Ann Oper Res (2007) 150:3–15
DOI 10.1007/s10479-006-0164-y

Multi-agent single machine scheduling

Alessandro Agnetis · Dario Pacciarelli · Andrea Pacifici

Published online: 10 January 2007

Abstract We consider the scheduling problems arising when several agents, each owning a set of nonpreemptive jobs, compete to perform their respective jobs on one shared processing resource. Each agent wants to minimize a certain cost function, which depends on the completion times of its jobs only. The cost functions we consider in this paper are maximum of regular functions (associated with each job), number of late jobs and total weighted completion time. The different combinations of the cost functions of each agent lead to various problems, whose computational complexity is analysed in this paper. In particular, we investigate the problem of finding schedules whose cost for each agent does not exceed a given bound for each agent.

Keywords Multi-agent · Scheduling · Complexity · Multi-criteria

1 Introduction and multi-agent literature

This paper addresses a class of scheduling problems in which multiple agents compete on the usage of a common resource. This kind of problems occurs in several different application environments and different methodological fields, such as artificial intelligence, decision theory, operations research etc. We focus on the following situation. There are k agents, each

A. Agnetis (✉)
Dipartimento di Ingegneria dell'Informazione,
Università di Siena, Via Roma, 56 – 53100 Siena, Italy
e-mail: agnetis@dii.unisi.it

D. Pacciarelli
Dipartimento di Informatica e Automazione,
Università Roma Tre, Italy
e-mail: pacciare@dia.uniroma3.it

A. Pacifici
Dipartimento di Ingegneria dell'Impresa and Centro Vito Volterra,
Università di Roma "Tor Vergata", Italy
e-mail: andrea.pacifici@uniroma2.it

owning a set of jobs. The agents have to schedule their jobs on a common machine, and each agent wants to minimize a cost function which depends on its own jobs' completion times. The problem is how to compute schedules which account for each agent's cost function, and that can be used to support the negotiation among the agents.

Multi-agent scheduling problems occur in several application environments in which the need for negotiation/bidding procedures arises. Most of the papers on this subject investigate heuristic approaches for the construction of schedules that are acceptable to the agents, with no particular concern on optimality. For instance, Kim et al. (2000) discuss complex negotiation procedures for project scheduling in a multi-agent environment, allowing the parties to come up with new schedules whenever unacceptable task timings occur. Other approaches are based on distributed artificial intelligence. Huang and Hallam (1995) address a multi-agent scheduling problem in terms of a constraint satisfaction problem where a subset of constraints can be relaxed but are expected to be satisfied as well as possible. Chen et al. (1999) propose a number of negotiation protocols for functional agent cooperation in a supply chain context. Brewer and Plott (1996) devise a bidding mechanism for the problem of scheduling trains (agents) on a shared single rail track. Arbib, Servilio, and Smriglio (2004) discuss the problem of the integration of multimedia telecommunication services for a Satellite-based Universal Mobile Telecommunication System (S-UMTS). The problem here is to fulfill the requirements of various integrated services (agents), such as Voice over IP, web browsing, file transfer via ftp, etc. Different agents have different objectives. For instance, the voice service may tolerate the loss of some packets, but under strict delay requirements. On the contrary, when transferring a data file, no packet can go lost, but some delay can be tolerated.

From the theoretical viewpoint, multi-agent scheduling problems are a special case of general multicriteria optimization problems. For instance, the two-agent problem in which Agent 1 wants to minimize the total completion time of the jobs in J^1 and Agent 2 wants to minimize the maximum lateness of the jobs in J^2 can be viewed as a bicriteria, single-agent problem with objective functions $\sum_{j\in J^1\cup J^2} q_j^{(1)} C_j$ and $\max_{j\in J^1\cup J^2}\{q_j^{(2)} T_j\}$, where $q_j^{(1)} = 1$ and $q_j^{(2)} = 0$ for $j \in J^1$, whereas $q_j^{(1)} = 0$ and $q_j^{(2)} = 1$ for $j \in J^2$. Hence, in principle, general methods for multicriteria optimization can be applied, although these may not exploit the peculiarity of the problem. In fact, in classical single-agent, multicriteria scheduling problems, all jobs contribute to all criteria, whereas, in a multi-agent situation, only the jobs belonging to an agent contribute to that agent's criterion. So, two-agent scheduling problems differ from the problems commonly referred to as *bicriteria scheduling problems*, see, e.g., T'kindt and Billaut (2002). As a consequence, the complexity results known for a certain bicriteria scheduling problem in which there are two objectives f and g, in general do not imply similar complexity results for the corresponding two-agent problem in which the two agents have objectives f and g respectively. Moreover, note that some cases only make sense in the two-agent setting, such as, for instance, when both agents have the goal of minimizing the total (unweighted) completion time of their respective jobs.

In the multi-agent setting, a key issue is the determination of *nondominated* (or Pareto-optimal) schedules, i.e., such that a better schedule for one agent necessarily results in a worse schedule for at least another agent.

For the two-agent and three-agent cases, Baker and Smith (2003) analyzed the problem of minimizing a convex combination of the agents' objective functions. They characterize the complexity of the problem in some scenarios, when the objective functions are chosen among $C_{\max}$, $L_{\max}$, $\sum C_i$ and $\sum w_i C_i$. The optimal solutions of their model are nondominated schedules, but not all of them can be obtained in this way. Agnetis et al. (2004) addressed the complexity of several two-agent, single-machine problems in which one agent computes the best solution for it, given that the other agent will not accept schedules of cost greater than a certain value. For various combinations of the two agents' cost functions the problem of generating nondominated schedules was addressed.

