RAIRO-Oper. Res. 48 (2014) 255–269
DOI: 10.1051/ro/2014005

RAIRO Operations Research
www.rairo-ro.org

SOLVING MULTI-AGENT SCHEDULING PROBLEMS ON PARALLEL MACHINES WITH A GLOBAL OBJECTIVE FUNCTION

F. Sadi[1], A. Soukhal[1] and J.-C. Billaut[1]

Abstract. In this study, we consider a scheduling environment with m ($m \geq 1$) parallel machines. The set of jobs to schedule is divided into K disjoint subsets. Each subset of jobs is associated with one agent. The K agents compete to perform their jobs on common resources. The objective is to find a schedule that minimizes a global objective function f^0, while maintaining the regular objective function of each agent, f^k, at a level no greater than a fixed value, ε_k ($f^k \in \{f^k_{\max}, \sum f^k\}, k = 0, \ldots, K$). This problem is a *multi-agent scheduling problem with a global objective function*. In this study, we consider the case with preemption and the case without preemption. If preemption is allowed, we propose a polynomial time algorithm based on a network flow approach for the unrelated parallel machine case. If preemption is not allowed, we propose some general complexity results and develop dynamic programming algorithms.

Keywords. Scheduling, multi-agent, complexity, dynamic programming.

Mathematics Subject Classification. 90C39.

1. Introduction

The multi-criteria scheduling problem has been widely studied during the last ten years (see [17, 29]). Traditionally, these problems are modeled by assuming

Received September 11, 2013. Accepted November 28, 2013.

[1] Université François-Rabelais de Tours, CNRS, LI EA 6300, OC ERL CNRS 6305, 64 avenue Jean Portalis, 37200 Tours, France.
{faiza.sadi, ameur.soukhal, jean-charles.billaut}@univ-tours.fr

Article published by EDP Sciences

that each job has the same criteria. In practice, this assumption may not be valid. Instead of using one or more criteria for the entire set of jobs, we may need to consider that some subsets of jobs are evaluated on individual criteria. Each subset is associated with one *agent*. For example, it is possible to consider a workflow where jobs have the following particulars: some jobs may have a soft due date with allowed tardiness (to be minimized); whereas some other jobs may have hard due dates (that must be respected) and still other jobs may have no due date (production for stock).

We analyse this problem using two cases. First, the sets of jobs associated with the agents are all disjoint. This specific problem has been addressed in the literature as the *multi-agent scheduling problem*. The terminology was introduced in [1, 3]. Other terms have been used in the literature for this type of problem, *e.g.*, the *interfering job set problem* [14] or [6]. For the second case, we assume that the subsets of agent's jobs are not disjoint. These problems, *i.e.*, multi-agent scheduling problems with a global objective function, have also been addressed in the literature [19]. Those authors assume that there are several agents with disjoint subsets of jobs plus one global agent for the entire set of jobs.

In this study, we are primarily interested in multi-agent scheduling problems, with a global objective function. These problems are related to the literature on multi-agent scheduling problems. Section 2 introduces the problem definition and notation. A literature survey on related problems is summarized in Section 3. Problems with preemption are discussed in Section 4 and problems without preemption in Section 5. Conclusions and directions for future research are presented in Section 6.

2. Problem statement and notation

We consider the problem of scheduling n jobs on m parallel machines with K agents and a global objective function. The machines are not necessarily identical.

We denote $\mathcal{N}$ as the set of all the jobs ($|\mathcal{N}| = n$), and $\mathcal{N}_k$ as the job set of agent k, with $|\mathcal{N}_k| = n_k$, $k = 1 \ldots, K$. For example, if $K = 2$, the jobs of agent 2 are numbered from $n_1 + 1$ to $n_1 + n_2 = n$.

The positive integer processing time of job J_i on machine M_j is denoted by $p_{i,j}$ $i = 1, \ldots, n$, $j = 1, \ldots, m$ (p_i if the machines are identical). We assume that all jobs are available at time zero. The machines are always available and can only process one job at a time. Conversely, a job cannot be processed on more than one machine at a time.

We denote C_i as the completion time of job J_i. For each agent k ($k = 1, \ldots, K$), we denote $f^k_{\max}$ as a linear monotonically increasing function defined by $f^k_{\max} = \max_{i \in \mathcal{N}_k}\{f^k(C_i)\}$, where f^k does not depend on the job $J_i \in \mathcal{N}_k$ and $f^k_{\max}$ is the min-max function of agent k. Similarly, we denote $\sum f^k$ as the min-sum function of agent k defined by $\sum f^k = \sum_{i \in \mathcal{N}_k} f^k(C_i)$ and $f^0_{\max}$ ($\sum f^0$, respectively) as the global min-max objective function (min-sum, respectively). For example, $C^0_{\max} = \max_{1 \leq i \leq n} C_i$ is the global makespan, $\sum C^0_i = \sum_{i=1}^{n} C_i$ is the global sum

of completion times, and $\sum C_i^1 = \sum_{i=1}^{n_1} C_i$ is the sum of completion times of the jobs of agent 1.

According to the three-field notation $\alpha|\beta|\gamma$ of scheduling problems introduced in [16] and extended in [29], the problems that we consider in the following are expressed as follows: $\alpha|\beta|f^0, f^1, \ldots, f^K$, with $\alpha \in \{1, P, R\}$, $\beta \in \{\emptyset, pmtn\}$ and $(f^0, f^1, \ldots, f^K) \in \{(f^0_{\max}, f^1_{\max}, \ldots, f^K_{\max}), (\sum f^0, \sum f^1, \ldots, \sum f^K)\}$. The approach that is used is the ε-constraint approach expressed as follows: $\alpha|\beta|\varepsilon(f^0/f^1, \ldots, f^K)$ or $\alpha|\beta, f^1 \leq \varepsilon^1, \ldots, f^K \leq \varepsilon^K|f^0$. In this expression, the objective is to minimize f^0 and the other objectives are bounded.

If $\mathcal{N}_k = \mathcal{N}$, $k = 1, \ldots, K$, the problem is reduced to the classical multi-criteria scheduling problem. If $K = 0$, the problem becomes a single objective problem. This implies that if a multi-criteria scheduling problem or a classical single objective scheduling problem is $\mathcal{NP}$-hard, then the corresponding problem with one or more agents and a global objective function is also $\mathcal{NP}$-hard [19].

