Multiobjective optimization for complex flexible job-shop scheduling problems

Karim Tamssaouet, Stéphane Dauzère-Pérès, Sebastian Knopp, Abdoul Bitar, Claude Yugma

▶ **To cite this version:**

Karim Tamssaouet, Stéphane Dauzère-Pérès, Sebastian Knopp, Abdoul Bitar, Claude Yugma. Multiobjective optimization for complex flexible job-shop scheduling problems. European Journal of Operational Research, 2022, 296 (1), pp.87-100. 10.1016/j.ejor.2021.03.069 . emse-03541839

HAL Id: emse-03541839

https://hal-emse.ccsd.cnrs.fr/emse-03541839

Submitted on 16 Oct 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Version of Record: https://www.sciencedirect.com/science/article/pii/S0377221721003751
Manuscript_d75726a6522d88cf52be017c1031e298

Multiobjective Optimization for Complex Flexible Job-Shop Scheduling Problems

Karim Tamssaouet[1,2*], Stéphane Dauzère-Pérès[1,2], Sebastian Knopp[3], Abdel Bitar[4], Claude Yugma[2],
karim.tamssaouet@bi.no, dauzere-peres@emse.fr, mail@sebastian-knopp.de, abdlbitar@gmail.com, yugma@emse.fr

[1]*Department of Accounting, Auditing and Business Analytics*
BI Norwegian Business School, Oslo, Norway

[2]*Mines Saint-Etienne, Univ Clermont Auvergne*
CNRS, UMR 6158 LIMOS
CMP, Department of Manufacturing Sciences and Logistics
Gardanne, F-13541, France

[3]*Untis GmbH*
Stockerau, 2000, Austria

[4]*Smart Flows SAPA*
Paris, 75003, France

[*]Corresponding author at BI Norwegian Business School, Department of Accounting, Auditing and Business Analytics, Nydalsveien 37, N-0484 Oslo
Email: karim.tamssaouet@bi.no, Tel.: +47 41 23 17 61.

© 2021 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

Abstract

In this paper, we are concerned with the resolution of a multiobjective complex job-shop scheduling problem stemming from semiconductor manufacturing. To produce feasible and industrially meaningful schedules, this paper extends the recently proposed batch-oblivious approach by considering unavailability periods and minimum time lags and by simultaneously optimizing multiple criteria that are relevant in the industrial context. A novel criterion on the satisfaction of production targets decided at a higher level is also proposed. Because the solution approach must be embedded in a real-time application, decision makers must express their preferences before the optimization phase. In addition, a preference model is introduced where trade-off is only allowed between some criteria. Two a priori multiobjective extensions of Simulated Annealing are proposed, which differ in how the simultaneous use of a lexicographic order and weights is handled when evaluating the fitness. A known a posteriori approach of the literature is used as a benchmark. All the metaheuristics are embedded in a Greedy Randomized Adaptive Search Procedure. The different versions of the archived GRASP approach are compared using large industrial instances. The numerical results show that the proposed approach provides good solutions regarding the preferences. Finally, the comparison of the optimized schedules with the actual factory schedules shows the significant improvements that our approach can bring.

Keywords: Scheduling, Multiobjective optimization, Flexible job-shop scheduling with batching, Metaheuristics, OR in semiconductor manufacturing

1. Introduction

Integrated circuits present in everyday electrical devices are produced by semiconductor manufacturing companies from thin slices of semiconductor material called *wafers*. The most technologically complex and capital intensive phase in this industry is wafer fabrication. This phase takes place in fabrication facilities called wafer *fabs*, where a single lot of wafers may go through up to 1,000 steps in different work areas. To create all the layers, each lot visits each work area many times, leading to reentrant flows. In high-mix fabs, where hundreds of products are produced at the same time, production resources must be shared among lots of different products and among lots that are at different stages. In this context, scheduling the production operations on the scarce resources is complex and has a critical impact on the operational performance of factories, such as cycle time and throughput. As scheduling requires a detailed modeling of the production flows and the production resources, a global scheduling model of the factory is hard to manage, and its optimization is computationally intractable. The work areas are thus scheduled locally while making sure that a mechanism ensures some coordination between the local schedules.

In this work, the proposed approach is applied to the cleaning and diffusion area in semiconductor manufacturing facilities. The scheduling problem in this area can be modeled as a Flexible Job-Shop scheduling problem with p-batching, unavailability periods, reentrant flows, sequence-dependent setup times, minimum time lags and release dates. Because of the large number of constraints considered by our approach, it can be applied to other workshops such as ion implantation (Horng et al. (2000)) and other industrial contexts. While satisfying the various constraints, different criteria that are related to cycle time, throughput and resource utilization must be optimized. In addition to the local performance of the scheduled area, schedules should contribute to the realization of global objectives at the factory level. The resulting problem to solve is NP-hard as it generalizes the classical job-shop scheduling problem. Since our goal to solve an NP-hard scheduling problem with large instances with on average 1,500 operations of 500 jobs on 70 batching machines, a heuristic approach is developed in this paper.

This work aims primarily to design a multiobjective optimization approach that solves the complex job-shop scheduling problem. Even if there are already several multiobjective approaches in the literature, two characteristics of the studied context require the design of new multiobjective approaches. First, the approach to be developed must be embedded in a real-time application where the time available for decision making is limited, which forces decision makers to express their preferences before the optimization phase. Second, in our problem, decision makers allow some criteria to compensate each other but not others. Therefore, the preferences of decision makers are expressed using both a lexicographic order and weights. To our knowledge, this generic modeling of preferences has not been considered in the multiobjective optimization literature. Our modeling is instead similar to the modeling used in lexicographic goal programming, e.g., Romero (1991) and Jones et al. (2010). Therefore, the paper aims to explore different ways of taking preferences into account in a multiobjective optimization approach.

The paper is divided into four sections. Section 2 reviews the literature related to the different aspects of our problem: Constraints, criteria and solution approaches. This section also motivates the different choices made in this work related to modeling and solution approaches. The industrial scheduling problem and its related inputs are formally modeled in Section 3. The large set of constraints are formalized in Section 3.1. Instead of the classical criteria found in the scheduling literature (Mati et al. (2011), Pinedo (2016) and García-León et al. (2019)), the criteria to optimize are adapted to the rolling horizon framework found in many manufacturing contexts. In addition to the criteria that translate the performance of a local area in the factory, a new flexible criterion is proposed to measure the contribution of a local schedule to the realization of production targets decided at the factory level. Section 3.2 motivates

and formally defines the relevant criteria. The description of the problem ends with the formal modeling of the preferences in Section 3.3. The various building blocks of the explored extensions of Simulated Annealing are introduced in Section 4. First, to construct and evaluate feasible schedules for our complex job-shop scheduling problem, the approach proposed in Knopp et al. (2017) is extended in Section 4.1 to include machine availability and minimum time lag constraints. Section 4.2 describes the neighborhood function, and Section 4.3 defines the concept of *archive* and highlights the information that can be extracted and used during the search. After describing the building blocks that are common to the Simulated Annealing approaches, Section 4.4 presents how each approach evaluates the fitness of solutions. Two variants of an a priori Simulated Annealing metaheuristic are proposed in this work, depending on how the preferences are used when computing the acceptance probability. AMOSA of Bandyopadhyay et al. (2008), presented in Section 4.4.3, is our reference of a posteriori Simulated Annealing based approach. To diversify the search and to benefit from the parallelism of modern CPUs, these various approaches are separately applied within a parallelized Greedy Randomized Adaptive Search Procedure (GRASP) approach, described in Section 5. Numerical results are presented in Section 6, where the three versions of Simulated Annealing are compared according to the quality of the final solutions. This section ends with an evaluation of the improvement that our approach can bring by comparing the proposed schedules to the actual schedules of a factory of an industrial partner. A final summary of the work and some perspectives are provided in Section 7.

2. Related Works and Motivations

2.1. Complex Job-Shop Scheduling Problem

A tremendous amount of research on scheduling has been conducted in the last decades (Pinedo (2016)). As a basis of the tackled problem in this work, the classical job-shop scheduling problem is a hard problem for which exact and approximation approaches are proposed in the literature (Błażewicz et al. (1996), Jain and Meeran (1999)). Successful solutions methods are often based on the disjunctive graph representation, introduced by Roy and Sussmann (1964), that models dependencies between operations in a concise way. To represent real scheduling problems, the job-shop scheduling problem is enriched by considering additional constraints. For example, when considering that an operation may be processed on more than one machine, this results in the flexible job-shop scheduling problem, another well-studied problem in the literature (Chaudhry and Khan (2016)). Due to its complexity, semiconductor manufacturing is an application field where the job-shop scheduling problem must be extended to include additional constraints. An overview of scheduling challenges in semiconductor manufacturing is provided in Mönch et al. (2011).

The problem tackled in this work is qualified as a complex flexible job-shop scheduling problem because of the additional constraints that must be considered in a job-shop environment. The presence of batching machines, capable of processing several jobs at the same time, is the main feature that characterizes the studied industrial problem. Surveys related to batching in general and to batching for semiconductor manufacturing can be found in Potts and Kovalyov (2000) and Mathirajan and Sivakumar (2006), respectively. As observed in these surveys, scheduling on a single or parallel batching machines received most of the research attention. Most existing solution approaches for complex job-shop scheduling problems with batching machines rely on the disjunctive graph representation (e.g., Mönch and Rose (2004), Mason et al. (2005), Yugma et al. (2012)). The modified disjunctive graph representation of Ovacik and Uzsoy (2012) introduces dedicated nodes to represent batching decisions explicitly. This paper uses the approach recently proposed by Knopp et al. (2017) that relies on a novel graph modeling approach, called batch-oblivious. Instead of inserting additional nodes and arcs, this modeling encodes

batching decisions in the arc weights. More details about this approach are given in Section 4.1. One of the advantages of this approach is its capability of considering additional constraints. In addition to the constraints considered in Knopp et al. (2017), the studied problem requires the integration of two additional hard constraints: Unavailability periods and minimum time lags. As a generalization of the classical job-shop scheduling problem, this problem is NP-hard. Since our goal is to solve an NP-hard scheduling problem with large instances of on average 1,500 operations of 500 jobs on more than 70 batching machines, a heuristic approach is developed in this paper.

