OPERATIONS RESEARCH
Vol. 52, No. 2, March–April 2004, pp. 229–242
ISSN 0030-364X | EISSN 1526-5463 | 04 | 5202 | 0229

DOI 10.1287/opre.1030.0092

Scheduling Problems with Two Competing Agents

Allesandro Agnetis
Dipartimento di Ingegneria dell'Informazione, Università di Siena, via Roma 56, 53100 Siena, Italy, agnetis@dii.unisi.it

Pitu B. Mirchandani
Department of Systems and Industrial Engineering, The University of Arizona, Tucson, Arizona 85721, pitu@sie.arizona.edu

Dario Pacciarelli
Dipartimento di Informatica e Automazione, Università Roma Tre, Rome, Italy, pacciare@dia.uniroma3.it

Andrea Pacifici
Dipartimento di Informatica, Sistemi e Produzione and Centro Vito Volterra, Università di Roma "Tor Vergata," Rome, Italy, pacifici@disp.uniroma2.it

We consider the scheduling problems arising when two agents, each with a set of nonpreemptive jobs, compete to perform their respective jobs on a common processing resource. Each agent wants to minimize a certain objective function, which depends on the completion times of its jobs only. The objective functions we consider in this paper are maximum of regular functions (associated with each job), number of late jobs, and total weighted completion times. We obtain different scenarios, depending on the objective function of each agent, and on the structure of the processing system (single machine or shop). For each scenario, we address the complexity of various problems, namely, finding the optimal solution for one agent with a constraint on the other agent's cost function, finding single nondominated schedules (i.e., such that a better schedule for one of the two agents necessarily results in a worse schedule for the other agent), and generating all nondominated schedules.

Subject classifications: production/scheduling: multiagent deterministic sequencing; games/group decisions: cooperative sequencing.
Area of review: Optimization.
History: Received July 2001; revisions received May 2002, January 2003, March 2003; accepted March 2003.

1. Introduction

In recent years, management problems in which multiple agents compete on the usage of a common processing resource are receiving increasing attention in different application environments and different methodological fields, such as artificial intelligence, decision theory, operations research, etc. One major stream of research in this context is related to multiagent systems (MASs), i.e., systems in which different entities (agents) interact to perform their respective tasks, negotiating among each other for the usage of common resources over time. In this paper, we focus on the following situation. There are two agents, each with a set of jobs. The agents have to schedule their jobs on a common processing resource, and each agent wishes to minimize an objective 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 between the two agents.

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 the other agent. In our analysis, we will adopt two different viewpoints. The first consists of determining the best solution for one agent, given that the other agent will not accept schedules of cost greater than a certain value for it. The second viewpoint consists of determining the set of all nondominated solutions. These two viewpoints lead to different (though related) optimization problems. The main focus of this paper is the analysis of the complexity of several problems, with different combinations of the two agents' objective functions (maximum of regular functions, total weighted completion time, number of late jobs) and different system structures (single machine or shop).

This paper is organized as follows. In §2, a review of related works is given. In §3, we formally introduce the notation and terminology used throughout the rest of this paper. From §§4 to 10, we characterize the complexity of several decision problems. In particular, from §§4 to 9, we consider single-machine problems, and in §10 two-machine shop problems. Section 11 is devoted to the problem of enumerating all nondominated solutions. Conclusions follow in §12.

2. Multiagent Scheduling Models

Scheduling situations in which agents (users, players) compete for common processing resources are close to both fields of combinatorial optimization and cooperative game theory. In fact, a few papers investigate the problem from an economic/market perspective. Curiel et al. (1989) define

a class of single-machine *sequencing games* as follows. There is a set of agents, each with exactly one job and having a cost function related to the job's completion time. Starting from an initial job sequence σ_0, jobs can be rearranged into a sequence σ^*, which minimizes the *overall* cost. Switching from σ_0 to σ^* produces an overall gain which may be redistributed among the agents. Curiel et al. (1989), as well as Hamers et al. (1996), prove that, using certain redistribution rules, the agents have no advantage in aggregating into smaller coalitions, i.e., agents are encouraged to cooperate. (In technical terms, the core of the game is nonempty.) In other papers, the analysis is extended to other structural properties of the games, as well as to different scenarios, including release times, or agents holding more than one job (e.g., Hamers et al. 1995, Fragnelli 2001).

A different view of the scheduling problem is provided by the so-called market-oriented programming (Wellman 1993). In this approach, an overall schedule is determined by means of *market prices* for the time slots in which processing resources are to be used. In particular, a *price equilibrium* is a situation in which each agent maximizes its utility, given those prices. The prices are established through a bidding mechanism (auction). For single-machine problems in which each agent holds exactly one job, and each job requires one time slot, Wellman et al. (2001) analyze several auction protocols, and for each of them they consider how far from the equilibrium the prices are obtained.

Crès and Moulin (2001) address an agent-based scheduling problem, in which agents arrive at a server, and each agent decides whether or not to stay in line depending only on how many other agents are ahead of him or her. They analyze simple, probabilistic ordering rules of the agents (e.g., random ordering), evaluating the expected number of agents served and theoretical properties such as fairness or incentive compatibility.

On the combinatorial optimization side, problems with two agents (namely, A and B, each with the respective set of jobs J^A and J^B) can be viewed as a special case of general bicriteria optimization models. For instance, the problem in which the agent A wants to minimize the total completion time of the jobs in J^A and the agent B wants to minimize the maximum lateness of the jobs in J^B can be viewed as a bicriteria, single-agent problem with weighted objective functions $\sum_{j\in J^A\cup J^B} q_j^{(1)} C_j$ and $\max_{j\in J^A\cup J^B} q_j^{(2)} T_j$, where $q_j^{(1)} = 1$ and $q_j^{(2)} = 0$ for $j \in J^A$, whereas $q_j^{(1)} = 0$ and $q_j^{(2)} = 1$ for $j \in J^B$. Hence, in principle, general methods for bicriteria optimization can be applied, although these may not exploit the peculiarity of the problem. However, two-agent scheduling problems differ from the problems commonly referred to as *bicriteria scheduling problems* (Hoogeveen 1992, Nagar et al. 1995), because in classical single-agent, bicriteria scheduling problems, all jobs contribute to both criteria, whereas in a two-agent situation, only the jobs belonging to an agent contribute to that agent's criterion. 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 agents A and B have objectives f and g, respectively. (By complexity of a bicriteria problem, we mean the complexity of minimizing one objective function with a constraint on the other.) For instance, the NP-hardness of the single-agent, bicriteria problem of minimizing the number of late jobs with a constraint on the maximum of regular functions (Lawler 1983), does not imply the NP-hardness of the corresponding two-agent problem (which is in fact polynomial; see §6). 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.

Table 1 compares the complexity status of single-machine, two-agent scheduling problems and the analogous bicriteria scheduling problems (where applicable). Besides the references in Table 1, there are several other more specific contributions to bicriteria scheduling, concerning problems with unit execution time jobs (Chen and Bulfin 1990), problems with target start times (Hoogeveen and van de Velde 2001), and nonregular objective functions (Hoogeveen 1992). An extensive review on other bicriteria scheduling problems can be found, for example, in the survey paper by Nagar et al. (1995).

Multiagent 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 multiagent 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 multiagent scheduling problem in terms of a constraint satisfaction problem where a subset of constraints can be relaxed but is 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 railtrack. Schultz et al. (2002) 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 file transfer protocol, 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 be lost,

Table 1. Summary of complexity results for the constrained optimization problems.

