Unrelated parallel machine scheduling with new criteria: Complexity and models

Abdoul Bitar, Stéphane Dauzère-Pérès, Claude Yugma

Unrelated parallel machine scheduling with new criteria: Complexity and models

Abdoul Bitar[1] Stéphane Dauzère-Pérès[1,2] Claude Yugma[1]

[1]Mines Saint-Etienne, Univ Clermont Auvergne
CNRS, UMR 6158 LIMOS
CMP, Department of Manufacturing Sciences and Logistics
Gardanne, France
E-mail: abdlbitar@gmail.com, {dauzere-peres, yugma}@emse.fr

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

Abstract

In this paper, a scheduling problem on non-identical parallel machines with auxiliary resources and sequence-dependent and machine-dependent setup times is studied. This problem can be found in various manufacturing contexts, and in particular in workshops of wafer manufacturing facilities. Three different criteria are defined and analyzed: The number of products completed before the end of a given time horizon, the weighted sum of completion times and the number of auxiliary resource moves. The first criterion is maximized, while the two others are minimized. The first and the third criteria are not classical in scheduling theory, but are justified in industrial settings. The complexity of the problem with each of the new criteria is characterized. Integer linear programming models are also proposed and numerical experiments are conducted to analyze their behavior.

Keywords: Scheduling, Unrelated parallel machines, Sequence-dependent setup times, Auxiliary resources, Complexity, Integer linear programming

1. Introduction

The photolithography workshop is critical in the fabrication process of many electronic devices. In this workshop, lots of silicon *products* have to be processed on non-identical parallel machines (see Moench et al. (2011)).

Each lot, called *job* in this paper, typically consists of its maximum size of 25 products that are processed as a whole. Although a job comes back multiple times (up to 40 times for the most complex processes) to the photolithography workshop in its manufacturing route, only one operation per job is considered on a short-term scheduling horizon. To be processed, each job requires one and only one auxiliary resource (often simply called *resource* in this paper). This resource is necessary to shape a pattern on the products. Hence, an operation can only be performed if both the job and the right auxiliary resource are available at a given time. Also, an auxiliary resource must be on the machine for the duration of the process and each auxiliary resource is unique because of its cost. Thus, if two jobs require the same auxiliary resource, they cannot be processed in parallel. Indeed, one auxiliary resource can be required by multiple jobs. Moreover, machines need a specific configuration to process a job and switching from one configuration to another on a machine requires a setup time, depending on the previous job, the next job and the machine. Thus, setup times are both sequence dependent and machine dependent. Finally, machines are only *qualified* (or eligible) for a limited number of jobs, i.e. they cannot process the other ones: For each job (or job family), the set of its qualified machines is the set of machines that can process the job. Also, machines do not always have the same processing times for the same job. Hence, machines are non-identical, or unrelated.

Three criteria are introduced and motivated in Section 2.3. Two of these criteria have, to our knowledge, never been studied in the scientific literature, except in Bitar et al. (2016) for one of them. The first original criterion is motivated by the fact that jobs are continuously arriving in the workshop and, although release dates are not considered, it is relevant to optimize the schedule of jobs on a limited time horizon. The second original criterion is related to the management of auxiliary resources.

The paper is organized as follows. The scheduling problem is described in Section 2, with its constraints and the three criteria. Section 3 surveys the existing work on related scheduling problems. Complexity results and Integer Linear Programming (ILP) models are provided in Sections 4, 5 and 6 for each criterion respectively. Some numerical results are discussed in Section 7, and Section 8 concludes the paper.

2. Problem description

In the following, the problem is defined. First, some notations are given, then the constraints and the criteria are introduced.

2.1. Notations

The notations below are used throughout the article to denote each instance of the unrelated parallel machine scheduling problem with processing set restrictions, sequence-dependent and machine-dependent setup times and auxiliary resources. Let us consider a set $\mathcal{J}$ of N jobs, a set $\mathcal{M}$ of M machines and a set $\mathcal{A}$ of A ($A \leq N$, in practice A is usually much smaller than N) auxiliary resources, necessary to process jobs on machines. We also have:

- For job $j \in \mathcal{J}$, its number of products $n_j \in \mathbb{N}$, its priority w_j, its set of qualified machines $\mathcal{M}_j \subseteq \mathcal{M}$, and the auxiliary resource $\varphi_j \in \mathcal{A}$ required to process it,
- For auxiliary resource $a \in \mathcal{A}$, the set $\mathcal{E}_a = \{j \in \mathcal{J} \mid \varphi_j = a\}$ of jobs in $\mathcal{J}$ that require this resource to be processed,
- For job j and machine m, $m \in \mathcal{M}_j$, the processing time $\rho_{jm} \in \mathbb{N}$ of job j on machine m,
- For jobs j and k, and machine m such that $m \in \mathcal{M}_j \cap \mathcal{M}_k$, the sequence-dependent and machine-dependent setup time $\beta_{jmk} \in \mathbb{N}$ required by machine m to process job k *immediately* after job j,
- For auxiliary resource $a \in \mathcal{A}$, its initial location $R_a \in \{0, \ldots, M\}$, the machine where the auxiliary resource is initially located, where 0 stands for the storage area,
- A specified time horizon $[0, H]$, where $H \in \mathbb{N}$ is the end of the time horizon,
- Let us denote by Π_h the problem of maximizing the number of products completed before H, by Π_w the problem of minimizing the weighted sum of completion times and by Π_M the problem of minimizing the number of auxiliary resource moves.

2.2. Constraints

The constraints of the problem are listed below:

- A job j is processed once and only once on a machine m qualified for this job, i.e. $m \in \mathcal{M}_j$,
- The start dates are integers and machines are available at time 0,

- There is no preemption, i.e. a job is not interrupted after its start,
- A machine can only process one job at a time,
- There are sequence-dependent and machine-dependent setup times between consecutive processes of jobs on the same machine (we assume that the first job scheduled on each machine does not require setup times, although the models proposed later could be extended to consider initial setup times),
- Two jobs j and k having the same required auxiliary resource (i.e. $\varphi_j = \varphi_k$) cannot be processed simultaneously on two different machines,
- Moving an auxiliary resource from one machine to another takes a unitary transport time.

2.3. Criteria

Three objective functions covering important goals, such as job priorities, productivity, and auxiliary resource moves, are detailed below. These choices are explained and motivated.

Minimize the weighted sum of completion times. Jobs have priorities because of their waiting times in the workshop or the fact that they are related to different customers or products. Hence, the weighted sum of completion times is considered. Let us denote by C_j the completion time of job j. The objective function to minimize is:

$$\sum_{j=1}^{N} w_j C_j.$$

This is a classical scheduling criterion. Let us denote this problem Π_w.

Maximize products completed before H. The number of completed products is an important criterion. This criterion makes no sense if there is no limit on the scheduling horizon, since all jobs will always be scheduled. Since there is a continuous flow of arriving jobs, specifying a limited time horizon $[0, H]$ is relevant because jobs arriving after the start of the horizon have to be taken into account before completing all jobs already available (for instance, to save setup times). Hence, the number of products completed before H is maximized. If, in a solution, we denote by m_j the machine assigned to job j and its start time by t_j, then the number of completed products before H can be written as follows:

$$\sum_{j=1}^{N} n_j \theta_j$$

where θ_j is the *ratio* of job j that is processed before H, over its processing time, i.e.

$$\theta_j = \begin{cases} \frac{\min(\rho_{jm_j}, H - t_j)}{\rho_{jm_j}} & \text{if } t_j \leq H, \\ 0 & \text{if } t_j > H. \end{cases}$$

which models the fact that the number of completed products of a job is proportional to the elapsed time before the end of the time horizon. Let us denote this problem Π_h.

Minimize the number of auxiliary resource moves. Moving an auxiliary resource from one position to another requires a human operator that needs to stop his current task to perform the move. Minimizing these interruptions helps to reduce the risk of errors and time waste. In the remainder of the paper, let us denote this problem Π_M. Note that machine eligibility constraints are critical since, in the specific case of total eligibility (where all machines can process all jobs), the problem becomes trivial. Indeed, it can be shown that an optimal solution for this criterion is to assign to the same machine all the jobs that share the same resource.

To illustrate the third criterion, let us analyze the number of auxiliary resource moves in Figure 1. Note that the solution in the example is not optimal with respect to the number of auxiliary resource moves, and is used to illustrate how the criterion is calculated:

- Jobs 1, 6 and 7 require the same auxiliary resource. Job 7 is processed on machine 2, then job 1 on machine 1 (which implies two move from machine 2 to machine 1), then job 6 again on machine 2 (which implies another move from machine 1 to machine 2).
- Jobs 2 and 5 require the same auxiliary resource, and are both processed on machine 2.
- Jobs 3, 4 and 8 require the same auxiliary resource. Job 4 cannot be processed by machine 1, and jobs 3 and 8 cannot be processed by machine 2 (eligibility constraints). Thus, one move from machine 1 to machine 2 is required, as observed on the schedule.
- Note that jobs start on machine 1 at time 1 and not time 0. The reason is that the auxiliary resource for job 8 is initially located in the storage area, so a unitary transport time is required before starting processing job 8. If the auxiliary resource required by job 8 had already been on machine 1, the process would have started at time 0, and a resource move would have been avoided.
- There are a total four auxiliary resource moves in this schedule. Note that, by processing job 6 before job 1, the number of moves can be reduced by one.