2.2. Multiobjective Scheduling Problem

Many real-world scheduling problems are multiobjective in nature, i.e., multiple performance measures should be optimized simultaneously. Several criteria are commonly used in the scheduling literature. The makespan is considered to optimize the throughput of resources, while the total weighted completion time or the total weighted flow time focus on the cycle times of products. Criteria based on due dates, such as the total tardiness or the number of tardy jobs, aim to meet the customers' delay requirements. A general overview of multiobjective optimization for scheduling problems is given in T'kindt and Billaut (2006). Surveys covering the solution approaches for the multiobjective flexible job-shop scheduling problem, which is the closest one to our problem, can be found in Genova et al. (2015), Chaudhry and Khan (2016) and Amjad et al. (2018).

Instead of the conventional criteria in the scheduling literature, such as the makespan or the total weighted completion time, criteria relevant to the industrial context must be optimized. The approach must be embedded in an automated scheduling system that runs in a highly dynamic environment. To cope with this environment, schedules are optimized in a rolling horizon, typically every 30 minutes on a scheduling horizon of a few hours. When optimizing a schedule with a fixed time horizon, criteria such as the makespan criterion are not relevant. The criteria, previously defined in Artigues et al. (2006), Yugma et al. (2012) and Bitar et al. (2016), that optimize the performance in our problem are recalled in Section 3.2.

Recall that optimizing the schedule of an entire semiconductor manufacturing facility is computationally intractable. However, it is important to ensure that locally optimized schedules contribute to the achievement of global objectives at the factory level. Therefore, a critical feature required by the industrial environment is that solution approaches ensure consistency between local schedules and global objectives. In 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 solutions, different operational objectives motivate production targets such as ensuring the "linearity" of the production line, reducing the inventory and cycle times, and maximizing tool utilization. Few works discussing the determination of production targets can be found in the literature (e.g., Chang et al. (1995), Wu et al. (1998)). Sadeghi et al. (2015) proposes a general framework that aims at supporting and controlling local decisions by considering global objectives and information. The general idea is to determine production targets for local scheduling to achieve. Kao and Chang (2018) proposes an approximation approach for computing production targets while taking into consideration the induced variation on the wafer flows. While these papers deal with the determination of production targets, to the best of our knowledge, the objective of following production targets is not studied within the scheduling literature. This objective must be considered when it is important to make sure that the produced schedules follow a production plan determined at a higher level. To include production targets, a new criterion is proposed in Section 3.2 to measure the satisfaction of the targets while allowing the decision maker to balance between the overall satisfaction and a fair satisfaction of all the targets.

As it is rare to find a solution that simultaneously optimizes all the criteria, multiobjective optimization provides instead of a set of solutions. Since only a single schedule can be implemented in the shop-floor, the decision maker must express his/her preferences at some stage and, in many industrial contexts, without knowing the shape of the trade-off curve of the problem instance to solve. As stated above, a schedule must be computed every 30 minutes or less for the next few hours. Due to the high frequency of the schedule computation and the high level of automation in the semiconductor manufacturing industry, it is most often not realistic to let decision makers choose one schedule in the set of good schedules for the different criteria. One way or another, decision makers must provide their preferences, and one must consider the fact that these preferences may evolve over time depending on the situation (factory fully loaded, crisis situation, ...). Therefore, the approach to be proposed for the industrial application must be fairly generic, and allow decision makers to provide preferences suited to the current situation in the shop floor.

Various preference models have been reported in the literature (Wang et al. (2017)). In the studied context, as the trade-off is allowed between some criteria and forbidden between others, a lexicographic order and weights are simultaneously used to model the preferences, which is not common in the multiobjective optimization literature. In the industrial application, there are more features than the ones presented in this work. For example, industrial characteristics require that some lots, qualified as mandatory, be completed within the scheduling horizon. Describing all of these features and how they are included in our approach would increase the size of this already long paper. However, ultimately, such constraints are considered as soft and criteria that model their violation are introduced. The lexicographic order is then introduced to forbid the trade-off between such criteria, along with the criterion modeling the satisfaction of production targets, with other criteria modeling the local performance of the work area. Weights are then used to express the preferences of the decision maker between these last criteria, as the lexicographic order is too extreme. In addition, the lexicographic order differentiates the criteria which correspond to important operational performance measures from those which model nice-to-have solution structures.

2.3. Multiobjective Optimization Approaches

In this section, we briefly review the different approaches proposed in the literature for solving multiobjective optimization problems. We focus on heuristic approaches and we motivate the approach adopted in this work. In recent years, multiobjective metaheuristics have received considerable attention. Some of these approaches are adaptations of local search methods such as Simulated Annealing, Tabu Search and iterated greedy algorithms. Evolutionary algorithms such as Genetic Algorithms are another class of approaches that are successfully applied to a wide range of applications. Despite this diversity of choices, the complexity of our problem makes Simulated Annealing the most suitable choice as motivated below.

When studying the literature dealing with different variants of the job-shop scheduling problem, it appears that Tabu Search has been the main choice (Dauzère-Pérès and Paulli (1997), Mastrolilli and Gambardella (2000), Nowicki and Smutnicki (2005), Mati et al. (2011), García-León et al. (2019)). However, the success of Tabu Search is conditioned by the efficiency of the neighborhood function, which is even more critical for large problem instances such as the ones we want to solve. Without an efficient neighborhood function, it is necessary to evaluate, on average, $30,000$ neighbors for each solution in our industrial instances. In five minutes, Tabu Search (that we implemented for testing purposes) performs less than 20 moves, while successful variants of Tabu Search for the job-shop scheduling problem are allowed to perform hundreds of thousands of moves. Extending the neighborhood functions proposed in the literature for the flexible job-shop scheduling problem is interesting. However, these neighborhood

functions only take a single criterion into account and strongly depend on the optimized criterion. Only recently, some studies have proposed approaches that can deal with any regular criterion (e.g., Mati et al. (2011)) and in the context of multiobjective optimization (e.g., García-León et al. (2019)). As the criteria to be optimized in our context are different from those classically treated in the literature, the design of efficient neighborhood functions for our problem is very challenging and requires additional research.

The iterated greedy algorithm has shown state-of-the-art performance for the permutation flow-shop scheduling problem (e.g.,Ruiz and Sttzle (2008), Ruiz and Sttzle (2008), Ciavotta et al. (2013)). However, its application to the classical job-shop scheduling problem is not straightforward, and only recent papers in the literature are exploring this line of research (e.g., Pranzo and Pacciarelli (2016)). Regarding our problem, applying an iterated greedy algorithm seems very difficult if we consider the fact that its success depends largely on a good construction heuristic which is not easy to design, even for the flexible job-shop scheduling problem. It can be argued that it is not worth the effort to develop such heuristics to solve an industrial problem with some complex criteria. Also, in the references given above, local search is used within the iterated greedy algorithms to improve the incumbent solution. As for Tabu Search, the large neighborhood of the solutions makes its use unrealistic. Because they have often been successfully applied, evolutionary algorithms such as genetic algorithms can also be seen as an interesting choice to solve our problem. However, the solution representation is not trivial to define for our complex flexible job-shop scheduling problem and, more importantly, classical crossover and mutation operators would lead to many infeasible solutions and also of poor quality. In addition, successful evolutionary approaches are often hybridized with local search procedures, and thus the arguments given above still apply.

The approaches described above could be interesting perspectives and are presented as such in the last section of the paper. However, further research is required to apply them to our problem. In light of the considerations above, Simulated Annealing seems the most appropriate approach, especially when considering its effectiveness, as reported in the literature (Van Laarhoven et al. (1992), Loukil et al. (2007), Knopp et al. (2017)). When considering preferences, most Simulated Annealing approaches in the literature assume that weights are assigned to criteria and use a weighted sum to transform the multiobjective optimization problem into a single objective problem. Conversely, few papers (e.g., Aggelogiannaki and Sarimveis (2007)) have adapted Simulated Annealing to the context where the criteria are prioritized. Therefore, to our knowledge, there is a need to adapt Simulated Annealing when it is targeted to integrate preferences within the search, and when a lexicographic order and weights must be used at the same time to reflect preferences correctly.

In the scheduling literature, several extensions of Simulated Annealing are proposed. Serafini (1994) present a first multiobjective Simulated Annealing approach, where several ways to calculate the acceptance probabilities are explored. Several other multiobjective Simulated Annealing techniques are proposed, which can be seen as an evolution of Serafinis early techniques, rather than the development of radically different approaches (e.g., Czyżżak and Jaszkiewicz (1998) and Ulungu et al. (1999)). In the following, we first review some of the approaches that were successfully applied to scheduling problems and motivate why we have not adopted them.

Simulated Annealing approaches are proposed in Lin and Ying (2013) and Varadharajan and Rajendran (2005) to solve bi-objective permutation flow-shop scheduling problems. Extending these approaches is not possible since finding a unique permutation of jobs for all machines, with all permutations being feasible, is much simpler than solving our complex flexible job-shop scheduling problem with batching and other characteristics. For instance, in the approach of Varadharajan and Rajendran (2005), before using Simulated Annealing, each initial solution is sequentially improved using heuristics that are only suitable for the permutation flow-shop scheduling problem. Also, both approaches are

specifically designed for bi-objective optimization problems, whereas our problem by definition needs to include more than two objectives. For instance, applying the approach in Lin and Ying (2013) to a multiobjective problem is not straightforward, as different cases are identified to study the acceptance of a new solution. Not only the number of these cases will increase, but also a new mechanism must be designed to compute the acceptance probability.