In this paper we provide new results for two types of problems, namely the (*decision*) problem of finding, if it exists, a single feasible solution, and the (*Pareto-optimization*) problem of finding all nondominated solutions. In particular:

- For the two-agent case, we show that
  - an exponential number of nondominated solutions may exist when the cost functions are: total weighted completion time for one agent and maximum of regular functions in each job's completion times for the other;
  - the decision problem can be solved in $O(n \log n)$, when each agent's cost function is the maximum of regular functions.
- For the general multi-agent case, we give polynomial time algorithms for the scenarios in which:
  - the cost function of $k-1$ agents is the maximum of regular functions, and the other agent's cost function is total completion time;
  - the cost function of all k agents is the maximum of regular functions;
  - the cost function of a fixed number p of agents is the number of late jobs, while the other agents' cost function is the maximum of regular functions.

The paper is organized as follows. In Section 2 we formally introduce the notation and terminology used through the rest of the paper. In Section 3, we show that when one agent wants to minimize the weighted sum of completion times and another wants to minimize the makespan, even if $k = 2$, there is an exponential number of nondominated solutions (this issue was left open in Agnetis et al. (2004)). In Section 4, a decomposition procedure is described, which proves useful to characterize the complexity of several special cases, as illustrated in Section 5. Conclusions follow in Section 6.

2 Problem definition and notation

In this section we introduce the notation and terminology we use throughout the paper.

There are k competing agents $A = \{1, 2, \dots, k\}$. Each of them has a set of non-preemptive *jobs* to be processed on a single common machine. Every agent $h \in A$ has to execute the job set $J^h = \{J_1^h, J_2^h, \dots, J_{n(h)}^h\}$. We call h-jobs the jobs of J^h. The processing time of job J_i^h will be denoted by p_i^h. In some cases we will consider job due dates as well, d_i^h. We always assume zero release dates for all jobs. Each agent will have to schedule its jobs on the machine complying with the presence of the other agents' jobs.

Let σ indicate a feasible schedule of the $n = \sum_{h \in A} n(h)$ jobs, i.e., a feasible assignment of starting times to the jobs of the k agents. The completion time of job J_i^h in σ will be denoted as $C_i^h(\sigma)$ (or, simply, C_i^h if it does not generate confusion).

Each agent wants to minimize its own objective function, which depends on the completion times of its jobs only. We indicate the k objective functions by $f^h(\sigma) : \{C_i^h(\sigma) : 1 \le i \le n(h)\} \to \mathbb{R}, h \in A$. In this paper we consider the following objective functions for the generic agent h:

- *Maximum of regular functions* $f^h_{\max}(\sigma) = \max_{i=1,\dots,n(h)}\{f^h_i(C^h_i(\sigma))\}$, where each $f^h_i(\cdot)$ is a nondecreasing function of the completion time of job J^h_i. This includes the makespan $C_{\max}$ and maximum tardiness $T_{\max}$ as special cases.
- *Number of late jobs* $\sum U^h_i(\sigma) = \sum_{i=1}^{n(h)} U^h_i(\sigma)$, where $U^h_i(\sigma) = 1$ if job J^h_i is late in σ and zero otherwise.
- *Total weighted completion time* $\sum w_i {C_i}^h(\sigma) = \sum_{i=1}^{n(h)} w_i C^h_i(\sigma)$. When all the weights w_i are equal, we obtain the special case of total completion time $\sum C_i$.

Since all these objective functions are regular (i.e., nondecreasing in the completion times) there is no convenience in keeping the machine idle, and therefore each job is started as soon as the previous job in the sequence is completed. We use $\sum C_i$ for $\sum w_i C_i$ when $w_i = 1$ for all i.

We say that a schedule σ is *nondominated* if there is no schedule $\bar{\sigma}$ such that $f^h(\bar{\sigma}) \le f^h(\sigma)$, $h \in A$, and at least one of the k inequalities is strict, i.e., a schedule is nondominated if a better schedule for one of the k agents necessarily results in a worse schedule for another. Distinct nondominated schedules $\sigma, \sigma', \dots$ may yield the same k-tuple of objective function values $(f^1(\sigma), \dots, f^k(\sigma)) = (f^1(\sigma'), \dots, f^k(\sigma')) = \cdots = (y^1, \dots, y^k)$. We call $(y^1, \dots, y^k)$ a *nondominated* k-tuple of objective function values. We say that $\sigma, \sigma', \dots$ are *equivalent* schedules, and for each nondominated k-tuple we are interested in finding *one* of them, not all of them.

Following the usual classification scheme for scheduling problems, we denote a problem with three fields, $\psi_1|\psi_2|\psi_3$, where ψ_1 indicates the scheduling environment, ψ_2 describes the job characteristics or restrictive requirements, and ψ_3 defines the set of objective functions, one for each agent. In particular, in this paper we address only single machine problems, so we let $\psi_1 = 1$. Under ψ_3, we will distinguish the two types of problems as follows.

Decision Problem $(1||f^1, f^2, \dots, f^k)$. Given k integers $Q^1, \dots, Q^k$ and, for each agent $h \in A$, the job set J^h and the objective function $f^h(\cdot)$ of the agent, find a schedule σ^* such that $f^h(\sigma^*) \le Q^h$ for all $h \in A$.