3. State-of-the-art literature survey

3.1. Single machine

Dealing with multi-agent scheduling problems (with no global function), Agnetis, Mirchandani, Pacciarelli and Pacifici pioneered the concept of multi-agent scheduling [3]. Their initial motivation was to solve a problem proposed by two companies, interested in a joint venture, to construct a modern flexible manufacturing system [1]. In this paper, they considered two agents, each one related to one company. The agents have disjoint subsets of jobs which are in conflict because they share the same resources and their cost functions are quasi-convex. The analyses led to a polynomial algorithm that generated a set of non-dominated solutions. As far as we know, before this article, no other work mentioned a scheduling problem with interfering job sets.

The single machine problem is also considered in [5]. The authors investigate regular objective functions ($C_{\max}, \sum w_j C_j, L_{\max}$) and propose an algorithm for the minimization of a linear combination of the objective functions. Complexity results are also given and polynomially solvable cases are identified. The authors in [31] propose some complementary results for these problems, summarizing the results presented in [5,31]. In the work by Agnetis *et al.* [3], the scheduling environments are a single machine flow shop and an open shop, with two disjoint subsets of jobs. Each agent associated with one subset wants to minimize a regular nondecreasing objective function, which depends only on the completion time of its jobs. The objective functions considered are regular min-max functions, the number of late jobs and the total weighted completion time. For various combinations of the two agents cost functions, the problem of generating non-dominated schedules is addressed and some complexity results are given. Furthermore, they proposed a dynamic programming algorithm for the problem with several agents, where each agent tries to minimize the total weighted completion time of its own jobs.

Chang *et al.* [9] consider a multi-agent scheduling problem on a single machine, where each agent wants to minimize the total weighted number of tardy jobs. They showed that the problem is strongly NP-hard. When the number of agents is fixed, they showed that the problem can be solved in pseudo-polynomial time for integral weights and in polynomial time for unit weights.

Chang *et al.* [10] consider a single machine problem with multiple agents and min-max objective functions (problem denoted by $1|f^k_{\max} \leq \varepsilon_k|-$). They show that a feasible problem can be solved in polynomial time even if the jobs are subjected to precedence constraints. They also identify some NP-hard problems.

Agnetis *et al.* [2] investigate the complexity of some scheduling problems in which several agents have to negotiate the usage of a common processing resource. The cost functions considered in their study are the maximum of regular functions (associated with each job), the number of late jobs and the total weighted completion time. The complexity of various problems resulting in combinations of cost functions was also addressed. Additionally, they investigated the problem of finding the set of all non-dominated solutions.

Parallel machines

The multi-agent scheduling problems, where the agents compete for the usage of more than one machine, have received increasing attention in the literature. In [30], the authors consider a two-agent scheduling problems, where the objective function of one agent is always of the max type, while several criteria are considered for the second agent (*e.g.*, $T^2_{\max}$ and $L^2_{\max}$). The authors prove the NP-hardness for the more general problems and propose polynomial time algorithms for other problems.

Balasubramanian, Fowler *et al.* [6] treat the scheduling problem on parallel machines with two agents, agent 1 and agent 2. Each agent has its own criteria to be minimized, *i.e.*, $C_{\max}$ for agent 1 and $\sum C_i$ for agent 2. The goal is to determine an optimal solution. Given that the problem is NP-hard, the authors propose an iterative SPT^2-LPT^1-SPT^2 heuristic (Shortest Processing Time first for the jobs of agent 2 and Longuest Processing Time first for agent 1) and a time-indexed integer programming formulation. To generate all non-dominated solutions, the authors propose a bicriteria genetic algorithm.

Lee *et al.* [22] solve a two-machine flow-shop scheduling problem with two agents, where the objective is to minimize the total tardiness for agent 1 with the restriction that the number of tardy jobs for agent 2 is zero. They provide several dominance properties to accelerate the search of the optimal solutions by the branch-and-bound algorithm. They also propose a simulated annealing heuristic algorithm to produce near-optimal solutions.

More recently, Elvikis and T'kindt [15] address the two-agent scheduling problem with equal-sized jobs on uniform parallel machines, where both agents try to minimize an arbitrary function, $f_{\max}$. They investigate the enumeration of strict Pareto optima solutions and propose an $O(n_1^2+n_2^2+n_1n_2log(n_2))$ time complexity

algorithm, where n_1 and n_2 equal the number of jobs of agent 1 and agent 2, respectively.

Since the early studies in multi-agent scheduling, few results are related to multi-agent problems with a global objective function. In [27, 28], the authors propose exact methods for identical parallel machine problems with one agent's objective function equal to $\sum C_i^1$ and the global function equal to $\sum C_i^0$. Huynh *et al.* [19] address the complexity of several single-machine problems in which the agents compete to perform their objectives, knowing that they all have an impact on the global objective function.

In the context of grid computing, Cordeiro *et al.* [13] consider organizations that share clusters to distribute peak workloads among all the participants. Each cluster is associated with one agent and the global objective function is to minimize the makespan. The authors propose a 2-approximation algorithm for finding collaborative solutions.

4. Parallel machines with preemption

In this section, it is assumed that preemption is allowed. That is, job processing can be interrupted and resumed later, eventually on another machine. We consider two cases. In the first case, the objective functions of the agents are of the min-max type, whereas in the second case, they are of the min-sum type. The problems are described by $Rm|pmtn|\varepsilon(f^0_{\max}/f^1_{\max}, \ldots, f^K_{\max})$ and $Pm|pmtn|\varepsilon(\sum f^0/\sum f^1, \ldots, \sum f^K)$.

4.1. Min-max functions

We consider m unrelated parallel machines and K agents with disjoint job sets. We have $\mathcal{N}_h \cap \mathcal{N}_\ell = \emptyset$, $\forall h \neq \ell$, $l \neq 0$ and $h \neq 0$.

