set of jobs have to be processed on a set of machines, and each job goes through several sequential operations (a routing) before being completed. A machine can only perform one operation at a time and preemption is not allowed. Each operation can only be performed on one specified machine. This scheduling problem is qualified as classical as the earliest formulations are proposed in the fifties (Bowman (1959)). It is a hard scheduling problem as it was proven to be NP-hard in Garey et al. (1976). An overview of solution methods, exact and approximation algorithms, is provided by Błażewicz et al. (1996) and Jain and Meeran (1999). Successful solutions methods are often based on the disjunctive graph representation that concisely models dependencies between operations. Roy and Sussmann (1964) introduced this representation.

In the last decades, different heuristic approaches efficiently solving the job-shop scheduling problem were proposed. Based on a decomposition approach, the shifting bottleneck heuristic of Adams et al. (1988) is one of the first heuristic approaches that effectively tackle the problem. The idea is to solve for each machine a one-machine scheduling problem to optimality under the assumption that many arc directions in the optimal one-machine schedules coincide with an optimal job shop schedule. As the name suggests, the shifting bottleneck heuristic schedules bottleneck machines first. Simulated Annealing was first used by Van Laarhoven et al. (1992) to solve the job-shop scheduling problem. As a neighborhood function, swapping resource arcs that are on the critical paths is proposed. This neighborhood function is shown to be connected, meaning that, from any solution, there is a finite number of transitions (swaps of resource arcs) that can lead to an optimal solution. It is also shown that swapping critical arcs always leads to a feasible solution and that no improvement can be expected from swapping resource arcs that are not critical. Taillard (1994) propose a parallel tabu search technique that uses the neighborhood function of Van Laarhoven et al. (1992). Nowicki and Smutnicki (1996) propose an efficient tabu search technique with a neighborhood function that uses the notion of block of Grabowski et al. (1986). A block corresponds to the maximal subsequence of critical operations without idle time on the same machine. Instead of swapping all critical arcs like in Taillard (1994), a single critical path is arbitrarily selected, and only critical arcs at the extremities of blocks are swapped. Besides simulated annealing and the tabu search, other trajectory metaheuristics are used to solve the job-shop scheduling problem, like GRASP in Aiex et al. (2003). Different population metaheuristics are also designed to solve this problem such as genetic algorithm in Gonçalves et al. (2005) and ant colony optimization that is combined with tabu search in Huang and Liao (2008).

This underlying problem is enriched by considering additional constraints. A well-studied problem that generalizes the job-shop scheduling problem is the flexible job-shop scheduling problem that was first studied by Brucker and Schlie (1990). Contrary to the classical job-shop scheduling problem, the flexible variant assumes that every operation can have more than one machine on which to be processed. The problem is thus to determine both an assignment and a sequence of the operations on the machines. Different solution approaches are used to tackle this problem, as shown in the survey of Chaudhry and Khan (2016). Only a few works proposing efficient heuristics are given here as examples. Brandimarte (1993) as well as Hurink et al. (1994) present a tabu search method for the problem. An extended version of the disjunctive graph model is proposed in Dauzère-Pérès and Paulli (1997), by taking into account the fact that operations must be assigned to machines. A connected neighborhood function, where there is no distinction between reassignment and resequencing, is used within a tabu search heuristic. Improved results are obtained by the tabu search approach of Mastrolilli and Gambardella (2000). The main contribution of this research is the reduction of the size of the neighborhoods by only focusing on moves that lead to the lowest makespan.

1.4.2.2 Constraints

Manufacturing scheduling is subject to a large variety of constraints. The reader can refer to the different general introductions on scheduling that are provided in the textbooks of Blazewicz et al. (2007), Brucker (2007), Framinan et al. (2014b) and Pinedo (2016). For the case of semiconductor manufacturing, the different encountered constraints are described for example in Mönch et al. (2011) and Yugma et al. (2015). In this section, we briefly describe the different constraints that are considered in this thesis and review some works that consider these constraints within a flexible job-shop environment. These constraints within the studied industrial context are described in detail in Section 2.3 and formalized in Section 4.1.

Release Times, Due Dates and Weights In the job-shop scheduling problem, it is assumed that all jobs are available at the beginning of the scheduling horizon. In practice, however, jobs may not be processed for various reasons including dynamic job arrivals or prior scheduling decisions. The earliest time a job can start is called *release time*. Furthermore, a job may have to be completed at some given time, called due date, for instance, due to delivery time commitments to customers and planning decisions. For each job, its release time and due date specify a time window within which its operations should or must be executed. Also, in practice, jobs may have a different level of importance. Depending on the context, the importance of a job can be evaluated depending, for instance, on the waiting time, the customer for which the job is assigned, the closeness of the due date given by the customer, the situation towards some process or planning constraints. To express this importance, each job is assigned a *weight*.

Re-entrance In a classical job-shop, each job usually visits a machine at most one time. Different situations can be found in different industries where jobs might visit one specific machine or stage more than once: Double firing processes in ceramic tile manufacturing, the repeated polishing operations in furniture manufacturing (Framinan et al. (2014b)) and semiconductor manufacturing (Ovacik and Uzsoy (2012)). In wafer fabrication, this is due to the fact that the basic set of processes required for a layer of circuitry are similar, and there may be more than twenty layers on a complex circuit such as a microprocessor. The terms *re-entrance* or *re-circulation* are used in the literature to qualify these situations. A study the re-entrant job-shop scheduling problem can be found in, for example, Zoghby et al. (2005).

Setup times In the classical job-shop scheduling problem, it is assumed that an operation can start on a machine as soon as this machine completes the previous operation. In practice, however, a machine may need for example adjustments, cleaning or testing before starting the newly available operation. In the literature, these operations carried out at machines that are not directly related with the processing of the jobs are commonly referred to as *changeovers* or *setup times* since they model the time that is needed to set up a machine. Extensive surveys on the integration of these constraints in scheduling problems can be found in Allahverdi (2015) and Allahverdi et al. (2008). Here, setup times are broadly differenti-

ated according to whether they depend or not on the job sequence and the fact they might be anticipatory or non-anticipatory. *Sequence-dependent setup times* model the situation where the duration of the setup time depends both on the job that was just processed on the machine and on the next job to be processed on the machine. Setup times are called *anticipatory* or *separable* if they can be performed as soon as the machine completes the previous job in the sequence. Conversely, *non-anticipatory* setup times require the availability of both the machine and the next job in the sequence before carrying out the setup. In semiconductor manufacturing, sequence-dependent setup times occur in some work areas, such as ion implantation and photolithography. Several works in the literature propose a heuristic to solve the job-shop scheduling problem with setup constraints. For example, the makespan is minimized using tabu search in Shen (2014) and simulated annealing in Naderi et al. (2010), while the maximum lateness is minimized using tabu search in González et al. (2013). Shen et al. (2018) address a flexible job-shop scheduling problem with sequence-dependent setup times and where the objective is to minimize the makespan.

Availability constraints The continuous availability of machines during the whole scheduling horizon is an assumption that might be justified in some cases but cannot apply to all industrial settings. Semiconductor manufacturing is an example where it is essential to consider machine availability constraints. In this industry, machines are complex, thus requiring frequent preventive maintenance, and very expensive, thus must be used as much as possible (Bureau et al. (2006)). Also, due to the complexity of scheduling problems in this industry, a rolling horizon procedure is necessary to decompose the problem over time (Ovacik and Uzsoy (2012)). When a scheduling problem is solved, some of the decisions from a previous schedule have to be considered, making some machines unavailable at the beginning of the horizon for the newly available jobs. It is then important to consider these machine unavailabilities in order to produce robust and feasible schedules. Surveys of scheduling problems with machine availability constraints can be found in Schmidt (2000) and Ma et al. (2010). Here, we only provide an overview of previous studies on scheduling problems with unavailability periods. In the research studying the integration of availability constraints in scheduling problems, different ways of modeling availability constraints have been proposed and investigated in different machine environments. Regarding the possibility for an operation to be interrupted by an unavailability period, Lee (1999) introduces the *semi-resumable* case that includes two cases: (i) *Resumable*, where the operation can be continued after being interrupted without any penalty after the end date of the unavailability period and (ii) *Non-resumable*, where the operation needs to be fully restarted. Aggoune (2002) introduces the fourth case for an operation to be interrupted by an unavailability period, called *non-preemptive*, to model the situation where an operation can be interrupted neither by another operation nor by an unavailability period.

When unavailability constraints are due to preventive maintenance, two additional cases can be distinguished. The first case, called *deterministic*, is when maintenance periods are fixed in advance. The second case, called *flexible*, corresponds to the situation where maintenance periods also have decision variables, i.e., each maintenance period has to be scheduled

in a given time window. Azem et al. (2012) propose heuristics for the job-shop problem with resource availability constraints where preemption between operations and unavailability periods are allowed and unavailability periods are flexible. Gao et al. (2006) tackle the flexible job-shop scheduling problem with non-fixed availability periods using genetic algorithms. In Zribi et al. (2008), a hierarchical approach is proposed that first solves the assignment problem and then the sequencing problem in the multipurpose machine job-shop scheduling problem. Mati (2010) proposes a tabu thresholding heuristic to solve the non-preemptive job-shop scheduling problem with machine unavailability for makespan minimization. The metaheuristic uses a new block-based neighborhood function, in which the block concept is generalized to include the unavailability periods of machines. Some sufficient conditions are also proposed to eliminate non-improving moves that involve operations at the borders of unavailability periods. Tamssaouet et al. (2018) investigate the job-shop scheduling problem with availability constraints. Changes that are related to the introduction of unavailability periods are highlighted, and a redefinition of critical operations is proposed. A move evaluation function that allows ignoring a large proportion of non-improving moves is used within simulated annealing and tabu search heuristics.

Batching processing A machine may be able to process several jobs simultaneously. These machines are called batching machines, while a *batch* is defined as a group of jobs that have to be processed jointly (Brucker et al. (1998)). A batch scheduling problem consists in grouping the jobs on each machine into batches and in scheduling these batches. Two types of batching problems can be found: Parallel processing (*p-batching*) and sequential processing (*s-batching*) (Mönch et al. (2012)). The term p-batching refers to the capability of machines to process more than one job at the same time. The s-batching refers to the presence of sequence-dependent setup times. In this case, jobs are grouped into families so that major setup times only occur when finishing production of one family while there is no setup times or minor ones between jobs of the same family. Surveys related to batching in general and to batching for semiconductor manufacturing can respectively be found in Potts and Kovalyov (2000) and Mathirajan and Sivakumar (2006). As can be seen in these surveys, scheduling on a single or parallel batching machines received most the research attention. In practice, these problems most of the time are solved using dispatching rules instead of optimization approaches. Below, some works that consider batching constraints within a job-shop environment are reviewed.

Most existing solution approaches for complex job-shop scheduling problems with batching machines rely on the disjunctive graph representation. A modified shifting bottleneck heuristic of Adams et al. (1988) is also proposed in Ovacik and Uzsoy (2012). The modified disjunctive graph representation of Ovacik and Uzsoy (2012), qualified as a *batch-aware* disjunctive graph in Knopp et al. (2017), introduces dedicated nodes to represent batching decisions explicitly. As the combination of delayed precedence constraints that are required by the shifting bottleneck heuristic and the dummy batching nodes produce cyclic graphs, Mason and Oey (2003) propose a cycle elimination procedure to promote cycle-free schedules. This paper considers a complex job-shop scheduling problem and batch processing ma-

chines while minimizing the total weighted tardiness. The same representation and heuristic are also used in Mason et al. (2005) and Mönch and Rose (2004). In Mönch et al. (2007), the considered job-shop environment contains parallel batching machines, machines with sequence-dependent setup times and re-entrant process flows. It is shown that the use of more advanced subproblem solution procedures, like a genetic algorithm, in a shifting bottleneck heuristic for complex job shops obtains better results compared to when using dispatching-based procedures. A simulated annealing heuristic is presented in Yugma et al. (2012) which relies on the batch-aware disjunctive graph and on batch specific moves where different objectives are optimized. This thesis uses the heuristic approach proposed by Knopp (2016) and Knopp et al. (2017) that relies on a novel modeling approach, called batch-oblivious. As in a classical conjunctive graph, the batch-oblivious conjunctive graph uses nodes to uniquely model operations and arcs to model precedence constraints on routes and resources. Instead of inserting additional nodes and arcs, batches are encoded in the arc weights. This new representation has several advantages. It reduces the structural complexity of the graph and allows reusing ideas and techniques for less complex problems such as the move proposed by Dauzère-Pérès and Paulli (1997) for the flexible job-shop scheduling problem. Last but not least, an integrated construction algorithm is proposed that simultaneously computes start dates and improves the solution during the graph traversal

Time lags In the classical job-shop scheduling problem, jobs are allowed to wait indefinitely in front of the machine of its next operation. In practice, there are many situations where this waiting time is constrained. The extreme situation, commonly known as *no-wait*, is where the operations of jobs must be carried out without interruptions. In the semiconductor manufacturing, intermediate situations between assuming indefinite waiting times and the no-wait case are found, where a maximum waiting time is allowed. In the literature, this is referred to as *maximum time lag* constraints. For example, there is often a time restriction between operations in the etch work area and the oxidation/deposition/diffusion work area (Mönch et al. (2011)). These time windows are installed by the process engineering department to prevent native oxidation and contamination effects on the wafer surface. In general, maximum time lags can occur for arbitrary pairs of operations of the same job. Thus, time lags can be adjacent or overlapping and, at the same operation, one maximum time lag can start, and another one can end. An overview and classification of different maximum time lag constraints that appear in semiconductor manufacturing is given in Klemmt and Mönch (2012). The addition of time lag constraints between successive job operations complicates even the usually simple task of finding a feasible schedule. Assuming that the waiting time can be indefinite, the classical job-shop scheduling problem also assumes that a job can start its next operation just after completing the previous operation. Similarly, jobs can be constrained in some situation to wait until a given minimum time has elapsed since the completion of the previous operation. Minimum time lags may be used to model transportation delays between two machines, the duration of activities that do not require resources (like drying or cooling down), or intermediate processes on non-bottleneck machines between two bottleneck machines (Zhang (2010)).

Few works considering time lags within job-shop scheduling problems can be found (González et al. (2015)). Based on a disjunctive graph modeling, a memetic algorithm is proposed in Caumond et al. (2008) to solve the job-shop scheduling the problem with maximum time lags between successive operations. Artigues et al. (2011) present a heuristic approach using an insertion heuristic and resource constraint propagation. Lacomme et al. (2012) consider generic time lags between arbitrary operations and propose a randomized heuristic. González et al. (2015) address the job-shop scheduling problem with time lags and sequence-dependent setup times. A scatter search approach, based on tabu search and path relinking, is proposed while using neighborhood structures aiming at reducing the makespan and regain feasibility. A generalization of the classical job-shop scheduling problem with release times, minimum time lags and a general precedence graph is considered in De Bontridder (2005). A tabu search algorithm is proposed to minimize the total weighted tardiness.

Complex machines In deterministic problems, it is usually considered in the literature that the processing time of an operation on a machine is fixed and known in advance. This modeling may be unrealistic when considering machines that show complex internal behavior. In this work, a machine is qualified as *complex* when the loading sequence of such machine influences the processing times of operations. A particular case of complex machines, found in semiconductor manufacturing and well studied in the literature, are *cluster tools*. A cluster tool combines several single-wafer processing chambers within a closed environment with a wafer handling robot. Cluster tools have been increasingly used for many processes, including photolithography, etching, deposition, and testing. Lee (2008) provides an overview of the literature on cluster tool scheduling. Most of the scheduling literature regarding cluster tools deals with internal scheduling (Mönch et al. (2011)), using dispatching rules or cyclic scheduling. Usually, from the perspective of semiconductor manufacturers, modifying these internal schedulers is difficult as they are proprietary to the manufacturer of the tools. The only degree of freedom to optimize the performances of such machines is the sequencing of jobs before their loading (Geiger et al. (1997)). The external scheduling of cluster tools, i.e., the job sequencing for this type of equipment, is challenging because of the cluster tool control and architecture (Dümmler (1999)). As a consequence, different sequences of lots will lead to different operation cycle times, i.e., the time between the processing start of the operation and its processing end on the machine. Regarding their modeling, there are in the literature two ways of dealing with the external scheduling of cluster tools. The first way consists in using a detailed simulation model of a cluster tool to evaluate the cycle times for job sequences for the scheduling algorithm (Dümmler (1999)). In the second way of modeling cluster tools, cycle time approximations are used, e.g., Niedermayer and Rose (2004).

Some machines found in the diffusion and wet cleaning area can be qualified as complex machines, without being cluster tools. However, as these two classes of tools share a complex behavior (Rotondo et al. (2015a)), studies conducted on cluster tools can provide useful insights into the behavior of diffusion machines and their modeling. Different reasons make, for example, wet benches complex: The possibility to have different batches running in parallel, the absence or the limited capacity of internal storage, the presence of bottleneck

components, the presence of internal scheduling algorithms, more or less sophisticated, and the diversity of processes with different processing times. Several works in the literature study the scheduling of wet benches. Lee et al. (2007a) consider the cyclic scheduling of jobs with different processing of a wet bench machine. Based on conditions for preventing deadlocks and collisions, deduced from the Petri net modeling of wet operations, a mixed integer programming model is proposed for determining the robot task sequences, the jobs in progress at the baths, and a timing schedule. The internal scheduling of wet bench machines is also solved using Constraint Programming (Novas and Henning (2012)) and branch and bound algorithms (Kim et al. (2012), Kim et al. (2014)). Geiger et al. (1997) formulate the problem of optimizing throughput of a wet bench machine as a permutation flow-shop scheduling problem with no-wait and blocking constraints, with the objective of minimizing the makespan. Construction heuristics and a tabu search are proposed to solve this problem. To evaluate the quality of a solution, each new permutation is passed through a robot control logic that was developed to mimic the actual controlling algorithm that is not available. Based on the same argument, Rotondo et al. (2015b) use a genetic algorithm to compute permutation sequences that are fed to a scheduling module that mimics the internal scheduling algorithm. In all the reviewed works dealing with the external scheduling of wet benches, it is always assumed that the batches are already formed.

In this thesis, the complexity of wet benches is handled by modeling in detail their internal resources and constraints. The proposed approach is based on the modeling, called *route-graph-aware conjunctive graph*, introduced by Knopp et al. (2014) in order to model complex non-batching machines. As the graph formulation of Knopp et al. (2014) is closely related to the job-shop scheduling formulation with processing alternatives introduced in Kis (2003), the results of the last work are applied and adapted. In Chapter 4, the batch-oblivious and route-graph-aware conjunctive graph representations are completely merged so that complex batching machines such as wet benches can be modeled in detail.

Production targets Different reasons can explain the complexity of semiconductor manufacturing: Multiple product types, long, re-entrant, and constrained process flows, diverse types of a large number of expensive and complex machines, unpredictable yield and machine downtimes. It is difficult and time-consuming to schedule lots within such a complex and large scale manufacturing system. Recall that the wafer fabrication process goes through multiple workshops, each consisting of machines with similar capabilities, specific constraints, and objectives. Accordingly, the scheduling problem at the fab level is decomposed into sub-problems, and the decisions are taken for each of the individual workshops (Ovacik and Uzsoy (2012). By doing this, to fully realize the potential improvements that can be brought by scheduling, the interrelation and interaction between these local solutions should be ensured. Planning and decision making in semiconductor manufacturing comprise several decision levels with scopes that range from the entire supply chain to the internal dispatching decisions within cluster tools. The scopes of different decision levels differ in their time horizon, the level of modeled detail, and the granularity of decisions to be taken. The decisions taken at a level become constraints to satisfy or targets to meet for the lower

levels. This hierarchical approach is widely accepted in semiconductor manufacturing (Uzsoy et al. (1994)). Shop-floor control solutions such as scheduling and dispatching are at the lower level in this hierarchy. When managing an individual work area, the consistency between local and operational decisions should be made consistent with the production plan determined at the fab level. A way of doing this is to feed aggregate information on global production objectives to the shop-floor controlling solution. From a scheduling perspective, release dates, due dates, job weights or *production targets* can take factory level objectives into account (Mönch and Drießel (2005), Sadeghi et al. (2015)). As the three first parameters are already discussed earlier, a focus is given below on production targets.

Due to the complexity of semiconductor manufacturing, it is common to set daily production targets by product type and by production stage. These production targets set a bridge between shop-floor control and the master production plan. In addition to the objective of guiding shop-floor controlling solutions, different operational objectives motivate production targets: Ensure the "linearity" of the production line, reduce the WIP and cycle times, and maximize tool utilization. Few works discussing the determination of production targets can be found in the literature. Chang et al. (1995) present an iterative algorithm for determining daily production targets and the corresponding machine allocation by product type and by production stage. After determining these targets with infinite capacity, they are modified by taking into consideration finite capacity and how many wafers may flow into the stage from its upstream stages within one day. In Wu et al. (1998), a computer-aided decision support system intended for daily target setting is described. The procedure computing the daily production targets at each stage and for each product is based on the master production schedule and capacity estimation. The targets serve as a guideline for dispatching by driving them to meet the higher level and mid-terms targets from the master production schedule and to maximize capacity utilization. Kao and Chang (2018) propose an approximation approach for computing production targets while taking into consideration the induced variation on the wafer flows. Sadeghi et al. (2015) proposes a general framework which aims at supporting and controlling local decisions by considering global objectives and information. The general idea is to provide to local policies a set of extra information in order to achieve global objectives while ensuring consistency between global and local decisions. The first proposed control mechanism is updating job weights. Optimization of the quantities of products to complete for each production stage and each period is the second proposed mechanism. These quantities become objectives to attain, but also constraints to satisfy, at the local level.

There is a patent interaction between planning and scheduling decisions. As higher level decisions, production planning decisions provide targets for scheduling decisions at a lower level. Also, poor scheduling decisions can degrade the performance and feasibility of planning decisions (Dauzère-Péres and Lasserre (2012)). If a hierarchical approach can be motivated, it is difficult to motivate the choice of ignoring production targets resulting from production planning when developing a shop-floor control solution. To the best of our knowledge, only Govind et al. (2008) explicitly mention the integration of production targets within shop-floor control. An integrated approach, adopted by Intel, to optimize fab efficiency is described. This approach consists of three main components: A linear program-

ming based optimization engine that provides production targets at product and operation level within the photolithography area; a scheduling component for complex areas and a dispatching component for less complex areas or to complete the scheduling component on complex areas. However, except the general description of this integrated approach, no detail is given about the approach in general or on how the production targets are considered within the scheduling component in particular. Chapter 4 proposes a new approach of considering production targets when scheduling individual areas in order to ensure the consistency between local decisions and global objectives. The modeling and the integration within the proposed scheduling approach are described in Section 4.2.3.

1.4.2.3 Criteria

Different criteria are considered in the scheduling literature. Different classifications of these criteria can be found. Depending on whether they involve information regarding due dates or deadlines, or not, they can be classified as due-date-related or non-due-date-related criteria. They can be feasibility related when they represent the satisfaction of a given constraint and can be rescheduling related if they involve information from two different schedules (Framinan et al. (2014a)). They can also belong to the "minimax" family where the maximum value of a set of functions is to be minimized or "minisum" if the sum of functions is to be minimized (T'kindt and Billaut (2006)). Most of the considered criteria are *regular*, i.e., a non-decreasing function of completion times of the jobs (e.g., makespan, maximum lateness, total weighted flow time, or total weighted tardiness). Definitions of these criteria are provided in the textbooks of Blazewicz et al. (2007), Brucker (2007), Framinan et al. (2014b) and Pinedo (2016). Total earliness is an example of non-regular criteria as it is a non-increasing function of completion times. A survey on scheduling problems with non-regular criteria is given in Baker and Scudder (1990).

When solving the job-shop scheduling problem, most of the literature deals with makespan minimization. An extension of the classical approaches which consider the makespan objective is the extension of the problem to other objective functions. A simulated annealing metaheuristic is developed in He et al. (1996) for the total tardiness based on left and right shift of operations to generate the neighborhoods. Pinedo and Singer (1999) present a shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. Essafi et al. (2008) propose a genetic algorithm that is combined with an iterated local search to minimize the total weighted tardiness. Mati et al. (2011) propose an algorithm for regular criteria that utilizes the neighborhood function that swaps critical arcs and an efficient method to evaluate moves. A general approach for minimizing any regular criterion in the flexible job-shop scheduling problem is proposed in García-León et al. (2015).

Within the context of semiconductor manufacturing, the criteria for scheduling problems are derived from performance measures for the entire fabs. The most important of these measures are the cycle time, the throughput and on-time delivery (Mönch et al. (2011)). For example, the cycle time can be minimized when using the total weighted flow time, the throughput of a machine or an area is maximized through makespan minimization and the

total weighted tardiness can be used as an on-time delivery measure. Due-date related criteria are used for example in Lee et al. (1992) and Mönch and Roob (2018). In practice, different criteria may be considered. For example, instead of considering the total weighted flow time for cycle time minimization, a new criterion with a close formulation is used in Artigues et al. (2006), Yugma et al. (2012), Bitar et al. (2016) and Knopp (2016). In addition to decomposing the scheduling problem at the fab level into sub-problems at individual workshops level, the problem is also decomposed in time. Instead of an infinite scheduling horizon, the scheduling problem is optimized over a finite horizon. Regarding throughput performance, it becomes irrelevant in this case to consider the makespan. In this context, the number of processed wafers within the given horizon is used for example as a way of maximizing the throughput (Artigues et al. (2006), Yugma et al. (2012), Bitar et al. (2016)). Due to the high cost of machines, criteria that are related to capacity utilization can be used. A non-regular criterion, called batching coefficient, is used in the context of batching machines by Artigues et al. (2006) and Yugma et al. (2012). Within the photolithography area where reticles are used as auxiliary resources, a non-regular criterion is used to minimize the number of their moves in Govind et al. (2008) and Bitar et al. (2016). Feasibility criteria can also be defined when it is not always possible to satisfy certain constraints. For example, Knopp (2016) proposes a new criterion that computes the violation of maximum time lags in a given schedule. This thesis uses all the relevant criteria that are defined in Artigues et al. (2006), Yugma et al. (2012), Bitar et al. (2016) and Knopp (2016). New criteria are defined to handle production targets and practical considerations.

1.5 Overview and Main Contributions

In this section, we present an overview of the structure of this manuscript and highlight the main contributions of the individual chapters.

Chapter 2, Problem Description

This chapter provides a detailed description of the diffusion and cleaning work area. The integration of the scheduling problems in these two areas is first motivated. The different features of the scheduling problem are described, and the adopted modeling choices are motivated. Then, the different criteria to be considered are detailed. They are either related to the area performance or to the satisfaction of constraints. The formulation of these criteria must answer the concerns of the area and the overall fab, and must consider the rolling horizon framework of the scheduling solution. This textual description of the studied scheduling problem serves as a basis to the formal modeling given of Chapter 4. The content of this chapter was kept alive during the whole project and periodically updated. The summarized knowledge is based on the expertise of many operators, process and production engineers, and managers, through immersion experiences and meetings.

Chapter 3, Improvements of the Batch-Oblivious Approach

The objective of this thesis is to design a scheduling solution to be used in an indus-

trial context. The starting point is the resolution approach proposed by Knopp et al. (2017), called the batch-oblivious approach, for a complex job-shop scheduling problem. This chapter first recalls the main components of the batch-oblivious approach, and the considered problem, less complex than the one described in Chapter 2. Then, different ideas to improve the efficiency of this approach are proposed. Efficient strategies that allow exploring more solutions are proposed. These strategies require a short computational time to explore the graph when searching for operations to complete unfilled batches. Based on the notion of active scheduling, new strategies are designed to accept delaying the start time of an unfilled batch in order to include late operations. The objective of these strategies is to reach solutions with good quality quickly.

Chapter 4, Extensions of the Batch-Oblivious Approach

This chapter extends the batch-oblivious approach in order to solve the industrial scheduling problem. The different features described in Chapter 2 are first formally modeled. A new objective that allows optimizing the throughput of the work area while taking into consideration the rolling horizon framework is proposed. To deal with production targets, a new criterion is proposed to increase the consistency between local scheduling decisions and the global production plan at the fab level. The integration of all considered constraints during the computation of schedules is detailed. A significant contribution is the possibility of modeling in detail complex batching machines such as wet benches in the cleaning area through a generalization of the batch-oblivious conjunctive graph. A construction algorithm that takes batching decisions on the fly while considering the internal constraints of complex machines is proposed.

Chapter 5, Multiobjective Optimization Approach

Due to the significant number of criteria, this chapter is dedicated to the study of the multiobjective aspect of the scheduling problem and proposes different approaches. In the two first approaches, the decision maker is given a flexible modeling of his preferences depending on whether the trade-off is permitted between any pair of criteria. The two approaches use differently these preferences during the search process and stores the set of nondominated solutions in a passive archive. These two approaches are compared to a third approach from the literature that uses the dominance status between the current solution and the set of nondominated solutions stored in an active archive. The comparison is performed based on the given preferences and known quality indicators, and shows that each approach can be more suitable depending on the context. This chapter ends by numerical experiments on industrial instances that attest the significant improvement that can be brought by the proposed approach.

Chapter 2

Problem Description

The main goal of this chapter is to provide a comprehensive, textual, in-depth description of all aspects that should be taken into account by a scheduling algorithm in order to be a viable component of a real scheduling system. Section 2.1 gives a brief description of the main considered work area and the processes that take place. The complex structures and behavior of the furnaces and wet benches are detailed in Section 2.2. Section 2.3 describes all the relevant constraints that have their origins from the physical system, processes or WIP management considerations in the wet cleaning and diffusion area. Finally, the different objectives to optimize are motivated and explained in Section 2.4.

2.1 Industrial Application: Cleaning and Diffusion Area

Diffusion processes are used in the production of semiconductors to deposit or grow a thin layer of insulating or conductive materials onto the wafers, using high temperature, in order to spread or to diffuse these impurities into the substrate. The main objective of these processes is primarily to alter the type and the level of conductivity of semiconductor materials. Even if the diffusion technologies are old, they are still widely used in the industry. This is mainly due to the ability of the machines performing these operations, called *furnaces* because of high temperatures (600°-1200°) that are required, to process simultaneously large batches with low cost. Some furnaces can process simultaneously up to 7 lots.

Due to batching capacity of the furnaces and to the fact that the operations performed on these machines have generally long processing times (3h -12h), the batching and scheduling decisions at the levels of the furnaces can affect the performance of the whole fab. Production schedulers must permanently decide whether to start processing an incomplete batch or to wait for additional lots. If a hasty decision is taken to load an incomplete batch, the long processing times make it difficult to absorb the negative effects of such a decision. Waiting for a long time negatively impacts the machine utilization and the cycle time of lots. Whatever system is used to schedule lots on the furnaces, it cannot be expected to obtain a high throughput if there is no communication with upstream operations. Production schedulers, whether human or computer systems, must orient upstream operations to process lots that can complete batches waiting in front of the furnaces. So in order to optimize the scheduling and batching decisions in the diffusion area, it is important to include upstream operations

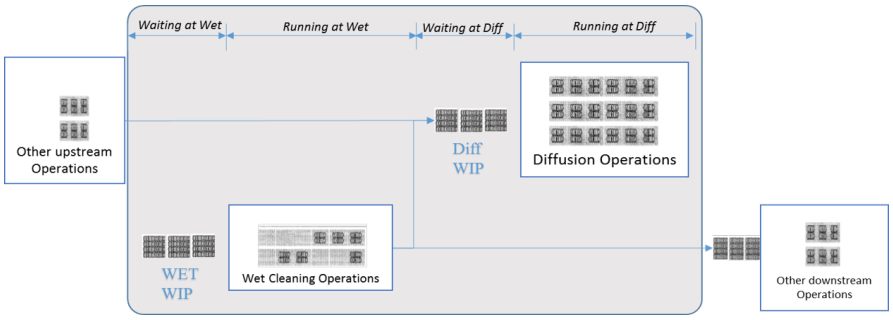


Figure 2.1 – *Optimization Scope in the Diffusion Area*

in the optimization scope. As it may be expected from the complexity of the semiconductor industry, almost all types of operations can precede diffusion operations. Including all of them can quickly transform the problem of scheduling the diffusion area to the problem of scheduling the whole fab.

Conveniently, the majority of upstream operations are wet cleaning operations that are performed on a small set of machines. Wet cleaning operations aim at removing contaminants (particles as well as metallic and organic) from the surface of the wafer, etching the thin oxide layer on the wafer and preparing the wafer surface for the next operations in the route. The cleaning is called wet as the operations are mainly performed using chemical solutions. The machines performing these cleaning operations are called *wet benches* and are also capable of processing multiple operations at the same time. However, considering that these cleaning machines process small batches with smaller processing times compared to diffusion machines and processes, it can be assumed that the filling of the batches in this sub-area is less critical, which allows stopping the extension of the optimization scope at this level. In addition to the first reason given above, the considerable number of time lag constraints between wet cleaning and diffusion operations also motivates the integration of wet cleaning operations in the optimization scope.

For the sake of conciseness, the whole area that is studied is called *diffusion area* in the remainder of the thesis. Figure 2.1 schematizes the optimization scope that is considered in the industrial application. The allocations decisions concern the machines that perform diffusion operations and machines that process wet cleaning operations. Scheduling decisions are taken for all the operations of lots that are waiting or currently being processed in the selected machines. Lots that are currently processed in other areas and will arrive soon in the diffusion area are also concerned by the scheduling decisions. These *arriving lots*, are integrated into the problem to solve thanks to the quite accurate estimation of their arrival time in the diffusion area. As shown in Figure 2.1, the wet benches, in addition to the furnaces, also feed machines in other areas, like dry etching, implantation, and lithography. Therefore, while optimizing the scheduling in the diffusion area, it is important to make sure that the situation in the other areas is not degraded.

2.2 Integration of Complex Machines

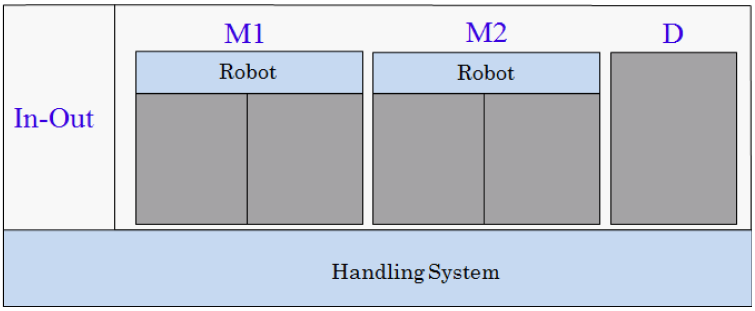
In most of the deterministic scheduling literature, the behavior of machines is assumed to be simple when the processing times of operations are fixed. This assumption can be realistic in many industrial contexts and situations. However, this modeling can be far from reality when the processing behavior of a machine cannot be simply reflected through fixed processing times of the operations processed on these machines. Cluster tools are an example of this kind of machines that present a complex behavior. These machines combine several processing modules with wafer handling robots in a closed environment (Lee (2008)). Most of the scheduling literature regarding cluster tools deals with internal scheduling (see Mönch et al. (2011)). When dealing with the external scheduling of these complex tools, there are in the literature two ways of modeling and integrating them in a larger scheduling problem. The difficulty of doing this lies in the fact that different sequences of lots will lead to different operation cycle times, i.e., the time between the processing start of the operation and its completion on the machine. This can be explained by the interaction between the different components, different internal constraints, and the internal scheduling algorithm. As a consequence, assuming a fixed processing time for an operation can be unrealistic. To deal with this, the first way consists in using a detailed simulation model of a cluster tool to evaluate the cycle times for job sequences for the scheduling algorithm, such as for example in Dümmler (1999). In the second way of modeling cluster tools, cycle time approximations are used, e.g., Niedermayer and Rose (2004).

The machines that are considered in the industrial application are not considered as cluster tools, but a subset of them can be described as complex machines. Relevant differences between these complex machines and cluster tools exist (Rotondo et al. (2015a)). However, as these two classes of tools share a complex behavior, studies conducted on cluster tools can provide useful insights into the behavior of diffusion machines and their modeling. In Section 2.2.2 and Section 2.2.1, descriptions of the detailed internal structures of the furnaces and wet benches are given, respectively.

2.2.1 Wet Benches

A wet bench is a tool used to carry out wet cleaning and etching operations in semiconductor manufacturing. Such tools are capable of batching, so multiple wafers from different lots can be processed at the same time. Benches commonly include several tanks (or modules or "baths"), each containing either cleaning or etching solutions and water in a rinsing tank. Due to the multiplicity of vendors, different types of wet benches in terms of structure can be found. The wet bench type described here is the prevalent one in the studied industrial context. A schematic representation of such machines is given in Figure 2.2. The processing part of this type of machines consists of two modules (M_1 and M_2) and a dryer (D) where the wafers are rinsed and dried. Each module consists of its own robot and two tanks. The handling part, in addition to the two robots dealing with the wafer handling inside the two modules, consists of: A loading port and an unloading port ; a robot that ensures the batching

Figure 2.2 – *Example of the structure of a wet bench machine.*



after the loading and the unbatching after the processing ; a robot that transports the wafers between the different components (M_1, M_2, D).

After loading all the lots of a batch into the machine, all the wafers are consolidated in one batch. After this, depending on the type of the required process, the batch is transported from one component to another one. There are two types of process that can be categorized according to the route followed by a batch through the components M_1, M_2 and D. If a batch is concerned with the first process type, called *short process*, it visits one of the processing modules before being dried in the dryer. If the second process type is used, called *long process*, the batch visits all the components following this sequence ($M_1 \rightarrow M_2 \rightarrow D$).

Different reasons explain why it is not realistic to only associate fixed processing times to operations performed on this type of wet benches. The first is the possibility of having several batches being processed at the same time. In other words, there may be a batch in each of the processing components (M_1, M_2, D), in addition to a batch that may be loaded and another one being unloaded. The second reason is the absence of storage capacity between modules. This is modeled in the scheduling literature as blocking constraints (Hall and Sriskandarajah (1996)). With blocking constraints, a batch, having completed processing on a module, remains on it until the next module becomes available for processing.

To illustrate this complex behavior, a small industrial instance is used. Table 2.1 provides four examples of processes with elementary processing times, given in minutes, on wet bench modules. The first three processes (1, 2 and 3) are short processes while the last one is a long process. The first step of Processes 2 and 3 can be performed either on Module M_1 or on Module M_2. For Process 1, the first step can only be performed on Module M_1. The considered problem is described in Table 2.2. Eight lots have to be scheduled on a wet bench machine, four of which require the same short process (Process 1) and four others the same long process (Process 4). Column *"Possible Sequences"* provides the sequence each operation follows in the wet bench machine. Note that the first step of the short process can only be processed by Module M_1. In order to optimize machine throughput, the optimization criterion is the makespan.

The optimal sequence, in terms of machine throughput, is shown on a Gantt chart from a job perspective in Figure 2.3(a) and from a machine perspective in Figure 2.3(b). Different

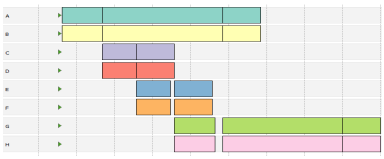
Table 2.1 – *Example of processes on a wet bench machine.*

Process	Type	M_1	M_2	D
1	Short	13	-	15
2	Short	38		15
3	Short	21		15
4	Long	16	47	15

Table 2.2 – *Small size industrial problem instance.*

LotID	**ProcessID**	**Process Type**	**Possible Sequences**
A	4	Long Process	$M_1 \rightarrow M2 \rightarrow D$
B	4	Long Process	$M_1 \rightarrow M2 \rightarrow D$
C	1	Short Process	$M_1 \rightarrow D$
D	1	Short Process	$M_1 \rightarrow D$
E	1	Short Process	$M_1 \rightarrow D$
F	1	Short Process	$M_1 \rightarrow D$
G	4	Long Process	$M_1 \rightarrow M_2 \rightarrow D$
H	4	Long Process	$M_1 \rightarrow M_2 \rightarrow D$

aspects of the complex behavior can be identified in these figures. First, the example shows the possibility of parallel processing of multiple batches. Even if the batch containing jobs A and B is the first one to be loaded on the machine, it is ready to be unloaded only when the two next batches are unloaded. Second, the consequence of the blocking constraint can be identified in Figure 2.3(b). Considering the step sequence on Module M_1, the last batch in the sequence, containing jobs G and H, does not start its processing directly after its predecessor, even if it is available at the beginning of the scheduling horizon. This is because Module M_1 is blocked by the batch containing jobs E and F. This last batch is waiting for the batch containing jobs D and E to free the dryer so that it can free Module M_1.



(a) *Job perspective Gantt Chart.*

(b) *Machine perspective Gantt Chart.*

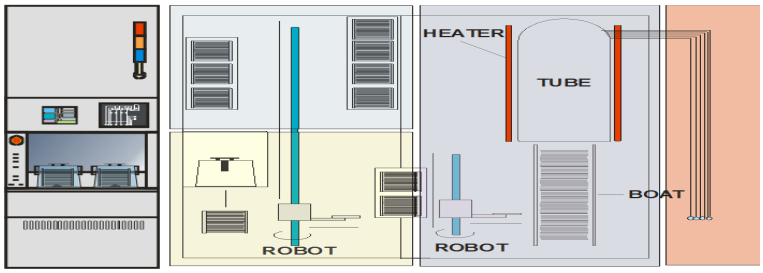
Figure 2.3 – *Scheduling* 8 *jobs on a wet bench machine, using route graph modeling.*

2.2.2 Furnaces

Furnaces perform deposition, oxidation and annealing processes. These machines are also capable of batching. Some furnaces can process up to 7 lots at the same time. Similarly to wet benches, different kinds of furnaces can be found in the shop floor. The furnace type described here is the prevalent one in the studied industrial context. A schematic representation of such machines is given in Figure 2.4. For the processing part, it is a vertically oriented thermal processor composed of a processing chamber called a *tube* and a separated movable wafer holder called a *boat*, which has a vertical array of notches into which the wafers are received and held horizontally. When the boat is fully loaded, it is moved inside the tube where the process takes place. When the process is finished and the tube cooled, the tube is moved down in order to be unloaded. The handling part consists of: Two *load ports* dealing with the loading of the lots from outside the machine and the unloading in the other direction; a *loading robot* dealing with the handling of the containers in all of their movements inside the machine; a *charging robot* dealing with the wafer transfer from the containers to the boat before the processing and in the other direction after the end of the processing of wafers inside the tube. In addition to these moving components, there is a *stocker* playing a role of the inventory stand where containers are stored , full or empty.

Typically, the loading robot received the containers of lots that are introduced into the machine by the loading ports and moves them to the stocker. When the batch is fully loaded into the stocker, the charging robot moves all the wafers into the boat. When all the wafers of the batch are in the boat, the boat is moved into the tube where the process takes place. At the end of the process, the boat is first cooled down before the charging robot puts back the processed wafers in their containers waiting in the stocker. Finally, the loading robot transfers the containers from the stocker to the unloading port. Following this description, it may be realistic to assign diffusion operations a fixed processing time that includes the actual processing time in the tube and the sum of all handling times. However, the internal buffer allows improving the throughput of the machine while making its behavior more complex. An additional batch B_2 can be loaded into the machine while another one B_1 is being processed, which advances its processing start time by the time needed to load the lots of B_2 into the stocker. Moreover, the charging robot can simultaneously load Batch B_2 into the boat while B_1 is unloaded.

Figure 2.4 – *Example of the structure of a furnace.*



2.3 Constraints

This section describes all the constraints that the scheduling algorithm should take into account in order to produce feasible and robust schedules that can be implemented in a real workshop. For each considered constraint, some statistics on the real application are given to illustrate the complexity of the problem to solve.

2.3.1 Lot

The silicon wafers are the input and the elementary production units in the wafer semiconductor manufacturing facility. These production units are transferred and processed in lots. Currently the convention is that a *lot* has a *size* equal to or less than 25 wafers. Each lot belongs to a certain product. From manufacturing perspective, products differ in the routes they follow in the fab. In addition to the difference in the size and the route in the fab, lots also differ in terms of *priority* where certain lots in a wafer fab are more important than others. Then, the scheduling system should assign the lots to machines and sequences them by adhering as much as possible to this priority. For each lot, a *release date* is given that specifies its arrival date in the diffusion area. Finally, the lots can be in different status at any time. A lot is considered *waiting* if it is in front of a machine. A lot can be *running* if it is currently being processed in a machine. A lot is *frozen* if at least one of its subsequent operations is already scheduled. For instance, lots of a batch that is already loaded in a furnace, while another one is being processed, are considered frozen as this decision cannot be changed in normal conditions. Within a rolling horizon framework, some of the waiting lots can be considered as frozen if it is decided to freeze the decisions that were previously taken and that are related to their next operations in their routes. In all these cases, the release date occurs before the start of the scheduling horizon. Finally, lots that are not yet available in the diffusion area, and for which a quite reliable arrival time estimation is given, are called *arriving lots*. The release dates of arriving lots occurs after the start of the scheduling horizon. In the industrial application, the scheduling algorithm has to solve instances with 500 lots on average. The average distribution of the different status over the population of the lots is: 40% waiting, 35% running, 15% frozen and 10% running.

2.3.2 Routing Constraints

A semiconductor product specifies the integrated circuit that is manufactured and the variety of these circuits is as wide as the variety of functions that they are meant to fulfill. However, all integrated circuits are made of the same few basic structures and manufacturing processes on the same set of tools. The differences between the products lie in the the variation in the number, the sequence and the combination of the process operations and the machines setups. It is the notion of the route that allows describing the unique process flow that a lot of a particular product takes. It can be defined as the sequence of operations that wafers must follow through the factory and the details how processing should be performed at each oper-

ation. It begins with lot start and ends with lot ship. It should be noted that not all operations must be performed; metrology operations can be skipped with a specified percentage.

Several difficulties in managing a fab come from the complexity of these routes. First, a route usually has several hundred operations. Especially in the case of high-mix fabs, there is a large number of routes that coexist in parallel or intersect at the level of many operations. Also, wafer manufacturing has a high degree of reentrancy, i.e. the same work area is visited many times to perform different operations within the same route. In the industrial application, as the objective is to optimize the scheduling of lots within the diffusion area, the routes of the different products will not be considered in their totality. Instead, only the portions of the routes containing the operations that have to be processed on the furnaces and the wet benches are considered. In each sub-route, there will only be diffusion or cleaning operations. If, for instance, only one operation separates a sequence of diffusion and cleaning operations, they will belong to different sub-routes. In the industrial application, the scheduling algorithm has to solve instances with 164 different routes on average. The number of operations per route varies from 1 to 7, with an average of 3 operations. On average, scheduling decisions should be taken for more than $1,500$ operations. Figure 2.5 shows the partition of the lots regarding the length of their routes.

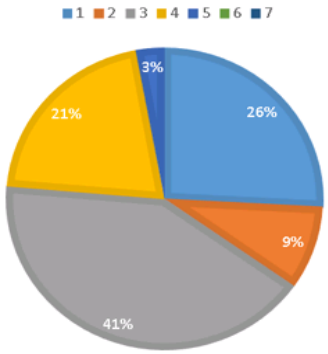


Figure 2.5 – *Distribution of route lengths (number of operations)*

2.3.3 Recipes

As described above, a lot of a given product has to follow a sequence of operations defined by the route of the product. A *recipe* is associated with each operation of each route. The recipe can be defined as a pre-planned and reusable set of instructions and settings that specify how an operation is to be performed by a machine on a wafer in order to get the desired output. In other words, a recipe specifies the type of process a lot must undergo at a given level of operation. In the industrial application, there are on average more than 130 recipes in each problem instance. On average, four machines are able to process a given recipe.

2.3.4 Qualifications

The fierce competition, the globalization, and the technological breakthrough have shortened the product lifecycle, requiring the companies to reduce their manufacturing response time. Therefore, flexible production systems are required to quickly respond to market fluctuation, increase the production agility and answer the diverse market demand. One of the categories of flexibility in a manufacturing system is machine flexibility. Machine flexibility is obtained through what is called a *qualification*. It is a kind of setup with the difference that a qualification is performed once and not before each production run. Making a machine flexible means qualifying this machine for several different recipes. It is also important to indicate that, in addition to the process of making the machine capable to perform an operation with a certain recipe, qualification defines also the capacity of the machine to process this recipe (Johnzén et al. (2011), Rowshannahad et al. (2015)).

However, due to technical and economic reasons, it is not possible to obtain total flexibility of the machines. The technical reasons can be hardware or software restrictions that make it impossible to qualify all the recipes on a machine. Besides, qualifying a recipe on a machine is often time and energy-consuming. Test products must be used for test runs. During test runs, the machines are under scheduled downtime status, therefore in a non-productive status. Metrology and defect inspection resources must also be extensively used. Hence, it is not economically wise to perform too many qualifications. Due to the restrictions explained above, the wet cleaning machines and the furnaces are qualified only for a subset of recipes. Therefore, we are given for each machine, the set of recipes for which it is qualified. This information will be given at the machine level. In the industrial instances, a furnace can process 4 recipes on average, and a wet bench can process 25 recipes on average.

2.3.5 Maximum Time Lags

The physical and chemical processes may set up time constraints between different operations. For example, the time between some operations must be limited to avoid contamination and oxidation. So, in order to prevent these risks, time windows are defined by process engineers to respect these time constraints. So, we can be given a *maximum time lag* constraints for each ordered pair of distinct operations of each lot. Such a constraint specifies the maximum period of time between the beginning (or end) of processing of an operation until the beginning (or end) of processing of the following operation in the route of the lot. It is important to outline that these time lag constraints can be chained in the sense that there can be operations where one maximum time lag constraint ends and another one starts. For resulting schedules, it is always required that both constraints are observed. As a consequence, in some cases, starting the first operation of the earlier constraint can be impossible because the later constraint cannot be fulfilled. In the real application, on average 35% of lots are affected by these constraints. At the beginning of the scheduling horizon, 20% of the lots are already under time constraints. The maximum period of time lies between 12 hours and one day.

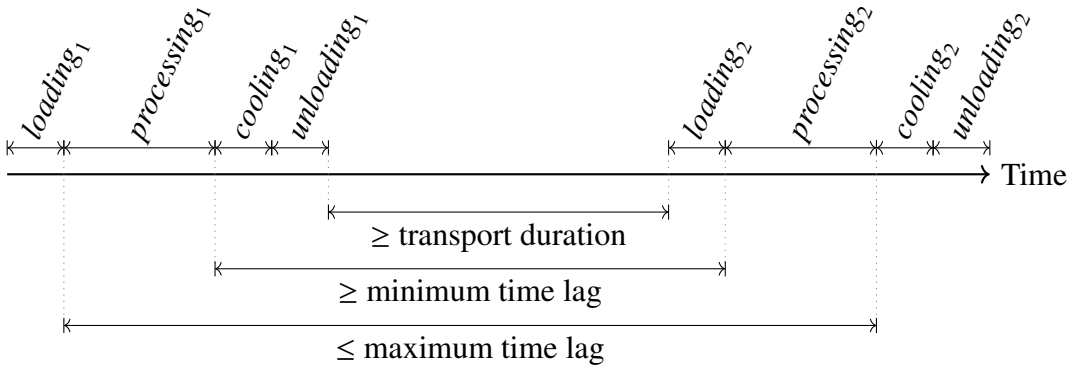


Figure 2.6 – *Time constraints for two consecutive operations of a lot*

As this will be justified later, these constraints are considered soft ones. Instead of looking for fully respecting these constraints, the objective is to minimize as much as possible their violation. In this case, the difference in the impact of their violation should be taken into consideration. The violation of a constraint that results in lot scrapping should be prioritized before a violation constraint that results in yield loss. For each maximum time lag constraint, we are given a *violation cost* that represents the impact on lot quality if it is violated.