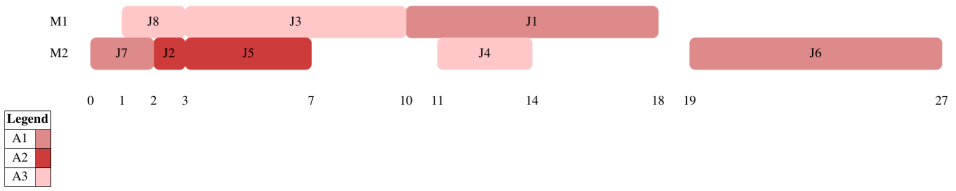

Figure 1: A schedule for an instance of our unrelated parallel machine scheduling problem, with 8 jobs, 2 machines and 3 auxiliary resources.

3. Literature review

In the literature, heuristics based on dispatching rules for the problem $P_m|r_j, aux|\sum w_jC_j$ are described, for example by Cakici and Mason (2007). Metaheuristics are also proposed to solve unrelated parallel machine problems with sequence-dependent setup times. We study in this paper an unrelated parallel machine scheduling problem with eligibility constraints, auxiliary resources and sequence-dependent and machine-dependent setup times, with three criteria: The weighted sum of completion times, the number of processed products within a time horizon and the number of auxiliary resource moves. To our knowledge, the second criterion has only been considered in Bitar et al. (2016) and the last criterion has never been studied. A contribution of this paper is to analyze these problems, in terms of complexity and mathematical modeling.

Many studies have been conducted considering non-identical parallel machines and when the weighted sum of completion times is minimized. For instance, Skutella (2001) studies the scheduling problem $R||\sum w_jC_j$ with convex quadratic program relaxation and approximation algorithms with ratio $\frac{3}{2}$. Bruno et al. (1974) prove that $R||\sum C_j$ is polynomial and they propose an $O(n^3)$ algorithm to solve it. More recently, Wang and Alidaee (2019) are also interested by solving the unrelated parallel machine scheduling problem. Very large-scale instances can be solved using a Tabu Search algorithm that embeds a multiple-jump strategy. In these problems, no setup times are considered, and there are neither eligibility constraints nor auxiliary resources.

With identical parallel machines and job priorities, some results already exist. Kawaguchi and Kyan (1986) give a $\frac{1}{2}(1+\sqrt{2}) \approx 1.207$ approximation ratio for $P||\sum w_jC_j$, obtained with a Weighted Shortest Processing Time sorting of jobs, which is the best known approximation ratio for this problem. Woeginger and Skutella (2000) give the first PTAS for the strongly NP-Hard problem $P||\sum w_jC_j$. Sahni (1976) gives an FPTAS for $P_m||\sum w_jC_j$ where

the number of machines is a constant, Bruno et al. (1974) show that, when the number of machines is a constant (not a parameter of the problem) and at least two, then the problem is NP-Hard. In particular, they show that $P2||\sum w_jC_j$ is weakly NP-Hard. Lee and Uzsoy (1992) propose a pseudo-polynomial dynamic programming algorithm for $P_m||\sum w_jC_j$ whose complexity depends on the sum of the weights whereas, in classical approaches, it depends on the sum of processing times. Smith (1956) shows that, in the case of a single machine, the problem is in P, using Smith's rule (WPST sorting).

Concerning the problem with sequence-dependent setup times (without eligibility constraints and auxiliary resources), Webster and Azizoglu (2001) consider the problem with setup families and the $\sum w_jC_j$ criterion. They propose two dynamic programming algorithms. When the number of machines and families are fixed, these algorithms are pseudo-polynomial. Otherwise, the authors show that the problem is strongly NP-Hard. If the number of machines is fixed and there is only one family, the problem is NP-Hard. So with a fixed number of families, the problem is NP-Hard. But when the job weights are unitary, the problem is in P. Monma and Potts (1989) show that, for the same problem, an optimal solution is such that all jobs of the same family are scheduled in the WPST order. Obeid et al. (2014) propose a linear programming model to solve a scheduling problem on unrelated parallel machines with setup times. In that case, setup times only depend on the job families, not the sequence. Ekici et al. (2019) also consider sequence-dependent setup times in a problem inspired from assembly lines in the electronic industry. They propose a mathematical model with precedence variables. Fanjul-Peyro et al. (2019) study a scheduling problem with unrelated parallel machines and makespan minimization and propose an exact algorithm.

Concerning auxiliary resources, Blazewicz et al. (1983) propose a classification of resource-constrained scheduling problems in the literature and introduce a generic notation for this kind of constraints. With this notation, the resource constraints studied in this article are written $res.11$, where "." means that the number of resources is a parameter of the problem, the first "1" means that there is only one resource of each type, and the other "1" means that each job only requires a unitary quantity of its required auxiliary resource (since there is only one auxiliary resource of each type). Blazewicz et al. (1983) provide interesting complexity results, and show, for example, that $P3|res.11, p_j = 1|\sum C_j$ is strongly NP-Hard. Edis et al. (2013) present an interesting review on the parallel machine scheduling problems with additional resources. Integer linear programming models are also developed

for the makespan criterion and without setup times. Vallada et al. (2019) present a new Scatter Search algorithm for the unrelated parallel machine problem with one additional resource where the makespan is minimized. A Repairing Mechanism is used to reach feasible solutions and various local search procedures are also introduced. Yepes-Borrero et al. (2020) also study a parallel machine scheduling problem where setups require additional resources. Two objectives are optimized, the makespan and the number of additional resources.

For the unrelated parallel machine scheduling problem with sequence-dependent setup times and additional resources, Fanjul-Peyro (2020) proposes a mixed integer linear program and a three-phase algorithm to minimize the makespan.

Some research have been made on eligibility constraints (also called *scheduling typed task systems* or *scheduling with processing set restrictions*), but usually without sequence-dependent setup times and auxiliary resources. Shchepin and Vakhania (2005) propose an approximation algorithm with ratio $2 - \frac{1}{m}$ for the case of identical parallel machines. Brucker et al. (1997) show that, when minimizing the (non-weighted) sum of completion times, the problem is in P. Leung and Li (2008) survey the existing work on scheduling with eligibility constraints. Nattaf et al. (2019) study an identical parallel machine problem with *machine qualifications*, i.e. where qualifying a machine for a specific process is part of the decision. The authors propose a mixed-integer linear program with time-indexed variables. In Perez-Gonzalez et al. (2019) and to minimize the total tardiness, constructive heuristics and a mixed integer liner program are proposed for the unrelated parallel machines scheduling problem with machine eligibility and sequence-dependent setup times. Inspired by the plastic injection industry, Bektur and Saraç (2019) address the unrelated parallel machine scheduling problem with sequence-dependent setup times and eligibility constraints with a common server. In their problem, jobs have release dates and the criterion is the total weighted tardiness. A mathematical model (with positional variables) and two heuristic approaches, tabu search and simulated annealing, are proposed and their results are compared. Note that there is only one server to set up machines in Bektur and Saraç (2019) and that the server has no transport times, whereas we consider multiple auxiliary resources and their transport times. Sahney (1972) and Cheng and Kovalyov (2003) consider two-machine scheduling problems (parallel machines in the first paper and flow shop in the second paper) also with a single server (operator), but with switching times of the server between the machines. In Werner and Kravchenko (2010), identical parallel machines require a server to setup

jobs. Although multiple servers are considered, they are identical and there is no transport time for the servers. Moreover, only the cases with unit setup times or constant setup and processing times are investigated. Very recently, Lee and Kim (2020) also consider several setup operators when scheduling jobs that can be split on parallel machines and with sequence dependent setup times. Here again, the machines are identical as well as the servers, and no transport time for the servers is taken into account. More importantly, compared to the previous literature, we consider different and new criteria.

Maximizing the number of products completed before H, the end of a time horizon, can be seen as minimizing the weighted number of late jobs with H as a common due date for all the jobs, except that we consider the ratio of the jobs that start before H and are completed after H, which is a major difference. We can cite Dauzère-Pérès and Sevaux (2004) and M'Hallah and Bulfin (2007), who propose exact methods to minimize the number of tardy jobs on a single machine.

Finally, in Bitar et al. (2016), we propose a memetic algorithm to solve the problem for the weighted sum of completion times and the number of products completed before H. The minimization of the number of auxiliary resource moves, considered in this article, is a new criterion to our knowledge.

4. Minimizing the weighted sum of completion times

4.1. Complexity results

A first statement about Problem Π_w is that it is strongly NP-Hard. Indeed, it is more general than $R||\sum w_jC_j$, which is strongly NP-Hard. Each instance of $R||\sum w_jC_j$ can be written as an instance of Π_w by stating $A = N$, i.e. a specific auxiliary resource to each job, and $R_a = 0$ for each auxiliary resource a. Then, the constraints on auxiliary resources and auxiliary resource transport times only refer to machine availability at time 1. Sequence-dependent and machine-dependent setup times are set to 0 and $\mathcal{M}_i = \{1, \ldots, m\}$. Because of the auxiliary resource unitary transport times, the first job on each machine starts at time $t = 1$, which does not change the problem complexity.

4.2. Integer Linear Programming (ILP) model

In this section, an Integer Linear Programming (ILP) model with time-indexed variables is proposed for Π_w. Time-indexed formulations for single-machine scheduling problems have been studied by van den Akker et al.

(2000) and Dyer and Wolsey (1990). Many valid inequalities were proposed and polyhedral studies conducted for this formulation, for instance by van den Akker et al. (2000) and Sousa and Wolsey (1992). It led to efficient linear programming-based and combinatorial lower bounds. In Obeid et al. (2014), a time-indexed ILP model is designed to solve an unrelated parallel machine scheduling problem. This model extends the time-indexed formulation for the single machine problem. In our case, adapting these constraints is not straightforward because of the sequence-dependent and machine-dependent setup time constraints. The ILP in Fanjul-Peyro (2020) relies on disjunctive variables, i.e. binary variables that model whether job i is processed right after j or not. Although unrelated parallel machines with sequence-dependent setup times and auxiliary resources are also considered, the makespan is minimized.

