Parallel Machine Scheduling with Time Constraints on Machine Qualifications

Margaux Nattaf, Stéphane Dauzère-Pérès, Claude Yugma, Cheng-Hung Wu

Version of Record: https://www.sciencedirect.com/science/article/pii/S0305054819300589
Manuscript_efca67a8bd96f2ecf7d74a3c59da583b

Parallel Machine Scheduling with Time Constraints on Machine Qualifications

Margaux Nattaf[1] Stéphane Dauzère-Pérès[1,2] Claude Yugma[1] Cheng-Hung Wu[3]

[1]Mines Saint-Etienne, Univ Clermont Auvergne
CNRS, UMR 6158 LIMOS
CMP, Department of Manufacturing Sciences and Logistics
F-13541 Gardanne, France
E-mail: margaux.nattaf@emse.fr, dauzere-peres@emse.fr, yugma@emse.fr

[2]Department of Accounting, Auditing and Business Analytics
BI Norwegian Business School
0484 Oslo, Norway

[3]Institute of Industrial Engineering
Department of Business Administration
Department of Mechanical Engineering
National Taiwan University
Taipei 106, Taiwan
E-mail: wuchn@ntu.edu.tw

Abstract

This paper studies the scheduling of jobs of different families on parallel machines, where not all machines are qualified (eligible) to process all job families. Originating from semiconductor manufacturing, an important constraint imposes that the time between the processing of two consecutive jobs of the same family on a machine does not exceed a given time limit. Otherwise, the machine becomes disqualified for this family. The goal is to minimize both the flow time and the number of disqualifications of job families on machines. To solve this problem, an integer linear programming model and a constraint programming model are proposed, as well as two improvement procedures of existing heuristics: A Recursive Heuristic and a Simulated Annealing algorithm. Numerical experiments on randomly generated instances compare the performances of each method.

Keywords: Scheduling, Parallel Machines, Time Constraints, Machine Qualifications, Constraint Programming, Integer Programming, Heuristics

1. Introduction

Nowadays, process industries are facing numerous challenges that are induced by continuous market changes, uncertainty in the demand, aggressive competition and recently more complex manufacturing technologies. These challenges require companies to continuously improve their production management and control to remain economically viable. To do so, several authors, such as Dauzère-Pérès and Lasserre (2002) and Gaudreault

et al. (2011), point out the importance of integrating operational scheduling decisions with tactical planning decisions. More recently, Yugma et al. (2015) show the opportunities related to integrating scheduling and process control in semiconductor manufacturing. This paper tackles a problem in this latter context, by integrating process control constraints when optimizing scheduling decisions.

The semiconductor industry is probably the most complex industry. Typical characteristics of semiconductor fabrication facilities include numerous products, each requiring hundreds of operations on hundreds of machines in different workshops. Scheduling all jobs in a semiconductor manufacturing facility is so complex that the problem needs to be decomposed, i.e. jobs are scheduled in each workshop separately (see Moench et al. (2011)). Still, in a workshop performing the same type of operations, machines are often not identical, i.e. a machine can usually process a limited number of job families. For instance, Yugma et al. (2012) and Jung et al. (2014), and more recently Knopp et al. (2017), consider scheduling problems in the cleaning and diffusion workshop, while Rotondo et al. (2015) consider the scheduling of jobs on wet-etch tools. Because it contains the most expensive machines, a critical workshop in wafer manufacturing facilities is the photolithography workshop. Scheduling approaches for this workshop have been proposed for instance in Cakici and Mason (2007) and Bitar et al. (2016).

Advanced Process Control (APC) aims at controlling machines and processes to ensure product quality, mainly by reducing variability. APC is usually associated with the combination of Statistical Process Control (SPC), Fault Detection and Classification (FDC), Run to Run (R2R) control, and more recently Virtual Metrology (VM) (see for instance Moyne et al. (2000)). Although usually studied separately, scheduling and APC are actually often related in semiconductor manufacturing (Yugma et al., 2015). In this paper, we are considering constraints induced by R2R controllers in scheduling decisions, and more specifically a maximum time constraint between two jobs of the same family to be processed on a machine. As shown in the survey paper of Tan et al. (2015), R2R control is becoming critical in high-mix semiconductor manufacturing processes.

A R2R controller is often associated with each machine and each job family, and uses data from past process runs to adjust the settings of the machine for the next run (see for example Musacchio et al. (1997) or Jedidi et al. (2011)). In order to keep the R2R parameters updated and valid, a R2R controller should regularly receive data. Hence, as presented in Obeid et al. (2014), an additional time constraint is defined on the scheduling problem to impose that the execution of two jobs of the same family lies within a given time interval on the same (qualified) machine. The value of this time threshold depends on several criteria such as the process type (critical or not) and the machine type. If this time constraint between two jobs of the same family is not satisfied, a qualification procedure is required for the machine to be able to process again the job family. This procedure ensures that the machine works within a specified tolerance and is usually time-consuming. In this paper, we assume that qualification procedures are not scheduled either because the scheduling horizon is not sufficiently long or because qualification procedures have to be manually performed and/or validated by process engineers. Therefore, maintaining machine qualifications as long as possible is crucial. More precisely, it is important to have as many remaining machine qualifications as possible at the end of the schedule, so that future jobs can also be scheduled. Note that defining the right qualifications of job

families to machines, studied for instance in Johnzén et al. (2011) and Rowshannahad et al. (2015), is outside the scope of this paper. We assume that the current set of qualifications are the ones that should be maintained for current and future short-term production mixes (i.e. number of jobs of each family). An extension of the approaches proposed in this paper would consist in not penalizing losses of the current qualifications that become unnecessary.

It is also important to note that the time constraints considered in this paper are different from the time constraints, also called time windows or maximum time lags, studied for instance in Wu et al. (2010), Klemmt and Moench (2012) and Sadeghi et al. (2015). In this latter case, the maximum time to satisfy is between two operations, usually performed on different machines, in the route (sequence of operations) of a job and not between two jobs on the same machine as it is the case in our problem.

To our knowledge, there are few articles dealing with scheduling decisions while integrating R2R constraints. Li and Qiao (2008) and Cai et al. (2011) study related problems, except that they allow qualification procedures to be performed, the number or the type of machines is different and the threshold is expressed in number of jobs instead of in time. The scheduling problem addressed in this paper has been studied in Obeid et al. (2014), where two Integer Linear Programs ($IP1$ and $IP2$) and two constructive heuristics are proposed. In this paper, the objective remains the same, i.e. to schedule jobs on non-identical parallel machines while satisfying time constraints and optimizing both the sum of completion times and the number of qualification losses. To solve this problem, we first propose a new Integer Linear Program $IP3$ that solves larger instances than $IP1$ and better models time constraints. Then, a CP model is presented to also solve exactly the problem. Finally, two heuristics are introduced that improve the solutions obtained in Obeid et al. (2014).

The paper is organized as follows. In Section 2, a more formal description of the problem is given. Section 3 is dedicated to exact methods. The new ILP is first introduced, and then the constraint programming model. Section 4 presents a Recursive Heuristic and a Simulated Annealing algorithm. Section 5 provides and discusses experimental results. Finally, conclusions and perspectives for future research are given in Section 6.

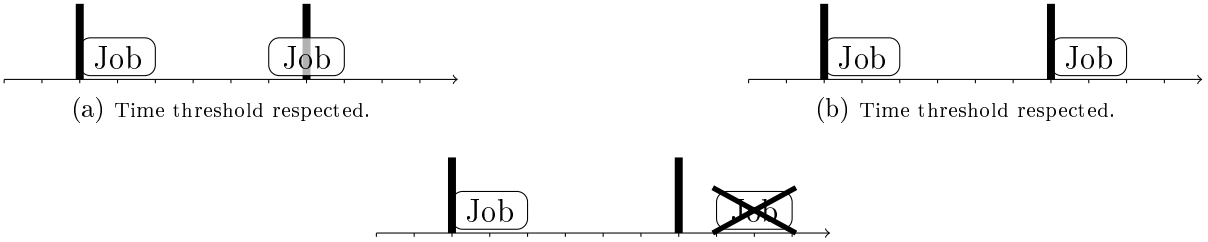
2. Problem description

The problem description and notations are taken from Obeid et al. (2014). This problem takes as input a set of jobs, $\mathcal{N} = \{1, \ldots, N\}$, belonging to different families, and a set of machines, $\mathcal{M} = \{1, \ldots, M\}$. The set of job families is denoted by $\mathcal{F}$, and $f(i)$ is the family of job i. A machine $m \in \mathcal{M}$ can process a limited number of job families. If m can process family f, m is said to be "qualified" (eligible) for f, and $\mathcal{M}(f)$ denotes the subset of machines qualified for f.

Each family f is associated with a number n_f of jobs to process, a processing time p_f needed to process a job of family f, a setup time s_f and a time threshold γ_f. Between two jobs of the same family, no setup time is required. The time threshold γ_f is used to model time constraints, i.e. γ_f is the maximum time interval between two jobs of f on a machine m to avoid losing the qualification of f on m. This time threshold is considered on a start-to-start basis, i.e. the threshold is counted from the start of a job of family f

to the start of the next job of f on machine m. If the constraint is not satisfied at time t, machine m becomes disqualified for family f and is no longer available to process jobs of f (see Example 2.1).

Example 2.1. *Figure 1 illustrates time constraints. In Figures* (1a) *and* (1b)*, a job of a given family is started during the time interval corresponding to its family, and hence the machine remains qualified to process jobs of the same family for another time interval. Figure* (1c) *represents the situation where a machine is no longer qualified to process a job family because no job of the family is scheduled during the considered time interval.*



(a) Time threshold respected.

(b) Time threshold respected.

(c) Time threshold violated: The machine can no longer schedule jobs of this family.

Figure 1: Illustration of time constraints.

The objective is to minimize both the sum of the completion times of jobs and the number of machine disqualifications. The objective considered through the paper is the weighted sum of both objectives with parameters α, for the sum of completion times, and β, for the number of machine disqualifications.

At least two reasons are motivating the minimization of the number of machine disqualifications. First, scheduling problems are usually solved in a rolling horizon setting, and it is thus interesting to preserve machine qualifications for future jobs, in particular if the scheduling horizon is relatively short. Second, the decision of qualifying a job family on a machine is often taken by process engineers once the machine qualification is lost. And qualification decisions are taken periodically by process engineers and not dynamically. Hence, again, it is relevant to preserve machine qualifications to avoid losing them for an extended period of time. As discussed in the perspectives in Section 6, considering automatic machine re-qualifications when scheduling lots is a future research topic.

The problem is defined as Scheduling **P**roblem with **T**ime **C**onstraints (PTC). It is important to notice that minimizing the sum of completion times for all jobs and minimizing the number of machine disqualifications are two conflicting criteria. Indeed, to maintain machine qualifications, one needs to regularly change the job family scheduled on machines, resulting in numerous setup times and then to a large value of the completion time. This assertion is illustrated in Section 5.

3. Exact Methods

This section starts by briefly recalling the two Integer Linear Programs (($IP1$) and ($IP2$)) presented in Obeid et al. (2014). Then, an improved Integer Linear Program

($IP3$) is presented, and finally a Constraint Programming model for PTC is detailed.

3.1. Integer Linear Programming

The two models described in Obeid et al. (2014), as well as the model proposed in this paper, are based on time-indexed variables. The time horizon T is discretized and let $\mathcal{T} = \{0, \ldots, T-1\}$ be the set of intervals. In such formulation, finding a good upper bound on T is therefore crucial. Since PTC considers two objectives that are conflicting, this bound is not easy to find. In the following formulations, the scheduling horizon is taken as the sum of all processing times, plus the setup time multiplied by the number of jobs per family. This is an extreme case where all jobs are scheduled on a single machine, and where a setup time is required for each job, i.e. $T = \sum_{f \in \mathcal{F}} n_f \cdot (p_f + s_f)$.