For every agent, the ε-constraint states that $f^k_{\max} \leq \varepsilon_k$, which is equivalent to the following: $f^k(C_i) \leq \varepsilon_k$, $\forall i \in \mathcal{N}_k$. We assume that the inverse function $(f^k)^{-1}$ is known and can be computed in polynomial time, *i.e.* $C_i \leq (f^k)^{-1}(\varepsilon_k)$, $\forall i \in \mathcal{N}_k$. Therefore, $(f^k)^{-1}(\varepsilon_k)$ is a common deadline for the jobs in $\mathcal{N}_k$. We denote this quantity by $\tilde{d}_k$ in the following. We assume that $\max_{1\leq j\leq m}(p_{i,j}) \leq \tilde{d}_k$, $\forall i \in \mathcal{N}_k$.

To solve this problem, we use a two-phase exact approach. In the first phase, we apply a linear program, which takes into consideration all the constraints of the feasible problem, denoted by $Rm|pmtn, f^1_{\max} \leq \varepsilon_1, \ldots, f^K_{\max} \leq \varepsilon_K|-$. This linear program returns the proportion of each job to execute on each machine. It also returns the optimal function value $f^0_{\max}$. In the second phase, using a max-flow approach, we calculate the schedule of jobs to execute on each machine.

Phase 1. Assignment of jobs to machines

Assume the agents are numbered in $\tilde{d}_k$ non-decreasing order. The decision variable $x_{i,j}$ ($\forall i = 1, \ldots, n$ and $j = 1, \ldots, m$) is the proportion of processing time units of job J_i executed on machine M_j. Then, $f^0_{\max}$ is a continuous variable associated

with the optimal value of the global objective function. The linear program we apply is the following:

$$(P1)\ \min \quad f^0_{\max}$$

$$s.t. \quad \sum_{j=1}^{m} x_{i,j} = 1, \qquad \forall i \in \mathcal{N}_k, \forall k = 1, \ldots, K \tag{4.1}$$

$$f^0_{\max} - f^0\left(\sum_{k=1}^{K} \sum_{i \in \mathcal{N}_k} p_{i,j} x_{i,j}\right) \geq 0, \qquad \forall j = 1, \ldots, m \tag{4.2}$$

$$f^0_{\max} - f^0\left(\sum_{j=1}^{m} p_{i,j} x_{i,j}\right) \geq 0, \qquad \forall i \in \mathcal{N}_k, \forall k = 1, \ldots, K \tag{4.3}$$

$$\varepsilon_k - f^k\left(\sum_{k'=1}^{k} \sum_{i \in \mathcal{N}_{k'}} p_{i,j} x_{i,j}\right) \geq 0, \qquad \forall k = 1, \ldots, K, \forall j = 1, \ldots, m \tag{4.4}$$

$$\varepsilon_k - f^k\left(\sum_{j=1}^{m} p_{i,j} x_{i,j}\right) \geq 0, \qquad \forall i \in \mathcal{N}_k, \forall k = 1, \ldots, K \tag{4.5}$$

$$f^0_{\max} \geq 0, \text{ and } x_{i,j} \in [0,1], \qquad \forall i \in \mathcal{N}_k, \forall k = 1, \ldots, K, \quad \forall j = 1, \ldots, m \tag{4.6}$$

Constraints (4.1) require that every job J_i is completely assigned to the machines. Constraints (4.2) require that the total processing time of the jobs assigned to machine M_j is less than or equal to $f^0_{\max}$. Constraints (4.3) require that the total processing time of each job, performed on several machines, is less than or equal to $f^0_{\max}$. Constraints (4.4) and (4.5) satisfy all the ε-constraints. Constraints (4.4) require that the sum of job processing times related to agents 1 to k and assigned to machine M_j is less than or equal to $\tilde{d}_k$. Constraints (4.5) guarantee that the total processing time of job $J_i \in \mathcal{N}_k$ is less than or equal to $\tilde{d}_k$.

We claim that if problem $(P1)$ does not have a feasible solution, then the main problem does not have a feasible solution either. In other words, for each feasible schedule verifying (4.1)−(4.6), it is possible to identify a schedule where there is no job overlapping, no machine overbooking and the total processing amount over all the machines of the jobs are respected. The value of the optimal solution of problem $(P1)$ is denoted by $f^{0*}_{\max}$. Note that $f^{0*}_{\max}$ is also the optimal makespan for the problem denoted by $Rm|pmtn, f^1_{\max} \leq \varepsilon_1, \ldots, f^K_{\max} \leq \varepsilon_K | f^0_{\max}$.

Phase 2. Construction of a feasible solution

Every quantity $x_{i,j}$ is the ratio of job J_i that must be performed on machine M_j. Therefore, the remaining problem is equivalent to a preemptive open-shop scheduling problem, where tardy jobs are not allowed (problem denoted by $Om|pmtn, \tilde{d}_i | f^0_{\max} \leq f^{0^*}{}_{\max}$, where $f^{0^*}{}_{\max}$ is the $C^0_{\max}$-value returned by $P1$), which

is again equivalent to the feasible open shop problem with preemption and deadlines, denoted by $Om|pmtn, \tilde{d}_i|-$. Later on, the quantities $x_{i,j}$ lead to "tasks" $o_{i,j}$ of job J_i of duration $p_{i,j} \times x_{i,j}$ to be performed on machine M_j.

The problem of finding the feasible preemptive open shop schedules with deadlines has been solved by Cho and Sahni [12].

The deadline for the jobs of agent k ($k = 1, \ldots, K$) are defined as follows: $\tilde{d}_k = \min\{(f^k)^{-1}(\varepsilon_k), f^{0*}_{\max}\}$, and $\tilde{d}_0 = f^{0*}_{\max}$. Let $\gamma_0 < \gamma_1 < \ldots < \gamma_H$ be the ordered sequence of all the different values of $\tilde{d}_k$, $k = 0, \ldots, K$ (assuming $\gamma_0 = 0$).