In Loukil et al. (2007), a flexible job-shop scheduling problem with specific constraints is solved using the Multiobjective Simulated Annealing of Ulungu et al. (1999), which is called UMOSA in the literature. This method works with a predefined set of diversified weight vectors. By using an aggregation function, Simulated Annealing is run for each weight vector in order to obtain a good approximation of the Pareto front. It is an interesting approach, but one that cannot be used to solve our problem because of the cardinal of the set of vectors which would be too large. For example, when four criteria are optimized, this cardinal must be at least 20, which means that 20 runs of Simulated Annealing must be performed. This is impractical in an industrial application where only a few minutes are allowed to return a solution of good quality. Instead of these approaches, we use AMOSA of Bandyopadhyay et al. (2008) as a reference, since it is a known adaptation of Simulated Annealing to multiobjective optimization that has been used in a wide range of applications. Moreover, to determine the acceptance probability of a new solution, this approach takes into account the domination status of the new solution with the current solution, as well as those in the archive.

Outside the multiobjective optimization context, the adopted preference modeling in this work is similar to the one defined in lexicographic goal programming. Despite the similarities in the preference modeling, preemptive goal programming does not apply to our problem. First, due to the use of the approach within an automated environment, it is not realistic to imagine having a decision maker setting the achievement levels for each criterion each time the approach is called to solve the scheduling problem in the shop floor. Even if this is possible, applying such an approach will require the decision maker to provide aspiration levels, in addition to the lexicographic ordering and the weights. This leads to more complexity for the decision maker and the approach designers. Even if we assume that the decision maker can provide all the necessary information, there are not so many suitable heuristic approaches that can solve the corresponding preemptive goal programming problem. To our knowledge, the Simulated Annealing of Baykasoğlu (2005) is the only attempt made to adapt Simulated Annealing to solve preemptive goal programming problems. As shown in Section 3, one of the approaches proposed in this work uses the adaptations proposed in Baykasoğlu (2005).

3. Problem Description

This section is devoted to the description of the scheduling problem encountered in semiconductor manufacturing. In terms of constraints, the problem is described in Section 3.1 as a flexible job-shop scheduling problem with p-batching, reentrant flows, sequence-dependent setup times, release times, availability constraints and minimum time lags (*complex job-shop scheduling problem*). Instead of classical criteria in the scheduling literature, criteria that are more suitable for industrial contexts and adapted to a rolling horizon framework are defined in Section 3.2.

3.1. Description of the Constraints

Let us consider a set of *jobs* $\mathscr{J}$ to be processed on a set of *machines* $\mathscr{M}$. Each job $j \in \mathscr{J}$ requires a sequence of *operations* $O_j = \{o_{1,j}, \ldots, o_{i,j}, \ldots, o_{|O_j|,j}\}$, a *release time* $r_j \in \mathbb{Z}$, a *size* $\sigma_j \in \mathbb{N}$ and a *prioriy* $\omega_j \in \mathbb{R}_{>0}$. The disjoint union $\mathscr{O} = O_1 \dot{\cup} O_2 \ldots \dot{\cup} O_{|\mathscr{J}|}$ denotes the set of all operations. Batching, an important feature of our problem, is the capability of machines to process at the same time several

jobs up to a given capacity, i.e., the jobs in a batch have the same start times and completion times. To model batching constraints, the notion of *recipe* is used. For a given set of *recipes* $\mathscr{R}$, each recipe $q \in \mathscr{R}$ prescribes a machine $m_q \in \mathscr{M}$, a *processing time* $p_q \in \mathbb{N}_0$ and a *batching capacity* $b_q \in \mathbb{N}_{>0}$ for machine m_q. Each operation $o_{i,j} \in \mathscr{O}$ is associated with a subset of recipes $R_{i,j} \subset \mathscr{R}$. A given mapping $s : \mathscr{R} \times \mathscr{R} \to \mathbb{N}$ prescribes *sequence-dependent setup times* between operations that are scheduled on the same machine. If q and q' are two recipes in $\mathscr{R}$ on the same machine, i.e. $m_q = m_{q'}$, then $s(q,q')$ is the setup time that is required when an operation with recipe q' is scheduled just after an operation with recipe q on the machine.

The constraints in Knopp et al. (2017) are extended by considering unavailability periods and minimum time lags. Unavailability periods model periods during which a machine is unavailable to process operations like those resulting from preventive or curative maintenance operations. Formally, we extend the problem as follows. For each machine $m \in \mathscr{M}$, let us consider a set of fixed unavailability periods $U_m = \{u_{1,m}, \ldots, u_{l,m}, \ldots, o_{|U_m|,m}\}$. Each unavailability period $u_{l,m} \in U_m$ has a fixed start date $S_{l,m}$ and a fixed end date $C_{l,m}$. Regarding the possibility for an operation to be interrupted by an unavailability period, the non-preemptive case is considered (Aggoune (2004)), i.e., an operation cannot be interrupted by another operation or by an unavailability period. The second extension of the problem results from the inclusion of minimum 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 $\mathscr{L}^{min} \subset \mathscr{J} \times \mathbb{N} \times \mathscr{R} \times \mathbb{N} \times \mathscr{R} \times \mathbb{N}$. The components of a time lag $\lambda = (j,i,q_{i,j},i',q_{i',j},d) \in \mathscr{L}^{min}$ have the following meaning: $j \in J$ identifies the job; i and $i' \in \mathbb{N}$ with $1 \leq i < i' \leq |O_j|$ identify operations $o_{i,j}$ and $o_{i',j} \in O_j$; $q_{i,j}$ and $q_{i',j} \in \mathscr{R}$ are the selected recipes for operations $o_{i,j}$ and $o_{i',j}$; $d \in \mathbb{N}_{>0}$ identifies the minimum time lag between the start time of $o_{i,j}$ and the start time of $o_{i',j}$. Time lags are used to model two features of the scheduling problem: 1) The transportation time to transfer a job from one machine to another one, and 2) Minimum delays imposed for process considerations, in which case the time is independent of the assigned machines.

A *schedule* for this problem 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 $o_{i,j} \in \mathscr{O}$.Recall that a recipe prescribes a a machine, a processing time and a batching capacity. Therefore, let us denote the machine, the processing time and the batching capacity of each operation $o_{i,j}$ in the schedule as $m_{i,j}$, $p_{i,j}$ and $b_{i,j}$, respectively. Also, let $\mathscr{B}$ denote the set of all batches formed in a schedule. As only operations with the same recipe can be processed in the same batch, q_B denotes the associated recipe to batch $B \in \mathscr{B}$, 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 $C_{i,j} = S_{i,j} + p_{i,j}$. Also, for each machine $m \in \mathscr{M}$, for each operation $o_{i,j} \in \mathscr{O}$ such that $m_{i,j} = m$ and for each unavailability period $u_{l,m} \in U_m$, $S_{i,j} + p_{i,j} \leq S_{l,m}$ or $C_{l,m} \leq S_{i,j}$ must hold. 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 $o_{i,j} \in \mathscr{O}$ with $i < |O_j|$. Regarding minimum time lag constraints, for each minimum time lag $\lambda = (j,i,q_{i,j},i',q_{i',j},d) \in \mathscr{L}^{min}$, $S_{i',j} \geq S_{i,j} + d$ must hold. 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 $j \in \mathscr{J}$. Operations performed on the same machine must not overlap. Hence, for two operations $o_{i,j}$ and $o_{i',j'} \in \mathscr{O}$ with $m_{i,j} = m_{i',j'}$, then $S_{i,j} = S_{i',j'}$, $S_{i,j} \geq C_{i',j'}$ or $C_{i,j} \leq S_{i',j'}$ must hold. Regarding the batching constraints, only operations with the same recipe can be processed together. So, for two operations $o_{i,j}$ and $o_{i',j'} \in \mathscr{O}$ with $q_{i,j} \neq q_{i',j'}$ and $m_{i,j} = m_{i',j'}$, either $S_{i,j} \geq C_{i',j'}$ or $C_{i,j} \leq S_{i',j'}$ must hold.

Any batch $B \in \mathscr{B}$ with a recipe q must respect the batching capacity of the machine to which it is assigned. Thus, $\sigma_B \leq b_q \; \forall B \in \mathscr{B}$ is required. To respect sequence-dependent setup times, for all operations $o_{i,j}$ and $o_{i',j'} \in \mathscr{O}$ with $m_{i,j} = m_{i',j'}$ and not in the same batch (i.e., $S_{i,j} \neq S_{i',j'}$), either $C_{i,j} + s(q_{i,j}, q_{i',j'}) \leq S_{i',j'}$ or $C_{i',j'} + s(q_{i',j'}, q_{i,j}) \leq S_{i,j}$ must hold.