2.3.6 Minimum Time Lags

A *minimum time lag* constraint specifies a minimum delay between the execution of two operations of the same job, not necessarily consecutive. These constraints can be used to model minimum delays that are imposed for process considerations. These time lags can also be used to model the transfer of a job from one machine to the next if it requires a transportation time. Finally, these constraints are also used to differentiate between the cycle time of an operation and the *actual* duration this operation uses its resources. This last case is used within a proposed modeling of wet benches.

Whatever is modeled through these constraints, a *minimum duration* during which the lot has to wait before starting the next operation is given. When process constraints are modeled, this duration is determined by process engineers. When transport times are modeled through these constraints, the minimum duration first depends on the distance that separates the machine to which the next operation of a lot is assigned to. When lot transportation is manually performed by operators, the minimum duration also depends on their speed and their availability, conditioned by the WIP level in the area. Finally, when used to model the operation cycle time, the minimum duration is equal to the theoretical processing duration. Figure 2.6 illustrates an example of time constraints between two consecutive operations.

2.3.7 Availability Constraints

Machines may be unavailable during certain periods of time for different reasons, such as failures or unexpected quality control problems. To avoid these failures without stopping production too often, preventive maintenance are planned on machines by trading off between planned unproductive downtimes and the risk of unscheduled downtimes due to machine failures. These preventive maintenance operations make machines unavailable for processing operations. As this work deals with deterministic scheduling, only unavailability periods that are known in advance are considered such as preventive maintenance operations, current machine failures or machine unavailabilities that are due to previous scheduling decisions within a rolling horizon framework.

The continuous availability of machines during the whole scheduling horizon is an assumption that might be justified in some cases but cannot apply to all industrial settings. Semiconductor manufacturing is an example where it is important to consider machine availability constraints. In this industry, machines are very expensive, thus must be used as much as possible (Bureau et al. (2006)). In the same time, due to their complexity, these machines require frequent preventive maintenance. Also, due to the complexity of scheduling problems in this industry, a rolling horizon procedure is necessary to decompose the problem over time (Ovacik and Uzsoy (2012)). When a scheduling problem is solved, some of the decisions from a previous schedule have to be considered, making some machines unavailable at the beginning of the horizon for the newly available jobs. Then, it is important to consider these machine unavailabilities in order to produce robust and feasible schedules. In the industrial application, when unavailability periods are used to model preventive maintenance, there are on average five maintenance opertions a day, with an average duration of 10 hours and a median duration of 4 hours. On average, 85% of machines are unavailable at the beginning of the scheduling horizon due to previous scheduling decisions (frozen and running lots).

Among the different possibilities of modeling availability constraints, the modeling that better fits the industrial setting is the deterministic and non-preemptive modeling. The unavailability periods are considered deterministic as it is assumed that each unavailability of each machine has a fixed start and end time. The non-preemptive modeling represents the situation where an operation can be interrupted neither by another operation nor by an unavailability period (Aggoune (2002)). Notice that when machines are modeled in details, these availability constraints may be given at component level.

2.3.8 Setup Times Constraints

After completing an operation, a machine may be set up, i.e. made ready, for the next operation by various tasks including machine cleaning, tool changing, and temperature adjusting. A machine setup before the processing of the first operation and after the last operation could also be necessary. The time needed for the setup, called *setup time* or change over time, may depend on the previous and next operations to be processed on the machine. Such setup times are called *sequence-dependent*. If the setup time only depends on the next operation to be

executed, it is called *sequence-independent*.

In the studied area, there is no actual setup time needed for the machine within the optimization scope. However, these constraints are used to model the inefficiencies that are induced by the direct succession of two recipes on the same machine. An inefficiency is modeled as an estimation of the sum of unproductive times of machine processing components. These artificial setup times are sequence-dependent. For each concerned machine, a setup duration is given for each recipe pair. It defines a minimum duration between the end of processing of an operation and the beginning of processing of the following operation on the same machine. These constraints, in addition to minimum time lags, are used in a proposed modeling of wet benches.

2.3.9 Batching Constraints

The most important constraint to consider in this study is parallel batching, abbreviated as p-batching, which refers to the capability of machines to process more than one job at the same time. All the machines considered in the industrial application are batching machines. All the lots in a batch are processed together, start at the same time and have the same processing time. On a qualified batching machine, an operation can only be batched with operations that share the same recipe.

While the objective is to process as many lots as possible in the same batch, the physical batching capacity of each machine cannot be exceeded. This physical capacity varies from three to seven lots for furnaces and is equal to two lots for wet bench machines. Due to process considerations, the maximal batching capacity for some recipes is lower than the physical capacity of a machine. While trying to optimize batch filling, the proposed schedule should respect the *maximal batching capacities* that are given for each couple (machine, qualified recipe). Moreover, also due to process consideration, some recipes on wet cleaning machines require a *minimal batch size* that forbids loading a batch on the concerned machines if the batch size is lower than the minimal given batch size.

2.3.10 Quality Control Tasks

As the semiconductor industry is subject to high-quality requirements, manufacturing processes must be permanently monitored. One of the means to ensure that the process is stable, the machine clean and under control is the machine-oriented inspection called *quality control task*, or quality task in short. To perform these quality tasks, specific monitoring lots are regularly added. As it is usually mandatory to monitor several parameters of the machine, several quality tasks should be conducted on each machine. In addition to the classification of the quality tasks according to the parameter or the state monitored in the machine, there is another classification, more relevant for the scheduling, that is done according to the conditions for a quality task to be performed on a machine. In this classification, two main types of quality tasks can be distinguished:

1. Calendar quality tasks: Quality tasks in this class should be done periodically, i.e. after a fixed period of time since the last one;

2. Dynamic Quality Tasks: Quality tasks in this class must be conducted when the number of wafers processed by the machine and impacted by the monitored parameter is approaching a certain limit corresponding to the maximum accepted risk.

All quality tasks should be conducted in order to meet quality requirements and prevent high cost due to the scrapping of production lots. However, production capacity is wasted if the quality tasks are performed when the risk is minimal. So, it is preferable to minimize the number of quality tasks as long as that does not lead to the breach of the fixed conditions.

For most of the furnaces, it is possible to ignore quality tasks as they are batched with production lots and have their own dedicated production capacity. Therefore, except for the loading and unloading durations, quality tasks can be ignored in the scheduling model. However, in our industrial setting, there is a group of furnaces for which quality tasks have to be done separately from the production lots and which last as long as processing operations. Therefore, it is imperative to include quality tasks in the scheduling model. For wet cleaning machines, there are different quality tasks that are conducted in order to monitor different parameters. These tasks consume production capacity especially as they cannot be batched with production lots.

In the proposed scheduling system, only calendar quality tasks are taken into considerations. Even if dynamic quality tasks are currently negligible, considering them could be important in the future. Regarding calendar tasks, instead of having a fixed mandatory start time, it is only mandatory to process them during the shift to which they are assigned. This can be modeled as flexible unavailability periods. As these constraints are currently not handled by our model, these tasks are considered as special lots to which are associated with artificial maximal time lags.

2.3.11 Production Targets

Because of the long and complex production processes in wafer manufacturing, linearity constraints are defined in order to have intermediate controls on lot manufacturing processes and to balance processing routes. Routes are divided into a specified number of blocks where activities within a block are considered as WIP level to avoid WIP accumulation in the fab. The goal is to control the production process within each block in order, for instance, to maintain the WIP levels close to predetermined WIP level targets. In our context, linearity consists of smoothing differences between the WIP level of a block and its fixed target (Wu (2014)).

The control of linearity constraints in semiconductor manufacturing is considered as a global objective. More precisely, we seek to guide the real-time local scheduling decisions to minimize the deviation of the WIP levels or the cycle times within each block from their desired targets. Different mechanisms can be used to orient local schedulers to the realization

of global objectives. Priorities of lots can be updated in order to include the information and the decisions at the global level. Also, quantities of products to complete in operations can be sent and imposed at the local level.

These quantities, also called *move targets*, are given usually on a daily basis and have been extensively used as a means of production control in the semiconductor industry (Wu (2014)). In contrast with their importance in practice, move targets did not capture much attention from researchers. Only few works on real-time scheduling mention these constraints without formalizing it (Ham et al. (2006), Lee et al. (2008), Lee et al. (2007b), Wu et al. (1998)). The problem of computing these targets is more investigated. Wu et al. (1998) propose a procedure to compute the daily targets based on the master production schedule, the capacity estimation. Chuang and Lin (2003) propose an algorithm that begins with setting the required delivery date to define other related requirements by backtracking the production line requirements to determine how many stages need to be passed before delivery, and to decide where and how to set up and arrange the production machine, in order to complete and deliver the ordered goods on time. Sadeghi (2017) studied the consistency of global and local scheduling decisions in semiconductor manufacturing. A general framework which aims at supporting and controlling local decisions by considering global objectives and information is proposed. A base Linear Programming model which considers a linearity objective is used to compute the move targets that local schedulers must realize.

At the scheduling level, these imposed quantities are considered as *production targets*. Fixed quantities to produce are given to certain group of operations. Formally, we consider n lots $L = \{L_1, L_2, ..., L_n\}$ to be processed on a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each lot L_i is composed of a linear sequence of n_i operations $O_i = \{O_{i1}, O_{i2}, \ldots, O_{in_i}\}$. Let us consider p *production targets* $T = \{T_1, T_2, \ldots, T_p\}$. For each operation O_{ij} of lot L_i, we associate a set T_{ij} to which the operation *contributes*:

$$T_{ij} = \{k \in T / O_{ij} \text{ contributes to the realization of target } k\} \tag{2.1}$$

Note that it is possible to have $T_{ij} = \emptyset$, which means that operation O_{ij} does not contribute to any of the production targets. It is also possible that $|T_{ij}| > 1$, which means that operation O_{ij} contributes to more that one production target. To each production target k is associated a *requested production volume* $D_k \in \mathbb{N}$. In the industrial context, these quantities are given for a horizon of one day and the area management tries to be close as much as possible to these objective volumes. There are production targets that are defined to manage the whole fab daily, and these are the most important targets. There is also another set of targets which stems from the monthly production plan. On average, there are 25 targets, and the requested volume, outside the those stemming from the monthly plan, varies from 150 to 2000 wafers.

2.3.12 Interlacing Constraints

As motivated in Section 1.3, the rolling horizon is used to cope with the dynamic and stochastic nature of the scheduling problem in the industrial setting. In this section, we describe

constraints that interlace the current scope with adjacent scopes. At a given time, when a schedule has to be computed, a *scheduling horizon* is given. Instead of being part of the problem definition, the duration of this horizon is a parameter that has to be determined carefully. When considering the state of the diffusion area at the beginning of the scheduling horizon, machines can be occupied by frozen and running lots. These machines are not available until all these lots are processed and unloaded. Additionally, the initial states of all machines and their relevant components must be given at the beginning of the scheduling horizon. These states can be the initial setup of the machine or its unavailability due to maintenance operations or quality tasks.

The time lags, either maximum or minimum, defines relative time windows. However, there are time lags that have their first operation in an upstream area that is not considered in the optimization scope. In such case, when a lot is concerned with such time lag, it is already under constraints when it arrives in the area. This situation applies also to time lags that have their two operations within the optimization scope. At the beginning of the scheduling horizon, a subset of lots have already gone through the first operation of a time lag. In such case, the start or completion times of the first operation of already triggered time lags should be given. The required information depends on whether the time lag is triggered before or after the processing of the first operation. So, instead of a relative window, such time lags prescribe absolute windows. As the maximum time lags can be chained, triggering one results in transforming all the successive linked time lags to absolute windows.

2.4 Criteria

After listing all the constraints to satisfy, we motivate and describe in the following criteria to optimize. As discussed in the literature review given in Section 1.4.2.3, the criteria that are classically investigated in the scheduling literature are not adapted to the industrial setting we are studying. Some of the criteria described in this section are defined by previous works dealing with the scheduling within the context of the semiconductor manufacturing (Artigues et al. (2006), Yugma et al. (2012), Bitar (2015) and Knopp (2016)). This section also motivates the definition of new criteria. A formal definition of these criteria is given in Section 4.2.1.

2.4.1 Weighted Number of Moves

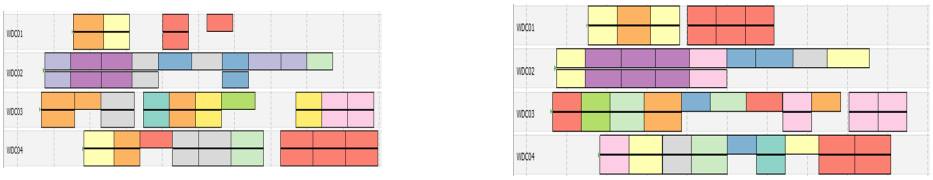
This objective is related to the throughput of the working area. A *move* defines the processing of a single wafer on a machine. The number of moves of a batch is the number of wafers of all lots contained in that batch. To promote the processing of operations of lots with high priority, the number of wafers of lot is weighted by the priority of the lot. The number of moves of the whole area is the sum of number of moves of all the batches.

$$(\text{Weighted number of moves of Batch } B) = \sum_{(\text{Lots } L \text{ in } B)} (\text{priority of } L) \cdot (\#\text{wafers of } L) .$$

When scheduling lots, a limited horizon should be considered because, if it is not the case, all the operations in the problem instance are scheduled and then all scheduling solutions are equivalent. Optimization would not make sense in this case. Hence, only those batches started within the scheduling horizon must be taken into account. If the processing of an operation is started within the scheduling horizon and finished after the end of the horizon, the weighted number of moves of a batch is multiplied by the proportion of the processing time that occurs within the horizon. This criterion is already defined and studied in several works (Artigues et al. (2006), Yugma et al. (2012), Bitar (2015)).

2.4.2 Discounted Weighted Number of Moves

When considering the weighted number of moves, all the solutions where the same operations are processed within the scheduling horizon are considered equivalent. When taking the rolling horizon framework into account, some solutions may be considered better than others on a long-term perspective. Between two solutions with the same weighted number of moves, it is better to choose the one where full batches are scheduled first. By doing this, incomplete batches that are scheduled later may be completed when the scheduler is run again. Figure 2.7 shows two solutions for the same problem: in Figure 2.7(a), only the number of started moves is considered; in Figure 2.7(b), the discounted number of moves is considered. In the same way, a solution in which lots with high priority are sequenced first is better than a solution in which these lots are produced at the end of the horizon. These different situations should be considered when only a part of the proposed schedule is really implemented in the shop floor.



(a) *without discounting* **(b)** *with discounting*

Figure 2.7 – *Example showing the advantage of using the discounted number of moves*

The considerations above motivate the definition of a new criterion, called *Discounted Weighted Number of Moves*, that promotes solution where full batches and lots with high priorities are sequenced at the beginning of the scheduling horizon. The term "discounted" expresses the idea that time affects the value of a move and that a wafer processed now is worth more than a wafer processed in one hour. A different discounting scheme can be imagined. In the industrial application of Bitar (2015), a stepwise discounting factor is used. In this case, the discounting factor is the same for all operations that are started within an

elementary period. For the used indicator to be relevant, the duration of this elementary period and the decreasing of the discounting factor should be preprocessed carefully. Instead of this, a linear discounting factor is used in this work.

2.4.3 Batching Coefficient

The *batching coefficient* is a performance indicator that describes the capacity usage of the operations performed on batching machines. Defined on the planning horizon, it is calculated as the number of moves divided by the sum of the number of batches performed on each machine, times the maximum capacity of that machine. Note that the denominator is the number of lots that could be performed if the machine is loaded up to its maximum capacity. With M denoting the set of machines and H denoting the planning horizon, the objective can be written as

$$\frac{(\text{\#moves in } H)}{\sum_{m \in M} (\text{\#batches performed on } m \text{ in } H) \cdot (\text{wafer capacity of } m)}.$$

Note that this quotient is equal to one if all batches are filled to their maximum wafer capacity. We include the objective of maximizing the batching coefficient into our objective function. This objective can be seen as a means to support the improvement of other goals. This objective also avoids the cost that is associated with each machine run. This criterion is also defined and studied in several works (Artigues et al. (2006), Yugma et al. (2012), Knopp (2016)).

2.4.4 Weighted Flow Factor

Competitive production cycle time is a critical performance for semiconductor factories for several reasons. For device prototyping, typically involving several design changes, a shorter product development time allows a quicker response to rapidly changing market needs. For production lines, a smaller cycle time improves the ability to satisfy customer demands. Also, for the same level of throughput, a shorter cycle time results in a smaller work in process that not only reduces the capital tied up but also leads to an uncluttered plant floor. Finally, the smaller the cycle time, the smaller the inventory buffer that needs to be maintained at the downstream machines. When product designs become obsolescent, such inventory may lose value. There is also a technological reason for reducing the cycle time. The shorter the time wafers are exposed to aerial contaminants while waiting for processing, the smaller the yield loss (Lu et al. (1994)).

Scheduling can contribute to the improvement of this operational indicator by reducing the waiting time of the lots in front of the machines in the optimized area. *Weighted flow factor* is the defined criterion to evaluate the waiting times of lots in the diffusion area in order to reduce the cycle times. To specify the criterion, we introduce the *theoretical route duration* of a lot, which is equal to the shortest possible manufacturing duration of a lot.

In order to define this, we choose the fastest machine for each operation of the lot. The fastest machine for an operation is the one with the smallest sum of the durations for loading, processing, and unloading. This sum is what we call the *minimum duration of an operation*. The theoretical route duration of a lot is the sum of the minimum duration of all operations of its route plus the minimum time lags between them.

The *actual route duration* of a lot is the time between the initiation date of the lot and the completion date of the lot (end of unloading after its final operation). The *flow factor* of a lot is its actual route duration divided by its theoretical route duration. Now, the *weighted flow factor* that we want to minimize is the weighted average of all flow factors. This criterion is also known as *x-factor*. In opposition to move related criteria, this criterion is independent of the scheduling horizon. With L denoting the set of lots to schedule, the objective can be written as

$$\frac{1}{\sum_{l \in L} (\text{priority of } l)} \cdot \sum_{l \in L} \frac{(\text{priority of } l) \cdot (\text{actual route duration of } l)}{(\text{theoretical route duration of } l)}.$$

2.4.5 Time Lag Violation Cost

Lots that have not yet started the processing of a time lag triggering operation can always be scheduled without maximum time lag violations because the start of the constraint can still be delayed. However, an ongoing maximum time lag implies fixed due date for the final operation of the time lag. Since time lags might be nested or chained, lots with due dates induced by time lags can imply other unstarted time lags that might become unsatisfiable as a consequence. As a schedule has to be always determined, it is more cautious to transform these constraints to the objective of minimizing their violation. Another reason that motivates this choice is the fact that the duration of such constraints are only experimental estimates and their violation does not automatically lead to yield loss or lot scrapping.

The scheduler has to deal with the possibility of such time lag violations and, as specified in Section 2.3.5, violation costs are given. We distinguish *reworkable* time lags from non-reworkable time lags. Lots with violated reworkable time lags need to be reworked in case a time lag violation occurs. This imposes a *rework cost*. Lots with violated non-reworkable time lags have a defectivity risk that rises increasingly with the duration of the maximum time lag violation. Once a non-reworkable lot is defective, it must be scrapped, which induces a *scrap cost*. We assume that scrapping is inevitable once a certain violation duration is reached. Lots that must be reworked or scrapped remain unscheduled since we know that processing them is pointless.

Thus, for each maximum time lag, a *violation cost* k, a *maximum duration* d (a relative due date) and an *ultimate duration* γ (a relative deadline) are given. The ultimate duration must be equal to or greater than the maximum duration. For a completion time C of the considered operation, the violation severity of the time lag is specified as

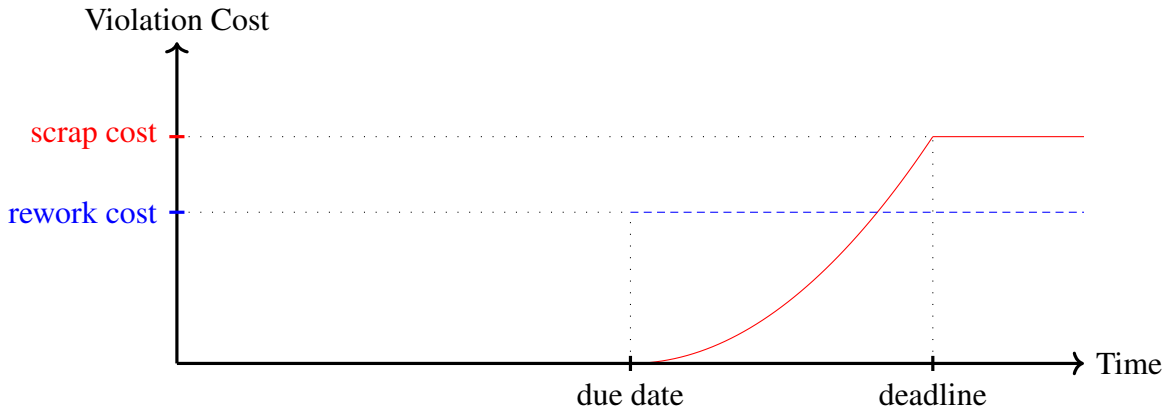


Figure 2.8 – *Time lag violation cost as a function of the completion date of a lot.*

$$\begin{cases} 0 & \text{if } C \leq d \\ k & \text{if } C > \gamma \\ k \cdot \frac{(C-d)^2}{(\gamma-d)^2} & \text{otherwise} \end{cases}$$

Figure 2.8 illustrates the violation cost as a function of the completion time of a lot. Maximum and ultimate durations are denoted as due date and deadline, since Figure 2.8 assumes that the time constraint has already started. For non-reworkable lots, the ultimate duration is greater than the maximum duration. For reworkable lots, the time lag maximum and ultimate durations are equal. We want to minimize the sum of all time lag violations over the maximum time lags of all scheduled operations.

2.4.6 Production Target Satisfaction

Considering production targets as constraints may be very restrictive. The quantities specified by these targets are given by an aggregated plan which makes them only estimations and ideal targets to achieve. It is possible to be in a situation where a target can never be fully satisfied by the available lots to schedule. As for the maximum time lags, the production targets are then transformed into an objective of maximizing their satisfaction. As move related objectives, the criterion that models the production target satisfaction should depend on the scheduling horizon.

Each production target k is associated with a *requested volume* $D_k \in \mathbb{N}$. Given a schedule, the *actual produced volume* for each target k in the specified planning horizon can be computed. Let $P_k \in \mathbb{N}$ be this quantity. Given these two parameters, we can compute the *completion rate* of the production target k as $X_k = \frac{P_k}{D_k}$. It is possible to have different levels of satisfaction for the same completion rate of two different production targets. A production target can define a minimum quantity to produce, or a quantity that is desirable to reach but not to exceed. So, instead of working with the completion rate X_k, the expected satisfaction

of the decision maker that depends on that completion rate can be modeled. Let Y_k be the expected satisfaction. As there could be a different level of satisfaction for the same completion rate depending on the production target, we consider for each production target k a function F_k that returns a satisfaction level for any completion rate. In the industrial application, as the targets only define a minimal quantity to produce, the satisfaction can be computed as shown below. Figure 2.9 illustrates the relation between the completion rate and the chosen satisfaction models.

$$F_k(X_k) = \begin{cases} X_k, & X_k \leq 1 \\ 1, & X_k > 1 \end{cases} \tag{2.2}$$

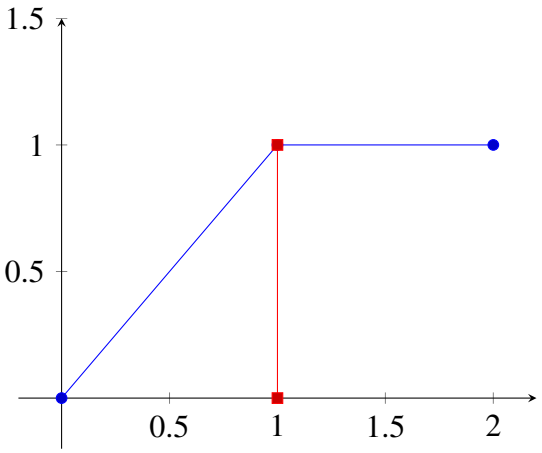


Figure 2.9 – *Minimal quantity and indifference to over-production case*

The chosen satisfaction model is just an example and more sophisticated ones can be defined. When the satisfaction level of a target is defined, it remains to define how to evaluate a solution based on the satisfaction of individual targets to compare different solutions. To define such an evaluation, we have to answer these questions: Is it preferable to maximize the average satisfaction level, even if this leads to penalize some production targets? Or is it preferable to minimize the dispersion between the satisfaction level of all the targets, even if this leads to producing less volumes?

2.5 Conclusion

This chapter provides a comprehensive definition of a complex scheduling problem arising from the diffusion area in semiconductor manufacturing. The objective of this thesis is to develop an approach that can handle all the features of the scheduling problem. To be feasible, the solution returned by the approach must satisfy a large number of constraints: job-related constraints such as routing and release dates; machine related constraints such as batching

capacities, qualifications, and availabilities; sequence-dependent setup times; minimum and maximum time lags. As the schedule is predictive, the operation times must be accurate as much as possible so that the predicted performances are not far from the actual performances of the implemented schedules. This can be achieved through accurate input data and adequate modeling. A significant challenge when modeling the practical scheduling problem arises when considering wet benches with complex behavior. Adequate modeling of these machines is a significant parameter when considering the perspective of industrializing the solution approach. Due to the stochastic and dynamic environment, the approach to be proposed must be integrated within a rolling horizon framework. As a consequence, the approach is called to solve a complex scheduling problem frequently, and every time there are significant changes within the area. Adding to this the large size of the problem instances, the efficiency of the approach appears as a critical criterion for possible industrialization.

Besides integrating all the constraints, the approach must propose solutions leading to high performance across the optimized area and contributing to the overall objectives at the fab level. In the studied context, as in most practical situations, different criteria must be optimized at the same time. In this chapter, relevant criteria that are defined in previous works are recalled: Weighted number of moves, batching coefficient, weighted flow factor and a criterion that is related to time lag violation. The definition of new criteria is motivated: Discounted weighted number of moves and a criterion that is related to production target satisfaction. The prime objective of this thesis is to design an approach capable of handling a significant number of criteria. An interesting perspective is to study the relation between the different criteria in order to define a minimal set of relevant ones. To deal with the different challenges, the proposed approach in this thesis is built upon the approach proposed by Knopp (2016) for solving complex job shop scheduling problems and the approach proposed by Bitar (2015) for handling the multiobjective aspect. The remaining of this report is devoted to the description of the proposed approach.

Chapter 3

Improvements of the Batch-Oblivious Approach

To solve the scheduling problem described in Chapter 2, the approach developed in this thesis is based on the batch-oblivious approach proposed in Knopp et al. (2017), which relies on a conjunctive graph where batching decisions are encoded in the arc weights. Along with this representation, an integrated construction algorithm is proposed that simultaneously computes start dates and improves the solution during the graph traversal. Not bound to one specific heuristic, the building blocks of the batch-oblivious approach can be applied within different heuristics. The objective of this chapter is to improve the efficiency and the effectiveness of the batch-oblivious approach. To evaluate the different propositions described in this chapter, the complex job-shop scheduling problem in Knopp et al. (2017) is solved. Section 3.1 provides the formal description of the problem, which considers a subset of the constraints defined in Chapter 2 and classical regular objective functions. Section 3.2 summarizes the different components of the original batch-oblivious approach. The framework that allows the graph to be modified during the start time computation while maintaining the feasibility of the solution is recalled and generalized in Section 3.3.1. Based on the results of Section 3.3.1, Section 3.3.2 describes the proposed ideas to improve a solution during the start time computation with a low computational cost. In Section 3.4, a new constructive algorithm is developed that intentionally inserts idle times in order to increase the size of the batches and to obtain more relevant solutions from an industrial perspective. Finally, using industrial instances, Section 3.6 assesses the improvements achived by our different propositions.

3.1 Formal Problem Description

This section provides a formal definition of the problem considered in Knopp et al. (2017). The characteristics defined in Chapter 2 and not considered here are: Multiple machines per operation, minimum and maximum time lags, availability constraints, minimum batch size constraints, control quality tasks, and production targets. The internal complexity of some machines is ignored by associating fixed processing times to operations. These characteristics are formalized and considered in Chapter 4. Also, only classical regular criteria are

considered in this chapter. The criteria that are specific to the industrial context are also formalized in Chapter 4. The problem considered here is a flexible job-shop scheduling problem with p-batching, reentrant flows, sequence-dependent setup times and release times (*complex job-shop scheduling problem*). Using the $\alpha|\beta|\gamma$ notation of Graham et al. (1979), this class of scheduling problems can be denoted as $FJc|r_j, s_{i,j}, B, recr|reg$.

A set of *jobs* $\mathcal{J}$ have to be processed using a given set of *machines* $\mathcal{M}$. Each job $j \in \mathcal{J}$ is associated to a set of *operations* $O_j = \{o_{1,j}, o_{2,j}, \ldots, o_{|O_j|,j}\}$, a *release time* $r_j \in \mathbb{Z}$. and a *size* $\sigma_j \in \mathbb{N}$. The disjoint union $\mathcal{O} = O_1 \dot{\cup} O_2 \ldots \dot{\cup} O_{|\mathcal{J}|}$ denotes the set of all operations. For a given set of *recipes* $\mathcal{R}$, each recipe $q \in \mathcal{R}$ prescribes a machine $m_q \in \mathcal{M}$, a *processing duration* $p_q \in \mathbb{N}_0$ and a *batching capacity* $b_q \in \mathbb{N}_{>0}$. As described in Section 2.3.3, a recipe specifies the process a job must undergo in a machine. Differently than in the industrial context, it is assumed that a recipe prescribes a unique machine. So, instead of associating a unique recipe to each operation that can be performed on multiple machines, each operation is associated with a subset of recipes $R_{i,j} \subset \mathcal{R}$. This modeling choice is motivated by the fact that the same recipe can prescribe different sets of qualified machines depending on the job (product). Let us define a mapping $f : \mathcal{O} \times \mathcal{M} \rightarrow \mathcal{R}$ that returns the required recipe $q \in \mathcal{R}$ by operation $o_{i,j} \in \mathcal{O}$ if it can be performed on machine $m \in \mathcal{M}$. A given mapping $s : \mathcal{R} \times \mathcal{R} \rightarrow \mathbb{N}_0$ prescribes *sequence-dependent setup times* between operations that are scheduled on the same machine. Let $O_m = \{o_{i,j} \in O \mid \exists q \in R_{i,j} \wedge m_q = m\}$ denote the set of operations that can be assigned to machine $m \in \mathcal{M}$.

A *schedule* is completely characterized by selecting recipes $q_{i,j} \in R_{i,j}$ and *start times* $S_{i,j} \in \mathbb{Z}$ for all operations $o_{i,j} \in O$. We denote the machines, processing durations and batching capacities related to this selection as $m_{i,j}$, $p_{i,j}$ and $b_{i,j}$, respectively. A completely characterized schedule is obtained after the partition of all operations into batches, the assignment of the formed batches to qualified resources, their sequencing and finally the assignment of start times. The three first decisions can all be represented by a family of batches $\mathcal{B} = \{B_{m,x}\}_{m \in \mathcal{M}, x \in \{1,\ldots,t_m\}}$, where $B_{m,x}$ is the batch sequenced at position x on machine m and $t_m \in \{0, \ldots, |O_m|\}$ the number of batches assigned to machine m. As only operations with the same recipe can be processed in the same batch, q_B denotes the associated recipe to the batch $B \in \mathcal{B}$, i.e., $q_B = q_{i,j}\ \forall o_{i,j} \in B$. Let $\sigma_B = \sum_{o_{i,j} \in B} \sigma_j$ denote the size of the batch B.

To describe a *feasible* schedule , selected recipes $q_{i,j}$ and start times $S_{i,j}$ of operations $o_{i,j}$ have to respect several constraints that are detailed in the following. Preemption is not allowed: Once the processing of operation has begun, it cannot be interrupted. Thus, the *completion time* of an operation $o_{i,j} \in O_j$ is given by $C_{i,j} = S_{i,j} + p_{i,j}$. Operations belonging to the same job have to be performed in the order given by the *route* of the job. So, $C_{i,j} \leq S_{i+1,j}$ has to be fulfilled for all $o_{i,j} \in \mathcal{O}$ with $i < \left|O_j\right|$. The first operation $o_{1,j} \in O_j$ of each job cannot be processed before its release time, so $S_{1,j} \geq r_j$ must hold for all $j \in \mathcal{J}$. Operations performed on the same machine must not overlap. Hence, for two operations $o_{i,j}$ and $o_{k,l} \in \mathcal{O}$ with $m_{i,j} = m_{k,l}$, either $S_{i,j} = S_{k,l}$ or $S_{i,j} \geq C_{k,l}$ or $C_{i,j} \leq S_{k,l}$ must hold.

Regarding batching constraints, only operations of the same family can be processed at the same time on the same machine. So, for two operations $o_{i,j}$ and $o_{k,l} \in \mathcal{O}$ with incompatible

families $q_{i,j} \neq q_{k,l}$ and $m_{i,j} = m_{k,l}$, $S_{i,j} \neq S_{k,l}$ is required. Any formed batch $B \in \mathcal{B}$ must respect the batching capacity of the machine to which it is assigned. Thus, $\sigma_B \leq b_B \; \forall B \in \mathcal{B}$ is required. To respect sequence-dependent setup times, for all operations $o_{i,j}$ and $o_{k,l} \in O$ with $m_{i,j} = m_{k,l}$ and $S_{i,j} \neq S_{k,l}$, either $C_{i,j} + s(q_{i,j}, q_{k,l}) \leq S_{k,l}$ or $C_{k,l} + s(q_{k,l}, q_{i,j}) \leq S_{i,j}$ must hold.

With the different constraints being satisfied, the objective is to optimize given criteria. A feasible schedule is completely defined when recipes $q_{i,j} \in R_{i,j}$ are selected and, start times $S_{i,j}$ and completion times $C_{i,j}$ are determined for all operations $o_{i,j} \in O$. The completion time of job $j \in J$ is the completion time of its last operation, i.e., $C_j = C_{|O_j|,j}$. The quality of a schedule is measured by an objective function, which is a function $f : \mathbb{R}^{|O|} \rightarrow \mathbb{R}$ that maps tuples of operation start times to a real number. In the scope of this chapter, we want to optimize objective functions that are *regular* (Brucker (2007)). With this type of objective functions, the quality of a schedule cannot deteriorate by advancing the start times of some of its operations. In other words, when considering regular objective functions, there always exists a left-justified schedule that is optimal. More formally, for any pair of tuples of start times $(S_1, \ldots, S_{|O|}), (S'_1, \ldots, S'_{|O|}) \in \mathbb{R}^{|O|}$ with $S_1 \leq S'_1 \wedge \cdots \wedge S_{|O|} \leq S'_{|O|}$, it follows that $f(S_1, \ldots, S_{|O|}) \leq f(S'_1, \ldots, S'_{|O|})$. Most of the papers in the literature deal with regular functions (see e.g. Mati et al. (2011), García-León et al. (2015)). The makespan, the total weighted completion time, the total weighted tardiness or the maximum lateness are examples of well-known regular functions. To consider some of these functions, each job $j \in \mathcal{J}$ also has a due date $d_j \in \mathbb{Z}$ and a weight $\omega_j \in \mathbb{R}$. In the numerical results of Section 3.6, the total weighted completion time and the total weighted tardiness are used.

3.2 Recalling the Batch-Oblivious Approach

Most existing solution approaches for Complex Job-Shop scheduling problems with batching machines rely on the disjunctive graph representation of Ovacik and Uzsoy (2012). This representation introduces dedicated nodes to represent explicitly batching decisions. A novel batch-oblivious modeling is introduced by Knopp et al. (2017). Like a classical conjunctive graph, the batch-oblivious conjunctive graph uses nodes to uniquely model operations and arcs to model precedence constraints on routes and resources. Instead of using additional nodes and arcs to model batches, batches are coded in the arc weights. This new representation has many advantages. It reduces the structural complexity of the graph and allows reusing ideas and techniques for a less complex problem like the move proposed by Dauzère-Pérès and Paulli (1997) for the flexible job-shop scheduling problem. Last but not least, it is possible to propose an integrated algorithm that computes start times and improves the solution during the graph traversal by filling underutilized batches through a combined resequencing and reassignment strategy. In this work, we adopt the same representation in the form of batch-oblivious modeling. We propose a new integrated algorithm that modifies the solution differently during graph traversal.

3.2.1 Batch-Oblivious Conjunctive Graph

A classical *conjunctive graph* $G = (V, E)$ is an acyclic directed graph with nodes $V = \mathcal{O} \cup \{0, *\}$ that correspond to the operations in $\mathcal{O}$ plus an artificial start node 0 and an artificial end node $*$. For a node $v \in \mathcal{O}$, we denote its *route successor* by $r(v) \in V \setminus \{0\}$ and its *machine successor* by $m(v) \in V \setminus \{0\}$. Analogously, its predecessors are denoted by $r^{-1}(v) \in V \setminus \{*\}$ and $m^{-1}(v) \in V \setminus \{*\}$. The artificial start node 0 has $|\mathcal{J}|+|\mathcal{M}|$ outgoing edges and no incoming edges. Analogously, the artificial end node $*$ has $|\mathcal{J}| + |\mathcal{M}|$ incoming edges and no outgoing edges. This graph can be used to determine start times S_v of operations $v \in \mathcal{O}$. A weight $l_{u,v} \in \mathbb{N}_0$ is assigned to each edge $(u, v) \in E$ in order to ensure a minimum duration between the beginning of adjacent operations: $S_v \geq S_u + l_{u,v}$ for each edge $(u, v) \in E$. Having this, start times of operations correspond to distances of longest paths from the artificial start node. Let us denote by $L(v, w) \in \mathbb{N}_0$ the distance of a longest path from node $v \in V$ to node $w \in V$. For each operation $v \in \mathcal{O}$, its start time is determined by $S_v = L(0, v)$. To reflect the constraints, we define edge weights as follows. For edge $(0, o_{1,j}) \in E$ connecting the artificial start node 0 with the initial operation $o_{1,j}$ of a job $j \in \mathcal{J}$, the edge weight is set to the release time r_j of job $j \in \mathcal{J}$. For edge $(0, o_m) \in E$ connecting the artificial start node 0 with the initial operation o_m scheduled on machine $m \in \mathcal{M}$, the edge weight is set to zero. For route edge $(v, r(v)) \in E$ with $v \neq 0$, the edge weight is set to the processing duration p_v of operation v. For machine edge $(v, m(v)) \in E$ with $v \neq 0$ of non-batching machines, the edge weight is set to the sum $p_v + s(q_v, q_{m(v)})$ of the processing duration of v and the sequence-dependent setup time between v and $m(v)$ on machine $m_v = m_{m(v)}$.

While the representation provided by Ovacik and Uzsoy (2012) introduces dedicated nodes to represent explicitly batching decisions, the approach proposed by Knopp et al. (2017) is not intrusive of the classical conjunctive graph and models batching decisions by only adapting the weights of resource edges $(u, v) \in E$, i.e., such that $m(u) = v$. The weight of a machine edge is set to zero if its adjacent operations should be processed in the same batch. Otherwise, the edge weight is set to $p_u + s(q_u, q_v)$, as in the non-batching case. However, setting $l_{u,v} = 0$ only guarantees that $S_u \leq S_v$ but not that $S_u = S_v$, which must be satisfied if p-batching constraints are considered. To make sure that batching decisions are feasible, an *invariant*, given in (3.1), must be satisfied when the possibility of batching two adjacent operations u and v is studied. Using this invariant, it follows that, for each operation $u \in V$, computing the longest path lead to scheduling the machine successor operation $v = m(u)$ either at the same time as u or at a later point in time where processing durations and sequence-dependent setup times are satisfied. This property propagates naturally: Multiple operations belonging to the same batch are connected in a path of zero weighted machine edges. In the remaining of this chapter, when batching an operation v with its predecessor u is possible, we assume that the following two conditions are already verified: $q_u = q_v = q,\ q \in \mathcal{R}$ and, if B is the batch that contains u, $\sigma_B + \sigma_v \leq b_q$.

$$\left(l_{u,v} = 0 \ \wedge \ S_u \geq S_{r^{-1}(v)} + l_{r^{-1}(v),v}\right) \ \vee \ (l_{u,v} = p_u + s(u, v, m_u)) \tag{3.1}$$

The invariant (3.1) can be interpreted as follows: An operation v can be batched with

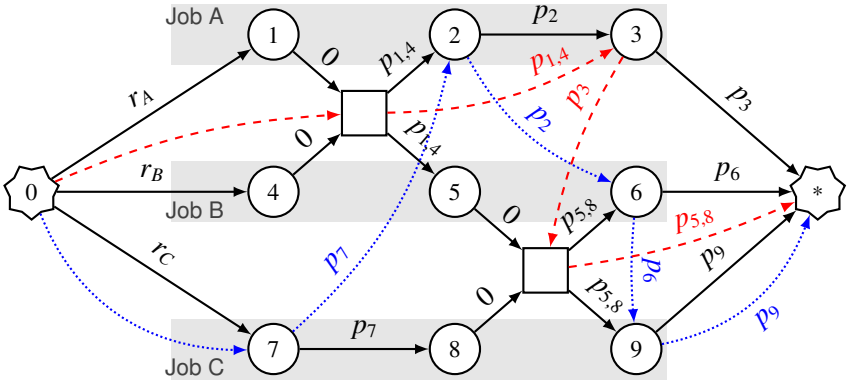
its resource predecessor u, if the job of v is available before the already determined start time of u, i.e., $S_{r^{-1}(v)} + l_{r^{-1}(v),v} \leq S_u$. In the original approach, the start time of u is never changed while it is proposed in Section 3.4 to recompute S_u so that it can be batched with v when it is relevant. As shown in Section 3.3.1, another important ingredient of the approach proposed in Knopp et al. (2017) is the dynamic modification of the conjunctive graph during the schedule construction, by advancing suitable nodes, in order to fill-up incomplete batches. Before going forward, most notations that are used within this chapter are listed in Table 3.1

Figure 3.1 shows an example to compare batch-aware and batch-oblivious representations. It shows a schedule with three jobs A, B, and C using two machines. We see two batches processed on machine 2, each consisting of two operations: The first batch with operation 1 and operation 4, and the second batch with operation 5 and operation 8. For brevity of notation, sequence-dependent setup times have been omitted and let us denote $p_{1,4} = p_1 = p_4$ and $p_{5,8} = p_5 = p_8$. Note that invariant (3.1) is not visualized in Figure 3.1 (b), so we assume that $S_1 \geq r_B$ and $S_5 \geq S_7 + p_7$.

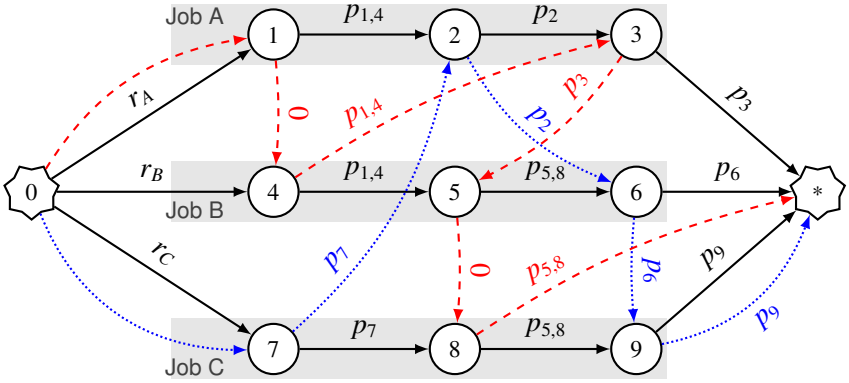
3.2.2 Adaptive Computation of Start Times

In order to develop heuristic algorithms to solve the considered problem, the move proposed by Dauzère-Pérès and Paulli (1997) for the flexible job-shop scheduling problem is adapted to modify a given batch-oblivious conjunctive graph. This move, which integrates the resequencing and reassignment of operations, does not require any specific knowledge of previous batching decisions as it only considers a single operation at a time. In addition to the improvement of the solution quality within the chosen heuristic by applying the move of Dauzère-Pérès and Paulli (1997), it is proposed to improve batching decisions during the computation of start times. When traversing the graph to compute the start times, suitable nodes are advanced by removing and reinserting them in the graph to "fill up" incomplete batches.

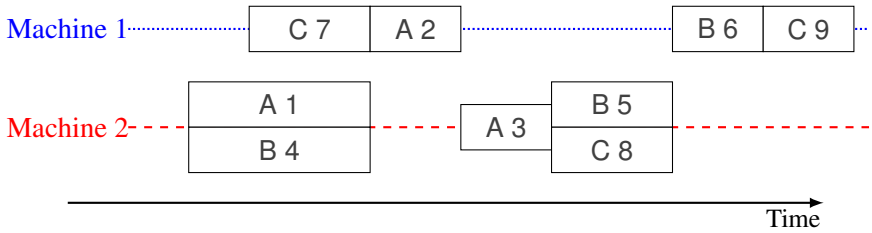
When the graph is not modified, it is possible to compute first a topological ordering. Then, all nodes are traversed in this order to compute start times and to take batching decisions regarding adjacent nodes only. This approach is no longer viable if the graph is modified while being traversed. To be effective, three main concerns must be addressed when improving a solution during the computation of start times: Feasibility of the solution, efficient search for potential candidates to complete batches and suitable choice of the candidates. The framework proposed by Knopp et al. (2017) to ensure the feasibility of the solution while it is dynamically modified is recalled and generalized in Section 3.3.1. The generalized framework helps to increase the efficiency of the search, and supports the integration within the batch-oblivious approach of some constraints as shown in Chapter 4. Section 3.3.2 recalls the different strategies proposed by Knopp et al. (2017) to choose the candidates to advance and describes new efficient strategies. The strategies presented in Section 3.3.2 are called *non-delay* as they only select candidates that can be included in incomplete batches without delaying their start times. Section 3.4 proposes new strategies to select operations that may require the delay of batch start times.



(a) *Batch-Aware Conjunctive Graph*



(b) *Batch-Oblivious Conjunctive Graph*



(c) *Gantt Chart*

Figure 3.1 – *A comparison of alternative representations of the same schedule (Knopp (2016))*

Table 3.1 – *Table of Notation*

$\mathcal{J}$	Set of jobs
$\mathcal{M}$	Set of machines
$\mathcal{O}$	Set of all operations
$\mathcal{R}$	Set of all recipes
$\mathcal{B}$	Family of batches
V	Nodes of batch-oblivious conjunctive graph ($V = \mathcal{O} \cup \{0, *\}$)
O_j	Operations of job j
O_m	Set of all operations that can be processed on machine m
r_j	Release time of job j
ω_j	Weight of job j
d_j	Due time of job j
$o_{i,j}$	Operation
$u_\square$,$v_\square$, $w_\square$	Nodes of graph G
$B^\square$	Batch
$m_\square$	Machine, if indexed, the index refers to an operation $o_{i,j} \in \mathcal{O}$, node $v \in V$ or batch $B \in \mathcal{B}$ that are using this machine
$q_\square$	Recipe, if indexed, the index refers to an operation $o_{i,j} \in \mathcal{O}$, node $v \in V$ or batch $B \in \mathcal{B}$
$p_\square$	Processing duration, if indexed, the index refers to a recipe $q \in \mathcal{R}$, an operation $o_{i,j} \in \mathcal{O}$, node $v \in V$ or batch $B \in \mathcal{B}$
$\sigma_\square$	Size of a job $j \in \mathcal{J}$, a job of a node v or an operation $o_{i,j}$, or a batch $B \in \mathcal{B}$
$l_{u,v}$	Weight of the edge between two nodes u and v in the batch-oblivious conjunctive graph G
$S_\square$, $C_\square$	Start and completion times, the index may refer to an operation $o_{i,j} \in \mathcal{O}$, node $v \in V$ or batch $B \in \mathcal{B}$
$t_\square$	Job availability of an operation $o_{i,j} \in \mathcal{O}$ or a node $v \in V$ ($t_v = S_{r^{-1}(v)} + l_{r^{-1}(v),v}$)
V^s	Set of the settled nodes
V^u	Set of the unsettled nodes
V^f	Set of the first job unsettled nodes
E^0	Set of non-delaying candidates to fill an incomplete batch
E^∞	Set of delaying candidates to fill an incomplete batch

3.3 Search Acceleration and New Search Strategies

This section introduces our different propositions to make the batch-oblivious approach more efficient. Section 3.3.1 recalls how a solution is kept feasible while it is improved during the computation of start times. Then, some properties are generalized, and new ones are highlighted. Based on the highlighted properties, Section 3.3.2 proposes new efficient strategies for node selection. The generalization proposed in this section are also necessary for the integration of new constraints, such as the minimum batch size, studied in Section 4.3.

3.3.1 Dynamic graph modification: Solution feasibility

It is proposed to improve the solution while a schedule is computed dynamically. To obtain a feasible solution, these modifications must respect all the constraints and ensure that no cycle is introduced in the graph. To do so, let us consider a batch-oblivious conjunctive graph $G = (V, E)$ to be used to compute a schedule: Consider feasible batching decisions by making the weights of related edge resources equal to zero and compute the start times of operations. During such algorithm, let us define *settled* nodes to be those for which start times are already computed and *unsettled* nodes those for which this is not yet done. Let us define a cut $C = (V^s, V^u)$ that is a partition of V of the graph G, where V^s is the set of all settled nodes, and V^u is the set of all unsettled nodes. To compute the start time of node $v \in V$, the start times of its predecessors must already be computed, i.e., they must already be settled. To ensure this, it is required that there is no edge from an unsettled node to a settled node. In this case, i.e. $E \cap (V^u \times V^s) = \emptyset$, we call the cut C *unidirectional*.