The first model, ($IP1$), is a job-based formulation where for each job $i \in \mathcal{N}$, for each time $t \in \mathcal{T}$ and for each machine $m \in \mathcal{M}$, a binary variable $x_{i,t}^m$ models the start time of job i. In addition, variable $y_{f,t}^m$ is defined to model machine disqualifications, i.e. $y_{f,t}^m$ is equal to 1 if and only if machine m is disqualified for family f at time t. Unfortunately, this model can only solve small size instances and poorly considers time constraints. Indeed, the model forces one and only one job of a family f to be scheduled in $]t-\gamma_f, t]$. However, in our problem, at least one job of f should be processed in $]t-\gamma_f, t]$ but several jobs of f can be scheduled in this time interval.

To solve larger instances, a new model, called ($IP2$), is introduced in Obeid et al. (2014). This formulation uses the fact that all jobs in a family can be interchanged in an optimal solution. Thus, to model the job start times, a binary variable $x_{f,t}^m$ is used to model the start time of jobs of family f. That is, $x_{f,t}^m$ is equal to 1 if and only if a job of family f starts at time t on machine m. With this model, larger instances can be solved but time constraints are still not modeled appropriately.

Indeed, in ($IP2$), the number of disqualifications depends on the time horizon T and on the makespan C_{max}. One of the main consequences is that, with this modeling, a machine can lose its qualification after C_{max}, i.e. the maximum completion time of all jobs. Thus, the solution determined by ($IP2$) may not be realistic.

Example 3.1. *In Figure 2, the dashed part corresponds to the scheduled jobs. Hence, C_{max} is the maximum completion time of all jobs. Furthermore, T is an upper bound on C_{max} and can be pretty far from it. In this figure, a job family f with a time threshold smaller than $T - C_{max}$ is disqualified in ($IP2$).*

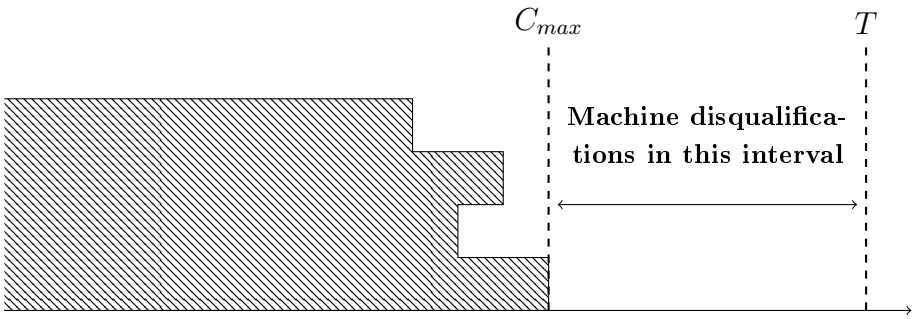


Figure 2: Illustration of machine qualification lost in $[C_{max}, T]$.

The main goal of the new model ($IP3$) is to tackle the previous issue. This is done by introducing a new binary variable Y_f^m, which models the fact that machine m is disqualified for family f at the end of the schedule, i.e. C_{max}. Hence, Y_f^m is now penalized in the objective function instead of $y_{f,t}^m$ in ($IP2$). New constraints are also added (Constraints (8) below) to ensure that Y_f^m is equal to 1 if m is disqualified for f at C_{max}, i.e. if $\exists t \in \{0, \ldots, C_{max}\}$ such that $y_{f,t}^m = 1$. Note that, if machine m is disqualified for family f after C_{max}, then $Y_f^m = 0$. Note also that the scheduling objective is not to minimize C_{max}. ($IP3$) is written below:

$$\text{min. } \alpha \cdot \sum_{f \in \mathcal{F}} C_f + \beta \cdot \sum_{f \in \mathcal{F}} \sum_{m \in \mathcal{M}} Y_f^m \tag{1}$$

$$\sum_{m \in \mathcal{M}(f)} \sum_{t=0}^{T-p_f} x_{f,t}^m = n_f \qquad \forall f \in \mathcal{F} \tag{2}$$

$$\sum_{m \in \mathcal{M}(f)} \sum_{t=0}^{T-p_f} (t + p_f) \cdot x_{f,t}^m \leq C_f \qquad \forall f \in \mathcal{F} \tag{3}$$

$$y_{f,t}^m + \sum_{\tau=t-p_f+1}^{t} x_{f,\tau}^m \leq 1 \qquad \forall f \in \mathcal{F},\ \forall p_f - 1 \leq t \leq T - p_f,\ \forall m \in \mathcal{M}(f) \tag{4}$$

$$n_f \cdot x_{f',t}^m + \sum_{\tau=t-p_f-s_{f'}+1}^{t} x_{f,\tau}^m \leq n_f$$
$$\forall f \neq f' \in \mathcal{F}^2,\ \forall m \in \mathcal{M}(f) \cap \mathcal{M}(f'),\ \forall p_f + s'_f - 1 \leq t \leq T - p_{f'} \tag{5}$$

$$y_{f,t}^m + \sum_{\tau=t-\gamma_f+1}^{t} x_{f,\tau}^m \geq 1 \qquad \forall f \in \mathcal{F},\ \forall t \geq \gamma_f \in \mathcal{T},\ \forall m \in \mathcal{M}(f) \tag{6}$$

$$y_{f,t-1}^m \leq y_{f,t}^m \qquad \forall f \in \mathcal{F},\ \forall t \in \mathcal{T} \setminus \{0\},\ \forall m \in \mathcal{M}(f) \tag{7}$$

$$y_{f,t-1}^m - 1 + \frac{1}{M \cdot (T-t)} \sum_{f' \in \mathcal{F}} \sum_{\tau=t-p_{f'}}^{T-1} \sum_{m' \in \mathcal{M}(f')} x_{f',\tau}^{m'} \leq Y_f^m$$
$$\forall t \in \mathcal{T} \setminus \{0\},\ \forall f \in \mathcal{F},\ \forall m \in \mathcal{M}(f) \tag{8}$$

$$x_{f,t}^m \in \{0,1\} \qquad \forall t \in \mathcal{T}, \forall f \in \mathcal{F},\ \forall m \in \mathcal{M}(f) \tag{9}$$

$$y_{f,t}^m \in \{0,1\} \qquad \forall t \in \mathcal{T}, \forall f \in \mathcal{F},\ \forall m \in \mathcal{M}(f) \tag{10}$$

$$Y_f^m \in \{0,1\} \qquad \forall f \in \mathcal{F},\ \forall m \in \mathcal{M}(f) \tag{11}$$

In this formulation, the objective function (1) is the weighted sum of the sum of completion times, i.e. $\sum_{f \in \mathcal{F}} C_f$, and the number of disqualifications, $\sum_{f \in \mathcal{F}} \sum_{m \in \mathcal{M}(f)} Y_f^m$. Note that C_f is the sum of completion times of all jobs in family f and is therefore equal to $\sum_{i \in \mathcal{N};\ f(i)=f} C_i$. In our experiments, different values for parameters α and β are considered and the results are discussed in Section 5.

Constraints (2) ensure that exactly n_f jobs of family f are scheduled. Constraints (3) are used to determine the sum of completion times of family f. Constraints (4) model both the fact that the execution of two jobs of the same family cannot occur simultaneously, i.e. $\sum_{\tau=t-p_f+1}^{t} x_{f,\tau}^m \leq 1$ and the fact that a machine has to be qualified to process a job, i.e. $y_{f,t}^m = 0$. Constraints (5) enforce the start of a job of a family f and the start of a job of family f' to be separated by at least $p_f + s'_f$, i.e. the processing time of the job

of f plus the setup time for the job of f'. Constraints (6) make sure that if no jobs of family f start on machine m during an interval $]t-\gamma_f, t]$, i.e. $\sum_{\tau=t-\gamma_f+1}^{t} x_{f,\tau}^{m} = 0$, then m becomes disqualified for family f at time t. Constraints (7) maintain the disqualification of a machine once it becomes disqualified.

Finally, Constraints (8) are the main difference between the model presented in this paper and ($IP2$) in Obeid et al. (2014). These constraints prevent the number of disqualifications to depend on the scheduling horizon T. The constraints ensure that it is no longer necessary to maintain a qualification on machine m if no job is started on any machine in the remainder of the horizon, i.e. $\frac{1}{M\cdot(T-t)} \sum_{f'\in\mathcal{F}} \sum_{\tau=t-p_{f'}}^{T-1} \sum_{m'\in\mathcal{M}(f')} x_{f',\tau}^{m'} = 0$.

3.2. Constraint Programming

Traditionally, scheduling problems have been tackled with various approaches. In the last 20 years, some methods based on artificial intelligence techniques have been successfully used to deal with different classes of scheduling problems, and in particular Constraint Satisfaction (CS) (Brailsford et al., 1999). The implementation of algorithms able to solve CS problems is known as Constraint Programming (Van Hentenryck, 1999). CP is able to address optimization problems since they can be expressed as a sequence of CS problems.

To date, there are several CP approaches that have been successfully employed to tackle scheduling problems in manufacturing environments, such as batch plants (see Jain and Grossmann (2001) and Maravelias and Grossmann (2004)). This section describes a CP model set up to solve PTC. First, the variables of the model are introduced and then the problem constraints are presented in detail.

In the CP model, the following set of variables are used:

- $masterJob_i$, $\forall i \in \mathcal{N}$: Interval variables that represent the jobs to schedule. Each variable $masterJob_i$ has a size $p_{f(i)}$ and its domain is $dom(masterJob_i) = \{[s,e) | [s,e) \subseteq [0,T), s + p_{f(i)} = e\}$.

- $altJobs_{mi}$, $\forall m \in \mathcal{M}$; $\forall i$ s.t. $m \in \mathcal{M}(f(i))$: Optional interval variables that model the different execution modes of a job (the different machines on which it can be scheduled). More precisely, such a variable is created for each machine m and for each job i that can be executed on m. In the final solution, only one variable $altJobs_{mi}$ is present for a job i and corresponds to the machine on which the job is scheduled. The domain of these variable is $dom(altJobs_{mi}) = \{[s,e) | [s,e) \subseteq [0,T), s + p_{f(i)} = e\}$.

- $disqualif_{fm}$, $\forall f \in \mathcal{F}$; $\forall m \in \mathcal{M}(f)$: Optional interval variables of size 0 that are used to model machine disqualifications. If the interval variable $disqualif_{fm}$ is present in the final solution, then machine m becomes disqualified for processing jobs of family f. The start time of the variable corresponds to the time at which the machine becomes disqualified.

- C_{max}: An integer variable that represents the end of the schedule. Its domain is $dom(C_{max}) = \{0, \ldots, T\}$.