Pareto-Optimization Problem $(1||f^1 \cdot f^2 \cdot \dots \cdot f^k)$. Given the job sets J^h and the k objective functions $f^h(\cdot)$ of each agent $h \in A$, find the set of all nondominated k-tuples $(f^1(\cdot), \dots, f^k(\cdot))$ and a corresponding schedule of $J^1, \dots, J^k$ for each k-tuple.

In some cases it will be necessary to specify the bounding integers in the decision problem, and so we will indicate the decision problem as $1||f^1 \le Q^1, f^2 \le Q^2 \dots, f^k \le Q^k$. If some of the values Q^h, $h = 1, \dots, k$ are too small, an instance of $1||f^1, f^2, \dots, f^k$ may not have feasible solutions. If there is at least one feasible solution, we say that the instance is *feasible*.

Given an instance of $1||f^1, f^2, \dots, f^k$, the problem of finding, among feasible schedules, one which is also nondominated can always be addressed by *binary search*. Suppose first that a feasible solution σ to $1||f^1 \le Q^1, f^2 \le Q^2, \dots, f^k \le Q^k$ is available. Now we want to find a schedule σ' such that $f^h(\sigma') \le Q^h$ for all $h \in A\backslash\{1\}$ and $f^1(\sigma')$ is minimum. We may try solving $1||f^1 \le \frac{1}{2}Q^1, f^2 \le Q^2, \dots, f^k \le Q^k$. If we get a feasible solution we may further decrease $\frac{1}{2}Q^1$ to $\frac{1}{4}Q^1$, otherwise we try $\frac{3}{4}Q^1$. This goes on until we determine the smallest value Q^{1*} such that instance $1||f^1 \le Q^{1*}, f^2 \le Q^2, \dots, f^k \le Q^k$ is feasible. Next, we apply the same procedure to find the smallest value Q^{2*} such that $1||f^1 \le Q^{1*}, f^2 \le Q^{2*}, f^3 \le Q^3 \dots, f^k \le Q^k$ is feasible and, iterating, we eventually get a solution σ^* such that $f^h(\sigma^*) \le Q^{h*}$ for all $h \in A$. It is not hard to see that σ^* is nondominated. Moreover note that in general we may obtain different nondominated schedules if we perform the binary searches on the values Q^h according to different permutations of the agents' sequence.

Table 1 Summary of complexity results for decision and Pareto-optimization problems for the two-agent ($k = 2$) case

Scenario ($k = 2$)	Computational complexity of $1\|\|f^1, f^2$	Number of nondominated pairs in $1\|\|f^1 \cdot f^2$
$(f^1_{\max}, f^2_{\max})$	$O(n \log n)$ (Section 5.1)	$O(n(1)n(2))$
$(\sum w_i C^1_i, f^2_{\max})$	Binary NP-hard	Exponential (Section 3)
$(\sum C^1_i, f^2_{\max})$	$O(n \log n)$	$O(n(1)n(2))$
$(\sum U^1_i, f^2_{\max})$	$O(n \log n)$	$O(n(1))$
$(\sum U^1_i, \sum U^2_i)$	$O(n^3)$	$\min\{n(1), n(2)\}$
$(\sum C^1_i, \sum U^2_i)$	Open	$O(n(2))$
$(\sum w_i C^1_i, \sum U^2_i)$	Binary NP-hard	$O(n(2))$
$(\sum C^1_i, \sum C^2_i)$	Binary NP-hard	Exponential

In the Pareto-optimization problem $1||f^1 \cdot f^2 \cdot \dots \cdot f^k$, the agents want to list all possible nondominated k-tuples, in order to negotiate the most acceptable trade-off for them.

The main focus of the paper is to analyze the complexity of these problems and propose solution algorithms. Table 1 summarizes known results for the $k = 2$ case. The second column reports the computational complexity of the problem $1||f^1, f^2, \dots, f^k$ and the third column the number of distinct nondominated pairs. Except where indicated, the details concerning the results of the table are contained in Agnetis et al. (2004).

3 Pareto-optimal solutions of $1||\sum w_i C^1_i \cdot f^2_{\max}$

In the following we show that $1||\sum w_i C^1_i \cdot f^2_{\max}$ may have an exponential number of nondominated solutions. This issue was left open in Agnetis et al. (2004).

Consider the following instance of $1||\sum w_i C^1_i \cdot C^2_{\max}$. Agent 1 has $n(1)$ jobs. For each job $J^1_i \in J^1$ the processing times and the weights are: $p^1_i = w^1_i = 2^{i-1}$, $i = 1, 2, \dots, n(1)$. Agent 2 has a single job of unit length.

Theorem 3.1. *Consider an instance of* $1||\sum w_i C^1_i \cdot C^2_{\max}$ *in which agent* 2 *has a single job of unit length, while agent* 1 *has* $n(1)$ *jobs. For each job* i ($i \in \{1, 2, \dots, n(1)\} = J^1$), $p^1_i = w^1_i = 2^{i-1}$. *Then, for every active schedule, the quantity* $C^2_1 + \sum_{i=1}^{n(1)} w_i C^1_i$ *is constant and equal to* $\frac{1}{3}(1 + 2^{2n(1)+1})$.