The choice of our modeling is motivated by the fact that time-indexed formulations are: (1) Powerful enough to linearly express a wide variety of constraints, as it is the case in our problem, (2) "Natural", i.e. there is a direct semantic relation between the variables of the model and the decision variables of our problem, and, importantly, (3) Known to provide good linear relaxation bounds, and thus effective in integer linear programming solvers. When the number of products completed before H is maximized, i.e. for problem Π_h studied in Section 5, time-indexed variables are actually necessary to model the problem.

Let T be an integer large enough, e.g. chosen to be larger than or equal to the *makespan* of any optimal solution of problem Π_w.

A feasible solution of the problem can be represented as follows:

$$(u_{jmt})_{j\in\{1,\dots,N\},m\in\mathcal{M}_j,t\in\{0,\dots,T-\rho_{jm}\}}$$

where $u_{jmt} \in \{0,1\}$, and u_{jmt} is equal to 1 if and only if job j is processed on machine m and begins its process at date t. Let $\mathcal{U}$ be the set of such vectors.

The constraints on variables (u_{jmt}) of $\mathcal{U}$ are listed below.

4.2.1. Classical constraints

The classical assignment constraints are:

$$\sum_{m\in\mathcal{M}_j}\sum_{t=0}^{T-\rho_{jm}} u_{jmt} = 1 \quad \forall j = 1,\dots,N. \tag{1}$$

By adapting classical capacity constraints from the time-indexed formu-

lation for single-machine problems, one obtains:

$$\sum_{j|m\in\mathcal{M}_j}\sum_{t_0=\max(0,t-\rho_{jm}+1)}^{\min(T-\rho_{jm},t)} u_{jmt_0} \leq 1, \forall m=1,\ldots,M, t=0,\ldots,T-\min_{j|m\in\mathcal{M}_j}\{\rho_{jm}\}. \tag{2}$$

Another valid inequality is further added to the model. Constraints (3) deal with the auxiliary resource constraints and model the fact that jobs requiring the same auxiliary resource cannot be processed simultaneously on different machines.

$$\sum_{j|\varphi_j=k}\sum_{m\in\mathcal{M}_j}\sum_{t_0=\max(0,t-\rho_{jm}+1)}^{\min(T-\rho_{jm},t)} u_{jmt_0} \leq 1, \quad \forall k\in\{1,\ldots,A\}, t\in\{0,\ldots,T\}. \tag{3}$$

4.2.2. Constraints on auxiliary resources, sequence-dependent and machine-dependent setup times

To represent a feasible solution of Π_w, variables $(u_{jmt}) \in \mathcal{U}$ must also satisfy the constraints below:

$$\begin{aligned} L(1-u_{jmt}) \geq & \sum_{\substack{j_0|m\in\mathcal{M}_{j_0}\\ j_0\neq j}}\sum_{t_0=t}^{\min(T-\rho_{j_0m},t+\rho_{jm}-1+\beta_{jmj_0})} u_{j_0mt_0} + \\ & \sum_{\substack{j_0|\varphi_{j_0}=\varphi_j\\ j_0\neq j}}\sum_{\substack{m_0\in\mathcal{M}_{j_0}\\ m_0\neq m}}\sum_{t_0=t}^{\min(T-\rho_{j_0m_0},t+\rho_{jm})} u_{j_0m_0t_0} \\ & \forall j=1,\ldots,N, \forall m\in\mathcal{M}_j, \forall t=0,\ldots,T-\rho_{jm}, \end{aligned} \tag{4}$$

where L is an upper bound of the right member sum. Here, L stands for a big-M constant, to avoid confusion with the number of machines. The big-M is computed as the maximum number of variables that are likely to be equal to 1 in the right-hand side of the inequality. We consider the number of distinct job indices in both sums, according to the assignment constraints (a job is assigned to at most one machine, and has a single start time). Then, for a given job j and a given machine m, the big-M coefficient can be set to the number of jobs, other than j, that either can be processed by

machine m or require the same resource as j. Here, we take advantage of other constraints of the model to derive a bound.

These big-M constraints can be removed and replaced by other inequalities. Indeed, each job j_0 of the right member sum can be considered separately:

$$u_{jmt} + \sum_{m_0 \in \mathcal{M}_{j_0} \setminus \{m\}} \sum_{t_0=\max(0,t-\rho_{j_0 m_0})}^{\min(T-\rho_{j_0 m_0}, t+\rho_{jm})} u_{j_0 m_0 t_0} + \sum_{m_0 \in \mathcal{M}_{j_0} \cap \{m\}} \sum_{t_0=\max(0,t-\rho_{j_0 m_0}+1)}^{\min(T-\rho_{j_0 m_0}, t+\rho_{jm}-1)} u_{j_0 m_0 t_0} \leq 1,$$

$$\forall j = 1, \ldots, N, \forall m \in \mathcal{M}_j, \forall t = 0, \ldots, T - \rho_{jm}, \forall j_0 \neq j \text{ such that } \varphi_{j_0} = \varphi_j \quad (5)$$

and

$$u_{jmt} + \sum_{t_0=\max(0,t-\rho_{j_0 m}+1-\beta_{j_0 m j})}^{\min(T-\rho_{j_0 m}, t+\rho_{jm}-1+\beta_{jm j_0})} u_{j_0 m t_0} \leq 1,$$

$$\forall j = 1, \ldots, N, \forall m \in \mathcal{M}_j, \forall t = 0, \ldots, T - \rho_{jm}, \forall j_0 \in \mathcal{Q}_m \setminus \{j\}. \quad (6)$$

Remark 1. *Note that Constraints 6 imply that setup times must follow a relaxed version of the* triangular inequality*: For each pair of jobs* (j_1, j_2) *and each machine* m*, there is no other job* j_3 *such that*

$$\beta_{j_1 m j_3} + \rho_{j_3 m} + \beta_{j_3 m j_2} \leq \beta_{j_1 m j_2},$$

which means that there must not be another job j_3 *that, being processed between jobs* j_1 *and* j_2*, can reduce the time elapsed between the end of* j_1 *and the start of* j_2*.*

Constraints (5) and (6) ensure that, if job j is scheduled on machine m and starts at time t, then no other job j_0 requiring the same auxiliary resource is started on another machine m_0 between $t-\rho_{j_0 m_0}$ and $t+\rho_{jm}$, and no other job j_0 is started on the same machine between $t - \rho_{j_0 m} + 1 - \beta_{j_0 m j}$ and $t + \rho_{jm} - 1 + \beta_{jm j_0}$.

Other models use this type of constraints to model the non-simultaneous processing of jobs on a machine. However, in our model, these two families of constraints also deal with auxiliary resources and sequence-dependent and

machine-dependent setup times. Constraints (2) are consequently used as valid inequalities, to provide better bounds.

Finally, the following constraints are added.

$$\sum_{j=1}^{N} \sum_{\substack{m \in \mathcal{M}_j \\ m \neq R_{\varphi_j}}} u_{jm0} = 0. \tag{7}$$

Constraints (5) and (6) take into account auxiliary resources but not their initial locations. The unitary auxiliary resource transport time prevents from using an auxiliary resource on a machine at time $t = 0$, unless this auxiliary resource is initially on the machine. Hence, a job j starts on a qualified machine m ($m \in \mathcal{M}_j$) at time $t = 0$ only if $R_{\varphi_j} = m$, as shown in constraints (7).

4.2.3. Complete model

Because of the assignment constraints (1), there exists a unique couple in $\mathcal{M}_j \times \{0, \ldots, T - \rho_{jm}\}$ such that $u_{jmt} = 1$ for any job j. Therefore,

$$C_j = \sum_{m \in \mathcal{M}_j} \sum_{t=0}^{T-\rho_{jm}} u_{jmt}(t + \rho_{jm}).$$

Hence the following ILP model is valid for Π_w.

$$\min \sum_{j=1}^{N} w_j \left(\sum_{m \in \mathcal{M}_j} \sum_{t=0}^{T-\rho_{jm}} u_{jmt}(t + \rho_{jm}) \right) \tag{8}$$

s. t.

$$\sum_{m \in \mathcal{M}_j} \sum_{t=0}^{T-\rho_{jm}} u_{jmt} = 1 \qquad \forall j = 1, \ldots, N \tag{1}$$

$$u_{jmt} + \sum_{m_0 \in \mathcal{M}_{j_0} \setminus \{m\}} \sum_{t_0 = \max(0, t-\rho_{j_0 m_0})}^{\min(T-\rho_{j_0 m_0}, t+\rho_{jm})} u_{j_0 m_0 t_0} + \sum_{m_0 \in \mathcal{M}_{j_0} \cap \{m\}} \sum_{t_0 = \max(0, t-\rho_{j_0 m_0}+1)}^{\min(T-\rho_{j_0 m_0}, t+\rho_{jm}-1)} u_{j_0 m_0 t_0} \leq 1$$

$$\forall j = 1, \ldots, N, \forall m \in \mathcal{M}_j, \forall t = 0, \ldots, T - \rho_{jm}, \forall j_0 \neq j \text{ such that } \varphi_{j_0} = \varphi_j \tag{5}$$

$$u_{jmt} + \sum_{t_0=\max(0,t-\rho_{j_0m}+1-\beta_{j_0mj})}^{\min(T-\rho_{j_0m},t+\rho_{jm}-1+\beta_{jmj_0})} u_{j_0mt_0} \leq 1$$

$$\forall j = 1,\ldots,N, \forall m \in \mathcal{M}_j, \forall t = 0,\ldots,T-\rho_{jm}, \forall j_0 \in \mathcal{Q}_m \setminus \{j\}. \quad (6)$$

$$\sum_{j=1}^{N} \sum_{\substack{m \in \mathcal{M}_j \\ m \neq R_{\varphi_j}}} u_{jm0} = 0 \quad (7)$$

$$u_{jmt} \in \{0,1\} \qquad \forall j = 1,\ldots,N, \forall m \in \mathcal{M}_j, \forall t = 0,\ldots,T-\rho_{jm}. \quad (9)$$

This model includes the same constraints, which are:

- Assignment constraints (1): Each job is processed by one and only one machine,
- Auxiliary resource constraints (5): Each pair of jobs that require the same auxiliary resource cannot be processed simultaneously on two machines,
- Capacity constraints (6): Each machine can process at most one job at a time (including the sequence-dependent setup times),
- Initial conditions (7): Impossible to start a job on a machine at $t = 0$ if the required resource is located elsewhere.