The processing time of the task $o_{i,j}$ that can be scheduled during the interval $[\gamma_{h-1}, \gamma_h[$ is denoted as $q_{i,j,h}$ and the length of this interval is denoted as $I_h = \gamma_h - \gamma_{h-1}$ ($h = 1, \ldots, H-1$, $[\gamma_{H-1}, \gamma_H]$ for the last interval). Consider the following system of linear constraints:

$$
\begin{aligned}
(P2)\quad & \sum_{i=1}^{n} q_{i,j,h} \leq I_h, && \forall h = 1, \ldots, H, \forall j = 1, \ldots, m && (4.7)\\
& \sum_{j=1}^{m} q_{i,j,h} \leq I_h, && \forall h = 1, \ldots, H, \forall i = 1, \ldots, n, && (4.8)\\
& \sum_{h=1}^{H} q_{i,j,h} = p_{i,j} x_{i,j}, && \forall i = 1, \ldots, n, \forall j = 1, \ldots, m && (4.9)\\
& q_{i,j,h} = 0 && \forall i = 1, \ldots, n, \forall j = 1, \ldots, m, && \\
& && \forall h = 1, \ldots, H \text{ with } \tilde{d}_k \notin [\gamma_{h-1}, \gamma_h[&& (4.10)\\
& q_{i,j,h} \geq 0 && \forall i = 1, \ldots, n, \forall j = 1, \ldots, m, && \\
& && \forall h = 1, \ldots, H && (4.11)
\end{aligned}
$$

Constraints (4.7) ensure that the amount of processing time assigned to each machine and during each time interval cannot exceed the interval length. Constraints (4.8) avoid any overlapping of tasks on the machines. Constraints (4.9) guarantee the assignment of the total tasks of jobs to the machines. Constraints (4.10) guarantee the assignment of tasks in their interval.

Using the solution of (P2), the last step is to plan the sequence of tasks on each machine. We apply Brucker's approach ([8], see Algorithm 1) to each interval, *i.e.*, each interval h, where $h = 0, \ldots, H$, is associated with a bipartite graph $G_h = (\mathcal{N}, \mathcal{M}, E, \phi_h)$, where $\mathcal{N}$ is the set of job nodes, $\mathcal{M}$ is the set of machine nodes and E is the set of edges (J_i, M_j) for $i \in \mathcal{N}$, $j \in \mathcal{M}$ and $\phi_h(i, j) = q_{i,j,h}$ ($i = 1, \ldots, n$, $j = 1, \ldots, m$, $h = 1, \ldots, H$) the weight of arc (J_i, M_j).

Using matching, at each iteration, the procedure selects δ processing time units for different jobs that can be scheduled during the interval $[\gamma_{h-1}, \gamma_h[$ and constitutes the matching solution R during interval I_h. Hence, we derive the following: $\delta = \min_{(J_i, M_j) \in R} q_{i,j,h}$. These time units are assigned to the machines. This technique avoids an overlapping of tasks in the final schedule.

Counting the number of iterations. At each iteration of the matching procedure for interval h, at least one arc (J_i, M_j) of the maximum matching is such that $\delta = \phi_h(i, j)$. Hence for interval h, there are at most $n \times m$ iterations. Because $H \leq K$ the procedure runs in $O(nmK)$.

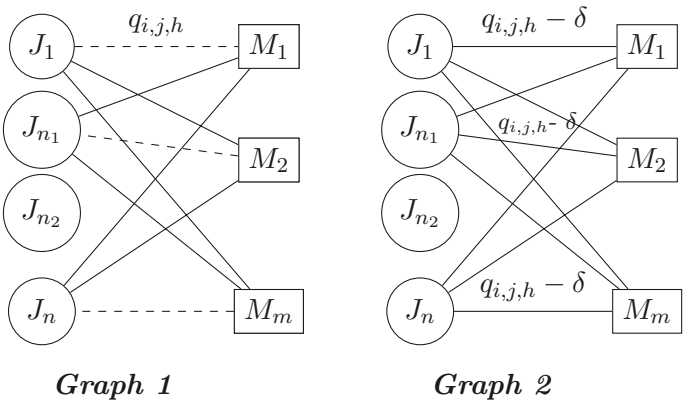

FIGURE 1. Graph G_h at Step 1 and at Step 2.

Algorithm 1 Matching procedure for a feasible solution [8]

Input: a graph G_h.
while there is an arc in the graph G_h **do**
Seek for the maximum matching R in G_h (see Fig. 1, Graph 1);
Let $\delta = \min_{(J_i,M_j)\in R} \phi_h(i,j)$;
For each $(J_i, M_j) \in R$, schedule δ processing time units of J_i at the end of machine M_j;
Reduce the capacity $\phi_h(i,j)$ of all arcs in R by δ (see Fig. 1, Graph 2);
Eliminate all arcs with zero capacity.
end while

The matching can be calculated in $O(nm\sqrt{n+m})$ by using the algorithm described in [18]. Thus, Step 2 runs in $O(n^2m^2K\sqrt{n+m})$ time.

4.2. Min-sum functions

Consider m identical parallel machines and K agents with disjoint job sets. For each agent k, we define a monotonically increasing function $\sum f^k = \sum_{i\in\mathcal{N}_k} f^k(C_i)$, $k = 1,\ldots,K$, where f^k does not depend on the job $J_i \in \mathcal{N}_k$. The problem is denoted by $Pm|pmtn|\varepsilon(\sum f^0/\sum f^1,\ldots,\sum f^K)$.

To prove that this problem is NP-hard, we prove that $1|pmtn|\varepsilon(\sum C_i^0/\sum C_i^1)$ is NP-hard.

Let S be a feasible schedule for the $1|pmtn|\varepsilon(\sum C_i^0/\sum C_i^1)$ problem, where job J_i is preempted (see Fig. 2). We have $\sum C_i^1(S) \le \varepsilon_1$. Let $\pi_1/i/\pi_2/i/\pi_3$ be sequence S where π_1, π_2 and π_3 are sub-sequences of jobs and notation a/b stands for the concatenation of a and b. We denote the duration of job J_i before sequence π_2 as p_{i_1}. Let S' be the same solution where J_i is not preempted (*i.e.*, shifted to the right). Therefore, $S' = \pi_1/\pi_2/i/\pi_3$.

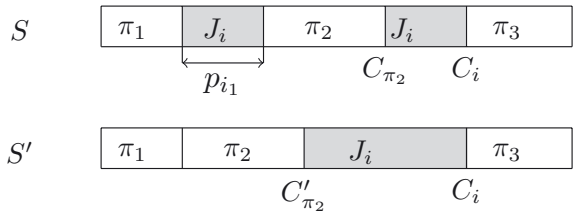

FIGURE 2. Preemption of job J_i.