Proof: Given any active schedule σ, consider two adjacent jobs j and k. Let t be the starting time of job j and $t + p_j$ the starting time of job k. The contribution to the objective function of the two jobs is then $w_j(t + p_j) + w_k(t + p_j + p_k)$. Consider now the schedule $\bar{\sigma}$ in which the two jobs are switched: the contribution of the two jobs to the objective function is now $w_k(t + p_k) + w_j(t + p_k + p_j)$. Observe now that $w_j p_k = w_k p_j$ for any pair of jobs in $J^1 \cup J^2$ (since $w_i = p_i$ for each job), thus proving that σ and $\bar{\sigma}$ have the same value of the objective function. This implies that any active schedule produces the same value of the quantity $C^2_1 + \sum_{i=1}^{n(1)} w_i C^1_i$. This value can be computed, for example, by considering the sequence: $J^2_1, J^1_1, J^1_2, \dots, J^1_n(1)$. We have:

$$C^2_1 + \sum_{i=1}^{n(1)} w_i C^1_i = 1 + \sum_{i=1}^{n(1)} 2^{i-1} 2^i = 1 + \sum_{i=1}^{n(1)} 2^{2i-1} = 1 + \sum_{i=1}^{2n(1)} 2^i - \sum_{i=1}^{n(1)} 2^{2i}.$$

Since $\sum_{i=1}^{n(1)} 2^{2i} = 2\sum_{i=1}^{n(1)} 2^{2i-1}$, we can write: $\sum_{i=1}^{n(1)} 2^{2i-1} = \sum_{i=1}^{2n(1)} 2^{i} - 2\sum_{i=1}^{n(1)} 2^{2i-1}$. Hence, we obtain: $\sum_{i=1}^{n(1)} 2^{2i-1} = \frac{1}{3}(\sum_{i=1}^{2n(1)} 2^{i}) = \frac{1}{3}(2^{2n(1)+1} - 2)$.

In conclusion, the quantity $C_1^2 + \sum_{i=1}^{n(1)} w_i C_i^1$ is equal to $\frac{1}{3}(1 + 2^{2n(1)+1})$, and the thesis follows. □

In order to prove that the above instance has an exponential number of nondominated pairs, consider that, for any value $1 \le Q^2 \le 2^{n(1)}$, there is a subset of J^1 whose total length equals $Q^2 - 1$. This implies that there is a feasible solution to $1||\sum w_i C_i^1, C_{\max}^2 \le Q^2$ where $C_{\max}^2 = Q^2$ and $\sum w_i C_i^1 = \frac{1}{3}(1 + 2^{2n(1)+1}) - Q^2$. This is clearly a nondominated solution and therefore we have $2^{n(1)}$ nondominated pairs.

4 Separability property for $f_{\max}$

In this section, we describe a procedure for reducing a k-agent problem to a $(k-1)$-agent problem, when the cost function of agent k is the maximum of regular functions. Under certain conditions on the other objective functions, this property may be exploited to devise efficient solution algorithms for the original problem.

The decomposition procedure operates in two phases, and can be outlined as follows. Consider a decision problem with k agents in which the cost function of the k-th agent is the maximum of regular functions:

$$1||g^1, g^2, \ldots, g^{k-1}, f_{\max}^k \tag{1}$$

The first phase consists in defining suitable *reserved time intervals* for the jobs of the k-th agent. Such intervals are *forbidden* to the remaining $k-1$ agents, who have therefore to deal with a problem with forbidden intervals. As it will be clarified by Lemmas 4.1 and 4.2, due to the regularity of the all cost functions, given a preemptive solution to the problem with forbidden intervals, we may easily obtain a non-preemptive solution yielding the same cost values for all agents. The second phase consists in actually solving the problem with forbidden intervals. For certain cost functions g^h of the remaining $k-1$ agents, this can be efficiently done by solving an equivalent instance of

$$1||g^1, g^2, \ldots, g^{k-1}. \tag{2}$$

The phases of the procedure are illustrated in the following sections.

4.1 Reserved intervals for agent k

In this section we always refer to the jobs of agent k, so we omit k in C_i^k, J_i^k, Q^k ....

Since $f_{\max}^k = \max_{i=1,\ldots,n(k)}\{f_i^k(C_i^k(\sigma))\}$, for each job of agent k we can define a *deadline* D_i such that $f_i(C_i) \le Q$ for $C_i \le D_i$ and $f_i(C_i) > Q$ for $C_i > D_i$. In other words, job J_i must complete no later than D_i in a feasible solution.

In what follows, we call the *latest start time* (LS_i) of job J_i the maximum value the starting time of J_i can attain in a feasible schedule such that $C_i \le D_i$ for all $J_i \in J^k$. The values LS_i can be computed as follows. Number the k-jobs in nondecreasing order of D_i. Schedule the last job $J_{n(k)}$ to start at time $D_{n(k)} - p_{n(k)}$. Continue backwards, letting $LS_i :=$

$\min\{D_i, LS_{i+1}\} - p_i$, for all $i = n(k) - 1, \ldots, 1$. Clearly, if job J_i starts after time LS_i, then at least one k-job (say, J_ℓ) attains $f_\ell(C_\ell) > Q$.

Consider now, for each k-job J_i, the latest processing interval $[LS_i, D_i]$. Let $I = \cup_{i=1}^{n(k)}[LS_i, D_i]$. Set I consists of a number $\beta \leq n(k)$ of intervals, $I_{1,h_1}, I_{h_1,h_2}, \ldots, I_{h_{\beta-1},n(k)}$, call them *reserved intervals* for agent k. Each reserved interval $I_{u,v}$ ranges from LS_u to D_v. Note that, by construction, $||I_{u,v}|| = D_v - LS_u = \sum_{i=u}^{v} p_i$. We say that jobs $J_u, J_{u+1}, \ldots, J_v$ are *associated with* $I_{u,v}$.