Computational experiments on this model are presented and discussed in Section 7.

5. Maximizing the number of products completed before H

5.1. Complexity results

Let us recall that Π_h is the scheduling problem introduced in Section 2, in which the number of products completed before H is maximized.

The following proves that Π_h is harder than a generalization of $R||C_{\max}$. An important link between the makespan criterion and the number of produced wafer before a time limit is established.

Lemma 1. *Let I be an instance of Π_h and $C(I)$ the minimum makespan for I. The following property holds: If $H \geq C(I)$, then the criterion for the optimal solution S^* of Π_h, i.e. the number of products completed before H, is $c(S^*) = \sum_{j=1}^{N} n_j$. If $H < C(I)$, then $c(S^*) < \sum_{j=1}^{N} n_j$.*

Proof. If $H \geq C(I)$, then there exists a schedule with makespan lower than H, i.e. where each job is completed before H. Hence, the criterion of the optimal solution is the total number of products completed before H, i.e. $\sum_{j=1}^{N} n_j$. If $H < C(I)$, the optimality property of $C(I)$ implies that there is no solution with makespan lower than or equal to H. For each solution, there exists at least one job that is completed after H, hence the inequality $c(S^*) < \sum_{j=1}^{N} n_j$. □

Let us denote by Π_C the problem Π_h in which the objective function is the minimum makespan. The following lemma shows that Π_h is harder than Π_C.

Lemma 2. *If there exists a polynomial (resp. pseudo-polynomial) time algorithm for Π_h, then there exists a polynomial (resp. pseudo-polynomial) time algorithm for Π_C.*

Proof. Let I be an instance of Π_h, $K_1 = 0$ and K_2 an upper bound of the minimum makespan for I. This upper bound can be for example $K_2 = \sum_{j=1}^{N} \max_{m \in \mathcal{M}_j}\{\rho_{jm}\} + (N-1)\max_{j,m,k}\{\beta_{jmk}\} + N$. We consider a polynomial (resp. pseudo-polynomial) time algorithm for Π_h.

We use this algorithm with $H = \lfloor \frac{K_1+K_2}{2} \rfloor$. If the criterion of the optimal solution is $\sum_{j=1}^{N} n_j$, then according to Lemma 1, H is larger than or equal to the minimum makespan, which leads to setting $K_2 = H$ and repeating the same operation. Otherwise, H is strictly lower than the minimum makespan, which leads to setting $K_1 = H$ and repeating the same operation.

Then we use binary search until $K_1 = K_2$, i.e. the minimum makespan is determined.

The number of iterations is $O(\log K_2)$, which is polynomial in the size of the instance (since K_2 can be expressed as an exponential function of the size of the instance), so binary search is in polynomial (resp. pseudo-polynomial) time since the algorithm for Π_h is in polynomial (resp. pseudo-polynomial) time. □

Theorem 1. *Problem Π_h is strongly NP-Hard.*

Proof. According to Lemmas 1 and 2, Π_h is harder than Π_C, which is a generalization of $R||C_{\max}$. And according to Garey and Johnson (1979), $R||C_{\max}$ is strongly NP-Hard, since it generalizes $P||C_{\max}$. Which proves the theorem. □

5.2. Integer Linear Programming (ILP) model

The linear programming model introduced in Section 4.2 can also be used to model problem Π_h.

Remark 2. *For problem Π_h, in every solution, jobs that begin after H do not change the value of the objective function. Then, by relaxing the constraints ensuring that all jobs must be processed, we obtain a similar ILP as for problem Π_h. The following constraints model the relaxed assignment constraint. The assignment equality is replaced by an inequality, because all that matters is what happens before H. The jobs starting after H may not be processed, since they do not impact the objective function.*

$$\sum_{m \in \mathcal{M}_j} \sum_{t=0}^{T-\rho_{jm}} u_{jmt} \leq 1 \qquad \forall j = 1, \ldots, N \tag{10}$$

The objective function of Π_h, $\sum_{j=1}^{N} n_j \theta_j$, can be expressed as follows:

$$\max \sum_{j=1}^{N} \sum_{m \in \mathcal{M}_j} \left(\sum_{t=0}^{H-\rho_{jm}} n_j u_{jmt} + \sum_{t=H-\rho_{jm}+1}^{H} n_j \frac{H-t}{\rho_{jm}} u_{jmt} \right) \tag{11}$$

since, for every job j:

$$\theta_j = \sum_{m \in \mathcal{M}_j} \left(\sum_{t=0}^{H-\rho_{jm}} u_{jmt} + \sum_{t=H-\rho_{jm}+1}^{H} \frac{H-t}{\rho_{jm}} u_{jmt} \right).$$

As described in Section 2.3, the following definition of θ_j is used:

$$\theta_j = \begin{cases} \frac{\min(\rho_{jm_j}, H-t_j)}{\rho_{jm_j}} & \text{if } t_j \leq H, \\ 0 & \text{if } t_j > H, \end{cases}$$

where t_j is the starting time of job j. Note that (11) is a linearization of this expression. Indeed, we can denote as m_j the machine to which job j is assigned and consider the related term in the sum over $m \in \mathcal{M}_j$. For each term of this sum, there is at most one index t for which $u_{jmt} \neq 0$. Then, there are two cases:

- If $t_j > H$, then all variables u_{jm_jt} are equal to 0 in the sum

$$\sum_{t=0}^{H-\rho_{jm_j}} u_{jm_jt} + \sum_{t=H-\rho_{jm_j}+1}^{H} \frac{H-t}{\rho_{jm_j}} u_{jm_jt},$$

since variable t is in the range $[0, H]$. Hence, as in the expression of θ_j, the value is 0.

- If $t_j \leq H$, then only one u_{jm_jt} is equal to 1 in the sum

$$\sum_{t=0}^{H-\rho_{jm_j}} u_{jm_jt} + \sum_{t=H-\rho_{jm_j}+1}^{H} \frac{H-t}{\rho_{jm_j}} u_{jm_jt},$$

since variable t is in the range $[0, H]$. If $t_j \leq H - \rho_{jm_j}$, then variable u_{jm_jt} that is equal to 1 is in the left-hand sum, i.e. $\sum_{t=0}^{H-\rho_{jm_j}} u_{jm_jt}$. Hence, the value of the term is equal to 1. As wanted in the expression of $\theta_j = \frac{\min(\rho_{jm_j}, H-t_j)}{\rho_{jm_j}}$, since $t_j \leq H - \rho_{jm_j}$, $\theta_j = \frac{\rho_{jm_j}}{\rho_{jm_j}} = 1$. If $t_j > H - \rho_{jm_j}$, then

$$\theta_j = \frac{\min(\rho_{jm_j}, H-t_j)}{\rho_{jm_j}} = \frac{H-t_j}{\rho_{jm_j}},$$

which is the coefficient of variable u_{jm_jt} in the right-hand sum.

Hence, Π_h can be solved by solving the ILP with objective function (11) and Constraints (5), (6) (7), (9) and (10) .

Remark 3. *As illustrated below, the proposed ILP models can be adapted to a generalization of the problem: If the transport times depend on the origin and destination of the auxiliary resources instead of being constant, then a matrix $(T_{uv})_{u,v\in\{0,\dots,M\}}$ indicates the transport time of an auxiliary resource from machine u to machine v. In this case, Constraints (5) can be rewritten:*

$$u_{jmt} + \sum_{m_0 \in \mathcal{M}_{j_0}} \sum_{t_0=\max(0, t-\rho_{j_0m_0}+\mathbf{1}-\mathbf{T_{m_0m}})}^{\min(T-\rho_{j_0m_0}, t+\rho_{jm}-\mathbf{1}+\mathbf{T_{mm_0}})} u_{j_0m_0t_0} \leq 1 \qquad (12)$$

$\forall j = 1, \dots, N, \forall m \in \mathcal{M}_j, \forall t = 0, \dots, T - \rho_{jm}, \forall j_0 \neq j$ *such that* $\varphi_{j_0} = \varphi_j$.

The only thing that changes compared to constraints (5) is the transport time T_{mm_0}, which is the transport time from machine m to machine m_0.

Furthermore, Constraints (7) are rewritten:

$$\sum_{j=1}^{N} \sum_{\substack{m \in \mathcal{M}_j \\ m \neq R_{\varphi_j}}} \sum_{t=0}^{\min(T_{R_{\varphi_j} m}-1, T-\rho_{jm})} u_{jmt} = 0. \tag{13}$$

Here, R_{φ_j} is the initial location of the required auxiliary resource of job j. Then, for each machine m, $T_{R_{\varphi_j} m}$ stands for the transport time between the initial location of the resource and machine m. This constraint states that no job can be processed before its required resource is transported from its initial location.

6. Minimizing the number of auxiliary resource moves

6.1. Complexity results

Preliminary properties are necessary to derive complexity results for the problem of minimizing the number of auxiliary resource moves, denoted by Π_M. Let us recall that, for each auxiliary resource a, $\mathcal{E}_a = \{j \in \mathcal{J} \mid \varphi_j = a\}$ is the set of jobs that require a.