The problem constraints are then presented one-by-one. The first set of constraints concerns the assignment of jobs to machines. *Alternative* constraints are used to model these features. Constraint $alternative(A, \{a_1, \ldots, a_n\})$ models an exclusive alternative between $\{a_0, \ldots, a_n\}$. If time-interval A is executed, then exactly one of the time-intervals $\{a_0, \ldots, a_n\}$ is executed and A starts and ends together with the selected time interval. Applying this constraint to our problem:

$$alternative(masterJob_i, \{altJobs_{mi} \mid m \in \mathcal{M}(f(i))\}),\ \forall i \in \mathcal{N} \tag{12}$$

To model the setup time, *noOverlap* constraints are used. This constraint ensures that the execution of several interval variables do not overlap. It can also handle the setup time. Let S be the matrix of setup times of the problem, i.e. $(S_{f',f}) = \begin{cases} 0 & \text{if } f = f', \\ s_f & \text{otherwise.} \end{cases}$ Then, the following *noOverlap* constraint makes sure that, for all pairs of jobs (i, j) s.t. both can be scheduled on m, either the start of $altJobs_{mj}$ occurs after the end of $altJobs_{mj}$ plus $s_{f(j)}$ or the opposite:

$$noOverlap(\{altJobs_{mi} \mid i \text{ s.t } m \in \mathcal{M}(f(i))\},\ S),\ \forall m \in \mathcal{M} \tag{13}$$

The next set of constraints ensure that all jobs are scheduled before C_{max}:

$$endOf(masterJob_i) \leq C_{max},\ \forall i \in \mathcal{N} \tag{14}$$

Finally, three sets of constraints are added to model machine disqualifications. The first set guarantees that once a machine m is disqualified for a certain family f, no job of family f is scheduled on m. In other words, if the interval variable $disqualif_{fm}$ is present in the solution, then there is no job of family f on m after $disqualif_{fm}$.

$$startOf(altJobs_{im}) + \gamma_{f(i)} \leq startOf(disqualif_{f(i)m}), \quad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}(f(i)) \tag{15}$$

The second constraint set enforces a machine to become disqualified if there is no job of family f scheduled on the machine during an interval of duration γ_f. More precisely, if a job of family f is scheduled on machine m, then either another job of f is scheduled in the next γ_f units of time or machine m becomes disqualified for family f. There is another case to consider which is the case where the job of family f is executed at "the end" of the schedule and then the disqualification occurs after the makespan. In this case, the machine does not become disqualified. Given a machine m and a job i, let $t_{mi} = startOf(altJobs_{mi}) + \gamma_{f(i)}$.

$$presenceOf(altJobs_{mi}) \Rightarrow \left(\bigvee_{i' \neq i\,;\, f(i)=f(i')} (startOf(altJobs_{mi'}) \leq t_{mi}) \right) \vee \left(startOf(disqualif_{f(i)m}) = t_{mi}\right) \vee \left(C_{max} \leq t_{mi}\right),\ \forall i \in \mathcal{N},\ \forall m \in \mathcal{M}(f(i)) \tag{16}$$

The last set of constraints imposes that if there is no job of family f scheduled on a qualified machine, then the machine becomes disqualified.

$$\left(\bigvee_{i \in \mathcal{N}\,;\, f(i)=f} (startOf(altJobs_{mi}) \leq \gamma_f) \right) \vee \left(startOf(disqualif_{f(i)m}) = \gamma_f\right) \vee$$
$$(C_{max} \leq \gamma_f)\,,\ \forall f \in \mathcal{F}, \forall m \in \mathcal{M}(f) \tag{17}$$

In our model, an additional constraint set is used to order the start time of jobs in the same family. Those ordering constraints are not mandatory but break some symmetries in the initial model.

$$\bigwedge_{j>i\,;\, f(i)=f(j)} (startOf(masterJob_i) < startOf(masterJob_j))\,,\ \forall i \in \mathcal{N} \tag{18}$$

The objective of the CP model is to minimize both the sum of completion times, i.e. sum of $endOf(masterJob_i)$), and the number of disqualifications, i.e. the number of interval variables $disqualif_{fm}$ in the solution. When either the first objective or the second objective is prioritized, experiments have been conducted to compare the use of the weighted sum of both objectives and of a lexicographical order. They show that the model, as this is the case for CP in general, does not perform well when combining objectives in a weighted sum. Therefore, in Section 5 and because one of the objectives is always prioritized, only results with a lexicographical order are presented.

4. Heuristics

Two constructive heuristics are presented in Obeid et al. (2014): The Scheduling-Centric Heuristic (SCH) and the Qualification-Centric Heuristic (QCH). SCH tries to minimize the sum of completion times by minimizing the number of setup times in the final solution, while QCH aims at minimizing the number of machine disqualifications. These heuristics are not described in this paper and the reader is referred to Obeid et al. (2014) for more details.

As constructive heuristics each focusing on one of the criteria, SCH and QCH provide good solutions but that can still be improved. This section presents two approaches to improve the solutions provided by SCH and QCH: A Recursive Heuristic (RH) and a Simulated Annealing (SA) algorithm. RH can be seen as a multi-start algorithm that aims at diversifying the search while remaining very fast. SA is a standard neighborhood-based metaheuristic that is known to be effective for numerous discrete optimization problems.

4.1. Recursive Heuristic

The general idea of the Recursive Heuristic is to slightly change the instance data to modify the behavior of the heuristic and explore other solutions. More precisely, consider a solution s obtained by any of the constructive heuristics (SCH or QCH). In s, although some machines are becoming disqualified in the scheduling horizon, the heuristic tries as much as possible to maintain these qualifications, sometimes at the expense of other

qualifications. Therefore, if the machine is disqualified from the beginning of the schedule, a better solution may be obtained. This reasoning can be extended to any subset of disqualified machines in s. The pseudo-code for RH is displayed in Algorithm 1.

Algorithm 1: Recursive Heuristic

```
Data: An instance I of PTC
Result: A solution s for I or NONE if no solution is found
if HEURISTIC(I) found a solution then
    s ← HEURISTIC(I);
    if s.disqualification > 0 then
        stopIter ← 2^{s.disqualification};
        cpt = 1;
        while cpt ≤ stopIter do
            I = DISQUALIFY(I, BINARY(cpt));
            s' ← HEURISTIC(I);
            if s'.score < s.score then
                s ← s';
                cpt = cpt + 1;
            end if
        end while
    end if
    return s
end if
else
    return NONE
end if
```

In the algorithm, the function $HEURISTIC$ returns a solution obtained by a given heuristic on a given instance. $s.disqualification$ is the number of disqualifications in solution s. This number is used as the stopping criterion of the algorithm. The function $BINARY$ transforms the integer cpt, which is a base-10 number, into the equivalent base-2 number. This guarantees that all possible combinations of machine disqualifications are covered by the algorithm. The score of a solution $s.score$ is based on the minimum of the sum of disqualifications prior to the sum of completion times.

Example 4.1. *Consider the following instance of PTC:*

$N = 10$, $\mathcal{M} = \{m_1, m_2\}$, $\mathcal{F} = \{f_1, f_2\}$,
$\mathcal{M}(1) = \{1\}$, $\mathcal{M}(2) = \{1, 2\}$,
$n_f = \{5, 5\}$, $p_f = \{7, 5\}$, $s_f = \{1, 5\}$, $\gamma_f = \{27, 24\}$.

When applying QCH on this instance, the solution in Figure 3 is obtained. The sum of completion times is equal to 237 *and the number of disqualifications is equal to* 2*: Machine* 1 *becomes disqualified for family* 2 *at time* $t = 18 + 24 = 42$*, and Machine* 2 *becomes disqualified for family* 2 *at time* $t = 10 + 24 = 34$*.*

In the first iteration of the RH procedure, Machine 1 *is removed from* $\mathcal{M}(2)$*, i.e. jobs of family* 2 *can no longer be scheduled on Machine* 1*. Applying QCH on this instance, the solution in Figure 4 is obtained. The sum of completion times is now equal to* 180 *and the number of disqualifications is equal to* 1*: Machine* 1 *becomes disqualified for family* 2 *at time* $t = 24$*.*

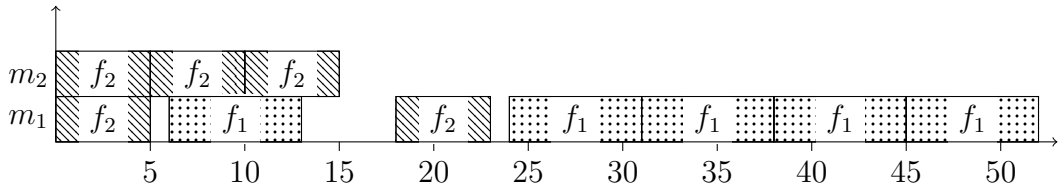


Figure 3: Example of a solution obtained by QCH before applying RH.

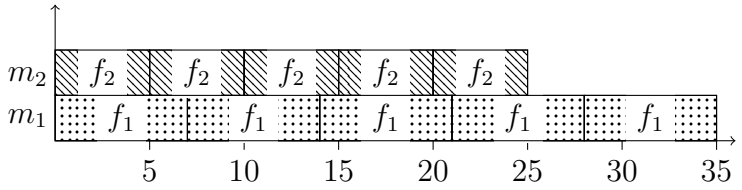


Figure 4: Example of a solution obtained by RH.

The complexity of the recursive heuristic RH is $\mathcal{O}(2^{|D|} \cdot (|N||M||F|))$ (respectively $\mathcal{O}(2^{|D|} \cdot (|N|^2|M|^2))$) for SCH (respectively for QCH), with D the number of qualification losses. Note that, for a large number of machine disqualifications, RH may not be usable in practice. However, since the number of pairs (machine, family) is not too large in the considered instances and since one of the criteria is the minimization of the number of disqualifications, RH can be used to solve the problem.

4.2. Simulated Annealing algorithm

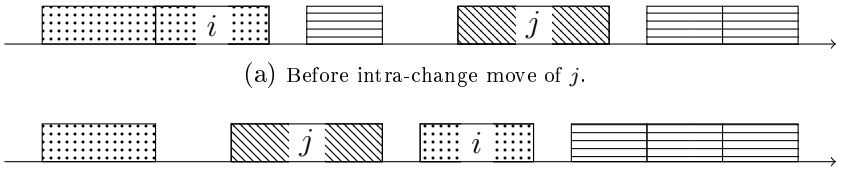
Simulated Annealing (SA) belongs to the class of randomized local search algorithms and was developed by Kirkpatrick et al. (1983) to handle hard combinatorial problems. SA has demonstrated its ability to solve scheduling problems (Teghem, 2002).

The limitations related to exact methods for PTC regarding the maximum number of jobs, machines and families led to the need for a more flexible method that can deal with large-scale instances. SA was chosen for this objective.

In a Simulated Annealing algorithm, an initial solution is used to generate a set of neighbouring solutions, which is considered to find a solution which has a better *score* than the score of the initial solution. To find the optimal or at least an improved solution, the solution search space needs to be explored effectively since the number of solutions is usually enormous. In SA, this exploration uses two major parameters which are: The temperature, and the number of iterations at each temperature. Actually, while exploring the solution space, we may step toward a solution which has a higher score than the current one. Hence, if the objective function is to minimize a certain criterion, then this solution should be ignored in a normal case. However, in SA, a worse solution is accepted with a probability that depends on the temperature. The acceptance of worse solutions helps to avoid the search to be stuck in a local optimum.

Other parameters of SA such as the cooling factor and the definition of a neighbouring solution are described later in the paper. First, we describe how the neighbourhood of a solution is generated, which impacts the efficiency of SA. Based on our preliminary experiments on neighbourhood structures, two different ways of generating a neighbour have been selected for our problem:

- *Intra-change insertion* of jobs. A job on a given machine is selected from the jth position and inserted before another job at the ith position on the same machine (see Figure 5). Note that this move also covers another typical way of generating neighbours, the *Intra-change Swapping*, where two jobs are selected randomly and swapped on the same machine. Intra-change insertion is found to be more flexible since any swap move may be achieved by two insertions but the inverse is not true. However, the difficulty in our neighbourhood generation lies in machine qualifications because some intra-change insertions may lead to an additional loss of qualifications and to a non feasible sequence of jobs on a given machine. Thus, each time an intra-change insertion is performed, the obtained sequence is tested for feasibility.