Let us denote by $G^s = (V^s, E^s)$ and $G^u = (V^u, E^u)$ the resulting subgraphs. The edges of each graph $G^\star \in \{G^s, G^u, G\}$ are given by $E^\star = E \cap (V^\star \times V^\star)$. Let us denote for a node $v \in V^\star$ its indegree in $G^\star$ by $deg^-_\star(v)$ and its outdegree in $G^\star$ by $deg^+_\star(v)$. A node $v \in V^\star$ without incoming edges (i.e. $deg^-_\star(v) = 0$) is called a *root node* of $G^\star$. A node $v \in V^\star$ without outgoing edges (i.e. $deg^+_\star(v) = 0$) is called a *leaf node* of $G^\star$. A settled node $v \in V^s$ that has its resource successor unsettled, i.e., $m(v) \in V^u$, is called *last machine settled node*. The set $V^l \subseteq V^s$ denotes the set of all last machine settled nodes and its cardinality is at most the number of machines $|M|$. Note that the set of leaf nodes in V^s is a subset of V^l. An unsettled node $v \in V^u$ that has its route predecessor settled, i.e., $r^{-1}(v) \in V^s$, is called *first job unsettled node*. Such a node can be either a root node in V^u or not. The set $V^f \subseteq V^u$ denotes the set of all first job unsettled nodes and its cardinality is at most the number of jobs $|J|$. Note also that the set of root nodes in V^u is a subset of V^f.

Figure 3.2 illustrates a partition of the nodes of the same batch-oblivious conjunctive graph in Figure 3.1. C represents the unidirectional cut. The green filled nodes are settled nodes, i.e., $V^s = \{0, 1, 2, 4, 7\}$. The grey filled nodes are unsettled nodes, i.e., $V^u = \{3, 5, 6, 8, 9, *\}$. The green filled nodes that are outlined in black correspond to the last machine settled nodes, i.e., $V^l = \{2, 4\}$. The grey filled nodes that are outlined in black correspond to the first job unsettled nodes, i.e., $V^f = \{3, 5, 8\}$.

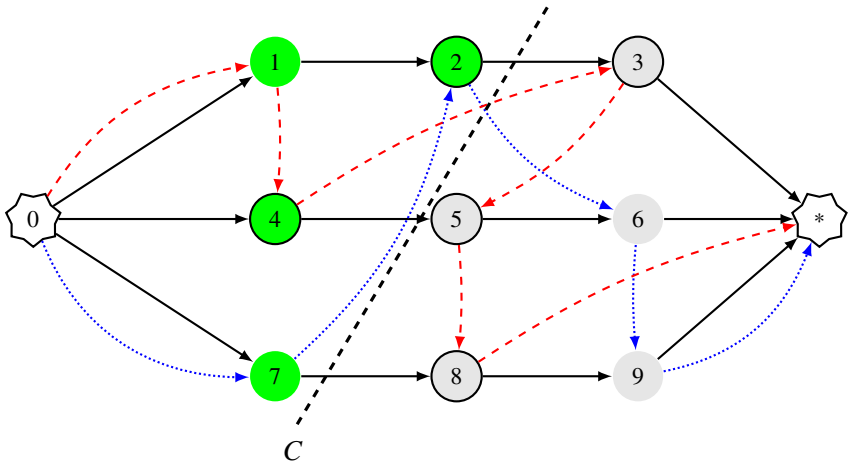


Figure 3.2 – *Partition of batch-oblivious conjunctive graph nodes*

To *settle* a first job unsettled node $w \in V^f$ after a last machine settled node $u \in V^l$ of G^s, w is removed from G^u and appended to G^s. If $m^{-1}(w) = u$, the operation remains assigned to machine m_w and sequenced after the same machine predecessor w. In this case, no edges need to be modified. Otherwise, if $m^{-1}(w) \neq u$, we modify the graph G as follows: The machine related conjunctive edges $(m^{-1}(w), w) \in E$ and $(w, m(w)) \in E$ of operation w are replaced by an edge $(m^{-1}(w), m(w))$ and the edge $(u, m(u)) \in E$ is replaced by two edges (u, w) and $(w, m(u))$. Settling a node does not change any route edge. If $m^{-1}(w) \neq u$ and $m_w = m_u$, then w is *resequenced*. If $m^{-1}(w) \neq u$ and $m_w \neq m_u$, then w is *reassigned*. Note that we require for a node $v \in V^u$ to be reassigned after a node $w \in V^s$ such that $\exists\, q \in R_w$ with $m_q = m_u$. After these graph modifications, the weight of edge (u, w) is determined in a way that satisfies the batching constraints and the invariant (3.1). Finally, the start time of w can be computed as its predecessors are already settled and the weight of the resource edge (u, w) is determined.

In Knopp et al. (2017), it is proved that, after settling a first job unsettled node w after a leaf node $u \in V^s$, the modified graph $G' = (E', V')$ does not contain any cycle and $C' = (V^{s'}, V^{u'})$, where $V^{s'} = V^s \cup \{w\}$ and $V^{u'} = V^u \setminus \{w\}$, is a unidirectional cut in G'. In this work, instead of restricting to the set that contains only leaf nodes in V^s, it is allowed to settle a node w after any last machine settle node $u \in V^l$. Theorem 3.1 ensures the previous result and that no cycle in introduced in the modified graph G' and that $C' = (V^{s'}, V^{u'})$ is a unidirectional cut in G'. The proof is not detailed here as the same arguments given in Knopp et al. (2017) can be used.

Theorem 3.1. *Let $G = (V, E)$ be a conjunctive graph and let $C = (V^s, V^u)$ be a unidirectional cut in G. When a first job unsettled node $w \in V^f \subseteq V^u$ is settled after a last machine settled node $u \in V^l \subseteq V^s$ of G^s, the modified graph $G' = (V', E')$ does not contain any cycle and $C' = (V^{s'}, V^{u'})$ is a unidirectional cut in G'.*

In the original approach, the start time of a settled node is never changed. In other words, when the start time of an operation u is computed, it is assumed that no decision related to

this operation, such as its assignment or its sequencing, is ever revised. Different reasons can motivate the generalization of the framework to reconsider already taken decisions. In this thesis, the possibility of reconsidering decisions that are related to settled nodes has two motivations:

- In the original approach, if v is the selected operation to be settled after operation u such that $S_{r^{-1}(v)} + l_{r^{-1}(v),v} > S_u$, u and v will be in different batches as invariant (3.1) is not satisfied. In Section 3.4, different approaches are proposed to allow including an operation v in an incomplete batch, even when the job of v is available after the already computed start time of the batch. However, to satisfy the p-batching constraint that imposes a common start time for all operations in a batch, it becomes necessary to recompute the start times of the operations already in the incomplete batch so that the selected late operation can be included in the batch.
- The other situation where it becomes necessary to reconsider already taken decisions that are related to settled nodes is when taking into account minimum batch size constraints. Our proposed approach to consider these constraints is described in Section 4.3.4. It requires the possibility of resequencing already settled nodes if their batches do not satisfy minimum batch size constraints.

To cope with these different situations, we extend the approach by giving the possibility to *unsettle* a node $v \in V^s$. First, let us define a *reachability relation* as $\prec \subset V \times V$, which contains $(u, v) \in V \times V$ if and only if there exists a path from u to v in the conjunctive graph G. To unsettle a node $u \in V^s$, u is removed from G^s and appended to G^u. Also, any node $v \in V^s$ such that $u \prec v$ must be unsettled. By performing this routine, it is ensured that the validity of the unidirectional cut is maintained. Regarding the feasibility of the solution, the risk of introducing a cycle depends on the changes made to the graph, and must be studied on a case-by-case basis.

3.3.2 Node Selection Strategies

To compute the start times and determine the weight of resource edges, the nodes of the graph are traversed in the topological order. Let v be the currently visited node. Before settling it and computing its start time, the possibility of batching it with its machine predecessor $u = m^{-1}(v)$ is checked. If the batch B containing u is complete, v is directly settled after u. If the batch is incomplete and it is possible to add v, v is also settled after u and the weight of edge $l_{u,v} = 0$. If the two previous conditions are not satisfied, it can be interesting to look for other nodes that can be advanced in order to complete batch B. In the original work of Knopp et al. (2017), it was shown that the quality of the obtained schedule strongly depends on the selection of the nodes to settle. If the batch containing $u \in V^s$ is incomplete, there may be nodes in V^f that can fill the unused capacity. The selection strategies are different in the way V^u is explored. In Knopp et al. (2017), after selecting a node $v \in V^u$, one of three strategies can be used to determine a node w to be settled after $u = m^{-1}(v)$. Let E^0 be the set of nodes found by a given strategy to fill an incomplete batch.

- A *static strategy* settles v after its resource predecessor, i.e., $E^0 = \emptyset$. This strategy does not modify the graph and iterates the nodes in the topological order, as any classical start time computation algorithm.

- Using a *resequencing strategy*, if the batch containing u is incomplete and v does not satisfy all the conditions to complete the batch, the machine successors of v are iterated until a candidate operation is found. The search is stopped at the first found compatible node. In this case, $E^0 = \{w \in V^u \wedge q_u = q_w \wedge S_{r^{-1}(w)} + l_{r^{-1}(w),w} \leq S_u\}$ and $|E^0| = 1$.

- If $E^0 = \emptyset$, a third strategy, called *reassignment strategy*, can be triggered. Using this last strategy, the search is extended to operations on other machine sequences. Instead of visiting all machine sequences as in the original approach, it is sufficient to restrict the search to any machine m such that $\exists R_{i,j} \supset \{q, q'\}$, where $q = q_u$ and $m_{q'} = m$. The search is also stopped at the first found candidate node. In this case, $E^0 = \{w \in V^u \wedge \exists q \in R_w \mid q = q_u \wedge m_u \neq m_w \wedge S_{r^{-1}(w)} + l_{r^{-1}(w),w} \leq S_u\}$ and $|E^0| = 1$.

If none of the used strategies find a candidate, node v is settled after its predecessor u. The complexity analysis of these strategies shows that, in the worst case, both the resequencing and the reassignment strategies explore $O(|V|)$ operations to select a node. Consequently, the runtime bound of the whole schedule computation algorithm is $O(|E||V|)$. As shown in the experimental results of Knopp et al. (2017), the search strategies lead to better solutions even if the computational cost is high. When they are triggered, these strategies search for filling nodes among all those belonging to V^u. However, only the nodes that have their route predecessors settled are potential candidates. Hence, instead of looking in V^u, it is sufficient to only look in its subset V^f defined in Section 3.3.1. So, instead of exploring the graph, we propose to maintain the set V^f in an auxiliary data structure. This data structure maps each job j to its first unsettled node $v \in V^f$. Whenever a node v is settled, its route successor $r(v)$ becomes the first unsettled node of the same job, which is done in constant time. When the set V^f is tracked, faster strategies can be proposed. These new strategies, as the search is only performed in V^f and as $|V^f| = |\mathcal{J}|$, explore at most $O(|\mathcal{J}|)$ nodes. Consequently, the runtime bound of the whole schedule computation algorithm is $O(|E||\mathcal{J}|)$. The search can be made faster with some preprocessing. Instead of considering all nodes in V^f, one can only focus on the subset of the first job unsettled node of jobs that have in their routing an operation that has among its eligible batch families the batch family of the batch to be completed. In other words, if q_B is the recipe of the batch to complete, the search is performed on the reduced subset $V^r \subset V^f$ such that $V^r = \{v \in V^f \mid v \in O_j \wedge \exists o_{i,j} \in O_j \wedge q_B \in R_{i,j}\}$. A simple preprocessing allows this by mapping each batch family to the set of jobs that have at least one operation that uses q_B.

The first proposed strategy is called the *integrated strategy*. If the batch containing u is incomplete and v does not satisfy all the conditions to complete the batch, the nodes in V^r are iterated. This strategy is called integrated as it does not differentiate between resequencing and reassignment. The nodes in V^r are iterated looking for a candidate that satisfies all the conditions. If such a node is found, the search is stopped. In this case, $E^0 = \{w \in V^r \wedge \exists q \in$

$R_w \mid q = q_u \wedge S_{r^{-1}(w)} + l_{r^{-1}(w),w} \leq S_u\}$ and $|E^0| = 1$. If none of the used strategies find a candidate, node v is settled after its predecessor u. In this strategy, the set V^r is always traversed in the same order.

In all previous strategies, if there is no node w such that $S_{r^{-1}(w)} + l_{r^{-1}(w),w} \leq S_u$, node v is settled after its resource predecessor u. We propose to relax this condition and potentially accept to integrate late operations in incomplete batches, at the cost of recomputing the start times of the operations in the incomplete batch. Also, within the strategies described above, the search is stopped when one candidate is found. If, after adding w to the batch of u, the batch is still incomplete, the search is triggered again and run through almost the same search space, i.e., $V'_f = V^f \setminus \{w\} \cup \{r(w)\}$. Instead of this, the search for all possible candidates that can complete the batch can be performed once. This strategy is motivated by the fact that late operations can be added to incomplete batches. If late operations are accepted and if the first encountered operation during the search is late, the batch start time may be delayed while there are operations that can be added without any delay on the initial batch start time.

The proposed strategy is called the *collecting strategy*. This strategy iterates nodes in V^r for operations that can be integrated into the incomplete batch without delaying its start time. Instead of stopping the search when a single node is found, the search is ongoing as long as the maximum capacity is not reached. The set of non-delaying operations is $E^0 = \{w \in V^r \mid \exists q \in R_w \wedge q = q_u \wedge S_{r^{-1}(w)} + l_{r^{-1}(w),w} \leq S_u\}$. As delaying operations are also potential candidates, in addition to set E^0, let $E^\infty = \{w \in V^r \mid \exists q \in R_w \wedge q = q_u \wedge S_{r^{-1}(w)} + l_{r^{-1}(w),w} > S_u\}$ be the set of delaying operations found during the search. As the objective is to search for a set of candidates instead of only one, we look indifferently operations to resequence or to reassign. If B is the incomplete batch to which u belongs, the search continues as long as there is still unused capacity, i.e., $\sigma_B + \sum_{w \in E^0} \sigma_w < b_q$. At the end of the search, if adding the nodes in E^0 results in a complete batch or if $E^0 \neq \emptyset$ and $E^\infty = \emptyset$, these nodes are settled one by one after u. If the two sets E^0 and E^∞ are empty, v is settled after its resource predecessor u. Finally, when the nodes in E^0 do not complete batch β and if E^∞ is not empty, the batch size can potentially increase if the late operations are added by delaying the start time of the whole batch. While doing this, an important question arises: What is the latest start time beyond which it is no longer interesting to try to increase the batch size. The proposed answer is detailed in Section 3.4.1 and consists in selecting a subset of E^∞.

The third proposed strategy is the *random integrated strategy*. Instead of traversing the element of V^r in the same order, this strategy searches for all non-delaying nodes E^0 as in the collecting strategy. At the end of the search, one node is randomly selected. The three proposed approaches are similar to the reassignment strategy as they visit all potential nodes. The resequencing strategy, by visiting only nodes on the sequence of the machine on which u is processed, only considers a subset of all the potential nodes. The experimental results showing the improvement brought by the new proposed strategies over the former ones are detailed in Section 3.6.1.

3.4 Active Scheduling Approaches

The way schedules are constructed in the original approach is described here as *locally non-delay*. When a node is visited during the graph traversal, the construction algorithm gives this node the earliest start time. The last condition only defines what is called semi-active schedules in the scheduling literature (Pinedo (2016)). Locally non-delay construction describes the way it is ensured that the invariant (3.1) is satisfied and the set of nodes that are candidates to be advanced. Regarding the satisfaction of the invariant that ensures the feasibility of batching decisions, operation v can be batched with its resource predecessor u if the job is available before the already determined start time of u, i.e., $S_{r^{-1}(v)} + l_{r^{-1}(v),v} \leq S_u$. In other words, operation v can be batched with its resource predecessor u only if it does not delay the start of the batch to which u belongs. However, it is still possible to ensure the satisfaction of the invariant by recomputing the start time of u and delaying the start time of the batch to include operation v. Also, in the original approach, the search for operations that can be advanced to complete a batch is restricted to compatible operations that are available before the already computed start time of the batch. Instead of this, the search can also be extended to operations that can potentially delay the start time of the incomplete batch.

Theoretically, it is always possible to construct an optimal solution using the original construction algorithm on the batch-oblivious conjunctive graph. A new construction algorithm is proposed in this work that is called *locally active* as idles times are intentionally added in order to complete batches. This was felt like a useful shortcut that accelerates the search by quickly constructing good solutions. Instead of constraining operation v to be available before the start time of its resource predecessor u in order to ensure the satisfaction of the invariant, it may be interesting to delay and recompute the start times of all the operations belonging to the same batch as u. Also, instead of restricting the search to operations that can complete a batch without delaying its start time, we extend the search for any operation that can be added to the batch without delaying its start time by more than a given maximal delay. By doing this, several questions arise: What is the maximal acceptable delay? If the start time of an incomplete batch is to be delayed, do we recompute only the start times of the operations already in the batch or do we sequence the delaying operation at the beginning of the batch sequence?

3.4.1 Computation of the Maximal Delay

If a batch B is still incomplete, the collecting strategy defined in Section 3.3.2 is triggered. The search returns two sets: $E^0 = \{w \in V^r \wedge \exists q \in R_w \mid q = q_u\}$ and $E^\infty = \{w \in V^r \wedge \exists q \in R_w \mid q_u = q \wedge S_{r^{-1}(w)} + l_{r^{-1}(w),w} > S_u\}$. If the batch remains incomplete after inserting all the nodes in E^0, it may be beneficial to insert nodes in E^∞. Adding all the delaying operations can be extreme. Different rules can be designed to decide which delaying nodes to integrate to the incomplete batch. Let us consider B^0 the incomplete batch obtained after including all the nodes in E^0 and let us denote its earliest start time by S, i.e., $S = S_u, \forall u \in B^0$. The recipe of the incomplete batch B^0 is q, its size is σ_{B^0} and its maximal size is b_q. Let t_v denote

the job availability time of node v, i.e., $t_v = S_{r^{-1}(v)} + l_{r^{-1}(v),v}$. Let us assume, w.l.o.g, that the nodes in E^∞ are ordered in the non-decreasing order of their job availability time, i.e., $\forall w_i \in E^\infty \mid 1 < i < |E^\infty|, t_{w_{i-1}} \leq t_{w_i} \leq t_{w_{i+1}}$. The question of whether to start B^0 as it is or to include nodes from E^∞ arises. Let B^l denote a formed batch that contains B^0 and all operations in E^∞ that are available before t_{v_l}, i.e., $B^l = B^0 \cup \{v_i \in E^\infty, i \leq l\}$ and let S_l be the time beyond which it is no longer beneficial to delay the start time of batch B^l. Finally, let B^f denotes the final batch.

The first rule, called *Simple Rule* and abbreviated by *SR*, relies on the idea that the acceptable delay must depend on the batch filling, i.e., the larger the size of the incomplete batch, the shorter must be the delay. The computation of the time beyond which it is not beneficial to delay the start time S of the batch B^0, denoted by S_0, is shown in (3.2). The delay is proportional to the processing duration and to the proportion of the incomplete size. When S_0 is computed, all the delaying nodes that are available before this time are added to the batch, i.e., $w_i \in E^\infty | t_{w_i} < S_0$. This rule gives as a final batch $B^f = B^0 \cup \{w_i \in E^\infty | t_{w_i} < S_0\}$.

$$S_0 = S + p_q \frac{b_q - \sigma_{B^0}}{b_q} \tag{3.2}$$

The idea of Cigolini et al. (2002) inspires the second rule for dynamic scheduling on batching machines. This rule is called *Cigolini Rule* and abbreviated to *CR*. Cigolini et al. (2002) suggest to dynamically determine the length of the time window production planners should wait and leave the batching machine idle to reduce the flow time without adversely affecting the machine utilization rate. The basic idea of the computation is to balance between the total delay of the already available operations, if the decision of delaying the batch start time is taken, and the upper bound of the avoided delay that could be gained by an hypothetical next-arriving operation that will be added to the batch, if the decision of immediate loading is taken. This idea, when applied to the batch B^0, can be expressed as in (3.3).

$$\sum_{u \in B^0} (S_0 - t_u) \leq p_q \tag{3.3}$$

Using this equation, the latest start time S_0 for the incomplete batch B^0 can be computed as

$$S_0 \leq \frac{p_q + \sum_{u \in B^0} t_u}{\left|B^0\right|} \tag{3.4}$$

When $S_0 > S$, it means that it may be beneficial to delay the start time of B^0 and check if it is possible to include nodes in E^∞ that are available before S_0, i.e., $w_i \in E^\infty | t_{w_i} < S_0$. Instead of adding all the operations that satisfy the previous condition, only the earliest available node w_1 is added. The same reasoning that was applied to B^0 should be applied to the new batch B^1 and iteratively for each new batch B^l that is obtained after adding operation $w_l \in E^\infty$ to batch B^{l-1}. The latest start time S_l of B^l should be recomputed to take the waiting time of

the latest added node $w_l \in E^\infty$ into account. This idea is expressed in (3.5), which generalizes (3.3). The latest start time of batch B^l can be computed as in (3.6).

$$\sum_{u \in B^l} (S_l - t_u) \leq p_q \tag{3.5}$$

$$S_l \leq \frac{p_q + \sum_{u \in B^l} t_u}{\left|B^l\right|} \tag{3.6}$$

All the computations above assume that the jobs are equally important. When different priorities are associated with jobs, a delay of x time units of a job with high priority should not be considered equal to a delay of x time units of a job with lower priority. The computation of the latest start time S_l should be adapted to take into account these new data. Let ω_u be the priority associated with the job of operation u. Let us assume it is decided to add operation w_l to a batch that is still incomplete after this. The delay of the already available operations $w_l \in B^l$ should be weighted with their job priority. In opposition to dynamic scheduling, the next arriving operation is known to be w_{l+1} and therefore its job priority. Considering this, (3.5) can be generalized to obtain (3.7). The latest start time of batch S_l can be computed as shown in (3.8). The complete procedure to select the final batch B^f is detailed in Algorithm 3.1. It is assumed that the elements of E^∞ are sorted in non-decreasing order of their job available times.

$$\sum_{u \in B^l} \omega_u (S_l - t_u) \leq \omega_{w_{l+1}} p_q \tag{3.7}$$

$$S_l \leq \frac{\omega_{w_{l+1}} p_q + \sum_{u \in B^l} \omega_u t_u}{\sum_{u \in B^l} \omega_u} \tag{3.8}$$

Algorithm 3.1 Selection of delaying nodes using the Cigolini Rule (CR)

CigoliniRule (B^0, E^∞)
  $B^f \leftarrow B^0$
  for $l \in \{0, 1, \ldots, |E^\infty| - 1\}$
    $S_l \leftarrow \frac{\omega_{w_{l+1}} p_q + \sum_{u \in B^l} \omega_u t_u}{\sum_{u \in B^l} \omega_u}$
    if $(t_{w_{l+1}} < S_l)$
      $B^f \leftarrow B^f \cup \{w_{l+1}\}$
    else
      break

The two above rules have different perspectives. The *Simple Rule* has a machine perspective and focuses on capacity utilization without considering the job characteristics such as their available times. The delay is mainly determined by the batching capacity of the machine and the batch processing time. Even if it tries not to have a negative impact on machine

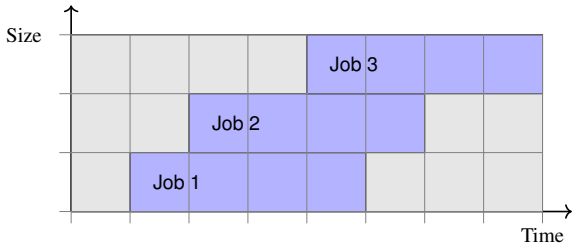
utilization, the *Cigolini Rule* focuses more on job flow times. A third rule, called *Geometric Rule* and abbreviated as *GR*, attempts to combine both the machine and job perspectives. Figure 3.3 is used to illustrate the use of this rule. Three jobs sharing the same batching family have to be scheduled on a machine with a capacity of 3 jobs. The absolute capacity of a batching machine is modeled as a surface whose length is the time during the machine is available, and the width is the batching capacity of the machine. In Figure 3.3, this capacity is modeled through the gray rectangle. The blue surface in Figure 3.3(a) represents the proportion of the absolute capacity that can be used by the available and arriving jobs of the same family when the p-batching constraint is relaxed, i.e., if a job is processed as soon when it is available. In the following, this capacity is called *actual capacity*.

The idea of the geometric rule is to minimize the unused actual capacity when a batching decision is taken. To illustrate this, let us consider the problem in Figure 3.3(a) for which there are three solutions. The first solution is illustrated in Figure 3.3(b) where the started batch B contains only Job 1. In this case, the unused actual capacity is the part of the actual capacity that can no longer be used anymore, i.e., since the machine becomes available until the end of the started batch. This capacity is represented with a dashed surface in all subfigures of Figure 3.3. When $B = \{1\}$, the lost actual capacity is equal to 4, where the unit is expressed as a time unit times a size unit. The second solution is illustrated in Figure 3.3(c) where the started batch $B = \{1, 2\}$. In this case, the lost actual capacity is equal to 3. In the last solution, where $B = \{1, 2, 3\}$, the lost actual capacity is equal to 5. So, according to the *Geometric Rule*, the best decision is to start the batch $B = \{1, 2\}$. The same result is obtained using the *Simple rule* or the *Cigolini Rule*.

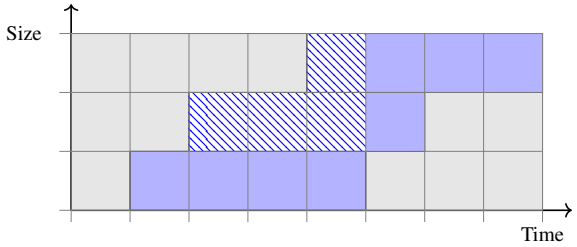
The procedure for selecting the delaying nodes from E^{∞} is detailed in Algorithm 3.2. To compute the actual capacity of the batching machine that is processing the incomplete batch B, its availability time t_m is first computed. When a regular function is considered, it can be shown that the lost actual capacity for a batch is minimized if the batch is started as soon as all the jobs are available. For each possible batch B^l, its earliest start time S_l and earliest completion time C_l are computed. The first component of the lost actual capacity corresponds to the capacity that could be used by the jobs that are available before the start time of the selected batch. The second component corresponds to the capacity that could be used by the job that arrives between the start time and the completion time of the selected batch.

Note that all these rules use local information about the concerned machine and the selected nodes. A delaying node can be selected to complete a batch while it can naturally be part of a complete batch with some unsettled operations. Different ways can be imagined to include more information about the whole graph in order to improve the selection of the delaying nodes. The way that was adopted in this work is quite simple: Exclude all delaying nodes that have a chance, even a small one, of being part of a batch. It is considered that a node has a chance of being in a full batch if there is at least one recipe for which the remaining unsettled nodes can be partitioned in full batches. In other words, a node $w_l \in E^{\infty}$ is excluded if: $\exists q \in R_{w_l} \mid A = \{w \in V^u \wedge q \in R_w\} \wedge \sum_{w \in A} \sigma_w \mod b_q = 0$.

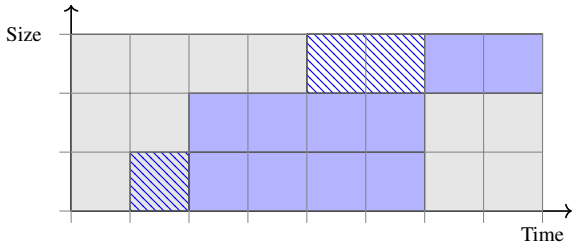
In addition to the non-delay selection strategies given in Section 3.3.2, six additional



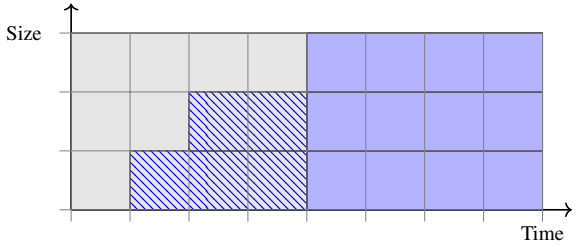
(a) *Batching problem with three jobs*



(b) *Actual lost capacity = 4,* $B^f = \{1\}$



(c) *Actual lost capacity = 3,* $B^f = \{1, 2\}$



(d) *Actual lost capacity = 5,* $B^f = \{1, 2, 3\}$

Figure 3.3 – *Example illustrating the use of the* Geometric Rule

Algorithm 3.2 Selection of delaying nodes using the Geometric Rule (GR)

GeometricRule (B^0, E^∞)
$t_m \leftarrow$ availability of machine m before processing B
for $l \in \{0, 1, ..., |E^\infty|\}$
$S_l \leftarrow \max_{w_i \in B^l}\{t_{w_i}, t_m\}$
$C_l \leftarrow S_l + p_q$
$lostCapacity_l \leftarrow \sum_{w_i \in B^l} \min\{S_l - t_m, S_l - t_{w_i}\}$
$lostCapacity_l \leftarrow lostCapacity_l + \sum_{w_i \in E^\infty \setminus B^l} \max\{0, C_l - t_w - t_{w_i}\}$
$B^f \leftarrow B^l \mid lostCapacity_l \leq lostCapacity_k, \forall k \in \{0, 1, ..., |E^\infty|\}$

strategies can be defined, according to the rule that is used to select delaying nodes and the level of information that is used, i.e., local information or global information about the whole solution. The collecting strategy returns the set of non-delaying nodes E^0 and the set of delaying nodes E^∞. Then, based on this last strategy, the six additional strategies are: *SR-L*, *SR-G*, *CR-L*, *CR-G*, *GR-L* and *GR-G*. The two first letters in the abbreviations define the selection rule, and the last letter refers to the level of used information, *G* for global information and *L* for local information.

3.4.2 Move Feasibility

In the case where a non-delay selection strategy is used, Theorem 3.1 ensures that no cycle is introduced if all the candidates are settled after the last node u of the incomplete batch B and that the unidirectional cut remains valid. When delaying nodes are selected, Theorem 3.1 is no longer sufficient. As they are available after the incomplete batch start time, sequencing these nodes after the last node u of the incomplete batch B is not sufficient as the invariant (3.1) cannot be satisfied. It becomes necessary to unsettle the already batched nodes and to assign new start times to these nodes that allow adding the delaying nodes in the incomplete batch B, i.e., so that the invariant becomes satisfied. If multiple delaying nodes are selected, the start time of the new batch is determined by the job availability of the latest delaying operation. Let us consider an incomplete batch $B = \{u_1, \ldots, u_{|B|}\}$ where the nodes are sorted in their topological order, a set of non-delaying nodes E^0 and a subset of delaying nodes $D = \{w_1, \ldots, w_l\} \subset E^\infty$ where the nodes are sorted in the non-decreasing order of their job availability. The obtained new batch is denoted by $B^f = B \cup E^0 \cup D$ and its earliest start time is given by the job availability of $w = w_l \in D$. All the nodes in E^0 are to be sequenced after the $u_{|B|}$, last sequenced operations in the incomplete batch.

If it is feasible to delay the start time until t_w, there are two alternatives for updating the solution.

1. The first alternative consists in sequencing w after $u_{|B|}$ and only recomputing the start times without modifying the graph. To ensure that w is batched with its resource predecessor, it is sufficient to give for u_1 a start time that is equal to $t_w = S_{r^{-1}(w)} +$

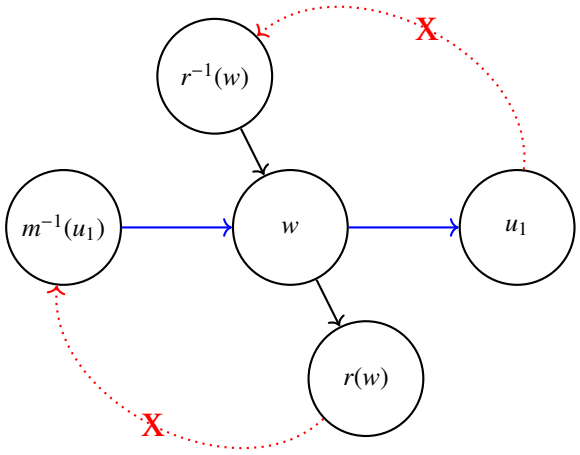


Figure 3.4 – *Resequencing a delaying node w before an incomplete batch*

$l_{r^{-1}(w),w}$ to make sure that all the remaining nodes in B^l will be in the same batch. However, preliminary computational results show that this way of proceeding does not lead to good results. This can be explained by the fact the batching decisions are not reflected in the graph structure. In other words, if a classical algorithm that computes left shifted schedule is used on the obtained batch-oblivious conjunctive graph where all the resource edge weights are determined, the schedule is not feasible if a late operation is added to a batch.

2. The second alternative is to make sure that batching decisions are reflected in the graph. After unsettling all the nodes of B, w is first sequenced between $m^{-1}(u_1)$ and u_1. In this way, the new batch B^f is naturally obtained. However, before doing this, it should be ensured that delaying the start time is feasible, i.e., no cycle is introduced in the graph.

Lemma 3.1 ensures that no cycle is introduced when the maximal delay of the start time of the incomplete batch is lower than its processing time. This condition is naturally satisfied when the Simple Rule is applied (3.2). It can also be proved that this condition is always satisfied when the Cigolini Rule and the Geometric Rule are applied to a problem where all the jobs are equally important (3.6). When jobs have different priorities, the Cigolini Rule can accept the batch start time to be delayed by its processing time.

Lemma 3.1. *Let $w \in D$ be a delaying operation that is sequenced before u_1. No cycle is introduced if the start time of the incomplete batch B is delayed by less than its processing time, i.e., $t_w < S_{u_1} + p_q$.*

Proof. As the operation w is sequenced before batch B, w is sequenced between $m^{-1}(u_1)$ and u_1. There are only two possible ways of creating a cycle by moving w, which are illustrated in Figure 3.4.

1. there was a path in the graph between w and $m^{-1}(u_1)$,
2. or (and) there was a path in the graph between u_1 and w

First, notice that the resource arcs before and after w are deleted when w is moved. In case 1, if such a path exists, this means that there is a path from an unsettled node w to a settled node $m^{-1}(u_1)$. This contradicts the definition of the unidirectional cut $C = (V^s, V^u)$.

In case 2, two situations can occur. Because of reentrant flows, the route predecessor $r^{-1}(w)$ can be u_1. In this case, $t_w = S_{u_1} + p_q$ which contradicts $t_w < S_{u_1} + p_q$. In the second situation, if such a path exists, $r^{-1}(w)$ belongs to this path which implies $S_{r^{-1}(w)} \geq S_{u_1} + p_q$. $r^{-1}(w)$ cannot be the artificial node 0 as this implies that there is a cycle in the graph before moving w, thus $p_{r^{-1}(w)} > 0$. This implies that $t_w - S_{u_1} \geq p_q$, which is in contradiction with $t_w < S_{u_1} + p_q$.

□

If it can be beneficial to accept this delay, the sufficient conditions introduced in Dauzère-Pérès and Paulli (1997) can be used to make sure that no cycle is introduced while the batch is delayed by more than its processing time. This can be the case when optimizing the batching coefficient described in Section 2.4, which is not a regular objective function. Instead of sufficient conditions, it is also possible to check that there is no path from u_1 to w, but also to any $w_l \in E^{\infty}$ verifying $t_{w_l} > S_{u_1} + p_q$. Algorithms like DFS or BFS (Cormen et al. (2009)) can be used to identify such paths. The exploration can be made efficient if the exploration is cut every time an unsettled node $v \in V^u$ is visited. Any delaying node that has its route predecessor not visited by the exploration algorithm can be added to the incomplete batch without making the solution unfeasible. Within this work, we do not allow delaying by more than the processing time of the incomplete batch for three reasons. From a flow time perspective, there is no benefit in delaying by more that the processing time. In a case where a different objective function is used, it is always possible to obtain a solution where the discarded delaying nodes can be added to the incomplete batch within the neighborhood-based metaheuristic. Third, it is not realistic from an industrial perspective to let a machine idle for such long time.

3.4.3 Active Schedule Construction

Using the results of the previous sections, a general algorithm can be designed to construct and improve schedules starting from a batch-oblivious conjunctive graph. As locally non-delay strategies can be considered as a special case of locally active strategies, only one algorithm is given here. The overall schedule construction algorithm is shown in Algorithm 3.3. Given a batch-oblivious conjunctive graph and a selection strategy, this algorithm returns a schedule.

Initially, only the artificial start node 0 is considered to be settled. The different sets, V^s, V^l and V^f defined in Section 3.3.1, are initialized. Then, nodes that meet the criteria of Theorem 3.1 can be successively settled without introducing any cycle. At each iteration,

a root node v is selected in V^u and its resource predecessor u is determined. If the batch containing u is incomplete and v is not a compatible candidate, candidates to complete the batch are searched for using the given selection strategy. If a non-delay strategy is chosen, only set E^0 can be non-empty. If this is the case, all the found nodes are sequenced after u and instead of settling v, the last inserted candidate in E^0 is settled. If an active strategy is chosen, potential delaying nodes can be given in E^∞. As explained above, this set can be reduced when global information about the whole graph is considered. Before moving the selected delaying nodes, all the nodes in the incomplete batch B and the nodes that are settled and reachable from the batch nodes are unsettled first. Then, the delaying nodes are sequenced one by one after the resource predecessor of the incomplete batch B. Lemma 3.1 ensures that no cycle is introduced. If E^∞ is not empty, instead of settling v, the last available selected node is settled. When the node to settle v and the resource predecessor u are determined, the invariant (3.1) and the conditions regarding the batching capacity and compatibility are used to decide whether it is possible to batch nodes u and v. Finally, the different sets are updated.

Algorithm 3.3 An active schedule construction algorithm for a given conjunctive graph G and a given selection strategy

computeStartDatesAdaptively (G, Strategy)

$S_0 \leftarrow 0$
$V^s \leftarrow \{0\}$; $V^u \leftarrow V \setminus \{0\}$; $V^f \leftarrow \left\{v \in V^u \mid r^{-1}(v) = 0\right\}$
$B_v \leftarrow \{v\}$ $(\forall\, v \in V)$; $\sigma_{B_v} \leftarrow \sigma_j$ $(\forall\, v \in V \mid v \in O_j)$
while $V^s \neq V$
  $v \leftarrow$ select $(v \in V^u \mid deg_u^-(v) = 0)$
  $u \leftarrow m^{-1}(v)$
  if $(\sigma_{B_u} < b_u$ **and** $(q_u \neq q_v$ **or** $S_{r^{-1}(v)} + p_{r^{-1}(v)} > S_{m^{-1}(v)}))$
    $V^r \leftarrow \left\{w \in V^f \mid w \in O_j \wedge \exists o_{i,j} \in O_j \wedge q_u \in R_{i,j}\right\}$
    $(E^0, E^\infty) \leftarrow$ Strategy(u, V^r)
    $E^\infty \leftarrow E^\infty \setminus \left\{w_l \in E^\infty \mid \exists q \in R_{w_l} \mid A = \{w \in V^u \wedge q \in R_w\} \wedge \sum_{w \in A} \sigma_w \mod b_q = 0\right\}$
    if $(E^0 \neq \emptyset)$
      for $(i \in \left\{1, \ldots, \left|E^0\right|\right\})$
        sequence w_i after u
      $v \leftarrow w_{|E^0|}$
    if $(E^\infty \neq \emptyset)$
      $u \leftarrow m^{-1}(u_i)$ $(u_i \in B_u \wedge m^{-1}(u_i) \notin B_u)$
      for $(i \in \{1, \ldots, |B_u|\})$
        $V^s \leftarrow V^s \setminus \{v \in V^s \wedge u_i \prec v\}$; $V^u \leftarrow V^u \cup \{v \in V^s \wedge u_i \prec v\}$;
        $V^f \leftarrow V^f \setminus \{r(u_i)\} \cup \{u_i\}$
        $B_{u_i} \leftarrow \{u_i\}$; $\sigma_{B_{u_i}} \leftarrow \rho_j$ $(u_i \in O_j)$
      for $(l \in \{1, \ldots, |E^\infty|\})$
        sequence w_l after u
      $v \leftarrow w_{|E^\infty|}$
  if $(S_{r^{-1}(v)} + p_{r^{-1}(v)} \leq S_u$ **and** $q_u = q_v$ **and** $\sigma_{B_u} + \sigma_v \leq b_v)$
    $S_v \leftarrow S_u$
    $B_v \leftarrow B_v \cup B_u$; $\sigma_{B_v} \leftarrow \sigma_{B_v} + \sigma_{B_u}$
  else
    $S_v \leftarrow \max\left(S_{r^{-1}(v)} + p_{r^{-1}(v)},\ S_u + p_u + s(\sigma_u, \sigma_v)\right)$
  $V^s \leftarrow V^s \cup \{v\}$; $V^u \leftarrow V^u \setminus \{v\}$; $V^f \leftarrow V^f \setminus \{v\} \cup \{r(v)\}$

3.5 Heuristic Approaches

Since the batch-oblivious methodology is not bound to one specific solution approach, it can be deployed within different heuristic frameworks. We developed a heuristic approach based on the idea of *Greedy Randomized Adaptive Search Procedures* (GRASP) of Feo and Resende (1995). Our heuristic creates many different starting solutions by randomizing a construction heuristic, and each solution is independently improved using a metaheuristic. The GRASP based approach is parallelized as follows. Each solution is constructed and improved independently and thus can be run in its own thread. Communication between threads is only needed to update the best overall solution once a thread has completed its computation. A fixed number of threads is used, and each thread restarts with a new initial solution once the chosen metaheuristic has met the stopping criterion, which is the maximum number of non-improving moves in our implementation. In the following, we describe the framework within which are applied the different building blocks developed in the previous sections.

First, we define a deterministic *construction heuristic* which adapts the methods presented in Yugma et al. (2012) and Knopp et al. (2014). If due dates and weights are given, jobs are initially sorted in decreasing order of their ratio $\frac{w_j}{d_j}$ (weight divided by due date). Otherwise, jobs are initially sorted in decreasing order of the sum of the shortest processing durations of their operations. The heuristic then iterates over the sorted list of jobs and successively inserts all operations of the current job. The operations of a job are greedily inserted, starting from the first operation, by selecting the best insertion position for each operation. The best insertion position is determined by the objective function value of the partial solution obtained by actually inserting the considered operation. The construction is completed when all operations of all jobs have been inserted. The randomization of the construction heuristic is done by perturbing the sorted list of jobs as follows. A tuning parameter $P_i \geq 1$ is introduced that steers the perturbation intensity. At each iteration of the construction heuristic, the next job to be inserted is determined by randomly selecting one of the first P_i elements in the sorted list of remaining jobs. The operations of the job are then greedily inserted as described earlier and the job is then removed from the list.

After the construction heuristic, each solution is independently improved using one of the following heuristics: Hill climbing, tabu search or simulated annealing. In all these heuristics, we combine the move of Dauzère-Pérès and Paulli (1997) that is adapted to the batch-oblivious graph with the adaptive start time computation from Section 3.4.3 as follows. After a batch-oblivious move is performed, an adaptive start time computation follows in order to determine start times and batching decisions. The combined result of both modifications is considered as one single move. If such a move is rejected, all involved changes are collectively reverted. The *hill climbing* approach starts with the solution found by the construction heuristic and explores the neighborhood using steepest descent. All moves reachable from the current solution are evaluated, and the one leading to the best solution is selected. The local search approach continues until no strictly better solution is found. The *tabu search* approach explores the same neighborhood as the hill climbing approach, and its implementa-

tion is similar to the approach described in Dauzère-Pérès and Paulli (1997). Due to the low performances of our GRASP approach when using one of these two improvement heuristics, the numerical results reported in this thesis are obtained by using the simulated annealing approach. Chapter 6 discusses the conditions that should help these approaches, particularly tabu search, to be effective when solving the studied industrial problem.

Our *simulated Annealing* metaheuristic is based on the same integrated move and also starts with the solution found by the construction heuristic. In each step, one node is randomly chosen to be moved, its feasible insertion positions are computed, and one of them is randomly selected and performed. In the remaining of the manuscript, this neighborhood operator is referred to as ($\mathcal{N}^1$). We use a geometric cooling schedule that maintains a temperature T which is multiplied by a cooling factor $P_c < 1$ after each iteration. At iteration n, the move is immediately accepted if the current value of the objective function f_n improves the previous objective function value f_{n-1}. Otherwise, the new solution is accepted with a probability of $exp(\frac{-\Delta}{T})$, where $\Delta = f_n - f_{n-1}$. If the new solution is not accepted, all changes related to the move are reversed. Note that this mechanism of accepting or rejecting a new solution is only valid when a unique objective function is optimized. The various changes that must be made to the simulated annealing approach when considering multiple criteria are described in Chapter 5. The search is stopped if the best solution does not improve during a specified number of iterations P_m. The initial temperature is determined by sampling a fixed number P_s of random moves. For each random move r, we evaluate its influence $\Delta = f_r - f_i$ on the objective function value f_i of the initial solution. Then, for a tuning parameter P_p, the P_p-th percentile of these values is selected as the initial value for temperature T.

3.6 Numerical Results

The algorithms presented in this chapter were implemented in C++14 and compiled using the GCC MinGW-W64 compiler in version 5.3. All numerical experiments are conducted on an Intel Xeon E3-1240 3.5 GHz machine (4 cores) running Microsoft Windows 7. Different types of industrial and academic instances are used to perform the experiments. Section 3.6.1 evaluates the improvements brought by the idea proposed in Section 3.3. Section 3.6.2 evaluates the active scheduling construction that was introduced in Section 3.4.

In these two sections, industrial instances are used to perform the evaluation. The instances provided by Knopp et al. (2017) are used. In this benchmark, 15 industrial instances are from the Manufacturing Execution System of a semiconductor manufacturing facility throughout one year. Smaller instances with around 25 machines represent a subset of the actual area while larger instances with around 100 machines correspond to the full area. The number of jobs per instance is between 119 and 346. For each job, between one and seven operations have to be performed. Only some of the machines are capable of processing multiple operations in the same batch. Sequence-dependent setup times are required only for some of the non-batching machines. Since no due times are provided, the total weighted completion time is minimized. Second, 15 random instances that are close to the industrial

instances were generated. The random instances include due times which are not present in the industrial instances. In addition to the instances of Knopp et al. (2017), new industrial instances are proposed. These 30 new industrial instances are larger and from another semiconductor manufacturing facility. The number of jobs per instance ranges from 324 to 503. For each job, between one and five operations have to be performed, with three operations on average. These jobs must be scheduled on an average of 68 machines, all capable of batching. The batching capacity lies between 2 and 7 jobs. Sequence-dependent setup times are required for a subset of machines. Here also, since no due times are provided, the total weighted completion time is minimized.

The sampling strategy of our simulated annealing implementation avoids the need to adapt parameters for individual instances. For all numerical experiments, we used the following identical parameter settings: A cooling factor of $P_c = 0.99999$, a number of samples $P_s = 100$, a maximum number of iterations $P_m = 100\,000$, a temperature percentile of $P_t = 5\,\%$, and a perturbation intensity of $P_i = 5$. All heuristics are run only once, and six parallel threads are used in all runs of the GRASP based approach.

3.6.1 Search Acceleration

This section analyzes the impact of the different selection strategies on the solution quality and the algorithm complexity. The impact on the solution quality is studied through the values of the objective function while the impact on the algorithm complexity is studied through the number of evaluated solutions during the search. The strategies proposed in the original approach are: Static (Static), resequencing (Reseq.) and reassignment (Reass.). The proposed selection strategies in this work are: Integrated (Integ.), random integrated (RandI) and collection (Colle.). We performed numerical experiments for the described industrial (I), random (R) and large industrial (LI) instances allowing a maximum computational time of 5 minutes per instance.

To compare the different selection strategies in terms of computational time, the static strategy is used as a reference. Using this strategy, the graph is traversed in a topological order without modifying it. Except determining the resource edge weights of adjacent nodes, the start time computation algorithm is similar to the one used to compute start times of a classical job-shop scheduling problem. All the remaining strategies modify the graph during the traversal and the objective of this section is to quantify the impact of this dynamic modification on the overall heuristic efficiency. Table 3.4 provides results in terms of the relative deviation from the number of evaluated solutions when using the static selection strategy. We provide average ($\overline{I}$, $\overline{R}$, $\overline{LI}$) and median ($\widetilde{I}$, $\widetilde{R}$, $\widetilde{LI}$) values of these relative deviations over all instances. The results show that the integrated (Integ.) strategy is the less computationally expensive strategy, even when compared to the resequencing strategy that only visits the sequence of the machine on which the incomplete batch is processed. In terms of search scope, the integrated strategy is similar to the reassigning strategy while its impact on the cost of the solution evaluation is insignificant compared to the reassigning strategy. The collection strategy is comparable to the resequencing strategy on industrial instances and a

	I	|*J*|	|*M*|	*Static*	*Reseq.*	*Reass.*	*Integ.*	*RandI.*	*Colle.*	*Best*
Industrial (total weighted completion time)	1	119	24	92859	92704	92847	92590	92729	92574	92213
	2	148	22	244072	242612	239958	239603	241160	239074	238885
	3	195	25	211428	202988	201591	202649	202106	201338	200403
	4	209	24	270566	260563	260685	262094	261235	260072	257881
	5	186	88	167725	164165	162814	163828	162711	163669	161895
	6	268	26	336934	329559	326863	327466	331198	333012	322253
	7	210	94	150225	150138	149527	149347	149746	150189	149123
	8	310	17	448961	450223	452413	449348	452473	453228	438563
	9	231	95	167311	167178	167182	165880	165501	165032	164967
	10	245	94	198503	198608	192045	191651	190891	192146	189736
	11	302	24	555015	551641	551996	554145	554133	555569	546987
	12	302	24	344315	340928	343984	341900	340457	341343	337527
	13	324	94	345414	332642	342826	322730	326592	322728	320299
	14	315	101	450482	450548	475399	452997	448138	447107	443049
	15	346	94	737808	705237	706577	695041	702964	706138	666736
Random (total weighted tardiness)	1	20	3	10259	10097	10277	10460	10311	10719	9760
	2	20	3	5993	5953	5910	5923	5917	5920	5807
	3	20	3	7026	7135	7029	7000	7090	7052	6919
	4	40	6	9139	8775	9230	9087	8818	9189	8355
	5	40	6	14415	14660	14298	14280	14591	14605	13819
	6	40	6	32151	31749	31579	31934	31775	30426	30247
	7	60	9	16095	15933	15975	16405	16360	16575	15055
	8	60	9	40607	40993	42348	40904	41829	40798	38681
	9	60	9	31250	28833	31507	31647	31008	28582	27163
	10	100	15	28969	26926	27424	27469	27799	27209	25923
	11	100	15	29229	30780	30626	32270	31280	32560	29229
	12	100	15	39357	37969	40997	39425	39266	38351	37641
	13	200	30	25805	25020	39772	39772	39772	39772	20533
	14	200	30	48187	46223	49237	46646	44373	44699	42915
	15	200	30	55882	52089	58693	51780	51212	52449	46796

Table 3.2 – *Detailed results for industrial (I) and random (R) instances*

	I	$\|J\|$	$\|M\|$	*Static*	*Reseq.*	*Reass.*	*Integ.*	*RandI.*	*Colle.*	*Best*
Large industrial (total weighted completion time)	1	387	68	1131106	1108559	1109041	1107291	1097131	1096921	1093221
	2	324	68	835842	822043	817539	808904	805291	811100	805291
	3	371	68	1031342	1014364	1010052	995683	997343	1001902	992560
	4	439	68	1394102	1394477	1383265	1359476	1355619	1349153	1338876
	5	430	68	1287692	1265294	1270913	1258167	1256222	1261125	1246551
	6	395	68	1082755	1077850	1102216	1057445	1069574	1069176	1050548
	7	411	67	1010895	1003652	1041410	996867	994410	1000492	990319
	8	365	69	887616	871965	873220	862148	863043	871351	860628
	9	503	69	1468135	1440749	1515790	1427672	1427048	1452309	1408230
	10	343	68	738287	735367	733741	730888	729417	730368	727274
	11	335	68	755049	746470	735004	724180	724567	728510	720125
	12	345	68	809212	802516	812959	804917	801155	799757	793489
	13	377	67	835125	835169	829729	828000	829105	830386	823082
	14	388	67	903892	905564	901933	897496	903206	901618	894565
	15	463	68	1122956	1096137	1152916	1105984	1109073	1116161	1091991
	16	391	69	889565	877978	932764	868486	869672	875613	864129
	17	444	68	1194587	1179057	1163692	1157813	1157872	1189496	1145017
	18	442	69	1019700	1019013	1025305	1017598	1018673	1032335	1000346
	19	434	68	1053419	1045067	1062705	1046149	1051990	1059316	1036134
	20	408	68	1012496	999743	1012212	992658	1000119	1005128	987737
	21	363	68	750520	729772	729138	728013	731591	737031	726536
	22	368	68	821328	817840	820089	802799	806050	811743	796646
	23	409	68	943067	934059	925559	912422	923405	924488	912422
	24	391	69	892340	875322	932764	868486	871322	875784	864129
	25	365	69	885972	874476	873168	860965	868457	868399	859919
	26	387	68	1126199	1113093	1105315	1108028	1103167	1099767	1095099
	27	371	68	1033292	1017538	1011163	997569	998017	1002329	994029
	28	324	68	840785	823818	814969	810101	814505	807115	805496
	29	439	68	1391421	1394592	1383596	1360509	1362046	1353407	1337919
	30	430	68	1287216	1261929	1271679	1250807	1250105	1265772	1245194

Table 3.3 – *Detailed results for large industrial (LI) instances*

bit more computationally expensive on the random instances. This can be explained by the fact that the collection strategy looks for a set of candidates instead of a unique candidate within the integrated strategy. Regarding the random integrated strategy, it is more expensive than all the other strategies except the reassignment strategy. This strategy is similar to the collection strategy as it searches for possible candidates, but instead of settling all the found nodes, it randomly picks one and settles it. To summarize, the proposed idea to accelerate the search significantly reduces the cost of modifying the graph dynamically. Also, the larger the number of batching machines and the larger the batching capacities, the more significant the impact of the strategies on the heuristic efficiency.