Problem Name	Computational Complexity	Reference Theorem	Complexity of Related Bicriteria One-Agent Problem
$1 \| \| f_{\max}^A : f_{\max}^B$	$O(n^2)$	4.1	—
$1 \| \| \sum w_i C_i^A : f_{\max}^B$	Binary NP-hard	5.2	Unary NP-hard Lenstra et al. (1977)
$1 \| \| \sum C_i^A : f_{\max}^B$	$O(n \log n)$	5.5	$O(n \log n)$ Smith (1956), Hoogeveen and Van de Velde (1995)
$1 \| \| \sum U_i^A : f_{\max}^B$	$O(n \log n)$	6.3	Binary NP-hard Lawler (1983)
$1 \| \| \sum U_i^A : \sum U_i^B$	$O(n^3)$	7.3	—
$1 \| \| \sum C_i^A : \sum U_i^B$	Open	—	Binary NP-hard Chen and Bulfin (1993)
$1 \| \| \sum w_i C_i^A : \sum U_i^B$	Binary NP-hard	5.2	Binary NP-hard Chen and Bulfin (1993)
$1 \| \| \sum C_i^A : \sum C_i^B$	Binary NP-hard	9.2	—
$F_2 \| \| f_{\max}^A : f_{\max}^B$	Binary NP-hard	10.1	—
$O_2 \| \| f_{\max}^A : f_{\max}^B$	Binary NP-hard	10.2	—

but some delay can be tolerated. In the work by Ling (1998), two agents, each with a set of jobs, compete for a single machine with the objective of minimizing total completion time. Negotiation alternatives are generated via implicit enumeration. No complexity analysis is carried out. Agnetis et al. (2000) consider a two-jobs job shop scenario in which jobs J_A and J_B belong to the agents A and B, respectively. Each agent's objective function only depends on the job's completion time (C_A and C_B, respectively). After characterizing the set of feasible solutions in the (C_A, C_B)-plane, it is shown that if the two objective functions are quasiconvex, the set of nondominated pairs (C_A, C_B) consists of a polynomial number of isolated points and line segments.

3. Problem Definition and Notation

In this section, we introduce the notation and terminology we use throughout this paper. For the sake of simplicity, the notation refers to the single-machine case. This notation will be generalized to more complex shop systems in §10.1.

There are two competing agents, called agent A and agent B. Each of them has a set of nonpreemptive *jobs* to be processed on a common machine. The agent A has to execute the job set $J^A = \{J_1^A, J_2^A, \ldots, J_{n_A}^A\}$, whereas the agent B has to execute the job set $J^B = \{J_1^B, J_2^B, \ldots, J_{n_B}^B\}$. We call A-jobs and B-jobs the jobs of the two sets. The processing time of job J_h^A (J_k^B) will be denoted by p_h^A (p_k^B). Also, let $P_A = \sum_{h=1}^{n_A} p_h^A$ and $P_B = \sum_{k=1}^{n_B} p_k^B$. In some cases we will consider job due dates as well: d_h^A (d_k^B). In this paper, we always assume zero release dates for all jobs. Each of the two agents will have to schedule his or her jobs on the machine complying with the presence of the other agent's jobs.

Let σ indicate a feasible schedule of the $n = n_A + n_B$ jobs, i.e., a feasible assignment of starting times to the jobs of both agents. The completion times of job J_h^A and J_k^B in σ will be denoted as $C_h^A(\sigma)$ and $C_k^B(\sigma)$, respectively. We will use the notation J_h, p_h, d_h, S_h, $C_h(\sigma)$ when referring to a job in the set $J^A \cup J^B$, and J_h^X, p_h^X, d_h^X, $S_h^X(\sigma)$, $C_h^X(\sigma)$ when referring to a job of a specific agent X, which can be either A or B.

Each agent has a certain objective function, which depends on the completion times of its jobs only. We indicate by $f^A(\sigma)$ and $f^B(\sigma)$ the two functions. In this paper, we consider the minimization of the following objective functions:

- $f_{\max}(\sigma) = \max_{i=1,\ldots,n}\{f_i(C_i(\sigma))\}$, where each $f_i(\cdot)$ is a nondecreasing function of the completion time of job J_i (*maximum of regular functions*).
- $\sum U_i(\sigma) = \sum_{i=1}^n U_i(\sigma)$, where $U_i(\sigma) = 1$ if job J_i is late in σ and zero otherwise (*number of late jobs*).
- $\sum w_i C_i(\sigma) = \sum_{i=1}^n w_i C_i(\sigma)$ (*total weighted completion time*).

Note that all these objective functions are regular (i.e., nondecreasing in the completion times). Hence, 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. Also, note that the first objective function includes the maximum completion time $C_{\max}(\sigma)$ and the maximum lateness $L_{\max}(\sigma)$ as special cases. 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 $f^A(\bar{\sigma}) \leqslant f^A(\sigma)$, $f^B(\bar{\sigma}) \leqslant f^B(\sigma)$ and at least one of the two inequalities is strict, i.e., a schedule is nondominated if a better schedule for one of the two agents necessarily results in a worse schedule for the other agent. Distinct nondominated schedules $\sigma, \sigma', \ldots$ may yield the same pair of objective function values $(f^A(\sigma), f^B(\sigma)) = (f^A(\sigma'), f^B(\sigma')) = \cdots = (y^A, y^B)$. We call (y^A, y^B) a *nondominated pair* of objective function values. We say that $\sigma, \sigma', \ldots$ are *equivalent* schedules, and

for each nondominated pair we are interested in finding *one* of them, not all of them.

The problems we address in this paper can be described as follows.

Constrained Optimization Problem (*CP*). Given the job sets J^A and J^B of the two agents, the two objective functions $f^A(\cdot)$ and $f^B(\cdot)$, and an integer Q, find a schedule σ^* such that $f^B(\sigma^*) \leqslant Q$, and $f^A(\sigma^*)$ is minimum.

Following the classification scheme for scheduling problems by Graham et al. (1979), we indicate this problem as $1\|f^A : f^B \leqslant Q$, or simply $1\|f^A : f^B$ whenever this does not generate confusion.

Pareto-Optimization Problem (*PP*). Given the job sets J^A and J^B of the two agents and the two objective functions $f^A(\cdot)$ and $f^B(\cdot)$, find the set of all nondominated pairs $(f^A(\cdot), f^B(\cdot))$ and a corresponding schedule of J^A and J^B for each pair. We indicate this problem as $1\|f^A \circ f^B$.

Note that the viewpoints are different in the two problems. In the former problem (*CP*), the agent A wants to find the best solution for it, given that the agent B will accept a schedule of cost up to Q. An instance of *CP* may not have feasible solutions (e.g., if Q is too small). If there is at least one feasible solution, we say that the instance is *feasible*. Note that the problem of finding, among optimal schedules, one which is also nondominated can be always addressed by *binary search.* In fact, let σ^* be the optimal solution to $1\|f^A : f^B \leqslant \widetilde{Q}$. We can solve $1\|f^A : f^B \leqslant \widetilde{Q}/2$, obtaining a solution σ'. If $f^A(\sigma') > f^A(\sigma^*)$, $\widetilde{Q}/2$ was indeed too small, so we try next $3\widetilde{Q}/4$, or else we decrease Q again to $\widetilde{Q}/4$. This goes on until we individuate the smallest value Q^* of Q such that the value of the optimal solution to $1\|f^A : f^B \leqslant Q$ is still equal to $f^A(\sigma^*)$. Clearly, $(f^A(\sigma^*), Q^*)$ is a nondominated pair. However, as we will see, in many cases the problem of finding a nondominated pair can be approached in a more efficient and straightforward way.

In the latter problem (*PP*), the two agents want to list all possible nondominated pairs, to negotiate the most acceptable trade-off for both.

The main focus of this paper is to analyze the complexity of these problems and propose solution algorithms. In some proofs, we make use of the following recognition problem:

Recognition Problem. Given two integers Q_A and Q_B, the job sets J^A and J^B of the two agents, and the two objective functions $f^A(\cdot)$ and $f^B(\cdot)$, find whether a feasible schedule σ exists such that $f^B(\sigma) \leqslant Q_B$ and $f^A(\sigma) \leqslant Q_A$. We refer to this problem as $1\|f^A \leqslant Q_A, f^B \leqslant Q_B$.