(a) Before intra-change move of j.

(b) After intra-change move of j: j is scheduled right before i.

Figure 5: Intra-change insertion.

- *Inter-change insertion* of jobs. Two job positions i and j are randomly selected on two different machines m and m' respectively. Then job j is inserted right before job i on machine m (see Figure 6). As for the intra-change insertion, the sequence on both machines should be checked for feasibility. Moreover, in an inter-change insertion, the problem of machines with no jobs after a move must be considered. In some situations, there may still only be one job on a machine and an insertion of this job on another machine leads to a machine with no jobs, thus the machine is not used at all and the number of machines in this case is decreased by 1. To overcome this difficulty, we check, each time a move is performed, whether the machine has strictly more than one job. This guarantees that there is no machine being idle on the whole scheduling horizon.

(a) Before inter-change insertion of j on m: i is scheduled on m and j on m'.

(b) After inter-change insertion of j on m: j is inserted before i on m.

Figure 6: Inter-change insertion.

In our implementation, 50% of the considered moves are intra-change insertions and 50% are inter-change insertions. Moves are randomly selected according to the uniform law. If a move is impossible due to a constraint violation, the previous solution is restored and another move is randomly selected.

The SA algorithm starts with an initial solution s_0 (found either by SCH or QCH in our case), and then tries to find better solutions by searching in the neighbourhood of the current solution and by applying a stochastic acceptance criterion. When a neighbour (a new solution s) of s_0 is selected, the difference $\Delta c = c(s) - c(s_0)$ ($c(.)$ is the score obtained with $\alpha \cdot (\sum_{f \in F} C_f) + \beta \cdot (\sum_{f \in \mathcal{F}} \sum_{m \in \mathcal{M}(f)} y^m_{f,T})$ as objective) is calculated. If Δc is negative, s has a better score than s_0, and the neighbour s replaces the current solution s_0. Otherwise, the neighbour s is accepted with a probability based on the Boltzmann distribution $P_{accept}(\Delta f) = exp(\Delta f / kT)$, where k is a constant and the temperature T is a control parameter.

The main idea is to start with a "high" initial temperature T_o and then to decrease it step by step. In our implementation, we start with an initial temperature $T_0 = 20,000$ and this temperature is gradually lowered following a geometrical function $g(T) = \delta T$ with $\delta = 0.95$.

Two other parameters need to be defined:

- The number of iterations at each temperature, $N_{iter} = 50$.
- A stopping criterion for the algorithm, N_{stop}. The algorithm stops because either the maximum number of temperature changes or the time limit of 600 seconds are reached. Parameters such as initial temperature and cooling schedule are taken into consideration for convergence to ensure that the temperature is sufficiently low when the stopping rule is satisfied. Experiments were conducted with different stopping rules, and $10,000$ temperature changes are used as the stopping criterion.

The pseudo code of the SA algorithm is given in Algorithm 2.

5. Numerical Experiments

Section 5.1 first describes the framework used for the experiments. The results obtained with the exact methods are presented and compared in Section 5.2. Section 5.3 discusses the results of the different heuristics. This section also includes a comparison between exact methods and heuristics. The impact of the time threshold duration is analyzed in Sections 5.2 and 5.3.

5.1. Framework

5.1.1. Instance generation

The benchmark instances used to perform our experiments are inspired by the ones of Obeid et al. (2014): 19 instance sets are generated with different number of jobs (N), machines (M), families ($|\mathcal{F}|$) and qualification schemes. Each of the instance sets is a group of 30 instances and are generated as follows.

First, we ensure that, at the beginning of the schedule, each family has at least one machine on which it can be processed and each machine is qualified to process at least one job family. Furthermore, in order to ensure a minimal bias to find a solution, the time thresholds of job families are chosen sufficiently large compared to their associated processing times. Indeed, short time thresholds may lead to very quick machine disqualifications. Then, it will not be possible to process all jobs before machines become

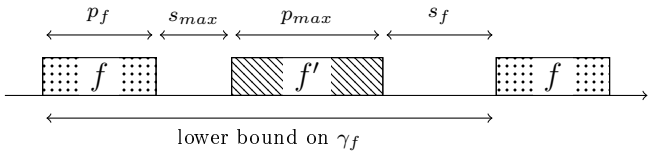
Algorithm 2: Simulated Annealing algorithm

Data: An instance $\mathcal{I}$ of PTC
Result: A solution s for $\mathcal{I}$ or $NONE$ if no solution is found
if $HEURISTIC(\mathcal{I})$ *found a solution* **then**
  $s \leftarrow HEURISTIC(\mathcal{I})$;
  $currentTemp \leftarrow T_0$;
  repeat
    $cpt = 0$;
    repeat
      $s' \leftarrow NEIGHBOR(s)$;
      $\Delta \leftarrow c(s') - c(s)$;
      if $\Delta < 0$ **then**
        $s \leftarrow s'$;
      end if
      else
        $s \leftarrow s'$ with probability $\exp \frac{-\Delta}{currentTemp}$;
      end if
      $cpt \leftarrow cpt + 1$;
    until $cpt = N_{iter}$;
    $currentTemp \leftarrow g(currentTemp)$;
  until *stopping criterion is met* N_{stop};
  return s
end if
else
  return $NONE$
end if

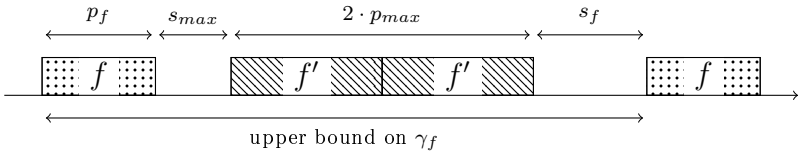
disqualified and then the instances will not be feasible. Thus, the initial family/machine qualification scheme is defined as follows. The minimum time threshold has to be larger than the longest processing time, i.e. $\max_{f\in\mathcal{F}} p_f \leq \min_{f\in\mathcal{F}} \gamma_f$. To ensure diversity, each set of instances contains 10 instances with short time thresholds, 10 with medium time thresholds and 10 with large time thresholds. Short time thresholds for a family f are in $[p_f + p_{max} + s_{max} + s_f, p_f + 2 \cdot p_{max} + s_{max} + s_f]$, with $p_{max} = \max_{f\in\mathcal{F}} p_f$ and $s_{max} = \max_{f\in\mathcal{F}} s_f$. These values correspond respectively to the duration needed to process one and two jobs of another family than f (see Figure 7). Then, medium time thresholds lie in $[p_f + 2 \cdot p_{max} + s_{max} + s_f, p_f + 3 \cdot p_{max} + s_{max} + s_f]$ and large time thresholds in $[p_f + 3 \cdot p_{max} + s_{max} + s_f, p_f + 4 \cdot p_{max} + s_{max} + s_f]$.

In addition, setup times are not chosen too large so that the risk of disqualifying a machine due to a setup time insertion is "acceptable", i.e. $\max_{f\in\mathcal{F}} s_f \leq \min_{f\in\mathcal{F}} p_f$.

Table 1 presents the parameters of the different instance sets. In the first column of Table 1, each instance type is indexed by a corresponding number (*No.*). These representative instance types summarize different possibilities of combinations between number of jobs N (second column), number of machines M (third column) and number of families $|\mathcal{F}|$ (fourth column). The fifth column, $\max p_f$, gives the upper bound of the generated processing times of all families in a given instance ($p_f, \forall f \in \mathcal{F}$). The maximum possible sum of machine-family qualifications $M \cdot |\mathcal{F}|$ is given in the sixth column. Finally, the seventh column provides the initial sum of qualified machines for all families f, $|\mathcal{M}(f)|$ of a given instance.



(a) Lower bound on time threshold for short time thresholds.



(b) Upper bound on the time threshold for short time thresholds.

Figure 7: Time threshold generation.

No.	N	M	$\lvert\mathcal{F}\rvert$	$\max p_f$	$M * \lvert\mathcal{F}\rvert$	$\sum_{f\in\mathcal{F}} (\lvert\mathcal{M}(f)\rvert)$
1	20	3	4	10	12	8
2	20	3	5	10	15	9
3	20	4	2	10	8	6
4	20	4	3	10	12	7
5	20	4	4	10	16	11
6	20	4	5	10	20	13
7	30	3	2	10	6	4
8	30	3	3	10	9	7
9	30	3	4	10	12	8
10	30	3	5	10	15	13
11	30	4	4	10	16	10
12	30	5	5	10	25	14
13	40	3	3	10	9	5
14	50	3	3	10	9	6
15	60	3	4	10	12	9
16	60	3	5	10	15	11
17	70	3	5	10	15	12
18	70	4	4	10	16	9
19	70	4	5	10	20	14

Table 1: Instance type characteristics.

5.1.2. Configuration

The experiments are conducted on an Intel Xeon E3-1240 processor of 3.5 GHz with 4 cores and 32 GB with a 64-bit Windows 10 operating system. The ILP model was solved using CPLEX 12.8 with 2 threads. The CP model was tested using CP Optimizer 12.8. All heuristics are coded in C++. The time limit for all solution methods is set to 600 seconds.

Exact solution methods (ILP and CP) were tested:

1. While prioritizing the minimization of the sum of completion times, i.e. $\alpha = \beta = 1$ and,
2. While prioritizing the minimization of the number of disqualifications, i.e. $\alpha = 1,\ \beta = N \cdot T$.

The CP model is actually using the lexicographical order to model the priority between criteria.

The Recursive Heuristic (RH) and the Simulated Annealing (SA) algorithm are tested with SCH and QCH as constructive heuristics to obtain a first feasible solution. The use of SCH corresponds to prioritizing the sum of completion times, while QCH corresponds to prioritizing the second objective, i.e. the number of disqualifications.

5.2. Exact Methods

5.2.1. ILP Results

Influence of first solution. For each of the two objectives being prioritized, the model has been solved with and without a first solution. When the sum of completion times (resp. the number of disqualifications) is prioritized, SCH (resp. QCH) is used to give the model a first solution to improve. For the first case (priority to the sum of completion times), the use of SCH has almost no impact on the quality of the solution of ILP and on the computational time. However, the use of QCH improves the number of instances solved by $(IP3)$. These results are detailed in Table 2. In this table, the first column corresponds to the instances types. The second and third columns compare the time needed to solve the instances without (w/o) and with (w/) QCH. The fourth and fifth columns show the difference between the percentage of instances solved, i.e. when at least one solution is found, and finally the sixth and seventh columns compare the percentage of instances solved optimally.

As shown in Table 2 when the number of disqualifications is prioritized, the use of QCH improves the performance of $(IP3)$. Indeed, the number of instances solved is larger when QCH is used to compute a first solution. Therefore, in the following, when the sum of completion times is prioritized, the model alone is used while, when it is the number of disqualifications which is prioritized, the model is used with QCH.