4.2 Preemptive problem with forbidden intervals

Consider now the variant of Problem (1) in which preemption is allowed:

$$1|pmtn|g^1, \ldots, g^{k-1}, f_{\max}^k. \tag{3}$$

The following lemmas extend known results on preemption redundancy for classical (single-agent) scheduling problems, see, e.g., Conway, Maxwell, and Miller (1967).

Lemma 4.1. *If a feasible solution to* $1|pmtn|g^1, \ldots, g^{k-1}, f_{\max}^k$ *exists, then there is a feasible solution to* $1||g^1, \ldots, g^{k-1}, f_{\max}^k$.

Proof: Just observe that if in the feasible solution to $1|pmtn|g^1, \ldots, g^{k-1}, f_{\max}^k$ there is a job J_i (of any agent), ending at C_i, which is preempted at least once, we can always schedule the whole J_i in interval $[C_i - p_i, C_i]$, moving other (parts of) jobs backwards, without increasing the completion time of any job. Repeating this for each preempted job, we eventually obtain a nonpreemptive solution. □

Lemma 4.2. *If there exists a feasible solution to* $1|pmtn|g^1, \ldots, g^{k-1}, f_{\max}^k$*, there is one in which each k-job is nonpreemptively scheduled in the reserved interval with which it is associated.*

Proof: In a feasible solution to $1|pmtn|g^1, \ldots, g^{k-1}, f_{\max}^k$, all the k-jobs associated with interval $I_{u,v}$ complete before D_v^k. Hence, if we move all the portions of each such job to exactly fit the interval $I_{u,v}$, we obtain a solution in which the completion time of no h-job $(h \in A\backslash\{k\})$ has increased, since we only moved pieces of h-jobs backwards. □

Lemma 4.2 allows us to fix the position of the k-jobs in a feasible solution to Problem (3).

We have therefore showed that, in order to solve the original Problem (1) with k agents we may equivalently solve an instance of a preemptive problem involving only agents $1, \ldots, k-1$, in which intervals $I = \{I_{1,h_1}, I_{h_1,h_2}, \ldots, I_{h_{\beta-1},n(k)}\}$ cannot be used. We indicate such problem as:

$$1|pmtn, I|g^1, \ldots, g^{k-1}. \tag{4}$$

and call it *problem with forbidden intervals.*

4.3 Solving problem $1|pmtn, I|g^1, \dots, g^{k-1}$

In some cases the preemptive problem with forbidden intervals, and consequently the original instance, can be solved efficiently by a simple sequencing rule. In other cases, it is necessary to define an auxiliary problem, as it is shown in the following paragraphs.

4.3.1 SPT sequencing for $1|pmtn, I|\sum C_i^1$

Consider the single-agent problem $1|pmtn, I|\sum C_i^1$. The following proposition holds.

Proposition 4.3. *Problem* $1|pmtn, I|\sum C_i^1$*, with set of forbidden intervals I, is solved by sequencing the* 1*-jobs in shortest-processing-time order.*

Proof: The result follows straightforwardly by a pairwise interchange argument. □

Referring to the case $k = 2$, Lemma 4.1, Lemma 4.2 and Proposition 4.3 ensure that a feasible solution to $1||\sum C_i^1, f_{\max}^2$ can be derived from a feasible solution to $1|pmtn, I|\sum C_i^1$, where I is the set of intervals reserved to agent 2. Observe that the same result does not hold if f^1 is $\sum w_i C_i^1$. In fact, sequencing the 1-jobs according to non-increasing w_i/p_i order (Smith's rule), does not ensure optimality when forbidden intervals exist. The problem $1||\sum w_i C_i^1, f_{\max}^2$ is in fact NP-hard (Table 1).

4.3.2 Auxiliary problem with modified deadlines

In other cases, for different combinations of the objective functions $g^1, \dots, g^{k-1}$, the problem with forbidden intervals (4) may be solved by resorting to an auxiliary instance of a problem *without* forbidden intervals. In particular if, for all $h \in A \setminus \{k\}$, the cost functions are of the type:

1. $g^h = f_{\max}^h$, or
2. $g^h = \sum U_i^h$,

we can define an auxiliary problem by Procedure 4.4 below, where we use the symbol d_i^h indifferently to denote a due date, if $f^h = \sum U_i$, or a deadline, computed as in Section 4.1, if $f^h = f_{\max}$.

Procedure 4.4

```
Input: Problem with forbidden intervals I: $1|pmtn, I|g^1, g^2, \dots, g^{k-1}$ and values $d_i^h$ for
    $h \in A \setminus \{k\}$, $i \in J^h$.
Output: Auxiliary instance of the problem $1|pmtn|g^1, g^2, \dots, g^{k-1}$ with modified due dates
    (or deadlines) $\mathcal{D}_i^h$, for $h \in A \setminus \{k\}$, $i \in J^h$.
Begin
 For each job $J_i^h$ of each agent $h \in A \setminus \{k\}$:
   1. if $d_i^h$ falls outside of any forbidden interval, subtract from $d_i^h$ the total length of all the
      forbidden intervals preceding $d_i^h$, i.e., define the modified due date
```

$$\mathcal{D}_i^h := d_h^A - \sum_{u,v:D_v^k \le d_i^h} ||I_{u,v}||$$

```
  2. else if $d_i^h$ falls within the forbidden interval $I_{p,q}$, do the same, but instead of $d_i^h$ use
     the left extreme of $I_{p,q}$, i.e.,
```

$$\mathcal{D}_i^h := LS_p - \sum_{u,v:D_v^k<d_i^h} ||I_{u,v}||$$

```
 Return $\mathcal{D}_i^h$ for all $h \in A\backslash\{k\}$, $i \in J^h$.
End
```

The following proposition summarizes the results obtained in the previous paragraphs.