	Reseq.	***Reass.***	***Integ.***	***RandI.***	***Colle.***
$\overline{I}$	-3.98%	-37.24%	**-1.92%**	-15.73%	-3.96%
$\widetilde{I}$	-4.21%	-44.93%	**-2.48%**	-16.07%	-4.03%
$\overline{R}$	-7.08%	-28.58%	**-5.81%**	-18.07%	-11.27%
$\widetilde{R}$	-7.83%	-25.35%	**-6.17%**	-19.15%	-11.71%
$\overline{LI}$	-9.74%	-72.25%	**-0.86%**	-13.49%	-7.98%
$\widetilde{LI}$	-9.71%	-73.08%	**-0.76%**	-13.72%	-8.02%

Table 3.4 – *Impact of the different strategies on the heuristic efficiency.*

	Initial	***Static***	***Reseq.***	***Reass.***	***Integ.***	***RandI.***	***Colle.***
$\overline{I}$	14.14%	3.62%	2.04%	2.26%	**1.35%**	1.48%	1.46%
$\widetilde{I}$	13.47%	2.37%	1.40%	1.22%	1.19%	0.95%	**0.92%**
$\overline{R}$	67.98%	8.70%	**6.27%**	14.13%	12.89%	12.15%	11.93%
$\widetilde{R}$	58.34%	6.29%	**5.31%**	6.11%	7.17%	5.65%	5.47%
$\overline{LI}$	10.05%	3.09%	1.89%	2.75%	**0.76%**	0.92%	1.37%
$\widetilde{LI}$	9.88%	3.20%	1.62%	2.01%	**0.58%**	0.90%	1.17%

Table 3.5 – *Aggregate results for all instances.*

Table 3.2 provides the obtained objective function values for the (I) and (R) instances using the GRASP based approach. Table 3.3 provides the same results for the (LI) instances. In column $|J|$ and $|M|$, the number of jobs and the number of machines are respectively given for each instance. The next six columns report the results for the different strategies. The last column "Best" reports the best values obtained after a long computational time, most of them using the integrated strategy by allowing a computational time of 2 hours. Table 3.5 provides results in terms of the relative deviation to the best values. We provide average ($\overline{I}$, $\overline{R}$,

$\overline{LI}$) and median ($\widetilde{I}$, $\widetilde{R}$, $\widetilde{LI}$) values of these relative deviations over all instances. The column "Initial" refers to the solution that is computed using the non-randomized version of the construction heuristic. Regarding the industrial instances, it can be noticed that the potential improvement cannot be expected to be large as the static strategy already obtains solutions after 5 minutes that are already quite close (3.61% on average) to the solutions obtained after 2 hours. However, an improvement of 1% in the industrial context is already significant. The results show that the new proposed strategies outperform the old ones when applied to the industrial instances (I, LI). Regarding the random instances, the results show that the resequencing strategy outperforms all others, especially when the average of relative deviations (row $\overline{R}$) is analyzed. The average of relative deviations for less performing strategies is mainly due to significantly high relative deviation for one particular instance (13). When $\widetilde{R}$ is analyzed, the gap between the different strategies to the resequencing one is reduced, especially the collection strategy.

3.6.2 Active Scheduling Approaches

This section analyzes the impact of the six active strategies proposed in Section 3.4 on the solution quality. Recall that the six strategies are: *SR-L*, *SR-G*, *CR-L*, *CR-G*, *GR-L* and *GR-G*. The two first letters in the abbreviations define the selection rule, and the last letter refers to the level of information that is used, *G* for global information and *L* for local information. The impact on the solution quality is studied through the objective function. As in the previous section, We performed numerical experiments for the described industrial (I) and random (R), and large industrial (LI) instances allowing a maximum computational time of 5 minutes per instance.

Table 3.7 provides the obtained objective function values for the (I) and (R) instances using the GRASP based approach. Table 3.8 provides the same results for the (LI) instances. In column $|J|$ and $|M|$, the number of jobs and the number of machines are respectively given for each instance. The next six columns report the results for the different strategies. The last column "Best" reports the best values obtained after a computational time of 2 hours. Table 3.6 provides results in terms of the relative deviation to the best values. We provide average ($\overline{I}$, $\overline{R}$, $\overline{LI}$) and median ($\widetilde{I}$, $\widetilde{R}$, $\widetilde{LI}$) values of these relative deviations over all instances. The column "Non-delay" reports the best results that were obtained by locally non-delay strategies. The six remaining columns report the aggregate results of the proposed active strategies.

When comparing the different active strategies, either using local or global information, the strategies using the Cigolini Rule (CR) show lower results compared to the two other rules. Most of the best results among active strategies are obtained when using the Geometric Rule (GR). The fact that the Geometric Rule (GR) performs better than the Simple Rule (SR) is confirmed when comparing their results. Except for the random instances, the results also show that the chosen way to use global information does not lead to statistically significant improvements compared to using local information. Considering what was discussed earlier, GR-L strategy can be considered as the best active strategy. This strategy shows

	Non-delay	*CR-G*	*CR-L*	*GR-G*	*GR-L*	*SR-G*	*SR-L*
$\overline{I}$	1.35%	1.15%	1.54%	1.00%	**0.98%**	1.26%	1.20%
$\widetilde{I}$	0.92%	1.01%	0.83%	0.91%	0.81%	0.81%	**0.72%**
$\overline{R}$	6.27%	21.67%	22.02%	**17.47%**	18.71%	**17.47%**	20.67%
$\widetilde{R}$	5.31%	21.13%	20.73%	**8.29%**	10.58%	11.56%	16.26%
$\overline{LI}$	0.76%	3.34%	2.82%	1.50%	**1.40%**	1.65%	1.69%
$\widetilde{LI}$	0.58%	2.61%	2.59%	**1.06%**	1.10%	1.26%	1.17%

Table 3.6 – *Aggregate results for active strategies over all instances.*

better results than all non-delay strategies on the small industrial instances (I). The same conclusion can be derived for almost all the other active strategies. As all active strategies, the GR-L strategy is not effective on the random instances (R). With 16.5% less moves than the static strategy, GR-L is less efficient than all the non-delay strategies, except the reassignment strategy. Even with this low efficiency, when considering the large industrial (LI) instances, GR-L is performing better than all the non-delay strategies, except the integrated strategies (deterministic or random). Globally, the results show that non-delay approaches dominate active strategies on most instances. After some analysis, we believe this is due to the fact that, until some point, there is a positive correlation between the filling of batches, which is the primary objective of the active approaches, and the minimization of the total weighted completion time. However, beyond this point, systematically trying to fill batches can have a negative impact on the solution quality. Active strategies are valid locally, but may negatively impact the remainder of the schedule, i.e., the completion times of the following operations in the routes and in the sequences on machines. Hence, by postponing too much operations that are early in the schedule, the completion times of many related operations in the schedule might increase, and thus the sum of completion times that is minimized.

The observation above supports the fact that active scheduling approaches are better than non-delay scheduling approaches on the (I) instances but are worst on the (LI) instances. This is probably because only a small subset of machines in the (I) instances are capable of batching while all machines are capable of batching in the (LI) instances. In other words, all operations of the jobs in the (LI) instances are performed on batching machines and systematically applying active scheduling approaches can only delay the completion times of the jobs. This negative impact is negligible on the (I) instances since only few operations of each job are to be performed on batching machines. The results on the (R) instances can also be explained by the fact that there may be no significant correlation between the filling of batches and minimization of the total weighted tardiness. The interest of using active scheduling approaches is shown in Section 4.5.2, where some of the criteria described in Chapter 2 are optimized. It is shown that active strategies quickly converge to solutions with a more relevant structure from an industrial point of view.

	I	|*J*|	|*M*|	*CR-G*	*CR-L*	*GR-G*	*GR-L*	*SR-G*	*SR-L*	*Best*
Industrial (total weighted completion time)	1	119	24	92739	92797	92958	92964	92957	92876	92213
	2	148	22	239790	239592	238943	241058	238970	240517	238885
	3	195	25	201961	202096	202716	201600	201722	201221	200403
	4	209	24	262321	261439	261518	258913	260511	260811	257881
	5	186	88	162430	162693	164144	162314	163612	162239	161895
	6	268	26	330303	323665	325194	329327	326392	328585	322253
	7	210	94	149661	149313	149573	150039	149616	149413	149123
	8	310	17	443011	442204	449679	451968	449993	452780	438563
	9	231	95	166225	166067	165224	166355	166216	166141	164967
	10	245	94	191865	192445	190847	190395	191075	191006	189736
	11	302	24	553066	546987	552216	547104	553212	548450	546987
	12	302	24	339197	347394	340130	339204	342865	340817	337527
	13	324	94	323552	338941	324935	324845	321338	322958	320299
	14	315	101	450905	459113	446484	446860	445681	447682	443049
	15	346	94	688804	690728	679048	678954	707048	699825	666736
Random (total weighted tardiness)	1	20	3	10757	11128	10988	10887	10757	10887	9760
	2	20	3	5847	5999	6007	6050	5953	6055	5807
	3	20	3	6919	7018	7000	6994	7022	7006	6919
	4	40	6	8929	9095	8903	9171	8748	9001	8355
	5	40	6	16070	16684	14857	14410	15721	16219	13819
	6	40	6	31401	30858	31668	31694	31196	31370	30247
	7	60	9	15948	17089	15814	15842	15386	16383	15055
	8	60	9	47741	47150	41887	42772	43545	44971	38681
	9	60	9	32902	30748	28740	28589	27892	29737	27163
	10	100	15	31840	32567	30055	31420	29040	32547	25923
	11	100	15	37082	36550	34830	36381	36821	36416	29229
	12	100	15	48143	45756	42405	44822	41992	45317	37641
	13	200	30	39772	39772	39772	39772	39772	39772	20533
	14	200	30	56158	56158	56158	56158	56158	56158	42915
	15	200	30	62981	62981	62981	62981	62981	62981	46796

Table 3.7 – *Detailed results for active strategies on industrial (I) and random (R) instances*

	I	|*J*|	|*M*|	*CR-G*	*CR-L*	*GR-G*	*GR-L*	*SR-G*	*SR-L*	*Best*
Large industrial (total weighted completion time)	1	387	68	1107391	1108829	1104264	1099900	1100460	1111277	1093221
	2	324	68	819461	816441	817861	813607	816252	812244	805291
	3	371	68	1003629	1004053	999882	992560	1000167	1001452	992560
	4	439	68	1382811	1380381	1352936	1348068	1354399	1361682	1338876
	5	430	68	1283581	1279889	1282336	1272651	1271828	1269531	1246551
	6	395	68	1092304	1083130	1072512	1070123	1079720	1072578	1050548
	7	411	67	1045562	1008866	1003780	1002627	1004605	999941	990319
	8	365	69	880299	884002	860845	870810	865953	868298	860628
	9	503	69	1541709	1529149	1464928	1471851	1485569	1473129	1408230
	10	343	68	742596	732379	730873	730143	732828	731675	727274
	11	335	68	730402	731720	723979	724048	729548	728564	720125
	12	345	68	808197	813386	800198	802408	800441	799376	793489
	13	377	67	850238	845181	831649	834439	833975	832272	823082
	14	388	67	911439	912696	903147	902567	900748	901895	894565
	15	463	68	1179034	1159050	1103211	1118703	1106526	1134971	1091991
	16	391	69	894942	892842	879157	873143	870932	871939	864129
	17	444	68	1223141	1189413	1175077	1166131	1187585	1181704	1145017
	18	442	69	1083392	1081267	1049754	1056137	1059130	1050512	1000346
	19	434	68	1077375	1085293	1068728	1058395	1070051	1069143	1036134
	20	408	68	1012612	1006108	1000042	997970	1005372	999306	987737
	21	363	68	738997	736468	734367	733876	730712	733143	726536
	22	368	68	828245	818219	812782	810161	810529	808839	796646
	23	409	68	950087	931594	928500	924353	924254	925620	912422
	24	391	69	887504	887613	873070	873494	874745	879765	864129
	25	365	69	877080	875241	860828	870883	865953	874568	859919
	26	387	68	1104112	1110921	1101706	1095953	1103186	1107208	1095099
	27	371	68	1005287	1009103	998316	998334	997809	1001273	994029
	28	324	68	819202	816981	810878	812359	814459	809237	805496
	29	439	68	1369409	1374306	1352330	1347685	1353412	1348199	1337919
	30	430	68	1298812	1284300	1279408	1267957	1275803	1280827	1245194

Table 3.8 – *Detailed results for active strategies on large industrial (LI) instances*

3.7 Conclusion

In this chapter, we considered the complex job-shop scheduling problem that is solved in Knopp (2016). After formalizing the considered constraints and criteria, the batch-oblivious approach is reviewed. The remainder of the chapter was dedicated to the generalization of some results in the original batch-approach and the improvement in terms of efficiency. New efficient strategies, qualified as locally non-delay strategies, that adaptively fill up incomplete batches during the graph traversal are proposed without delaying their start times. The numerical results show that the new strategies improve the batch-oblivious approach in terms of efficiency and, even more importantly, in terms of solution quality on the industrial instances. In the second part of the chapter, a new construction algorithm that intentionally inserts idle times is developed. The numerical results show that one of the proposed active strategies is outperforming non-delay strategies on one set of industrial instances.

The scheduling problem considered in this chapter is a subproblem of the one defined in Chapter 2. Chapter 4 extends the improved batch-oblivious approach by considering all the constraints defined in Chapter 2. The numerical results support three different strategies: The resequencing strategy for the random instances, the integrated strategy for large industrial instances and the active strategy that uses the proposed geometric rule for small industrial instances. An interesting perspective is to use an offline or online learning algorithm that chooses the best strategy for the instance to solve and probably also depending on the optimization phase. After studying several strategies in this chapter, it can be concluded that it is not obvious to obtain significant improvements by lying only on more sophisticated node selection strategies. A more promising perspective is the improvement of the neighborhood structure by choosing interesting nodes to move and their insertion positions. Instead of simulated annealing, metaheuristics such as Variable Neighborhood Search can be interesting to solve the industrial instances that are characterized by their large size.

Chapter 4

Extensions of the Batch-Oblivious Approach

Modeling complex batching machines is the most challenging feature considered in this chapter. In Knopp et al. (2014), the concept of *route graph* is proposed to model complex machines in detail. However, this modeling does not support the modeling of complex machines with batching capabilities such as wet benches in the diffusion area. One of the important contributions of this thesis is the generalization of the batch-oblivious conjunctive graph through the use of route graphs to model complex batching machines. The resulting graph is referred to as *extended batch-oblivious conjunctive graph*. In general, this chapter extends the improved batch-oblivious approach described in Chapter 3 so that all the constraints defined in Chapter 2 are taken into account. Section 4.1 recalls the route graph modeling and provides the formal description of all considered constraints. Section 4.2 formalizes the considered criteria. Besides the criteria that are defined in previous works, this section suggests two new criteria: The *discounted weighted number of moves* to model the throughput within a rolling horizon framework and the *target satisfaction indicator* to model the satisfaction of production targets.

After formalizing all the features of the industrial scheduling problem in the two first sections, Section 4.3 is dedicated to the description of the different building blocks that allow to design a general approach that considers all the features of the studied problem. Section 4.3.1 specifies the extended batch-oblivious conjunctive graph that fully merges the original batch-oblivious conjunctive graph recalled in Section 3.2.1 and the route graph modeling. With the obtained graph, it becomes possible to model in detail complex batching machines such as wet benches. Section 4.3.2 describes the computation of accurate start times while considering the different features: Minimum time lags, batching constraints, unavailability periods, multiple resources per operation and sequence-dependent setup times. When modeling in detail complex batching machines, a batching decision not only leads to the modification of some edge weights, but also to the modification of the graph structure. Thus, Section 4.3.3 specifies the different conditions that must be met so that the batching decisions to be taken on the fly are feasible. Thanks to the generalization of the original approach proposed in Section 3.3.1 where already taken decisions are reconsidered, minimum batch size constraints can naturally be integrated within the batch-oblivious approach as described in Section 4.3.4. The different building blocks developed in Section 4.3 are used to propose a general solution

approach in Section 4.4 that can efficiently solve complex job-shop scheduling problems while considering all the features of the studied industrial problem. Section 4.5 ends this chapter with a discussion on the modeling of complex machines, and reports numerical results that show the advantages of using the integrated and active strategies introduced in Chapter 3 when optimizing some of the criteria defined in Section 4.2.

4.1 Formal Modeling of the Problem Constraints

We are given a set of *jobs* $\mathcal{J}$ to schedule. Each job $j \in \mathcal{J}$ has a *release date* $r_j \in \mathbb{Z}$ and a *size* $\sigma_j \in \mathbb{N}$. A release date is negative when the job is available in the area before the start of the scheduling horizon. It takes a positive value when the job arrives in the area after the beginning of the scheduling horizon. In the real-world problem, each job has a linear route of processing operations. In order to model machines in detail, their internal resources should be modeled explicitly. For the sake of clarity, the term *resource* designates the production entities at which level the scheduling decisions are taken. The term *machine* is reserved for independent production entities found on the shop floor. For instance, a wet bench, as described in Section 2.2.1, is called a machine and its processing components (M_1, M_2, D) are called resources. If a machine is complex and is modeled in detail, it has at least two internal resources in the problem definition. If a machine is not modeled in detail, the two terms "machine" and "resource" can be used indifferently. For the same reason, a *step* corresponds to the processing of a job within a whole machine such as a furnace or a wet bench. The term *operation* is used to describe the elementary processing of a job within an internal resource of a machine. Therefore, the set of jobs $\mathcal{J}$ have to be processed on a given set of *resources* $\mathcal{M}$. Each operation must be assigned to a subset of resources chosen from a set of qualified resources. Contrary to Chapter 3 where operations require a unique resource, it is considered here that an operation may require multiple resources. For example, if a furnace is modeled in detail, a step performed on this machine will be replaced by a sequence of internal operations requiring multiple resources. For instance, we have seen in Section 2.2.2 that the process requires two resources, i.e., a boat and a tube. The resulting approach is also capable of solving scheduling problems where steps may require multiple machines. For example, it is possible to solve the scheduling problem encountered in the photolithography area where steps require the photolithography tool and reticles at the same time. However, to make the description easier and as the focus of this thesis is the scheduling of the diffusion area, we assume that each step requires a single machine.

The inputs describing the industrial problem include, for each job, a linear route of processing steps to be sequenced on the qualified machines. If complex machines should be modeled in detail, a sequence of operations must be scheduled on the internal resources of the machine rather than having to sequence a single step on a complex machine. For a solution to be feasible, it is necessary to ensure that, when a resource of a machine performs an operation of a step of a job, all the subsequent operations must be performed on the internal resources of the same machine. To include these dependencies between internal components of complex machines, the concept of *route graph* is introduced in Knopp et al.

(2014). For each job, this graph is constructed during the preprocessing phase, i.e., before solving the problem. In Section 4.1.1, the preprocessing of the route graph is formalized. In Section 4.1.2, the basic problem description is given. Section 4.1.3 extends the problem description by introducing batching constraints. Section 4.1.4 introduces the modeling of unavailability periods and Section 4.1.5 formalizes the modeling of minimum and maximum time lag constraints.

4.1.1 Route Graph Modeling

For each job in $\mathcal{J}$, a linear sequence of steps $S = (o_1, \ldots, o_i, \ldots, o_n)$ is given, where n is the total number of steps. The route graph G of each job, where each node represents an operation, is obtained from the given sequence as follows:

- **Association of separator nodes:** For each step $o_i \in S$, associate two *separator nodes* α_i and ϕ_i.
- **Static assignment of machines:** For each step $o_i \in S$ that can be processed on m_i machines, create a set $P_i = \{o_{i,1}, \ldots, o_{i,j}, \ldots, o_{i,m_i}\}$ with a cardinality equal to the number of machines that are qualified to process this step.
- **Step decomposition into operations:** For each step $o_{i,j} \in P_i$, if the assigned machine is considered as complex, a sequence of $n_{i,j}$ operations $Q_{i,j} = (o_{i,j,1}, \ldots, o_{i,j,k}, \ldots, o_{i,j,n_{i,j}})$ is created. Each sequence only describes the process type without specifying the required resources. For the case of a wet bench, an example of such sequence is a chemical etching operation that is followed by a drying operation. If the assigned machine is not considered as complex, the step $o_{i,j} \in P_i$ is considered as an operation, i.e., $Q_{i,j} = (o_{i,j,1}) = (o_{i,j})$.
- **Static assignment of resources:** Each operation $o_{i,j,k} \in Q_{i,j}$ may require a subset of the complex machine resources and there may be flexibility over these resources. If for example a furnace with two tubes and two boats is modeled in detail, the processing operation requires two resources and can sometimes be performed indifferently using one of the couples (tube, boat). Let $m_{i,j,k}$ be the number of possible combinations of resources that can process operation $o_{i,j,k} \in Q_{i,j}$. As the resources are statically assigned to each operation $o_{i,j,k}$ in the sequence $Q_{i,j}$, the total number of sequences after assignment is equal to $m_{i,j} = \prod_{o_{i,j,k} \in Q_{i,j}} m_{i,j,k}$. Let $C^l_{i,j} = (o^l_{i,j,1}, \ldots, o^l_{i,j,k}, \ldots, o^l_{i,j,n_{i,j}})$, where $l \in \{1, \ldots, m_{i,j}\}$. A sequence $C^l_{i,j}$ is called in the remaining of the chapter a *movable component* after the resources are assigned. As stated earlier, when a machine is not considered to be complex, it is considered as a resource. Therefore, each step $o_{i,j} \in P_i$ that was assigned to a non-complex machine corresponds to a unique movable component $C^1_{i,j} = Q_{i,j} = (o_{i,j})$ with one node. Let C be the set of all movable components.
- **Parallel composition:** For each movable component $C^l_{i,j}$, add an arc from α_i to the first node in the sequence and an arc from the last node in the sequence to ϕ_i, i.e., $(\alpha_i, o^l_{i,j,1})$

and $(o^l_{i,j,n_{i,j}}, \phi_i)$.

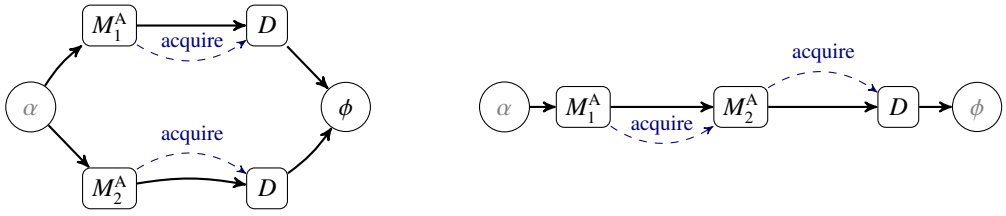
- **Serial composition:** Add an arc from each ϕ_i to α_{i+1}, where $i < n$.

Starting with a linear sequence of steps, after executing the algorithm given above, a *two-terminal series-parallel graph* (Eppstein (1992)) $G_j = (O_j, E_j, \alpha_j, \phi_j)$ is associated with each job $j \in \mathcal{J}$, where O_j denotes the set of nodes and E_j the set of arcs. As described above, the resulting movable components are inserted between two separator nodes. In $G_j = (O_j, E_j, \alpha_j, \phi_j)$, α_j denotes the left separator node of the first step of the job route and ϕ_j represents the right separator node of the last step of the job route. To schedule a job, the assignment decisions are obtained by selecting a unique movable component $C \in \mathcal{C}$ between two successive separator nodes α and ϕ and the selected operations are sequenced on their assigned resources. Through the notion of movable components, the use of the route graph ensures that, when a resource of a machine performs an operation of a step of a job, all the subsequent operations must be performed on the internal resources of the same machine. We extend the basic version of the problem description by an additional constraint. Consider two operations that are part of the same movable component and require a shared resource. In some cases, we want to acquire the resource between the two operations exclusively, i.e., other operations cannot use the resource in between. This can model for example the use of a boat within a furnace. A boat is used as follows: First, wafers are loaded from its containers to the boat using the charging robot. Then, the boat is moved into the tube where the process is conducted. Afterward, the boat is removed from the tube and has to cool down before its wafers can be unloaded using the charging robot. After loading wafers in a boat, this resource cannot be used by other batches until the wafers of the current batch are unloaded after cooling down.

More formally, let $C = (o_1, \ldots, o_i, \ldots, o_{n_c})$ be a movable component. For each movable component $C \in \mathcal{C}$, we are given a set of resource acquisitions as a subset $A_i \subset M_i \times O_j$ for each operation $o_i \in C$. Each acquisition $a = (m, o_l) \in A_i$ specifies the acquired resource m and the operation o_l at which level the resource is released. Acquisition constraints are restricted to operations within the same movable component, i.e., $o_i, o_l \in C$. The path $P_m = (o_i, \ldots, o_l)$ in G_j must be minimal in the sense that, for all operations $o_k \in P_m$ with $k \neq i$, $k \neq l$, we must have $m \notin M_k$. Note that a resource can be immediately reacquired, i.e., $m \in A_k$ is allowed. In Knopp (2016), it is considered that, if a resource m is acquired between two operations o_i and o_l, then m becomes available after the completion time of o_l. This modeling must be generalized so that it is possible to handle blocking constraints (Hall and Sriskandarajah (1996)) encountered for example in wet benches. When dealing with these constraints, resource m is released as soon as operation o_l starts its processing. To model blocking constraints through resource acquisitions, when $a = (m, o_l) \in A_i$, we consider that $m \in M_l$ even if m is not actually required by v. By allowing different processing times for an operation depending on the resources, it becomes possible to model blocking constraints.

Figures 4.1(a) and 4.1(b) illustrate two route graphs of two jobs with a wet cleaning step using, respectively, a short process and a long process on the wet bench machine. As already specified, the wet bench machine consists of two modules (M_1 and M_2) and a dryer D. First,

let us consider the case of a short process that uses one of the processing modules and module D. The modeling of an operation of a job that should be performed on this machine is shown in the route graph of Figure 4.1(a). Since resources are statically assigned to operations, the possibility of performing the processing part of a short process is expressed through the flexibility of choosing one of the sequences in the route graph. Resource acquisitions are indicated by superscript "A" and a dashed line (which is not an edge of the route graph) to the release operation. The scheduling algorithm is allowed to use resource M_1 only when resource D becomes available. Resource acquisition constraints are used to express blocking constraints that model the absence of storage capacity between machine components. As a consequence of this absence of storage capacity, a batch that completes a step remains on the machine component until a downstream component becomes available for processing. Our modeling considers that, in addition to D, M_1 is part of the resources required by the drying operation even if actually it is not the case. However, the drying operation requires M_1 for a duration 0. The same notations, given above for the case of short processes, can be used to understand the modeling of long processes, i.e., processes that follow this sequence ($M_1 \rightarrow M_2 \rightarrow D$) as shown in Figure 4.1(b).



(a) *Short process* **(b)** *Long process*

Figure 4.1 – *Modeling of a wet bench machine.*

4.1.2 Basic Problem Description

Formally, for each job $j \in \mathcal{J}$, feasible sequences of operations are specified by a given *route graph* $G_j = (O_j, E_j, \alpha_j, \phi_j)$, where O_j denotes the set of nodes and E_j the set of arcs. As described above, the resulting movable components are inserted between two separator nodes. In $G_j = (O_j, E_j, \alpha_j, \phi_j)$, α_j denotes the left separator node of the first step of the job route and ϕ_j represents the right separator node of the last step of the job route. For each operation $v \in O_j$, we are given a set of resources $M_v \subset \mathcal{M}$. As a generalization of the problem considered in Knopp (2016), instead of using all the resources for the same duration, each resource may be used for a specific duration. Formally, a processing time $p_{v,m} \in \mathbb{N}$ is given for each operation v on resource $m \in M_v$. It is possible to have $p_{v,m} = 0$ to represent for example the time during which a resource is used at the level of the operation that releases it to model blocking constraints. If the example in Figures 4.1(a) is considered, the drying operations, at which level resource M_1 is released, uses M_1 during $p = 0$. However, the processing time $p_v \in \mathbb{N}_0$ of operation v, computed as $p_v = \max_{m \in M_v}(p_{v,m})$, cannot be null. For each resource $m \in M_v$ that is required by an operation v, a setup family $f_{v,m} \in \mathcal{F}$ from

a given set of *setup families* $\mathcal{F}$ is specified. A given mapping $s : \mathcal{F} \times \mathcal{F} \to \mathbb{N}_0$ prescribes *sequence-dependent setup times* between scheduled operations that use the same machine. Setup times must fulfill the triangle inequality: For all $(f_1, f_3, f_3) \in \mathcal{F} \times \mathcal{F} \times \mathcal{F}$, $s(f_1, f_3) \leq s(f_1, f_2) + s(f_2, f_3)$ must hold. For separator operations $v \in O$, we assume without loss of generality that $M_v = \emptyset$ and $p_v = 0$.

A schedule is completely characterized by providing for each job $j \in \mathcal{J}$ a *route selection* $R_j \subset O_j$ and *start dates* $S_{i,j} \in \mathbb{Z}$ for all *selected operations* $o_{i,j} \in R_j$. The route selection R_j must describe a path $(\alpha_j = o_{1,j}, \ldots, o_{|R_j|,j} = \phi_j)$ in the route graph G_j. We denote the resources and processing durations related to this selection as $M_{i,j}$ and $p_{i,j}$, respectively. The disjoint union $R = R_1 \dot\cup R_2 \ldots \dot\cup R_{|J|}$ denotes all selected operations. To describe a schedule that is *feasible*, selected routes R_j and start dates $S_{i,j}$ have to respect several constraints that are detailed in the following:

- Preemption is not allowed: Once the processing of operation has begun, it cannot be interrupted. Thus, the *completion time* of an operation $o_{i,j} \in O_j$ on resource $m \in M_{i,j}$ is given by $C_{i,j,m} = S_{i,j} + p_{i,j,m}$.
- The completion time of operation $o_{i,j} \in O_j$ on all its resources is given by $C_{i,j} = S_{i,j} + p_{i,j} = \max_{m \in M_{i,j}}(C_{i,j,m})$.
- Operations belonging to the same job have to be performed in the order that is specified by the route selection. So, $C_{i,j} \leq S_{i+1,j}$ has to be fulfilled for all $o_{i,j} \in R$ with $i < \left|R_j\right|$.
- The first operation $o_{1,j} \in R_j$ of each job cannot be processed before its release date, so $S_{1,j} \geq r_j$ must hold for all $j \in J$.
- For two operations $o_{i,j}, o_{k,l} \in R$ with $M_{i,j} \cap M_{k,l} \neq \emptyset$, having $S_{i,j} = S_{k,l}$ in general means that a batch is created, which is only allowed in the cases specified in Section 4.1.3. In all other cases, for all common resources $m \in M_{i,j} \cap M_{k,l}$ of two operations $o_{i,j}, o_{k,l} \in R$, either $S_{i,j} + p_{i,j,m} + s(\sigma_{i,j,m}, \sigma_{k,l,m}) \leq S_{k,l}$ or $S_{k,l} + p_{k,l,m} + s(\sigma_{k,l,m}, \sigma_{i,j,m}) \leq S_{i,j}$ must hold.
- The resource acquisition constraint now imposes that, for an acquisition $a = (m, o_{k,j}) \in A_{i,j}$ of a resource $m \in M_{i,j}$ at an *acquisition operation* $o_{i,j}$ with a corresponding *release operation* $o_{k,j}$, there must not be any other operation that uses m between $S_{i,j}$ and $C_{k,j,m}$. So, for all operations $o_{x,y} \in O$ $(\neq o_{i,j}, \neq o_{k,j})$ with $m \in M_{x,y}$, either $S_{x,y} \geq C_{k,j,m}$ or $C_{x,y} \leq S_{i,j}$ must hold.

4.1.3 Batchable Resources

In this section, the problem is extended to include batching constraints. The possibility of modeling in details batching machines is one of the contributions of this thesis. In Knopp (2016), batching is restricted to movable components of one operation. In this thesis, this restriction is relaxed, and batching is considered for movable components of any length. Formally, we extend the problem as follows. We are given a set of *recipes* $\mathcal{R}$. For each recipe

$q \in \mathcal{R}$, we are given a maximal batching capacity $b_q^{max} \in \mathbb{N}_{>0}$ and a minimal batching capacity $b_q^{min} \in \mathbb{N}_{>0}$. A recipe $q_v \in \mathcal{R}$ is assigned to each operation $v \in O$. All operations $v \in O$ that are assigned to the same recipe $q \in \mathcal{R}$ require an identical set of resources M_v, and must have the same processing duration $p_{v,m}$ and the same setup family $f_{v,m}$ for each resource $m \in M_v$. As the movable components model a sequence of operations within complex machines, it is required that all recipes that are assigned to nodes of the same component to have same maximal and minimal batching capacities b_q^{max} and b_q^{min}. Also, it is required that recipes of operations belonging to the same movable components are distinct.

A completely characterized schedule is obtained after the partition of all operations into batches, the assignment of the formed batches to qualified resources, their sequencing and finally the assignment of start times to batches. The three first decisions can all be represented by a family of batches $\mathcal{B} = \{B_{m,x} |\}_{m \in M, x \in \{1,\dots,t_m\}}$, where $B_{m,x}$ is the batch sequenced at position x on resource m and $t_m \in \{0, \dots, |O_m|\}$ is the number of batches assigned to resource m. As a batch can be processed by several resources, we require $B \cap B' = \emptyset, \forall B, B' \in \mathcal{B}$. Only operations of the same recipe can be processed at the same time on the same resource. So, for two operations $o_{i,j}, o_{k,l} \in R$ with $q_{i,j} = q_{k,l}$, we relax the resource sequencing constraints of Section 4.1.2 and allow $S_{i,j} = S_{k,l}$. If $S_{i,j} \neq S_{k,l}$, the resource sequencing constraints from Section 4.1.2 have to be fulfilled. Also, as only operations with the same recipe can be processed in the same batch, let q_B denote the associated batch family to batch $B \in \mathcal{B}$, i.e., $q_B = q_{i,j}\ \forall o_{i,j} \in B$. Let $\sigma_B = \sum_{o_{i,j} \in B} \sigma_j$ denote the size of batch B. Any formed batch $B \in \mathcal{B}$ must respect the batching capacity of the machine to which it is assigned. Thus, we require $\sigma_B \leq b_{q_B}^{max}\ \forall B \in \mathcal{B}$. The size of any batch should also respect minimum batch size constraints, i.e., $\sigma_B \geq b_{q_B}^{min}\ \forall B \in \mathcal{B}$.

When considering batching on movable components larger than one node, additional conditions must be satisfied for a solution to be feasible. If $o_i \in C = (o_1, \dots, o_i, \dots, o_{n_c})$ and $o'_i \in C' = (o'_1, \dots, o'_i, \dots, o'_{n_{c'}})$ belong to the same batch $B \in \mathcal{B}$, it should be ensure that $\forall k = \{1, \dots, n_c\}$, $\exists B \in \mathcal{B}$ such that $\{o_k, o'_k\} \subseteq B$. Considering batching machines also lead the relaxation of the resource acquisition constraints defined in Section 4.1.2. It was prescribed that, for an acquisition $a = (m, o_{k,j}) \in A_{i,j}$ of a resource $m \in M_{i,j}$ at an acquiring operation $o_{i,j}$ with a corresponding release operation $o_{k,j}$, there must not be any other operation that uses m between $S_{i,j}$ and $C_{k,j,m}$. When m is a batching resource, it is allowed for other operations $o_{x,y} \in O\ (\neq o_{i,j}, \neq o_{k,j})$ with $m \in M_{x,y}$ to use m between $S_{i,j}$ and $C_{k,j,m}$ provided that $S_{i,j} = S_{x,y}$ or $S_{k,j} = S_{x,y}$. In other words, operations $o_{x,y}$ must be batched either with $o_{i,j}$ or $o_{k,j}$.

4.1.4 Unavailability Periods

This section extends the previous problem description in order to include resource availability constraints. These constraints model periods during which a resource is unavailable to process operations like those resulting from preventive or curative maintenance operations. Formally, we extend the problem as follows. For each resource $m \in \mathcal{M}$, we are given a set of fixed unavailability periods U_m. Each unavailability period $u_{m,l} \in U_m$ has a fixed start date $S_{m,l}$ and a fixed end date $C_{m,l}$. Regarding the possibility for an operation to be interrupted

by an unavailability period, the non-preemptive case is considered. This models the situation where an operation can be interrupted neither by another operation nor by an unavailability period. Thus, for each resource $m \in \mathcal{M}$, for each operation $o_{i,j} \in R$ such that $m \in M_{i,j}$ and for each $u_{m,l} \in U_m$, $S_{i,j} + p_{i,j,m} \leq S_{m,l}$ or $C_{m,l} \leq S_{i,j}$ must hold.

4.1.5 Minimum and Maximum Time Lags

In Chapter 2, two types of times lags are described: *Minimum time lags* and *maximum time lags*. As these time lags are related to operations of the same job, it is possible to extend the route graph by including them. However, to avoid enlarging the disjunctive graph of the problem that is obtained by aggregating all route graphs of all the jobs, these constraints are stored in an auxiliary data structure. To illustrate the different situations that can be modeled by these constraints, a toy example of a route graph is given in Figure 4.2. The job has two steps, the first one to be performed on a wet bench and the second one on furnaces (F_1, F_2 and F_3). The step to be performed on a wet bench is modeled with three movable components. The two movable components with two nodes represent the case where the wet bench is modeled in detail, and this is similar to the case considered in Figure 4.1(a). The third movable component with one node represents the case where the wet bench is considered as a resource. As a unique modeling of complex machines must be considered, this example is given only to illustrate the integration of time lags.

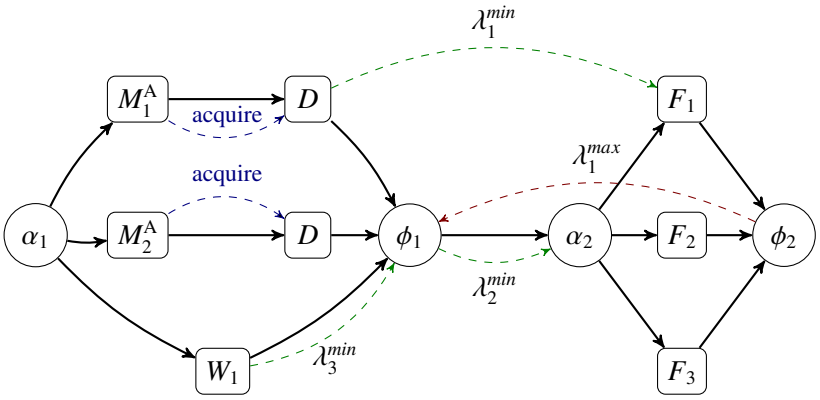


Figure 4.2 – *Route graph with time lags*

A minimum time lag specifies a minimum delay between the execution of two operations of the same job (Zhang (2010)), not necessarily consecutive. Recall that the set of operations for a job $j \in J$ is denoted as $O_j = \{o_{1,j}, \ldots, o_{|O_j|,j}\}$. Formally, let us consider a set of minimum time lags $\mathcal{L}^{min} \subset J \times \mathbb{N}^3$. The components of a time lag $\lambda = (j, k, l, d) \in \mathcal{L}^{min}$ have the following meaning: $j \in J$ identifies the job; $k, l \in \mathbb{N}$ with $1 \leq k < l \leq |O_j|$ identify operations $o_{k,j}$ and $o_{l,j} \in O_j$; $d \in \mathbb{N}_{\geq 0}$ identifies the minimum time lag duration between the start time of $o_{k,j}$ and the start time of $o_{l,j}$. Thus, for each minimum time lag

$\lambda = (j, k, l, d) \in \mathcal{L}^{min}$, $S_{l,j} \geq S_{k,j} + d$ must hold. Such a time lag may for instance arise if the transfer of a job from one machine to the next requires a given transportation time (λ_1^{min} in Figure 4.2). These constraints can also be used to model minimum delays that are imposed for process considerations (λ_2^{min} in Figure 4.2). Finally, these constraints are also used to differentiate between the cycle time of an operation and the *actual* duration of this operation on its resources(λ_3^{min} in Figure 4.2). The use of minimum time lags to model this last situation is illustrated in the data-driven analytical modeling proposed in Section 4.5.1.

Regarding maximum time lags, they model in the industrial context time windows that are defined to prevent risks to the product quality. Such constraints specify the maximum period of time between the start (or end) of a step and the start (or end) of another step in the route of a job. In Figure 4.2, λ_1^{max} models a maximum time lag between the end of two successive steps in the route of a job. Formally, let us consider a set of maximum time lag constraints $\mathcal{L}^{max} \subset J \times \mathbb{Z}^4 \times \mathbb{R}_{>0}$. The components of a time lag $(j, k, l, d, \gamma, c) \in \mathcal{L}^{max}$ have the following meaning: $j \in J$ identifies the job; $k, l \in \mathbb{N}$, with $1 \leq k < l \leq \left|O_j\right|$, identify separator operations $o_{k,j}$ and $o_{l,j} \in O_j$; $d \in \mathbb{N}_{\geq 0}$ identifies a maximum time lag duration; $\gamma \in \mathbb{N}_{\geq 0}$, with $\gamma \geq d$, identifies an ultimate time lag duration; and $c \in \mathbb{R}_{>0}$ identifies the cost of a time lag violation. When considering maximum time lags as hard constraints, the two last parameters γ and c are not relevant. In such case, for each $\lambda = (j, k, l, d, \gamma, c) \in \mathcal{L}^{max}$, $S_{l,j} \leq S_{k,j} + d$ must hold so that the schedule is feasible. However, within the studied industrial context, maximum time lags are considered as soft constraints. This modeling is motivated in Section 2.4.5.

4.2 Formal Modeling of the Problem Criteria

As already stated, a schedule is completely characterized by providing for each job $j \in \mathcal{J}$ a route selection $R_j \subset O_j$ and start dates $S_{i,j} \in \mathbb{Z}$ for all selected operations $o_{i,j} \in R_j$. The route selection R_j must describe a path $(\alpha_j = o_{1,j}, \ldots, o_{|R_j|,j} = \phi_j)$ in the route graph G_j. We denote the resources and processing durations related to this selection as $M_{i,j}$ and $p_{i,j}$, respectively. The disjoint union $R = R_1 \dot{\cup} R_2 \ldots \dot{\cup} R_{|J|}$ denotes all selected operations. The completion time of a job $j \in J$ is the completion time of its last operation, i.e., $C_j = C_{|O_j|,j} = C_{\phi_j}$. The quality of a schedule is measured by the given objective function. An *objective function* is a function $f : \mathbb{R}^{|O|} \to \mathbb{R}$ that maps tuples of operation start times to a real number.

In the scope of this chapter, we want to optimize all the criteria defined and industrially motivated in Chapter 2. As the problem described in this chapter is a generalization of the problem in Chapter 3, the regular objective functions are also handled. Section 4.2.1 formalizes the different criteria that translate actual operational performances: Cycle time, machine throughput and utilization. Section 4.2.2 recalls the modeling, proposed by Knopp (2016), where the maximum time lag constraints become an objective of minimizing maximum time lag violations. Section 4.2.3 introduces our new modeling for handling the satisfaction of production targets.

4.2.1 Performance Criteria

First, let us consider the criteria that were already defined in previous works. The weighted flow factor, also called *average X-Factor*, is used in Artigues et al. (2006), Yugma et al. (2012), Bitar (2015) and Knopp (2016). This criterion is designed to reduce the cycle time and the WIP in the considered optimization scope. We are given ϵ_j a minimum possible time to process all operations of a job $j \in \mathcal{J}$. The flow factor of a lot is its actual route duration divided by its theoretical route duration. Now, the weighted flow factor f_1 that we want to minimize is the weighted average of all flow factors as shown in (4.1). In Artigues et al. (2006), Yugma et al. (2012) and Knopp (2016), instead of considering all jobs as in Bitar (2015), only those that are completed within the scheduling horizon are taken into consideration. When doing this, poor operational performances may be obtained. The objective of this criteria is to contribute to competitive cycle times in the whole fab by minimizing as much as possible the number of *static jobs*, jobs that stay for a long time within the optimization scope. When considering only jobs that are completed within the horizon, a solution where a large number of static jobs are completed outside the scheduling horizon may be evaluated as a solution with good quality. Also, by allowing release dates with a negative value, the actual route duration is not restricted by the beginning of the scheduling horizon.

$$f_1 = \frac{1}{\sum_{j\in\mathcal{J}} \omega_j} \sum_{j\in\mathcal{J}} \frac{\omega_j(C_j - r_j)}{\epsilon_j} \tag{4.1}$$

Contrary to the weighted flow factor, the remaining criteria defined in this section depend on the scheduling horizon. We consider, w.l.o.g., the time 0 as the beginning of the horizon and H its *end time*. H is also used as the duration of the scheduling horizon. When considering a scheduling horizon, an operation $o_{i,j} \in R$ in a given schedule may be completed before the end of the horizon, i.e., $C_{i,j} \leq H$, be started after the end of the horizon , i.e., $S_{i,j} \geq H$ or the end of the horizon may fall within its processing, i.e., $S_{i,j} < H \wedge C_{i,j} > H$. To consider these different situations, a *completion rate* $\theta_{i,j}$ is defined in (4.2) for each operation $o_{i,j} \in R$. Assuming that σ_j denotes the size of a job $j \in J$ in number of wafers, the weighted number of moves f_2 can be computed as in (4.3).

$$\theta_{i,j} = \begin{cases} \frac{min(p_{i,j}, H - C_{i,j})}{p_{i,j}} & \text{if } C_{i,j} \leq H, \\ 0 & \text{else} \end{cases} \tag{4.2}$$

$$f_2 = \sum_{o_{i,j}\in R} \omega_j \sigma_j \theta_{i,j} \tag{4.3}$$

The weighted number of moves expresses the objective of maximizing the overall throughput of the machines considered in the optimization scope. All solutions where the same operations are processed within the scheduling horizon are considered as equivalent. When considering a rolling horizon framework, some solutions may be considered better than others in the long term. Between two solutions in which the same weighted number of moves are started, it is better to choose the one with a larger number of moves and where

more important jobs are performed at the beginning of the horizon. The discounted weighted number of moves is proposed to cope with these considerations. Let us define in (4.4) $\alpha_{i,j}$ as the *discount rate* of operation $o_{i,j} \in R$. This rate expresses the idea that the further the completion time of an operation, the lower its contribution to the solution quality. Using this rate, the discounted weighted number of moves f_3 is defined in (4.5).

$$\alpha_{i,j} = 1 - \frac{C_{i,j}}{H} \tag{4.4}$$

$$f_3 = \sum_{o_{i,j} \in R} \omega_j \sigma_j \alpha_{i,j} \theta_{i,j} \tag{4.5}$$

The last criterion defined here is the batching coefficient. This measure represents the average of the actual size of each batch divided by its maximal size. This criterion also depends on the scheduling horizon. Thus, let $\mathcal{B}^H = \{B \in \mathcal{B} \mid S_{i,j} < H, \forall o_{i,j} \in B\}$ denote the set of batches started before the end of the scheduling horizon. The batching coefficient f_4 of a schedule can be defined then as in (4.6)

$$f_4 = \frac{\sum_{B \in \mathcal{B}^H} \sigma_B}{\sum_{B \in \mathcal{B}^H} b_B} \tag{4.6}$$

4.2.2 Maximum Time Lag Violation

Considering maximum time lags as soft constraints was motivated in Section 2.4.5. In Section 4.1.5, it is considered that a set of maximum time lags constraints $\mathcal{L}^{max} \subset J \times \mathbb{Z}^4 \times \mathbb{R}_{>0}$ is given. As already described, the components of a time lag $\lambda = (j, k, l, d, \gamma, c) \in \mathcal{L}^{max}$ have the following meaning: $j \in J$ identifies the job; $k, l \in \mathbb{N}$ with $1 \leq k < l \leq \left|O_j\right|$ identify operations $o_{k,j}$ and $o_{l,j} \in O_j$; $d \in \mathbb{N}_{\geq 0}$ identifies a maximum time lag duration; $\gamma \in \mathbb{N}_{\geq 0}$ with $\gamma \geq d$ identifies an ultimate time lag duration; and $c \in \mathbb{R}_{>0}$ identifies the cost of a time lag violation.