3.1. Symmetric Scenarios

Observe that $1\|f : g$ and $1\|g : f$ are indeed equivalent, because they are reducible to each other by means of a binary search. In fact, suppose we want to solve $1\|f : g \leqslant \overline{Q}$ for a given $\overline{Q}$, and we have an algorithm for solving $1\|g : f \leqslant Q$ for any Q. Because the optimal solution value of $1\|g : f \leqslant Q$ is nonincreasing for increasing values of Q, by iteratively solving $1\|g : f \leqslant Q$ for different values of Q, we can find the threshold value f^* such that, when $Q < f^*$, the optimal value of g is strictly greater than $\overline{Q}$, while for $Q = f^*$ the optimal value is smaller or equal to $\overline{Q}$. Clearly, such f^* is the optimal solution value of $1\|f : g \leqslant \overline{Q}$. If Q_0 is an upper bound on the optimal solution value of $1\|f : g \leqslant \overline{Q}$, then f^* can be found by solving $O(\log_2 Q_0)$ instances of $1\|g : f \leqslant Q$. Because of this symmetric relationship, in the single-machine case we consider only six distinct scenarios, addressed in §§4 to 9. The corresponding results are summarized in Table 1.

4. $1\|f^A_{\max} : f^B_{\max}$

Here we address the problem of finding an optimal solution to $1\|f^A_{\max} : f^B_{\max}$. We show that this problem can be efficiently solved by an easy reduction to the well-known, single-agent problem $1|prec|f_{\max}$. For the latter problem, Lawler (1973) devises an $O(n^2)$ time algorithm, which can be seen as an extension of a previous algorithm proposed by Livshits (1969) for the problem without precedence constraints.

Given a schedule σ for the two job sets $i \in J^A$ and J^B, we indicate, as usual, with $C_i(\sigma)$ the completion time of the ith job ($i \in J^A \cup J^B$). Let

$$f_{\max} = \max_{i \in J^A \cup J^B} \{f_i(C_i(\sigma))\},$$

where, for $t \geqslant 0$,

$$f_i(t) = \begin{cases} f_i^A(t) & \text{if } i \in J^A, \\ \infty & \text{if } i \in J^B \text{ and } f_i^B(t) > Q, \\ -\infty & \text{if } i \in J^B \text{ and } f_i^B(t) \leqslant Q. \end{cases}$$

With these positions, if a feasible schedule σ^* exists solving $1|prec|f_{\max}$ with finite objective function value $f^*_{\max}$, then σ^* is also feasible and optimal for $1\|f^A_{\max} : f^B_{\max} \leqslant Q$, and achieves the same optimal objective function value $f^A_{\max} = f^*_{\max}$. Otherwise, if $f_{\max} = \infty$, then no feasible solution exists for the two-agents problem.

In view of the above reduction, Lawler's algorithm for this special case may be sketched as follows. At each step, the algorithm selects, among unscheduled jobs, the job to be scheduled last. If we let $\bar{\tau}$ be the sum of the processing times of the unscheduled jobs, then any unscheduled B-job J^B_k such that $f^B_k(\bar{\tau}) \leqslant Q$ can be scheduled to end at $\bar{\tau}$. If there is no such B-job, we schedule the A-job J^A_h for which $f^A_h(\bar{\tau})$ is minimum. If, at a certain point in the algorithm, all A-jobs have been scheduled and no B-job can be scheduled last, the instance is not feasible. (We observe that the above algorithm can be easily extended to the case in which precedence constraints exist among jobs, even across

the job sets J^A and J^B. This may be the case, for instance, of assembly jobs that require components machined and released by the other agent.)

For each B-job J_k^B, let us define a *deadline* D_k^B, such that $f_k^B(C_k^B) \leqslant Q$ for $C_k^B \leqslant D_k^B$ and $f_k^B(C_k^B) > Q$ for $C_k^B > D_k^B$. (If the inverse function $f_k^{B^{-1}}(\cdot)$ is available, the deadlines can be computed in constant time; otherwise this requires logarithmic time.) The job set J^B can be ordered a priori, in nondecreasing order of deadlines D_k^B, in time $O(n_B \log n_B)$. At each step the only B-job that needs to be considered is the unscheduled one with largest D_k^B. On the other hand, for each job in J^A, the corresponding $f_h^A(\bar{\tau})$ value must be computed. If we suppose that each $f_h^A(\cdot)$ value can be computed in constant time, whenever no B-job can be scheduled, all unscheduled A-jobs may have to be tried out. Because this happens n_A times, we may conclude with the following:

THEOREM 4.1. *$1\|f_{\max}^A : f_{\max}^B$ can be solved in time $O(n_A^2 + n_B \log n_B)$.*

4.1. Finding a Nondominated Schedule

Using the above algorithm, we obtain an optimal solution σ^* to $1\|f_{\max}^A : f_{\max}^B \leqslant Q$. Let $Q_A = f_{\max}^A(\sigma^*)$ and $Q_B = f_{\max}^B(\sigma^*)$. In general, we are not guaranteed that σ^* is nondominated. To find an optimal solution which is also nondominated, we only need to exchange the roles of the two agents, and solve an instance of *CP* in which the cost for the agent A is bounded by Q_A, i.e., $1\|f_{\max}^B : f_{\max}^A \leqslant Q_A$. Call $\tilde{\sigma}$ the optimal schedule obtained in this way. Note that $f_{\max}^A(\tilde{\sigma}) = Q_A$ (because otherwise σ^* would not be optimal for the original problem).

THEOREM 4.2. *The schedule $\tilde{\sigma}$ is nondominated.*

PROOF. Suppose a schedule σ' exists which dominates $\tilde{\sigma}$. The schedule σ' cannot be strictly better than $\tilde{\sigma}$ for both agents, because otherwise $\tilde{\sigma}$ would not be optimal for both $1\|f_{\max}^A : f_{\max}^B \leqslant Q$ and $1\|f_{\max}^B : f_{\max}^A \leqslant Q_A$. If $f_{\max}^A(\sigma') = f_{\max}^A(\tilde{\sigma})$, for σ' to dominate $\tilde{\sigma}$ it must be $f_{\max}^B(\sigma') < f_{\max}^B(\tilde{\sigma})$. This is not possible, because $f_{\max}^A(\tilde{\sigma}) = Q_A$ and $\tilde{\sigma}$ is optimal for $1\|f_{\max}^B : f_{\max}^A \leqslant Q_A$. However, if $f_{\max}^B(\sigma') = f_{\max}^B(\tilde{\sigma})$, then, for σ' to dominate $\tilde{\sigma}$ it must be $f_{\max}^A(\sigma') < f_{\max}^A(\tilde{\sigma})$. This is also not possible, because $Q_B \leqslant Q$ and hence σ' would be better than σ^* in $1\|f_{\max}^A : f_{\max}^B \leqslant Q$. □

5. $1\|\sum w_i C_i^A : f_{\max}^B$

Here we address the problem in which the agent A wants to minimize total weighted completion time, given that the agent B only accepts schedules such that $\max_k\{f_k^B(C_k^B)\}$ does not exceed Q. As in §4, because all functions $f_k^B(C_k^B)$ are regular, for each job J_k^B we can define a deadline D_k^B.

The complexity of the problem is different for the weighted and the unweighted cases. For this reason we address them separately.

5.1. Weighted Case

We next show that $1\|\sum w_i C_i^A : f_{\max}^B$ is binary NP-hard, even when the objective function of the agent B is maximum completion time, i.e., $f_{\max}^B = C_{\max}^B$. We use the well-known NP-complete Knapsack Problem (Garey and Johnson 1979):

PROBLEM 5.1. KNAPSACK. Given two sets of nonnegative integers $\{u_1, u_2, \ldots, u_n\}$ and $\{w_1, w_2, \ldots, w_n\}$, and two integers b and W, is there a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i\in S} u_i \leqslant b$ and $\sum_{i\in S} w_i \geqslant W$?

THEOREM 5.2. *$1\|\sum w_i C_i^A : C_{\max}^B$ is binary NP-hard.*

PROOF. We reduce Knapsack to $1\|\sum w_i C_i^A \leqslant Q_A, C_{\max}^B \leqslant Q_B$. Given an instance of Knapsack, let $\hat{u} = \sum_{i=1,\ldots,n} u_i$ and $\hat{w} = \sum_{i=1,\ldots,n} w_i$. Consider the following instance of $1\|\sum w_i C_i^A \leqslant Q_A, C_{\max}^B \leqslant Q_B$. The agent A has $n_A = n$ jobs, having processing times $p_i^A = u_i$ and weights $w_i^A = w_i$, $i = 1, \ldots, n$. The agent B has only one job, having processing time $p_1^B = \hat{w}\hat{u}$. Finally, $Q_B = b + p_1^B$ and $Q_A = \hat{w}\hat{u} + (\hat{w} - W)p_1^B = (1 + \hat{w} - W)p_1^B$.