3.2. Formal Modeling of the Criteria

While constructing feasible schedules that satisfy all the constraints detailed in the Section 3.1, our goal is to optimize several criteria simultaneously. Within the context of semiconductor manufacturing, the used criteria for scheduling problems are derived from performance measures of the entire factory. The most important measures are cycle time, throughput and on-time delivery (Mönch et al. (2011)). First, let us consider the criteria considered in this work that were already defined in previous works. The *weighted flow factor (WFF)*, also called *average X-Factor*, is used in Artigues et al. (2006), Yugma et al. (2012) and Bitar et al. (2016). This criterion is designed to reduce the cycle time and the work-in-process in the considered optimization scope. Let us consider ε_j a minimum possible time to process all operations of job $j \in \mathscr{J}$. From the schedule, the actual cycle time of each job $j \in \mathscr{J}$ is computed as the difference between the completion time of its last operation denoted by $C_j = C_{|O_j|,j}$ and its release date r_j. The flow factor of a job is its actual cycle time divided by its theoretical cycle time. Now, the weighted flow factor (WFF) to minimize is the weighted average of all flow factors, as shown in (1).

$$\text{WFF} = \frac{1}{\sum_{j \in \mathscr{J}} \omega_j} \sum_{j \in \mathscr{J}} \frac{\omega_j (C_j - r_j)}{\varepsilon_j} \tag{1}$$

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 length of the scheduling horizon. When considering a scheduling horizon, an operation $o_{i,j} \in \mathscr{O}$ in a given schedule may be 1) Completed before the end of the horizon, i.e., $C_{i,j} \leq H$, 2) Started after the end of the horizon , i.e., $S_{i,j} \geq H$ or 3) Started before the end of the horizon and completed after the end of the horizon, i.e., $S_{i,j} < H \wedge C_{i,j} > H$. To consider these cases, a *completion rate* $\theta_{i,j}$ is defined in (2) for each operation $o_{i,j} \in R$. Assuming that σ_j denotes the size of job $j \in J$ in number of wafers, the *weighted number of moves (WNM)* can be computed as in (3).

$$\theta_{i,j} = \begin{cases} \frac{min(p_{i,j}, H - S_{i,j})}{p_{i,j}} & \text{if } S_{i,j} \leq H, \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\text{WNM} = \sum_{o_{i,j} \in R} \omega_j \sigma_j \theta_{i,j} \tag{3}$$

A third criterion to optimize is the *batching coefficient (BC)*, which 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 $\mathscr{B}^H$ denote the set of all batches in $\mathscr{B}$ for which the start times occur before the end of the scheduling horizon, i.e. $\mathscr{B}^H = \{B \in \mathscr{B} \mid S_{i,j} < H, \forall o_{i,j} \in B\}$. Then, the batching coefficient (BC) of a schedule can be defined as in (4). Contrary to the other criteria considered in this paper, BC is a non-regular criterion, i.e., it can be degraded by advancing the start times of operations.

$$\text{BC} = \frac{\sum_{B \in \mathscr{B}^H} \sigma_B}{\sum_{B \in \mathscr{B}^H} b_B} \tag{4}$$

In addition to the previous criteria introduced in previous papers, a new criterion is proposed. In semiconductor manufacturing, production targets, also called daily move targets, are extensively used as a way to set a bridge between shop-floor control and the master production plan. Their goal is to smooth the differences between the work-in-process level of a production stage and its fixed production target. The satisfaction of production targets allows local scheduling decisions to be consistent with global objectives at the wafer fabrication level. The criterion proposed below is relevant in any situation where optimized schedules must follow a production plan determined at a higher level.

Formally, let us consider a given set $\mathscr{T}$ of production targets, where the set $T_{i,j}$ denotes the set of production targets to which an operation $o_{i,j} \in \mathscr{O}$ contributes. Note that it is possible to have $T_{i,j} = \emptyset$, which means that operation $o_{i,j}$ does not contribute to any production target. It is also possible that $|T_{i,j}| > 1$, which means that operation $o_{i,j}$ contributes to more than one production target. To each production target $\tau \in \mathscr{T}$, let us associate a requested volume $D_\tau \in \mathbf{N}$ and a weight w_τ. We assume, without loss of generality, that $\sum_{\tau \in \mathscr{T}} w_\tau = 1$. 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 $\tau \in \mathscr{T}$.

Given a feasible schedule, $R_\tau = \frac{P_\tau}{D_\tau}$ denotes the completion rate of a production target $\tau \in \mathscr{T}$. 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 R_τ, the expected satisfaction level L_τ of the decision maker that depends on the completion rate R_τ can be modeled. In the industrial application, as targets only define minimal quantities to produce, the satisfaction level can be computed as $L_\tau = \min(R_\tau, 1)$.

After defining the satisfaction level of a single production target, it is still necessary to model the overall satisfaction of multiple production targets in a schedule. It should be noted that such a problem can be formulated as a multiobjective optimization problem by considering the satisfaction of each target as an independent criterion. However, because there are on average 25 production targets in the industrial instances, an aggregation function is preferable. 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. The proposed criterion is called *Target Satisfaction Indicator (TSI)* and is computed using (5). When requiring, without loss of generality, that $\sum_{j \in \mathscr{J}} \omega_j = 1$, this indicator has the form of what is called *weighted power mean*. This aggregation function has several properties that make it practically attractive. As the weighted power mean always produces values that lie between the smallest and the largest of the satisfaction levels, and as these levels are in the interval $[0,1]$, this indicator produces values that can be easily interpreted by decision makers. More importantly, this criterion is flexible as it allows the decision maker to prioritize, depending on the context, overall satisfaction or balancing, without ignoring the other.

$$\mathrm{TSI} = \left(\sum_{\tau \in \mathscr{T}} w_\tau L_\tau^\alpha \right)^{1/\alpha} \tag{5}$$

The proposed criterion meets this requirement through the flexibility given by parameter α. When $\alpha = 1$, the criterion takes the form of a weighted arithmetic mean. Fully compensatory, this particular form can be used to express the idea of overall satisfaction while disregarding the balancing between satisfaction levels. When it is relevant to only focus on the balancing, by using a very small value for α (i.e. when α tends to $-\infty$), the weighted power mean takes the values of the smallest term, i.e.,

$TSI = \min\{L_1, ..., L_\tau, ..., L_p\}$. In other words, maximizing the TSI is equivalent to maximizing the least satisfied target. 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 importantly, by selecting an appropriate value for α, TSI can discriminate solutions that are equivalent on overall satisfaction, resp. balancing, but that are different on balancing, resp. overall satisfaction.

3.3. Formal Modeling of the Preferences

Our goal is to simultaneously optimize n criteria $f = (f_1, \ldots, f_i, \ldots, f_n)$. Let X be the set of feasible solutions, and $Y = f(X) \subset \mathbb{R}^n$ be the image in the criteria space of X by f. Note that we assume, without loss of generality, that the criteria have to be minimized. It would be exceptional to find a solution that simultaneously optimizes all the criteria considered in this paper. A more general definition of optimality is obtained by defining a *dominance relation*. Different dominance relations are proposed in the literature, and the most common one, adopted in this work, is the *Pareto dominance*, formally recalled in Definition 1. In Definition 2, a point in the criteria space is said to be *nondominated* if it is not possible to improve one criterion without degrading others. In Definition 3, a solution $x \in X$ is called an *efficient solution* if its projection in the criteria space results in a nondominated point. The set of nondominated points is called *Pareto front*, denoted as Y_N. When the Pareto front is not known because it is too difficult to determine, which is our case, the obtained set is referred to as the nondominated set found.

Definition 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 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 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*

Instead of a set of good quality solutions, many practical contexts require an optimization approach to provide a single solution. In this case, it becomes mandatory for decision makers to express their preferences. To make the problem as general as possible and answer the industrial requirements, we consider that the decision maker is offered two ways of expressing his/her preferences. These preferences are used during the search process of the heuristics detailed in Section 4.4.1 and 4.4.2, whereas they are only used to select a final solution among the set of nondominated solutions in the AMOSA heuristic proposed by Bandyopadhyay et al. (2008) and briefly described in Section 4.4.3. The first way that can suit the preferences of the decision maker is the use of a lexicographic order. This modeling fits the situation where no trade-off between the criteria is allowed. In the industrial context, this concerns the criteria that model constraint satisfaction, such as 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 the weighted flow factor. When the trade-off is possible, the decision maker is given the possibility of prioritizing some criteria over others through weights.

Formally, n being the number of criteria, 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 multiobjective optimization, we consider that each criterion i is associated with a *lexicographic rank* $l_i \leq n$. It is assumed that the ranks are contiguous and the smaller the rank, the more important the criterion. Let us assume without loss of generality that the indices of the criteria are given in the non-decreasing order of lexicographic rank, i.e., $l_i \leq l_{i+1}, \forall i < n$. Since such relations meet the properties of antisymmetry, transitivity and totality, lexicographic orders are total orders. This allows pairwise comparisons of objective vectors. 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., $\exists i < n$ s.t. $l_i = l_{i+1}$. 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 $c \in \mathbb{R}^n_{>0}$. Each criterion 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. Let $\mathcal{P} = \{(l_i, c_i), \forall i = 1,\ldots,n\}$ denote all the preferences of the decision maker. 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.

4. Simulated Annealing Approaches

To solve the scheduling problem described in Section 3, Simulated Annealing, whose generic pseudo-code can be found in Algorithm 1, is adopted. Starting from an initial solution, Simulated Annealing explores the search space for better solutions that optimize the different criteria. Any nondominated solution is stored in an *archive A* that corresponds, at the end of the optimization procedure, to an approximation of the Pareto front. Within AMOSA, the content of the archive at any iteration contributes to the decision to accept a new solution, i.e., A is *active*, while it is passive in the approaches proposed in this work. To handle the multiobjective aspect of the problem, our approaches use the preferences of the decision maker $\mathcal{P}$. Within AMOSA, these preferences are only used to select the final solution. In this work, a geometric cooling schedule is adopted. Therefore, an initial temperature T and a cooling rate P_c are required. Finally, "Stop" refers to the stopping criterion chosen by the user, for example a final temperature, a maximum computational time or a maximum number of non-improving moves.

This section details each iteration of the Simulated Annealing metaheuristic and highlights the differences between the compared variants. In Section 4.1, the approach of Knopp et al. (2017) is recalled and modified so that feasible schedules of our problem can be evaluated within the Simulated Annealing metaheuristic. Section 4.2 describes the best available neighborhood function for our complex scheduling problem. How the archive is maintained and how it is used in the different approaches are presented in Section 4.3. Last but not least, Section 4.4 describes the acceptance conditions that each variant of Simulated Annealing uses to decide whether the new solution x' should be selected or not. Note that, instead of the objective vectors, the acceptance probability is computed using a function that takes a solution as input. This is due to the fact that, in some cases with AMOSA, the new solution x' is replaced by one of the solutions in archive A.