Now consider a feasible schedule with start dates $S_{i,j} \in \mathbb{N}$ that are given for all scheduled operations $o_{i,j} \in R$. For each time lag $\lambda = (j, k, l, d, \gamma, c) \in \mathcal{L}^{max}$, its delay $L = S_{l,j} - S_{k,j}$ is and its *violation severity* is defined as

$$V_\lambda = \begin{cases} 0 & \text{if } L \leq d, \\ c & \text{if } L > \gamma, \\ \frac{(L-d)^2}{(\gamma-d)^2} \cdot c & \text{else.} \end{cases} \tag{4.7}$$

Note that, in cases where the initial operation of a time lag refers to an operation that has started its processing in the past, $k \leq 0$ is allowed and we assume for notational consistency that (though the operation is not part of the considered scheduling problem) its start date is still given by $S_{k,j}$. Overall, the *total maximum time lag violation severity* f_5 to minimize is defined in 4.8.

$$f_5 = \sum_{\lambda \in \mathcal{L}^{max}} V_\lambda. \tag{4.8}$$

This definition comprises the cases described in Section 2.4.5. For reworkable lots, we have $d = \gamma$, which corresponds to a constant violation cost regardless of the duration of the delay. For non-reworkable lots, we have $d < \gamma$, which corresponds to a cost that increases quadratically with the duration of the delay as long as the delay does not exceed the ultimate duration which reflects the very probable scrapping of the lot. Note that this objective function is not regular, since advancing an operation that starts a maximum time lag could increase the total maximum time lag violation severity. Note also that this objective function is computed for all the operations, not only those that are processed within the given scheduling horizon.

4.2.3 Production Target Satisfaction

As motivated in Section 2.3.11, production targets, also called daily move targets, are extensively used as a means of production control in semiconductor manufacturing. Their goal is to smooth the differences between the WIP level of a production stage and its fixed target. When computed accurately, the satisfaction of production targets allows the local scheduling decisions to be consistent with global objectives at the fab level. The formulation given here, instead of being specific to semiconductor manufacturing, may be used to any situation where it is felt important to make sure that the produced schedules stick to a production plan determined at a higher level.

We are given a set $\mathcal{T}$ of production targets. Let a given set $T_{i,j}$ denote the set of production targets to which an operation $o_{i,j} \in R$ contributes. Note that it is possible to have $T_{ij} = \emptyset$, which means that the operation o_{ij} does not contribute to any production target. It is also possible that $|T_{ij}| > 1$, which means that operation o_{ij} contributes to more than one production target. To each production target $\tau \in \mathcal{T}$, we associate a requested volume $D_\tau \in \mathbf{N}$ and a weight w_τ. We assume that, without loss of generality, $\sum_{\tau \in \mathcal{T}} w_\tau = 1$. An operation is defined as contributing to its associated production targets if it starts its processing within the scheduling horizon. Let O^H be the set of operations that are started within the horizon. Given a feasible schedule, $P_\tau = \sum_{o_{i,j} \in O^H, \tau \in T_{i,j}} \sigma_j$ defines the produced volume for each target $\tau \in \mathcal{T}$.

Given a feasible schedule, $X_\tau = \frac{P_\tau}{D_\tau}$ denotes the completion rate of a production target $\tau \in \mathcal{T}$. It is possible to have different levels of satisfaction for the same completion rate of two different production targets. A production target may define a minimum quantity to produce, but it may also define a quantity that is desirable to reach but not to exceed. So, instead of only using the completion rate X_τ, the expected satisfaction Y_τ of the decision maker that depends on the completion rate X_τ can be modeled. In the industrial application, as targets only define minimal quantities to produce, the satisfaction can be computed as in (4.9).

$$Y_\tau = \begin{cases} X_\tau & \text{if } X_\tau \leq 1, \\ 1 & \text{else} \end{cases} \tag{4.9}$$

After defining the satisfaction level of a production target, it remains to define how to evaluate the satisfaction of multiple production targets in a schedule. It appears that decision makers may require a high level of global satisfaction and, at the same time, may want to balance the satisfaction levels of the different production targets. As a first modeling, the weighted sum of the satisfaction of the production targets $\sum_{\tau\in\mathcal{T}} w_\tau Y_\tau$ could be used to express the idea of overall satisfaction. Fully compensatory, this modeling does not ensure to correctly balance targets, which can lead to unfair solutions. Even if global satisfaction is maximized, it could happen that the satisfaction of some targets will be far from the average. Even worse, this indicator is not very discriminating since multiple solutions may be equivalent to the same average satisfaction level, although the balance can be very different.

Balancing means satisfying as fairly as possible. This often corresponds to minimizing the difference between the most and the least satisfied targets. Maximizing the satisfaction of the least satisfied target $\max \min_{\tau\in\mathcal{T}}(Y_\tau)$ is one of the usual ways of modeling satisfaction balance. This modeling is the extreme opposite of the first one as it only focuses on balancing. This focus on balancing may come with a significant cost in total satisfaction. Even worse, the obtained criterion is not very discriminating since multiple solutions remain equivalent from the point of view of the worst satisfaction level analysis. With the same minimum satisfaction level, the solution with the largest overall satisfaction level may not be proposed.

The two models above have the merit of being easy to understand. However, they either focus on overall satisfaction or on balancing. Here, we present a new criterion. We consider three requirements that the criterion to be proposed should meet. First, it should produce values that can be easily interpreted by decision makers. Second, the criterion must take into consideration the weights of the production targets. Finally, it should consider balancing and overall satisfaction simultaneously. It should also be flexible. In other words, depending on the context, the decision maker can prioritize overall satisfaction or balancing, without ignoring the other. This should be done through a parameter in the indicator for which a value is given as input, depending on the context and the decision maker preferences.

The proposed criterion is called *TSI*, which stands for *Target Satisfaction Indicator*, and is computed using (4.10). This indicator has the form of what is called *weighted power mean*. Below, we show that this indicator meets the three requirements listed above. It is also shown that this indicator is a generalization of the two first models.

$$\text{TSI} = \left(\sum_{\tau\in\mathcal{T}} w_\tau Y_\tau^\alpha \right)^{1/\alpha} \tag{4.10}$$

Regarding the first requirement, it is easy to verify that the weights w_τ and the satisfaction levels Y_τ of the production targets are included in 4.10. As a weighted power mean, we can quickly check that the second requirement is also satisfied. It is proven that functions, in the form of weighted power mean, always produce values that lie between the smallest and the

largest of the Y_τ values. There is no case where the indicator has a value smaller than the satisfaction level of the least satisfied target, nor a case where the indicator has a value larger than the satisfaction level of the most satisfied target. Also, as the individual satisfaction levels Y_τ lie between 0% and 100%, TSI is also in this interval. Finally, let us show that TSI satisfies the last requirement. Before this, let us show that TSI is a generalization of the two previous models. The weighted sum can be obtained by considering $\alpha = 1$. It is less obvious to see that MaxMin modeling can be obtained by giving some values for α. We have in the literature the result shown in (4.11). This means that TSI is equivalent to MaxMin modeling if for a sufficiently small parameter α. Less relevant in the case of production target satisfaction, it is interesting to mention that the maximum satisfaction level is returned if parameter α is chosen large enough, i.e. $\lim_{\alpha\to+\infty} TSI = max\{Y_1, ..., Y_\tau, ..., Y_p\}$.

$$\lim_{\alpha\to-\infty} TSI = min\{Y_1, ..., Y_\tau, ..., Y_p\} \tag{4.11}$$

By varying α, it is then possible to move from a focus on overall satisfaction to a focus on balancing. The decision maker can, by giving intermediate and less extreme values to α, choose to focus more on overall satisfaction or balancing, depending on the context. More important, by a right choice of the parameter α, TSI can discriminate solutions that are equivalent on total satisfaction, resp., balancing, but that are different on balancing, resp., total satisfaction. To illustrate the flexibility that TSI offers, let us consider two toy examples. In the first example, there are two production targets with the same importance. Let us assume there are two solutions S_1 and S_2. In S_1, the satisfaction of the two production targets is, respectively, 100% and 0%. In S_2, the satisfaction level is 50% for each of the production targets. S_1 and S_2 are equivalent if the weighted sum is used with total satisfaction of 50%. This does not meet the decision maker preferences as S_2 is preferred since it is better balanced. If TSI is computed for the two solutions with $\alpha = 0.5$, it will be respectively 25.2% for S_1 and 50% for S_2, i.e., S_2 is better and corresponds to the decision maker preferences.

To illustrate the behavior of TSI depending on α, let us consider in Figure 4.3 another toy example with two production targets. Let us assume that one of the production targets always has a satisfaction level of 50%. The satisfaction levels of the other production target are represented in the x-axis. The y-axis corresponds to values of TSI. The different curves correspond to the different values that are given to parameter α.

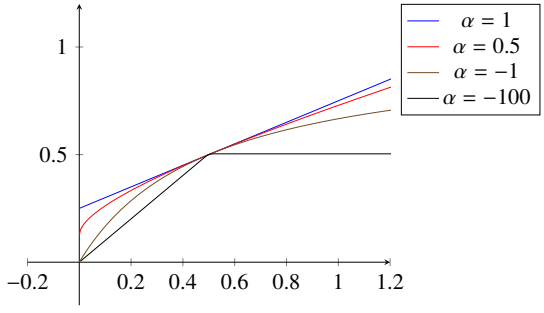


Figure 4.3 – *TSI behavior with* $\alpha \leq 1$

4.3 Extending the Batch-Oblivious Approach

We tackle the problem described in Sections 4.1 and 4.2 by extending the batch-oblivious approach. This section generalizes the batch-oblivious conjunctive graph representation of Section 3.2 such that all properties of the scheduling problem described in Section 4.1 are taken into account. We provide a self-contained description by introducing the notations and detailing the graph. Section 4.3.1 describes the extended graph that considers the route graph concept and all the constraints handled in Chapter 3, except batching constraints. Section 4.3.2 details the integration of minimum time lags and unavailability periods when computing the start time of a node. Section 4.3.3 describes the proposed approach to handle the batching constraints, especially for complex machines that are modeled in detail. Finally, Section 4.3.4 describes how minimum batch size constraints are handled.

4.3.1 Extended Batch-Oblivious Conjunctive Graph

The disjunctive graph representing the problem to solve is obtained by combining all the route graphs of jobs in the problem, in addition to the classical start and end dummy nodes. Let us now introduce an extended batch-oblivious conjunctive graph representation. Schedules are represented as a directed acyclic graph $G = (V, E)$ with the set of nodes $V = O \cup \{0, *\}$ that corresponds to the set of operations O plus an artificial start node 0 and an artificial end node $*$. Note that all operations can be considered to be part of the conjunctive graph, even if they are not scheduled. We denote the edges involved in the sequencing of operations on resource $m \in M$ by $E_m^R \subset E$. Analogously, edges involved in the sequencing of the operations of a job $j \in J$ are denoted by $E_j^J \subset E$. The disjoint union $E = \dot{\bigcup}_{j \in J} E_j^J \ \dot{\cup} \ \dot{\bigcup}_{m \in M} E_m^R$ of these paths yields all edges of the graph. One corresponds to the route of its job and all others to the sequencing of its assigned resources. For a node $v \in R$, we denote its route successor by $r(v) \in V \setminus \{0\}$ and the set of its resource successors by $m(v) \subset V \setminus \{0\}$. For each node $v \in V$, $succ(v)$ denotes the set of all the successors, i.e., $succ(v) = \{r(v)\} \cup m(v)$. $out(v)$ denotes the number of the successors, i.e., $out(v) = |succ(v)|$ Analogously, its predecessors are denoted by $r^{-1}(v) \in V \setminus \{*\}$ and $m^{-1}(v) \subset V \setminus \{*\}$. The set of all predecessors of v is denoted by $pred(v)$ and its cardinality by $in(v)$. Each unscheduled operation $u \in O \setminus R$ corresponds to a disconnected node in the conjunctive graph, i.e. $in(u) = out(u) = 0$. The artificial start node 0 has $|J| + |M|$ outgoing edges and no incoming edges. The artificial end node $*$ has $|J| + |M|$ incoming edges and no outgoing edges. Overall, the graph has $|E| = |J| + |M| + |R| + \sum_{v \in R} |M_v|$ edges.

Start dates $S_v \in \mathbb{Z}$ of operations $v \in O$ are determined from conjunctive graphs using the longest paths. Some adaptations are needed for our generalized problem. Again, a weight $l_{u,v}$ is associated to each edge $(u, v) \in E$ to ensure a minimum duration between the start dates of adjacent operations, i.e. $S_v \geq S_u + l_{u,v}$ is required for each edge $(u, v) \in E$. For each operation $v \in O$, its start date S_v is determined by the distance $L(0, v)$ of the longest path from the artificial start node 0 to node v. To respect the constraints given in our scheduling problem, edge weights are defined as follows. For edges $(0, o_{1,j}) \in E_j^J$ connecting the

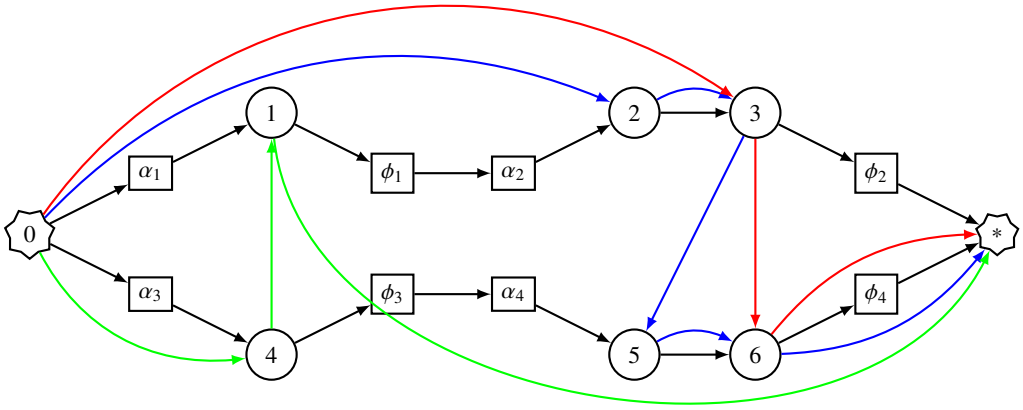


Figure 4.4 – *Example of generalized batch-oblivious graph*

artificial start node 0 with the first operation $o_{1,j}$ of job $j \in J$, the edge weight is set to the release date $max(0, r_j)$ of job j. For edges $(0, o_m) \in E_m^R$ connecting the artificial start node 0 with the first operation o_m sequenced on resource $m \in M$, the edge weight is set to zero. For route edges $(v, r(v)) \in E$ with $v \neq 0$, the edge weight is set to the processing duration p_v of operation v. For resource edges $(v, w) \in E_m^R$ with $v \neq 0$ and $w \in m(v)$, the edge weight is set either to zero when both operations are in the same batch, or to $p_{v,m} + s(f_{v,m}, f_{w,m})$, i.e., the sum of the resource-dependent processing duration $p_{v,m}$ of operation v on resource m and the sequence-dependent setup time $s(f_{v,m}, f_{w,m})$ between operation v and operation w on machine m. To avoid making the graph larger and more complex, minimum time lags and unavailability periods are not modeled explicitly in the graph. They are considered only during the start date computation, using auxiliary data structures.

When modeling non-batching machines in detail, we include the resource acquisition constraint using Property 4.1. It is then required that for each resource acquisition $a = (m, v) \in A_u$ of each operation $u \in R$, there must be an edge $(u, v) \in E_m^R$. This property must be preserved if the graph is modified. After examining the example below, this property is reformulated and generalized in Property 4.2 to take into account batching machines.

Property 4.1. *Let $a = (m, v) \in A_u$ be the acquisition of a resource m at an operation $u \in R_j$ with a corresponding release operation $v \in R_j$. For all operations $w \in R$ ($\neq u$, $\neq v$) with $m \in M_w$ there must not be any path in the disjunctive graph that has the form $(u, \ldots, w, \ldots, v)$. Consequently, v must directly follow u in the sequencing of operations scheduled on resource m. Thus, there must be an edge $(u, v) \in E_m^R$.*

Let us consider an example of two jobs (J_1, J_2) with two steps each. The first steps of both jobs require different recipes on a furnace which is modeled as a resource F_1. The second steps of the two jobs can be batched together on a wet bench described in Section 2.2.1 that

is modeled in detail. The processing part of this machine includes two processing resources (M_1 or M_2) and a drying resource (D). The jobs to be scheduled require the same short process, i.e., through one of the modules (M_1 or M_2) and the dryer (D). The maximum batching capacity of this machine is two jobs. Figure 4.4 represents a generalized batch-oblivious representation of a solution of this problem. The nodes $\alpha_\square$ and $\phi_\square$ represent the separator nodes of the two jobs while 0 and $*$ represent the artificial start and end nodes. The black arcs represent routing constraints; the blue arcs represent the sequencing on module M_1; the red arcs represent the sequencing on the dryer D and the green arcs represent the sequencing on furnace F_1. Both jobs J_1 and J_2 use M_1 and D. As there is no internal buffer, resource acquisition constraints are used to model blocking constraints. Operation 2 acquires M_1 and will only release it when operation 3 can start its processing and thus the processing of operation 5. In the same way, operation 5 acquires M_1 and release it at the processing start of operation 6. The resulting graph, representing the blocking constraints, is similar to the *alternative graph* proposed by Mascis and Pacciarelli (2002).

Recall that it was required in Property 4.1 that there must be an edge $(u, v) \in E_m^R$ for each acquisition constraint $a = (m, v) \in A_u$. If there is a compatible operation w with operation u and if this requirement is not relaxed when considering batching, w can never be batched with u as it can never be a direct successor of u on m. If we consider the example of Figure 4.4, the acquisition constraint impedes operation 5 to be batched with operation 2. If a batching resource is acquired, the requirement above is redefined in Property 4.2. It is then required that for each acquisition $a = (m, v) \in A_u$ of batching resource m of each operation $u \in R$, it is allowed to sequence operations between u and v on resource m provided that all these operations are batched either with u or v.

Property 4.2. *Let $a = (m, o_{k,j}) \in A_{i,j}$ be the acquisition of a resource m at an operation $o_{i,j} \in R_j$ with a corresponding release operation $o_{k,j} \in R_j$. If there is a path $P = (u, w_1, \ldots, w_i, \ldots, w_n, v) \subset E_m^R$, it is required that all w_i such that $i \in \{1, \ldots, n\}$ are batched either with u or v.*

4.3.2 Start Time Computation

To compute the start times and to determine the weights of resource edges, the nodes of the graph are traversed in topological order. In Chapter 3, the computation of the start time of an operation is based on the already computed start times of the direct predecessors and the determined weights of the incoming edges. When dealing with the problem described here, it is necessary to include additional constraints. First, the computation of the start time should take into account the minimum time lags. For each node v and for each $\lambda = (j, u, v, d) \in \mathcal{L}^{min}$, $S_v \geq S_u + d$ must hold. In the remainder of this chapter, r_v defines the job availability time of operation v and is computed as in (4.12).

$$r_v = \max(S_{r^{-1}(v)} + l_{r^{-1}(v),v}, \max_{(j,u,v,d)\in\mathcal{L}^{min}} (S_u + d)) \tag{4.12}$$

In addition to the minimum time lags, availability constraints are not also explicitly modeled in the structure of the conjunctive graph. In the industrial context, a resource can be unavailable to process production operations because of preventive maintenance, curative maintenance and quality tasks. Quality tasks are described in Section 2.3.10, where it is specified that only calendar quality tasks are considered in this thesis. These quality tasks must be performed on the concerned machines during a given shift. This can be modeled as what is called in the literature flexible unavailability periods (Gao et al. (2006), Azem et al. (2012)). This modeling corresponds to the situation where unavailability periods also have decision variables, i.e., each unavailability period has to be scheduled in a given time window. Instead of this modeling, each quality task is considered as a production job with a route containing one step that is only qualified on the machine on which the quality task must be performed. It is then assumed that these tasks are included in the set of given jobs $\mathcal{J}$. Regarding the fact that these tasks must be performed within a given time window $I_j = [s_j, e_j]$, this is handled through maximum time lag constraints. It is also assumed that the set of maximum time lags $\mathcal{L}^{max}$ contains the time lags concerning quality tasks. Note that in this case, as the route of a job modeling a quality task has only one step, the initial operation u of a time lag $\lambda = (j, u, v, d, \gamma, c) \in \mathcal{L}^{max}$ cannot refer to an actual step of the job. We allow $u = 0$ and assume its completion time equal to the time window start time, i.e., $C_u = s_j$. The maximum time lag duration $d \in \mathbb{N}_{\geq 0}$ and the ultimate time lag duration $\gamma \in \mathbb{N}_{\geq 0}$ are equal to the length of the time window, i.e., $d = \gamma = e_j - s_j$. It is assumed that the cost c of performing the quality task outside its assigned time window is given as input.

Resources can also be unavailable due to maintenance, either preventive or curative. As mentioned in Section 4.1.4, each resource $m \in \mathcal{M}$ has a set of unavailability periods $U_m = \{u_{m,1}, \ldots, u_{m,l}, \ldots, u_{m,|U_m|}\}$. As the unavailability periods are not explicitly modeled in the graph, the classical longest path calculation algorithm is not sufficient to determine accurate start times for operation. Algorithm 4.1 is proposed to adjust the earliest start times of operations while taking into consideration the additional constraints in this chapter: Multiple resource requirement, batching constraints, sequence-dependent setup times and minimum time lags.

Algorithm 4.1 starts by computing the job availability time r_v of operation v as defined in (4.12). The initial earliest start time S_v is computed based on the start times of all the predecessors of v. Recall that the length $l_{u,v}$ of the resource edge (u, v) is set either to zero when both operations are included in the same batch, or to $p_{u,m} + s(f_{u,m}, f_{v,m})$, i.e., the sum of the resource-dependent processing duration p_u of operation u on resource m and the sequence-dependent setup time $s(f_{u,m}, f_{v,m})$ between operation u and operation v on machine m. Then, the algorithm adjusts the earliest start time by scanning the unavailability periods on all the resources in M_v performing operation v. The set of all unavailability periods U^{M_v} on all the resources in M_v is obtained by merging the sets of unavailability periods U_m where $m \in M_v$. It is assumed that the unavailability periods $u_l \in U^{M_v}$ are scanned in non-decreasing order of their start times S_l. The loop ends when $S_v + p_v \leq S_l$, which means that operation v can be started and completed without preemption.

Algorithm 4.1 Algorithm for computing the earliest start time of an operation

computeStartTime (v)
$r_v \leftarrow \max(S_{r^{-1}(v)} + l_{r^{-1}(v),v}, \max_{(j,u,v,d)\in\mathcal{L}^{min}}(S_u + d))$
$S_v \leftarrow \max(r_v, \max_{u\in m^{-1}(v)}(S_u + l_{u,v}))$
$U^{M_v} \leftarrow \bigcup_{m\in M_v} U_m$
for $u_l = [S_l, C_l] \in U^{M_v}$
  if $S_v < S_l$
    if $S_v + p_v \leq S_l$
      break
    $S_v \leftarrow C_l$
  else if $S_v \leq C_l$
    $S_v \leftarrow C_l$

4.3.3 Integration of Batching Constraints

The novelty brought by the batch-oblivious approach is a representation of batching decisions in conjunctive graphs which is non-intrusive regarding the structure of the graph. No dedicated batching nodes or additional arcs are introduced and the structure of the classical conjunctive graph remains as is. Batching decisions are modeled by adapting the weights of machine edges $(u, v) \in E_m^R$. The weight of a machine edge is set to zero if its adjacent operations should be processed in the same batch. Otherwise, the edge weight is set to $p_{v,m} + s(f_u, f_v)$, as in the non-batching case. To make sure that batching decisions are feasible, the proposed invariant (3.1) is recalled in Chapter 3. When the possibility of batching two adjacent operations u and v is studied, this invariant can be interpreted as follows: An operation v can be batched with its resource predecessor u, if the job of v is available before the already determined start time of u, i.e., $S_{r^{-1}(v)} + l_{r^{-1}(v),v} \leq S_u$. Different adaptations of the original batch-oblivious approach must be made to tackle the more general problem considered in this thesis.

The first generalization of the invariant (3.1) includes minimum time lag constraints. In the problem of Chapter 3, the job availability of a node v is equal to the completion time of the route predecessor $r^{-1}(v)$. With minimum time lag constraints, the job availability of an operation v is given by r_v, computed as shown in (4.12). The invariant is redefined as shown in (4.13).

$$(l_{u,v} = 0 \ \wedge \ S_u \geq r_v) \ \vee \ (l_{u,v} = p_{u,m_u} + s(f_u, f_v, m_u)) \tag{4.13}$$

An important extension of the original batch-oblivious approach is the possibility of modeling in detail complex machines, which is done through the route graph modeling. Instead of handling individual operations, the scheduling algorithm is constrained to handle sequences of operations in order to take the interdependence of internal resources of complex machines and their internal constraints into account. In Section 4.1.1, these sequences of operations are called movable components. In Knopp (2016), batching is restricted to movable compo-

nents consisting of one single node. In other words, batching machines cannot be modeled in detail. However, it is proposed in Knopp (2016) to generalize the approach to deal with operations that can require multiple batching machines at the same time. First, while adapting it to handle minimum time lag constraints, the proposed invariant is recalled. Then, the approach is generalized so that batching machines can be modeled in detail, i.e., batching is no longer restricted to movable components consisting of one single node.

The new invariant is shown in (4.14). As in the invariant (4.13), an operation $v \in R$ can extend an incomplete batch if its job availability r_v occurs before the already computed start time of the incomplete batch. When multiple resources are required, an additional condition must be fulfilled, i.e., $\left|m^{-1}(v)\right| = 1$. This means that an operation v requiring multiple resources can only extend a batch when the same operation u precedes v on all its required resources. The invariant must be fulfilled for all edges $(u, v) \in E_m^R$ where $u \in m^{-1}(v)$.

$$\left(l_{u,v} = 0 \ \wedge \ \left|m^{-1}(v)\right| = 1 \ \wedge \ S_u \geq r_v\right) \ \vee \ (l_{u,v} = p_{u,m} + s(f_{u,m}, f_{v,m})) \tag{4.14}$$

In the studied industrial context, all the machines within the optimization scope are batching machines. Some of these machines, specifically wet benches, are qualified as complex machines because it is not realistic to associate a fixed processing time to the operations they perform. The route graph modeling in Section 4.1.1 allows these machines to be modeled in detail. In addition to the different internal constraints that must be satisfied when constructing a schedule, an additional constraint that is related to the batching decision must be satisfied. The batches are formed outside of the machine. The jobs of the same batch stay together during all the operations that the batch goes through inside the machine, i.e., they will all use the same internal resources and simultaneously. If two nodes of two movable components are part of a batch, all the other nodes of the same two movable components, two by two, must be batched together. Instead of taking batching decisions regarding individual operations, these decisions are made for sequences of operations.

Previously, different conditions had to be satisfied when checking the possibility of batching two individual operations. The concerned operations must have the same recipe and satisfy the invariant. when considering a sequence of operations, in addition to the mentioned conditions, before batching two nodes of two distinct movable components, it should be ensured that the same decision can be taken for all the other remaining nodes of the same components. Definition 4.1 introduces the notion of *equivalence* of movable components. Similar to the notion of a recipe when considering individual nodes, this equivalence conditions the possibility of batching two sequences of operations. In brief, the two sequences should belong to different jobs, have the same length and each node at a given rank in a sequence must require the same recipe of the node with the same rank in the other sequence. This is a translation of the fact that all the jobs of the same batch stay together during all the operations that the batch goes through inside the machine.

Definition 4.1. *Let $C = (u_1, \ldots, u_l, \ldots, u_n) \subset E_j^J$ and $C' = (v_1, \ldots, v_{l'}, \ldots, v_{n'}) \subset E_{j'}^J$ be two movable components. They are said to be* **equivalent***, denoted by $C \equiv C'$ if:*

- $j \neq j'$,
- $n = n'$,
- $q_{u_l} = q_{v_{l'}}$, $\forall u_l \in C, v_{l'} \in C'$ *and* $l = l'$,
- $\forall \lambda = (j, u_l, u_{l+k}, d) \in \mathcal{L}^{min}$ *where* $u_l, u_{l+k} \in C$, $\exists \lambda = (j', v_{l'}, v_{l'+k}, d) \in \mathcal{L}^{min}$ *such that* $v_l, v_{l'+k} \in C'$.

One of the main innovations of the batch-oblivious approach is the fact that batching decisions are taken on the fly. During the traversal of the graph, the possibility of batching two adjacent nodes u and v is considered. If they cannot be batched together, and if the batch containing u is incomplete, the graph can be explored for compatible candidates using one of the strategies described in Section 3.3.2. This leads to the dynamic modification of the graph. If the static strategy is chosen, the batching decisions are only restricted to adjacent nodes, and the graph is not modified during the start time computation. However, when constraining the construction algorithm to take batching decisions regarding a sequence of operations instead of individual operations, it may become necessary to modify the graph even if the static strategy is chosen.

The proposed approach to handle batching decisions on complex machines is to give freedom to the construction algorithm regarding the first nodes in the movable components $u_1 \in C$ and constrain the construction algorithm to take the same batching decisions regarding the remaining nodes of the concerned movable components. While doing this, if $u_1 \in C$ and $v_1 \in C'$ are batched together and $C \equiv C'$, it should be ensured that it is always possible to modify the graph so that any two nodes $u_l \in C$ and $v_{l'} \in C'$, where $l = l' \neq 1$, can be batched together. More specifically, the graph should be modified so that u_l and $v_{l'}$ become adjacent on all the resource edges. Proposition 4.1 gives us this guarantee. Before stating it, we give below some results that make the proof of the proposition easier.

Lemma 4.1. *Let us consider two operations u and v with $M_u \cap M_v \neq \emptyset$. If u is sequenced before v on a resource $m \in M_u \cap M_v$ within a feasible schedule, it is also sequenced before v on all the other resources $M_u \cap M_v \setminus \{m\}$*

Proof. Otherwise, there will be a cycle, which contradicts the assumption of the solution feasibility. □

Lemma 4.2. *Let us consider two operations u and v with $q_u = q_v$, i.e., $M_u = M_v$. If u is sequenced before v in a feasible solution, it is possible to advance v and make it a direct successor of u on all the resources $m \in M_u$ without creating a cycle if $S_u \geq S_{r^{-1}(v)} + p_{r^{-1}(v)}$.*

Proof. To resequence v after u on all the resources $m \in M_u$, the resource arcs before and after v are deleted. Let $w \in m(u)$ the resource successors in the initial solution before resequencing v. A cycle is introduced after resequencing v if:

1. there was a path between v and u before the move, or
2. there was a path between $w \in m(u)$ and v before the move.

In case 1, if there was a path between v and u, then there was a path between $r(v)$ and u as all the resource arcs after v are deleted. As u is sequenced before v on all resources $m \in M_u$, this implies that there was a cycle in the initial solution, which contradicts the assumption of its feasibility.

In case 2, if there was a path between any resource successor $w \in m(u)$ and v, then there was a path between w and the $r^{-1}(v)$ as all the resource arcs before v are deleted. As $S_u \leq S_w \forall w \in m(u)$ and $p_{r^{-1}(v)} > 0$, this implies that $S_u \leq S_{r^{-1}(v)} < S_{r^{-1}(v)} + p_{r^{-1}(v)}$, a contradiction to the condition $S_u \geq S_{r^{-1}(v)} + p_{r^{-1}(v)}$. □

The proposed approach to handle batching decisions on complex machines is to give freedom to the construction algorithm regarding the first nodes in the movable components $u_1 \in C$. During the start time computation, if there is a node $v_1 \in C'$ such that $C \equiv C'$ and $S_{u_1} \geq r_v$, it is possible to consider the inclusion of v_1 and u_1 in the same batch. As $S_{u_1} \geq r_v$ implies that $S_{u_1} \geq S_{r^{-1}(v_1)} + p_{r^{-1}(v_1)}$, Lemma 4.2 ensures that no cycle is created when sequencing v_1 after u_1 on all the resources $m \in M_{u_1}$. If such a move is performed, it results that $\left|m^{-1}(v)\right| = 1$ which leads to the satisfaction of the invariant (4.14). If we consider the example in Figure 4.4, these results allow operation 5 to be sequenced after operation 2 so that they belong to the same batch. The new solution after moving operation 5 is shown in Figure 4.5. In the resulting conjunctive graph, we can see that, instead of having operation 3 as a direct successor of operation 2 to model the acquisition constraint, operation 5 is sequenced between these two operations which is allowed as operations 2 and 5 belong to the same batch. The same comment applies to the acquisition constraint between operations 5 and 6. However, before considering moving operation 5 after operation 2, it should be ensured that it is possible to batch the other nodes of the two movable components, i.e., operations 3 and 6. It should be ensured that it is possible to modify the graph when necessary and that the invariant (4.14) is satisfied. This guarantee is given by Proposition 4.1.

Proposition 4.1. *Let* $C = (u_1, \ldots, u_l, \ldots, u_n)$ *and* $C = (v_1, \ldots, v_{l'}, \ldots, v_n)$ *be two movable components such that* $C \equiv C'$. *If the two first operations in the two movable components* u_1 *and* v_1 *can be batched together, it is also possible to batch together all the subsequent operations in the two movable components, i.e.,* u_l *with* $v_{l'}$ *such that* $l = l', \forall l \in \{1, \ldots, n\}$.

Proof. It is possible to batch u_1 and v_1 together when $S_{u_1} \geq r_{v_1}$. When the previous condition is satisfied, Lemma 4.2 allows v_1 to be moved directly after u_1 on all resources $m \in M_{u_1} = M_{v_1}$ which results in the satisfaction of the invariant (4.14). If the two first operations $u_1 \in C$ and $v_1 \in C'$ are in the same batch, we have $S_{u_1} = S_{v_1}$, and consequently $r_{u_2} = r_{v_2}$. As $C \equiv C'$,we have $q_{u_2} = q_{v_2}$ inducing in turn $M_{u_2} = M_{v_2}$. Thanks to Lemma 4.1, either u_2 is sequenced before v_2 on all the resources $M_{u_2} = M_{v_2}$, or the opposite. Let us consider w.l.o.g. the case where u_2 is sequenced before v_2. As $r_{u_2} \leq S_{u_2}$ and $r_{u_2} = r_{v_2}$, it results that $r_{v_2} \leq S_{u_2}$.

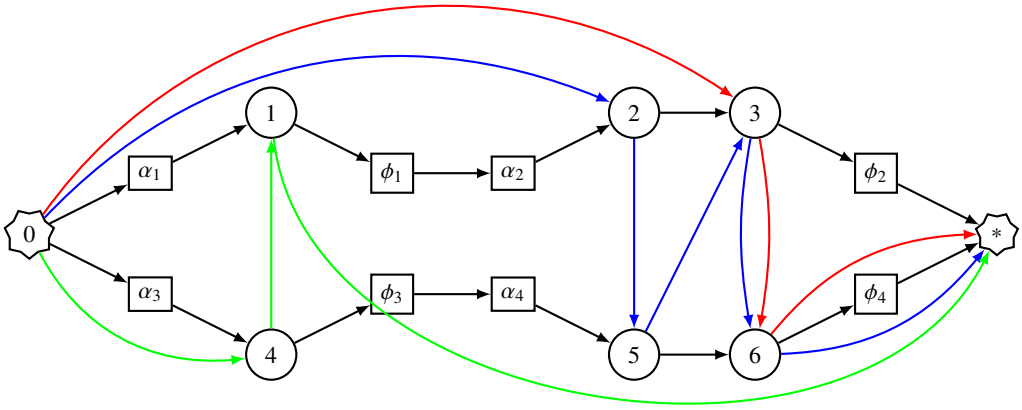


Figure 4.5 – *Example of batching long movable components*

Thanks to Lemma 4.2, it is possible to advance v_2 and make it a direct successor of v_2 on each resource $m \in M_{u_2} = M_{v_2}$ without creating any cycle. Making $v_{k'+1}$ a direct successor of v_{k+1} on each resource $m \in M_{v_{k+1}} = M_{v_{k'+1}}$ leads to $|m(v_{k+1})| = 1$. Given this last result, i.e. $\left|m^{-1}(v_2)\right| = 1$ and $S_{u_2} \leq r_{v_2}$, the invariant (4.14) is fulfilled, which allows the two operations u_2 and v_2 to be batched.

If u_l and v_l are batched, the same arguments can be used to prove that operations u_{l+1} and v_{l+1} can also batched and so forth for all the operations $u_k \in C$ and $v_k \in C'$ with $k > l+1$. □

Proposition 4.1 ensures that, if two first operations of equivalent movable components are batched together, it is possible to batch all the subsequent operations of the two movable components. Then, during the graph traversal, invariant (4.14) must be checked to decide if two equivalent operations u and v that are the first in their respective movable component should be batched. Each time such a decision is taken, the graph must be modified to ensure that all the subsequent operations of v are batched with the subsequent equivalent operations of u. In other words, this modification should lead to the satisfaction of invariant (4.14). When dealing with batching decisions on movable components with more that one node, because of the resource acquisition constraints, it may become necessary to modify the graph even if the static strategy is chosen. When considering the example in Figure 4.5, after moving operation 5 after operation 2, it is not necessary to modify the graph again as all the conditions are already met so that operations 3 and 6 are in the same batch. This modification is necessary for example when considering jobs requiring the long process in Figure 4.1(b).

In addition to adapting the edge weights, it may also be necessary to modify the graph when dealing with batching decisions on long movable components. When only the edge weights are modified, if the batching decisions that are taken within a solution are no longer valid within the new solution obtained after applying the chosen neighborhood operator, the

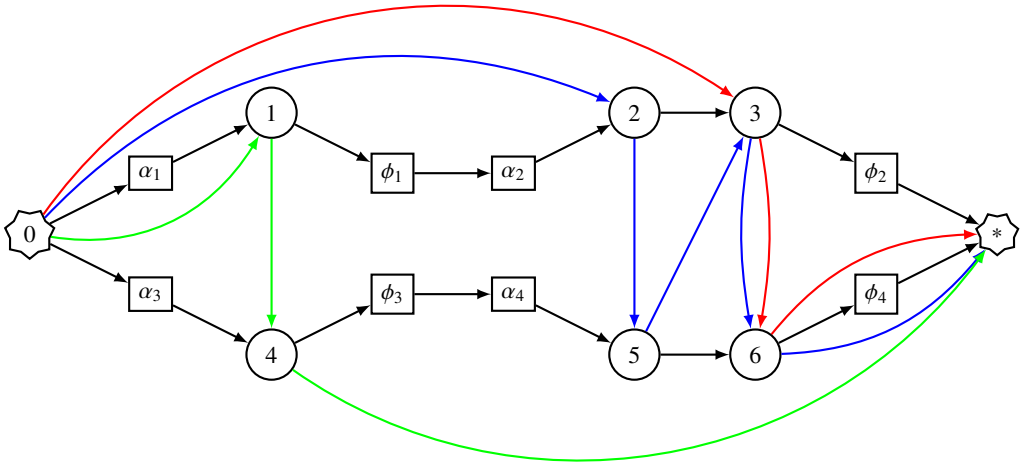


Figure 4.6 – *Example of an infeasible solution*

edge weights are adapted accordingly. Similarly, it should be ensured that the graph can be modified if the batching decisions that are already taken in the previous solution are no longer valid so that Property 4.2 is satisfied. For example, when operations 1 and 4 are resequenced on furnace F_1, the batching decisions taken within the solution shown in Figure 4.5 are not valid within the new solution shown in Figure 4.6. In the new solution, invariant (4.14) is no longer satisfied since $r_5 > S_2$. In this case, considering the resource acquisition constraints, operation 3 must be the direct successor of 2 on module M_1 and the same thing for operations 5 and 6. In this case, it is sufficient to sequence operation 3 before operation 5 on module M_1.

In the general case, Algorithm 4.2 can be used during the traversal of a graph G to undo invalid batching decisions that are inherited from a previous solution. This algorithm is used when the invalid batching decisions involve long movable components $C = (u_1, \ldots, u_l, \ldots, u_n)$ and $C' = (v_1, \ldots, v_l, \ldots, v_n)$ where some operations imposes resource acquisition constraints. When visiting v_1, batching C and C' is no longer valid when there is a resource acquisition $a(m, u_{l'}) \in A_{u_l}$ such that $(u_l, u_{l'}) \notin E_m^R$ while $r_{v_1} > S_{u_1}$. To obtain a feasible solution, all the resource acquisitions that are related to operations in C must be satisfied. For each resource acquisition constraint $a = (m, u_{l'}) \in A_{u_l}$, it must be ensured that $u_{l'}$ is sequenced before v_l, which means that $u_{l'}$ is sequenced directly after u_l on resource m. Also, if there are resources that are required by both u_l and $u_{l'}$ without being acquired by u_l, $u_{l'}$ must be also sequenced before v_l on all the concerned resources. If there are other components that belong to the same batch than C and C', this routine is applied to undo the batching decisions of any adjacent components. For example, if there is a third component $C'' = (w_1, \ldots, w_l, \ldots, w_n)$ batched with C and C', the routine in Algorithm 4.2 is applied when visiting the first node w_1 in C'' if $r_{w_1} > S_{v_1}$.

Algorithm 4.2 Algorithm to undo an invalid batching decision

unbatch $(C = (u_1, \ldots, u_l, \ldots, u_n), C' = (v_1, \ldots, v_l, \ldots, v_n), G)$
  assert $(r_{v_1} > S_{u_1})$
  for $l \in \{1, \ldots, n\}$
    for $a = (m, u_{l'}) \in A_{u_l}$
      for $m' \in M_{u_l} \cap M_{u_{l'}}$
        sequence $u_{l'}$ before v_l on m'

4.3.4 Integration of Minimum Batch Size Constraints

In the industrial setting, if a batch does not satisfy the minimal batching capacity of a machine, the processing of the concerned jobs is delayed until some jobs arrive and complete the batch. The same principle is adopted here. Working within a rolling horizon framework, it is sufficient to schedule the concerned batches outside of the scheduling horizon. Assuming that there are enough jobs to load all the machines during the whole scheduling horizon, scheduling the concerned operations at the end of their respective resource sequences means that they are scheduled outside of the scheduling horizon. If batches that do not satisfy the minimum batch size constraints are started within the scheduling horizon, the operations belonging to these batches and all the subsequent operations in the routes are excluded when computing the different criteria. This way of considering minimum batch size constraints can naturally be integrated to the batch-oblivious approach as it is designed to modify the graph during the start time computation and also thanks to the possibility, introduced in Section 3.3.1, of reconsidering already taken decisions.

Algorithm 4.3 Algorithm for resequencing a batch with a size lower than the minimum batch size

resequenceBatchSmallSize (B, G, V^s, V^u, V^f)
  for $v \in B$
    $u \leftarrow v_1 \mid v_1 \in C_v = (v_1, \ldots, v, \ldots, v_n)$
    $I \leftarrow \{w \in V^s \mid u \prec w\}$
    $V^s \leftarrow V^s \setminus I$; $V^u \leftarrow V^u \cup I$; $V^f \leftarrow V^f \setminus \left\{w \in O_j \mid u \in O_j\right\} \cup \{u\}$
    while $u \neq \phi_i \mid u \in G_j = (O_j, E_j, \alpha_j, \phi_j)$
      sequence u before $*$ on all resources M_u
      $u \leftarrow r(u)$
    return I

During the graph traversal, let us consider that the current visited operation v, which is sequenced after an incomplete batch B. Let q be the recipe of batch B and let us consider the situation where the size of this batch is below the given minimum batch size, i.e., $\sigma_B < b_q^{min}$. Using one of the strategies presented in Section 3.3.2, candidate operations can be looked

up in order to complete B. If no candidate is found ($E^0 \cup E^\infty = \emptyset$), settling v after B means that an infeasible decision is proposed within the considered solution. When implementing the proposed schedule on the shop floor, this can be ignored by executing the next batch in the sequence. It is also possible to post-process the solution so that the concerned batches are removed. By doing this, there is a significant risk of degrading the solution quality if this situation occurs several times within the proposed solution. To reduce the negative impact of this situation on the implemented solution, it is then proposed to sequence the incomplete batches at the end of their assigned resource sequences.

The modification of the graph during its traversal is described in Algorithm 4.3. Let us consider that during the traversal of the extended batch-oblivious conjunctive graph G, a batch B does not satisfy the minimum batch size constraint, i.e., $\sigma_B < b_q^{min}$. Recall that as discussed in Chapter 3, a cut $C = (V^s, V^u)$ is defined to ensure that the different modifications of the graph do not introduce cycles. The unidirectional cut $C = (V^s, V^u)$ is a partition of the graph G, where V^s is the set of settled nodes, and V^u is the set of unsettled nodes. Also, the set $V^f \subseteq V^u$ is defined as the set of all first job unsettled nodes, and its cardinality is at most the number of jobs $|J|$. When resequencing the incomplete batches at the end of their assigned resource sequences, these sets must be maintained. In Algorithm 4.3, for each node $v \in B$, if v is not the first node in its movable component C_v, the resequencing must start with the first node $u = v_1$. The set I contains all the settled nodes that are reachable from u. Resequencing u requires recomputing the start times of all the nodes in I which leads to their removal from the set of settled nodes V^s and their insertion in the set of unsettled nodes V^u. Operation u becomes the first job unsettled node. After maintaining these sets, the modification of the graph is performed within the loop. All the scheduled operations that are succeeding u in the route graph G_j are resequenced, one after another, before the artificial end node $*$. By proceeding in this way, it can be ensured that no cycle is introduced in the graph and that the acquisition constraints are satisfied. It is worth highlighting that it is not impossible for the moved operation to be part of a batch that satisfies the minimum batch size constraints. The selected strategy can discover them in order to complete another incomplete batch with the same recipe q. Finally, the integration of minimum batch size constraints is another reason that makes the use of the static strategy irrelevant.

4.4 General Solution Approach

Using the results of the previous sections, an algorithm that generalizes the one proposed in Section 3.4.3 can be designed to construct and improve schedules starting from an extended batch-oblivious conjunctive graph. For clarity, we focus on the construction of schedules that uses locally non-delay strategies. The overall schedule construction algorithm that considers all the constraints of the problem studied in this chapter is shown in Algorithm 4.4. Given an extended batch-oblivious conjunctive graph G and a selection strategy, this algorithm returns a schedule. Initially, only the artificial start node 0 is considered to be settled. The different sets, V^s, V^l and V^f defined in Section 3.3.1, are initialized. Then, nodes that meet the criteria of Theorem 3.1 can be successively settled without introducing any cycle. At each iteration,

a root node v is selected in V^u and one of its resource predecessors $u \in V^s$ is determined. The proposed approach to handle batching decisions on complex machines is to give freedom to the construction algorithm regarding the first nodes in the movable components. This also applies when it is necessary to undo batching decisions that are no longer valid within the current solution. Then, if u is the first node in its movable component, batching decisions that are inherited from previous solutions are first checked. If the movable components C_u and C_v of nodes u and v are batched together while invariant (4.14) is not satisfied because $r_v > s_u$, the graph must be modified to undo these batching decisions and to make sure that the resource acquisition constraints within the movable component C_u are satisfied. This is accomplished through the procedure detailed in Algorithm 4.2. As v may no longer be a root unsettled node, i.e., $deg_u^-(v) > 0$, the iteration is ended in order to select another node from V^u.

If there is no infeasible batching decision involving u and v and if it is not possible to extend the batch of u by adding v, the given strategy is used to search for potential candidates that can increase the batch. When the given strategy is a locally non-delay strategy, all the nodes in $w \in E^0$ are available before the start time of u, i.e., $S_u \geq r_w$. Lemma 4.2 ensures that no cycle is created when sequencing w after u on all the resources $m \in M_u$ and Proposition 4.1 ensure us that if u and v can be batched together, it is also possible to batch all the subsequent operations of the two movable components C_u and C_v. When the node to settle v and one of its resource predecessor u are determined, invariant (4.14) and the conditions regarding the batching capacity and compatibility are used to decide whether it is possible to batch nodes u and v. If it not possible to batch these two operations, it should be ensured that the size of the batch containing u is at least equal to the minimum batch size. If it is not the case, the procedure described in Algorithm 4.3 is used to sequence all the concerned operations at the end of their resource sequences. If the batch containing u satisfies its minimum batch size constraint and as v cannot be batched with u, the start time of v is computed using Algorithm 4.1 which considers the minimum time lags, the unavailability periods and the sequence-dependent setup times. At the end of the iteration, the sets V^s, V^l and V^f are updated.