Given a feasible solution to Knapsack, we can define a solution to $1\|\sum w_i C_i^A \leqslant Q_A, C_{\max}^B \leqslant Q_B$, sequencing the A-jobs in the set $\{J_i^A \mid i \in S\}$ before the B-job (and the others after the B-job). Hence, the completion time of the B-job does not exceed $b + p_1^B = Q_B$, while the total cost of the A-jobs, $\sum_{i=1,\ldots,n} w_i C_i^A$, can be written as

$$\sum_{i\in S} w_i C_i^A + \sum_{i\notin S} w_i (C_i^A - p_1^B) + p_1^B \sum_{i\notin S} w_i. \tag{1}$$

The sum of the first two terms in (1) is smaller than $\hat{w}\hat{u}$. Because $\sum_{i\in S} w_i \geqslant W$, then $\sum_{i\notin S} w_i \leqslant \hat{w} - W$, and the last term is therefore smaller than $(\hat{w} - W)p_1^B$. Hence, $\sum_{i=1,\ldots,n} w_i C_i^A \leqslant \hat{w}\hat{u} + (\hat{w} - W)p_1^B = Q_A$, and the solution is feasible for $1\|\sum w_i C_i^A \leqslant Q_A, C_{\max}^B \leqslant Q_B$.

However, given a feasible solution to $1\|\sum w_i C_i^A \leqslant Q_A$, $C_{\max}^B \leqslant Q_B$, let us define a solution to the corresponding Knapsack instance by letting S be the set of A-jobs sequenced before the B-job. Then,

$$\begin{aligned} Q_A &= (1 + \hat{w} - W)p_1^B \\ &\geqslant \sum_{i=1,\ldots,n} w_i C_i^A > \sum_{i\notin S} w_i C_i^A > p_1^B \sum_{i\notin S} w_i. \end{aligned} \tag{2}$$

Comparing the first and the last term in (2), we obtain $1 + \hat{w} - W > \sum_{i\notin S} w_i$ and hence, considering that all weights are integer, $\hat{w} - W \geqslant \sum_{i\notin S} w_i$, which implies $\sum_{i\in S} w_i \geqslant W$. Because the jobs in S are sequenced before the B-job, then $\sum_{i\in S} p_i^A \leqslant Q_B - p_1^B = b$, and the set S provides a feasible solution to Knapsack. □

5.2. Unweighted Case

In this section, we show that $1\|\sum C_i^A : f_{\max}^B$ is polynomially solvable. Two lemmas allow us to devise the solution algorithm for this problem.

LEMMA 5.3. *Consider a feasible instance of* $1\|\sum C_i^A : f_{\max}^B$, *and let* $\tau = P_A + P_B$. *If there is a B-job* $J_{\tilde{k}}^B$ *such that* $f_{\tilde{k}}^B(\tau) \leqslant Q$, *then there is an optimal schedule in which* $J_{\tilde{k}}^B$ *is scheduled last, and there is no optimal schedule in which an A-job is scheduled last.*

PROOF. Let σ' be an optimal schedule in which $J_{\tilde{k}}^B$ is not scheduled last, and let σ^* be the schedule obtained by moving $J_{\tilde{k}}^B$ to the last position. For any job J_i^X other than $J_{\tilde{k}}^B$, $C_i^X(\sigma^*) \leqslant C_i^X(\sigma')$, and therefore $\sum C_i^A(\sigma^*) \leqslant \sum C_i^A(\sigma')$. In particular, if an A-job is last in σ', then $\sum C_i^A(\sigma^*) < \sum C_i^A(\sigma')$, thus contradicting the optimality of σ'. For what concerns $J_{\tilde{k}}^B$, its completion time is now τ, and by hypothesis $f_{\tilde{k}}^B(\tau) \leqslant Q$. Hence, due to the regularity of $f_k^B(\cdot)$ for all k, the schedule σ^* is still feasible and optimal. □

The second lemma specifies the order in which the A-jobs must be scheduled.

LEMMA 5.4. *Consider a feasible instance of* $1\|\sum C_i^A : f_{\max}^B \leqslant Q$, *and let* $\tau = P_A + P_B$. *If for all B-jobs* J_k^B, $f_k^B(\tau) > Q$, *then in any optimal schedule a longest A-job is scheduled last.*

PROOF. The result is established by a simple interchange argument. Let $J_{\tilde{h}}^A$ be a longest A-job, and let σ' be an optimal schedule in which $J_{\tilde{h}}^A$ is not scheduled last. By hypothesis, the last job in σ' is an A-job; call it J_l^A, such that $p_l^A \leqslant p_{\tilde{h}}^A$. Let σ^* be the schedule obtained by swapping J_l^A and $J_{\tilde{h}}^A$. For any job preceding $J_{\tilde{h}}^A$ in σ' nothing has changed, while $C_i^X(\sigma^*) \leqslant C_i^X(\sigma')$ for any job J_i^X between $J_{\tilde{h}}^A$ and J_l^A. In particular, if $p_l^A < p_{\tilde{h}}^A$, then $C_l^A(\sigma^*) + C_{\tilde{h}}^A(\sigma^*) < C_l^A(\sigma') + C_{\tilde{h}}^A(\sigma')$. Because the functions $f_h^B(\cdot)$ are regular, $f_{\max}^B(\sigma^*) \leqslant f_{\max}^B(\sigma')$, while also $\sum C_i^A(\sigma^*) \leqslant \sum C_i^A(\sigma')$, the equality holding if and only if $p_l^A = p_{\tilde{h}}^A$. □

The solution algorithm is similar to the one in §4. At each step, the algorithm selects a job to be scheduled last among unscheduled jobs. If possible, a B-job is selected. Otherwise, the longest A-job is scheduled last. If all A-jobs have been scheduled and no B-job can be scheduled last, the instance is infeasible. The algorithm is summarized in Figure 1.

THEOREM 5.5. $1\|\sum C_i^A : f_{\max}^B$ *can be solved in time* $O(n_A \log n_A + n_B \log n_B)$.

PROOF. Job set J^A can be ordered according to shortest processing times (SPT) in time $O(n_A \log n_A)$. J^B can be ordered in nondecreasing order of deadline D_k^B in time $O(n_B \log n_B)$. At each step there is only one candidate from each job set. Hence, the complexity of this algorithm is dominated by the ordering of the jobs, i.e., $O(n_A \log n_A + n_B \log n_B)$. □

5.2.1. Finding a Nondominated Schedule. The optimal solution obtained by the above algorithm may not be nondominated. The next lemma specifies the structure of any optimal solution to $1\|\sum C_i^A : f_{\max}^B$, thus including the nondominated ones. Given a feasible sequence σ, in what follows we define *B-block* as a maximal set of consecutive B-jobs in σ.

Figure 1. Scheme of the algorithm for $1\|\sum C_i^A : f_{\max}^B$.

```
{
σ := ∅; τ = P_A + P_B
while there are unscheduled jobs
  {
  if an unscheduled B-job J_k^B is such that f_k^B(τ) ≤ Q
  then J_i := J_k^B
  else if all A-jobs have been scheduled
       then no solution exists, STOP.
       else
         let J_h^A be the longest unscheduled A-job, J_i := J_h^A;
  σ := {J_i} ∘ σ
  τ := τ − p_i
  }
}
```

LEMMA 5.6. *Given a feasible instance of* $1\|\sum C_i^A : f_{\max}^B$, *for all optimal solutions*

(1) *the partition of B-jobs into B-blocks is the same, and*

(2) *the B-blocks are scheduled in the same time intervals.*

PROOF. Let σ^* and σ' be two optimal solutions. Lemmas 5.3 and 5.4 imply that the A-jobs are SPT ordered in both σ^* and σ'. If the sequences of the A-jobs are not the same in σ^* and σ', we can always swap some *identical* A-jobs, without affecting the starting time of the B-jobs, until we find a new optimal solution σ'' in which the sequence of the A-jobs is the same as in σ^*.