Proposition 4.5. *Problem (1)*

$$1||g^1, g^2, \ldots, g^{k-1}, f_{\max}^k$$

with $g^h = f_{\max}^h$, *or* $g^h = \sum U_i^h$ *for all* $h \in A\backslash\{k\}$, *is feasible if and only if the instance of Problem (2)*

$$1||g^1, g^2, \ldots, g^{k-1}$$

obtained modifying the due dates and deadlines according to Procedure 4.4 is feasible.

Proof: Lemma 4.1 ensures that Problem (1)

$$1||g^1, g^2, \ldots, g^{k-1}, f_{\max}^k$$

and Problem (3)

$$1|pmtn|g^1, g^2, \ldots, g^{k-1}, f_{\max}^k$$

are equivalent from the feasibility viewpoint. On the other hand, Lemma 4.2 shows that we may always non-preemptively schedule the k-jobs in the associated reserved intervals, and therefore Problem (3) can be solved by solving an instance of Problem (4)

$$1|pmtn, I|g^1, g^2, \ldots, g^{k-1}.$$

When the cost functions of the first $k-1$ agents consist of minimizing the maximum of regular functions or the number of late jobs ($g^h = f_{\max}^h$, or $g^h = \sum U_i^h$ for all $h \in A\backslash\{k\}$), we may define an auxiliary instance of the Problem

$$1|pmtn|g^1, g^2, \ldots, g^{k-1} \tag{5}$$

in which the due dates and deadlines of the jobs have been modified according to Procedure 4.4. Such modified due dates and deadlines account for the reserved intervals associated with the k-jobs. Since preemption is allowed, it is not hard to see that if and only if the auxiliary instance of (2) is feasible then so is the instance of Problem (4) with forbidden intervals.

Invoking again Lemma 4.1, Problem (5) is in turn equivalent to the nonpreemptive Problem (2)

$$1||g^1, g^2, \dots, g^{k-1}.$$

This proves that, when $g^h = f^h_{\max}$, or $g^h = \sum U^h_i$ for all $h \in A\setminus\{k\}$, Problems (2) and (1) are indeed equivalent. □

The previous proposition emphasizes that it is always possible to reduce an instance of Problem (1), when $g^h = f^h_{\max}$, or $g^h = \sum U^h_i$ for all $h \in A\setminus\{k\}$, to an equivalent instance of Problem (2) in polynomial time.

5 Polynomial special cases

In this section, we apply the concepts introduced in Section 4 to provide polynomial algorithms for some relevant cases.

5.1 $1||\sum C^1_i, f^2_{\max}, \dots, f^k_{\max}$ and $1||f^1_{\max}, f^2_{\max}, \dots, f^k_{\max}$

Let us first address the polynomial solvability of problem $1||\sum C^1_i, f^2_{\max}, \dots, f^k_{\max}$.

This result can be stated by observing that all the jobs of agents having $f_{\max}$ as cost function may be indeed regarded as the jobs of a *single* agent, and the approach described in Section 4.3.1 can be adopted.

Theorem 5.1. $1||\sum C^1_i, f^2_{\max}, \dots, f^k_{\max}$ *can be solved in time* $O(n(1)\log n(1) + \tilde{n}\log\tilde{n})$*, where* $\tilde{n} = \sum_{h\in A\setminus\{1\}} n(h)$.

Proof: If $k = 2$, the problem is $1||\sum C^1_i, f^2_{\max}$. From Proposition 4.3, we may easily solve the problem with forbidden intervals by sequencing the 1-jobs in SPT order. Consider now the general case of k agents. We may proceed exactly in the same way as for $k = 2$ and consider all the jobs of agents $2, 3, \dots, k$ *as if they were jobs of a single agent*. Thus we order all the jobs in $\bigcup_{h\in A\setminus\{1\}} J^h$ in nondecreasing order of the deadlines D_i and, starting from one with the largest deadline, compute backward the $LS_i := \min\{D_i, LS_{i+1}\} - p_i$ in order to define proper reserved intervals for those jobs. After this, the k-agent problem is indeed equivalent to $1||\sum C^1_i, f^2_{\max}$.

Let us turn to complexity issues. In the $k = 2$ case, the complexity can be measured for the basic steps of the procedure as follows. The definition of reserved intervals for the 2-jobs implies that the 2-jobs are ordered first. The complexity for this is $O(n(2)\log n(2))$. Moreover, the computation of the reserved intervals takes time $O(n(2))$. The definition and the solution of the auxiliary preemptive instance requires $O(n(1)\log n(1))$ time for the SPT ordering of the 1-jobs and $O(n(1) + n(2))$ time to reconstruct the preemptive schedule. The solution of the original instance of $1||\sum C^1_i, f^2_{\max}$ is obtained from the preemptive schedule in time $O(n(1) + n(2))$. The overall complexity is thus dominated by the ordering steps and it is therefore $O(n(1)\log n(1) + n(2)\log n(2))$.