Detailed results of $(IP3)$. Tables 3 and 4 present the detailed results obtained by $(IP3)$. In these tables, the first column corresponds to the instances types. The second column presents the time needed to solve the instance. In this column, either the instances are solved to optimality or the time limit of 600 seconds has been reached. The third column details the percentage of solved instances, i.e. when at least one solution not necessarily optimal is found. The fourth and fifth columns show the objective values. The number

No.	Time (sec.)		%solved		%opt.	
	w/o QCH	w/ QCH	w/o QCH	w/ QCH	w/o QCH	w/ QCH
1	119.7	120.4	100	100	97	93
2	250.9	265	100	100	77	70
3	15.2	21.2	100	100	100	100
4	43	36.9	100	100	97	100
5	199.5	184.5	100	100	80	77
6	187.1	189.3	100	100	80	77
7	54.3	41.5	100	100	97	97
8	419.5	425.2	100	100	43	43
9	431.2	450.5	100	100	40	40
10	563.7	562	100	100	10	10
11	325.7	323.3	100	100	57	60
12	332.7	326.4	100	100	50	53
13	204.5	202.4	100	100	83	80
14	529.4	530.5	100	100	17	17
15	600	600	93	97	0	0
16	600	600	87	90	0	0
17	600	600	47	83	0	0
18	600	600	87	93	0	0
19	600	600	60	93	0	0

Table 2: Influence of using QCH to give a first feasible solution when optimizing with ($IP3$) ($\alpha = 1,\ \beta = N \cdot T$).

of setup times in the solution is given in the sixth column. Finally, the seventh column presents the percentage of instances solved optimally.

No.	Time (sec.)	%solved	$\sum_{f \in \mathcal{F}} C_f$	#disqualif.	#setups	%opt.
1	3.5	100	336	3.47	2.73	100
2	7.8	100	384	4.47	3.17	100
3	1.9	100	336	1.2	0.97	100
4	2.7	100	351	2.23	1.47	100
5	8.3	100	314	2.93	3.57	100
6	9.4	100	268	3.23	3.77	100
7	1.9	100	894	1.73	0.7	100
8	22	100	956	3.9	2.7	100
9	30.5	100	816	4.57	3.53	100
10	147.4	100	743	8.23	4.77	87
11	19.2	100	629	4.8	2.63	100
12	26	100	530	5.9	3.83	100
13	14.9	100	1527	2.73	1.7	100
14	59.1	100	2141	3.73	2.03	97
15	308.3	100	3068	6.3	5.07	63
16	429.4	97	2692	8.38	8	47
17	559.5	73	3567	8.86	11.41	17
18	260.8	97	3490	6.21	3.76	77
19	411.9	93	2616	10.25	7.04	50

Table 3: Model ($IP3$), ($\alpha = \beta = 1$): Minimizing the sum of completion times and the number of disqualifications.

The results of Table 3 show that ($IP3$) can solve instances up to 40 jobs optimally

when the sum of completion times is prioritized over the sum of disqualifications. Indeed, for these instances, the model finds the optimal solution in less than 30 seconds (except for one instance type: $No.$ 10). For the instances of larger size (instance types $No.$ $14-19$), $(IP3)$ finds solutions for most of the instances but is not able to solve them optimally.

$No.$	Time (sec.)	%solved	$\sum_{f\in\mathcal{F}} C_f$	#disqualif.	#setups	%opt.
1	120.4	100	399	0.47	7.17	93
2	265	100	447	1.43	7.37	70
3	21.2	100	365	0.03	2.80	100
4	36.9	100	420	0.20	3.77	100
5	184.5	100	360	0.57	6.27	77
6	189.3	100	305	1	5.97	77
7	41.5	100	1099	0.10	3.70	97
8	425.2	100	1133	0.70	9.53	43
9	450.5	100	970	1.10	10.57	40
10	562	100	901	3.73	15.03	10
11	323.3	100	723	1.17	8.47	60
12	326.4	100	609	2	8.47	53
13	202.4	100	1775	0.27	8.63	80
14	530.5	100	2488	0.90	12.07	17
15	600	97	3639	3.21	23.76	0
16	600	90	3345	5.30	28.44	0
17	600	83	5517	7.28	44.84	0
18	600	93	4252	3.46	20.75	0
19	600	93	3466	7.89	31.32	0

Table 4: Model $(IP3)$. $(\alpha = 1,\ \beta = N \cdot T)$: Minimizing the number of disqualifications and the sum of completion times.

When the number of disqualifications is prioritized (Table 4), the model also solves instances up to 50 jobs but not optimally. This illustrates that the complexity of the problem is also based on the chosen objective function, where prioritizing the qualification criterion over the scheduling criterion makes instances more difficult to solve.

Furthermore, the results for both criteria in both tables show that, when the number of disqualifications is prioritized (Table 4), the sum of completion times is not minimized. In addition, when the sum of completion times is prioritized over the sum of disqualifications (Table 3), the results show that the sum of disqualifications is not minimized. The number of setups while prioritizing the sum of disqualifications in Table 4 is larger than the number of setups in Table 3 for all instances. This is due to the fact that job families have to change frequently in order to satisfy the time threshold constraints, to minimize $\sum_{f\in\mathcal{F}} \sum_{m\in\mathcal{M}(f)} y^m_{f,T}$.

Influence of the time threshold duration. Tables 5 and 6 present the results on some instance sets according to the time threshold duration. The first column corresponds to the instance types and the second column to the time threshold duration where S (resp. M and L) stand for short time thresholds (resp. medium and large). The last five columns are similar to Tables 3 and 4.

Table 5 shows that instances with short time thresholds are harder to solve for $(IP3)$ compared to instances with larger time thresholds, since computational times are larger and percentages of instances solved to optimality are smaller. This can be explained by

No.	Thres.	Time (sec.)	%solved	$\sum_{f\in\mathcal{F}} C_f$	#disqualif.	#setups	%opt.
6	L	8.4	100	247	1.1	3.8	100
	M	8.8	100	271	2.6	3.7	100
	S	16.3	100	287	6	3.7	100
9	L	13	100	771	4	2.8	100
	M	36.8	100	795	4.4	3.4	100
	S	60.4	100	882	5.5	4.3	100
13	L	7.7	100	1345	2.2	2.2	100
	M	23.2	100	1500	3	1	100
	S	13.9	100	1735	3	1.9	100
16	L	312.7	100	2598	8.1	6.3	80
	M	426.2	100	2349	7.8	7.5	50
	S	549.2	90	3176	9.33	10.44	10
19	L	398.5	80	2648	9.75	6	50
	M	302.7	100	2462	9.7	7.6	70
	S	534.4	100	2744	11.2	7.3	30

Table 5: Model ($IP3$): Influence of time threshold duration ($\alpha = \beta = 1$).

the fact that instances with short time thresholds are more constrained than instances with larger time thresholds.

Furthermore, the sum of completion times is almost always smaller for instances with large time thresholds. This is mainly due to the fact that, for shorter time thresholds, there are many family changes in the solution. Indeed, these changes are necessary to maintain machine qualifications. Therefore, solutions have more setup times and thus larger completion times.

However, for instance type 19, the best results are obtained on instances with medium time thresholds. This can be explained by the fact that a model with too many or too few constraints can be harder to solve.

No.	Thres.	Time (sec.)	%solved	$\sum_{f\in\mathcal{F}} C_f$	#disqualif.	#setups	%opt.
6	L	23.2	100	282	0.3	4.4	100
	M	135	100	305	0.6	5.6	80
	S	409.6	100	329	2.1	7.9	50
9	L	357.5	100	911	0.3	9.5	60
	M	469.7	100	934	0.6	10.4	40
	S	524.1	100	1066	2.4	11.8	20
13	L	40.8	100	1422	0	5.8	100
	M	253	100	1769	0	10.1	70
	S	313.5	100	2134	0.8	10	70
16	L	600	100	3337	5	27.8	0
	M	600	100	2996	4.3	29.7	0
	S	600	70	3854	7.14	27.57	0
19	L	600	100	3944	8.1	28.1	0
	M	600	100	3026	5.9	32.7	0
	S	600	80	3416	10.13	33.63	0

Table 6: Model ($IP3$): Influence of time threshold duration ($\alpha = 1,\ \beta = N \cdot T$).

Table 6 shows that, when $\alpha = 1$ and $\beta = N \cdot T$, instances with short time thresholds are also harder to solve for ($IP3$). Indeed, for short time thresholds, less instances are

solved optimally and it takes more time to solve them. Furthermore, the number of disqualifications in the final solution is larger for instances with short time thresholds.

5.2.2. CP Results

As for ($IP3$), we investigate the influence of using SCH and QCH to compute a first solution to give to the model. When the number of disqualifications is prioritized (resp. the sum of completion times), the use of QCH (resp. SCH) has almost no impact on the performances of the model (similar computational time and number of instances solved optimally or not). Therefore, SCH and QCH are no longer used in our experiments.

Detailed results of the CP model. Tables 7 and 8 give the detailed results obtained by the CP model and have almost the same format than Tables 3 and 4 with one extra column. This column, the third one in the table, shows the time needed by the solver to find the best possible solution. For example, for instance type 1, after 12 seconds, the solver does not improve the solution.

No.	Time (sec.)	Time best (sec.)	%solved	$\sum_{f\in\mathcal{F}} C_f$	#disqualif.	#setups	%opt.
1	600	12	100	336	3.53	2.73	0
2	600	22.5	100	384	4.53	3.3	0
3	600	0.7	100	336	1.20	0.97	0
4	600	1.5	100	351	2.23	1.47	0
5	600	5.5	100	314	3.17	3.27	0
6	600	5.3	100	268	3.23	3.67	0
7	600	2.2	100	894	1.73	0.7	0
8	600	22.7	100	958	3.97	2.6	0
9	600	56.1	100	817	4.7	3.57	0
10	600	116.7	93	716	8.18	5.07	0
11	600	53.1	100	630	4.8	2.67	0
12	600	50.5	100	530	6.17	3.83	0
13	600	8.6	100	1527	2.73	1.7	0
14	600	19.9	100	2148	3.8	1.93	0
15	600	92.3	77	2793	6.3	4.87	0
16	600	69	47	2323	8	6.29	0
17	600	71.7	23	2455	8.57	8	0
18	600	70.1	93	3521	6.14	4.32	0
19	600	52.6	57	2777	10.65	6.18	0

Table 7: CP Model: Minimizing the sum of completion times and then the number of disqualifications.

Table 7 shows that, when priority is given to the sum of completion times, the CP model is able to find a solution for all instances up to 50 jobs (except for instance type 10). It can also solve larger instances but not all the instances. Another remark is that the model fails at proving the optimality of the solutions. Consequently, it always reaches the time limit of 600 seconds. However, the quality of the solutions found is good as shown in Section 5.2.3. Finally, the best solution determined by the solver is found in less than 120 seconds. Therefore, this model can be used as a heuristic to quickly find a good solution but with no guarantee of optimality.

No.	Time (sec.)	Time best (sec.)	%solved	$\sum_{f \in \mathcal{F}} C_f$	#disqualif.	#setups	%opt.
1	270	19.7	100	400	0.47	7.23	63
2	365	2.6	100	452	1.37	7.07	43
3	106	0.4	100	365	0.03	2.8	93
4	124	11.1	100	420	0.2	3.77	83
5	335	14.9	100	367	0.5	6.57	60
6	371	27.2	100	314	0.87	6.23	47
7	325	6.5	100	1047	0.23	2.87	67
8	583	50.8	100	1120	0.67	9.67	3
9	595	35.9	100	949	1.03	10.4	3
10	600	128	87	843	2.65	14.15	0
11	542	7.5	100	724	0.97	8.87	10
12	600	12.9	100	624	1.7	9.43	0
13	538	45.1	100	1678	0.43	7.4	20
14	600	53.2	100	2302	1.43	7	0
15	600	87.8	80	3029	2.63	15.83	0
16	600	109.6	47	2698	3.64	19.21	0
17	600	64	23	3375	4.14	27.29	0
18	600	102.5	90	3729	3.04	10.26	0
19	600	75.4	50	2886	5.67	16.93	0

Table 8: CP Model: Minimizing the number of disqualifications and the sum of completion times.