Algorithm 1 Generic Multiobjective Simulated Annealing

procedure SA($x, A, \mathcal{P}, T, P_c$, Stop)
  while Stop $= False$ **do**
    $y \leftarrow f(x)$ ▷ Evaluate current solution x
    $x' \leftarrow \mathcal{N}(x)$, $y' \leftarrow f(x')$ ▷ Generate neighbor x' of x
    ADD(x', A) ▷ If possible, add x' to archive A
    if ACCEPT($x', x, \mathcal{P}, A, T$) **then**
      $x \leftarrow x'$ ▷ Replace x by x' if acceptance conditions are satisfied
    $T \leftarrow P_c * T$ ▷ Apply geometric cooling schedule

4.1. Solution Representation and Evaluation

To solve our complex job-shop scheduling problem, the batch-oblivious approach proposed in Knopp et al. (2017) is adopted and modified to consider two additional constraints: Unavailabilities and minimum time lags. This approach uses a conjunctive graph to represent solutions where nodes only model operations and arcs only model precedence constraints on routes and resources. Instead of using additional nodes and arcs to model batches as in Ovacik and Uzsoy (2012), batches are coded in the arc weights in the batch-oblivious approach. This new representation has many advantages. It reduces the structural complexity of the graph. It allows ideas and techniques for less complex problems to be reused, such as the move proposed by Dauzère-Pérès and Paulli (1997) and Dauzère-Pérès et al. (1998) 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.

Let $G = (V, E)$ be the batch-oblivious conjunctive graph with set of 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 $m^{-1}(v) \in V \setminus \{*\}$. 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 time 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 avoid increasing the complexity of the graph, the minimum time lags and availability constraints are not explicitly modeled in the graph. Then, it should be ensured that they are taken into consideration during the computation of the start times. To reflect the other constraints, we define edge weights as follows. For edge $(0, o_{1,j}) \in E$ that connects 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 time 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 time of v and the sequence-dependent setup time between v and $m(v)$ on machine $m_v = m_{m(v)}$.

To compute the start times, the nodes of the graph are traversed in topological order. When dealing with a classical conjunctive graph, the start time of a node is computed based on the start times of its predecessors and the weights of its incoming edges. To model batching decisions, the weights of resource edges $(u, v) \in E$, i.e., such that $u = m^{-1}(v)$, are adapted within the batch-oblivious approach.

Then, before computing the start time of a node v, the adequate weight of the resource edge $(u,v) \in E$ must be determined. The weight of this edge $(u,v) \in E$ is 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, the *invariant* (6) must be satisfied when considering the possibility of batching two adjacent operations u and v. This invariant can be interpreted as follows: An operation v can be batched with its resource predecessor u, which allows setting the weight $l_{u,v}$ if v occurs before u, i.e., if the availability time of v is smaller than the already computed start time of u. Using this invariant, it follows that, for each operation $u \in V$, computing the longest path leads 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 times and sequence-dependent setup times are satisfied.

$$\left(l_{u,v} = 0 \ \wedge \ S_u \geq S_{r^{-1}(v)} + l_{r^{-1}(v),v}\right) \ \vee \ (l_{u,v} = p_u + s(q_u, q_v)) \tag{6}$$

When dealing with the problem described in Section 3.1, it is required to reformulate invariant (6) and to consider minimum time lags and availability constraints when computing the start times. For each node v and for each $\lambda = (j,u,q_u,v,q_v,d) \in \mathscr{L}^{min}$, $S_v \geq S_u + d$ must hold. Then, let r_v denote the job availability time of operation v, which is computed in (7). As invariant (6) relies on the notion of job availability time of operation v, it must be reformulated to consider minimum time lags. Therefore, invariant (6) is no longer valid and is generalized in (8). Invariant (8) can then be used to determine the weight of resource edge $(u,v) \in E$ based on the feasibility of batching operations u and $v \in V$ together. After setting the weight of the resource edge incident to v, it becomes possible to compute the start time S_v. In addition to the start times of the predecessors and the edge weights, this computation must consider the minimum time lags and the unavailability periods. The computation of the job availability of operation v, denoted above by r_v, already considers the minimum time lags. Regarding the availability constraints, the start times must be adjusted to make sure that there is no overlap between operation v and unavailability periods of the machine on which the operation is processed. The necessary adjustment can be performed as proposed in Tamssaouet et al. (2018) and Mati (2010).

$$r_v = \max(S_{r^{-1}(v)} + l_{r^{-1}(v),v}, \max_{(j,u,q_u,v,q_v,d) \in \mathscr{L}^{min}} (S_u + d)) \tag{7}$$

$$(l_{u,v} = 0 \ \wedge \ S_u \geq r_v) \ \vee \ (l_{u,v} = p_u + s(q_u, q_v)) \tag{8}$$

Algorithm 2 provides the pseudo-code for a static graph evaluation algorithm. It tracks the used capacity σ_B for each node and checks if the recipes $q_{m^{-1}(v)}$ and q_v or consecutive operations are equal. The algorithm greedily creates batches while preserving invariant (8). The computation of the start time takes into consideration the minimum time lags and the availability constraints. In addition to the algorithm that computes the start times by traversing the batch-oblivious conjunctive graph, another algorithm that performs the same task while dynamically improving the solution is also proposed in Knopp et al. (2017). In our implementation, we use the latter algorithm, which is not detailed here, as several other concepts would have to be introduced. The only changes in the algorithm are that invariant (6) is replaced by invariant (8), and that the availability constraints are considered as illustrated in Algorithm 2.

Algorithm 2 Evaluation of a batch-oblivious conjunctive graph

procedure COMPUTESTARTDATESSTATICALLY(G)
  $S_0 \leftarrow 0$
  $B_v \leftarrow \{v\} \quad (\forall\, v \in V)$
  for $v \in computeTopologicalOrdering(G \setminus \{0\})$ **do**
    $r_v \leftarrow \max(S_{r^{-1}(v)} + l_{r^{-1}(v),v}, \max_{(j,u,q_u,v,q_v,d)\in\mathscr{L}^{min}}(S_u + d))$
    if $r_v \leq S_{m^{-1}(v)}$ and $q_{m^{-1}(v)} = q_v$ and $\sigma_{B_{m^{-1}(v)}} < b_v$ **then**
      $S_v \leftarrow S_{m^{-1}(v)}, \quad B_v \leftarrow B_v \cup B_{m^{-1}(v)}$
    else
      $S_v \leftarrow \max\left(r_v,\ S_{m^{-1}(v)} + p_{m^{-1}(v)} + s(m^{-1}(v), v)\right)$
    for $u_{l,m} = [S_{l,m}, C_{l,m}] \in U_m$ **do**
      if $S_v < S_{l,m}$ **then**
        if $S_v + p_v \leq S_{l,m}$ **then**
          break
        $S_v \leftarrow C_{l,m}$
      else if $S_v \leq C_{l,m}$ **then**
        $S_v \leftarrow C_{l,m}$

4.2. Neighborhood Function

The batch-oblivious approach described in the previous section requires a neighborhood operator $\mathscr{N}$ that modifies a given batch-oblivious conjunctive graph. The move introduced in Dauzère-Pérès and Paulli (1997), which integrates the resequencing and reassignment of single operations, is used. To obtain a new solution x' from a solution x, one node of the batch-oblivious conjunctive graph associated with x is randomly chosen, its feasible insertion positions are computed, and one of them is randomly selected and performed. As explained in Section 2.3, the large size of the neighborhood motives the use of such approaches as long as efficient neighborhood functions are not available.

4.3. Archive and Reference Points

The archive stores solutions not yet dominated by any other solutions found so far. This is the set of solutions finally returned by the optimization algorithm. This archive passively stores nondominated solutions in the a priori heuristics as it plays no role in the search process. On the opposite, it is actively used with the search in the AMOSA heuristic. In this work, the update of the archive with the new solutions found during the search, and which are potentially nondominated, relies on Pareto dominance, which is recalled in Definition 1. A new solution x' is added to archive A only if it is not dominated by any solution already in A. When this is the case, any solution in the archive dominated by x' must be removed from the archive.

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. When this strategy turns out to be computationally expensive, 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. 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)).

Preliminary numerical results show that better results are obtained when constraining the size of the archive, especially in the case of AMOSA. Therefore, similarly to Bandyopadhyay et al. (2008), we kept the archive size limited in our implementation. There are two limits on the size of the archive: A hard limit denoted by HL, and a soft limit denoted by SL such that $SL \geq HL$. The size of the archive is allowed to increase up to SL, after which the solutions are grouped in HL clusters using the single linkage algorithm (Jain and Dubes (1988)). The member within each cluster whose average distance to the other members is minimum is considered to be the representative member of the cluster.

If the set of efficient solutions is known, it is possible to define three reference points: The ideal point, the utopian point and the nadir point. These points are respectively defined in Definitions 4, 5 and 6. The nadir and ideal points produce important information on a multiobjective optimization problem. For a decision maker, they show the possible range of all the criteria over the Pareto set: They are respectively exact upper and lower bounds of the set of nondominated points. As the set of efficient solutions is unknown in our case, y^{id}, y^{ut} and y^{na} denote in the remainder of this paper approximations of the ideal point, the utopian point, and the nadir point, respectively. Using archive A_k, the nadir point can be estimated at any iteration of the search process by replacing the Pareto front Y_N in Definition 6 by its approximation A_k. Without knowledge of the Pareto front, we use the utopian point instead of the ideal point.

Definition 4 (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 (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 6 (Nadir Point). *A point* $y^{na} = (r_1^{na}, \ldots, r_n^{na})$ *is called* nadir *(or* anti-ideal point*) if and only if* $y_i^{na} = \max_{y \in Y_N} y_i, \ \forall\, i \in \{1, \ldots, n\}$.

To determine utopian point $y^{ut} \in \mathbb{R}^n$, all the resource constraints in the conjunctive graph are relaxed. So, the conjunctive graph only consists of the edges related to the routes of the jobs. For each job, the shortest path is chosen. In other words, the processing time of each operation of a job is its processing time when assigned to its fastest machine. Note that the release dates of jobs are still modeled in the resulting graph. Then, the computation of the earliest start dates of operations in this graph yields a schedule without waiting periods. The computation of the criteria for such schedules leads to lower bounds of the considered regular objective functions to minimize and upper bounds for the regular objective functions to maximize. By relaxing the resource constraints, the notion of batch disappears. As it is not trivial to compute the context-dependent upper bound for the batching coefficient criterion, it is considered equal to 1.