Shifting J_i to the right does not modify its completion time, and allows the jobs of π_2 to complete p_{i_1} earlier time units. If $J_i \in \mathcal{N}_1$, then $\sum C_i^1(S) \leq \varepsilon_1$. Because the completion of J_i does not change and because the completion times for the jobs of $\mathcal{N}_1$ in π_2 (if any) only decrease (the completion times of other jobs does not change), the following applies: $\sum C_i^1(S') \leq \varepsilon_1$. The same reasoning applies if $J_i \in \mathcal{N} \setminus \mathcal{N}_1$.

In that case, we can state:

$$\sum C_i^1(S') \leq \sum C_i^1(S) \leq \varepsilon_1$$

$$\sum C_i^0(S') = \sum C_i^0(S) - p_{i_1}|\pi_2| < \sum C_i^0(S).$$

Therefore, S' dominates S and there is no need to preempt a job to minimize the sum of completion times; thus there is no preemption in any optimal solution.

We conclude that problem $1|pmtn|\varepsilon(\sum C_i^0/\sum C_i^1)$ is NP-hard. Suppose that this problem can be solved in polynomial time by Algorithm A. Then, A could be used to solve problem $1||\varepsilon(\sum C_i^0/\sum C_i^1)$, because the optimal solution returned by A has no preemption. As the scheduling problem $1||\varepsilon(\sum C_i^0/\sum C_i^1)$ is NP-hard in the ordinary sense [19], this is not possible, which provides the proof. Therefore the problem $Pm|pmtn|\varepsilon(\sum f^0/\sum f^1, \ldots, \sum f^K)$ is also NP-hard.

5. Parallel machines without preemption

In this section, we first describe the case of a single machine with min-max functions, which can be solved in polynomial time (with min-sum functions, the problem is NP-hard [19]). Then, we prove that the problems with identical parallel machines are NP-hard for any objective function. Finally, we propose dynamic programming algorithms in the case of one agent, when the objective functions f^0 and f^1 belong to $\{C_{\max}, \sum C_i\}$.

5.1. A single machine with min-max function

Consider the $1||\varepsilon(f^0_{\max}/f^1_{\max}, \ldots, f^K_{\max})$ single machine multi-agent scheduling problem. Because the objective functions are regular, the optimal schedule of this

problem has no idle time. Therefore, the problem can be solved by the Lawler procedure in $O(n^2)$ units of time.

Algorithm 2 Backward algorithm for the $1||\varepsilon(f^0_{\max}/f^1_{\max}, \ldots, f^K_{\max})$

$P = \sum_{i=1}^{n} p_i$
$\mathcal{S} = \{J_i \in \mathcal{N} / \tilde{d}_i \geq P\}$
while $\mathcal{S} \neq \emptyset$ **do**
  Select the job J_j in $\mathcal{S}$ such that $f^0_j = \min_{J_i \in \mathcal{S}} f^0_i$
  // Schedule J_j so that it completes at time P //
  $P = P - p_j$
  Update $\mathcal{S}$ (delete J_j and insert new jobs that can complete at time P)
end while

Consider the deadline $\tilde{d}_k$ defined in Section 4.1 as follows: $d_k = (f^k)^{-1}(\varepsilon_k)$ with $k = 1, \ldots, K$. If the inverse function $(f^k)^{-1}$ is available, the deadline $\tilde{d}_k$ can be computed in constant time, otherwise the value of $\tilde{d}_k$ is obtained by binary search. In the following, we assume that all $\tilde{d}_k$ can be readily computed. This situation is isomorphic to the problem with ready times associated with the jobs. It is possible to obtain an optimal solution by applying a backward algorithm (similar to the one described in [20] for the $1|prec|f_{\max}$ problem), starting at time $t = \sum_{i=1}^{n} p_i$. The algorithm is described in Algorithm 2. P is the completion time of the next job to schedule and $\mathcal{S}$ contains the set of candidate jobs that can be completed at time P.

5.2. Complexity analysis

In this section, we introduce some general reductions. If the decision problem π reduces to π', we use the notation $\pi \propto \pi'$. Additionally, Γ_1 and Γ_2 are two subsets of regular criteria defined by $\Gamma_1 = \{C_{\max}, L_{\max}, \sum U_i, \sum T_i, \sum w_i T_i, \sum w_i U_i\}$, $\Gamma_2 = \{\sum C_i, \sum w_i C_i\}$, and $\Gamma = \Gamma_1 \cup \Gamma_2$.

Proposition 5.1. *For all $f^1 \in \Gamma$ the following reductions from the classical scheduling problem hold:*

1. $Pm||C^0_{\max} \propto Pm||\varepsilon(C^0_{\max}/f^1)$, $\forall f^1 \in \Gamma$.
2. $Pm||\varepsilon(C^0_{\max}/f^1) \propto Pm||\varepsilon(f^0/f^1)$, $\forall f^0 \in \Gamma^1$, $\forall f^1 \in \Gamma$.

Proof. These claims are direct consequences of the complexity of the $Pm||C_{\max}$ problem, if the boundary ε-value is sufficiently large for the problem. □

Proposition 5.2. The scheduling problems $Pm||\varepsilon(f^0/f^1)$ are NP-hard, $\forall f^0$ and $f^1 \in \Gamma$.

Proof. We turn out to the decision version of the problems. We first show that $Pm|f^0 \leq \varepsilon_0, f^1 \leq \varepsilon_1|-$, $\forall f^0 \in \Gamma^1$, $\forall f^1 \in \Gamma$, is NP-complete. Then, we show that $Pm|f^0 \leq \varepsilon_0, f^1 \leq \varepsilon_1|-$, $\forall f^0 \in \Gamma^2$, $\forall f^1 \in \Gamma$, is NP-complete.

The first claim is given by Step 2 of Proposition 5.1.