Now consider an A-job J_h^A in σ^*. Let t^* be its completion time in σ^* (i.e., $t^* = C_h^A(\sigma^*)$), and indicate by A_{prec} and B_{prec} the sets of A-jobs and B-jobs, respectively, preceding J_h^A in σ^*. We now observe two points:

(i) From Lemma 5.3, it follows that if $J_k^B \in B_{prec}$, then $f_k^B(t^*) > Q$, because otherwise J_k^B would have been scheduled to end at t^*, and J_h^A could have been scheduled earlier.

(ii) The A-jobs preceding J_h^A in σ^* and in σ'' are the same.

Now we claim that in σ'', J_h^A cannot complete earlier than t^*. In fact, suppose by contradiction that J_h^A completes before t^* in σ''. Point (i) above shows that all jobs in B_{prec} must complete strictly before t^* in any feasible solution, while point (ii) implies that also all jobs in A_{prec} would complete strictly before t^*. However, $\sum_{i \in A_{prec} \cup B_{prec}} p_i + p_h^A = t^*$, so there would be no room for other jobs to end in t^*, a contradiction. In conclusion, it is proved that

$$C_h^A(\sigma'') \geqslant C_h^A(\sigma^*). \tag{3}$$

Formula (3) holds for each A-job J_h^A in σ^*. Because σ'' is also optimal,

$$\sum_{h=1}^{n_A} C_h^A(\sigma^*) = \sum_{h=1}^{n_A} C_h^A(\sigma''). \tag{4}$$

(3) and (4) together imply $C_h^A(\sigma'') = C_h^A(\sigma^*)$ for $h = 1, \ldots, n_A$. So, the B-jobs preceding each A-job are the same in σ^* and σ''. Because the whole discussion can be repeated starting from any optimal solution, it turns out that in all optimal solutions the B-blocks occupy exactly the same time intervals. As a consequence of Lemma 5.3, exactly the same B-jobs are selected in each interval. □

Note that Lemma 5.6 completely characterizes the structure of the optimal solutions. The completion times of the A-jobs are the same in all optimal solutions, modulo permutations of identical jobs. The B-blocks are also the same in all optimal solutions, the only difference being the internal scheduling of each B-block. To find an optimal nondominated schedule, it is sufficient to slightly modify the algorithm in Figure 1. If $\bar{\tau}$ is the sum of the lengths of the currently unscheduled jobs, the algorithm in Figure 1 schedules *any* B-job J_k^B such that $f_k^B(\bar{\tau}) \leqslant Q$. Instead, we schedule the job J_l^B such that $f_l^B(\bar{\tau}) = \min_{k \in U^B}\{f_k^B(\bar{\tau})\}$ (where U^B stands for the unscheduled B-jobs). Ties are broken arbitrarily. Let $\tilde{\sigma}$ be the optimal schedule generated in this way.

THEOREM 5.7. *The schedule $\tilde{\sigma}$ is nondominated.*

PROOF. Lemma 5.6 shows that the partition of B-jobs into B-blocks is the same for all optimal solutions. Because A-jobs do not play any role within each block, the problem decomposes into several instances of $1\|f_{\max}^B$, one for each B-block. When building the schedule, if we pick every time the B-job for which $f_k^B(\bar{\tau})$ is minimum, the very same proof of Lawler's algorithm (Lawler 1973) restricted to $1\|f_{\max}$ implies that $\tilde{\sigma}$ minimizes $f_{\max}^B$. □

Note that selecting at each step the B-job of lowest cost implies an explicit computation of the $f_k^B(\cdot)$ functions. As a result, we cannot order the B-jobs a priori, and the following theorem holds.

THEOREM 5.8. *A nondominated optimal solution to $1\|\sum C_i^A : f_{\max}^B$ can be computed in time $O(n_A \log n_A + n_B^2)$.*

6. $1\|\sum U_i^A : f_{\max}^B$

Let us turn to the problem in which the agent A wants to minimize the number of late jobs, given that the agent B only accepts schedules in which $\max_k\{f_k^B(C_k^B)\} \leqslant Q$. As in the previous sections, we define a deadline D_k^B for each B-job J_k^B such that $f_k^B(C_k^B) \leqslant Q$ for $C_k^B \leqslant D_k^B$ and $f_k^B(C_k^B) > Q$ for $C_k^B > D_k^B$.

In what follows, we call the *latest start time* (LS_k) of job J_k^B the maximum value the starting time of J_k^B can attain in a feasible schedule such that $C_k^B \leqslant D_k^B$ for all $J_k^B \in J^B$. The values LS_k can be computed as follows. Order the B-jobs in nondecreasing order of D_k^B. Start from the last job, $J_{n_B}^B$. Schedule job $J_{n_B}^B$ to start at time $D_{n_B}^B - p_{n_B}^B$. Continue backwards, letting $LS_k := \min\{D_k^B, LS_{k+1}\} - p_k^B$ for all $k = n_B - 1, \ldots, 1$. Clearly, if job J_k^B starts after time LS_k, at least one B-job attains $f_k^B(C_k^B) > Q$.

Now consider, for each B-job J_k^B, the latest processing interval $[LS_k, D_k^B]$. Let $I = \bigcup_{k=1}^{n_B}[LS_k, D_k^B]$. Set I consists of a number $\beta \leqslant n_B$ of intervals, $I_{1,h_1}, I_{h_1,h_2}, \ldots, I_{h_{\beta-1},n_B}$, called *reserved intervals*. Each reserved interval $I_{u,v}$ ranges from LS_u to D_v^B. Note that, by construction, $\|I_{u,v}\| = D_v^B - LS_u = \sum_{k=u}^{v} p_k^B$. We say that jobs $J_u^B, J_{u+1}^B, \ldots, J_v^B$ are associated with $I_{u,v}$.

Let $1|pmtn|\sum U_i^A : f_{\max}^B$ be the preemptive variant of $1\|\sum U_i^A : f_{\max}^B$.

LEMMA 6.1. *Given an optimal solution to $1|pmtn|\sum U_i^A : f_{\max}^B$, there exists an optimal solution to $1\|\sum U_i^A : f_{\max}^B$ with the same number of late A-jobs.*

PROOF. Observe that if in the optimal solution to $1|pmtn|\sum U_i^A : f_{\max}^B$ 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 6.2. *There exists an optimal solution to $1|pmtn|\sum U_i^A : f_{\max}^B$ in which each B-job is nonpreemptively scheduled in the reserved interval it is associated with.*

PROOF. In an optimal solution to $1|pmtn|\sum U_i^A : f_{\max}^B$, all the B-jobs associated with interval $I_{u,v}$ complete before D_v^B. Hence, if we move all the pieces of each such job to exactly fit the interval $I_{u,v}$, we obtain a solution in which the completion time of no A-job has increased, because we only moved pieces of A-jobs backward. □

Lemma 6.2 allows us to fix the position of the B-jobs in an optimal solution to $1|pmtn|\sum U_i^A : f_{\max}^B$. The position of the A-jobs can then be found by solving an auxiliary instance of the well-known single-agent (nonpreemptive) $1\|\sum U_i$, solvable by Moore's algorithm (Moore 1968). Given an instance of $1|pmtn|\sum U_i^A : f_{\max}^B$, such an auxiliary instance consists of the A-jobs only, with modified due dates as follows. For each job J_h^A, if d_h^A falls outside any reserved interval, we subtract from d_h^A the total length of all the reserved intervals preceding d_h^A, i.e., we define the modified due date $\mathscr{D}_h^A$ as $\mathscr{D}_h^A = d_h^A - \sum_{u,v:\, D_v^B \leqslant d_h^A} \|I_{u,v}\|$. If d_h^A falls within the reserved interval $I_{p,q}$, we do the same, but instead of d_h^A we use the left extreme of $I_{p,q}$, i.e., we let $\mathscr{D}_h^A = LS_p - \sum_{u,v:\, D_v^B < d_h^A} \|I_{u,v}\|$.