4.4. Fitness Assignment to Solutions

At each iteration of the Simulated Annealing metaheuristic, the decision of keeping or discarding the new solution x' must be taken. When a single criterion is optimized, x' is automatically accepted with probability equal to one if it is not worse than the current solution x. Otherwise, it is accepted with a probability lower than one. Because of the difficulty of comparing different solutions and computing the

acceptance probability, several ways of assigning fitness to candidate solutions during the search can be found in the literature. The general formulation of the preferences provided in this work, by considering simultaneously a lexicographic order and weights, requires the Simulated Annealing metaheuristic to be extended. The acceptance conditions in the two proposed approaches are detailed in Sections 4.4.1 and 4.4.2. These approaches differ in the way the preferences are used. As the different objective functions have different units and significant different orders of magnitude, they must be normalized. The chosen normalization, considered as robust (Arora (2017)), uses the approximation of the nadir and utopian points as shown in (9).

$$y_i = \frac{f_i(x) - y_i^{ut}}{y_i^{na} - y_i^{ut}} \tag{9}$$

Concerning the aggregation function, the weighted sum is adopted in this work. Another known aggregation function in the context of multiobjective optimization is the weighted Tchebychev metric (see T'kindt and Billaut (2006)). The advantage of this second aggregation function, when the parameters are well chosen, is that it makes it possible to reach any efficient solution. This property is not guaranteed when the weighted sum is used, because efficient solutions (called *unsupported* solutions) may not be achievable with a set of coefficients. Despite the absence of this important property, the weighted sum leads to better results in our numerical experiments. Therefore, only the results obtained using the the weighted sum are reported. In the approaches described in Sections 4.4.1 and 4.4.2, the aggregation function is used in two cases: (1) To aggregate criteria with the same lexicographic rank l, i.e., $\sum_{i,l_i=l}^{n} c_i y_i$, and (2) To aggregate all the criteria after dropping all the preferences, i.e., $\sum_{i}^{n} y_i$.

The approaches we propose are compared to a successful a posteriori approach, AMOSA. To decide on accepting or not a new solution, AMOSA relies on the dominance status between the new solution and the current solution, and between the new solution and the solutions in the current archive. Section 4.4.3 briefly summarizes the main features of AMOSA, and the reader can refer to Bandyopadhyay et al. (2008) for a full description of the approach.

4.4.1. SA-I: Sticking to the Preferences

The general idea of approach SA-I is to stick to the preferences of the decision maker when deciding if the new solution is not worse than the current solution, and when computing the acceptance probability when it is not the case. The way the approach uses the preferences $\mathcal{P}$ to decide on the acceptance of a new solution x' is detailed in Algorithm 3. First, using (9), SA-I starts by normalizing the objective vector $f(x)$ and $f(x')$ to obtain y and y'. As a total ordering, a lexicographic order allows pairwise comparisons of objective vectors. However, as some criteria may share the same lexicographic rank, this property can no longer be used. Therefore, the idea is to use, for each lexicographic rank, the weighted sum to aggregate the related criteria. By doing this, the obtained vectors $\overline{y}$ and $\overline{y}'$ can be compared lexicographically. If $\overline{y}'$ is not worse than $\overline{y}$, the new solution x is accepted. Otherwise, the acceptance probability should be lower than one. As the general idea of SA-I is to stick to the preferences, the acceptance probability relies only on the lexicographic order where $\overline{y}'$ is worse than $\overline{y}$. When analyzing the approach of Baykasoğlu (2005) proposed for lexicographic goal programming, it appears that this approach and SA-I share the same adaptations of Simulated Annealing.

4.4.2. SA-II: Loosening the Preferences

The assumption behind SA-I is that following 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 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 towards regions where

Algorithm 3 Acceptance conditions of SA-I with weighted sum

procedure ACCEPT(x', x, P, A, T)
  $y \leftarrow \text{NORMALIZE}(f(x))$, $y' \leftarrow \text{NORMALIZE}(f(x'))$
  $\bar{y} \leftarrow (\bar{y}_1, \ldots, \bar{y}_l, \ldots, \bar{y}_{l_n})$, $\bar{y}_l \leftarrow \sum_{i,l_i=l}^{n} c_i y_i$
  $\bar{y}' \leftarrow (\bar{y}'_1, \ldots, \bar{y}'_l, \ldots, \bar{y}'_{l_n})$, $\bar{y}'_l \leftarrow \sum_{i,l_i=l}^{n} c_i y'_i$
  if $\bar{y}' \preceq \bar{y}$ **then**
    return True
  $\Delta = \bar{y}'_m - \bar{y}_m \mid \bar{y}'_m > \bar{y}_m \wedge \bar{y}'_l = \bar{y}_l \forall l < m$
  if $exp(\frac{-\Delta}{T}) \leq \text{RANDOM}(0,1)$ **then**
    return True
  return False

the most important criteria have better values. The way SA-II uses the preferences $\mathcal{P}$ to accept a new solution x' is detailed in Algorithm 4. In SA-II, the preferences are still considered when deciding whether the new solution x' is not worse than the current solution x. Therefore, SA-II differs from SA-I in the computation of the acceptance probability. If the lexicographic order is to be ignored, it does not seem consistent to use the weights either. Therefore, all the criteria of y and y' are aggregated using the chosen function without considering the weights given by the decision maker. In other words, this approach considers that all the criteria have the same importance.

Algorithm 4 Acceptance conditions of SA-II with weighted sum

procedure ACCEPT(x', x, P, A, T)
  $y \leftarrow \text{NORMALIZE}(f(x))$, $y' \leftarrow \text{NORMALIZE}(f(x'))$
  $\bar{y} \leftarrow (\bar{y}_1, \ldots, \bar{y}_l, \ldots, \bar{y}_{l_n})$, $\bar{y}_l \leftarrow \sum_{i,l_i=l}^{n} c_i y_i$
  $\bar{y}' \leftarrow (\bar{y}'_1, \ldots, \bar{y}'_l, \ldots, \bar{y}'_{l_n})$, $\bar{y}'_l \leftarrow \sum_{i,l_i=l}^{n} c_i y'_i$
  if $\bar{y}' \preceq \bar{y}$ **then**
    return True
  $\hat{y} \leftarrow \sum_i^n y_i, \hat{y}' \leftarrow \sum_i^n y'_i$
  $\Delta = \hat{y}' - \hat{y}$
  if $exp(\frac{-\Delta}{T}) \leq \text{RANDOM}(0,1)$ **then**
    return True
  return False

4.4.3. Archived Multiobjective Simulated Annealing: AMOSA

The third alternative explored in this work is the Archived Multiobjective Simulated Annealing (*AMOSA*) of Bandyopadhyay et al. (2008). The purpose of this section is to highlight the differences between AMOSA and the proposed approaches. As an a posteriori heuristic, AMOSA does not require the preferences of the decision maker during the search process, but relies instead on Pareto dominance. In addition to the current solution x, the acceptance of the new solution x' is also based on the solutions in the archive A. To determine the acceptance probability of a new solution, an elaborate procedure is followed that considers the dominance status of the new solution with the current solution, as well as with the solutions in the archive. A measure of the amount of dominance between two solutions is also used for this purpose. The reader can refer to Bandyopadhyay et al. (2008) for a full description of AMOSA. All the main building blocks described in the original work are implemented to perform the

experiments in Section 6, except for the initialization phase. AMOSA begins with the initialization of a number of solutions. For a number of iterations, each of the solutions is improved by using a simple hill-climbing technique, which is prohibitive in the case of our complex scheduling problem. Therefore, our implementation does not include this initialization phase.

5. Multiobjective GRASP Approaches

In this section, we present heuristics that rely on the idea of Greedy Randomized Adaptive Search Procedure (GRASP) of Feo and Resende (1995), and that is used within this work to diversify the search and to benefit from the parallelism of modern CPUs. In this multi-start procedure, each iteration consists basically of two phases: Construction and local search. A feasible solution is built in the construction phase and improved in the local search phase. Section 5.1 describes our construction heuristic. One of the Simulated Annealing variants presented in Section 4 is used to improve the initial solutions. Therefore, Section 5.2 only describes how these improvements heuristics are used within the global approach and how their parameters are determined.

5.1. Construction Heuristic

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 times 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 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 values. 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 they are not dominated by solutions already in the archive.

5.2. Improvement Heuristics

As described in Section 5.1, our heuristic creates many different starting solutions by randomizing a construction algorithm that greedily inserts operations. Then, each constructed solution is independently improved using one of the Simulated Annealing metaheuristics: SA-I, SA-II and AMOSA. The GRASP approach ends when the maximum allowed computational time is reached. As this computational time is short in the industrial setting, The global 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 following 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, i.e. procedure ADD(x',A) in Algorithm 1. 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.

Unlike the global approach, the Simulated Annealing metaheuristics are stopped when the maximum number of non-improving moves P_m is reached. On the one hand, a non-improving move within the proposed a priori approaches might refer to a move that does not improve the global best solution. On the other hand, as the search of AMOSA relies on the archive, it is more natural to let a non-improving move refers instead to a move that does not lead to an update of the archive A. Therefore, the value of parameter P_m must be carefully chosen in Section 6 to make sure that the comparison is fair between these two approach classes. 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 of the new solution x' obtained after performing a selected move and the criterion of the initial solution x, i.e., $\Delta = y' - y'$. 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. When considering multiple criteria, similarly to the computation of the acceptance probability in SA-II, the difference is computed after aggregating the criteria using the weighted sum, i.e., $\Delta = \sum_i^n y'_i - \sum_i^n y_i$.