It is known that $Pm|C^1_{\max} \leq \varepsilon_1|-$ is NP-complete. Hence $Pm|C^1_{\max} \leq \varepsilon_1, f^0 \leq \varepsilon_0|-$ is NP-complete, $\forall f^0 \in \Gamma$. This is also true for $f^0 = \sum C^0_i$. And because it is true for $C^1_{\max}$, it is also true for any $f^1 \in \Gamma_1$. Thus, $Pm|f^1 \leq \varepsilon_1, \sum C^0_i \leq \varepsilon_0|-$, $\forall f^1 \in \Gamma_1$ is NP-complete. We know that $Pm|\sum C^1_i \leq \varepsilon_1, \sum C^0_i \leq \varepsilon_0|-$ is NP-complete (see [19]). So $Pm|f^1 \leq \varepsilon_1, \sum C^0_i \leq \varepsilon_0|-$ is NP-complete $\forall f^1 \in \Gamma$. This is also true for $\sum w_i C^0_i$, *i.e.* $Pm|f^1 \leq \varepsilon_1, f^0 \leq \varepsilon_0|-$, $\forall f^1 \in \Gamma$, $\forall f^0 \in \Gamma_2$. □

5.3. Dynamic programming algorithms for problems with $(f^0, f^1) \in \{C_{\max}, \sum C_i\}^2$

To illustrate our approach, consider one agent and a global objective function scheduling problem, where f^1 and f^0 are in $\{C_{\max}, \sum C_i\}$. We solve problems $Pm||\varepsilon(f^0/f^1)$ using dynamic programming algorithms.

$1 \leq i \leq n_1$ are the subscripts of the jobs in $\mathcal{N}_1$, $n_1+1 \leq j \leq n$ are the subscripts of the jobs in $\mathcal{N} \setminus \mathcal{N}_1$. All jobs in $\mathcal{N}_1$ are numbered in SPT order and all jobs in $\mathcal{N} \setminus \mathcal{N}_1$ are also in SPT order as follows: $p_1 \leq p_2 \leq \ldots \leq p_{n_1}$ and $p_{n_1+1} \leq \ldots \leq p_n$.

The parameters used to define the dynamic programs are the following: P_j is the makespan on machine M_j, Q_1 is the total completion time of jobs in $\mathcal{N}_1$ and P_{i_1,i_2} is the total processing time of the first i_1 jobs in $\mathcal{N}_1$ plus the $(i_2 - n_1)$ first jobs in $\mathcal{N} \setminus \mathcal{N}_1$. For this problem, $UB = \sum_{i=1}^{n} p_i$ is an upper bound for P_1, and $LB = \max\{\frac{1}{m}\sum_{i=1}^{n} p_i, \max_{1\leq i\leq n} p_i\}$ represents the lower bound.

5.4. Problem with $f^0 = C^0_{\max}$

Problem 5.3. Consider that $C^1_{\max}$ is also the agent objective function. The problem is denoted by $Pm||\varepsilon(C^0_{\max}/C^1_{\max})$. It is equivalent to the mono-objective problem $Pm|\tilde{d}_i|C_{\max}$, where $\tilde{d}_{i_1} = \varepsilon$, $\forall i_1 \in \mathcal{N}_1$ and $\tilde{d}_{i_2} = UB$, $\forall i_2 \in \mathcal{N} \setminus \mathcal{N}_1$.

We define the following recursive function (see [7]) as follows: $F_{i_1,i_2}(t_1, \ldots, t_m)$ is *true* if jobs $1, \ldots, i_1$ of $\mathcal{N}_1$ and $n_1+1, \ldots, i_2$ of $\mathcal{N} \setminus \mathcal{N}_1$ can be scheduled on $M_1, \ldots, M_m$ in such a way that each machine M_j is busy in the interval $[0, t_j]$, and *false* otherwise.

Applying $F_{0,n_1}(t_1, \ldots, t_m) = false$ $\forall (t_1, \ldots, t_m) \in \{0, 1, \ldots, UB\}^m$ and $F_{0,n_1}(0, \ldots, 0) = true$, the recursive relation is given as follows:

$$F_{i_1,i_2}(t_1, \ldots, t_m) = \bigvee_{j=1}^{m} (F_{i_1-1,i_2}(t_1, \ldots, t_j - p_{i_1}, \ldots, t_m) \wedge (t_j \leq \varepsilon_1))$$

$$\vee \bigvee_{j=1}^{m} F_{i_1,i_2-1}(t_1, \ldots, t_j - p_{i_2}, \ldots, t_m)$$

$$\forall i_1, 1 \leq i_1 \leq n_1, \forall i_2, n_1+1 \leq i_2 \leq n, \forall t_j \in [0, UB], \forall j, 1 \leq j \leq m$$

Then, the optimal makespan value is given by

$$C^0_{\max} = \min\left(\max_{\forall t_j \in [0,UB]} (\{t_1, t_2, \ldots, t_m\} | F_{n_1,n}(t_1, \ldots, t_m) = true\right).$$

This algorithm runs in $O(n^2 UB^m)$.

Problem 5.4. Consider that $\sum C_i$ is the agent objective function. The problem is denoted by $Pm||\varepsilon(C^0_{\max}/\sum C^1_i)$.

We define the following recursive function: $F_{i_1,i_2}(t_1, \ldots, t_m, Q_1)$ is *true* if jobs $1, \ldots, i_1$ of $\mathcal{N}_1$ and $n_1 + 1, \ldots, i_2$ of $\mathcal{N} \setminus \mathcal{N}_1$ can be scheduled on $M_1, \ldots, M_m$ such that each machine M_j is busy in the interval $[0, t_j]$ and the sum of completion times of jobs in $1, \ldots, i_1$ is equal to Q_1, and *false* otherwise. Q_1 is the total completion time of jobs in $\mathcal{N}_1$, where $0 \leq Q_1 \leq \varepsilon_1$.

Applying $F_{0,0}(t_1, \ldots, t_m, Q_1) = false\ \forall (t_1, \ldots, t_m) \in \{0, 1, \ldots, UB\}^m; 0 \leq Q_1 \leq \varepsilon_1$ and $F_{0,n_1}(0, \ldots, 0) = true$, the recursive relation is given as follows:

$$F_{i_1,i_2}(t_1, \ldots, t_m, Q_1) = \bigvee_{j=1}^{m} \left(F_{i_1-1,i_2}(t_1, \ldots, t_j - p_{i_1}, \ldots, t_m, Q_1 - t_j) \wedge (Q_1 \leq \varepsilon_1)\right)$$

$$\vee \bigvee_{j=1}^{m} F_{i_1,i_2-1}(t_1, \ldots, t_j - p_{i_2}, \ldots, t_m, Q_1).$$

This dynamic programming algorithm determines the assignment of jobs to machines, which is sufficient to compute an optimal schedule.