THEOREM 6.3. *$1\|\sum U_i^A : f_{\max}^B$ can be solved in time $O(n_A \log n_A + n_B \log n_B)$.*

PROOF. Given a schedule σ for the auxiliary instance of $1\|\sum U_i$, it is possible to define a solution σ' to $1|pmtn|\sum U_i^A : f_{\max}^B$ by reinserting the reserved intervals (with the associated B-jobs) in the schedule, one at a time, from the first to the last, every time shifting everything forward. Each reinsertion can possibly preempt one A-job. From the definition of the due dates in the auxiliary instance, it follows immediately that each A-job is early

in σ' if and only if it is early in σ. Hence, from an optimal solution to the auxiliary instance we obtain an optimal solution to $1|pmtn|\sum U_i^A : f_{\max}^B$. Applying Lemma 6.1, we can obtain an optimal solution to $1\|\sum U_i^A : f_{\max}^B$ by rearranging those A-jobs that had been preempted during the reinsertion phase. Let us turn to complexity issues. The B-jobs are ordered first; complexity for this is $O(n_B \log n_B)$. Then, the computation of the reserved intervals takes time $O(n_B)$. The auxiliary instance can be defined in time $O(n_A)$ and solved in time $O(n_A \log n_A)$ by Moore's algorithm. The optimal solution to $1|pmtn|\sum U_i^A : f_{\max}^B$ can be reconstructed in time $O(n_A + n_B)$. Finally, the optimal solution to $1\|\sum U_i^A : f_{\max}^B$ is obtained in time $O(n_A + n_B)$. The overall complexity is therefore dominated by the ordering steps, and the theorem follows. □

7. $1\|\sum U_i^A : \sum U_i^B$

In this section, we address the problem in which the agent A wants to minimize the number of late A-jobs, while the agent B only accepts schedules in which at most Q late B-jobs are scheduled. We next show that this problem can be efficiently solved by dynamic programming. The following lemma relates to the structure of an optimal schedule.

LEMMA 7.1. *There is an optimal schedule σ^* of $1\|\sum U_i^A : \sum U_i^B$ 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, $\sum U_i^A(\sigma') \leqslant \sum U_i^A(\sigma^*)$, because we are moving the early jobs backward. Now consider 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 jobs in $J^A \cup J^B$ are numbered from J_1 to $J_{n_A+n_B}$ according to EDD order.

We next illustrate a recursion relation that can be exploited to design a polynomial dynamic programming algorithm for $1\|\sum U_i^A : \sum U_i^B$.

Let $C(i, h, k)$ be the minimum completion time of the last early job in a partial schedule of the job set $\{J_1, \ldots, J_i\}$ in which there are at most h late A-jobs and at most k late B-jobs. By definition, we set $C(i, h, k) = +\infty$ if no such schedule exists.

The following relations hold:

Boundary Conditions

$$C(0, 0, 0) = 0,$$

$$C(i, h, k) = +\infty \quad \text{if } i < 0 \text{ or } h < 0 \text{ or } k < 0.$$

Recursion Relation

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

$$C(i, h, k) = \begin{cases} \min\{C(i-1, h, k) + p_i + f(i, h, k); \\ \qquad C(i-1, h-1, k)\} & \text{if } J_i \in J^A, \\ \min\{C(i-1, h, k) + p_i + f(i, h, k); \\ \qquad C(i-1, h, k-1)\} & \text{if } J_i \in J^B. \end{cases}$$

LEMMA 7.2. *If $C(i, h, k)$ is finite, then it is the minimum completion time of the last early job over all feasible schedules for the job set $\{J_1, \ldots, J_i\}$, with at most h late A-jobs and k late B-jobs. If $C(i, h, k)$ 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 $h, k = 1, \ldots, n$. Now, assume that the property holds until $(i-1)$. We will show that the property holds also for i and for any h, k.

Let σ be a feasible schedule for the job set $\{J_1, \ldots, J_i\}$, such that the completion time τ of the last early job in σ is minimum among all feasible schedules with at most h late A-jobs and k late B-jobs. First, assume that J_i is an A-job. If J_i is late in σ, then, from the inductive hypothesis, $\tau = C(i-1, h-1, k)$. If J_i is early in σ, then, again from the inductive hypothesis, $\tau = C(i-1, h, k) + p_i$. Hence, the above recursion relation $\sum U_i^A$, $\sum U_i^B$ correctly chooses the smallest between the two quantities. Note that the schedule attaining $C(i, h, k)$ is feasible if either $C(i-1, h, k) + p_i < d_i$ or $C(i-1, h-1, k) < +\infty$. If neither of the two holds, there can be no feasible schedule of $\{J_1, \ldots, J_i\}$ with at most h late A-jobs and k late B-jobs, and the algorithm sets $C(i, h, k) = +\infty$. The proof is absolutely symmetrical if J_i is a B-job. If J_i is late in σ, then $\tau = C(i-1, h, k-1)$, whereas if J_i is early in σ, then $\tau = C(i-1, h, k) + p_i$. The schedule attaining $C(i, j, k)$ is feasible if either $C(i-1, h, k) + p_i < d_i$ or $C(i-1, h, k-1) < +\infty$. □

THEOREM 7.3. *The value $h^* = \min\{h : C(n_A + n_B, h, Q) < +\infty\}$ is an optimal solution value to $1\|\sum U_i^A : \sum U_i^B$, and it can be computed in time $O(n_A^2 n_B + n_A n_B^2)$.*

PROOF. Suppose that an optimal schedule σ for $1\|\sum U_i^A : \sum U_i^B$ exists in which $\sum U_i^A(\sigma) < h^*$. Without loss of generality, we can assume that σ has the structure illustrated in Lemma 7.1. Note that by definition of h^*, $C(n_A + n_B, \sum U_i^A(\sigma), Q) = +\infty$. Now let J_j be the last early job in σ. From Lemma 7.2, $C_j(\sigma) \geqslant C(n_A + n_B, \sum U_i^A(\sigma), Q)$. This implies $C(n_A + n_B, \sum U_i^A(\sigma), Q) < +\infty$, a contradiction. Therefore h^* is the optimal value for $1\|\sum U_i^A : \sum U_i^B$.

We now turn to complexity. Computing each $C(i, h, k)$ requires constant time, and therefore computing all of them requires $O(n_A^2 n_B + n_A n_B^2)$ time. Computing h^* requires $O(n_A)$ time. The overall complexity is therefore dominated by the former quantity. □

8. $1\|\sum w_i C_i^A : \sum U_i^B$

Here we observe that the problem in which the agent A wants to minimize the weighted sum of completion times, and the agent B accepts that up to a certain number of his or her own jobs are late is NP-hard. This follows straightforwardly from the NP-hardness of $1\|\sum w_i C_i^A : C_{\max}^B$, proved in §5 (Theorem 5.2). In fact, given an instance of $1\|\sum w_i C_i^A : C_{\max}^B \leqslant Q$, we can define an instance of $1\|\sum w_i C_i^A : \sum U_i^B \leqslant Q'$ having exactly the same jobs, all B-jobs have due date Q, and $Q' = 0$. Clearly, a feasible solution to the latter problem is a feasible solution to the former, and vice versa. Note that the complexity of the unweighted problem $1\|\sum C_i^A : \sum U_i^B$ is open.

9. $1\|\sum C_i^A : \sum C_i^B$

Here we consider the problem in which both agents wish to minimize their own total completion time. We show that even in the unweighted case, the problem is NP-hard.

First, the very same argument of Lemma 5.4 shows that, with no loss of generality, we can suppose that both agents order their jobs in SPT order. Hence, for simplicity we number the jobs of each agent accordingly. We use the following well-known NP-complete problem (Karp 1972):

PROBLEM 9.1. PARTITION. Given a set of k integers, $S = \{p_1, p_2, \ldots, p_k\}$, let $P = \sum_{i=1}^k p_i$. Is there a bipartition $(S, \bar{S})$ of S such that $\sum_{i\in S} p_i = \sum_{i\in\bar{S}} p_i = P/2$?

For the sake of simplicity, we number the integers in nondecreasing order, i.e., $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_k$. First, consider its recognition form. The following theorem establishes the complexity of $1\|\sum C_i^A : \sum C_i^B$.