Some additional notations are introduced:

- For each machine m, $\mathcal{Q}_m$ denotes the set of jobs j such that $m \in \mathcal{M}_j$, i.e. for which m is qualified.
- For each machine m, for each auxiliary resource a, $\mathcal{M}_m(a) = \mathcal{Q}_m \cap \mathcal{E}_a$ denotes the set of jobs in $\mathcal{E}_a$ for which machine m is qualified; furthermore $\mathcal{M}_0(a) = \emptyset$.
- For each auxiliary resource a, $N_a = |\mathcal{E}_a|$ is the number of jobs that require a.

6.1.1. Solution representation

The set $\chi \subset \{1, \ldots, M\}^N \times \{1, \ldots, N\}^N$ represents feasible solutions of Π_M, where each vector $(x, y) \in \chi$ has the following components:

- A vector
$$y = (y^1, \ldots, y^A)$$
where $y^a = (y^a_1, \ldots, y^a_{N_a})$ and $y^a_j \in \{1, \ldots, N\}$ is the index of the j-th job (in the chronological order) requiring auxiliary resource a. For example, if jobs 3, 7 and 4 require auxiliary resource a and are

processed in the order 3, 7, 4, (since they require the same auxiliary resource, they cannot be processed in parallel), then $y_1^a = 3$, $y_2^a = 7$ and $y_3^a = 4$.

- A vector
$$x = (x^1, \ldots, x^A)$$
where $x^a = (x_1^a, \ldots, x_{N_a}^a)$ and $x_j^a \in \{1, \ldots, M\}$ is the machine processing the j-th job (in chronological order) requiring auxiliary resource a. Using the same example, if jobs 3 and 4 are on machine 2 and job 7 is on machine 1, then $x_1^a = x_3^a = 2$ and $x_2^a = 1$.

Let $f : \chi \to \mathbb{N}$ be the mapping that sends each feasible solution $(x, y) \in \chi$ to the number of auxiliary resource moves induced by this solution. For all $(x, y) \in \chi$:

$$f(x, y) = \sum_{a=1}^{A} \left| \left\{ j \in \{0, \ldots, N_a - 1\} \middle| x_j^a \neq x_{j+1}^a \right\} \right|$$

with, for all $a \in \{1, \ldots, A\}$, $x_0^a = R_a$.

Note that the start times of the jobs have no influence on the objective function. An $O(N)$ algorithm to compute feasible start times from a given solution (x, y) is proposed.

According to the following result, we can search the optimal solution within χ.

Proposition 1. *For each $(x, y) \in \chi$, a solution (x, y, t) with start times can be built in $O(N)$ time with Algorithm 1, where parameters β_{0mj}, y_0^a, t_0 and ρ_{0m} are fixed to 0.*

Algorithm Building-Start-Times *returns a feasible schedule in $O(N)$ time.*

Note that array D includes the last jobs scheduled on the machines at the end of each iteration. Initially, all components of D are equal to 0, since the solution is empty (see line 5). Array D is used to compute the setup time when starting a new job on a machine (setup times are sequence dependent, hence the need to keep track of the previous job on each machine), and is updated at the end of each iteration of the main loop (line 15). The same goes for array S (the current locations of the resources), which is used to compute the transport times of the resources (see *if* condition, lines 10 and 11).

Algorithm 1 BUILDING-START-TIMES(x, y)

for $a \leftarrow 1$ to A **do**
  $S[a] \leftarrow R_a$ {Current location of auxiliary resources}
end for
for $m \leftarrow 1$ to M **do**
  $D[m] \leftarrow 0$ {Last job scheduled on each machine}
end for
for $a \leftarrow 1$ to A **do**
  for $j \leftarrow 1$ to N_a **do**
    $\tau \leftarrow 0$
    if $S[a] \neq x_j^a$ **then**
      $\tau \leftarrow 1$ {For unitary auxiliary resource transport times}
    end if
    $S[a] \leftarrow x_j^a$ {Updates current locations of auxiliary resources}
    $t_{y_j^a} \leftarrow \max(t_{y_{j-1}^a} + \rho_{y_{j-1}^a x_{j-1}^a} + \tau, t_{D[x_j^a]} + \rho_{D[x_j^a] x_j^a} + \beta_{D[x_j^a] x_j^a y_j^a})$
    $D[x_j^a] \leftarrow y_j^a$ {Updates the last scheduled job}
  end for
end for
return $(x, y, t_1, \ldots, t_N)$

Proof.

- *Complexity:* Each iteration of the main loop takes constant time $O(1)$ and the maximum number of iterations is $\sum_{j=1}^{A} N_a = N$.
- *Correctness:* We show that, at the end of each iteration of the main loop, the start time of job y_j^a
  - Is in the same order as in (x, y) for all jobs that require the same auxiliary resource a
  - And satisfies the constraints (sequence-dependent and machine-dependent setup times, non simultaneous process of jobs on the same machine, unitary auxiliary resource transport time and non simultaneous process of jobs requiring the same auxiliary resource).

  Let us first show that, in the solution returned by Algorithm 1, the order of jobs requiring the same auxiliary resource a is the same as in (x, y). Let us denote by D_m the index of the last scheduled job on machine m. In the algorithm, this value is given by $D[m]$.

  Job y_j^a is always processed on machine x_j^a. Moreover, for all $a \in \{1, \ldots, A\}$, start times are computed in the increasing order of index

j (from 1 to N_a) and the equality (line 14)

$$t_{y_j^a} = \max(t_{y_{j-1}^a} + \rho_{y_{j-1}^a x_{j-1}^a} + \tau, t_{D_{x_j^a}} + \rho_{D_{x_j^a} x_j^a} + \beta_{D_{x_j^a} x_j^a y_j^a})$$

(where τ is equal to 1 if an auxiliary resource move is required, 0 otherwise) ensures the inequality $t_{y_j^a} \geq t_{y_{j-1}^a}$, which confirms that the solution returned by the algorithm satisfies the process order of jobs requiring the same auxiliary resource, given in (x, y).

Then, we show that the start times $t_{y_j^a}$ satisfy the sequence-dependent setup times and the auxiliary resource and machine capacity constraints, with the following inequalities:

$$t_{y_j^a} \geq t_{D_{x_j^a}} + \rho_{D_{x_j^a} x_j^a}$$
$$t_{y_j^a} \geq t_{y_{j-1}^a} + \rho_{y_{j-1}^a x_{j-1}^a} + \tau$$
$$t_{y_j^a} \geq t_{D_{x_j^a}} + \rho_{D_{x_j^a} x_j^a} + \beta_{D_{x_j^a} x_j^a y_j^a}.$$

The first inequality ensures the non-simultaneous process of different jobs on the same machine. The second inequality ensures the auxiliary resource transport times and the non-simultaneous process of jobs requiring the same auxiliary resource. The last inequality corresponds to the sequence-dependent and machine-dependent setup time constraints.

□

6.1.2. Splitting the problem

Proposition 1 allows the following problem Π_M to be considered:

$$\min_{(x,y)\in\chi} f(x, y)$$

which is equivalent to:

$$\min_{(x,y)\in\chi} f(x, y) + A$$

since A is a constant term.

Indeed, in the following, we prove that this problem can be split into A subproblems, one per auxiliary resource. Then we prove that every A-tuple of solutions of these subproblems corresponds to a solution of the initial problem. Thus, solving the subproblems provides a method to solve Π_M.

For all $a \in \{1, \ldots, A\}$, let us consider the set

$$\chi^a = \{(x^a, y^a) | \exists (x^1, \ldots, x^a, \ldots, x^A, y^1, \ldots, y^a, \ldots, y^A) \in \chi\}$$

and introduce the function $f_a : \chi^a \to \mathbb{N}$, defined as follows:

$$f_a(x^a, y^a) = \left| \left\{ j \in \{0, \ldots, N_a - 1\} \middle| x^a_j \neq x^a_{j+1} \right\} \right|,$$

which implies

$$f(x, y) = \sum_{a=1}^{A} f_a(x^a, y^a).$$

Proposition 2. *The following equality holds.*

$$\min_{(x,y)\in\chi} f(x, y) = \sum_{a=1}^{A} \min_{(x^a,y^a)\in\chi^a} f_a(x^a, y^a).$$

and, for all optimal solutions (x^a, y^a)*,* $a \in \{1, \ldots, A\}$*, an optimal solution* $(x^1, \ldots, x^A, y^1, \ldots, y^A)$ *of* Π_M *is obtained.*

Proof. Let us recall that, for every auxiliary resource a, $\mathcal{E}_a$ is the set of jobs that require a. We consider, for every auxiliary resource a, the problem

$$\tilde{\Pi}^a : \min_{(x^a,y^a)\in\chi^a} f_a(x^a, y^a).$$

First, let us prove that each A-tuple of solutions $((x^1, y^1), \ldots, (x^A, y^A)) \in \chi^1 \times \ldots \times \chi^A$ of Problems $\tilde{\Pi}^a$ corresponds to a solution $(x^1, \ldots, x^A, y^1, \ldots, y^A)$ of Π_M.

Indeed, each job belongs to one and only one set $\mathcal{E}_a$. These sets form a partition of $\mathcal{J}$. This implies that the constraints on the unique process of jobs are satisfied by each solution $(x^1, \ldots, x^A, y^1, \ldots, y^A)$ built from the A solutions $((x^1, y^1), \ldots, (x^A, y^A))$ of Problems $\tilde{\Pi}^a$. Machine eligibility constraints are also satisfied since these constraints are the same in Problems $\tilde{\Pi}^a$.