Table 8 shows that, when the number of disqualifications is prioritized, the CP model is able to solve almost all the instances up to 50 jobs but fails at proving the optimality for most solutions. For larger instances, fewer instances are solved and none of the solution found is proved optimal. Nevertheless, the quality of solutions is very good (cf. Section 5.2.3) and the best solutions are found in less than 110 seconds.

Influence of the time threshold duration. As for ($IP3$), let us analyze the influence of the time threshold duration on the performances of the model. Tables 9 and 10 present the results on some instance sets according to the time threshold duration and have the same format than Tables 5 and 6 with an extra column for the time needed by the solver to find the best solution.

As for ($IP3$), Tables 9 and 10 show that instances with short time thresholds are harder to solve, in terms of both solution times and optimality gaps, and have a larger sum of completion times and a larger number of disqualifications.

5.2.3. Comparison of exact methods

This section compares the results obtained by the CP model and ($IP3$). Figure 8a shows the deviation of the sum of completion times, when this objective is prioritized, with respect to the best solution found either by the CP model or by ($IP3$). For ($IP3$), if the instances are solved both by the CP model and by ($IP3$), it is computed as:

$$100 - \frac{100 \times OBJ_{C_f}^{IP3}}{OBJ_{C_f}^{BEST}}$$

No.	Thres.	Time (sec.)	Time best (sec.)	%solved	$\sum_{f\in\mathcal{F}} C_f$	#disqualif.	#setups	%opt.
6	L	600	2.4	100	247	1.1	3.8	0
	M	600	1.6	100	271	2.6	3.5	0
	S	600	12	100	287	6	3.7	0
9	L	600	40.9	100	771	4	2.8	0
	M	600	78.4	100	799	4.5	3.6	0
	S	600	49.1	100	882	5.6	4.3	0
13	L	600	3.3	100	1345	2.2	2.2	0
	M	600	1.7	100	1500	3	1	0
	S	600	20.9	100	1735	3	1.9	0
16	L	600	135.4	70	2381	7.86	7.29	0
	M	600	67.9	50	2210	8	5.8	0
	S	600	3.6	20	2399	8.5	4	0
19	L	600	71.6	70	2766	9.71	6.71	0
	M	600	40.5	50	2376	10.4	7	0
	S	600	45.7	50	3192	12.2	4.6	0

Table 9: CP Model: Influence of time threshold duration (priority on sum of completion times).

No.	Thres.	Time (sec.)	Time best (sec.)	%solved	$\sum_{f\in\mathcal{F}} C_f$	#disqualif.	#setups	%opt.
6	L	261	2.2	100	282	0.3	4.3	70
	M	335	16	100	315	0.5	5.9	50
	S	516	63.4	100	344	1.8	8.5	20
9	L	584	7.3	100	905	0.3	9.3	10
	M	600	14	100	938	0.5	11.1	0
	S	600	86.6	100	1004	2.3	10.8	0
13	L	443	0.7	100	1422	0	5.9	50
	M	600	49.2	100	1658	0.3	7.8	0
	S	572	85.5	100	1954	1	8.5	10
16	L	600	188.1	70	2886	2.86	21.14	0
	M	600	97.7	50	2498	3.8	19.2	0
	S	600	43	20	2536	6	12.5	0
19	L	600	34	60	2821	3.67	20.83	0
	M	600	121.7	50	2637	5.2	17.8	0
	S	600	70.5	40	3294	9.25	10	0

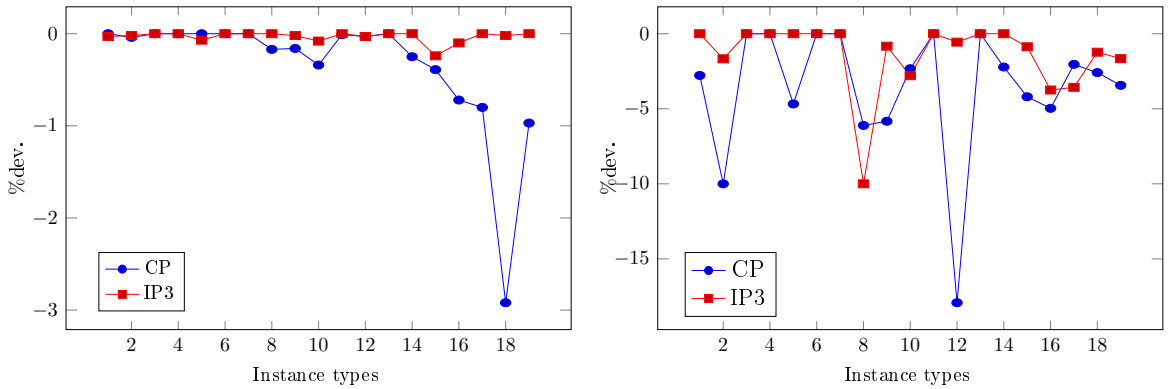
Table 10: CP Model: Influence of time threshold duration (priority on number of disqualifications).

with $OBJ_{C_f}^{IP3}$ the value of $\sum_{f\in\mathcal{F}} C_f$ determined by $(IP3)$ and $OBJ_{C_f}^{BEST}$ the best value of $\sum_{f\in\mathcal{F}} C_f$ determined by either $(IP3)$ or the CP model. For the CP model, the deviation is computed with the same formula, replacing $OBJ_{C_f}^{IP3}$ by $OBJ_{C_f}^{CP}$. Figure 8b shows the deviation of the number of disqualifications, when priority is given to the sum of completion times, with respect to the best solution found by either the CP model or $(IP3)$. For $(IP3)$, if the instance is solved by both the CP model and $(IP3)$, it is

computed as:

$$\begin{cases} 0 & \text{if } OBJ^{IP3}_{Y^m_f} = 0 \\ -100 \times OBJ^{IP3}_{Y^m_f} & \text{if } OBJ^{CP}_{Y^m_f} = 0 \\ 100 - \frac{100 \times OBJ^{IP3}_{Y^m_f}}{OBJ^{BEST}_{Y^m_f}} & \text{otherwise} \end{cases}$$

with $OBJ^{IP3}_{Y^m_f}$ the value of $\sum_{f \in \mathcal{F}} \sum_{m \in \mathcal{M}} Y^m_f$ determined by $(IP3)$, $OBJ^{CP}_{Y^m_f}$ the objective value of $\sum_{f \in \mathcal{F}} \sum_{m \in \mathcal{M}} Y^m_f$ determined by the CP model and $OBJ^{BEST}_{Y^m_f} = \min\left(OBJ^{IP3}_{Y^m_f}, OBJ^{CP}_{Y^m_f}\right)$. For the CP model, the deviation is computed with the same formula, exchanging $OBJ^{IP3}_{Y^m_f}$ with $OBJ^{CP}_{Y^m_f}$.



(a) Deviation of sum of completion times.

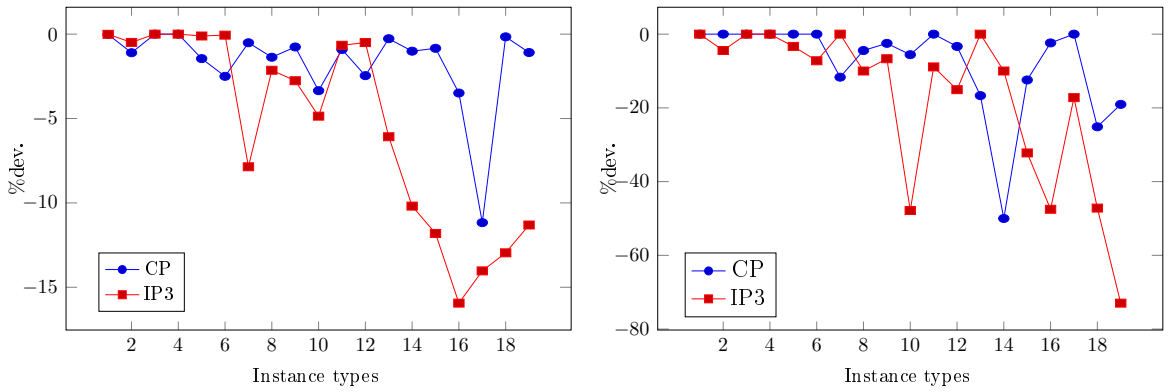
(b) Deviation of number of disqualifications.

Figure 8: Deviation of the objective values wrt. the best solution found ($\alpha = \beta = 1$).

Figure 8 shows that, when the sum of completion times is prioritized, the CP model finds solutions as good as the ones determined by $(IP3)$ for the sum of completion times for instances with less than 50 jobs. For larger instances, $(IP3)$ finds better solutions when considering the sum of completion times, but the solutions found by the CP model are at most 2% far from the ones determined by $(IP3)$. When considering the number of disqualifications, $(IP3)$ almost always finds better solutions than the CP model (except for instance types 8 and 16). Therefore, even if the CP model cannot prove the optimality of the solutions it determines, it can find quite good solutions quickly. However, the solutions of $(IP3)$ are better, but the solutions of the CP model are not so far (3% for the sum of completion times and 20% for the number of disqualifications which corresponds to less than one extra disqualification).

Figure 9 shows the deviation of both the number of disqualifications, when it is prioritized, and the completion time with respect to the best solution found by either the CP model or $(IP3)$.

Figure 9 shows that, when the number of disqualifications is prioritized, the best solutions for the number of disqualification are found by the CP model (except for instance types 7 and 14). This is also true for the sum of completion times for which the CP model obtains the best results for instances larger than 30 jobs.



(a) Deviation of sum of completion times. (b) Deviation of number of disqualifications.

Figure 9: Deviation of the number of disqualifications wrt. the best solution found ($\alpha = 1,\ \beta = N \cdot T$).

Therefore, the solution found by the CP model tends to be better than the one determined by ($IP3$) when the number of disqualifications is prioritized. Thus, even if the CP model is not able to prove the optimality of the solution it determines, it can find quite good solutions quickly.

Figures 8 and 9 show the deviation of the objective value only for instances solved by both ($IP3$) and the CP model. Note that, when considering the percentage of solved instances, ($IP3$) obtains better results than the CP model. Table 11 compares the number of solved instances by ($IP3$) and by the CP model. This table has two parts. In the upper part, the percentages of solved instances by ($IP3$) and by the CP model for $\alpha = \beta = 1$ are provided. The lower part shows the same data for $\alpha = 1$ and $\beta = N \cdot T$. In both parts, the first line corresponds to the instance type ($1 - 9$ means instance type 1 to 9). The second line shows the percentage of solved instances by the CP model and the third line by ($IP3$).

$\alpha = \beta = 1$								
type	1-9	10	11-14	15	16	17	18	19
CP	100	93	100	76	46	23	93	56
$IP3$	100	100	100	100	96	73	96	93
$\alpha = 1,\ \beta = N \cdot T$								
type	1-9	10	11-14	15	16	17	18	19
CP	100	86	100	80	46	23	90	50
$IP3$	100	100	100	96	90	83	93	93

Table 11: Comparison of the percentages of solved instances for ($IP3$) and CP.

Table 11 shows that, for small instances (up to 50 jobs), ($IP3$) and the CP model solve a similar number of instances, with slightly better results for ($IP3$). However, for larger instances, the number of instances solved by ($IP3$) is significantly larger than for the CP model.