THEOREM 9.2. *$1\|\sum C_i^A : \sum C_i^B$ is binary NP-hard.*

PROOF. Membership in NP is trivial. Given an instance of Partition, define an instance of $1\|\sum C_i^A \leqslant Q_A, \sum C_i^B \leqslant Q_B$, as follows. The two job sets J^A and J^B are identical, and each contains k jobs, having length $p_1, p_2, \ldots, p_k$. Moreover, we choose $Q_A = Q_B = (3/2)P + 2(\sum_{i=1}^k (k-i)p_i)$.

Consider a schedule σ having the following structure. The two jobs of length p_1 are scheduled first, followed by the two jobs of length $p_2, \ldots,$ followed by the two jobs of length p_k. We call SPT such a schedule. Note that there exist exactly 2^k SPT schedules, obtained by choosing in all possible ways the agent who has the precedence in a pair of jobs having the same length. Also, note that $\sum C_i^A(\sigma) + \sum C_i^B(\sigma)$ is the same for all SPT solutions. Simple arithmetic shows that this value is $T = 3P + 4\sum_{i=1}^k (k-i)p_i$, where $P = \sum_{i=1}^k p_i$. Note that $Q_A = Q_B = T/2$.

Now we consider the cost of an SPT schedule for each agent. We denote by $J[i]$ the pair of jobs J_i^A and J_i^B (both of length p_i). Given an SPT schedule, the notation $A \prec_i B$ means that in this schedule J_i^A precedes J_i^B in $J[i]$. Given an SPT schedule, observe that the contribution to the total completion time of the jobs in $J[1]$ is p_1 for one agent and $2p_1$ for the other, the contribution of the jobs in $J[2]$ is $2p_1 + p_2$ for one agent and $2p_1 + 2p_2$ for the other, $\ldots$, the contribution of the jobs in $J[h]$ is $2p_1 + 2p_2 + \cdots + 2p_{h-1} + p_h$ for one agent and $2p_1 + 2p_2 + \cdots + 2p_{h-1} + 2p_h$ for the other. Hence, in a given SPT schedule, the contribution of $J[h]$ to $\sum C_i^A$ can be obtained by adding to $2p_1 + 2p_2 + \cdots + 2p_{h-1} + p_h$ either 0 or p_h, depending on whether the agent A precedes the agent B in $J[h]$ or vice versa for each $h = 1, \ldots, k$. For the agent B the opposite holds, i.e., if $A \prec_i B$, then the value p_h is added. Given an SPT schedule, let $x(0, p_h) = 0$ if $A \prec_h B$ and $x(0, p_h) = p_h$ if $B \prec_h A$ in the solution, and let $x = \sum_{h=1}^k x(0, p_h)$. Hence, in any SPT schedule, the total completion time for the agent A is

$$\begin{aligned} &p_1 + x(0, p_1) + \\ &+2p_1 + p_2 + x(0, p_2) + \\ &+2p_1 + 2p_2 + p_3 + x(0, p_3) + \\ &\ldots \\ &+2p_1 + 2p_2 + \cdots + 2p_{h-1} + p_h + x(0, p_h) + \\ &\ldots \\ &+2p_1 + 2p_2 + \cdots + 2p_{k-1} + p_k + x(0, p_k) = \\ &P + 2\left(\sum_{i=1}^k (k-i)p_i\right) + x, \end{aligned} \tag{5}$$

whereas for the agent B it is

$$P + 2\left(\sum_{i=1}^k (k-i)p_i\right) + (P - x). \tag{6}$$

We next prove that if a feasible schedule σ for $1\|\sum C_i^A \leqslant T/2,\ \sum C_i^B \leqslant T/2$ exists, then σ is an SPT schedule.

Suppose in fact that a feasible schedule σ' exists that is *not* SPT. Then, there must be at least two consecutive jobs in σ', say J_i^A and J_j^B, such that $p_i > p_j$. Now if we swap the two jobs, we obtain a new schedule σ^* such that $\sum C_i^A(\sigma^*) = \sum C_i^A(\sigma') + p_j$ and $\sum C_i^B(\sigma^*) = \sum C_i^B(\sigma') - p_i$. Because $p_i > p_j$, one has $\sum C_i^A(\sigma') + \sum C_i^B(\sigma') > \sum C_i^A(\sigma^*) + \sum C_i^B(\sigma^*)$, i.e., the *overall* total completion time $\sum C_i^A + \sum C_i^B$ has decreased by the amount $p_i - p_j$. So, each B-job following a *longer* A-job can be swapped with the A-job, until no such pair of jobs exists. At each swap, the overall total completion time of the solution decreases. We could have symmetrical discussion for each A-job following a longer B-job. This time, if we swap them, the agent A gains p_u and the agent B loses p_v, and so the overall total completion time of the solution decreases by $p_u - p_v$. By repeatedly applying the above swaps, we eventually find an SPT solution. However, because the solution we started with, σ', was feasible, its overall total completion time cannot exceed $3P + 4\sum_{i=1}^k (k-i)p_i$. At each swap, the overall total completion time of the solution actually decreased. However, we ended up with an SPT schedule, whose weight is exactly $3P + 4\sum_{i=1}^k (k-i)p_i$, a

contradiction. Therefore, only SPT schedules can be feasible. For a schedule to be feasible, the total completion time for both agents must be $T/2$. Recalling the expressions (5) and (6) of the completion times for the agents A and B in an SPT solution, we observe that a feasible solution to $1\|\sum C_i^A \leqslant T/2, \sum C_i^B \leqslant T/2$ may exist if and only if $x = P/2$, i.e., if and only if there is a solution to the instance of Partition. □

We next illustrate a dynamic-programming, pseudopolynomial algorithm for $1\|\sum C_i^A : \sum C_i^B$. The approach exploits the property that the jobs of J^A and J^B are SPT ordered. In what follows, we denote by $P(i, j)$ the sum of the processing times of the i shortest A-jobs and the j shortest B-jobs.

Let $F(i, j, q)$ denote the value of an optimal solution to the instance of $1\|\sum C_i^A : \sum C_i^B$ in which only jobs $J_1^A, J_2^A, \ldots, J_i^A$ and $J_1^B, J_2^B, \ldots, J_j^B$ are considered. In an optimal solution to this problem, the last job is either J_i^A or J_j^B. In the former case, the contribution of J_i^A (given by $P(i-1, j) + p_i^A$) must be added to the optimal solution up to that point. In the latter case, the completion time of J_j^B is $P(i, j)$. Therefore, using the following dynamic-programming formula

$$F(i, j, q) = \min\{F(i-1, j, q) + P(i-1, j) + p_i^A; \; F(i, j-1, q - P(i, j))\}, \quad (7)$$

$F(n_A, n_B, Q)$ gives the optimal solution value. Because each quantity $F(i, j, q)$ can be computed in constant time, the following theorem holds:

THEOREM 9.3. *$1\|\sum C_i^A : \sum C_i^B$ can be solved in time $O(n_A n_B Q)$.*

Formula (7) must be suitably initialized, by setting $F(0, 0, q) = 0$ for all $q = 0, \ldots, Q$ and $F(i, j, q) = +\infty$ for $q < 0$.

10. Shop Problems

In this section, we address the case in which the processing resource shared by the two agents is a complex shop system. We will investigate the two simplest cases, i.e., two-machine flow shop and two-machine open shop, with the simplest objective function, i.e., $C_{\max}$. We will show that even in these cases, the problems are hard.

Let us briefly rule out a special shop problem, namely the case of a job shop in which each agent has exactly *one* job. For general nonregular, quasiconvex objective functions of the two jobs' completion times (note that this includes regular functions as a special case), Agnetis et al. (2000) show that the solution to *CP* can be found in time $O(n_A n_B \log n_A n_B + \log P)$, where n_A and n_B are the number of tasks of the two jobs, and P denotes the sum of all the processing times of the two jobs.

10.1. $F2\|C_{\max}^A : C_{\max}^B$

Let us consider a two-machine flow shop and let us limit ourselves to the simplest case, where both agents wish to complete their jobs as soon as possible. We show that even this problem is binary NP-hard. It is easy to show that any other combination of objective functions (among those considered in this paper) is also NP-hard.

Let p_{ih}^X be the processing time of job J_i^X on machine h $(X = A, B,\ h = 1, 2)$.