The extended batch-oblivious approach is not bounded to one specific solution approach and can be applied within different heuristics. To evaluate the proposed approach, the same framework as in Chapter 3 is used. The GRASP approach creates many different starting solutions by randomizing the construction heuristic. To improve these initial solutions, the simulated annealing metaheuristic is used. Due to the various extensions, the neighborhood function ($\mathcal{N}^1$) described in Section 3.5 is no longer adapted. Either when constructing or improving an initial solution, the problem considered in this chapter imposes a different framework for the insertion and the removal of nodes from a partial or complete solution. A move works in two phases: First, it removes all nodes belonging to a currently scheduled movable component from the conjunctive graph. Second, it inserts all nodes that belong to a movable component into the conjunctive graph. The latter movable component could either be the same that was removed before or it could be a parallel (i.e., alternative) movable component. Removing nodes from a conjunctive graph is a straightforward procedure where no

Algorithm 4.4 A schedule construction algorithm for an extended batch-oblivious conjunctive graph G and a selection strategy

computeStartDatesAdaptively (G, Strategy)
$S_0 \leftarrow 0$
$V^s \leftarrow \{0\}$; $V^u \leftarrow V \setminus \{0\}$; $V^f \leftarrow \left\{v \in V^u \mid r^{-1}(v) = 0\right\}$
$B_v \leftarrow \{v\} \quad (\forall\, v \in V)$; $\sigma_{B_v} \leftarrow \sigma_j \quad (\forall\, v \in V \mid v \in O_j)$
while $V^s \neq V$
  $v \leftarrow$ select $(v \in V^u \mid deg_u^-(v) = 0)$
  $u \leftarrow w \mid w \in m^{-1}(v)$
  if $(u \in C_u = (u_1, \ldots, u_l, \ldots, u_n)$ **and** $u = u_1)$
    if $(r_v > s_u \wedge A_u \neq \emptyset \wedge \exists a = (m, w) \mid w \notin m(u))$
      unbatch(C_u, C_v)
      continue
    if $(\sigma_{B_u} < b_u^{max}$ **and** $(C_u \not\equiv C_v$ **or** $q_u \neq q_v$ **or** $r_v > S_u))$
      $(E^0, E^\infty) \leftarrow$ Strategy(u, V^f)
      if $(E^0 \neq \emptyset)$
        for $(w \in E^0)$
          $v \leftarrow w$
          for $(i \in \{1, \ldots, l, \ldots, n\})$
            sequence w_i after u_i on all resources M_{u_i}, $(w_i \in C_w, u_i \in C_u)$
  if $(\left|m^{-1}(v)\right| = 1$ **and** $S_u \geq r_v$ **and** $C_u \equiv C_v$ **and** $q_u = q_v$ **and** $\sigma_{B_u} + \sigma_v \leq b_v)$
    $S_v \leftarrow S_u$
    $B_v \leftarrow B_v \cup B_u$; $\sigma_{B_v} \leftarrow \sigma_{B_v} + \sigma_{B_u}$
  else
    if $(\sigma_{B_u} < b_u^{min}$ **and** $S_u < H)$
      $I \leftarrow$ **resequenceBatchSmallSize** (B_u, G, V^s, V^u, V^f)
      $B_w \leftarrow \{w\} \quad (\forall\, w \in I)$; $\sigma_{B_w} \leftarrow \sigma_j \quad (\forall\, w \in I \mid w \in O_j)$
      continue
    $S_v \leftarrow$ **computeStartTime** (v)
  $V^s \leftarrow V^s \cup \{v\}$; $V^u \leftarrow V^u \setminus \{v\}$; $V^f \leftarrow V^f \setminus \{v\} \cup \{r(v)\}$

decisions have to be taken. However, inserting nodes efficiently is challenging since a meaningful and feasible insertion position has to be found while coping with multiple resources per operation and resource acquisition constraints. The route graph formulation proposed by Knopp (2016) is closely related to the job-scheduling problem with processing alternatives studied in Kis (2003), where multiple resources per operation are also considered. Then, the insertion technique for nodes proposed by Kis (2003), which avoids enumerating dominated insertion positions, is adapted by Knopp (2016) by considering resource acquisition constraints. As the extended batch-oblivious graph proposed in this chapter is an aggregation of route graphs, the algorithm proposed in Knopp (2016) for determining feasible insertion positions is still applicable when solving the general problem considered in this thesis. Then, in short, the neighborhood operation, referred to as ($\mathcal{N}^2$), randomly select a movable component and randomly select one of the insertion positions determined by the adapted algorithm of Kis (2003). By applicable, it is meant that the insertion positions returned by the algorithm are feasible when applied to our extended batch-oblivious conjunctive graph. However, the algorithm of Kis (2003) avoids enumerating dominated insertion positions in the sense of the makespan. This may lead to discarding interesting moves or to choosing dominated moves when considering other criteria. An interesting research direction is to generalize these findings when considering other criteria than the makespan.

4.5 Numerical Results

The approach presented in this chapter is implemented in C++14 and compiled using the GCC MinGW-W64 compiler in version 5.3. All numerical experiments are conducted on an Intel Xeon E3-1240 3.5 GHz machine (4 cores) running Microsoft Windows 7. The parameters of the metaheuristic are the same than the ones used in Chapter 3. The objective of this section is to discuss the integration of some features of the industrial scheduling problem. Section 4.5.1 focuses on the modeling of complex batching machines. Most of the results and the discussion are presented in Tamssaouet et al. (2018b). In addition to the detailed modeling of the wet benches, an aggregated modeling where these machines are considered as black boxes is described. This new modeling is called here *data-driven analytical modeling*. As the data management module was initially designed to collect the necessary data for data-driven analytical modeling, a small industrial instance is used to perform the comparison between the two modelings. Besides the algorithm efficiency, the comparison is performed according to different practical considerations. Section 4.5.2 evaluates the the selection strategies described in Chapter 3 while considering the relevant criteria in the industrial context. The objective is to show that, when using the active strategy, solutions with a more relevant structure from the industrial perspective are obtained.

4.5.1 Modeling Complex Batching Machines: Wet Benches

In the industrial context, all the machines of the diffusion area are capable of batching. While it is realistic to associate fixed processing times with steps on the furnaces, cleaning steps on the wet benches have variable processing times depending on the loading sequences. Recall that, for clarity, a step corresponds to the entire processing of a job within a machine like a furnace or a wet bench. The term operation is used to describe the elementary processing of a job within an internal resource of a machine. In order to choose a suitable modeling approach for real applications, different practical aspects should be considered. The first and direct impact of a given modeling is the prediction accuracy, that can be defined as the difference between the planned and realized times. In the context of the diffusion area, several aspects make the prediction accuracy critical. Most of the jobs in wet-etch steps must proceed for further processing to furnaces without exceeding a maximum time lag between the two batching processes. As efficient schedules for wet-etch steps are essential to ensure high productivity at the furnaces (Ham and Fowler (2008)), an accurate prediction of completion times on wet bench machines helps to ensure that the satisfaction of time constraints. Also, the prediction accuracy is crucial when considering batching constraints. Poorly estimated job arrival times to furnace steps reduce the expected improvement from a scheduling algorithm over dispatching rules, e.g., Kohn and Rose (2013). The second aspect that should be considered when choosing a modeling technique is its impact on the efficiency of the scheduling algorithm. With only a few minutes to determine a solution, it is essential to make sure that the solution evaluation is not too time-consuming with the adopted modeling. Third, the resulting complexity of the scheduling system should be considered. In real applications, modeling a scheduling problem and developing an optimization algorithm to solve it is only part of the story. The impact of the modeling choice on the data management module must be taken into consideration.

Two models of wet benches are proposed in this thesis. The first uses the concept of route graph to model the machines in detail. Based on the route graph modeling, the generalized batch-oblivious conjunctive graph proposed in Section 4.3 allows the internal resources of these batching machines to be modeled. The second modeling is called *data-driven analytical modeling*. In this second approach, machines are considered as black boxes. To reflect their complex behavior, additional constraints are added to the model. To illustrate these two alternatives, the small industrial instance given in Section 2.2.1 is recalled. Table 4.1 provides four examples of recipes with the elementary processing times, given in minutes, on wet bench modules. The three first recipes (1, 2 and 3) are short processes while the last one is a long process. The first step of recipes 2 and 3 can be performed either on Module M_1 or on Module M_2. For Recipe 1, the first operation can only be performed by Module M_1. The considered problem is described in Table 4.2. Eight jobs have to be scheduled on a wet bench machine, four of which require the same short process (Recipe 1) and four others the same long process (Recipe 4). Column "Possible Sequences" provides the sequence of operations each step follows in the wet bench machine. Note that Module M_1 can only process the first operation of the short process. In order to optimize machine throughput, the optimization criterion is the makespan.

Table 4.1 – *Example of processes on a wet bench machine.*

Recipe	Type	M_1	M_2	D
1	Short	13	-	15
2	Short	38		15
3	Short	21		15
4	Long	16	47	15

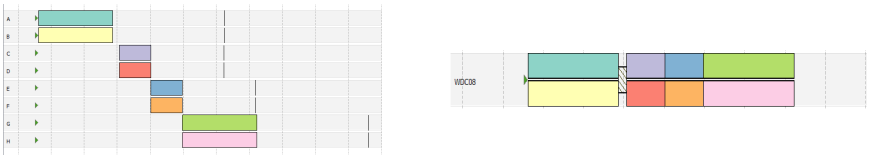
Table 4.2 – *Small size industrial problem instance.*

LotID	**RecipeID**	**Recipe Type**	**Possible Sequences**
A	4	Long Recipe	$M_1 \rightarrow M2 \rightarrow D$
B	4	Long Recipe	$M_1 \rightarrow M2 \rightarrow D$
C	1	Short Recipe	$M_1 \rightarrow D$
D	1	Short Recipe	$M_1 \rightarrow D$
E	1	Short Recipe	$M_1 \rightarrow D$
F	1	Short Recipe	$M_1 \rightarrow D$
G	4	Long Recipe	$M_1 \rightarrow M_2 \rightarrow D$
H	4	Long Recipe	$M_1 \rightarrow M_2 \rightarrow D$

The two modelings are used to solve this small instance. No preprocessing is required when using the generalized batch-oblivious graph modeling, which is not the case for the other alternative. With the data-driven analytical modeling, wet benches are modeled as black boxes. Step processing times are computed as the averages of historical cycle times. Due to the parallel processing of the batches, instead of directly using historical process cycle times as step processing times, a factor $\alpha < 1$ is estimated and used to compute the *actual* processing times. As it is possible to have three batches inside the processing part, the actual processing time is considered to be equal to $\alpha = \frac{1}{3}$ times the historical cycle time. The use of actual processing times is relevant from a machine perspective. From a job perspective, it should be ensured that jobs stay within wet bench steps at least during the historical cycle times. To do this, the historical cycle times are used as additional minimum time lags that are introduced to force jobs to stay at wet bench steps before moving to the next step on their routes. Finally, mainly due to blocking constraints, the inefficiencies that are generated by the succession of two steps are represented as sequence-dependent setup times. After classifying wet-etch processes, experts working in this area provided us with estimated inefficiencies that are generated by the succession of two process classes. However, this is not enough as the completion time of a step is not only influenced by its direct first predecessor, but also up to its fourth predecessor. To deal with this, we use simple decision rules that are given to operators on the shop floor. A set of *efficient* sequences with a length up to five steps are displayed for operators in front of the concerned machines. *Inefficient* sequences are also identified. To make the scheduling algorithm promote efficient sequences, setup times between successive steps are adjusted to make sure that the difference between the largest sum of setup times among efficient sequences and the smallest sum of setup times among

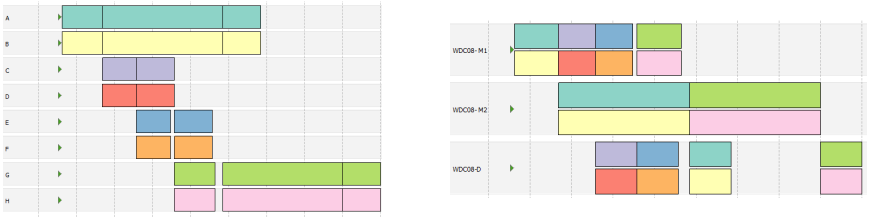
inefficient sequences is maximized. After formulating constraints on what is considered as acceptable changes, an integer linear program is used to obtain these adjustments.

Using the described data-driven analytical modeling, the optimal solution is represented through a Gantt chart from a job perspective in Figure 4.7(a) and from a resource perspective in Figure 4.7(b). Each color represents a distinct job; rectangles represent steps and parallel rectangles batches. In this solution, four batches are constructed, each containing two jobs. The length of the colored rectangles represents the actual processing time of each step. In Figure 4.7(a), vertical lines to the right side represent the constraint that forces each job to wait during the cycle time before going to the next step in its route. In Figure 4.7(b), the setup time is represented as a small rectangle with a downward diagonal pattern.



(a) *Job perspective Gantt Chart.* **(b)** *Machine perspective Gantt Chart.*

Figure 4.7 – *Scheduling* 8 *jobs on a wet bench machine, using data-driven analytical modeling.*



(a) *Job perspective Gantt Chart.* **(b)** *Machine perspective Gantt Chart.*

Figure 4.8 – *Scheduling* 8 *jobs on a wet bench machine, using route graph modeling.*

Using the route graph modeling, the optimal solution is shown on a Gantt chart from a job perspective in Figure 4.8(a) and from a resource perspective in Figure 4.8(b). Different aspects of complex behavior can be identified in these figures. First, note that this modeling can represent the possibility of parallel processing of multiple batches. Even if the batch containing jobs A and B is the first one to be loaded on the machine, it is ready to be unloaded only when the two next batches are unloaded. Finally, the consequence of the blocking constraint can be identified in Figure 4.8(b). Considering the step sequence on Module M_1, the last batch in the sequence, containing jobs G and H, does not start its processing directly after its predecessor even if it is available at the beginning of the scheduling horizon. This is because Module M_1 is blocked by the batch containing jobs E and F. This last batch is waiting for the batch containing jobs D and E to free the dryer so that it can free Module M_1.

The two considered models allow the same optimal loading sequence to be obtained in terms of machine throughput. However, let us compare the models according to other practical aspects. The obtained prediction accuracy using the data-driven analytical modeling

is not satisfactory. This can be explained by the use of rough estimates and adjustments, time averages and a loss of relevant information after too many data aggregations. Regarding the route graph modeling, the prediction accuracy is satisfactory as the machine is modeled in detail and the processing times are explicitly considered for each process type. Considering the fast-changing environment, the data-driven analytical modeling is also not satisfactory regarding the scheduling system complexity. This model should be periodically reviewed in order to update the process classification and time estimations, which could be reflected in the implementation of the database management subsystem. In the same way, efficient and inefficient sequences should be reviewed together with the estimated setup times. Also, as the internal modules can be unavailable without making the whole machine unavailable, data extraction has to be conditioned by the production status of these internal modules. The processing and setup times have to be adapted to the current configuration: Both modules are available, one of the modules is unavailable, or both modules are unavailable. This leads to a sophisticated scheduling system and more effort to maintain it.

Using the route graph modeling, this additional burden and complexity can be avoided. It requires only the processing times of the operations, that are already part of the recipe definition. As the internal resources are explicitly considered, their unavailabilities are naturally integrated. Finally, an important aspect that should be considered when choosing a modeling is its impact on the efficiency of the scheduling algorithm. Having only a few minutes to determine a solution, it is essential to make sure that the solution evaluation is not too time-consuming with the adopted modeling. Regarding this aspect, the driven analytical modeling is more advantageous as each step is represented by a single node in the conjunctive graph. Instead of a single node, a step is represented by a set of nodes when using the route graph modeling, one node for each operation. For example, when ignoring the artificial and separator nodes, the conjunctive graph of the instance above contains 10 nodes with the data-driven analytical modeling and 20 nodes with the route graph modeling. The solution evaluation, already the most consuming time phase in the solution approach, becomes more expensive. Then, the GRASP metaheuristic is slower and explores fewer solutions.

Though the route graph modeling has some advantages, the data-driven analytical modeling is adopted in the remaining of the thesis. This choice is constrained by the fact that the data management module was initially designed to collect the necessary data for the data-driven analytical modeling. Regarding the route graph modeling, different perspectives can be explored. An experimental study should be conducted to evaluate the impact on the metaheuristic efficiency. If the increase in the size of the generalized batch-oblivious graph negatively impacts the efficiency of the metaheuristic, the perspective of combining the two modelings can be interesting. The GRASP metaheuristic can be run with the data-driven analytical modeling, and each improving solution is recomputed using the route graph modeling. Comparing the solutions obtained with only the route graph modeling and those obtained combining the route graph modeling and the data-driven analytical modeling should help to decide which approach is the most relevant. Modeling complex machines where the handling part and the internal scheduling algorithm cannot be ignored is another challenge to address. The route graph modeling is only capable of capturing the complex behavior resulting from

the structure and the internal constraints of the machines. When this complex behavior is also due to the internal scheduling algorithm, using data-driven analytical modeling may be more interesting if it is not possible to incorporate the control logic to the detailed modeling.

4.5.2 Selection Strategies: Weighted Number of Moves

In Chapter 3, efficient selection strategies are proposed, and their performances are compared to those proposed by Knopp et al. (2017). Though industrial instances are used to conduct the experiments, a classical objective function is used: Total weighted completion time. This section proposes to conduct a comparative study between the different strategies using industrial instances encompassing all the constraints described in Chapter 2 and formalized in Section 4.1. The weighted number of moves (WNM) (4.3) is the chosen objective function. From an area management perspective, productivity is the most important objective, and the weighted number of moves expresses this objective. Besides, to study the solution quality in the criteria space, it is also proposed to study the structure of the solutions. This is done by using two objective functions as indicators: Discounted weighted number of moves (DWNM) (4.5) and batching coefficient (BC) (4.6). The discounted weighted number of moves is used to measure the tendency of a strategy to perform more operations and operations with higher job priorities at the beginning of the scheduling horizon. The batching coefficient is the average ratio of the actual size of each batch over its maximal size.

Five selection strategies are compared: Active, Integrated (Integ.), Resequencing (Reseq.), reassignment (Reass.) and Static. Within the active strategy, the geometric rule is used to compute the maximal delay. In addition to the active strategy, another strategy that is proposed in this thesis is the integrated strategy, which looks for operations to complete a batch in all the sequences of machines that are capable of processing the recipe of the incomplete batch. The resequencing and reassignment strategies are proposed in Knopp (2016). When using the static strategy, the start time computation algorithm only computes the start times of operations without modifying the solution. To perform the evaluation, two sets of instances are extracted from the Manufacturing Execution System of the company. The first set of instances, referred to as Large Industrial (LI), includes the data with the real status of the diffusion area at 15 different instant. The second set of instances, referred to as Very Large Industrial (VLI), includes the data with the status of the diffusion area in 10 different full days.

Given the chosen objective function, the relevant constraints that are considered in the two sets are the release dates, minimum time lags, sequence-dependent setup times, availability constraints and batching constraints. For each job, between one and five operations have to be performed, with three operations on average. These jobs must be scheduled on average on 68 machines, all capable of batching. The batching capacity lies between 2 and 7 jobs. The two sets of instances are different regarding the number of jobs. In the LI instances, the number of jobs varies from 350 to 550, while this number varies from 1500 to 1800 in the VLI instances. As the chosen indicators are all horizon dependent, a scheduling horizon of 8 hours is selected for the LI instances and 24 hours for the VLI instances. A computational

time of 300 seconds is allowed for both instance sets. Potentially confidential, the detailed results are not reported here. Table 4.3 provides results in terms of the relative deviation to the best values. We provide average and median values of these relative deviations for each instance set and selection strategy. In addition to the results relative to the solutions obtained with a computational time of 300 seconds, the results relative to the solutions obtained with a computational time of 60 seconds are reported for the LI instances.

	Strategy	***WNM***		***DWNM***		***BC***	
		Average	Median	Average	Median	Average	Median
LI (60s)	*A*ctive (GR)	**-0.6%**	**-0.5%**	**-1.9%**	**-1.7%**	**-0.4%**	**0.0%**
	*I*nteg.	-0.7%	-0.6%	-3.9%	-4.0%	-1.5%	-1.2%
	*R*eseq.	-1.7%	-1.6%	-10.0%	-10.5%	-4.4%	-4.4%
	*R*eass.	-1.6%	-1.5%	-3.1%	-2.6%	-2.2%	-2.0%
	*S*tatic.	-21.8%	-28.8%	-28.5%	-31.8%	-19.7%	-20.2%
LI (300s)	*A*ctive (GR)	**-0.1%**	**0.0%**	**-2.5%**	-2.5%	**-0.8%**	**-0.7%**
	*I*nteg.	-0.2%	**0.0%**	-2.7%	**-2.4%**	-1.2%	**-0.7%**
	*R*eseq.	-0.5%	-0.6%	-7.9%	-7.8%	-3.9%	-4.5%
	*R*eass.	-0.3%	-0.2%	-3.0%	-2.9%	-1.4%	-1.2%
	*S*tatic.	-2.4%	-1.8%	-15.0%	-14.6%	-8.0%	-7.3%
VLI (300s)	*A*ctive (GR)	-2.3%	-2.3%	-3.8%	-4.7%	**0.0%**	**0.0%**
	*I*nteg.	**-0.1%**	**0.0%**	**-0.3%**	**0.0%**	-2.2%	-2.3%
	*R*eseq.	-4.1%	-4.0%	-6.2%	-6.2%	-7.2%	-6.9%
	*R*eass.	-9.3%	-9.3%	-12.6%	-12.7%	-4.0%	-4.1%
	*S*tatic.	-45.3%	-50.4%	-54.2%	-58.6%	-28.5%	-31.0%

Table 4.3 – *Aggregate results for different strategies when optimizing the weighted number of moves*

The analysis of the results in Table 4.3 leads to several conclusions. Before going further, let us recall that Section 3.6 shows that, except for the static strategy, the integrated strategy is the most efficient strategy, meaning that, a metaheuristic using the integrated strategy explores more solutions than when using other strategies. When solving the instances used in this section, compared to the integrated strategy, the active and resequencing strategies explore on average 5% fewer solutions while the reassignment strategy explores 50% fewer solutions. Globally, the results show that the two proposed strategies perform better than those proposed in the original batch-oblivious approach regarding the chosen optimization criterion (WNM) and the criteria that are only used as indicators (DWNM). For example, regarding WNM and the LI instances, the solutions obtained using the active strategy with a computational time of 60 seconds are equivalent to those obtained using the resequenc-

ing strategy with a computational time of 300 seconds. For the VLI instances, the average and median of the relative deviations when using the integrated strategy are almost null while these two metrics have a value of −9.3% when using the reassignment strategy. An important observation that can also be made when looking to the results is the significant improvement brought by the adaptive start time computation compared to the static strategy, especially for the VLI instances. For example, the median of relatives deviations over the VLI instances is equal to 54% when using the static strategy while this same metric takes a null value when the integrated strategy is used.

In addition to the comparison between all strategies, the experiments are conducted in particular to compare the active strategy and the integrated strategy. The active strategy is designed to allow a batch to be completed by delaying operations when no other operation can be added without delaying the start time of the concerned batch. In the same situation, the integrated strategy will let the batch incomplete even if there are operations that can be added with a small delay in the already computed start time of the batch. When there are permanently enough jobs in the queue in the horizon, it may be unnecessary to delay a batch that is almost complete in order to include an additional operation that can be part of a large size batch that is going to be processed soon. This situation corresponds to the VLI instances. When the VLI instances are extracted, all the jobs that go through the diffusion area during a horizon of 24 hours are included, and this horizon corresponds to the scheduling horizon when solving these instances. These instances are used in Chapter 5 to compare the solution proposed by our approach to the actual schedules implemented in the studied diffusion area. On the contrary, the LI instances are those the scheduling approach must solve in real conditions. When optimizing these instances over a given horizon such as 8 hours, it is unrealistic to include all the jobs that are going to be available during the horizon with accurate release dates. Thus, in case of the LI instances, except in the first hours of the horizon, there is a low level of work in process.

In the light of what has been mentioned and the results in Table 4.3, it may be possible to conjecture that the active strategy quickly obtains better solutions in a context of a low level of work in process in comparison with the integrated strategy. Indeed, when looking at the results of these strategies when solving the VLI instances, the integrated strategy performs better. However, regarding the LI instances, even if it explores fewer solutions, the active strategy performs better than the integrated one. Regarding the WNM considered as the objective function, the difference between the two strategies is too small to be significant. Regarding the DWNM and the BC, the active and the integrated strategies can be considered equivalent with a computational time of 300 seconds. With a computational time of 60 seconds, the active strategy is better than all other strategies regarding the DWNM and the BC. For example, the average and the median of the DWNM obtained by the integrated strategy is at least lower by 1.3% than the values obtained with the active strategy. Also, the average and the median of the BC is at least lower by 1.1% than the values obtained with the active strategy. While it can be expected from the active strategy to perform well on the BC, it may be surprising to obtain the best values on the DWNM, as the principle of the active strategy is to delay the start times of incomplete batches to include additional operations. It

can be concluded that even though it explores fewer solutions, the active strategy quickly reaches good quality solutions. Moreover, considering the industrial concerns and the rolling horizon framework, the found solutions have a more relevant structure.

4.6 Conclusion

In this chapter, we considered the industrial scheduling problem described in Chapter 2. The different constraints and criteria are formalized. In addition to the criteria defined in previous works, two new criteria are proposed: Discounted weighted number of moves to model throughput within the rolling horizon framework and the target satisfaction indicator to integrate production targets. The batch-oblivious graph is generalized to handle all the industrial criteria defined in Chapter 2. The invariant that must be satisfied when making batching decisions is reformulated to take into consideration minimum time lags and multiple resources per operation. As the availability constraints are not explicitly expressed in the graph, an algorithm is proposed to adjust operation start times when multiple resources are required. From a structure perspective, the batch-oblivious graph presented in Chapter 3 is generalized by modeling job routes through the route graph that is introduced by Knopp (2016) instead of a linear sequence of operations. The route graph is initially proposed to allow complex machines to be modeled in detail. In this chapter, this modeling is generalized to tackle complex machines that are capable of batching. In the original batch-oblivious approach, batching decisions are encoded through edge weights. When dealing with complex batching machines, in addition to the edge weights that must be modified, the graph must also be modified during its traversal to make sure that the internal constraints of the machines are satisfied. Necessary conditions that allow the graph to be modified while guaranteeing the feasibility of the solution are presented. This chapter ends with numerical results where the integration of some features of the industrial scheduling problem is discussed. It is shown that the generalized batch-oblivious graph is capable of handling the studied industrial problem. The practical advantages of modeling complex batching machines in detail are highlighted. The numerical experiments also show that the use of the active strategy allows the solution approach to obtain solutions with more relevant structures from an industrial perspective. So far, only one optimization criterion is used at the same time in the numerical experiments. Chapter 5 is dedicated to the different approaches that are used to handle the multiobjective aspect of the scheduling problem at hand.

Chapter 5

Multiobjective Optimization Approach

The preceding chapters of this thesis focused on the improvement and the extension of the batch-oblivious approach so that all the constraints found in the industrial context are efficiently handled. Chapter 4 defines and formalizes different optimization criteria that suit the industrial application. However, a single criterion is optimized so far in the numerical experiments. This chapter is devoted to the development of different approaches to handle the multiobjective aspect of the scheduling problem at hand. Section 5.1 gives notations and some known definitions in multiobjective optimization, such as the dominance relations and reference points that are used in this work. The chosen modeling of the decision maker preferences is discussed and formally defined in Section 5.1.3. After this introductory section, two types of approaches are suggested. In Section 5.2, the simulated annealing approach is adapted to solve the multiobjective problem by using given preferences, and the weighted augmented Tchebychev metric to aggregate the criteria. Two ways of computing the acceptance probability of a new solution within simulated annealing are proposed which result in two versions of the simulated annealing approach. Section 5.3 motivates and describes approaches where a set of nondominated solutions is maintained during the search. The two versions of the simulated annealing approach in Section 5.2 are modified to allow the archiving of nondominated solutions during the search process. In addition to these two approaches that require the preferences of the decision maker, the Archived Multiobjective Simulated Annealing (*AMOSA*) of Bandyopadhyay et al. (2008) that does not require the preferences of the decision maker is the third investigated approach. Using industrial instances, these three approaches are compared in Section 5.5.1 and in Section 5.5.2 by considering the given preferences and some quality indicators defined in Section 5.3.3. This chapter is concluded with a comparison between the results of our approach and the actual results of the diffusion area.

5.1 Definitions and Notations

In the preceding chapters, different criteria to optimize are motivated and defined. However, only one criterion is optimized so far. The remainder of this chapter is devoted to the different approaches that are proposed to handle the multiobjective aspect of the scheduling problem at hand. In the following, the problem is first formally described. Then, some well-known

definitions in the literature of multicriteria optimization, see e.g., T'kindt and Billaut (2006), are recalled. Section 5.1.1 describes different dominance relations that introduce a general definition of optimal solutions in the context of multiobjective optimization. Section 5.1.2 introduces the definitions of some reference points that are used in this thesis. Finally, different ways of modeling of the preferences of the decision maker are discussed in Section 5.1.3. In this last section, the chosen modeling is also formally defined.

5.1.1 Dominance Relations

The objective is to simultaneously optimize n given criteria $f = (f_1, \ldots, f_i, \ldots, f_n)$. Let X be the set of feasible solutions and $Y \subset \mathbb{R}^n$ the image in the criteria space of X by f. Note that we assume, without loss of generality, that the given objectives are to be minimized. For a single criterion optimization, i.e., $n = 1$, the structure associated with $\mathbb{R}$ is a total order, i.e., there is no incomparability between two solutions, which makes the definition of an optimal solution straightforward. Due to the conflicting nature of the considered criteria, it is rare to find a solution that concurrently optimizes all the criteria. A more general definition of optimality is obtained by defining a *dominance relation*.

Different dominance relations are described in the literature. The most used one is the *Pareto dominance*, formally defined in Definition 5.1. This relation introduces a new definition of optimal solutions. In Definition 5.2, a point in the criteria space is said *nondominated* if it is not possible to improve one criterion without degrading others. In Definition 5.3, a solution $x \in X$ is called an *efficient solution* if its projection in the criteria space results in a nondominated point. Pareto dominance introduces a class of Pareto optima. Let us denote by X_E the set of efficient solutions and Y_N the projection of X_E in the criteria space. Y_N defines the trade-off curve in the criteria space, also called the *Pareto front*.

Definition 5.1 (**Pareto dominance**). *Let y and $y' \in Y$ be such that $y = f(x), y' = f(x')$ and $x, x' \in X$. It is said that y* dominates *y' and noted $y \leq y'$ if and only if $y_i \leq y'_i, \forall i = 1, \ldots, n$ and $\exists i \in \{1, \ldots, n\}$ such that $y_i < y'_i$. By extension, a solution $x \in X$ dominates a solution $x' \in X$, noted $x \leq x'$, if and only if $f(x) \leq f(x')$.*

Definition 5.2 (**Nondominated point**). *An objective vector $y \in Y$ is nondominated if and only if $\nexists y' \in Y$ such that $y' \leq y$*

Definition 5.3 (**Pareto optimum**). *A solution $x \in X$ is a* Pareto optimum*, also called an* efficient solution *, if and only if $\nexists x' \in X$ such that $f(x') \leq f(x)$, i.e. $f(x)$ is nondominated*

In addition to the Pareto dominance, several relations are defined in the literature. Definitions 5.4 and 5.5 illustrate two additional relations that are classically used in the literature. Each of these relations introduces a new definition of an optimal solution in case of multiobjective optimization. Other relations like the $\epsilon-$dominance, cone $\epsilon-$dominance, and angle dominance can be found in the literature (see Liu et al. (2019)).

Definition 5.4 (**Weak dominance**). *Let $y, y' \in Y$. It is said that y* weakly dominates *y' and noted $y \leqq y'$ if and only if $y_i \leq y'_i, \forall i = 1, \ldots, n$*

Definition 5.5 (**Strict dominance**). *It is said that* y strictly dominates y' *and noted* $y < y'$ *if and only if* $y_i < y'_i, \forall i = 1, \ldots, n$

5.1.2 Reference Points

Generally speaking, a *reference point* is any vector y^{ref} which is considered as an objective to reach. The objective is to find the closest possible solution to this point, in the sense of a function to be optimized. If the set of efficient solutions X_E is known, it is possible to define three reference points: *Ideal point*, the *utopian point* and the *nadir point*. These points are respectively defined in Definitions 5.6, 5.7 and 5.8. The Nadir and ideal point produce important information about a multiobjective optimization problem. For a decision maker, they show the possible range of the objective values of all the criteria over the Pareto set: They are respectively exact upper and lower bounds for the set of efficient points. They may also be important within optimization algorithms (Ehrgott and Tenfelde-Podehl (2003)). As the set of efficient solutions is unknown in our case, y^{id}, y^{ut} and y^{na} denote in the remainder of the chapter approximations of the ideal point, the utopian point, and the nadir point, respectively. The way these reference points are approximated is discussed in Section 5.2.

Definition 5.6 (**Ideal point**). *A point* $y^{id} = (y_1^{id}, \ldots, y_n^{id}) \in \mathbb{R}^n$ *is called* ideal *if and only if, for each* $i \in \{1, \ldots, n\}$, $y_i^{id} = \min_{y \in Y} \ y_i$ *holds.*

Definition 5.7 (**Utopian point**). *A point* $y^{ut} = (y_1^{ut}, \ldots, y_n^{ut}) \in \mathbb{R}^n$ *is called an* utopian point *if and only if it dominates the ideal point* y^{id}*, i.e.* $y^{ut} \leq y^{id}$ *. This point does not correspond to any feasible solution.*

Definition 5.8 (**Nadir Point**). *A point* $y^{na} = (r_1^{na}, \ldots, r_n^{na})$ *is called* nadir *(or* anti-ideal point*) if and only if* $y_i^{na} = \max_{y \in Y_N} \ y_i, \quad \forall \ i \in \{1, \ldots, n\}$.

5.1.3 Preferences of the Decision Maker

Instead of a unique optimal solution, the dominance relations defined in Section 5.1.1 lead to a set of incomparable solutions. Hence, at some stage of the problem-solving process, the decision maker has to explicit his preferences about the objectives in order to choose a single solution. Evans (1984) presents three occasions where the decision maker can provide his preferences: Before, during or after the search process. Methods that require the decision maker to intervene before the resolution process are called *a priori* methods. Examples of such methods are the two approaches proposed in Section 5.2. When the decision maker is allowed to provide his preferences during the resolution process, a method is called *interactive*. Methods in the last category are called *a posteriori*, as they allow the decision maker to formulate his preferences after the resolution process. An example of such methods is the approach discussed in Section 5.3.4.

While preferences are the basis of tie-breaking between solutions in the efficient set, their incorporation is difficult because of uncertainties arising from a lack of prior problem knowledge and fuzziness of human preferences. Different ways of modeling preferences can be

encountered in the literature. When the trade-off between criteria is acceptable, the different criteria may be aggregated into a single criterion where different *weights* are associated with the criteria. The weighted sum or the weighted Tchebychev metric are examples of such modeling. When no trade-off between the criteria is allowed, the order in which the criteria are considered is related to their importance, the most important criterion being in the first place. In this case, the *lexicographic order* is used. The decision maker can also present goals to reach for each criterion. Therefore, instead of optimizing the criteria, the objective is to find a solution which satisfies the goals, even if this solution does not correspond to an efficient solution (T'kindt and Billaut (2006)). Other ways of modeling the preferences can be found in the literature (see e.g., Marler and Arora (2004) and Rachmawati and Srinivasan (2006)).

To make the problem as general as possible, we consider in this thesis that the decision maker is offered two ways of expressing his preferences. These preferences are used during the search process of the approaches detailed in Section 5.2 whereas they may only be used to select a final solution among the set of nondominated solutions in the approaches of Section 5.3. The first way that can suit the preferences of the decision maker is the use of a lexicographic order. As mentioned before, this modeling fits the situation where no trade-off between the criteria is admitted. In the industrial context, this concerns the criteria that model constraint satisfaction, i.e., maximum time lags and production targets. These constraints are considered as soft constraints, not because the decision maker considers their violation as acceptable, but because the context can make their satisfaction impossible. Regarding these particular criteria, it may be unlikely for the decision maker to accept a trade-off with pure performance criteria such as the weighted number of moves or X-factor. When the trade-off is possible, the decision maker is given the possibility of prioritizing some criteria over others through weights.

Formally, a relation $\preceq \subseteq \mathbb{R}^n \times \mathbb{R}^n$ is called a lexicographic order if $A \preceq B \Leftrightarrow A \prec B \vee A = B$ with $(a_1, \ldots, a_n) \prec (b_1, \ldots, b_n) \Leftrightarrow \exists\, m \leq n : \forall\, i < m : a_i = b_i \wedge a_m < b_m$. In the context of multicriteria optimization, we consider that each criterion f_i is associated with a *lexicographic rank* $l_i \leq n$. It is assumed that these ranks are contiguous and the smaller is the rank, the more important is the criterion. Let assume without loss of generality that the indices of the criteria are given in the non-decreasing order of lexicographic rank, i.e., $\forall i < n, l_i \leq l_{i+1}$. Since such relations meet the properties of antisymmetry, transitivity and totality, lexicographic orders are total orders. This allows pairwise comparisons of objective function values. As the lexicographic ranks are required to be contiguous, having $l_n = n$ implies that each criterion is given a distinct rank, i.e., $l_i = i$. In this case, if $y^1 \preceq y^2$ where $y^1, y^2 \in Y$, then $y^1 \prec y^2 \vee y^1 = y^2$ with $(y^1_1, \ldots, y^1_n) \prec (y^2_1, \ldots, y^2_n) \Leftrightarrow \exists\, m \leq n : \forall\, i < m : y^1_i = y^2_i \wedge y^1_m < y^2_m$.

When $l_n < n$, this means that at least two criteria share the same rank, i.e., $\exists i < n \mid l_i = l_{i+1}$. In this case, it is understood that the trade-off between the concerned criteria is acceptable. It is not enough to use a lexicographic order to compare two solutions. In this situation, a second way of taking into account the preferences of the decision maker is to use weights. We consider that we are given weights $c \in \mathbb{R}^n_{>0}$. Each criterion f_i is associated with weight $c_i \in \mathbb{R}_{>0}$ that translates the priority of the decision maker.

It should be noted that weights only allow discriminating between criteria that share the same lexicographic rank. Weights are used to aggregate the concerned criteria in a single function. In this thesis, this aggregation is done as a weighted augmented Tchebychev metric within the local search heuristics. As the use of the weighted augmented Tchebychev metric requires the computation of some reference points, which requires, in turn, to have initial solutions, the weighted sum is used as an aggregation function within the construction heuristic. This modeling is general as it encompasses three situations. The first one is when all the criteria have different ranks. In this case, weights are meaningless. The second situation is when all the criteria have the same rank, and the priorities are only expressed through weights. The last situation is a hybrid one where there are at least two lexicographic ranks and at least two criteria that share the same rank and that are differentiated through weights.

.

5.2 A Priori Approaches

Within our GRASP approach, the simulated annealing is used to improve initial solutions. Since local search algorithms, such as simulated annealing, have been initially designed for single-objective optimization, they are single-trajectory algorithms, meaning that they follow a single solution. A common approach for multiobjective optimization is to reduce the multidimensional objective problem to a single-dimensional one by weighting all objectives in one objective function and to use a standard acceptance probability for the resulting evaluation function. This section develops such an approach by using the given preferences of the decision maker. This proposed approach is a generalization of the approaches given in Bitar (2015) and Knopp (2016). In Bitar (2015), it is assumed that preferences are expressed through weights that are associated with the criteria. As simulated annealing is used in the proposed memetic algorithm, the weighted augmented Tchebychev metric is used to aggregate the different criteria, which is necessary to compute the acceptance probability and compare between nondominated solutions. In Knopp (2016), it is supposed that preferences are expressed through a lexicographic order. To calculate the acceptance probability, the metric proposed in Bitar (2015) is used to aggregate all criteria.

The simulated annealing approach described in this section keeps most of the components described in Section 4.4. Our Simulated Annealing metaheuristic is based on the same neighborhood function and also starts with the solution found by the construction heuristic. In each iteration, the neighborhood operation $\mathcal{N}^2$ described in Section 4.4 randomly selects one movable component be moved, its feasible insertion positions are computed, and one of them is randomly selected and performed. We use a geometric cooling schedule that maintains a temperature of T which is multiplied by a cooling factor $P_c < 1$ after each iteration. The original metaheuristic must be adapted when it comes to studying whether to accept or not the newly generated solution. At iteration k, let x be the current solution and x' be the new generated one and let $y = f(x)$ and $y' = f(x')$ be their projection in the criterion space, respectively. In the single objective case, as objective values are scalars, the move is imme-

diately accepted if the new value of the objective function y' improves the current objective function value y. Otherwise, the new solution is accepted with a probability of $exp(\frac{-\Delta}{T})$, where $\Delta = y' - y$. If the new solution is not accepted, all changes related to the move are reversed.

With n criteria, the objective values y and y' are vectors in $Y \subset R^n$. With no preference information, dominance relations as the Pareto dominance described in Section 5.1.1 can be used to compare the two solutions. However as these relations describe partial orders, the solutions may be incomparable. Also, the most important feature of simulated annealing that consists in accepting non-improving solutions requires the computation of an acceptance probability based on a scalar that depends on the objective function values. In this section, this problem is solved by considering the given preferences of the decision maker, through lexicographic ranks and weights, during the search process. Two ways of using these preferences during the computation of the acceptance probability are investigated, leading to two versions of the simulated annealing approach referred to as *SA-I* and *SA-II*.

When aggregation is needed, the weighted augmented Tchebychev metric is used. This metric is described in Section 5.2.1 while the approximation of the reference points that are required is detailed in Section 5.2.2. The main objective of this section is to go through the different ways of considering the given preferences during the search process. To compute the acceptance probability, it may seem more natural to aggregate criteria when only weights are provided. The main difference between the two versions of the simulated annealing approach lies in the aggregation or not of criteria with different lexicographic ranks. Section 5.2.3 provides different ways to compare between the current solution x and the new solution x'.

5.2.1 Weighted Augmented Tchebychev Metric

The weighted Tchebychev metric is a known aggregation function in the context of multicriteria optimization (see T'kindt and Billaut (2006)). It determines the weighted distance to a reference point $y^{ref} \in \mathbb{R}^n$ using weights $\lambda \in \mathbb{R}^n$ and the maximum norm defined as $L_\infty : \mathbb{R}^n \to \mathbb{R},\ (y_1, \ldots, y_n) \mapsto \max_{i=1}^{n} |y_i|$. Note that weights $\lambda \in \mathbb{R}^n$ are different from the weights $c \in \mathbb{R}^n_{>0}$ given by the decision maker. The determination of $\lambda \in \mathbb{R}^n$ and $y^{ref} \in \mathbb{R}^n$ is described in Section 5.2.2. The advantage of this aggregation function, when the parameters are well chosen, is that it allows any efficient solution to be reached. This property is not guaranteed when the weighted sum is used, since efficient solutions (called *non-supported solutions*) might not be attainable with any set of coefficients. Instead of the weighted Tchebychev metric, we use the *weighted augmented Tchebychev metric* that is obtained by augmenting the first metric by a term. This term is added to break ties between solutions that are equal regarding the objective that dominates the maximum norm but is different regarding other objectives.

In Section 5.2.3, different ways of considering preferences are exposed. Regarding the use of the weighted augmented Tchebychev metric, it may be only needed to aggregate criteria with the same lexicographic rank or all criteria of the optimization problem. When only criteria with the same rank are aggregated, the weighted augmented Tchebychev metric of

criteria with rank l is denoted by $\overline{a}_{\lambda, y^{ref}, \rho, l}$ and is given in (5.1).

$$\begin{aligned} \overline{a}_{\lambda, y^{ref}, \rho, l} &: \mathbb{R}^n \to \mathbb{R}, \\ \overline{a}_{\lambda, y^{ref}, \rho, l}(y_1, \ldots, y_n) &\mapsto max^n_{i=1, l_i=l} \lambda_i \left| y_i - y_i^{ref} \right| + \rho \sum_{i=1, l_i=l}^{n} \lambda_i \left| y_i - y_i^{ref} \right|. \end{aligned} \tag{5.1}$$

As mentioned above, the term $\rho \sum^n_{i=1, l_i=l} \lambda_i \left| y_i - y_i^{ref} \right|$ is added in order to break ties between solutions that are equal regarding the objective that dominates the maximum norm but different regarding other objectives. The L_∞ norm should always be the dominating term to guarantee that the weighted sum is only relevant for breaking ties between solutions that are equally rated by the L_∞ norm. Thus, the parameter ρ must be chosen sufficiently small. With

$$\sum^n_{i=1, l_i=l} \lambda_i \left| y_i - y_i^{ref} \right| \leq n \cdot \max^n_{i=1, l_i=l} \lambda_i \left| y_i - y_i^{ref} \right| < 2 \cdot n \cdot \max^n_{i=1, l_i=l} \lambda_i \left| y_i - y_i^{ref} \right|$$

(for each $y \in Y$ except the ideal point for which $\overline{a}_{\lambda, y^{ref}, \rho, l}(y^{id})$ might be equal to zero), fixing the parameter to $\rho = \frac{1}{2 \cdot n}$ does always satisfy the desired property, i.e. the L_∞ norm is always the dominating term in the weighted augmented Tchebychev metric. After estimating the parameter ρ, other parameters of the weighted augmented Tchebychev metric are discussed in the next section. The weighted augmented Tchebychev metric of all criteria is denoted by $\overline{a}_{\lambda, y^{ref}, \rho}$ and is given in (5.2).

$$\begin{aligned} \overline{a}_{\lambda, y^{ref}, \rho} &: \mathbb{R}^n \to \mathbb{R}, \\ \overline{a}_{\lambda, y^{ref}, \rho}(y_1, \ldots, y_n) &\mapsto max^n_{i=1} \lambda_i \left| y_i - y_i^{ref} \right| + \rho \sum_{i=1}^{n} \lambda_i \left| y_i - y_i^{ref} \right|. \end{aligned} \tag{5.2}$$

When there are n criteria, the objective values y and y' are vectors in $Y \subset R^n$. In this first approach, the comparison between y and y' is performed using the given preferences of the decision maker. It is assumed that these preferences are given in the form of a lexicographic rank l_i and a weight c_i for each criterion f_i. As it is assumed without loss of generality that the indices of the criteria are given in the non-decreasing order of lexicographic rank, l_n denotes both the lexicographic rank of the criteria f_n and the total number of lexicographic ranks. Let $g : \mathbb{R}^n \to \mathbb{R}^{l_n}$, $y \mapsto \overline{y}$ be the function, given in (5.3), that aggregates the criteria with the same lexicographic rank using the weighted augmented Tchebychev metric. After this aggregation, there are l_n values to consider lexicographically.

$$\begin{aligned} g &: \mathbb{R}^n \to \mathbb{R}^{l_n}, \\ g(y_1, \ldots, y_n) &\mapsto (\overline{a}_{\lambda, y^{ref}, \rho, 1}, \ldots, \overline{a}_{\lambda, y^{ref}, \rho, l_n}). \end{aligned} \tag{5.3}$$

5.2.2 Approximation of Parameters of the Metric

An important parameter in the considered aggregation function is the reference point $y^{ref} \in \mathbb{R}^n$ from which the distance of a solution is computed. To determine this reference point, all the resource constraints in the conjunctive graph are relaxed. So, the conjunctive graph consists only of edges related to the routes of the job. For each job, the shortest path is chosen. In other words, each operation of a job is given as a processing time the time it requires when assigned to the fastest machine. Note that the release dates of all jobs are still modeled in this graph. Then, the computation of the earliest start dates in this graph yields a schedule without waiting periods. The computation of the objective function values for such schedules leads to lower bounds of the considered regular objective functions to minimize and upper bounds for objective functions to maximize. The construction of this graph also leads to lower bounds for the objectives that were derived from maximum time lag constraints. By relaxing the resource, the notion of batch disappears. As it is not trivial to compute the context-dependent upper bound for the batching coefficient criterion, it was considered equal to 1. Thus, the obtained objective function values can either represent an ideal point or a utopian point, used as the reference point within the weighted augmented Tchebychev metric.

After describing how the reference point and the parameter ρ are determined, let us deal with the weights $\lambda \in \mathbb{R}^n$ of the aggregation function. In addition to discriminating between the criteria, these weights are essential to normalize the values to be compared. The considered objective functions do not use the same unit of measurement and have a different magnitude. For example, the total maximum time lag violation severity and weighted flow factors cannot be directly compared. The computation of these weights uses the estimated nadir point y^{na} (see Definition 5.8). During preprocessing, a set of sample solutions $S \subset X$ is chosen by performing random moves on a starting solution. Note that the same set is also used to calibrate simulated annealing (it is $|S| = P_s$, for the parameter P_s). Then, the set $Q \subseteq S$ is defined to determine the sampled solutions that are best with respect to at least one considered objective function. This is given by

$$Q = \bigcup_{i=1}^{n} \operatorname*{argmin}_{x \in S} \; f_i(x).$$

An estimation of the nadir point is then determined by the point $y^{na} \in \mathbb{R}^n$ which is defined by

$$y_i^{na} = \max_{s \in Q} \; f_i(s) \quad \forall \; i \in \{1, \ldots, n\}.$$

When having the reference and the nadir points, it becomes possible to normalize the values of all criteria. The weights $c \in \mathbb{R}^n_{>0}$ provided by the decision maker are used when determining $\lambda \in \mathbb{R}^n$ so that he can have the possibility to prioritize some criteria over others. Based on this, the weights $\lambda \in \mathbb{R}^n$, used in the weighted augmented Tchebychev metric, are determined as

$$\lambda_i = \frac{c_i}{y_i^{na} - y_i^{ref}} \qquad (\forall \; i \in \{1, \ldots, n\}),$$

where $y^{na} \in \mathbb{R}^n$ is the estimation of the nadir point and $y^{ref} \in \mathbb{R}^n$ is the reference point. For consistency, weights c_i are ignored when all criteria are aggregated and when there are more that one lexicographic rank.

5.2.3 Acceptance Conditions of a New Solution

As an a priori approach, the simulated annealing approach developed in this section assumes that preferences of the decision maker are known in advance and are applied during the search process. When preferences are available, the question is how to use them during the search. A first alternative consists of strictly adhering to the preferences. For example, when a lexicographic order is given, if a new solution significantly improves a less important criterion with the cost of a small deterioration of a more important criterion, the new solution will not be kept. In this case, it is assumed that sticking to the preferences will always lead to final solutions that are satisfactory, which may be not the case. Instead of the first alternative, it is also plausible to loosen the way the preferences are considered. This second alternative may have the benefit of getting solutions with satisfying values for the less important criteria, and may also lead the search to a region where the most important criteria have better values. In this section, we define two ways of considering the preferences given as lexicographic ranks and weights. Numerical results are given in Section 5.5.

The pseudo-code of the simulated annealing approach is given in Algorithm 5.1. It starts with the solution x returned by the construction heuristic, the initial temperature T and the cooling factor α. As the simulated annealing approach is embedded in the GRASP framework, x^* denotes the global best solution found so far. "Stop" denotes the stopping criteria. In the industrial application, the maximum number of non-improving iterations is taken as the stopping criterion. At each iteration of the loop, x denotes the current solution and x' the new solution generated by applying the neighborhood operator $\mathcal{N}$. One node of the graph representing x is randomly chosen to be moved, its feasible insertion positions are computed, and one of them is randomly selected and performed. y, y' and y^* are the projection in the criterion space of x, x' and x^*, respectively. At each iteration of the simulation annealing, the decision of keeping or discarding the new solution x' must be taken. When a single criterion is optimized, the solution is automatically accepted if it improves the current solution x. In our case, a solution is judged as improving by using the lexicographic ranks and the weights. First, the values of y and y' that correspond to criteria having identical ranks are aggregated using the function g described in (5.3) to obtain $\overline{y}$ and $\overline{y}'$. As a total ordering, a lexicographic order allows pairwise comparisons of objective function values. Therefore, if $\overline{y}' \preceq \overline{y}$, the new solution x' is automatically accepted. Also, if the new solution x' improves the global best solution found so far within GRASP, i.e, $\overline{y}' \prec \overline{y}^*$, x' becomes the new best solution.

If x is lexicographically better than x', the simulated annealing enables x' to be accepted with a probability that depends on the exploration stage. In the single objective case, as objective values are scalars, the new solution is accepted with a probability of $exp(\frac{-\Delta}{T})$, where $\Delta = y' - y$. If the new solution is not accepted, all changes related to the move are reversed. To compute this probability in case of multiple criteria, two alternatives are investigated. A

first approach consists of strictly sticking to the preferences of the decision maker by only basing the calculation of the acceptance probability on the lowest rank m in the lexicographic order where $\overline{y}'_m > \overline{y}_m$. Recall that the lower the lexicographic rank, the more important a criterion. Within this approach, Δ_1 is computed using the Tchebychev metric given in (5.1), i.e., $\Delta_1 = \overline{a}_{\lambda, y^{ref}, \rho, l}(y') - \overline{a}_{\lambda, y^{ref}, \rho, l}(y)$. The preferences are respected by disregarding higher ranks in the computation of the probability and by using the weights c_i in the determination of parameters λ_i. The use of this approach is what defines the first version of the simulated annealing approach denoted by SA-I.