5.3. Heuristics

Tables 12 and 13 show the impact of the Recursive Heuristic (RH) and the Simulated Annealing (SA) algorithm on the constructive heuristics presented in Obeid et al. (2014), SCH and QCH. In these tables, SCH (resp. QCH) denotes the constructive heuristic used alone to solve the instances, RHSCH (resp. RHQCH) corresponds to the use of RH together with SCH (resp. QCH) and SASCH (resp. SAQCH) the use of SA and SCH (resp. QCH). Then, the first column corresponds to the instances types. The second column presents the percentage of instances solved by the heuristics. Note that the use of RH and SA has no impact on the number of solved instances, i.e. if the constructive heuristic fails to find a solution, then neither RH nor SA solve the instance. Conversely, if the constructive heuristic finds a solution, then so do RH and SA. The third, fourth and fifth columns compare the value of the sum of completion times for SCH/QCH, RHSCH/RHQCH and SASCH/SAQCH. The sixth, seventh and eighth columns compare the value of the number of disqualifications and the last three columns show the different solution times. The quality of the solutions compared to the ones obtained by exact methods is discussed in the following paragraph.

		$\sum_{f\in\mathcal{F}} C_f$			#disqualif.			Time (sec.)		
No	%solved	SCH	RHSCH	SASCH	SCH	RHSCH	SASCH	SCH	RHSCH	SASCH
1	97	374	346	342	4.72	4	3.86	0	0	32.3
2	80	430	388	365	5.25	4.63	4.25	0	0	27.5
3	100	378	344	320	2.30	1.40	1.27	0	0	49.8
4	100	398	362	324	2.93	2.47	1.90	0	0	38.3
5	100	358	330	318	4.60	3.77	3.03	0	0	41.2
6	100	318	282	274	5.77	4.60	3.90	0	0	38.8
7	100	998	912	785	2.23	2	1.50	0	0	97
8	90	1060	966	949	4.59	4.04	4	0	0	35.1
9	73	837	773	789	5.86	4.91	4.86	0	0	26.7
10	87	770	718	754	8.50	7.88	8.08	0	0	26.2
11	90	696	613	616	6.78	5.19	5.74	0	0	45.9
12	97	615	547	548	8.93	7.48	7.07	0	0	51.7
13	70	1494	1432	1383	3.48	3.43	3	0	0	38.5
14	73	2196	2047	2080	4.64	3.95	4.18	0	0	46
15	70	2901	2682	2874	6.76	6.33	6.86	0	0	44.2
16	43	2364	2143	2346	8.92	8	9	0	0	23.3
17	30	3374	3193	3365	9.89	8.56	9.89	0	0	21.3
18	63	3691	3258	3527	7.26	6.58	7.32	0	0	52.3
19	60	2759	2508	2742	11.94	11.11	11.33	0	0.1	55.6

Table 12: Impact of RH and SA on SCH.

Table 12 shows that RH and SA significantly improve the performances of SCH. Indeed, the objective value (sum of completion times as well as the number of disqualifications) is reduced by the use of either RH or SA. However, the performances of RH and SA are not identical. For small instances (types 1 to 8), SA is more efficient in terms of objective value improvement than RH. For larger instances, RH becomes more efficient than SA. Furthermore, the time needed to solve the instances is larger for SA but does not exceed 60 seconds. Note also that SCH, and therefore SASCH and RHSCH, fails to find

a feasible solution in some cases. This is due to the fact that SCH focuses on setup time minimization and therefore sometimes let machine disqualification occurs. This implies that, at some point, there is no machine left to schedule one (or more) job family, and thus SCH cannot find a feasible solution.

		$\sum_{f\in\mathcal{F}} C_f$			#disqualif.			Time (sec.)		
No	%solved	QCH	RHQCH	SAQCH	QCH	RHQCH	SAQCH	QCH	RHQCH	SAQCH
1	97	395	388	393	3.97	2.66	0.45	0	0	32.4
2	87	464	455	408	5.15	3.69	0.88	0	0	29.1
3	100	377	386	331	1.80	1.10	0.07	0	0	50.1
4	100	413	433	344	2.70	2	0.13	0	0	38.2
5	93	371	359	332	4.21	2.32	0.57	0	0	40
6	93	319	303	286	4.96	2.89	1.04	0	0	37.4
7	100	1040	1079	827	1.90	1.40	0.03	0	0	98.8
8	97	1226	1196	1102	3.10	2.07	0.34	0	0	38.1
9	90	1033	971	970	4.70	3.11	0.78	0	0	30.9
10	77	875	835	949	6.35	4.17	2.57	0	0	24.4
11	97	819	769	762	5.59	3.34	1.34	0	0	48.7
12	90	663	642	620	8	5.15	2	0	0	50.5
13	93	1803	1810	1672	2.54	1.86	0.11	0	0	46.6
14	100	2909	2893	2642	3.87	2.37	0.40	0	0	59.1
15	93	3891	3590	3851	5.79	3.79	1.82	0	0	57.3
16	83	3513	3391	3511	7.80	4.56	2.96	0	0	45
17	80	5819	5372	5533	9.33	4.96	4.38	0	0	57
18	90	4604	4255	4371	6.56	4.22	2.04	0	0	74.3
19	87	3737	3574	3778	10.65	5.65	4.58	0	0.1	78.3

Table 13: Impact of RH and SA on QCH.

Table 12 shows that RH and SA also improve the performances of QCH. The number of disqualifications (as well as the sum of completion times) is reduced by the use of either RH or SA. However, the performances in terms of solution quality of SA are better than the performances of RH for all instances when the number of disqualifications is prioritized. Furthermore, the time needed to solve the instances for SA does not exceed 100 seconds. Note also that SCH has more difficulty to find a feasible solution than QCH, since QCH focuses on keeping machines qualified.

Comparison with Exact Methods. Table 14 compares the results obtained by the exact methods ($(IP3)$ and the CP model) with the results obtained by SA and RH. In this table, the first column corresponds to the instances types. The second, third, fourth and fifth columns compare the computational time needed to solve the instances with the CP model, $(IP3)$, SA and RH, respectively. The sixth, seventh and eighth columns show the difference between the percentage of solved instances with the CP model, $(IP3)$ and SA/RH (the number of solved instances is the same for SA and RH), respectively. Finally, the ninth, tenth, eleventh and twelfth columns present the deviation of the objective value for SA and RH with respect to the best solution found by either the CP model or $(IP3)$. The deviation of the sum of completion times for SA is computed as:

$$dev_{SA/CP}^{C_f} = 100 - \frac{100 \times OBJ_{C_f}^{SA}}{OBJ_{C_f}^{BEST}}$$

with $OBJ_{C_f}^{SA}$ the objective value of $\sum_{f \in \mathcal{F}} C_f$ determined by SA and $OBJ_{C_f}^{BEST}$ the best objective value of $\sum_{f \in \mathcal{F}} C_f$ determined by either $(IP3)$ or the CP model.

The deviation of the number of disqualifications for SA is computed as the minimum of:

$$\begin{cases} 0 & \text{if } OBJ_{Y_f^m}^{SA} = 0 \\ -100 \times OBJ_{Y_f^m}^{SA} & \text{if } OBJ_{Y_f^m}^{IP3} = 0 \\ 100 - \frac{100 \times OBJ_{Y_f^m}^{SA}}{OBJ_{Y_f^m}^{BEST}} & \text{otherwise} \end{cases}$$

and

$$\begin{cases} 0 & \text{if } OBJ_{Y_f^m}^{SA} = 0 \\ -100 \times OBJ_{Y_f^m}^{SA} & \text{if } OBJ_{Y_f^m}^{CP} = 0 \\ 100 - \frac{100 \times OBJ_{Y_f^m}^{SA}}{OBJ_{Y_f^m}^{BEST}} & \text{otherwise} \end{cases}$$

with $OBJ_{Y_f^m}^{IP3}$ (resp. $OBJ_{Y_f^m}^{CP}$ and $OBJ_{Y_f^m}^{SA}$) the objective value of $\sum_{f \in \mathcal{F}} \sum_{m \in \mathcal{M}} Y_f^m$ determined by $(IP3)$ (resp. by the CP model and by SA) and

$$OBJ_{Y_f^m}^{BEST} = \min\left(OBJ_{Y_f^m}^{IP3}, OBJ_{Y_f^m}^{CP}\right)$$

The same deviations are computed for RH, replacing SA by RH in the previous formulas.

	Time (sec.)				%solved			$\sum_{f \in \mathcal{F}} C_f$	$\sum_{f \in \mathcal{F}} C_f$	#disqualif.	#disqualif.
No	CP	IP3	SASCH	RHSCH	CP	IP3	SA/RHSCH	SASCH	RHSCH	SASCH	RHSCH
1	600	3.5	32.3	0	100	100	97	-2.3%	-2.6%	-33.9%	-45.9%
2	600	7.8	27.5	0	100	100	80	-1.7%	-3.5%	-38.9%	-40.5%
3	600	1.9	49.8	0	100	100	100	-0.3%	-2.7%	-41.7%	-26.7%
4	600	2.7	38.3	0	100	100	100	-0.2%	-3.6%	-31.9%	-27.2%
5	600	8.3	41.2	0	100	100	100	-3.1%	-5.5%	-54.3%	-60.7%
6	600	9.4	38.8	0	100	100	100	-3.9%	-5.8%	-58.9%	-78.1%
7	600	1.9	97	0	100	100	100	-0.4%	-2.5%	-26.7%	-33.3%
8	600	22	35.1	0	100	100	90	-3.9%	-4.2%	-44.8%	-33.3%
9	600	30.5	26.7	0	100	100	73	-6.2%	-3.8%	-29%	-37%
10	600	147.4	26.2	0	93	100	87	-7.9%	-1.8%	-14.7%	-9.3%
11	600	19.2	45.9	0	100	100	90	-5%	-4.1%	-45.7%	-35%
12	600	26	51.7	0	100	100	97	-6.1%	-3.2%	-83.4%	-133.7%
13	600	14.9	38.5	0	100	100	70	-2.7%	-2.9%	-37.3%	-41.3%
14	600	59.1	46	0	100	100	73	-6.6%	-3.9%	-31.4%	-13.6%
15	600	308.3	44.2	0	77	100	70	-9.6%	-2.4%	-19.9%	-10.2%
16	600	429.4	23.3	0	47	97	43	-12%	-2.2%	-17.4%	-7.3%
17	600	559.5	21.3	0	23	73	30	-7.7%	-1.3%	-18.3%	-4.7%
18	600	260.8	52.3	0	93	97	63	-10%	-1.7%	-30.2%	-14.6%
19	600	411.9	55.6	0.1	57	93	60	-11%	-2.2%	-20.5%	-16.8%

Table 14: Comparison of SASCH and exact methods ($\alpha = \beta = 1$).

Table 14 shows that, when the sum of completion times is prioritized, ($IP3$) is better than SA for small instances (up to 50 jobs). Indeed, the computational time needed to solve the instances is smaller when ($IP3$) is used. Furthermore, the number of instances solved is larger with ($IP3$). Also, the sum of completion times determined by SA is close to the sum of completion times determined by the exact methods.

When compared to RH, the exact methods can solve more instances but require much longer computational times than RH. However, the solutions determined by RH are not optimal and the sum of completion times determined by RH can be 5% larger than the one determined by the exact methods.

For larger instances, the gap between the solutions of SA and the exact methods increases with the size of the instances, but this gap decreases for RH when comparing with the solutions of the exact methods. Furthermore, the time needed to solve the instances is significantly smaller with SA than with the exact methods, and is even smaller for RH (less than 0.01 seconds). However, the number of solved instances remains larger with ($IP3$).

Table 15 compares the results obtained with the exact methods (($IP3$) and the CP model) with the results obtained with SA. The format of the table is the same than Table 14 minus the columns corresponding to RH.