The optimal makespan value is given by

$$C^0_{\max} = \min\left(\max_{\forall t_j \in [0,UB]} (\{t_1, t_2, \ldots, t_m\} | F_{n_1,n}(t_1, \ldots, t_m, Q_1) = true\right).$$

For a given upper bound ε_1 on the total completion time of agent jobs, this algorithm runs in $O(n^2 UB^m \varepsilon_1)$ time.

5.5. Problem with $f^0 = \sum C_i$

Problem 5.5. Consider the problem with $C_{\max}$ as the objective function of agent 1. The problem is denoted by $Pm||\varepsilon(\sum C^0_i / C^1_{\max})$. $F_{i_1,i_2}(P_1, \ldots, P_m)$ is the total completion time when i_1 jobs of $\mathcal{N}_1$ and $i_2 - n_1$ jobs of $\mathcal{N} \setminus \mathcal{N}_1$ are scheduled on m machines.

$$F_{0,n_1}(0, \ldots, 0) = 0; \quad F_{0,n_1}(P_1, \ldots, P_m) = +\infty, \forall (P_1, \ldots, P_m) \neq (0, \ldots, 0)$$

$$F_{i_1,i_2}(P_1, \ldots, P_m) = +\infty, \text{ if } i_1 \notin [1, n_1] \text{ or } i_2 \notin [n_1 + 1, n] \text{ or } P_j \notin [0, UB].$$

$$F_{i_1,i_2}(P_1,\ldots,P_m) =$$

$$\min_{j=1,\ldots,m,} \begin{cases} \begin{cases} F_{i_1-1,i_2}(P_1,\ldots,P_j-p_{i_1},\ldots,P_m)+P_j, & \text{if } P_j \le \varepsilon_1 \\ +\infty, & \text{if } P_j > \varepsilon_1 \end{cases} \\ F_{i_1,i_2-1}(P_1,\ldots,P_j-p_{i_2},\ldots,P_m)+P_j \end{cases}$$

$$\forall i_1 \in \{1,\ldots,n_1\}, \forall i_2 \in \{n_1+1,\ldots,n\}, \forall P_j \in [0,UB].$$

The optimal total completion time is given by

$$\sum C_i^0 = \min_{\forall P_j \in [0,UB]} F_{n_1,n}(P_1,\ldots,P_m).$$

This algorithm runs in $O(n^2UB^m)$ time.

Problem 5.6. We initially examine the problem $Pm||\varepsilon(\sum C_i^0/\sum C_i^1)$. In addition to the total completion time of jobs in $\mathcal{N}_1$, denoted Q_1, we must save the information related to the $C_{\max}$ on each machine, denoted as $P_{j,1\le j\le m}$. Let $F_{i_1,i_2}(P_1,\ldots,P_m,Q_1)$ be the total completion time when i_1 jobs of $\mathcal{N}_1$ and i_2-n_1 jobs of $\mathcal{N}\backslash\mathcal{N}_1$ are scheduled on m machines, where $Q_1 \le \varepsilon_1$.

$$F_{0,n_1}(0,\ldots,0)=0; \quad F_{0,n_1}(P_1,\ldots,P_m,Q_1)=+\infty, \forall (P_1,\ldots,P_m,Q_1) \ne (0,\ldots,0)$$

$$F_{i_1,i_2}(P_1,\ldots,P_m,Q_1)=+\infty, \text{ if } i_1 \notin [1,n_1] \text{ or } i_2 \notin [n_1+1,n] \text{ or } P_j \notin [0,UB] \text{ or } Q_1<0.$$

$$F_{i_1,i_2}(P_1,\ldots,P_m,Q_1) =$$

$$\min_{j=1,\ldots,m} \begin{cases} \begin{cases} F_{i_1-1,i_2}(P_1,\ldots,P_j-p_{i_1},\ldots,P_m,Q_1-P_j)+P_j, & \text{if } Q_1 \le \varepsilon_1 \\ +\infty, & \text{if } Q_1 > \varepsilon_1 \end{cases} \\ F_{i_1,i_2-1}(P_1,\ldots,P_j-p_{i_2},\ldots,P_m,Q_1)+P_j \end{cases}$$

$$\forall i_1 \in \{1,\ldots,n_1\}, \forall i_2 \in \{n_1+1,\ldots,n\}, \forall P_j \in [0,UB], \forall Q_1 \in [0,\varepsilon_1].$$

The optimal solution is obtained by $\min_{\forall P_j\in[0,UB],\forall Q_1\in[0,\varepsilon_1]} F_{n_1,n}(P_1,\ldots,P_m,Q_1)$ and the computational time is of $O(n^2UB^m\varepsilon_1)$.

Remark 5.7. The ideas of the DP algorithms can be extended to problems with K agents. We summarize the recursive functions and the complexity of the DP algorithms in Table 1, where $\varepsilon = max_{1\le k\le K}(\varepsilon_k)$ and i_{K+1} is index of job from $\mathcal{N}\backslash\bigcup_{k=1}^{K}\mathcal{N}_k$.

TABLE 1. Extensions of the DP algorithms to the case of K agents.

Problem	Generic function	Complexity
$Pm\|\|\varepsilon\left(C^0_{\max}/C^1_{\max},\ldots,C^K_{\max}\right)$	$F_{i_1,\ldots,i_K,i_{K+1}}(t_1,\ldots,t_m)$	$O(n^{K+1}UB^m)$
$Pm\|\|\varepsilon\left(C^0_{\max}/\sum C^1_i,\ldots,C^K_i\right)$	$F_{i_1,\ldots,i_K,i_{K+1}}(t_1,\ldots,t_m,Q_1,\ldots,Q_K)$	$O\left(n^{K+1}UB^m\varepsilon^K\right)$
$Pm\|\|\varepsilon\left(\sum C^0_i/C^1_{\max},\ldots,C^K_{\max}\right)$	$F_{i_1,\ldots,i_K,i_{K+1}}(P_1,\ldots,P_m)$	$O(n^{K+1}UB^m)$
$Pm\|\|\varepsilon(\sum C^0_i/\sum C^1_i,\ldots,\sum C^K_i)$	$F_{i_1,\ldots,i_K,i_{K+1}}(P_1,\ldots,P_m,Q_1,\ldots,Q_K)$	$O\left(n^{K+1}UB^m\varepsilon^K\right)$

6. Conclusions

Scheduling problems where K agents compete to perform their jobs on common parallel machines are studied. Two cases are considered, *i.e.*, with and without preemption. The objective is to find a schedule that minimizes the global objective function while keeping the regular objective function of each agent k no greater than a fixed value ε_k. The different combinations of the cost functions lead to various types of problems. Polynomial and pseudo-polynomial time algorithms are derived. The models derived in this study can be extended in various directions, such as to analyze other combinations of objective functions and search for the Pareto front. It would also be interesting to study the scheduling problem $Pm|pmtn, f^k_{\max}\leq\varepsilon_k|\sum f^0$, *i.e.*, when preemption is allowed.