In the other approach, preferences are neglected when computing the acceptance probability of x' by aggregating all the values of the criteria using the weighted augmented Tchebychev metric in (5.2). As the lexicographic order is ignored, it does not seem consistent to use the weights c_i in the determination of parameters λ_i. With this second approach, Δ_2 is computed by aggregating all the criteria. The use of this approach is what defines the second version of the simulated annealing approach denoted by SA-II. Then, depending on the chosen approach, Δ takes the value of Δ_1 or Δ_2 and the acceptance probability is computed subsequently.

Algorithm 5.1 *A priori* Simulated Annealing (Prio-SA)

Prio-SA $(x, x^*, T, \alpha, \text{Stop})$
  while Stop = $False$
    $x' \leftarrow \mathcal{N}(x)$
    $y \leftarrow f(x)\,,\ \ y' \leftarrow f(x')\,,\ \ y^* \leftarrow f(x^*)$
    $\overline{y} \leftarrow g(y')\,,\ \ \overline{y}' \leftarrow g(y')\,,\ \ \overline{y}^* \leftarrow g(y^*)$
    if $(\overline{y}' \preceq \overline{y})$
      $x \leftarrow x'$
      if $(\overline{y}' \prec \overline{y}^*)$
        $x^* \leftarrow x'$
    if $(\overline{y} \prec \overline{y}')$
      $\Delta_1 = \overline{y'_m} - \overline{y_m} \mid \overline{y'_m} > \overline{y_m} \wedge \overline{y'_l} = \overline{y_l} \forall l < m$ **(SA-I)**
      $\Delta_2 = \overline{a}_{\lambda, y^{ref}, \rho}(y') - \overline{a}_{\lambda, y^{ref}, \rho}(y)$ **(SA-II)**
      $\Delta = \Delta_1 \veebar \Delta = \Delta_2$
      if $(exp(\frac{-\Delta}{T}) \leq random(0, 1))$
        $x \leftarrow x'$
    $T \leftarrow \alpha * T$

5.3 Archive-Based Approaches

5.3.1 Introduction

The central concept within the approaches discussed in this section is the concept of *Archive*. Instead of keeping a unique best solution during the search process, while using the given preferences, this approach stores in the archive the set of nondominated solutions visited so far. This approach is adopted in order to save and update all well spread nondominated solutions generated by the algorithm during the search. After each iteration, the archive is updated with the new non-dominated solution. This set of non-dominated solutions represents an approximation of the Pareto front. Below are given the different practical considerations that motivate and show the interest of producing an approximation of the Pareto front.

The first consideration that can motivate the choice of this approach is that actual values of the different optimization criteria can be of a little relevance in practice. In the studied industrial context, it is difficult for a human to decide whether a solution is good or not, based only for example on the value of the time lag violation severity or the X-factor. This observation can be made even for criteria that are a direct translation of performance indicators, for example the weighted number of moves. As the scheduler does not have visibility of all the lots that will enter the area during the scheduling horizon, the predicted number of moves is always underestimated. Comparing these values to historical performances of the area does not lead to a fair evaluation of solutions. Even if a decision maker has an idea of what are the best performances for a period, it is challenging to produce at each run of the scheduler aspiration goals that take into consideration the actual context and the expectation of the decision maker. Rather than focusing on exact values, decision-makers are more oriented towards goals of minimum/maximum achievements levels for a set of preferences. In addition to the irrelevance of actual values, because of the imprecision of input data and the gap between the predicted and the actual performances of the proposed solution, focusing on improving a single criterion by a few percentage points makes little, if any, sense (Framinan et al. (2014a)).

The choice of keeping a set of non-dominated solutions makes it possible to handle these practical considerations, provided that the set is a good approximation of the Pareto front. For each solution, it becomes manageable to decide whether a solution is good or not on a criterion when using the range of the criterion on all the solutions belonging to the approximated Pareto front. Also, as decision-makers are reluctant to extreme values, the final solution can be chosen among the set of nondominated solutions after excluding extreme solutions. Having a set of nondominated solutions may also prevent the risk of focusing on improving by a low percentage a criterion with a high priority while significantly degrading a less important priority. Instead of blindly applying the given preferences, attention can be given to such situations.

Several other considerations support the use of an approximation of the Pareto front. While taking into consideration a set of criteria and a set of constraints during the optimization process, it may be interesting to include additional criteria and constraints in the choice

of the final solution. As a particular case of this possibility, different values of some problem parameters can be used to reevaluate the solutions. For instance, instead of having one TSI criteria with a unique value of parameter α, multiple TSI can be added by changing the value of this parameter. This can lead to more robust solutions that are less sensitive to the imprecision of the given preferences. With a set of nondominated solutions, it also becomes possible to study the question of whether all objectives are necessary and whether some of the objectives may be omitted. The number of considered objectives strongly influences both the performance of the optimization algorithms and the decision making process in general. First, with more objectives, more incomparable solutions can arise, the number of which affects the generating method's performance. Second, the larger the number of objectives, the more complex the choice of an appropriate solution for a decision maker. The question of criteria reduction is closely linked to the fundamental issue of conflicting and non-conflicting optimization criteria and can be answered by analyzing the Pareto set approximation (Brockhoff and Zitzler (2007)).

Since local search algorithms, such as simulated annealing that is used in the proposed GRASP approach, have been originally designed for single-objective optimization, they are single-trajectory algorithms, meaning that they follow a single solution. Various extensions of these algorithms can be found in the literature. Different works propose a simple extension of the initial algorithms, in the sense that a single *current* solution is considered and moved through the search space while keeping a set of nondominated solutions. Multiobjective simulated annealing (MOSA) of Ulungu et al. (1999) and archived multiobjective simulated annealing (AMOSA) of Bandyopadhyay et al. (2008) are examples of such extension. Instead of focusing on single solutions, other works propose extensions that are based more on improving the full archive iteratively. The Pareto local search (PLS) of Angel et al. (2004) and the indicator-based multiobjective local search (IBMOLS) of Basseur et al. (2012) are examples of such extensions (Blot et al. (2018)).

As specified above, to adequately handle the different practical considerations, it is essential that the obtained set of nondominated solutions is a good approximation of the efficient set. Before using classical indicators that can be used to evaluate the quality of the approximated Pareto front, it should first be shown that the final solution obtained after applying the given preferences on the approximated Pareto front is not worse than the final solution obtained by one of the a priori approaches SA-I and SA-II. One of the specific questions we want to answer is to decide whether it is more advisable to use a given preference model during the search (a priori selection) or, on the contrary, to generate an approximation of the Pareto optimal frontier and then to apply the same preference model once on that approximation. Eppe et al. (2011) investigated how preference models can be compared to each other and to gain further insight into how they interact with multiobjective metaheuristics. This study concludes that a priori preference articulation should be preferred when addressing many-objective combinatorial problems, i.e., problems that have more than three conflicting objectives (Ishibuchi et al. (2009)). As the problem studied in this thesis considered many objective functions, it should be ensured that the performance of the a priori approach is not deteriorated.

In this thesis, three archive-based approaches are investigated. The two first approaches are based on SA-I and SA-II that are proposed in the previous section. In addition to keeping the best solution regarding the given preferences, a passive archive stores the nondominated solutions discovered during the search. The archive is qualified passive as there is no interaction between its content and the search process. The third approach is the Archived Multiobjective Simulated Annealing (*AMOSA*) of Bandyopadhyay et al. (2008). The choice of this approach is motivated by the fact that it is only an extension of the simulated annealing that is already used as an improvement heuristic approach within our GRASP framework and that the reported experimental results show that it is well performing when solving many-objective problems. The main component of this approach is recalled in Section 5.3.4. Before this, Section 5.3.2 discusses the different questions that arise around the concept of the archive. Section 5.3.3 defines the used quality indicators that are known in the literature to assess an approximated set of a Pareto front.

5.3.2 Archive

The archive stores potential Pareto optimal solutions, i.e., solutions not yet dominated by any other solutions found so far. This is the set of solutions finally returned by the optimization algorithm. Generally, the update of the archive with the new solutions found during the search and which are potentially nondominated generally lies on Pareto dominance. In the literature, other dominance relations are used (Liefooghe (2009)). Let A_k denotes the archive at the end of iteration k. This set must satisfy two conditions:

1. Any two points of A_k must be non-dominated with respect to each other. More formally, $\forall y, y' \in A_k, y \nleq y' \wedge y' \nleq y$.
2. Any found point not in A_k is dominated by at least one point in A_k. In other words, $\forall y_i \notin A_k$ where $i \leq k, \exists y \in A_k$ such that $y \leq y_i$.

An important question regarding the size arises when an archive is used to store nondominated solutions found so far. Depending on the answer to this question, different archiving strategies can be distinguished: Unconstrained archive, constrained archive, and fixed archive size. The unconstrained archiving is discussed in Fieldsend et al. (2003). An unconstrained archive can be used to store all the nondominated solutions found during the search process. However, some problems may have an exponential number of nondominated solutions (infinite for some continuous problems), and it becomes impossible to store all of them. Also, preserving all elite points is costly in time (due to the linear comparison with all archived solutions needed before a new point can be inserted into the archive). Thus, different strategies can be implemented to reduce the number of stored solutions. The constrained archiving is discussed in Knowles and Corne (2004). When the archive size exceeds an a priori hard bound, different techniques can be used to reduce the size of the archive. For example, in Bandyopadhyay et al. (2008), by using clustering, the member within each cluster whose average distance to the other members is minimum is considered as the representative member

of the cluster. A last archiving strategy is based on a constant storage capacity. This strategy is similar to the bounded archiving when there are too many nondominated solutions but differs when the archive is not full, in which case dominated solutions are also added to the archive (Liefooghe (2009)).

Before considering the possibility of having large size archive, it was preferred in this thesis to adopt an unconstrained archive strategy. This choice allows a better understanding of the problem. Also, in case it becomes necessary to use a constrained archive, a better estimation of the size can be obtained through the analysis of the results of an unconstrained archive strategy. Different reasons can be found in Fieldsend et al. (2003) in favor of the adopted strategy. Two negative consequences of using a constrained strategy are highlighted. The main consequence is the small extent of the estimated front, i.e., loss of diversity. Second, extra search time is required in order to rediscover the extremes of the estimated Pareto front. Also, for the problem studied in Bandyopadhyay et al. (2008), it is highlighted that the clustering algorithm is the most time-consuming procedure in AMOSA. As mentioned before, the main drawback of an unconstrained strategy is the computational and memory cost of maintaining the archive when its size increases. To minimize the computational cost, Fieldsend et al. (2003) proposes data structures to facilitate the use of an unconstrained archive, without the need for a linear comparison with the nondominated set for every new point inserted. In this thesis, a less sophisticated way is adopted in order to improve the efficiency of maintaining the archive.

5.3.3 Quality Indicators

A good set of nondominated solutions should satisfy two goals: Convergence and diversity. Convergence refers to finding a set of solutions that lies on or is close to the actual Pareto-optimal front. Diversity refers to finding a set of solutions which are diverse enough to represent the entire range of the Pareto-optimal front. The assessment of these sets is far from being a trivial issue. It is stated in Zitzler et al. (2003) that unary quality indicators, i.e., which assign a single value to each non-dominated point set, are inherently limited in their inferential power. Different indicators must be combined for better quality evaluation of a set of nondominated solutions. In this thesis, the quality indicators used in García-León et al. (2019) are adopted. Two measures are used to evaluate the convergence: Hypervolume (HV) and Mean Ideal Distance. The two other measures evaluate the diversity of the final archive: Maximum Spread (D) and Spacing (SP).

As its name suggests, the hypervolume consists of the measure of the region which is simultaneously dominated by the points in archive A and bounded above by a reference point $y^{hv} \in R^n$. To compute this indicator in the numerical experiments, we used the implementation proposed by Fonseca et al. (2006). The mean ideal distance is the second indicator assessing the convergence of an archive by measuring the closeness between this archive and an ideal point. When there are only criteria to minimize, it is common to use the origin point as an ideal point. As some criteria are to be maximized in our context, the origin point cannot be an ideal point. To use this indicator, we used in the experiments a unique point y^{mid} for

all the instances that dominate the utopian points described in Section 5.1.2. This indicator is formalized in (5.4).

$$MID = \frac{\sum_{x \in A} \sqrt{\sum_{i=1}^{n} (f_i(x) - y_i^{mid})^2}}{|A|} \tag{5.4}$$

$$D = \sqrt{\sum_{i=1}^{n} f_i^{max} - f_i^{min}} \tag{5.5}$$

The two remaining indicators are used to assess the diversity of the produced set of nondominated solutions. The maximum spread (D) is used for measuring the diagonal length of a hyperbox that is formed by extreme function values observed in the Pareto curve (Zitzler (1999)). This indicator is computed using (5.5), where $f_i^{max} = max_{x \in A}(f_i(x))$ and $f_i^{min} = min_{x \in A}(f_i(x))$ are respectively the maximum and minimum values of criterion f_i for all the solutions in A. The last metric is spacing (SP) that was introduced in Schott (1995) for measuring the average distance between consecutive solutions in A. The metric is formalized in (5.6), where $\hat{d}_j = min_{x_k \in A, k \neq j} \sum_{i=1}^{n} \left| f_i(x_j) - f_i(x_k) \right|$ is the distance between a nondominated solution x_j and its nearest solution, and $\overline{d} = \frac{1}{|A|} \sum_{j=1}^{|A|} \hat{d}_j$ is the average of these distances for all solutions in A.

$$SP = \sqrt{\frac{1}{|A|} \sum_{j=1}^{|A|} (\hat{d}_j - \overline{d})^2} \tag{5.6}$$

These quality measures are proposed in Section 5.5.2 to assess the final archives produced by the three investigated approaches. An archive is a better approximation of the Pareto front if the values of the hypervolume and the maximum spread are maximum, and if the values of the mean ideal distance and spacing are minimum.

5.3.4 Archived Multiobjective Simulated Annealing

For completeness and notational consistency, this section summarizes the main ideas of AMOSA that was proposed by Bandyopadhyay et al. (2008). The different propositions are reformulated to consider a minimization problem. Contrary to the original approach, instead of a constrained archive, the size of the archive in our implementation is unconstrained. Also, instead of storing the nondominated solutions that are produced by a single AMOSA, the archive is shared by all AMOSAs running in parallel within the GRASP framework. Section 5.3.4.1 defines the concept of *amount of dominance* that measures the amount by which a solution dominates another one. Section 5.3.4.2 goes through the different cases depending on the dominance status between the new solution and the current solution and between the new solution and the solutions in the current archive.

5.3.4.1 Amount of Dominance

The amount of dominance is an important concept that distinguishes AMOSA from other extensions of the simulated annealing to handle multiobjective problem optimization. Instead of formulating the acceptance criterion between the current solution and the new solution in terms of the difference in the number of solutions that they dominate, AMOSA takes into consideration the amount by which this dominance takes place. Recall that n is the number of considered objectives. Let x and $x' \in X$ be two solutions, and y and y' be their projection on the criterion space, respectively. The amount of dominance is defined in (5.7), where R_i denotes the range of the ith objective. As this range is not usually known *a priori*, the solutions in the archive along with the new and current solutions are used for computing it.

$$\Delta dom_{y,y'} = \prod_{i=1, y_i \neq y'_i}^{n} \frac{\left|y_i - y'_i\right|}{R_i} \tag{5.7}$$

5.3.4.2 AMOSA process

As mentioned above, the archived multiobjective simulated annealing (AMOSA) of Bandyopadhyay et al. (2008) is an example of the extension of the initially proposed simulated annealing approach where a single solution is considered and moved through the search space. The pseudo-code of this adapted simulated annealing approach is given in Algorithm 5.2. AMOSA starts with the solution x returned by the construction heuristic, the initial temperature T and the cooling factor α. As this heuristic is embedded in the GRASP framework, A denotes the set of nondominated solutions found so far. "Stop" denotes the stopping criteria. As in the *a priori* simulated annealing described in Section 5.2, the maximum number of non-improving iterations is taken as the stopping condition.

Let x be the current solution at each iteration, y the projection of x in the criterion space, x' the new obtained solution after applying the neighborhood operation $\mathcal{N}^2$ and y' its projection in the criterion space. The decision of accepting x' is based on the dominance status of y' with respect to y and solutions in the archive A. In addition to the amount of dominance defined in Section 5.3.4.1, two additional measures are necessary to define this dominance status. The *dominance rank* r is defined as the number of points in A that dominate point y'. This measure is first used in the multiobjective genetic algorithm proposed by Fonseca and Fleming (1993). The third measure is the *dominance count* c that is defined as the number of point in A that are dominated by point y' (Liefooghe (2009)). As the dominance status between the new point y' and the current point y are explicitly analyzed and as y may or may not be part of the archive, it is assumed that the computation of these two measures exclude the current point y when it belongs to the archive. Also, using the transitivity property of the dominance relation, it can be shown that if $r > 0$, then $c = 0$.

Based on the dominance status between y and y', three different cases are distinguished. For each case, different sub-cases are identified based on the values of r and c. These different situations are recalled below:

Algorithm 5.2 Archive Multiobjective Simulated Annealing (AMOSA)

AMOSA $(x, A, T, \alpha, \text{Stop})$
  while Stop = $False$
    $x' \leftarrow \mathcal{N}(x)$
    $y \leftarrow f(x)\,,\ \ y' \leftarrow f(x')$
    $r \leftarrow |\{y^a \in A \setminus \{y\} \mid y^a \leq y'\}|$
    $c \leftarrow |\{y^a \in A \setminus \{y\} \mid y' \leq y^a\}|$

    if $(y \leq y'$ **and** $r \geq 0)$ (**Case 1**)
      $\Delta dom_{avg} = \frac{\sum_{y^a \in A \setminus \{y\} | y^a \leq y'} \Delta dom_{y^a,y'} + \Delta dom_{y,y'}}{r+1}$
      if $(exp(\frac{-\Delta dom_{avg}}{T}) \leq random(0,1))$
        $x \leftarrow x'$

    if $(y \not\leq y'$ **and** $y' \not\leq y)$ (**Case 2**)
      if $(r \geq 1)$ (**Case 2** .1)
        $\Delta dom_{avg} = \frac{\sum_{y^a \in A \setminus \{y\} | y^a \leq y'} \Delta dom_{y^a,y'}}{r}$
        if $(exp(\frac{-\Delta dom_{avg}}{T}) \leq random(0,1))$
          $x \leftarrow x'$
      if $(r = 0$ **and** $c = 0)$ (**Case 2** .2)
        $x \leftarrow x'$
        $A \leftarrow A \cup \{x'\}$
      if $(c \leq 1)$ (**Case 2** .3)
        $x \leftarrow x'$
        $A = (A \setminus \{x \in A \mid x' \leq x\}) \cup \{x'\}$

    if $(y' \leq y)$ (**Case 3**)
      if $(r \geq 1)$ (**Case 3** .1)
          $\Delta dom_{min} \leftarrow \min_{y^a \in A} \Delta dom_{y^a,y'}$
          if $(\exp^{-\Delta dom_{min}}) \leq random(0,1))$
            $x \leftarrow x^{min}$
      if $(r = 0$ **and** $c = 0)$ (**Case 3** .2)
          $x \leftarrow x'$
          $A \leftarrow (A \setminus \{x\}) \cup \{x'\}$
      if $(c \leq 1)$ (**Case 3** .3)
          $x \leftarrow x'$
          $A \leftarrow (A \setminus \{x \in A \mid x' \leq x\}) \cup \{x'\}$

- **Case 1:** $y \leq y'$ and $r \geq 0$. In this case, the current solution y dominates the new obtained solution y' and r points from the archive A dominate y'. Even being a non-improving solution, there is a non-zero probability of accepting the new solution y'. The probability of acceptance is equal to $exp(\frac{-\Delta dom_{avg}}{T})$ where T denotes the current temperature. Δdom_{avg}, computed as shown in (5.8), denotes the average amount of dominance of the new point y' by $(r + 1)$ points, namely, the current solution y and r points of the archive A.

$$\Delta dom_{avg} = \frac{\sum_{y^a \in A \setminus \{y\} | y^a \leq y'} \Delta dom_{y^a,y'} + \Delta dom_{y,y'}}{r + 1} \tag{5.8}$$

- **Case 2:** $y \not\leq y'$ and $y' \not\leq y$. In this case, the new point y' and the current point y are nondominating with respect to each other. Based on the values of r and c, three situations may arise:

  1. $r \geq 1$: the new point y' is dominated by at least one point in the archive A. Note that here the current solution y may or may not be on the front of the archive. In this situation, the probability of acceptance of y' is $exp(\frac{-\Delta dom_{avg}}{T})$ where Δdom_{avg} is defined in (5.9).

$$\Delta dom_{avg} = \frac{\sum_{y^a \in A \setminus \{y\} | y^a \leq y'} \Delta dom_{y^a,y'}}{r} \tag{5.9}$$

  2. $r = 0$ and $c = 0$: y' is nondominating with respect to the other points in the archive as well. In this case, y' is on the same front as the archive. Therefore, x' is selected as the current solution and added to the archive, i.e., $x = x'$ and $A = A \cup \{x'\}$.
  3. $c \leq 1$: the new point y' dominates c points of the archive. Again, x' is selected as the current solution, i.e., $x = x'$. Regarding the archive, x' is added and all the dominated solutions are removed, i.e., $A = (A \setminus \{x \in A \mid x' \leq x\}) \cup \{x'\}$.

- **Case 3:** $y' \leq y$. In this case, the new point y' dominates the current point y. As in the previous case, three situations may arise based on the values of r and c:

  1. $r \geq 1$: the new point y' is dominated by at least one point in the archive A. Contrary to the same situation in Case 2, it is certain that y is not in the archive. According to the simulated annealing paradigm, y' is accepted with probability 1 as it improves the current solution y. However, due to the presence of the archive in AMOSA, some solutions are still better than y'. Therefore, it is proposed to make y and the closest point to y' in the archive compete for acceptance. The closest point in the archive to y', denoted y^{min}, is the point with the minimum amount of dominance, denoted Δdom_{min}. The computation of this amount of dominance can be found in (5.10). The solution x^{min} from the archive which corresponds to the minimum difference is selected as the current point with probability $\exp^{-\Delta dom_{min}}$). Otherwise, x is selected as the current point, i.e., $x = x$.

$$\Delta dom_{min} = \min_{y^a \in A} \Delta dom_{y^a,y'} \tag{5.10}$$

2. $r = 0$ and $c = 0$: y' is nondominating with respect to the other points in the archive except the current point y if it belongs to the archive. Recall that the computation of r and c does not consider the current point y if it belongs to the archive. Thus, the new point y' is accepted, i.e., $x = x'$. The new solution is added to the archive and the current solution is removed if it belongs to the archive, i.e., $A = (A \setminus \{x\}) \cup \{x'\}$.
3. $c \leq 1$: The new point y' dominates c points of the archive. Note that c does not count the current point y that may or may not be in the archive. Again, x' is selected as the current solution, i.e., $x = x'$. Regarding the archive, x' is added and all the dominated solutions are removed, i.e., $A = (A \setminus \{x \in A \mid x' \leq x\}) \cup \{x'\}$.

As shown through the different situations, the decision of accepting or not a new solution depends on the current solution and on the archived solutions. Contrary to AMOSA, SA-I and SA-II are a priori approaches that require the preferences of the decision maker. Another difference is that the archive within these two a priori approaches does not influence the acceptance decision of a new solution. In Section 5.5.2, we compare AMOSA to the archive version of the two a priori approaches (SA-I and SA-II) using the quality indicators in Section 5.3.3. As stated before, one of the objectives in the numerical experiment is to study whether an a posteriori approach such as AMOSA can compete with an a priori approach when taking into account given preferences. In Section 5.5.1, in addition to comparing SA-I and SA-II, the given preferences are used to select a final solution from the AMOSA archive so that the three approaches are compared.

5.4 Multiobjective GRASP Approach

As explained in Chapter 3, our heuristic creates many different starting solutions by randomizing a construction algorithm. Each solution is then independently improved using a local search method. We use a construction method which greedily inserts operations. To improve constructed solutions, we use a Simulated Annealing metaheuristic. When considering multiple criteria, some changes have to be made on this framework. In this section, we describe the Multiobjective GRASP approach and summarize the different building blocks discussed in this chapter.

As in Chapter 3, the construction heuristic sorts the jobs in decreasing order of their ratio $\frac{w_j}{d_j}$ (weight divided by due date). When dues dates are not part of the problem definition, as it is the case with the industrial instances, only weights are used to sort the jobs. Otherwise, jobs are initially sorted in decreasing order of the sum of the shortest processing durations of their operations. As the construction heuristic is used within a GRASP approach, the construction is randomized by perturbing the sorted list of jobs. A tuning parameter $P_i \geq 1$ is

used to steer the perturbation intensity. At each iteration of the construction heuristic, the next job to be inserted is determined by randomly selecting one of the first P_i elements in the sorted list of remaining jobs. The heuristic then iterates over the sorted list of jobs and successively inserts all operations of the current job. The best insertion position for each operation is the insertion position that leads to the best values of the criteria for of the partial solution. To find the best insertion position, the operation is inserted in all feasible positions, and the obtained partial solutions are compared. If there are two or more criteria with the same lexicographic rank, the weighted sum is used to aggregate their value. It is not possible, at least for the first constructed solution, to use the weighted augmented Tchebychev metric as the determination of one of its parameters already requires initial solutions. The construction is completed when all operations of all jobs have been inserted. The first initial solution is automatically stored in the empty archive. The next constructed and improved solutions are added if the solutions already in the archive do not dominate them.

Each constructed solution is then independently improved using one of the three simulated annealing based heuristics: SA-I, SA-II, and AMOSA. In each step of these metaheuristics, a new solution is obtained by applying to the current solution the neighborhood function described in Section 4.4. The differences on how a new solution is accepted in the three approaches is studied. The acceptance conditions used by the a priori approaches SA-I and SA-II are given in Section 5.2.3 and those used within AMOSA can be found in Section 5.3.4.2. In AMOSA, accepting the new solution is a necessary condition for adding it to the archive. However, in SA-I and SA-II, as there is no interaction between the search process and the passive archive, the new solution, whether accepted or not, is compared with the archived ones at any iteration. For all the metaheuristics, we use the same geometric cooling schedule that maintains a temperature T which is multiplied by a cooling factor $P_c < 1$ after each iteration. The initial temperature is determined by sampling a fixed number P_s of random moves. When optimizing a single criterion, we compute the difference Δ between the criterion value of the new solution x' obtained after performing a selected move and the initial solution x, i.e., $\Delta = f(x') - f(x)$. In case of multiple criteria, the weighted augmented Tchebychev metric in (5.2) is used to compute these values, i.e, $\Delta = \overline{a}_{\lambda, y^{ref}, \rho}(f(x')) - \overline{a}_{\lambda, y^{ref}, \rho}(f(x))$. Then, for a tuning parameter P_p, the P_p-th percentile of these values is selected as the initial value for the temperature T. The search within SA-I and SA-II is stopped if the best solution does not improve during a specified number of iterations P_m. In AMOSA, the search is stopped when it does not succeed to update the archive during the same specified number of iteration P_m.

The GRASP based approach is parallelized as follows. Each solution is constructed and improved independently and thus can be run in its thread. In the numerical experiments, the same improvement heuristic is used within all threads. Thus, for example, SA-I refers in the next section to the whole GRASP approach that uses SA-I to improve the initial solution. Communication between threads is only needed to update the shared information. When AMOSA is used, the archive is shared between the different threads. When SA-I and SA-II are used, the best overall solution must be updated every time one of the threads finds a better solution. When a passive archive is built during the search process of the a priori approaches, the new solution at any iteration of any thread must be compared to the solutions

already in the archive. The control of the selection and the removal of solutions in the shared archive is performed using Pareto dominance. A fixed number of threads is used, and each thread restarts with a new initial solution once its improvement heuristic has met the stopping criterion. To evaluate the different versions of our GRASP approach, industrial instances are used in the next section. The comparison is made according to the given preferences and the quality indicators described in Section 5.3.3. Section 5.5.3 concludes this chapter by evaluating the impacts of our approach by comparing its results to the actual results of the studied diffusion area.

5.5 Numerical Results

The objective of this section is to study the different proposed approaches to handle the multiobjective aspect of an industrial scheduling problem. The three approaches are SA-I, SA-II, and AMOSA, for which, we used the following identical parameter settings for all numerical experiments,: A cooling factor $P_c = 0.99999$, a number of samples $P_s = 100$, a maximum number of non-improving iterations $P_m = 100\,000$, a temperature percentile of $P_t = 5\,\%$, and a perturbation intensity $P_i = 5$. All heuristics are run only once, and six parallel threads are used in all runs of the GRASP based approach. The two instance sets of Section 4.5.2 are again used to conduct the numerical experiments. The first set of instances, called LI, contains 15 large industrial instances. The number of jobs ranges from 350 to 550. The second set, called VLI, contains ten very large instances where the number of jobs ranges from $1,500$ to $1,800$. For each job in these two instance sets, between one and five operations have to be performed, with three operations on average. The jobs must be scheduled on average on 68 machines, all capable of batching. The batching capacity lies between 2 to 7 jobs. Similarly to Section 4.5.2, the scheduling problem include the following constraints: Release dates, minimum time lags, sequence-dependent setup times, availability constraints, batching constraints. Moreover, the multiobjective scheduling problem includes additional characteristics. The problem also encompasses maximum time lag constraints as soft constraints by minimizing their violation. Four violation costs are defined depending on the impact of the violation on the quality of products. Production targets are also modeled as soft constraints by defining a criterion that reflects the overall level of their satisfaction.

The solutions are evaluated according to five criteria: Total maximum time lag violation severity (TVS), target satisfaction indicator (TSI), weighted number of moves (WNM), weighted flow factor (WFF) and batching coefficient (BC). Each criterion is respectively given a lexicographic rank and a weight as a pair (l, c): TVS (1,1), TSI(2,1), WFF(3,1), WNM(3,1) and BC(4,1). Only the weighted flow factor and the weighted number of moves share the same lexicographic rank. By considering $\alpha = 1$, the objective is to maximize the overall satisfaction disregarding the balancing between the satisfaction of the individual targets. The values of the horizon-dependent criteria (TSI, WNM, and BC) are computed for a horizon of 8 hours for LI instances and 24 hours for VLI instances. The experiments are conducted allowing a computational time of 5 minutes for LI instances and 15 minutes for VLI instances. Using these settings and instances, the different sections below pursue different

objectives. Section 5.5.1 compare the three approaches from the perspective of the decision maker preferences. Section 5.5.2 evaluates the approaches from the perspective of the quality of the final sets of nondominated solutions. Finally, Section 5.5.3 focuses on VLI instances to evaluate the potential impact of the approach developed in this thesis by comparing its results to the actual results of the diffusion area.

5.5.1 Comparison Based on Preferences

The objective of this section is to study the impact of the approach when the preferences are considered during the search process on the final solution quality. In this thesis, it is assumed that the decision maker expresses his preferences through lexicographic ranks and weights. The trade-off is not allowed between criteria with different lexicographic ranks. The importance of criteria with the same rank is expressed through weights. In Section 5.2.3, two approaches that consider the preferences of the decision maker are introduced. The first approach, referred to as SA-I, strictly adheres to the preferences by computing the acceptance probability based on the criteria with the lowest rank. In the second approach, referred to as SA-II, the acceptance probability relies on all criteria and ignores the provided preferences. In addition to comparing SA-I and SA-II, the best solutions for the preferences are selected from the final archives of AMOSA. To make sure that the cost of maintaining the archive does not penalize the results of AMOSA, the implementations of SA-I and SA-II also store the nondominated solutions in passive archives.

The results are reported in Table 5.1 and Table 5.2 for the LI and VLI instances, respectively. The results are reported in terms of the relative deviation to the best value for each criterion among the solutions that are lexicographically the best. As no maximum time lag constraint is violated in any solution of any approach, the results regarding TVS are not reported. The results for the remaining criteria are sorted in the lexicographic order: TSI, WFF, WNM, and BS. The results for each criterion are reported in three adjacent columns, and for each of the approach (SA-I, SA-II, and AMOSA). For the objectives to minimize (WFF), the relative deviations are positive, and the closer the value to 0, the better the solution. For the objectives to maximize (TSI, WNM, and BS), the relative deviations are negative and the closer the value to 0, the better the solution.

Instances	*TSI*			*WFF*			*WNM*			*BC*		
	SA-I	SA-II	AMOSA	SA-I	SA-II	AMOSA	SA-I	SA-II	AMOSA	SA-I	SA-II	AMOSA
1	0.0%	-1.6%	-1.5%	9.6%	0.0%	0.1%	-6.1%	-0.2%	0.0%	0.0%	-0.8%	-0.8%
2	0.0%	-0.6%	-2.6%	10.2%	0.0%	3.7%	-5.9%	0.0%	-2.8%	-1.5%	0.0%	-0.4%
3	-0.1%	-0.2%	-1.1%	5.6%	0.6%	0.0%	-1.9%	0.0%	-1.3%	-5.3%	0.0%	-2.7%
4	-0.9%	-3.7%	-4.2%	18.1%	4.2%	0.0%	-11.4%	-1.8%	0.0%	-4.0%	-3.3%	0.0%
5	-0.3%	-1.7%	-1.9%	13.1%	0.0%	2.2%	-10.9%	0.0%	-2.5%	-2.3%	-0.3%	0.0%
6	-0.2%	-0.6%	0.0%	4.6%	0.0%	1.2%	-4.9%	-4.3%	0.0%	-0.4%	-0.1%	-0.4%
7	0.0%	-0.6%	-1.2%	0.0%	0.0%	0.0%	-4.2%	0.0%	-4.5%	0.0%	0.0%	-1.0%
8	0.0%	-0.6%	-2.6%	0.0%	0.0%	0.0%	-0.6%	0.0%	-0.8%	-3.3%	0.0%	-1.2%
9	0.0%	-0.6%	-1.6%	0.0%	0.0%	0.0%	0.0%	-0.4%	-0.8%	-2.1%	0.0%	-0.9%
10	0.0%	-0.6%	-1.8%	0.0%	0.0%	0.0%	-1.5%	-1.3%	-1.5%	-2.5%	-3.3%	-2.7%
11	0.0%	-2.0%	-3.5%	0.0%	0.0%	0.0%	-3.9%	-6.2%	0.0%	-2.9%	-1.5%	0.0%
12	-0.6%	-4.2%	-5.3%	0.0%	0.0%	0.0%	-4.2%	-2.4%	0.0%	0.0%	-4.5%	-1.2%
13	0.0%	-2.0%	-1.8%	0.0%	0.0%	0.0%	-3.6%	-2.8%	0.0%	-1.5%	0.0%	0.0%
14	-0.4%	-1.0%	-3.2%	0.0%	0.0%	0.0%	-0.2%	-1.2%	0.0%	0.0%	-0.7%	-1.0%
15	0.0%	-0.9%	-21.7%	0.0%	0.0%	0.0%	-2.1%	-1.2%	-19.9%	-6.1%	-1.5%	-0.8%
Average	**-0.2%**	-1.4%	-3.6%	4.1%	**0.3%**	0.5%	-4.1%	**-1.4%**	-2.3%	-2.1%	-1.1%	**-0.9%**
Median	**0.0%**	-0.9%	-1.9%	**0.0%**	**0.0%**	**0.0%**	-3.9%	-1.2%	**-0.8%**	-2.1%	**-0.3%**	-0.8%

Table 5.1 – *Comparison of SA-I, SA-II and AMOSA on LI instances*

Instances	*TSI*			*WFF*			*WNM*			*BC*		
	SA-I	SA-II	AMOSA	SA-I	SA-II	AMOSA	SA-I	SA-II	AMOSA	SA-I	SA-II	AMOSA
1	0.0%	-8.5%	-53.8%	0.0%	70.8%	279.0%	0.0%	-21.3%	-61.7%	-3.2%	0.0%	-5.1%
2	0.0%	-3.6%	-43.6%	0.0%	40.1%	255.6%	0.0%	-8.4%	-51.6%	-3.2%	-0.2%	0.0%
3	0.0%	-11.5%	-43.6%	0.0%	96.1%	240.2%	0.0%	-23.4%	-53.8%	-4.8%	0.0%	-3.1%
4	0.0%	-1.4%	-30.0%	1.5%	70.8%	292.1%	-0.4%	-9.4%	-47.2%	-4.3%	0.0%	-2.8%
5	-0.7%	-0.2%	-36.9%	0.0%	10.8%	223.6%	0.0%	-3.3%	-52.9%	-4.0%	-0.5%	-0.1%
6	0.0%	-4.7%	-41.7%	0.0%	66.0%	261.5%	0.0%	-13.7%	-47.3%	-6.6%	0.0%	-2.5%
7	0.0%	-4.1%	-40.4%	0.0%	47.1%	173.4%	0.0%	-12.8%	-48.2%	-4.3%	0.0%	-0.3%
8	0.0%	-3.0%	-40.3%	0.1%	42.0%	223.1%	0.0%	-9.4%	-49.6%	-5.7%	0.0%	-4.3%
9	0.0%	-13.9%	-43.8%	0.0%	93.6%	193.5%	0.0%	-24.4%	-46.6%	-4.3%	0.0%	-3.3%
10	0.0%	-0.7%	-47.5%	0.2%	7.7%	273.0%	0.0%	-2.3%	-58.0%	-0.1%	-0.1%	-0.9%
Average	**-0.1%**	-5.2%	-42.2%	**0.2%**	54.5%	241.5%	**0.0%**	-12.8%	-51.7%	-4.0%	**-0.1%**	-2.2%
Median	**0.0%**	-3.8%	-42.6%	**0.0%**	56.5%	247.9%	**0.0%**	-11.1%	-50.6%	-4.3%	**0.0%**	-2.7%

Table 5.2 – *Comparison of SA-I, SA-II and AMOSA on VLI instances*

When analyzing the solutions of the LI instances reported in Table 5.1, it appears that the best approach that adheres to the given preferences is SA-I. SA-I obtains the best values for TSI for 14 over 15 instances. As the most crucial criterion after TVS, better values of TSI imply lexicographically better solutions. These results show that adhering to the preferences throughout the search when dealing with the LI instances, leads to solutions that are satisfying regarding these same preferences. However, the results also show that optimizing in priority the most important criterion has a significant negative impact on the least important criteria. Indeed, SA-I leads to solutions with lower quality regarding the three remaining criteria when compared to SA-II and AMOSA. The close values of the average and median of the relative deviations on WFF, WNM, and BC do not allow to draw a conclusion about which approach, between SA-II and AMOSA, performs better. However, as SA-II is better than AMOSA on TSI, it seems that SA-II, as it can be predicted by considering its design, leads to solutions where all the criteria are pulled as close as possible to the best values. Different conclusions can be made when analyzing the results for the VLI instances reported in Table 5.2. As when solving the LI instances, SA-I obtains the best solution for 9 VLI instances over 10 when considering the preferences. Contrary to the LI instances, SA-I is by far better than the two other approaches on all the criteria, except for the least important criterion BC. The results of SA-II and AMOSA can be explained by a significant drop in the number of explored solutions that approximately represents 35% of the number of explored solutions by SA-I. In the industrial context, only instances that are similar to LI instances are going to be solved. VLI instances, as in the case of Section 5.5.3, may be solved exceptionally to validate and quantify the improvements of operational performances that can be brought by an optimized scheduling approach. The results for such instances seem much better with SA-I. This is why Section 5.5.2 only uses the LI instances to compare the three approaches by using quality measures of the produced sets of non-dominated solutions.

5.5.2 Comparison Based on Quality Indicators

An ideal approach for solving a multiobjective optimization problem is an approach that enables convergence with a guaranteed spread of solutions. Convergence ensures that the set of solutions is as close as possible to the optimal Pareto front. To measure the convergence of an algorithm, different indicators are defined in the literature. The hypervolume (HV) and the mean ideal distance (MID) are used here to evaluate the convergence. The reference point having the worst values of the criteria is used to compute the hypervolume. For example, the worst value for TSI and BC is O. When considering only minimization problems, the mean ideal distance (MID) is the average distance between non-dominated solutions and the origin point. The origin point cannot be used in our case since there are at the same time criteria to maximize and criteria to minimize. Instead of the origin point, we consider an ideal point. For example, the ideal value for TSI and BC is 1. Diversity is related to the sparsity of solutions to ensure that the decision maker has multiple representative trade-off solutions among conflicting objectives. To measure the diversity, we use the maximum spread (D) which gives the longest diagonal of the hyperbox formed by the extreme values

of the criteria. We also use the spacing metric (SP) that represents the average distance between consecutive solutions in the archive. The results related to convergence indicators are reported in Table 5.3 and those related to diversity indicators can be found in Table 5.4.

Instances		***HV***			***MID***	
	SA-I	SA-II	AMOSA	SA-I	SA-II	AMOSA
1	98878934	96643427	102762325	9115	7484	9226
2	122149619	122149641	117219409	6817	6391	8052
3	58489634	58967724	61630159	9347	10937	11776
4	74860649	73930565	82180998	9987	9897	10830
5	53787143	54183952	56026335	11808	11796	12389
6	46340977	48382832	49075639	13004	12140	13577
7	63342626	63193650	67070949	15013	14947	15605
8	52718120	53399896	51175823	13912	13779	15111
9	54658751	55458791	56462692	12720	12629	13952
10	43250728	43088713	45432540	15978	15637	16686
11	73233089	72036060	75912776	13714	13476	14500
12	67077945	66614927	69416988	12018	11471	12774
13	64070792	63893194	67087905	11544	10869	11206
14	55138201	55703626	55873980	12534	12627	13922
15	45466194	46109040	45662555	13249	12970	15018
Average	-3,8%	-3,5%	**-0,6%**	3,7%	**1,2%**	12,1%
Median	-3,8%	-4,0%	**0,0%**	1,8%	**0,0%**	10,5%

Table 5.3 – *Comparison of SA-I, SA-II and AMOSA for hypervolume HV and mean ideal distance MID*

Before analyzing the results of the experiments, we first discuss the size of the archive and the impact of its maintenance during the search on the number of explored solutions. When solving the LI instances, the average number of nondominated solutions in the final archive of SA-I, SA-II, and AMOSA is 21, 18 and 61, respectively. Even with a larger final archive, the impact of maintaining the archive within AMOSA is approximately close to the one observed within the two simulated annealing approaches. On average, the number of visited solutions is reduced by 5% when maintaining the archive compared to the number of explored solution within simulated annealing approaches without archive. Regarding convergence, Table 5.3 reports the hypervolume HV and mean ideal distance MID for each approach. Recall that the larger HV, the better the approach, and the smaller MID, the better the approach. The table

also provides the average and the median over the LI instances of the relative deviations to the best values (maximum value for HV and minimum value for MID). According to HV, the results show that AMOSA performs better than SA-I and SA-II, and that there is no significant difference between these two last approaches. According to MID, the results show that SA-II is better than SA-I which in turn performs better than AMOSA. In the previous section, the results show that SA-II is the best approach that optimizes all the considered criteria simultaneously and this is confirmed with the lowest values on MID.

Instances		***D***			***SP***	
	SA-I	SA-II	AMOSA	SA-I	SA-II	AMOSA
1	5199	51	4747	2219	13	1330
2	2672	81	4472	760	17	1563
3	2811	266	4638	562	96	1453
4	2923	125	4247	685	12	1540
5	1375	151	2541	398	46	762
6	2646	375	3393	921	108	1179
7	2646	177	3783	842	33	997
8	1150	150	4524	316	59	1487
9	864	49	4279	145	11	1485
10	1523	51	3655	338	21	1045
11	1853	571	3249	498	169	1056
12	2835	517	4692	1021	121	1401
13	3766	703	2662	1200	204	776
14	474	425	4598	74	131	1795
15	1946	25	3511	482	13	503
Average	-42,6%	-93,6%	**-2,5%**	2684,5%	**5,0%**	4563,2%
Median	-40,3%	-95,3%	**0,0%**	759,2%	**0,0%**	2440,5%

Table 5.4 – *Comparison of SA-I, SA-II and AMOSA for maximum spread D and spacing SP*

Regarding diversity, Table 5.4 reports the maximum spread D and spacing SP for each approach. Recall that the larger D, the better the approach, and the smaller SP, the better the approach. The table also provides the average and the median over all the LI instances of the relative deviations to the best values (maximum value for D and minimum value for SP). According to D, the results show that AMOSA performs better than SA-I which in turn perform better than SA-II. When considering SP, the order is reversed as SA-II is the best approach while AMOSA is the approach that produces archives where solutions are far

from each other. The different quality indicators help better understand how the different approaches work when solving the LI instances. SA-II converges to the region that is closer to the ideal point, and the produced archive contains solutions that are crowded in this region. This can be seen through the low values of MID and SP. This convergence comes with the drawback of only producing an approximation of a portion of the actual Pareto front and not its full extent which can be seen through the low values of D and which may also explain the low values of HV. The results show that AMOSA is quite the opposite of SA-II. The values of D and SP show that AMOSA produces archives covering larger portions of the Pareto front with solutions that are more distant from each other when comparing to the two other approaches. Maintaining a rich set of diverse solutions and making the approximation set closer to the Pareto front comes with the drawback of a low convergence to the trade-off region where solutions are the closest to the ideal point. As the four quality indicators always rank SA-I in second position, the behavior of SA-I lies between the two extreme behaviors of AMOSA and SA-II. SA-I converges to the trade-off area faster than AMOSA while ensuring a better spread than SA-II as it is designed to converge to the extreme values of the most important criteria.

The objective of the experiments is not to decide which approach to use. The numerical results, while allowing a better understanding of the different approaches, can be considered as preliminary. No definitive conclusions can be drawn without extensive experiments where additional quality indicators are used, and the different parameters of the approaches are finely tuned. In the numerical experiments of this chapter, the three approaches share the same parameters such as the initial temperature, the cooling factor and the number of non-improving moves. The setting of these parameters may be more suitable for one approach than another. As six threads are used to improve the initial solutions, the number of non-improving moves is never reached by SA-I and SA-II which means that only six initial solutions are improved within the GRASP approach. On the contrary, up to 12 initial solutions are improved for some of the LI instances when AMOSA is used as an improvement heuristic. This means that AMOSA spends $100,000$ iterations without updating the archive. Maybe more importantly than conducting additional numerical experiments, a better understanding of the preferences of the decision maker is critical to choose an approach. If the preferences are precise instructions rather than general orientations, the numerical results show that SA-I is the most fitting approach. However, our practical experience showed us that, when a decision maker formalizes his preferences, they should be often be understood as general orientations. In this case, even if an approach is better than another one on different quality indicators, choosing a final solution is still a challenge that cannot be taken on without a clear understanding of the preferences. Finally, it should be noted that the parallel implementation of the GRASP approach allows different improvement approaches to be used in different threads. This may result in a global approach that combines the different strength of the three improvement approaches.

5.5.3 Potential Impacts of the Proposed Approach on Industrial Instances

The purpose of this section is to validate the approach described throughout the thesis by comparing its solutions to the actual solutions of the studied work area. To conduct the comparison, all relevant data of 10 different days over six months are extracted from the Manufacturing Execution system. The obtained instances are the VLI instances that are already used in the previous sections. The choice of a period of one day is mainly motivated by the fact that production targets are set daily. To perform a fair comparison, all the constraints and events that can affect the schedule quality must be reflected in the instances. The instances are then built as follows:

- All lots that were in the area during the chosen period are included in the set of jobs to schedule.
- Due to reentrance, lots may return several times to the work area during a day. The reentrance makes it impossible to consider as equivalent a lot and a job. Instead, a job is equivalent to the couple lot and release date. For example, a lot that arrived in the area at 06h00, left the area for another one at 14h00 pm to come back again at the diffusion area at 19h00 will be represented by two distinct jobs. This modeling may results in some inconsistency in the solutions returned by an approach that solves the problem a posteriori. For example, operations of the job available at 06h00 may be performed after 19h00 which does not make sense.
- Concerning the characteristics of lots, their size and release dates do not change over time while their priority is thought as a dynamic attribute to influence the decisions of the operators. To make the study manageable, it is assumed that the priority of a job is the priority of the represented lots just after it was released in the area. This assumption does not introduce any bias in the comparison as the computation of the performance indicators of the actual solution makes the same assumption.
- An availability constraint is attached to the problem whenever a machine became unavailable due to preventive or curative maintenance or to quality problems.
- At the beginning of the horizon, a machine can already be processing batches. These situations are also modeled through availability constraints from a machine perspective, and future release dates from a job perspective.
- To model the wet benches, the analytical data-driven modeling described in Section 4.5.1 is used.
- In opposition to furnaces on which the quality tasks described in Section 2.3.10 do not utilize production capacity, these tasks must be separately carried out on wet benches. Due to the unavailability of historical data, these constraints are not considered in the experiments. Thus we consider during the analysis of the results that quality tasks consume a capacity that is equivalent to what is required by 1% of moves.

- Another feature that was not described in this thesis is measurement operations. After some production operations, some jobs are measured on inspections machines to monitor the process stability of the machines and the quality of products. It should be noted that not all jobs are measurable and measurement operations do not imperatively follow a production operation. These operations are not explicitly considered in the problem. To make the model realistic, the transport times between machines are overestimated to take into consideration the waiting times of measurable jobs in front of inspection machines and the corresponding measurement times. Minimum time lags are used to model both transport times and the needed time by the measurement operations, with a minimum duration of one hour. Hence, a minimum time lag constraint is systematically added between any two operations of any job, whether a job is measurable or not, and whether there is an actual measurement step or not between the corresponding operations.

The obtained VLI instances are then used to conduct the experiments. The number of jobs varies between $1,500$ and $1,800$ with an average of three operations per job. Jobs must be scheduled on average on 68 machines, all capable of batching. The batching capacity lies between 2 and 7 jobs. The resulting scheduling problem includes the following constraints: Release dates, minimum and maximum time lags, sequence-dependent setup times, availability constraints and batching constraints. On average, there are 25 production targets classified into two subgroups: Priority targets translating the objective of ensuring the linearity of the production line at the fab level; less important targets stemming from the monthly production plan. The solutions for the problems described by the instances, either actual ones or those found by our approach, are evaluated according to five criteria: Total maximum time lag violation severity (TVS), target satisfaction indicator (TSI), weighted number of moves (WNM), weighted flow factor (WFF) and batching coefficient (BC). By considering $\alpha = 1$, the objective is to maximize the overall satisfaction disregarding the balancing between the satisfaction of the individual targets. To facilitate the interpretation of the results, the job weights are ignored when computing WNM and are only considered when computing WFF. The values of the horizon-dependent criteria (TSI, WNM, and BC) are calculated for a horizon of 24 hours for the actual solutions and the solutions of our approach. As extracted data only include operations that are processed over the chosen horizon of 1 day, the values of the horizon-free criteria are also computed over a horizon of one day for the actual solutions. However, when optimized, the values of these criteria are not restricted to the horizon in our approach.