	Time (sec.)			%solved			$\sum_{f\in\mathcal{F}} C_f$	#disqualif.
No	CP	IP3	SAQCH	CP	IP3	SAQCH	SAQCH	SAQCH
1	269.6	120.4	32.4	100	100	97	-4.1%	-12.1%
2	365.4	265	29.1	100	100	87	-3.6%	-14.7%
3	105.8	21.2	50.1	100	100	100	-0.5%	-6.7%
4	124.3	36.9	38.2	100	100	100	-1%	-10%
5	334.5	184.5	40	100	100	93	-4.4%	-33.9%
6	370.7	189.3	37.4	100	100	93	-6.3%	-41.7%
7	325.5	41.5	98.8	100	100	100	-0.8%	0%
8	582.6	425.2	38.1	100	100	97	-7.2%	-6.9%
9	594.8	450.5	30.9	100	100	90	-12.2%	-11.1%
10	600	562	24.4	87	100	77	-19.6%	-34.1%
11	542	323.3	48.7	100	100	97	-8.7%	-39.1%
12	600	326.4	50.5	100	100	90	-8.4%	-56.8%
13	538.3	202.4	46.6	100	100	93	-5.9%	0%
14	600	530.5	59.1	100	100	100	-16.9%	-3.3%
15	600	600	57.3	80	97	93	-23.2%	-10.7%
16	600	600	45	47	90	83	-18.7%	-6%
17	600	600	57	23	83	80	-11.7%	0%
18	600	600	74.3	90	93	90	-18.4%	-16%
19	600	600	78.3	50	93	87	-17.9%	-8.3%

Table 15: Comparison of SAQCH and exact methods ($\alpha = 1,\ \beta = N \cdot T$).

Table 15 shows that, when the number of disqualifications is prioritized (with QCH), SA performs better than SCH. Indeed, the time needed to solve the instances is almost always shorter with SA compared to ($IP3$) or the CP model. This time does not exceed 100 seconds for SA and almost always exceeds 100 seconds with ($IP3$) and the CP model. Furthermore, the performances of SA in terms of objective value are very good as well. Indeed, for some instance types, the number of disqualifications is as good as the one

determined by the exact methods. For the other instances, the difference is smaller than 60%, which is, in the case of qualification losses, a good performance. Finally, the number of solved instances is slightly smaller with SA than with ($IP3$), but for large instances (> 50 jobs), SA solves more instances than the CP model (which provides the best results for the number of disqualifications).

Influence of the time threshold duration. As for exact methods, the influence of the time threshold duration on the performances of SA is analyzed. Tables 16 and 17 have the same format than Tables 5 and 6. Table 16 shows that the time needed to solve the instances is smaller for instances with short time thresholds. However, the number of solved instances is larger when the time thresholds are large because problems are less constrained than with small time thresholds and thus easier to solve. Table 17 also shows that the time needed to solved the instances is smaller for instances with short time thresholds. Furthermore, as for SASCH, the number of solved instances is larger when the instances have larger time thresholds. This is due to the difficulty of the instances. As a consequence, the number of disqualifications is also smaller with larger time thresholds.

No.	Thres.	Time (sec.)	%solved	$\sum_{f\in\mathcal{F}} C_f$	#disqualif.	#setups
6	L	50	100	250	0.30	7.60
	M	40.6	100	280	2.80	7
	S	25.9	100	293	8.60	7.30
9	L	46.5	100	809	4.30	9.30
	M	26.6	90	800	4.78	9.56
	S	7.1	30	693	7	7.33
13	L	76.5	90	1306	2.33	7.89
	M	25.9	70	1364	3.43	12.43
	S	13.2	50	1549	3.60	12.80
16	L	45.8	80	2482	8.75	12.25
	M	24.1	50	2129	9.40	8.60
	S	–	0	–	–	–
19	L	77.2	70	3040	10.14	20.86
	M	65.8	80	2538	12.13	10.75
	S	23.9	30	2593	12	11.33

Table 16: SASCH: Influence of time threshold duration.

6. Conclusions and perspectives

An original parallel machine scheduling problem was studied where some Advanced Process Control constraints are integrated: Minimal time constraints between jobs of the same family to be processed on a qualified machine to avoid losing the qualification. Two criteria to minimize are considered: The sum of completion times and the number of disqualifications. An integer linear programming model ($IP3$) and a constraint programming (CP) model were proposed to solve the problem, as well as two heuristics improving existing constructive heuristics: A Recursive Heuristic (RH) and a Simulated Annealing (SA) algorithm.

Computational experiments were conducted on randomly generated instances. The results showed that both ($IP3$) and the CP model are efficiently solving the problem.

No.	Thres.	Time (sec.)	%solved	$\sum_{f \in \mathcal{F}} C_f$	#disqualif.	#setups
6	L	50	100	251	0	8.30
	M	40.5	100	293	0.20	9.50
	S	21.6	80	321	3.38	10.38
9	L	47.3	100	939	0	17.40
	M	29.6	100	1019	0.50	18.20
	S	15.8	70	946	2.29	17.86
13	L	81.6	100	1482	0	14.90
	M	36.8	100	1645	0	19.40
	S	21.4	80	1941	0.38	23
16	L	61.7	100	3641	1.90	44.70
	M	50	100	3390	3.10	44.70
	S	23.3	50	3494	4.80	45
19	L	108.5	100	4189	2.90	53.20
	M	83.1	100	3394	4.10	51.80
	S	43.4	60	3735	8.17	48.83

Table 17: SAQCH: Influence of time threshold duration.

Indeed, ($IP3$) allows more instances to be solved optimally but produces on average solutions that are of lower quality than the CP model. This is particularly true when the number of disqualifications is the primary criterion. The numerical results also showed that RH and SA are significantly improving the initial constructive heuristics SCH, prioritizing the sum of completion times, and QCH, prioritizing the number of disqualifications. RHSCH, respectively SASCH, corresponds to using SCH in RH, respectively SA, while RHQCH, respectively SAQCH, corresponds to using QCH in RH, respectively SA. When the sum of completion times is prioritized, SASCH is more efficient than RHSCH on small instances (up to 50 jobs) and RHSCH is more efficient than SASCH on larger instances. On the opposite, when the number of disqualifications is prioritized, SAQCH is better than RHQCH on all instances. The comparison between SA and the exact methods showed that SASCH and SAQCH are capable of producing good solutions with much shorter computational times. The numerical experiments also emphasized the difficulty of solving instances with short time thresholds.

Future research includes developing other neighborhood-based metaheuristics, such as Tabu Search, population-based metaheuristics, such as Genetic Algorithms, or combinations of both types of metaheuristics, such as Genetic Local Search. This could be particularly relevant when investigating the integration of time constraints to maintain machine qualifications in other types of workshops, such as workshops with batching machines and multiple processing stages. Other criteria will probably have to be considered. Studying the possible balance between the two criteria considered in this paper is another interesting research avenue. Finally, following the recent work of Kao et al. (2018), considering the dynamic status of machines ("equipment health") seems very relevant to allow a machine to "lose" its qualification depending on its status.

Another relevant research perspective consists in scheduling jobs on a longer time horizon, where lost qualifications could be automatically recovered after a given qualification procedure. Qualification procedures, requiring time on machines, would then also be scheduled. From a broader perspective, defining the "critical" qualifications of job fa-

milies to machines to maintain in a schedule, based on the work on flexibility measures in Johnzén et al. (2011) and Rowshannahad et al. (2015), should be appealing for production managers.

Acknowledgements

This work was supported in part by Agence Nationale de la Recherche (ANR), France, under grants ANR-15-CE10-0003 and the Ministry of Science and Technology, Taiwan, ROC, under grants MOST 104-2917-I-002-030, MOST 103-2221-E- 002-220-MY2, MOST 103-2911-I-002-513, MOST 105-2923-E-002-009-MY3.

References

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.

Brailsford, S. C., Potts, C. N., Smith, B. M., 1999. Constraint satisfaction problems: Algorithms and applications. European Journal of Operational Research 119 (3), 557 – 581.

Cai, Y., Kutanoglu, E., Hasenbein, J., Qin, J., 2011. Single-machine scheduling with advanced process control constraints. Journal of Scheduling, 1–15.

Cakici, E., Mason, S., 2007. Parallel machine scheduling subject to auxiliary resource constraints. Production Planning and Control 18 (3), 217–225.

Dauzère-Pérès, S., Lasserre, J.-B., 2002. On the importance of sequencing decisions in production planning and scheduling. International transactions in operational research 9 (6), 779–793.

Gaudreault, J., Frayret, J.-M., Rousseau, A., D'Amours, S., 2011. Combined planning and scheduling in a divergent production system with co-production: A case study in the lumber industry. Computers and Operations Research 38, 1238–1250.

Jain, V., Grossmann, I. E., 2001. Algorithms for hybrid MILP/CP models for a class of optimization problems. INFORMS Journal on Computing 13 (4), 258–276.

Jedidi, N., Sallagoity, P., Roussy, A., Dauzere-Peres, S., 2011. Feedforward run-to-run control for reduced parametric transistor variation in cmos logic 0.13 μm technology. IEEE Transactions on Semiconductor Manufacturing 24 (2), 273 –279.

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

Jung, C., Pabst, D., Ham, M., Stehli, M., Rothe, M., 2014. An effective problem decomposition method for scheduling of diffusion processes based on mixed integer linear programming. IEEE Transactions on Semiconductor Manufacturing 27 (3), 357–363.

Kao, Y.-T., Dauzère-Pérès, S., Blue, J., Chang, S.-C., 2018. Impact of integrating equipment health in production scheduling for semiconductor fabrication. Computers & Industrial Engineering.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., May 1983. Optimization by simulated annealing. Science 220 (4598), 671 –680.

Klemmt, A., Moench, L., 2012. Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing. In: Winter Simulation Conference 2012 (WSC 2012). Winter Simulation Conference, p. 194.

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

Li, L., Qiao, F., 2008. The impact of the qual-run requirements of APC on the scheduling performance in semiconductor manufacturing. In: Proceedings of 2008 IEEE International Conference on Automation Science and Engineering (CASE). pp. 242–246.

Maravelias, C. T., Grossmann, I. E., 2004. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. Computers & Chemical Engineering 28 (10), 1921 – 1949, special Issue for Professor Arthur W. Westerberg.

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

Moyne, J., Del Castillo, E., Hurwitz, A. M., 2000. Run-to-run control in semiconductor manufacturing. CRC press.

Musacchio, J., Rangan, S., Spanos, C., Poolla, K., 1997. On the utility of run to run control in semiconductor manufacturing. In: Proceedings of 1997 IEEE International Symposium on Semiconductor Manufacturing Conference. pp. 9–12.

Obeid, A., Dauzère-Pérès, S., Yugma, C., Feb 2014. Scheduling job families on non-identical parallel machines with time constraints. Annals of Operations Research 213 (1), 221–234.

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

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

Sadeghi, R., Dauzère-Pérès, S., Yugma, C., Lepelletier, G., 2015. Production control in semiconductor manufacturing with time constraints. In: 26th annual SEMI Advanced semiconductor manufacturing conference (ASMC 2015). IEEE, pp. 29–33.

Tan, F., Pan, T., Li, Z., Chen, S., 2015. Survey on run-to-run control algorithms in high-mix semiconductor manufacturing processes. IEEE Transactions on Industrial Informatics 11 (6), 1435–1444.

Teghem, J., May 2002. Optimisation Approchée en Recherche Opérationnelle : Recherche locale, Réseaux Neuraux et Satisfaction de Contraintes. Hermes Science Publications.

Van Hentenryck, P., 1999. The OPL Optimization Programming Language. MIT Press, Cambridge, MA, USA.

Wu, C.-H., Lin, J. T., Chien, W.-C., 2010. Dynamic production control in a serial line with process queue time constraint. International Journal of Production Research 48 (13), 3823–3843.

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

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