For the general case with k agents, the complexity result immediately follows observing that the cardinality of the job set $\bigcup_{h\in A\setminus\{1\}} J^h$ is equal to $\sum_{h\in A\setminus\{1\}} n(h)$. □

In Agnetis et al. (2004), a different procedure was proposed for solving $1||\sum C^1_i, f^2_{\max}$, having the same complexity, based on an adaptation of a well-known algorithm for the single-agent problem $1|prec|f_{\max}$ due to Lawler (1973).

Let us now turn to problem $1||f^1_{\max}, \dots, f^k_{\max}$. By the same arguments of Theorem 5.1, it is easy to prove that the following result holds.

Theorem 5.2. $1||f^1_{\max}, \dots, f^k_{\max}$ *can be solved in time* $O(n \log n)$ *where* $n = \sum_{h \in A} n(h)$.

This result allows to solve the decision problem $1||f^1_{\max}, f^2_{\max}$ with a better complexity than using the algorithm presented in Agnetis et al. (2004), i.e., $O(n(1)^2 + n(2) \log n(2))$.

5.2 $1||\sum U^1_i, \dots, \sum U^p_i, f^{p+1}_{\max}, \dots, f^k_{\max}$

In view of the decomposition procedure described in Section 4, this problem reduces to solving an auxiliary instance of $1||\sum U^1_i, \dots, \sum U^p_i$. Hence, for $p = 1$ the problem is solvable by an algorithm due to Moore (1968), while for $p = 2$ the problem can be solved by dynamic programming with an algorithm described in Agnetis et al. (2004). We next show that it is indeed possible to solve the problem in polynomial time for any *fixed* number $p \geq 1$ of agents. The following lemma relates to the structure of a feasible schedule.

Lemma 5.3. *For any feasible instance of* $1||\sum U^1_i, \dots, \sum U^p_i$, *there is a feasible schedule* σ^* *in which all the late jobs are scheduled consecutively at the end of the schedule, and all the early jobs are scheduled consecutively in Earliest Due Date (EDD) order at the beginning of the schedule.*

Proof: Consider an optimal schedule σ^* and move all the late jobs to the end of the schedule, thus obtaining a new schedule σ'. Clearly, $\sum U^h_i(\sigma') \leq \sum U^h_i(\sigma^*)$, for $h = 1, \dots, p$, since we are moving backward the early jobs. Consider now all the early jobs in σ', that are sequenced consecutively at the beginning of the schedule, and resequence them in EDD order. This does not increase the number of late jobs, thus completing the proof. □

In the remaining part of this section we assume that the $\hat{n} = \sum_{h=1}^p n(h)$ jobs in $\bigcup_{h=1}^p J^h$ are numbered from J_1 to $J_{\hat{n}}$ according to the overall EDD order.

We next illustrate a recursion relation that can be exploited to design a polynomial time (for fixed p) dynamic programming algorithm for $1||\sum U^1_i, \dots, \sum U^p_i$.

Let $C(i, l_1, \dots, l_p)$ be the minimum completion time of the last early job in a schedule of the first i jobs (i.e., job set $\{J_1, \dots, J_i\}$) in which there are at most l_h late h-jobs, $h = 1, \dots, p$, and denote by $\sigma(i, l_1, \dots, l_p)$ the corresponding schedule. By definition, we set $C(i, l_1, \dots, l_p) = \infty$ if no such schedule exists.

The basic idea of the algorithm is that if job $J_i \in J^h$ is early in $\sigma(i, l_1, \dots, l_p)$, then it is the last early job in this schedule, and its completion time is given by $C(i-1, l_1, \dots, l_p) + p_i$. If $J_i \in J^h$ is late, then the makespan of the early jobs is the same as in $\sigma(i-1, l_1, \dots, l_h - 1, \dots, l_p)$. In conclusion, the following relations hold:

Boundary Conditions

$$C(0, l_1, \dots, l_p) = 0 \qquad \text{if } l_h \geq 0\,,\ h = 1, \dots, p$$

$$C(i, l_1, \dots, l_p) = \infty \quad \text{if } i < 0 \text{ or } l_h < 0 \text{ for some } h = 1, \dots, p$$

Recursion Relation

$$f(i, l_1, \dots, l_p) = \begin{cases} \infty & \text{if } C(i-1, l_1, \dots, l_p) + p_i > d_i \\ 0 & \text{otherwise} \end{cases}$$

$$C(i, l_1, \dots, l_p) = \min \left\{ \begin{array}{l} C(i-1, l_1, \dots, l_p) + p_i + f(i, l_1, \dots, l_p); \\ C(i-1, l_1, \dots, l_{h-1}, l_h - 1, l_{h+1}, \dots, l_p), \quad \text{where} \quad J_i \in J^h \end{array} \right\}$$

Lemma 5.4. *If $C(i, l_1, \dots, l_p)$ is finite, then it is the minimum completion time of the last early job over all feasible schedules for the job set $\{J_1, \dots, J_i\}$, with at most l_h late h-jobs, $h = 1, \dots, p$. If $C(i, l_1, \dots, l_p)$ is infinite, then there is no such feasible schedule.*