Furthermore, to each feasible solution $(x^1, \ldots, x^A, y^1, \ldots, y^A)$ can be associated A feasible solutions

$$((x^1, y^1), \ldots, (x^A, y^A))$$

for Problems $\tilde{\Pi}^a$.

Let us consider the optimal solutions $((x^{1*}, y^{1*}), \ldots, (x^{A*}, y^{A*}))$ of Problems $\tilde{\Pi}^a$.

Each optimal solution

$$(\hat{x}^1, \ldots, \hat{x}^A, \hat{y}^1, \ldots, \hat{y}^A)$$

of Π_M has a criterion of $\sum_{a=1}^{A} f_a(\hat{x}^a, \hat{y}^a)$. Also,

$$\sum_{a=1}^{A} f_a(x^{a*}, y^{a*}) = \sum_{a=1}^{A} \min_{(x^a, y^a) \in \chi^a} f_a(x^a, y^a) \leq \sum_{a=1}^{A} f_a(\hat{x}^a, \hat{y}^a).$$

Because $(x^{1*}, \ldots, x^{A*}, y^{1*}, \ldots, y^{A*})$ is feasible, it is optimal.

Let $(x^*, y^*) = (x^{1*}, \ldots, x^{A*}, y^{1*}, \ldots, y^{A*})$ be an optimal solution of Π_M. If there exists (x^{k*}, y^{k*}) such that

$$f_k(x^{k*}, y^{k*}) \neq \min_{(x^k, y^k) \in \chi^k} f_k(x^k, y^k),$$

then

$$f(x^*, y^*) = \sum_{a=1}^{A} f_a(x^{a*}, y^{a*}) < \sum_{a=1}^{A} \min_{(x^a, y^a) \in \chi^a} f_a(x^a, y^a)$$

which leads to a contradiction and proves the proposition. □

Corollary 1. *Solving Problem Π_M is equivalent to solving the A problems*

$$\tilde{\Pi}^1, \ldots, \tilde{\Pi}^A$$

and building a solution in $O(A)$ time (see Algorithm 2).

Algorithm 2 MIN-AUXILIARY-RESOURCE-MOVES(I)

for $a \leftarrow 1$ to A **do**
  $(x^a, y^a) \leftarrow$MIN-AUXILIARY-RESOURCES($\mathcal{E}_a, \mathcal{M}, a$)
end for
return $(x^1, \ldots, x^A, y^1, \ldots, y^A)$ {Return a combination in $O(A)$ time}

Then, analyzing the complexity of Problems $\tilde{\Pi}^a$ helps to characterize the complexity of Π_M, which is a generalization.

6.1.3. Solving subproblems

Let us recall that we consider, for each solution (x^a, y^a), the values $x_0^a = R_a$.

Proposition 3. *Each solution* $(x_1^a, \ldots, x_{N_a}^a, y_1^a, \ldots, y_{N_a}^a)$ *of* $\tilde{\Pi}^a$ *satisfying the following property is dominant:*

$$\forall j \in \{0, \ldots, N_a - 1\} \text{ s. t. } x_j^a \neq x_{j+1}^a,\ x_j^a \neq x_k^a \quad \forall k \in \{j+2, \ldots, N_a\}.$$

Proof. Let $(x^a, y^a) = (x_1^a, \ldots, x_{N_a}^a, y_1^a, \ldots, y_{N_a}^a)$ be a solution of Π_M that does not verify this property. Then let us consider an index $j \in \{0, \ldots, N_a - 1\}$ such that $x_j^a \neq x_{j+1}^a$ and there exists $k \in \{j+2, \ldots, N_a\}$ such that $x_j^a = x_k^a$.

Let $k' = \min\{k \in \{j+2, \ldots, N_a\} \mid x_j^a = x_k^a\}$, which exists according to the hypothesis.

Then, let us consider the solution

$$\begin{aligned}(x'^a, y'^a) = (&x_1^a, \ldots, x_j^a, x_{k'}^a, x_{j+1}^a, \ldots, x_{k'-1}^a,\\ &x_{k'+1}^a, \ldots, x_{N_a}^a, y_1^a, \ldots, y_j^a, y_{k'}^a, y_{j+1}^a, \ldots, y_{k'-1}^a, y_{k'+1}^a, \ldots, y_{N_a}^a).\end{aligned}$$

We have

$$f_a(x'^a, y'^a) = \begin{cases} f_a(x^a, y^a) - 1 & \text{if } k' = N_a \\ f_a(x^a, y^a) & \text{si } k' < N_a \text{ and } x_{k'}^a = x_{k'+1}^a \\ f_a(x^a, y^a) - 2 & \text{if } k' < N_a,\ x_{k'}^a \neq x_{k'+1}^a \text{ and } x_{k'-1}^a = x_{k'+1}^a \\ f_a(x^a, y^a) - 1 & \text{otherwise.} \end{cases}$$

then $f_a(x'^a, y'^a) \leq f_a(x^a, y^a)$.

It is possible to show that a non-dominant solution cannot be optimal. Indeed, let us assume the case where $k' < N_a$ and $x_{k'}^a = x_{k'+1}^a$, then the criterion remains the same. But we can apply the same transformation to the solution (x'^a, y'^a) until we reach one of the three other cases. This is possible since $k' + 1$ is the minimum index in $\{j+2, \ldots, N_a\}$ such that $x_j'^a = x_{k'+1}^a$, and since the maximum index of this set cannot reach this case. □

Corollary 2. *Determining an optimal solution among dominant solutions is equivalent to determining the minimum number of machines to which auxiliary resource a is assigned. Hence, an upper bound on the minimum number of auxiliary resource moves is* $M \times A$.

Proof. In the dominant solutions (x^a, y^a), the number of distinct values of $(x_0^a, x_1^a, \ldots, x_{N_a}^a)$ is the number of machines to which auxiliary resource a is moved. This is the number of auxiliary resource moves increased by one. □

Remark 4. *The storage area is considered as a machine, with regards to the auxiliary resource moves.*

For example, if $x^a = (2, 1, 2, 6, 5, 4, 4, 4, 3, 3, 6)$, the corresponding solution is not dominant and cannot be optimal, because machine 1 is used once between two processes of machine 2. A dominant solution would be, for instance, $(2, 2, 1, 6, 6, 5, 4, 4, 4, 3, 3)$. Here, the number of used machines is 6, and the number of auxiliary resource moves is $6 - 1 = 5$, whereas in the first solution, for the same number of used machines, there are 7 auxiliary resource moves.

According to Corollary 2, solving $\tilde{\Pi}^a$ is equivalent to determining the minimal number of machines to which auxiliary resource a is assigned.

6.1.4. Polynomial reduction

The following theorem links the problem to another combinatorial optimization problem, well-known in the literature. The *Set Cover* problem is strongly NP-Hard and generalizes the *Vertex Cover*. It is one of the first problems for which approximation algorithms were analyzed (Johnson (1974) and Lovász (1975)).

Theorem 2. *The Set Cover problem and* $\tilde{\Pi}^a$*,* $a \in \{1, \ldots, A\}$*, are equivalent.*

Proof. First, let us prove that there is a polynomial time reduction from the *Set Cover* problem to Problem $\tilde{\Pi}^a$, $a \in \{1, \ldots, A\}$. The Set Cover problem has the following entries:

- A set T to cover,
- s subsets $T_1, \ldots, T_s$ of T such that $T_1 \cup \ldots \cup T_s = T$.

The question is to determine a subset $\{V_1, \ldots, V_k\}$ of $\{T_1, \ldots, T_s\}$ of minimum cardinality such that $\cup_{i=1}^{k} V_i = T$.

Let us consider an instance of Set Cover, and show how to build an instance of $\tilde{\Pi}^a$:

- $\mathcal{E}_a = T$,
- $\mathcal{M}_m(a) = T_m$, for all $1 \leq m \leq M$.

The optimal solution of this instance gives the minimal number of machines that can process the jobs of $\mathcal{E}_a$, i.e. the minimal number of machines to which auxiliary resource a is assigned. In this way, we obtain the minimal cover of the set T by subsets $T_1, \ldots, T_s$.

Let $\{\mathcal{M}_{m_1}(a), \ldots, \mathcal{M}_{m_d}(a)\}$ be an optimal solution of the problem. The dominance property imposes first to assign all jobs in $\mathcal{M}_{m_1}(a)$ on m_1 and to process them consecutively. Then, all jobs in $\mathcal{M}_{m_2}(a) \setminus \mathcal{M}_{m_1}(a)$ are assigned to m_2 and processed consecutively. This is repeated until all jobs in $\mathcal{M}_{m_d}(a) \setminus \left(\cup_{r=1}^{d-1}\mathcal{M}_{m_r}(a)\right)$ are assigned to m_d and processed consecutively. This procedure gives the minimal number of subsets that cover $T = \mathcal{E}_a$. Otherwise we could find another set with lower cardinality as a solution, and thus a lower number of machines that would use auxiliary resource a, which leads to a contradiction.

Similarly, each optimal solution of the Set Cover problem corresponds to an optimal solution of $\tilde{\Pi}^a$. The same contradiction proof scheme can be used.

Let us also show that there exists a polynomial time reduction from $\tilde{\Pi}^a$ to Set Cover. Let I be an instance of $\tilde{\Pi}^a$. We show how to build an instance I' of Set Cover.

The set to cover T is equal to the set $\mathcal{E}_a \cup \{0\}$ where 0 represents a dummy job, and the covering sets are the sets $\mathcal{M}_m(a)$ $(0 \leq m \leq M)$, where $\mathcal{M}_{R_a}(a)$ contains the additional dummy job 0. This is to ensure that the solution includes the initial location of auxiliary resource a.