6. Numerical Results

The batch-oblivious approach and the extensions proposed in Section 4.1 make our approach capable of tackling the generic industrial scheduling problem found in a work area in semiconductor manufacturing facilities. Thus, this section focuses on the multiobjective aspect of the scheduling problem and relies on large industrial instances to study the performance of the three GRASP approaches, i.e., SA-I, SA-II and AMOSA. The global approach with the different blocks described earlier is implemented in C++14. All computational experiments are conducted on an Intel Core i7-7700 3,60 Ghz machine (4 cores) running Microsoft Windows 10. The relevant details of the experiments are provided in Section 6.1. The comparison between the different studied approaches is based on the quality of the final selected solution. The results are compared and analyzed in Section 6.2. Finally, Section 6.3 aims at estimating the improvement that our approach can bring by comparing its proposed schedules with the actual factory schedules.

6.1. Experiment Design

For each instance, each GRASP approach is run only once since several independent runs of Simulated Annealing are performed on different threads. The sampling strategy of our approach avoids the need to adapt parameters for individual instances. We used the following identical parameter settings for all experiment: A cooling factor $P_c = 0.99999$, a number of samples $P_s = 100$ and a perturbation intensity $P_i = 5$. The values of these parameter are the ones used in Knopp et al. (2017), where a large number of instances is used to tune them.

To conduct a fair comparison between the different approaches, we believe that the maximum number of non-improving iterations P_m and the temperature percentile P_t must be carefully chosen. As indicated in Section 5.2, different ways of counting the number of non-improving iterations can be considered for AMOSA and the proposed a priori approaches. Therefore, to make the comparison fair and simple, we used a large maximum number of non-improving iterations ($P_m = 10\,000$) to make sure that the stopping criterion of the individual Simulated Annealing approaches is actually the maximum allowed computational time. The initial temperature is another critical parameter that has a significant impact on the performance of any Simulated Annealing approach. In our GRASP approach, the sampling strategy helps setting reasonably good values for the initial temperature. Nevertheless, it should be ensured that the value selected for the temperature percentile P_t that controls the sampling strategy does not

favor one approach over others. The experiments are therefore conducted by using different values of $P_p \in \{5\%, 10\%, 20\%, 30\%, 40\%, 50\%\}$. The results show that using $P_p = 5\%$ leads to the best results for AMOSA and most of the proposed approaches, and this corresponds to the value chosen in Knopp et al. (2017). Finally, the hard limit on the size of the archive is $HL = 100$ and the soft limit is $SL = 150$.

Two instance sets are used to conduct the numerical experiments. The first set of instances, called LI, contains 15 large industrial instances. Each instance captures the situation of the work area at a given point in time, where the number of jobs to schedule ranges from 350 to 550. These instances correspond to the scheduling problem to be solved each time the proposed approach is used to control the shop floor. Thus, the comparison between the different GRASP approaches in Section 6.2 uses the LI instances. The second set of instances, called VLI, contains ten very large instances where the number of jobs ranges from 1,500 to 1,800. Each instance describes the scheduling problem of the studied work area within a period of one day. These instances are described in more detail and used in Section 6.3 to evaluate the potential impact of the approach developed in this work by comparing its results to the actual industrial results. For each job in these two sets of instances, between one and five operations have to be performed, with three operations on average. On average, the jobs must be scheduled on 68 machines, all capable of batching, and the batching capacity is between 2 and 7 jobs. As defined in Section 3.1, the scheduling problem includes the following constraints: Release dates, minimum time lags, sequence-dependent setup times, availability constraints and batching constraints.

The criteria described in Section 3.2 are considered in the experiments: Target Satisfaction Indicator (TSI), Weighted Number of Moves (WNM), Weighted Flow Factor (WFF) and Batching Coefficient (BC). To study the influence of the chosen preferences on the a priori approaches search, the experiments are conducted using three realistic sets of preferences shown in Table 1. Each criterion is respectively given a lexicographic rank and a weight as a pair (l, c) to reflect the preferences of the industrial partner.

	TSI	**WNM**	**WFF**	**BC**
$\mathcal{P}_1$	(1,1)	(2,1)	(2,1)	(3,1)
$\mathcal{P}_2$	(1,1)	(2,1)	(2,5)	(3,1)
$\mathcal{P}_3$	(1,1)	(2,5)	(2,1)	(3,1)

Table 1: Used preferences

Besides the preferences, the values of additional criteria related parameters must be fixed. First, through the parameter α, the proposed criterion TSI allows the decision maker to prioritize, depending on the context, the overall satisfaction or the balancing, without ignoring the other. For the sake of space, $\alpha = 1$ is used in the experiments. Another important parameter that must be determined before solving the scheduling problem is the horizon that is required to compute the values of TSI, WNM, and BC. When solving the LI instances, this horizon is equal to 8 hours, which corresponds to a shift. Regarding the VLI instances, 24 hours is the chosen horizon, which corresponds to the period fully described by the instance data.

Finally, Another industrially critical parameter is the allowed computational time. Because of the dynamic aspect of the industrial setting, the scheduling approach must be run frequently on instances that are similar to the LI instances for a computational time that does not exceed 5 minutes. The experiments using the VLI instances in Section 6.3 are conducted by allowing a computational time of 30 minutes. The choice of this computational time is motivated by the fact that the average number of explored solutions in 30 minutes for the VLI instances (approximately 650,000) is equal to the average number of explored solutions for the LI instances in 5 minutes.

6.2. *Comparison Based on Preferences*

The objective of this section is to compare the different GRASP approaches through the quality of the final solution which is selected using the preferences provided by the decision maker. In the preferences shown in Table 1, only WNM and WFF share the same lexicographic order. After the aggregation of these two criteria, the lexicographic order makes it possible to select a unique solution. Table 2 presents a summary of the obtained results. For each instance and approach, the relative deviation of the value of each criterion of each final solution from the best value is computed. To facilitate the analysis, the relative deviations of WNM and WFF are aggregated using the related preferences, as they share the same lexicographic rank in the three vectors of preferences. Due to the limited space of the paper, only the mean and maximum of the resulting relative deviations are reported in Table 2. The results are grouped by the preferences $\mathcal{P}_j$, and each row corresponds to the aggregated results of each approach. Columns 1, 2 and 3 correspond respectively to the average or to the maximum of the relative difference for lexicographic ranks 1, 2 and 3. The best values within the results of each of the preferences vector are in bold.

Preferences	**Approach**	**Mean**			**Max**		
		1	**2**	**3**	**1**	**2**	**3**
$\mathcal{P}_1$	**AMOSA**	4.7	5.3	5.9	9.4	12.1	10.3
	SA-I	0.8	8.3	7.3	**2.9**	24.2	12.1
	SA-II	0.8	**4.6**	**1.1**	**2.9**	**12.4**	**4.3**
$\mathcal{P}_2$	**AMOSA**	4.4	6.8	5.5	8.3	18.2	9.1
	SA-I	**0.7**	8.0	6.7	**2.9**	15.1	10.9
	SA-II	0.9	**4.9**	**1.3**	**2.9**	**12.1**	**4.0**
$\mathcal{P}_3$	**AMOSA**	4.6	5.6	6.4	8.8	8.4	11.1
	SA-I	**0.7**	7.0	7.0	**2.9**	18.8	11.7
	SA-II	0.9	**3.0**	**1.0**	3.5	**9.3**	**3.0**

Table 2: Quality of the final solutions.

Let us first analyze the results by considering each lexicographic rank separately. Regarding the first lexicographic rank that corresponds to the criterion TSI, SA-I provides the best results in terms of mean and maximum relative deviation. For this rank, AMOSA obtains results of poor quality because its average and maximum relative deviations are respectively larger by at least 3% and 5% than those of SA-I. On the contrary, whatever the preferences, the difference between the SA-II and SA-I approaches is quite insignificant. More precisely, a difference of at most 0.2% from the mean and not more than 0.4% from the maximum relative deviation can be observed. The results of SA-I can be explained by its design, since the preferences are used when deciding whether the new solution is not worse than the current one and when computing the acceptance probability. However, this choice to stick closely to the preferences during the search comes at the cost of poor results for the higher lexicographic ranks. Note that AMOSA obtains better results than SA-I at lexicographic ranks 2 and 3. For example, the mean relative deviation of SA-I at lexicographic rank 2 is worse than the deviation of AMOSA by 3% when using preferences $\mathcal{P}_1$.

The superiority of SA-II can be observed when we focus on the two higher lexicographic ranks because it always obtains the best results. By design, this approach uses preferences only to decide whether the new solution is not worse than the current one. When calculating the acceptance probability, the pref-

erences are not taken into account, leading to the equal consideration of all criteria. This explains the insignificant difference with SA-I regarding lexicographic rank 1 and the best results for the remaining ranks. Considering all the results presented so far, it can be concluded that SA-II is the most appropriate approach in the industrial context.

Beyond the industrial application, the results in Table 2 can be used to analyze the impact of the fitness assignment to solutions during the search process and the way the preferences are considered. In the studied context, where decision makers have to express their preferences before the optimization, the results show that using the preferences during the search (our approaches) generally leads to better results than when only using the preferences to select the final solution from the archives produced by an a posteriori approach (AMOSA). However, special attention should be paid to the way the preferences are taken into account into the search process. For example, except for TSI, SA-I shows poor results compared to AMOSA. In case of extreme preferences such as a lexicographic order, strictly following the preferences during the search usually leads to poor results for the least important criteria without necessarily ensuring better results regarding the most important criteria. This is illustrated in the comparison between SA-I and SA-II. Relaxing the preferences when computing the acceptance probability leads to better results for the least important criteria without any significant negative impact on the most important criterion.