Before carrying the experiments, different choices must be made. First, a selection strategy must be chosen among all those proposed in this thesis and those proposed by Knopp (2016). In Section 4.5.2, the numerical results point out that the best strategy when dealing with the VLI instances is the integrated strategy. In the current chapter, different approaches to handle the multiobjective aspect of the industrial scheduling problem were proposed. In Section 5.5.1, the numerical results show that better results are obtained when using SA-I. Thus, to compare the solutions proposed by our approach with the actual solutions, we use the integrated strategy to improve solutions during the start time computation and SA-I to

handle the given preferences during the search process. The experiments are conducted by allowing a computational time of 30 minutes. In the industrial setting, the proposed approach should solve instances such as the LI instances used in the previous section, and it is practical to allow a computational time of five minutes. The choice of 30 minutes as the computational time is motivated by the fact that the average number of explored solutions in 30 minutes for instances of one day, which is approximately $650,000$, is equal to the average number of explored solutions for instances to be solved in real conditions during five minutes. Similarly to the previous sections, the results of our approach are reported as a relative deviation to the actual results instead of the best values. The detailed results per instance are reported in Table 5.5 which are aggregated in Table 5.6.

Instances	*TVS*	*TSI*	*WWF*	*WNM*	*BC*
1	-100%	7.7%	-9.4%	9.2%	4.8%
2	-100%	22.1%	-5.9%	2.7%	5.0%
3	-100%	10.6%	11.4%	2.3%	5.2%
4	-100%	9.3%	-19.9%	11.2%	4.2%
5	-100%	8.3%	4.6%	9.7%	4.8%
6	-100%	12.8%	-5.1%	3.4%	2.8%
7	-100%	9.6%	-5.8%	2.1%	7.0%
8	-100%	7.7%	-7.0%	3.6%	8.4%
9	-100%	7.9%	-15.4%	7.4%	7.5%
10	-100%	10.1%	4.3%	11.1%	6.6%

Table 5.5 – *Detailed results comparing the solutions determined by our approach and the actual solutions*

	TVS	*TSI*	*WWF*	*WNM*	*BC*
Average	-100%	10.6%	-4.8%	6.3%	5.6%
Median	-100%	9.6%	-5.8%	6.3%	5.2%

Table 5.6 – *Aggregated results comparing the solutions determined by our approach and the actual solutions*

The numerical results show that our approach can bring a significant improvement in the operational performances of the diffusion area. Indeed, there are eight instances out of ten where the proposed solutions are dominating the actual solutions on all criteria. For

the two other instances (3 and 5), the actual solutions are only better for the weighted flow factor (WFF). Because of the preferences, all the solutions obtained by our approach are actually better than the actual ones. While there are on average four maximum time lags that are violated in the actual solutions, no violation is observed in the proposed solutions. The satisfaction of the production targets and the utilization of the batching capacity are approximately increased by 10% and 5%, respectively. The weighted flow factor, representing the waiting time of jobs weighted by their priorities in front of the different machines in the area, is approximately decreased by 5%. Finally, if we consider the weighted number of moves and the assumption that the integration of the quality task will reduce the value of this criterion by 1%, the proposed solutions still improve the actual performances by more than 5%. These results demonstrate that there is room for improving the operation performances of the studied work area and that the proposed approach can bring such improvement as it efficiently solves the industrial scheduling problem while taking the rich set of constraints and criteria into account.

5.6 Conclusion

By detailing how we handle the multicriteria aspect of the industrial scheduling problem, this chapter achieves the description of the whole approach developed through the two preceding chapters. Three approaches for optimizing multiple criteria are examined. In the two first approaches, the decision maker is provided with a flexible modeling of his preferences depending on whether the trade-off is permitted between any pair of the considered criteria. The two approaches use differently these preferences during the search process and stores the set of nondominated solutions in a passive archive. These two approaches are compared to a third approach from the literature that uses the dominance status between the current solution and the set of nondominated solutions stored in an active archive. The comparison is performed based on the given preferences and known quality indicators and demonstrates that each approach can be more suitable depending on the context. This chapter ends with numerical experiments that attest the significant improvement that can be brought by the proposed approach.

The numerical results show that each of the three approaches investigated in this chapter may be more interesting for a given context. In order to choose the approach to use or to show how to combine all of them, more numerical experiments should be carried out, and additional quality indicators may be necessary. In addition to more experiments, a better perception of the preferences of the decision maker is a prime condition to design the final approach. This understanding will also benefit to the design of a suitable procedure that picks a final solution from the final archive.

Chapter 6

Conclusions and Perspectives

6.1 Conclusions

The objective of this thesis is to model and solve a multiobjective complex job-shop scheduling problem in the diffusion area of semiconductor manufacturing facilities. The rich set of constraints and criteria that are considered lead to a general scheduling problem that can be applied to other work areas in semiconductor manufacturing and other industrial contexts. The starting point of this thesis is the batch-oblivious approach proposed in Knopp et al. (2017) for solving complex job-shop scheduling and the approach proposed by Bitar (2015) regarding multiobjective optimization. The contributions of this thesis are the improvement of the efficiency of the batch-oblivious approach, its extension to additional constraints and a better management of the optimization of multiple objectives. This leads to a solution approach that can efficiently tackle the industrial scheduling problem described in Chapter 2.

To ensure that feasible and realistic schedules are proposed, a large set of constraints are identified and modeled within the diffusion area. The original batch-oblivious approach is extended by considering additional constraints that model essential features of the industrial scheduling problem: Minimum time lags, minimum batch size and availability constraints. For example, minimum time lag constraints are used to model actual minimum time lags, but also transport times and are part of the set of constraints used to model complex machines as black boxes while this modeling, referred to as data-driven analytical modeling, is kept realistic. Also, regarding complex batching machines, the batch-oblivious conjunctive graph is generalized to offer the possibility of modeling at the same time the internal structures and batching capacities of machines. It is shown that the use of the extended batch-oblivious conjunctive graph has the advantage of providing more accurate predictive schedules and requires less data collection and maintenance. The approach is also extended by integrating more relevant industrial criteria: Weighted number of moves and batching coefficient. As the proposed method is meant to be used within a rolling horizon framework, the discounted weighted number of moves is a new criterion that is introduced to prioritize the throughput and the priority adherence of a solution at the beginning of the scheduling horizon. An original contribution of this thesis is the integration of production targets, which are set to guide the local work areas towards the realization of daily and monthly objectives of fabs. The Target Satisfaction Indicator (TSI) is proposed as a criterion that enables the decision

maker to emphasize either the overall satisfaction of the production targets or the balancing between the production targets.

Besides extending the original batch-oblivious approach, this work improves its efficiency. An important component within the batch-oblivious approach is the capability to improve a schedule during the start time computation. When a batch is incomplete, a search for additional operations that can be selected to complete the batch is triggered. New selection strategies, such as the integrated strategy, are introduced, and experimental results attest their low computational cost. Apart from its efficiency, the active strategy intentionally inserts delays to complete batches, which leads to more interesting and acceptable solutions from the industrial point of view.

6.2 Perspectives

The different experiments conducted in this thesis show that the proposed approach obtains promising results. Though the approach can be used in the industrial setting with a significant impact on the operational performances, the solution approach can still be improved. In addition to the different perspectives described in the different chapters, this section lists new perspectives that we have already started to explore or those to be studied in the future.

Section 6.2.1 is dedicated to the description of ideas that can improve the different components of the proposed solution approach. As already mentioned in Chapter 3, the GRASP approach obtains way much better results when the simulated annealing approach is used to improve the initial solutions compared to the hill climbing and the tabu search approaches. The low performance of the tabu search approach can be explained by the large size of the neighborhood when dealing with industrial instances. The computational cost of fully exploring the neighborhood is too prohibitive. To make such an approach suitable to solve the industrial instances, new efficient neighborhood functions should be designed. The different identified perspectives in this direction and the different cautions that must be considered are described in Section 6.2.2. Besides the improvement of the approach efficiency and effectiveness and though a rich set of features are already considered in the studied scheduling problem, the model still can be enriched by integrating other operational decisions. Section 6.2.3 provides some examples of possible integrations. Finally, some industrial perspectives are discussed in Section 6.2.4.

6.2.1 Improvement of the Solution Approach

This section explores several perspectives to improve the different components of the solution approach. Regarding the solution representation, Chapter 4 proposes an extended batch-oblivious conjunctive graph that can model complex batching machines in detail. However, instead of the route graph modeling, the data-driven analytical modeling described in Section 4.5 is used in the industrial application as the data management module was initially designed to collect the data this modeling requires. Before modeling in detail complex batching

machines, it must be ensured that the size increase of the extended batch-oblivious approach due to the detailed modeling of machines does not negatively impact the efficiency of the solution approach. An experimental study can help to evaluate this impact. If the increase in the size of the extended batch-oblivious conjunctive graph negatively impacts the efficiency of the metaheuristic, the possibility of combining the two modelings can be interesting. The GRASP metaheuristic can be run with the data-driven analytical modeling, and each improving solution is recomputed using the route graph modeling. Comparing the solutions obtained with only the route graph modeling and those obtained combining the route graph modeling and the data-driven analytical modeling should help to decide which approach is the most relevant.

Modeling complex machines where the handling part and the internal scheduling algorithm cannot be ignored is another challenge to address. The route graph modeling is only capable of capturing the complex behavior resulting from the structure and the internal constraints of the machines. When this complex behavior is also due to the internal scheduling algorithm, using the data-driven analytical modeling may be more interesting if it is not possible to include the control logic in the detailed modeling.

Another component of the approach that requires special attention is the evaluation of solution quality. The importance of this procedure lies in the fact that it is the computationally most expensive component of the approach. This procedure is even more important within the batch-oblivious approach as it also improves schedules during the traversal of conjunctive graphs. Chapter 3 proposes efficient strategies such as the integrated strategy that enables a solution to be dynamically improved with little computational effort. The numerical results show that three different strategies are dominating: The resequencing strategy for the random instances, the integrated and active strategies for large industrial instances. An interesting approach that can increase the positive impact of this procedure on the quality of the final solution determined by the global approach is to choose the strategy that is the most adapted to the solved instance and to the stage of the search process. The use of machine learning techniques can be a good solution to automate the selection and the combination of the search strategies.

An interesting approach to reduce the computational time of the solution quality evaluation is to incrementally maintain the longest paths as proposed by Michel and Van Hentenryck (2003), Pearce and Kelly (2007) and Sobeyko and Mönch (2016). The idea is to recalculate, after each move, only the start times that might have changed instead of updating the start time of every scheduled node in the conjunctive graph. It can be assumed that the number of explored solutions can significantly be increased when partially updating the conjunctive graph. This is, in particular, promising for large problem instances, such as the studied industrial ones, since the expected gain grows with the number of nodes in the graph. Also, if the application of this idea results in a significant reduction in the computational cost of the solution quality, the potential negative impact of modeling complex machines in detail may be lower and the choice of the route modeling safer. However, even though this idea can be applied to flexible job-shop scheduling problems with standard objective functions, different reasons can be mentioned that makes not straightforward the application of this idea

in the batch-oblivious approach with the objective functions of the industrial context:

- In addition to the changes induced by the application of the neighborhood operator, the new solution depends on the changes that are performed during the adaptive start time computation algorithm. When considering a flexible job-shop scheduling problem, the region of affected nodes does not change during the schedule computation while, in the batch-oblivious approach, this region may potentially change each time a node is selected to complete a batch.

- Also, if the start times are only updated for the nodes that are topologically ordered after the directly impacted nodes, it could be possible to fill batches that are topologically ordered before these nodes as the invariant can be satisfied for nodes that are affected by the move.

- Regarding the objective functions, the classical ones that are studied in the literature are based on the completion times of jobs. The computation of several criteria considered in this thesis, such as the weighted number of moves, the target satisfaction indicator and the time lag violation severity, are based on the start and completion times of operations. In this case, maintaining the values of these criteria incrementally must be done in parallel to the start times of operations.

In addition to the improvement of the different components, the performance of the GRASP approach can likely be improved by fine-tuning its parameters. Even if some parameters are adapted to each instance through the sampling strategy, thorough numerical experiments should be performed to determine the best values for the minimum temperature and the maximum number of non-improving moves. The GRASP approach could be improved by adding a memory mechanism through path-relinking (see Resendel and Ribeiro (2005)). When considering multiple criteria, if an archive of nondominated solutions is maintained, the elements of the archive are the best elite solutions determined during the search. Path-relinking can provide an intensification strategy by exploring trajectories that connect the nondominated solutions. The performances of the GRASP approach may be improved by applying other metaheuristics than simulated annealing. Tabu search is an attractive alternative as it is among the most effective approaches for solving scheduling problems. Experimental results that are not reported in this report show that tabu search is not performing well compared to simulated annealing when all possible moves are explored, i.e., all operations and all feasible insertion positions are considered. The large size of the neighborhood can explain this poor performance. For example, a solution for an industrial instance has dozens of thousands of neighbors, and tabu search only performs a few iterations if the computational time is limited to a few minutes. The effectiveness and efficiency of a tabu search approach heavily depends on the neighborhood functions which are addressed in Section 6.2.2.

6.2.2 Design of New Neighborhood Functions

In the industrial setting, it is required to determined good solutions in a short computational time. To achieve this, the heuristic to be used must explore as many solutions as possible. The idea of a partial update of the start times can contribute to the increase in the efficiency of our solution approach. Besides increasing the number of explored solutions in a maximum computational time, a more promising and challenging perspective is to increase the number of effective moves. In this thesis, depending on the solved problem, two neighborhood functions are used:

- In the approach described in Chapter 3, a neighbor is obtained by a move where a node and a feasible insertion position are randomly selected. This neighborhood structure is referred to as ($\mathcal{N}^1$).
- In the approach described in chapters 4 and 5, as sequences of operations are to be moved together while these operations may require multiple resources and impose resource acquisition constraints, neighbors are selected differently. After randomly selecting a movable component, a move works in two phases: First, it removes all nodes belonging to a movable component from the conjunctive graph. Second, it inserts all nodes that belong to a movable component into the conjunctive graph. The efficient insertion of the sequence of operations is challenging since a meaningful and feasible insertion position has to be found while coping with multiple resources per operation and resource acquisition constraints. Then, the insertion technique for nodes proposed by Kis (2003), which avoids enumerating dominated insertion positions in the sense of the makespan, is adapted by Knopp (2016) by considering resource acquisition constraints. One of the insertion positions determined by the adapted algorithm is randomly selected. This neighborhood structure is referred to as ($\mathcal{N}^2$).

In both approaches, the nodes to move are randomly selected. Regarding the determination of insertion positions, the first approach randomly picks one among all feasible insertion positions. In the second approach, one insertion position is randomly selected among all dominating insertion positions in the sense of makespan. There is clearly room to increase the proportion of effective moves. It is interesting for example to generalize the insertion technique for nodes proposed by Kis (2003) to other optimization criteria. However, it is a more manageable task to design new neighborhood functions when solving the problem of Chapter 4 without considering the detailed modeling of complex machines. In the remainder of this section, preliminary findings and future perspectives in this direction are described.

To improve the performance of our approach, it is important to be able to identify improving moves. Identifying such moves can be done through characterizing the properties of the interesting nodes and insertion positions or by designing efficient procedures that can evaluate the effect of a move on the objective function without actually making the move. Several efficient heuristics proposed for the job-shop scheduling problem (e.g., Van Laarhoven et al. (1992), Taillard (1994), Mati et al. (2011)) use the operation criticality as a property to select

operations to move. In addition to restricting the set of candidate operations to critical ones, other approaches (e.g., Nowicki and Smutnicki (1996), Mati (2010)) only focus on operations that are at the extremes of critical blocks. Before describing some perspectives in this direction, we review two works (Tamssaouet et al. (2018) and Tamssaouet et al. (2018a)) that study sub-problems of the complex job-shop scheduling problem described in Chapter 4 and for which the results are not reported in this thesis.

In Tamssaouet et al. (2018a), we study the scheduling problem of minimizing makespan on parallel identical batching machines with dynamic job arrivals and incompatible families. The study of the batch-oblivious conjunctive graph shows it lacks a fundamental property of efficient neighborhood functions (Van Laarhoven et al. (1992)): The removal of an operation from a machine sequence cannot increase start times. To ensure this property, the construction algorithm is modified so that operations in the same batch are sequenced in the non-increasing order of their job availability. With the first neighborhood function ($\mathcal{N}^3$), only critical nodes are candidates, while in the second function ($\mathcal{N}^4$), the set of candidates is restricted to the set of nodes that are critical and in the first position of their batch sequences. The experimental results show that the static strategy using ($\mathcal{N}^3$) is significantly better than when using ($\mathcal{N}^1$) where all nodes are potential candidates. With the same strategy, no significant difference is observed between the results obtained using ($\mathcal{N}^4$) and the ones obtained using ($\mathcal{N}^1$). When allowing the solution to be improved during the traversal of the graph, no significant difference is observed between the three neighborhood functions, while the results are globally better than when using the static strategy. The results of using neighborhood functions such as ($\mathcal{N}^3$) cannot be impressive as it is known that the use of critical paths is less appealing in parallel machine scheduling problems compared to job-shop or flow-shop problems. This work was conducted as a first step to make our heuristic more efficient when solving the industrial scheduling problem considered in this thesis.

However, in the perspective of improving our solution approach by increasing the number of effective moves, different cautions should be considered.

- First, attention must be given to the definition of critical operations and their properties. In Tamssaouet et al. (2018), we address the job-shop scheduling problem in which the machines are not available during the whole planning horizon and to minimize the makespan. The disjunctive graph model is used to represent job sequences and to adapt and extend known structural properties of the classical job-shop scheduling problem to the problem at hand. These results have been included in two metaheuristics (Simulated Annealing and Tabu Search) with specific neighborhood functions and diversification structures. Computational experiments on problem instances of the literature show that our Tabu Search approach outperforms Simulated Annealing and existing approaches. In this paper, it is brought out that, when considering availability constraints, a zero slack for an operation is neither a necessary nor a sufficient condition for being a critical operation. When considering regular classical criteria, the definition of critical operations and their properties must be reviewed when a new constraint is introduced.

- Another important point of attention when designing neighborhood functions that rely on critical operations is the computational cost of identifying such operations. As it is necessary to compute the latest start times, the number of explored solutions is on average half the number of explored solutions when only the earliest start times are computed. This computational cost may be prohibitive in different situations:
  - When the average ratio of critical operations for the studied instances is large, where it may become uninteresting to reduce the set of candidates operations to the critical ones.
  - When these neighborhood functions are used within heuristics where a unique neighbor is visited such as simulated annealing. It may be more interesting to use heuristics where a set of neighbors is visited, such as tabu search so that the cost of identifying the critical operations is amortized. This is more beneficial when considering the fact that only the computation of the earliest start times is necessary to evaluate the neighbor quality. The computation of the latest start times becomes essential for the accepted neighbor.
  - When it is possible to find a move of a non-critical operation that leads to a better improvement than any move of a critical operation. In the case of a job-shop scheduling problem, no improvement can be obtained by moving a non-critical operation. When considering the makespan minimization for the scheduling of parallel batch processing machines, it is shown in Tamssaouet et al. (2018a) that a move of a non-critical operation can improve the solution, but there is always a move of a critical operation that can lead at least to the same improvement. This is no longer valid when considering batching constraints within a flexible job-shop scheduling problem. A simple example can be constructed to show that a move of a non-critical operation leads to a larger improvement than any move of critical operations. However, the experimental results that are not reported in this thesis show that the occurrence frequency of such situations is very low. Using a hill-climbing heuristic, the numerical results show that a move of a non-critical operation is better than all moves of critical operations only in 0.5% of the iterations.

- In addition to the challenges raised above, there are different limitations when considering neighborhood functions that rely on the notion of critical operations. In the industrial context, some of the defined criteria are regular: Weighted flow factor, move-related criteria, and target satisfaction indicator. The weighted flow factor, based on the job completion times, is quite similar to the standard weighted total completion time and the notion of critical operation is still relevant. Even though they are regular, other criteria in the list above are however different as the notion of critical operation is not relevant. When considering such criteria, there is a need to characterize interesting moves. For example, it can be shown that there is no direct improvement after moving an operation having a resource predecessor that completes its processing outside of the

horizon. The irrelevance of operation criticality can also be observed when considering non-regular criteria, such as the time lag violation severity and the batching coefficient.

All the reasons above make the definition of efficient neighborhood functions based on the criticality of operations challenging, and even impossible depending on the criteria. The problem becomes more complex when considering multiple criteria simultaneously. A perspective that seems both relevant and challenging is to design scoring functions that evaluate, even approximately, the impact of moving operations and the impact of insertion positions. When optimizing classical objective functions such as the makespan or the total weighted completion time, operation properties such as being critical or being located in the extremes of a critical block can be expressed through boolean functions. If specific criteria such as the weighted number of moves are optimized, it may be possible to design adequate functions to assess the impact of moving operations. The main advantage of such functions is the possibility to aggregate the score for each criterion in multiobjective optimization.

Instead of characterizing interesting operation candidates and insertion positions, the second alternative is to design efficient functions that evaluate the impact of a move on the objective function without actually making it. Such functions for regular criteria can be found for the job-shop scheduling problem in Mati et al. (2011) and the flexible job-shop scheduling problem in García-León et al. (2015). When well designed, these functions can be very effective in characterizing moves. For instance, the evaluation function in Dauzère-Pérès and Paulli (1997) discards all the moves that are identified as uninteresting in Nowicki and Smutnicki (1996). Generalizing these functions when considering batching constraints is a promising perspective. The batch-oblivious graph can support the design of such functions, although it can be challenging because of the dynamic change of edge weights. Finally, regarding the neighborhood functions, it is felt that analyzing their connectivity property is important.

6.2.3 Integration of Scheduling Decisions with other Operational Decisions

An interesting long-term perspective could be to integrate scheduling decisions with the different operational decisions that are described in Section 1.2. In the current approach, the transportation of jobs between the different machines is modeled through minimum time lags. A challenging problem is to solve in an integrated way the production and transportation scheduling problems, by explicitly considering the vehicles of the AMHS along with the production machines. We believe that the extended batch-oblivious conjunctive graph is already rich enough to solve such a problem.

In addition to the transport time, the duration of minimum time lags includes the possible necessary time to measure a lot on inspection machines before going to the next production operation. It should be noted that not all jobs are measurable, and measurement operations are not mandatory. Due to the limited measurement capacity, only jobs that can significantly reduce the risk should be measured (see e.g. Dauzere-Péres et al. (2010) and Rodriguez-

Verjan et al. (2015)). Two levels of integration can be studied when taking the measurement and inspection operations into account, and in both situations, it becomes necessary to compute the risk level of machines during the schedule computation and to consider additional criteria.

- The first level is to ignore the measurement capacity while making sure that measurable lots are processed on machines that see their risk increases.
- In addition to production operations, the second level of integration is to schedule measurement operations on the inspection machines.

Another promising perspective is to better integrate availability constraints that are partially imposed by preventive maintenance planning. In this thesis, it is assumed that all unavailability periods have a fixed start time and a fixed duration. It might be too challenging to solve in an integrated way production and maintenance scheduling, but it can be practical and beneficial to assume that unavailability periods are flexible. Instead of fixed unavailability periods, it is possible to consider that preventive maintenance operations should be completed within a time window. This flexibility can improve the operational performances of the work area.

6.2.4 Industrial Perspectives

Finally, let us conclude this manuscript by providing some industrial perspectives. The different numerical experiments show that the proposed approach can bring substantial improvement in the operational performances. By improving the efficiency of the approach and taking the realistic modeling of the scheduling problem into account, we believe that the approach is ready to be industrialized. However, this seems insufficient to convince management up to now. In addition to taking more time to communicate better and explain the general principles of the approach, it may be interesting to look for other validation approaches that can convince management of the advantages of using optimized scheduling systems.

However, we believe that real-time validation is the best approach to draw conclusions. First, all the necessary components for a scheduling system are already available. Data and the display interface are available in the semiconductor industry. The optimization engine and the data management module were developed during this thesis. The solutions of the optimization engine should be now industrialized in the display interface. In the fab where this thesis was conducted, the risk of disrupting the work area is very low when performing real-time validation. The operators still have the freedom to derogate to the solutions proposed by the optimized scheduling approach. If the industrialization proves to be a success, it is practicable and relevant to apply the proposed approach to other work areas of a fab. Because of the rich set of constraints and criteria be considered in our method, it should be possible to quickly deploy it in other areas such as the photolithography and ion implantation areas.

Bibliography

Adams, J., Balas, E., and Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401.

Aggoune, R. (2002). *Ordonnancement d'ateliers sous contraintes de disponibilité des machines*. PhD thesis, University of Metz, France.

Aiex, R. M., Binato, S., and Resende, M. G. (2003). Parallel grasp with path-relinking for job shop scheduling. *Parallel Computing*, 29(4):393–430.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345–378.

Allahverdi, A., Ng, C., Cheng, T. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European journal of operational research*, 187(3):985–1032.

Angel, E., Bampis, E., and Gourvés, L. (2004). Approximating the pareto curve with local search for the bicriteria tsp (1, 2) problem. *Theoretical Computer Science*, 310(1-3):135–146.

Anthony, R. N. (1965). *Planning and control systems: A framework for analysis [by]*. Division of Research, Graduate School of Business Administration, Harvard ....

Artigues, C., Huguet, M.-J., and Lopez, P. (2011). Generalized disjunctive constraint propagation for solving the job shop problem with time lags. *Engineering Applications of Artificial Intelligence*, 24(2):220–231.

Artigues, C., Péres, S. D., Derreumaux, A., Sibille, O., and Yugma, C. (2006). A batch optimization solver for diffusion area scheduling in semiconductor manufacturing. *IFAC Proceedings Volumes*, 39(3):733–738.

Azem, S., Aggoune, R., and Dauzère-Pérès, S. (2012). Heuristics for job shop scheduling with limited machine availability. *IFAC Proceedings Volumes*, 45(6):1395–1400.

Bai, X. and Gershwin, S. B. (1989). A manufacturing scheduler's perspective on semiconductor fabrication. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE MICROSYSTEMS RESEARCH CENTER.

Baker, K. R. and Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: a review. *Operations research*, 38(1):22–36.

Bandyopadhyay, S., Saha, S., Maulik, U., and Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE transactions on evolutionary computation*, 12(3):269–283.

Basseur, M., Zeng, R.-Q., and Hao, J.-K. (2012). Hypervolume-based multi-objective local search. *Neural Computing and Applications*, 21(8):1917–1929.

Bitar, A. (2015). *Ordonnancement sur machines parallèles appliqué à la fabrication de semi-conducteurs : ateliers de photolithographie*. PhD thesis, Université de Lyon.

Bitar, A., Dauzère-Pérès, S., Yugma, C., and Roussel, R. (2016). A memetic algorithm to solve an unrelated parallel machine scheduling problem with auxiliary resources in semiconductor manufacturing. *Journal of Scheduling*, 19(4):367–376.

Błażewicz, J., Domschke, W., and Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European journal of operational research*, 93(1):1–33.

Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J. (2007). *Handbook on scheduling: from theory to applications*. Springer Science & Business Media.

Blot, A., Kessaci, M.-É., and Jourdan, L. (2018). Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation. *Journal of Heuristics*, 24(6):853–877.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308.

Bowman, E. H. (1959). The schedule-sequencing problem. *Operations Research*, 7(5):621–624.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183.

Brockhoff, D. and Zitzler, E. (2007). Dimensionality reduction in multiobjective optimization: The minimum objective subset problem. In Waldmann, K.-H. and Stocker, U. M., editors, *Operations Research Proceedings 2006*, pages 423–429, Berlin, Heidelberg. Springer Berlin Heidelberg.

Brucker, P. (2007). *Scheduling algorithms*. Springer.

Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M. Y., Potts, C. N., Tautenhahn, T., and Van De Velde, S. L. (1998). Scheduling a batching machine. *Journal of scheduling*, 1(1):31–54.

Brucker, P. and Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing*, 45(4):369–375.

Bureau, M., Dauzère-Pérès, S., and Mati, Y. (2006). Scheduling challenges and approaches in semiconductor manufacturing. *IFAC Proceedings Volumes*, 39(3):739–744.

Caumond, A., Lacomme, P., and Tchernev, N. (2008). A memetic algorithm for the job-shop with time-lags. *Computers & Operations Research*, 35(7):2331–2356.

Chang, S.-C., Lee, L.-H., Pang, L.-S., Chen, T.-Y., Weng, Y.-C., Chiang, H.-D., and Dai, D.-H. (1995). Iterative capacity allocation and production flow estimation for scheduling semiconductor fabrication. In *Seventeenth IEEE/CPMT International Electronics Manufacturing Technology Symposium.'Manufacturing Technologies-Present and Future'*, pages 508–512. IEEE.

Chaudhry, I. A. and Khan, A. A. (2016). A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3):551–591.

Chuang, K.-F. and Lin, K.-C. (2003). Target generation system based on unlimited capacity allocation. US Patent 6,654,655.

Cigolini, R., Perona, M., Portioli, A., and Zambelli, T. (2002). A new dynamic look-ahead scheduling procedure for batching machines. *Journal of scheduling*, 5(2):185–204.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

Dauzère-Péres, S. and Lasserre, J.-B. (2012). *An integrated approach in production planning and scheduling*, volume 411. Springer Science & Business Media.

Dauzère-Pérès, S. and Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70:281–306.

Dauzere-Péres, S., Rouveyrol, J.-L., Yugma, C., and Vialletelle, P. (2010). A smart sampling algorithm to minimize risk dynamically. In *2010 IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 307–310. IEEE.

De Bontridder, K. (2005). Minimizing total weighted tardiness in a generalized job shop. *Journal of Scheduling*, 8(6):479–496.

Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE.

Dümmler, M. A. (1999). Using simulation and genetic algorithms to improve cluster tool performance. In et al. Evans, G., editors, *Proceedings of the 31st Conference on Winter Simulation: Simulation*, WSC '99, pages 875–879, New York, NY, USA. ACM.

Ehrgott, M. and Tenfelde-Podehl, D. (2003). Computation of ideal and nadir values and implications for their use in mcdm methods. *European Journal of Operational Research*, 151(1):119–139.

Eppe, S., López-Ibáñez, M., Stützle, T., and Smet, Y. D. (2011). An experimental study of preference model integration into multi-objective optimization heuristics. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 2751–2758.

Eppstein, D. (1992). Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55.

Essafi, I., Mati, Y., and Dauzère-Pérès, S. (2008). A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. *Computers & Operations Research*, 35(8):2599–2616.

Evans, G. W. (1984). An overview of techniques for solving multiobjective mathematical programs. *Management Science*, 30(11):1268–1282.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.

Fieldsend, J. E., Everson, R. M., and Singh, S. (2003). Using unconstrained elite archives for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323.

Fonseca, C. M. and Fleming, P. J. (1993). Multiobjective genetic algorithms. In *Genetic algorithms for control systems engineering, IEE colloquium on*, pages 6–1. IET.

Fonseca, C. M., Paquete, L., and Lopez-Ibanez, M. (2006). An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1157–1163.

Framinan, J. M., Leisten, R., and García, R. R. (2014a). *Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools*. Springer Science & Business Media.

Framinan, J. M., Leisten, R., and Ruiz García, R. (2014b). *Multi-Objective Scheduling*, pages 261–288. Springer London, London.

Gao, J., Gen, M., and Sun, L. (2006). Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. *Journal of Intelligent Manufacturing*, 17(4):493–507.

García-León, A., Dauzère-Pérès, S., and Mati, Y. (2015). Minimizing regular criteria in the flexible job-shop scheduling problem. In *7th Multidisciplinary International Conference on Scheduling: Theory & Applications, Prague, Czech Republic*, pages 443–456.

García-León, A. A., Dauzère-Pérès, S., and Mati, Y. (2019). An efficient pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria. *Computers & Operations Research*, 108:187 – 200.

Garey, M. R., Johnson, D. S., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129.

Geiger, C. D., Kempf, K. G., and Uzsoy, R. (1997). A tabu search approach to scheduling an automated wet etch station. *Journal of Manufacturing Systems*, 16(2):102 – 116.

Glover, F. (1989). Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206.

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99.

Gonçalves, J. F., de Magalhães Mendes, J. J., and Resende, M. G. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95.

González, M. A., Vela, C. R., González-Rodríguez, I., and Varela, R. (2013). Lateness minimization with tabu search for job shop scheduling problem with sequence dependent setup times. *Journal of Intelligent Manufacturing*, 24(4):741–754.

González, M. A., Oddi, A., Rasconi, R., and Varela, R. (2015). Scatter search with path relinking for the job shop with time lags and setup times. *Computers & Operations Research*, 60:37 – 54.

Govind, N., Bullock, E. W., He, L., Iyer, B., Krishna, M., and Lockwood, C. S. (2008). Operations management in automated semiconductor manufacturing with integrated targeting, near real-time scheduling, and dispatching. *IEEE Transactions on Semiconductor Manufacturing*, 21(3):363–370.

Grabowski, J., Nowicki, E., and Zdrzałka, S. (1986). A block approach for single-machine scheduling with release dates and due dates. *European Journal of Operational Research*, 26(2):278–285.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. (1977). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

Hall, N. G. and Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations research*, 44(3):510–525.

Ham, M. and Fowler, J. W. (2008). Scheduling of wet etch and furnace operations with next arrival control heuristic. *The International Journal of Advanced Manufacturing Technology*, 38(9-10):1006–1017.

Ham, M., Raiford, M., Dillard, F., Risner, W., Knisely, M., Harrington, J., Murtha, T., and Park, H. (2006). Dynamic wet-furnace dispatching/scheduling in wafer fab. In *The 17th Annual SEMI/IEEE ASMC 2006 Conference*, pages 144–147.

Hansen, P. and Mladenović, N. (1997). Variable neighborhood search for the p-median. *Location Science*, 5(4):207–226.

He, Z., Yang, T., and Tiger, A. (1996). An exchange heuristic imbedded with simulated annealing for due-dates job-shop scheduling. *European Journal of Operational Research*, 91(1):99–117.

Huang, K.-L. and Liao, C.-J. (2008). Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & operations research*, 35(4):1030–1046.

Hurink, J., Jurisch, B., and Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, 15(4):205–215.

Ishibuchi, H., Sakane, Y., Tsukamoto, N., and Nojima, Y. (2009). Evolutionary many-objective optimization by nsga-ii and moea/d with large populations. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 1758–1763. IEEE.

Jain, A. S. and Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European journal of operational research*, 113(2):390–434.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, 1(1):61–68.

Johnzén, C., Dauzère-Pérès, S., and Vialletelle, P. (2011). Flexibility measures for qualification management in wafer fabs. *Production Planning and Control*, 22(1):81–90.

Kao, Y.-T. and Chang, S.-C. (2018). Setting daily production targets with novel approximation of target tracking operations for semiconductor manufacturing. *Journal of manufacturing systems*, 49:107–120.

Kim, H., Lee, J., and Lee, T. (2012). Scheduling a wet station using a branch and bound algorithm. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2097–2102.

Kim, H., Lee, J., and Lee, T. (2014). Non-cyclic scheduling of a wet station. *IEEE Transactions on Automation Science and Engineering*, 11(4):1262–1274.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986.

Kis, T. (2003). Job-shop scheduling with processing alternatives. *European Journal of Operational Research*, 151(2):307–332.

Klemmt, A. and Mönch, L. (2012). Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*. IEEE.

Knopp, S. (2016). *Complex Job-Shop Scheduling with Batching in Semiconductor Manufacturing*. PhD thesis, Université de Lyon, Department of Manufacturing Sciences and Logistics, Gardanne, France.

Knopp, S., Dauzère-Pérès, S., and Yugma, C. (2014). Flexible job-shop scheduling with extended route flexibility for semiconductor manufacturing. In et al., A. T., editor, *Proceedings of the 2014 Winter Simulation Conference*, WSC '14, pages 2478–2489, Piscataway, NJ, USA. IEEE Press.

Knopp, S., Dauzère-Pérès, S., and Yugma, C. (2017). A batch-oblivious approach for complex job-shop scheduling problems. *European Journal of Operational Research*, 263(1):50 – 61.

Knowles, J. and Corne, D. (2004). Bounded pareto archiving: Theory and practice. In *Metaheuristics for multiobjective optimisation*, pages 39–64. Springer.

Kohn, R. and Rose, O. (2013). The impact of accuracy in lot arrival prediction on solution quality for the parallel batch machine scheduling problem in wafer fabrication. In et al., D. W., editor, *Simulation in Produktion und Logistik 2013*, pages 121–132, Paderborn. Heinz Nixdorf Institut.

Lacomme, P., Tchernev, N., and Huguet, M. (2012). Job-shop with generic time lags: a heuristic based approach. In *9th International Conference of Modeling, Optimization and Simulation-MOSIM*, volume 12, pages 06–08. Citeseer.

Lee, C.-Y. (1999). Two-machine flowshop scheduling with availability constraints. *European Journal of Operational Research*, 114(2):420–429.

Lee, C.-Y., Uzsoy, R., and Martin-Vega, L. A. (1992). Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, 40(4):764–775.

Lee, T.-E. (2008). A review of scheduling theory and methods for semiconductor manufacturing cluster tools. In *Proceedings of the 40th conference on winter simulation*, pages 2127–2135. Winter Simulation Conference.

Lee, T.-E., Lee, H.-Y., and Lee, S.-J. (2007a). Scheduling a wet station for wafer cleaning with multiple job flows and multiple wafer-handling robots. *International Journal of Production Research*, 45(3):487–507.

Lee, Y., Jiang, Z., Huai, Z., Ko, C., Zambri, M., Yi, D., and Kumar, A. (2008). A study of multiple objectives real-time dispatcher for wafer fabrication. In *2008 International Symposium on Semiconductor Manufacturing (ISSM)*, pages 163–166.

Lee, Y.-Y., Lin, C.-F., Hsu, P.-M., Lo, K.-W., Hsu, M.-F., and Hsieh, S.-L. (2007b). An integrated photolithography system framework for automatic manufacturing in mass production 300mm fab. In *2007 International Symposium on Semiconductor Manufacturing*, pages 1–4.

Liefooghe, A. (2009). *Métaheuristiques pour l'optimisation multiobjectif: Approches coopératives, prise en compte de l'incertitude et application en logistique*. PhD thesis, Université des Sciences et Technologie de Lille-Lille I.

Liu, Y., Zhu, N., Li, K., Li, M., Zheng, J., and Li, K. (2019). An angle dominance criterion for evolutionary many-objective optimization. *Information Sciences*.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer.

Lu, S. C. H., Ramaswamy, D., and Kumar, P. R. (1994). Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants. *IEEE Transactions on Semiconductor Manufacturing*, 7(3):374–388.

Ma, Y., Chu, C., and Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, 58(2):199–211.

Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395.

Mascis, A. and Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517.

Mason, S., Fowler, J., Carlyle, W., and Montgomery, D. (2005). Heuristics for minimizing total weighted tardiness in complex job shops. *International Journal of Production Research*, 43(10):1943–1963.

Mason, S. J. and Oey, K. (2003). Scheduling complex job shops using disjunctive graphs: a cycle elimination procedure. *International Journal of Production Research*, 41(5):981–994.

Mastrolilli, M. and Gambardella, L. M. (2000). Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, 3(1):3–20.

Mathirajan, M. and Sivakumar, A. (2006). A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *The International Journal of Advanced Manufacturing Technology*, 29(9-10):990–1001.

Mati, Y. (2010). Minimizing the makespan in the non-preemptive job-shop scheduling with limited machine availability. *Computers & Industrial Engineering*, 59(4):537–543.

Mati, Y., Dauzère-Pérès, S., and Lahlou, C. (2011). A general approach for optimizing regular criteria in the job-shop scheduling problem. *European Journal of Operational Research*, 212(1):33–42.

Michel, L. and Van Hentenryck, P. (2003). Maintaining longest paths incrementally. In *Principles and Practice of Constraint Programming–CP 2003*, pages 540–554. Springer.

Mönch, L. and Drießel, R. (2005). A distributed shifting bottleneck heuristic for complex job shops. *Computers & Industrial Engineering*, 49(3):363–380.

Mönch, L., Fowler, J., and Mason, S. (2012). *Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems*, volume 52 of *Operations Research Computer Science Interfaces Series*. Springer-Verlag, New York.

Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., and Rose, O. (2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, 14(6):583–599.

Mönch, L. and Rose, O. (2004). Shifting-Bottleneck-Heuristik für komplexe Produktionssysteme: Softwaretechnische Realisierung und Leistungsbewertung. *Quantitative Methoden in ERP und SCM, DSOR Beiträge zur Wirtschaftsinformatik*, 2:145–159.

Mönch, L., Schabacker, R., Pabst, D., and Fowler, J. W. (2007). Genetic algorithm-based subproblem solution procedures for a modified shifting bottleneck heuristic for complex job shops. *European Journal of Operational Research*, 177(3):2100–2118.

Moore, J. M. (1968). An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management science*, 15(1):102–109.

Mönch, L. and Roob, S. (2018). A matheuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective. *Applied Soft Computing*, 68:835 – 846.

Naderi, B., Ghomi, S. F., and Aminnayeri, M. (2010). A high performing metaheuristic for job shop scheduling with sequence-dependent setup times. *Applied Soft Computing*, 10(3):703 – 710.

Niedermayer, H. and Rose, O. (2004). Approximation of the cycle time of cluster tools in semiconductor manufacturing. In et al., R. K., editor, *Proceedings of the industrial engineering research conference*, pages 1–6, Georgia. IIE.

Novas, J. M. and Henning, G. P. (2012). A comprehensive constraint programming approach for the rolling horizon-based scheduling of automated wet-etch stations. *Computers & Chemical Engineering*, 42:189 – 205. European Symposium of Computer Aided Process Engineering - 21.

Nowicki, E. and Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management science*, 42(6):797–813.

Ovacik, I. M. and Uzsoy, R. (2012). *Decomposition methods for complex factory scheduling problems*. Springer Science & Business Media.

Pearce, D. J. and Kelly, P. H. (2007). A dynamic topological sort algorithm for directed acyclic graphs. *Journal of Experimental Algorithmics (JEA)*, 11:1–7.

Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing AG, Basel.

Pinedo, M. and Singer, M. (1999). A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Naval Research Logistics (NRL)*, 46(1):1–17.

Potts, C. N. and Kovalyov, M. Y. (2000). Scheduling with batching: a review. *European Journal of Operational Research*, 120(2):228–249.

Rachmawati, L. and Srinivasan, D. (2006). Preference incorporation in multi-objective evolutionary algorithms: A survey. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 962–968. IEEE.

Resendel, M. G. and Ribeiro, C. C. (2005). Grasp with path-relinking: Recent advances and applications. In *Metaheuristics: progress as real problem solvers*, pages 29–63. Springer.

Rodriguez-Verjan, G. L., Dauzère-Pérès, S., and Pinaton, J. (2015). Optimized allocation of defect inspection capacity with a dynamic sampling strategy. *Computers & Operations Research*, 53:319–327.

Rotondo, A., Young, P., and Geraghty, J. (2015a). Sequencing optimisation for makespan improvement at wet-etch tools. *Computers & Operations Research*, 53:261 – 274.

Rotondo, A., Young, P., and Geraghty, J. (2015b). Sequencing optimisation for makespan improvement at wet-etch tools. *Computers & Operations Research*, 53:261 – 274.

Rowshannahad, M., Dauzere-Peres, S., and Cassini, B. (2015). Capacitated qualification management in semiconductor manufacturing. *Omega*, 54:50–59.

Roy, B. and Sussmann, B. (1964). Les problèmes d'ordonnancement avec contraintes disjonctives. *Note ds*, 9.

Sadeghi, R. (2017). *Consistency of global and local scheduling decisions in semiconductor manufacturing*. PhD thesis, Université de Lyon.

Sadeghi, R., Dauzère-Pérès, S., Yugma, C., and Vermarien, L. (2015). Consistency between global and local scheduling decisions in semiconductor manufacturing: an application to time constraint management. In *International Symposium on Semiconductor Manufacturing Intelligence (ISMI)*, page 5.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1):1–15.

Schott, J. R. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH.

Shen, L. (2014). A tabu search algorithm for the job shop problem with sequence dependent setup times. *Computers & Industrial Engineering*, 78:95–106.

Shen, L., Dauzère-Pérès, S., and Neufeld, J. S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 265(2):503 – 516.

Sobeyko, O. and Mönch, L. (2016). Heuristic approaches for scheduling jobs in large-scale flexible job shops. *Computers & Operations Research*, 68:97 – 109.

Taillard, E. D. (1994). Parallel taboo search techniques for the job shop scheduling problem. *ORSA journal on Computing*, 6(2):108–117.

Tamssaouet, K., Dauzère-Pérès, S., and Yugma, C. (2018). Metaheuristics for the job-shop scheduling problem with machine availability constraints. *Computers & Industrial Engineering*, 125:1 – 8.

Tamssaouet, K., Dauzère-Pérès, S., and Yugma, C. (2018a). Minimizing makespan on parallel batch processing machines. In *16th International Conference on Project Management and Scheduling (PMS 2018)*, pages 229–232.

Tamssaouet, K., Dauzère-Pérès, S., Yugma, C., Knopp, S., and Pinaton, J. (2018b). A study on the integration of complex machines in complex job shop scheduling. In *2018 Winter Simulation Conference (WSC)*, pages 3561–3572.

T'kindt, V. and Billaut, J.-C. (2006). *Multicriteria scheduling: theory, models and algorithms*. Springer Science & Business Media.

Ulungu, E., Teghem, J., Fortemps, P., and Tuyttens, D. (1999). Mosa method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236.

Uzsoy, R., Lee, C.-Y., and Martin-Vega, L. A. (1994). A review of production planning and scheduling models in the semiconductor industry part ii: Shop-floor control. *IIE transactions*, 26(5):44–55.

Van Laarhoven, P. J., Aarts, E. H., and Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125.

Voudouris, C. and Tsang, E. (1999). Guided local search and its application to the traveling salesman problem. *European journal of operational research*, 113(2):469–499.

Wu, G.-L., Wei, K., Tsai, C.-Y., Chang, S.-C., Wang, N.-J., Tsai, R.-L., and Liu, H.-P. (1998). Tss: a daily production target setting system for fabs. In *1998 Semiconductor Manufacturing Technology Workshop (Cat. No. 98EX133)*, pages 86–98. IEEE.

Wu, K. (2014). A Unified View on Planning, Scheduling and Dispatching in Production Systems. *ArXiv e-prints*, page arXiv:1407.2709.

Yugma, C., Blue, J., Dauzère-Pérès, S., and Obeid, A. (2015). Integration of scheduling and advanced process control in semiconductor manufacturing: review and outlook. *Journal of Scheduling*, 18(2):195–205.

Yugma, C., Dauzère-Pérès, S., Artigues, C., Derreumaux, A., and Sibille, O. (2012). A batching and scheduling algorithm for the diffusion area in semiconductor manufacturing. *International Journal of Production Research*, 50(8):2118–2132.

Zhang, X. (2010). *Scheduling with Time Lags*. PhD thesis, Erasmus University Rotterdam.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Citeseer.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.

Zoghby, J., Barnes, J. W., and Hasenbein, J. J. (2005). Modeling the reentrant job shop scheduling problem with setups for metaheuristic searches. *European Journal of Operational Research*, 167(2):336–348.

Zribi, N., El Kamel, A., and Borne, P. (2008). Minimizing the makespan for the mpm job-shop with availability constraints. *International Journal of Production Economics*, 112(1):151–160.

École Nationale Supérieure des Mines de Saint-Étienne

NNT: 2019LYSEM016

Karim TAMSSAOUET

Multiobjective Complex Job-Shop Scheduling: Application to Semiconductor Manufacturing

Specialization: Industrial Engineering

Keywords:
Scheduling, Batching, Multiobjective Optimization, Metaheuristics, Semiconductor Manufacturing

Abstract:
This work deals with a real-life scheduling problem arising in semiconductor manufacturing where dispatching rules are still widely used. Optimization algorithms are a promising alternative to dispatching rules, provided that the solved problem encompasses the rich set of complex constraints and criteria. We consider a flexible job-shop scheduling problem with p-batching, reentrant flows, sequence-dependent setup times, unavailability periods, time lags and release dates. Different criteria must be considered to optimize the different operational performances: Overall throughput, target satisfaction, machine utilization and cycle time.

The proposed heuristic approach relies on the adaptation of the disjunctive graph that was introduced in a previous thesis, called *batch-oblivious*, where batching decisions are encoded in the arc weights. This graph is extended to allow the modeling of the internal resources of complex batching machines. An efficient algorithm is proposed to simultaneously compute start times and improve the solution during the graph traversal by filling underutilized batches. In addition to this integrated algorithm, the solution is improved within a simulated annealing metaheuristic. Depending on whether the preferences of the decision-maker are given before the search process, different approaches to handle the multiobjective aspect of the problem are studied and compared. The different components are embedded within a parallelized implementation of the GRASP metaheuristic. Different experiments on large size industrial instances show the significant improvement that can be brought by the proposed approach in computational times of several minutes.

École Nationale Supérieure des Mines de Saint-Étienne

NNT : 2019LYSEM016

Karim TAMSSAOUET

Ordonnancement multiobjectif d'ateliers complexes de type job-shop : application à la fabrication de semiconducteurs

Spécialité : Génie Industriel

Mots clefs :
Ordonnancement, Traitement par fournées, Optimisation Multiobjectif, Métaheuristiques, Fabrication de semi-conducteurs

Résumé :
Ce travail traite d'un problème d'ordonnancement complexe rencontré dans la fabrication de semi-conducteurs où l'utilisation de règles de priorité reste encore largement répandue. Les algorithmes d'optimisation constituent une alternative prometteuse à ces règles, à condition de prendre en compte le nombre important de contraintes complexes et de critères. Nous considérons un problème d'ordonnancement de type job-shop flexible avec "p-batching", des flux rentrants, des temps de préparation dépendant de la séquence, des périodes d'indisponibilité, des délai entre opération et des dates de début au plus tôt. Différents critères doivent être pris en compte pour optimiser les différentes performances opérationnelles: débit global, satisfaction des objectifs de production, utilisation des machines et temps de cycle.

L'approche proposée repose sur l'adaptation du graphe disjonctif proposée dans une thèse précédente, appelée "batch-oblivious", où les décisions de "batching" sont modélisées à travers les poids des arcs. Cette représentation a été étendue pour permettre la modélisation des ressources internes des machines complexes. Un algorithme efficace est proposé pour calculer les dates de début et, en même temps, améliorer la solution pendant le parcours du graphe. Une deuxième phase d'amélioration, plus classique, est assurée par une métaheuristique de type recuit simulé. Selon que les préférences du décideur sont ou non exprimées avant l'optimisation, différentes approches traitant l'aspect multiobjectif du problème sont étudiées et comparées. Les différents composants sont intégrés dans une métaheuristique de type GRASP. Différentes expérimentations sur des données industrielles de grande taille montrent l'amélioration significative que peut apporter l'approche dans des temps de calcul de quelques minutes.