Proof: The proof is by induction on i. Clearly the property holds for $i = 1$, for any $l_h = 1, \dots, n(h)$, with $h = 1, \dots, p$. Now, assume that the property holds until $(i-1)$. We will show that the property holds also for i and for any $l_h = 1, \dots, n(h)$, and $h = 1, \dots, p$. Let σ be a feasible schedule for the job set $\{J_1, \dots, J_i\}$, such that the completion time τ of the last early job in σ is minimum among all feasible schedules with at most l_h late h-jobs, $h = 1, \dots, p$. Without loss of generality, assume that $J_i \in J^{\hat{h}}$. If J_i is late in σ, then, from the inductive hypothesis, $\tau = C(i-1, l_1, \dots, l_{\hat{h}-1}, l_{\hat{h}} - 1, l_{\hat{h}+1}, \dots, l_p)$. If J_i is early in σ then, again from the inductive hypothesis, $\tau = C(i-1, l_1, \dots, l_p) + p_i$. Hence, the above recursion relation correctly chooses the smallest between the two quantities. Note that the schedule attaining $C(i, l_1, \dots, l_p)$ is feasible if either $C(i-1, l_1, \dots, l_p) + p_i < d_i$ or $C(i-1, l_1, \dots, l_{\hat{h}-1}, l_{\hat{h}} - 1, l_{\hat{h}+1}, \dots, l_p) < \infty$. If none of the two holds, there can be no feasible schedule of $\{J_1, \dots, J_i\}$ with at most l_h late h-jobs, $h = 1, \dots, p$, and the algorithm sets $C(i, l_1, \dots, l_p) = \infty$. □

Theorem 5.5. *An instance of $1||\sum U_i^1 \le Q_1, \dots, \sum U_i^p \le Q_p$ is feasible if and only if $C(\hat{n}, Q_1, \dots, Q_p) < \infty$. This value can be computed in time $O(\sum_{h=1}^p n(h) \prod_{j=1}^p n(j))$.*

Proof: The first part of the theorem is a straightforward consequence of Lemma 5.4. Let us now turn to complexity. Computing each $C(i, l_1, \dots, l_p)$ requires constant time, and therefore computing all of them requires $O(\sum_{h=1}^p n(h) \prod_{j=1}^p n(j))$ time. □

Note that the algorithm qualifies as polynomial if the number p of agents whose cost function is the number of late jobs is fixed.

6 Conclusions

In this paper we have investigated the complexity of some scheduling problems in which several agents have to negotiate the usage of a common processing resource. The focus of this paper was the problem of finding a schedule attaining minimum quality requirements specified by each single agent. The model presented in this paper can be regarded as a tool for complex negotiation processes. A lot has to be done in the area of modeling multi-agent scheduling situations.

- From the modeling viewpoint, connections with concepts of bargaining theory (e.g. Nash solution or other solution concepts in the case of $k = 2$, see for instance Osborne and

Rubinstein (1994) when specialized to the scheduling context might deserve further investigation.

- From the theoretical viewpoint, the problem $1||\sum U_i^1, \sum C_i^2$ is minimally open. Depending on its status, other multi-agent problems might deserve further investigation.
- From the algorithmic viewpoint, there is a need for effective enumeration algorithms as well as approximation algorithms and polynomial approximation schemes for NP-hard cases.

Acknowledgments The authors wish to thank one anonymous referee for helpful comments and suggestions.

References

Agnetis, A., P.B. Mirchandani, D. Pacciarelli, and A. Pacifici. (2004). "Scheduling Problems with Two Competing Agents." *Operations Research*, 52(2), 229–242.

Arbib, C., S. Smriglio, and M. Servilio. (2004). "A Competitive Scheduling Problem and its Relevance to UMTS Channel Assignment." *Networks*, 44(2), 132–141.

Baker, K.R. J.C. Smith. (2003). "A Multiple-Criterion Model for Machine Scheduling." *Journal of Scheduling*, 6(1), 7–16.

Brewer, P.J. and C.R. Plott (1996). "A Binary Conflict Ascending Price (BICAP) Mechanism for the Decentralized Allocation of the Right to Use Railroad Tracks." *International Journal of Industrial Organization*, 14, 857–886.

Chen, Y., Y. Peng, T. Finin, Y. Labrou, S. Cost, B. Chu, J. Yao', R. Sun, and B. Wilhelm. (1999). "A Negotiation-Based Multi-Agent System for Supply Chain Management." In *Proc. of the Agents'99 Workshop "Agent-Based Decision-Support for Managing the Internet-Enabled Supply-Chain,"* Seattle, WA, pp. 15–20.

Conway, R.W., W.L. Maxwell, and L.W. Miller. (1967). *Theory of Scheduling*. Reading, MA: Addison Wesley.

Huang, X. and J.C. Hallam. (1995). "Spring-Based Negotiation for Conflict Resolution in AGV Scheduling." In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*.

Kim, K., B.C. Poulon, C.J. Petrie, and V.R. Lesser. (2000). "Compensatory Negotiation for Agent-Based Project Schedule Coordination." CIFE Working Paper #55. Stanford University.

Lawler, E.L. (1973). "Optimal Sequencing of a Single Machine Subject to Precedence Constraints." *Management Science*, 19, 544–546.

Moore, J.M. (1968). "An *n* Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs." *Management Science*, 15, 102–109.

Osborne, M.J. and A. Rubinstein. (1994). *A Course in Game Theory*. Cambridge: MIT Press.

T'kindt, V. and J.C. Billaut. (2002). *Multicriteria Scheduling*. Berlin: Springer-Verlag.