6.3. Potential Impacts of the Proposed Approach on Industrial Instances

This section aims at validating our approach by comparing its schedules to the actual schedules of the studied work area. To conduct the comparison, all relevant data for ten different days over six months have been extracted from the Manufacturing Execution System. The obtained instances are VLI instances. To perform a fair comparison, all the constraints and events that can affect the schedule quality must be reflected in the instances. After some manufacturing operations, some jobs are measured on inspections machines to monitor the process stability of machines and the quality of products (see, for example, Dauzère-Pérès and Hassoun (2019)). Measurement operations are not explicitly considered in our instances. To make the model realistic, the transport times between machines are overestimated to one hour to take into consideration the waiting times of jobs that are measured in front of the inspection machines and the corresponding measurement times. Minimum time lags are used to model the transport times, i.e., a minimum time lag constraint of one hour is systematically added between any two consecutive operations of any job. As explained above, the experiments are conducted by allowing a computational time of 30 minutes. The results of the GRASP 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 3.

The numerical results show that our approach can bring significant improvements in the operational performance. Indeed, there are eight instances out of ten for which the proposed solutions are dominating the actual solutions on all criteria. For the two remaining instances (3 and 5), the actual solutions are only better for the weighted flow factor (WFF). Because of the decision-maker preferences, all the solutions obtained by our approach are better than the actual ones. 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 and the weighted number of moves are improved by more than 5%. These results demonstrate that there is room for improving the operational performance of the studied work area, and that the proposed approach efficiently solves the industrial scheduling problem while taking the rich set of constraints and criteria into account.

Instances	*TSI*	*WWF*	*WNM*	*BC*
1	7.7%	-9.4%	9.2%	4.8%
2	22.1%	-5.9%	2.7%	5.0%
3	10.6%	11.4%	2.3%	5.2%
4	9.3%	-19.9%	11.2%	4.2%
5	8.3%	4.6%	9.7%	4.8%
6	12.8%	-5.1%	3.4%	2.8%
7	9.6%	-5.8%	2.1%	7.0%
8	7.7%	-7.0%	3.6%	8.4%
9	7.9%	-15.4%	7.4%	7.5%
10	10.1%	4.3%	11.1%	6.6%
Mean	10.6%	-4.8%	6.3%	5.6%
Median	9.6%	-5.8%	6.3%	5.2%

Table 3: Detailed results comparing the schedules determined by our approach and the actual schedules

7. Conclusions

In this paper, we proposed an approach to solve a multiobjective complex job-shop scheduling problem stemming from semiconductor manufacturing. To construct feasible schedules, the recently proposed batch-oblivious approach (Knopp et al. (2017)) is adopted and first extended by considering unavailability periods and minimum time lags. Different criteria, suitable to optimize the local performance of the work area, are optimized. Besides, a novel criterion is proposed to model the contribution of the produced schedule to the realization of production targets defined at the factory level. Having a multiobjective problem where a trade-off is only allowed among some criteria, the preferences of the decision maker are modeled through a lexicographic order and weights. As this approach must be embedded in a real-time application where the available time for decision making is limited, the decision maker is constrained to express his/her preferences before the optimization phase.

To solve our multiobjective complex scheduling problem, Simulated Annealing appears to be one of the most suitable approaches. To our knowledge, the simultaneous use of a lexicographic order and weights to model the preferences of decision makers has not been studied in the multiobjective optimization literature, although it is relevant in real-world applications. Due to this generic modeling of the preferences, Simulated Annealing must be extended if the preferences have to be used during the search. Two extensions of Simulated Annealing are investigated, depending on how the preferences are exploited during the search. To study if a new solution is not worse than the current one, the chosen aggregation function is applied to all criteria sharing the same lexicographic rank within all proposed approaches. Therefore, the lexicographic relation can be used to compare the resulting vectors. If the new solution is worse than the current one, an acceptance probability must be computed. The first approach (SA-I) consists of sticking to the preferences by aggregating only the criteria on the most important lexicographic rank where the two solutions differ. In the second approach (SA-II), all the criteria are aggregated by ignoring the lexicographic ranks and weights. To represent a posteriori approaches, AMOSA proposed

in Bandyopadhyay et al. (2008) is adopted.

The numerical results show that the proposed approaches, by taking the preferences into account, provide good quality solutions regarding these preferences. Another important lesson learned from the analysis of the computational experiments is that relaxing the preferences during the search (e.g., computation of the acceptance probability) may lead to solutions that are of high quality regarding the most important criteria as well as the least important ones. By relaxing the preferences, optimizing the least important criteria becomes possible while the most important ones benefit from the diversification of the search.

Despite the encouraging results obtained by our approaches, several research perspectives can be explored. As shown in Section 2.3, the choice of the solution approach is highly constrained by the large size of the neighborhood. To overcome this obstacle and make it possible to use other heuristics relying on local search, the design of efficient neighborhood functions is a necessary step. Also, due to the successful application of Genetic Algorithms to a broad range of multiobjective optimization problems, a potential future research direction is to define a solution representation for our complex scheduling problem that can be coupled with an efficient decoding algorithm.

References

Aggelogiannaki, E., Sarimveis, H., 2007. A simulated annealing algorithm for prioritized multiobjective optimization implementation in an adaptive model predictive control configuration. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37 (4), 902–915.

Aggoune, R., 2004. Minimizing the makespan for the flow shop scheduling problem with availability constraints. European Journal of Operational Research 153 (3), 534–543.

Amjad, M. K., Butt, S. I., Kousar, R., Ahmad, R., Agha, M. H., Faping, Z., Anjum, N., Asgher, U., 2018. Recent research trends in genetic algorithm based flexible job shop scheduling problems. Mathematical Problems in Engineering 2018.

Arora, J. S., 2017. Chapter 18: Multi-objective optimum design concepts and methods. In: Arora, J. S. (Ed.), Introduction to Optimum Design (Fourth Edition), fourth edition Edition. Academic Press, Boston, pp. 771 – 794.

Artigues, C., Dauzère-Pérès, S., Derreumaux, A., Sibille, O., Yugma, C., 2006. A batch optimization solver for diffusion area scheduling in semiconductor manufacturing. IFAC Proceedings Volumes 39 (3), 733–738.

Bandyopadhyay, S., Saha, S., Maulik, U., Deb, K., 2008. A simulated annealing-based multiobjective optimization algorithm: Amosa. IEEE Transactions on Evolutionary Computation 12 (3), 269–283.

Baykasoğlu, A., 2005. Preemptive goal programming using simulated annealing. Engineering Optimization 37 (1), 49–63.

Bitar, A., Dauzère-Pérès, S., Yugma, C., 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., Pesch, E., 1996. The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research 93 (1), 1–33.

Chang, S.-C., Lee, L.-H., Pang, L.-S., Chen, T.-Y., Weng, Y.-C., Chiang, H.-D., Dai, D.-H., 1995. Iterative capacity allocation and production flow estimation for scheduling semiconductor fabrication. In: Seventeenth IEEE/CPMT International Electronics Manufacturing Technology Symposium. pp. 508–512.

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

Ciavotta, M., Minella, G., Ruiz, R., 2013. Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study. European Journal of Operational Research 227 (2), 301 – 313.

Czyżżak, P., Jaszkiewicz, A., 1998. Pareto simulated annealinga metaheuristic technique for multiple-objective combinatorial optimization. Journal of Multi-Criteria Decision Analysis 7 (1), 34–47.

Dauzère-Pérès, S., Hassoun, M., 2019. On the importance of variability when managing metrology capacity. European Journal of Operational Research .

Dauzère-Pérès, S., 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.

Dauzère-Pérès, S., Roux, W., Lasserre, J., 1998. Multi-resource shop scheduling with resource flexibility. European Journal of Operational Research 107 (2), 289–305.

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

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

García-León, A. A., Dauzère-Pérès, S., 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.

Genova, K., Kirilov, L., Guliashki, V., 2015. A survey of solving approaches for multiple objective flexible job shop scheduling problems. Cybernetics and Information Technologies 15 (2), 3–22.

Horng, S.-M., Fowler, J. W., Cochran, J. K., 2000. A genetic algorithm approach to manage ion implantation processes in wafer fabrication. International Journal of Manufacturing Technology and Management 1 (2-3), 156–172.

Jain, A. K., Dubes, R. C., 1988. Algorithms for clustering data. Prentice-Hall, Inc.

Jain, A. S., Meeran, S., 1999. Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research 113 (2), 390–434.

Jones, D., Tamiz, M., et al., 2010. Practical goal programming. Vol. 141. Springer.

Kao, Y.-T., 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.

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

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

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

Lin, S.-W., Ying, K.-C., 2013. Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm. Computers & Operations Research 40 (6), 1625–1647.

Loukil, T., Teghem, J., Fortemps, P., 2007. A multi-objective production scheduling case study solved by simulated annealing. European journal of operational research 179 (3), 709–722.

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

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

Mathirajan, M., 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., 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.

Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., 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., 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.

Nowicki, E., Smutnicki, C., 2005. An advanced tabu search algorithm for the job shop problem. Journal of Scheduling 8 (2), 145–159.

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

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

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

Pranzo, M., Pacciarelli, D., 2016. An iterated greedy metaheuristic for the blocking job shop scheduling problem. Journal of Heuristics 22 (4), 587–611.

Romero, C., 1991. Handbook of critical issues in goal programming. Pergamon Press.

Roy, B., Sussmann, B., 1964. Les problèmes dordonnancement avec contraintes disjonctives. Note ds 9.

Ruiz, R., Sttzle, T., 2008. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. European Journal of Operational Research 187 (3), 1143 – 1159.

Sadeghi, R., Dauzère-Pérès, S., Yugma, C., 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). p. 5.

Serafini, P., 1994. Simulated annealing for multi objective optimization problems. In: Multiple criteria decision making. Springer, pp. 283–292.

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

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

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

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

Varadharajan, T., Rajendran, C., 2005. A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. European Journal of Operational Research 167 (3), 772–795.

Wang, H., Olhofer, M., Jin, Y., 2017. A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. Complex & Intelligent Systems 3 (4), 233–245.

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

Yugma, C., Dauzère-Pérès, S., Artigues, C., Derreumaux, A., 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., 07 2010. Scheduling with time lags. Ph.D. thesis, Erasmus University Rotterdam.