References

[1] A. Agnetis, P. Mirchandani, D. Pacciarelli and A. Pacifici, Nondominated schedules for a job-shop with two competing users. *Comput. Math. Organ. Theor.* **6** (2000) 191–217.
[2] A. Agnetis, D. Pacciarelli and A. Pacifici, Multi-agent sincle machine scheduling. *Ann. Oper. Res.* **150** (2007) 3–15.
[3] A. Agnetis, P. Mirchandani, D. Pacciarelli and A. Pacifici, Scheduling problems with two competing agents. *Oper. Res.* **52** (2004) 229–242.
[4] A. Agnetis, G. Pascale and D. Pacciarelli, A Lagrangian approach to single-machine scheduling problems with two competing agents. *J. Scheduling* **12** (2010) 401–415.
[5] K.R. Baker and J.C. Smith, A multiple-criteria model for machine scheduling. *J. Scheduling* **6** (2003) 7–16.
[6] H. Balasubramanian, J. Fowler, A. Keha and M. Pfund, Scheduling interfering job sets on parallel machines. *Eur. J. Oper. Res.* **199** (2009) 55–67.
[7] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt and J. Weglarz, *Handbook on scheduling: From Theory to Applications*. International handbooks on information systems. Springer (2007).
[8] P. Brucker, *Scheduling algorithms*. Fifth Edition. Springer (2005).
[9] T.C.E. Cheng, C.T. Ng, J.-J. Yuan, Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theor. Comput. Sci.* **362** (2006) 273–281.
[10] T.C.E. Cheng, C.T. Ng and J.-J. Yuan, Multi-agent scheduling on a single machine with max-form criteria. *Eur. J. Oper. Res.* **188** (2008) 603–609.
[11] T.C.E. Cheng, S.-R. Cheng, W.-H. Wu, P.-H. Hsu and C.-C. Wu, A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning considerations. *Comput. Ind. Engrg.* **60** (2001) 534–541.

[12] Y. Cho and S. Sahni, Preemptive scheduling of independent jobs with release and due times on open, flow and job shops. *Oper. Res.* **29** (1981) 511–522.
[13] D. Cordeiro, P.-F. Dutot, G. Mounié and D. Trystram, Tight Analysis of Relaxed Multi-Organization Scheduling Algorithms. In Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS), Anchorage, AL, USA, *IEEE Comput. Soc.* (2011) 1177–1186.
[14] D. Elvikis, H.W. Hamacher and V. T'kindt, Scheduling two interfering job sets on uniform parallel machines with makespan and cost functions. *J. Scheduling* **14** (2011) 471–481.
[15] D. Elvikis and V. T'kindt, Two-agent scheduling on uniform parallel machines with min-max criteria. *Ann. Oper. Res.* (2012) 1–16.
[16] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.* **5** (1979) 287–326.
[17] H. Hoogeveen, Multicriteria scheduling. *Eur. J. Oper. Res.* **167**(2005) 59–623.
[18] J.E. Hopcroft and R.-M. Karp, A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.* **2** (1973) 22–231.
[19] N. Huynh Tuong, A. Soukhal and J.-C. Billaut, Single-machine multi-agent scheduling problems with a global objective function. *J. Scheduling* **15** (2012) 311–321.
[20] E.L. Lawler, Optimal sequencing of a single machine subject to precedence constraints. *Manage. Sci.* **19** (1973) 544–546.
[21] K. Lee, B.-C. Choi, J.Y.-T. Leung and M. Pinedo, Approximation algorithms for multi-agent scheduling to minimize total weighted completion time. *Inform. Process. Lett.* **16** (2009) 913–917.
[22] W.-C. Lee, S.-k. Chen and C.-C. Wu, Branch-and-bound and simulated annealing algorithms for a two-agent scheduling problem. *Exp. Syst. Appl.* **37** (2010) 6594–6601.
[23] J.Y.-T. Leung, M. Pinedo and G. Wan, Competitive two agent scheduling and its applications. *Oper. Res.* **58** (2007) 458–469.
[24] L. Peng, Y. Na and Z. Xiaoye, Two-agent single-machine scheduling problems under increasing linear deterioration. *Appl. Math. Model.* **35** (2011) 2290–2296.
[25] A. Sedeno-Noda, D. Alcaide and C. Gonza-Martin, Network flow approaches to pre-emptive open-shop scheduling problems with time-windows. *Eur. J Oper. Res.* **18** (2005) 1501–1518.
[26] R .Soltani, F. Jolai and M. Zandieh, Two robust meta-heuristics for scheduling multiple job classes on a single machine with multiple criteria. *Exp. Syst. Appl.* **37** (2010) 5951–5959.
[27] A. Soukhal, N. Huynh Tuong and Z. Dao, *Parallel machine scheduling with interfering jobs*, in 8th International Conference on Multiple Objective and Goal Programming (MOPGP'08), Portsmouth, UK (2008).
[28] A. Soukhal, N. Huynh Tuong and Z. Dao, Méthodes exactes et approchées pour l'ordonnancement de travaux interférant (in French), in *Int. Symposium on Oper. Res.*, ISOR'08 Algers, Algeria (2008).
[29] V. T'kindt and J.-C. Billaut, *Multicriteria scheduling.* Second Edition. Springer (2006).
[30] G. Wan, J.-Y. Leung and M. Pinedo, Scheduling two agents with controllable processing times. *Eur. J. Oper. Res.* **205** (2007) 528–539.
[31] J. Yuan, W.-P. Shang and Q. Feng, A note on the scheduling which two families of jobs. *J. Scheduling* **8** (2005) 537–542.