Every optimal solution of I' corresponds to an optimal solution of $\tilde{\Pi}^a$ (the set of machines for the auxiliary resource). Indeed, if there exists a set of machines with lower cardinality r that can process all the jobs requiring auxiliary resource a, then there exists a set $\{\mathcal{M}_{m_1}(a), \ldots, \mathcal{M}_{m_r}(a)\}$ that covers the set $T = \mathcal{E}_a$, which contradicts the fact that the previous solution was optimal for I'. The reverse remains true, with a similar reasoning.

□

Corollary 3. *Problem Π_M is strongly NP-Hard.*

Remark 5. *Corollary 3 is true when the number M of machines is not fixed, i.e. it is a parameter of the problem. The following theorem shows that it does not stand when M is fixed.*

Theorem 3. *If M is a constant, Problem Π_M is in P.*

Proof. Proving this theorem, according to Corollary 1 and Proposition 2, is the same than showing that the Set Cover problem is in P when the number of covering sets is constant. In that case, the number of feasible solutions of the problem is the number of subsets of the set of the M covering sets, which is 2^M. Then, using Proposition 2, there are $A \times 2^M$ solutions for one instance of Problem Π_M among which we can search an optimal solution in $O(A)$ time, and 2^M is assumed constant. □

6.2. Integer Linear Programming (ILP) model

A well-known ILP model exists for the Set Cover problem and, according to the preliminary remarks and properties, we can show how to adapt it and formulate an ILP model for Π_M. In fact, the solution will only specify which auxiliary resource goes to which machine. Because of the dominance property, this is sufficient since we know the initial location R_a of each auxiliary resource a and there is always a way to build a feasible solution with correct start times from the order of jobs requiring the same auxiliary resource and their machine assignment.

The ILP model of Set Cover (with set T and subsets $T_1, \ldots, T_s$) uses binary variables $x_i \in \{0,1\}$ such that $x_i = 1$ if and only if subset T_i is in the solution.

$$\min \sum_{j=1}^{s} x_j \tag{14}$$

$$\text{s. t.}$$

$$\sum_{j|k\in T_j} x_j \geq 1 \qquad \forall k \in T \tag{15}$$

$$x_j \in \{0,1\} \qquad \forall j = 1, \ldots, s \tag{16}$$

When adapted to Problem $\tilde{\Pi}^a$, the model becomes (with $T = \mathcal{E}_a \cup \{0\}$, $T_m = \mathcal{M}_m(a)$ for all $m \in \{0, \ldots, M\} \setminus \{R_a\}$, and $T_{R_a} = \mathcal{M}_{R_a}(a) \cup \{0\}$).

$$\min \sum_{m=0}^{M} x_m \tag{17}$$

$$\text{s. t.}$$

$$\sum_{m|j\in T_m} x_m \geq 1 \qquad \forall j \in \mathcal{E}_a \cup \{0\} \tag{18}$$

$$x_m \in \{0,1\} \qquad \forall m = 0, \ldots, M \tag{19}$$

Theorem 4. *The following formulation is valid for Problem* Π_M.

$$\min \sum_{a=1}^{A} \sum_{m=0}^{M} u_{am} - A \tag{20}$$

$$s.\ t.$$

$$\sum_{m|j\in\mathcal{M}_m(a)} u_{am} \geq 1 \qquad \forall a = 1,\ldots,A, \quad \forall j \in \mathcal{E}_a \cup \{0\} \tag{21}$$

$$u_{aR_a} = 1 \qquad \forall a = 1,\ldots,A \tag{22}$$

$$u_{am} \in \{0,1\} \qquad \forall a = 1,\ldots,A, \quad \forall m = 0,\ldots,M \tag{23}$$

Proof. We consider that variable $u_{am} \in \{0,1\}$ is equal to 1 if and only if auxiliary resource a goes to machine m in the solution.

Hence, $\sum_{m=0}^{M} u_{am}$ is the total number of machines to which auxiliary resource a is assigned. Then let

$$(x,y) = (x^1,\ldots,x^A,y^1,\ldots,y^A)$$

be the corresponding solution. Hence,

$$f_a(x^a,y^a) + 1 = \sum_{m=0}^{M} u_{am}.$$

This implies that $\sum\limits_{a=1}^{A}\left(\sum\limits_{m=0}^{M} u_{am}\right) = \sum\limits_{a=1}^{A}(f_a(x^a,y^a)+1) = \tilde{f}(x,y) + A$.

Then, the objective function of this integer linear program is the objective function of Π_M.

As a constraint, for each auxiliary resource a and each job requiring that resource, there exists at least one eligible machine for the job to which auxiliary resource a is assigned, otherwise it is impossible to process the job. This is ensured by Constraints (21).

Constraints (22) correspond to the fact that an auxiliary resource is on its initial location.

Then, each feasible solution of this program is a feasible solution of $\tilde{\Pi}$ and, using Algorithm 1, it is possible to build a solution of Π_M in $O(N)$ time, with the same criterion. □

7. Computational experiments

In the following, computational experiments are presented and discussed to evaluate the performances of the three ILP models. The time-indexed formulation has been tested with the big-M Constraints (4) (column big-M) and compared to the model where they are removed and replaced by Constraints (5) with right-hand side 1 (column RHS 1). These experiments are performed on randomly generated instances that have common characteristics with industrial data. They were run on an Intel Core i7-2640M CPU, 2.80GHz, using the SCIP C++ library and the solver IBM ILOG CPLEX 12.9.

Most of the jobs have 25 products and, with a probability of 10%, between 1 and 25 products following a uniform distribution. This relates to the industrial context of semiconductor manufacturing where lots contain at most 25 wafers, and most of the lots exactly 25 wafers. The maximum number of eligible machines for a job is set to $\max(2, \frac{M}{2})$. The processing times are in $\{1, \ldots, 10\}$ and the sequence-dependent and machine-dependent setup times are in $\{0, \ldots, 5\}$. These relatively limited ranges for the processing and setup times again fit the industrial context, and are helpful for the type of formulations we adopted (time-indexed formulation). Our hypothesis is that there is not a significant variability in the processing times (which is the case in our industrial context), which can be normalized to be bounded to the sets $\{1, \ldots, 10\}$ and $\{0, \ldots, 5\}$. Indeed, setup times are rarely larger than 50% of the processing times on average. The 1-to-10 and 1-to-5 ratios are actually relatively extreme, which explains why we did not want to exceed them. Finally, at most 50% of the pairs of jobs that can go on the same machine have a sequence-dependent and machine-dependent setup time that is not equal to 0. The reason is that some jobs can be processed without any setup, to create what are called "trains" of jobs, because they share the same properties on a given machine. We wanted to reproduce this aspect of the photolithography workshop in the generated instances.

In the tables, the instances are given in the format ($N-M-A$) stands for (number of jobs − number of machines − number of resources)), except for problem Π_h, for which there is a fourth parameter (H, end of time horizon).

As shown in Table 1, our Set Cover based approach works very well on industrial-like instances. Because the number of machines is limited, the number of covering sets is very small, which leads to an extremely quick resolution.

The experiments on problems Π_w and Π_h are detailed in Table 2. For each of the 11 instance sizes, 10 randomly generated instances are solved by

Inst.	Π_M	
	Gap(%)	Time (s)
(8-2-3)	0	1
(32-4-8)	0	1
(128-15-15)	0	1
(500-18-25)	0	1
(1600-20-25)	0	1

Table 1: Results for Problem Π_M with the Set Cover type model.

Inst.	Π_w Resolution time		**Inst.**	Π_h Resolution time	
	big-M	RHS 1		big-M	RHS 1
(8-2-3)	9.0	**1.2**	**(8-2-3-20)**	6.7	**1.0**
(10-2-3)	29.7	**7.6**	**(10-2-3-25)**	35.5	**15.8**
(10-2-4)	22.0	**6.3**	**(10-2-4-25)**	31.3	**19.2**
(10-3-3)	21.4	**4.8**	**(10-3-3-16)**	10.9	**1.8**
(15-2-3)	**194.4**	273.9	**(15-2-3-16)**	39.4	**11.0**
(15-3-3)	87.2	**59.6**	**(15-3-3-25)**	**89.7**	161.7
(15-3-5)	45.4	**35.9**	**(15-3-5-25)**	**91.7**	205.1
(20-2-3)	**2 798.5**	7 909.9	**(20-2-3-28)**	**500.8**	$>$ 12 337.1
(20-3-3)	**582.1**	1 343.2	**(20-3-3-20)**	**162.9**	246.5
(25-3-3)	**7 951.1**	23 130.7	**(25-3-3-20)**	**397.6**	1 022.7
(25-3-12)	**913.9**	1517.5	**(25-3-12-20)**	**149.6**	$>$ 3 471.12

Table 2: Resolution times (in seconds as an average of 10 randomly generated instances) for problems Π_w and Π_h with time indexed model.

our exact method, with two different versions of the model: (1) With big-M constraints (column "big-M"), and (2) With right-hand side 1 constraints (column "RHS 1"). For problem Π_w (minimization of weighted completion times) and for small size instances, note that the RHS 1 model provides better results with shorter resolution times, but it is outperformed by the big-M formulation, starting from 20 jobs, when the size of the problem increases. Indeed, for 25 jobs and 3 machines for instance, it takes on average three times less resolution time with the big-M formulation than the *RHS 1* formulation (7 951 seconds versus 23 130 seconds). Note also that the resolution time decreases when the number of auxiliary resources increases.

Inst.	No Const. (2)		Const. (2)		Best known
	big-M	RHS 1	big-M	RHS 1	solution
(10-2-4)	69.9	95.9	137.3	143.2	**170**
(10-3-3)	73.0	95.4	107.0	113.6	**134**
(15-2-3)	88.8	132.9	210.1	216.8	**278**
(15-3-10)	100.3	126.0	195.6	200.4	**246**
(15-3-3)	86.3	121.0	148.4	160.7	**214**
(15-3-5)	94.3	137.6	168.5	182.1	**229**
(16-3-6)	100.8	137.8	198.1	206.4	**243**
(20-2-3)	106.7	167.2	322.0	329.9	**485**
(20-3-3)	147.0	206.9	336.4	348.6	**478**
(25-2-3)	172.9	257.0	686.6	690.9	**872**
(25-3-3)	152.4	220.5	387.4	396.2	547
(25-3-12)	156.8	216.2	420.6	431.2	531
(30-3-4)	192.6	280.1	577.3	589.9	**756**
(32-4-8)	230.6	311.5	523.4	542.6	723
(40-3-4)	270.5	393.4	1004.0	1014.2	1291
(50-3-6)	321.2	456.2	1448.0	1456.8	1781

Table 3: Lower bound improvements induced by valid inequalities (2) for Problem Π_w

The comparison of the $(25-3-3)$ and the $(25-3-12)$ instances shows that, in both cases, the average resolution time is 93% lower with the *RHS 1* formulation (1 517 seconds with 12 resources versus 23 130 seconds with 3 resources) and 88% lower with the big-M formulation (913 seconds with 12 resources versus 7 951 seconds with 3 resources). This can be explained by the fact that increasing the number of resources reduces the number of jobs that are linked by shared resources, and thus mitigates the impact of the auxiliary resource constraints.

The same observations can be made for problem Π_h. Here, the sizes of the instances are given as $(N-M-A-H)$, where the last value is H, the end of the time horizon on which the number of products is computed. The same efficiency for the big-M formulation is observed on large enough instances. For instance, it takes on average less than 150 seconds to solve the $(25-3-12-20)$ instances, while the *RHS 1* formulation provides optimal results in more than 3 471 seconds on average. The $>$ sign on the table means that the solver was stopped because of too large resolution times (more than 29 000 seconds). In all these cases (4 instances in total: 3 in the

$(20 - 2 - 3 - 28)$ instances and 1 in the $(25 - 3 - 12 - 20)$ instances), the optimal solution was found but the upper bound decreased too slowly and the branch-and-bound took hours to improve the gap.

The valid inequalities added to the models for Π_w and Π_h lead to better bounds. Table 3 shows the lower bound improvements induced by both types of valid inequalities, with big-M constraints (column "big-M") and with the right-hand side member 1 (column "RHS 1"). Bold values in the "Best known solution" column mean the solution is optimal. On large instances, the difference is larger. The relaxation bound is even 5 times larger in some cases. Therefore, these constraints can be relevant to optimally solve larger instances. The relaxation bound improvement is very small when using the model where big-M Constraints (4) are replaced by Constraints (5) with right hand side 1. Constraints (2) help to reduce the gap between both formulations.

8. Conclusions and perspectives

We presented an original scheduling problem on unrelated parallel machines, by considering sequence-dependent and machine-dependent setup times, auxiliary Resources and three criteria that are important in practical settings. The problem complexity was analyzed for each criterion, leading to two NP-hardness results. Integer linear programming models were also presented, and were tested on instances randomly generated with characteristics from industrial data. They show that very large instances can be solved very quickly when minimizing the number of auxiliary resource moves, but that the computational times rapidly increase with the size of the instances for the two other criteria.

Our future research aims at developing multi-criteria approaches relying both on the models proposed in this paper, in particular the most efficient one for the number of auxiliary resource moves, and the memetic algorithm proposed in Bitar et al. (2016). One of the challenges is to characterize the relevant solutions in the Pareto front. This is particularly important because the criteria have different units, but also can somehow compensate each other, e.g. additional moves of auxiliary resources are acceptable if the productivity is improved enough, i.e. if the increase of the number of products completed before H is large enough. Another perspective it to use the mathematical models introduced in this paper to analyze the performance of heuristics for the problems studied in this paper.

References

van den Akker, M., Hurkens, C., Savelsbergh, M., 2000. Time-Indexed Formulations for Machine Scheduling Problems: Column Generation. INFORMS Journal on Computing 12, 111–124.

Bektur, G., Saraç, T., 2019. A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. Computers & Operations Research 103, 46–63.

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, 367–376.

Blazewicz, J., Lenstra, J., Kan, A.R., 1983. Scheduling subject to resource constraints : Classification and complexity. Discrete Applied Mathematics 5, 11–24.

Brucker, P., Jurisch, B., Kramer, A., 1997. Complexity of scheduling problems with multi-purpose machines. Annals of Operations Research 70, 57–73.

Bruno, J., Coffman, E., Sethi, M., 1974. Scheduling independent tasks to reduce mean finishing time. Journal of the ACM 17, 382–387.

Cakici, E., Mason, S., 2007. Parallel machine scheduling subject to auxiliary resource constraints. Production Planning and Control 18, 217–225.

Cheng, T., Kovalyov, M., 2003. Scheduling a single server in a two-machine flow shop. Computing 70, 167–180.

Dauzère-Pérès, S., Sevaux, M., 2004. An exact method to minimize the number of tardy jobs in single machine scheduling. Journal of Scheduling 7, 405–420.

Dyer, M., Wolsey, L., 1990. Formulating the single machine sequencing problem with release dates as a mixed integer program. Discrete Applied Mathematics 26, 255–270.

Edis, E.B., Oguz, C., Ozkarahan, I., 2013. Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. European Journal of Operational Research 230, 449–463.

Ekici, A., Elyasi, M., Örsan Özener, O., Sarıkaya, M., 2019. An application of unrelated parallel machine scheduling with sequence-dependent setups at vestel electronics. Computers & Operations Research 111, 130–140.

Fanjul-Peyro, L., 2020. Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources. Expert Systems with Applications 5, 15 pages.

Fanjul-Peyro, L., Ruiz, R., Perea, F., 2019. Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. Computers & Operations Research 101, 173–182.

Garey, M., Johnson, D., 1979. Computers and intractability : A Guide to the Theory of NP-Completeness. volume 1. W.H. Freeman and Company.

Johnson, D., 1974. Approximation Algorithms for Combinatorial Problems. Journal of Computer Systems Science 9, 256–278.

Kawaguchi, T., Kyan, S., 1986. Worst case bound of an lrf schedule for the mean weighted flow-time problem. SIAM Journal on Computing 15, 1119–1129.

Lee, C., Uzsoy, R., 1992. A new dynamic programming algorithm for the parallel machines total weighted completion time problem. Operations Research Letters 11, 73–75.

Lee, J.H., Kim, H.J., 2020. A heuristic algorithm for identical parallel machine scheduling: splitting jobs, sequence-dependent setup times, and limited setup operators. Flexible Services and Manufacturing Journal , 1–35.

Leung, J., Li, C., 2008. Scheduling with processing set restrictions: A survey. International Journal of Production Economics 116, 251–262.

Lovász, L., 1975. On the ratio of the optimal integral and fractional covers. Discrete Mathematics 13, 383–390.

M'Hallah, R., Bulfin, R., 2007. Minimizing the weighted number of tardy jobs on a single machine with release dates. European Journal of Operational Research 176, 727–744.

Moench, L., Fowler, J., 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, 583–599.

Monma, C., Potts, C., 1989. On the complexity of scheduling with batch setup times. Operations Research 37, 798–804.

Nattaf, M., Dauzère-Pérès, S., Yugma, C., Wuc, C., 2019. Parallel machine scheduling with time constraints on machine qualifications. Computers & Operations Research 107, 61–76.

Obeid, A., Dauzère-Pérès, S., Yugma, C., 2014. Scheduling job families on non-identical parallel machines with time constraints. Annals of Operations Research 213, 221–234.

Perez-Gonzalez, P., Fernandez-Viagas, V., García, M.Z., Framinan, J.M., 2019. Constructive heuristics for the unrelated parallel machines scheduling problem with machine eligibility and setup times. Computers & Industrial Engineering 131, 131–145.

Sahney, V.K., 1972. Single-server, two-machine sequencing with switching time. Operations Research 20, 24–36.

Sahni, S., 1976. Algorithms for scheduling independent tasks. Journal of the ACM 23, 116–127.

Shchepin, E., Vakhania, N., 2005. An optimal rounding gives a better approximation for scheduling unrelated machines. Operations Research Letters 33, 127–133.

Skutella, M., 2001. Convex quadratic and semidefinite relaxations in scheduling. Journal of the ACM 48, 206–242.

Smith, W., 1956. Various optimizers for single-stage production. Naval Research Logistics Quarterly 3, 59–66.

Sousa, J., Wolsey, L., 1992. A time indexed formulation of non-preemptive single machine scheduling problems. Mathematical Programming 54, 353–367.

Vallada, E., Villa, F., Fanjul-Peyro, L., 2019. Enriched metaheuristics for the resource constrained unrelated parallel machine scheduling problem. Computers & Operations Research 111, 415–424.

Wang, H., Alidaee, B., 2019. Effective heuristic for large-scale unrelated parallel machines scheduling problems. Omega 83, 261–274.

Webster, S., Azizoglu, M., 2001. Dynamic programming algorithms for scheduling parallel machines with family setup times. Computers & Operations Research 28, 127–137.

Werner, F., Kravchenko, S., 2010. Scheduling with multiple servers. Automation and Remote Control 71, 2019–2021.

Woeginger, G., Skutella, M., 2000. A PTAS for minimizing the total weighted completion time on identical parallel machines. Mathematics of Operations Research 25, 63–75.

Yepes-Borrero, J.C., Perea, F., Ruiz, R., Villa, F., 2020. Bi-objective parallel machine scheduling with additional resources during setups. European Journal of Operational Research .