THEOREM 10.1. *$F2\|C_{\max}^A : C_{\max}^B$ is NP-hard.*

PROOF. We reduce Partition (see §9) to this problem. Denote the k integers of Partition as p_i, $i = 1, \ldots, k$. Let $P = \sum_{i=1}^k p_i$ and $\epsilon = 1/(k+1)$.

Consider an instance of the problem in recognition form, $F2\|C_{\max}^A \leqslant Q_A,\ C_{\max}^B \leqslant Q_B$, where the agent A has $n_A = k$ jobs, and the agent B has only one job $(n_B = 1)$. All the A-jobs have processing time $p_{i1}^A = \epsilon$ on the first machine, while the length of the jobs on the second machine is equal to $p_{i2}^A = p_i$ for all $i = 1, \ldots, k$. The B-job has processing times $p_{11}^B = P/2 - (k-1)\epsilon$ and $p_{12}^B = P/2$. Finally, let $Q_A = 3P/2 + \epsilon,\ Q_B = P + \epsilon$. The constraint $C_{\max}^B \leqslant Q_B = P + \epsilon$ can be satisfied only if the B-job starts processing on the second machine not after time $P/2 + \epsilon$, thus completing within time $P + \epsilon$. Hence, in a feasible solution, the total processing time of the A-jobs *preceding* J_1^B on the second machine (call S this subset of J^A) cannot be greater than $P/2$, otherwise J_1^B would complete after Q_B. Moreover, J_1^B cannot complete on the second machine before $p_{11}^B + p_{12}^B = P - (k-1)\epsilon$, and therefore the constraint $C_{\max}^A \leqslant Q_A = 3P/2 + \epsilon$ can be satisfied only if the total processing time of the A-jobs *following* J_1^B on the second machine is smaller or equal to $\lfloor P/2 + k\epsilon \rfloor = P/2$ (see Figure 2). Hence, because the total length of the A-jobs is P, the total length of the A-jobs preceding and following the B-job must equal $P/2$; that is, a feasible schedule exists if and only if the corresponding instance of Partition is a yes instance. □

10.2. $O2\|C_{\max}^A : C_{\max}^B$

A similar result holds for the open shop with two machines when both agents wish to complete their jobs as soon as possible. Again, p_{ih}^X denotes the processing time of job J_i^X on machine h $(X = A, B,\ h = 1, 2)$.

Figure 2. Reduction of Partition to $F2\|C_{\max}^A \leqslant Q_A$, $C_{\max}^B \leqslant Q_B$.

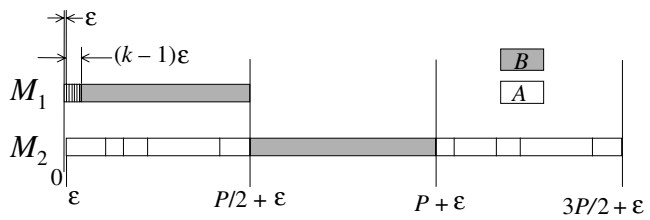
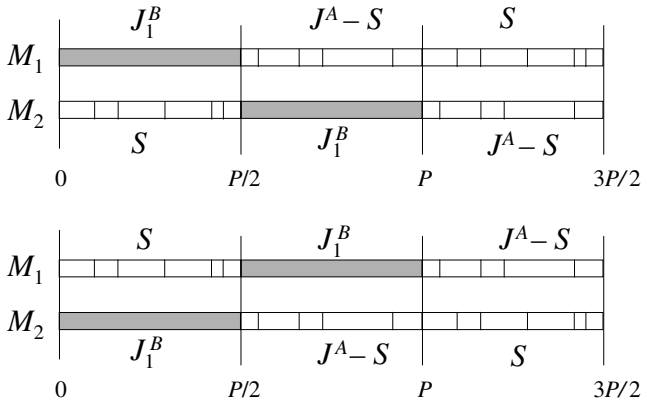


Figure 3. Reduction of Partition to $O2\|C^A_{\max} \leqslant Q_A$, $C^B_{\max} \leqslant Q_B$.



THEOREM 10.2. *$O2\|C^A_{\max} : C^B_{\max}$ is NP-hard.*

PROOF. We reduce Partition (see §9) to this problem. Given an instance of Partition, consider an instance of $O2\|C^A_{\max} \leqslant Q_A$, $C^B_{\max} \leqslant Q_B$, where the agent A has $n_A = k$ jobs, and the agent B has only one job ($n_B = 1$). Each A-job has the same processing time on the two machines, equal to the integers of Partition ($p^A_{i1} = p^A_{i2} = p_i$, $i = 1, \ldots, k$). The B-job has processing time $P/2$ on both machines, i.e., $p^B_{11} = p^B_{12} = P/2$. Finally, choose $Q_A = 3P/2$, $Q_B = P$. Note that for the B-job only two cases are possible, i.e., visit first M_1 and then M_2 or vice versa (see Figure 3). The constraint $C^B_{\max} \leqslant Q_B = P$ can be satisfied only if the B-job starts processing on one of the two machines at time 0 and on the other machine at time $P/2$, thus completing at time P. Hence, in a feasible solution, all the A-jobs processed from 0 to $P/2$ are processed on the same machine, and hence their total processing time must be smaller or equal to $P/2$. On the other hand, because no A-job completes after $Q_A = 3P/2$, and because $\sum_{i=1}^{n_A} p^A_i + \sum_{i=1}^{n_B} p^B_i = 2P$, the total processing time of the A-jobs processed from 0 to $P/2$ must equal $P/2$. Hence, a feasible schedule exists if and only if the corresponding instance of Partition is a yes instance. □

11. Pareto-Optimization Problems

In this section, we consider the situation in which the two agents wish to determine all the nondominated pairs $(y^A_1, y^B_1), (y^A_2, y^B_2), \ldots, (y^A_k, y^B_k)$. With each nondominated pair (y^A_i, y^B_i), we associate one nondominated schedule σ^i such that $f^A(\sigma^i) = y^A_i$, $f^B(\sigma^i) = y^B_i$. Let $\mathcal{S} = \{\sigma^1, \ldots, \sigma^k\}$. *PP* consists of determining $\mathcal{S}$.

A straightforward way to do this is to repeatedly solve instances of the corresponding *CP* for decreasing values of Q. After computing a nondominated optimal schedule for $1\|f^A : f^B \leqslant Q$, let Q' be the value of f^B in such a schedule (clearly, $Q' \leqslant Q$). We then solve $1\|f^A : f^B \leqslant Q' - \epsilon$, where $\epsilon > 0$ is small enough to ensure that no nondominated schedule is missed, and so on. Thus, we have the scheme shown in Figure 4.

Figure 4. Scheme for enumerating nondominated schedules.

```
PP {
   S := ∅; Q := +∞; i := 0
   while 1||f^A : f^B ≤ Q is feasible
      {
      i := i + 1
      σ^i := nondominated optimal solution to 1||f^A : f^B ≤ Q
      S := S ∪ σ^i
      Q' := f^B(σ^i)
      Q := Q' − ε
      }
   }
```

Note that, depending on the particular problem, we may have a number of nondominated pairs (and thus a cardinality for $\mathcal{S}$) which is polynomially bounded (or not). The scheme in Figure 4 turns out to be polynomial for those problems with a polynomial number of nondominated pairs, of course provided that the corresponding *CP* version is polynomially solvable. As we saw in §§4.1 and 5.2.1, in some cases we are able to find an optimal solution for *CP* which is also nondominated. Otherwise, we have to resort to a binary search (§3.1), but the computation time is still polynomial.

In this section, we elaborate on the cardinality of $\mathcal{S}$ in the single-machine cases for which we have solved the corresponding *CP*.

11.1. $1\|f^A_{\max} \circ f^B_{\max}$

Here we address the situation in which both agents want to minimize the maximum of regular functions.

LEMMA 11.1. *Let (y^A, y^B) and $(\tilde{y}^A, \tilde{y}^B)$ be two nondominated pairs, with $y^A < \tilde{y}^A$ and $y^B > \tilde{y}^B$. Given an A-job J^A_h and a B-job J^B_k, there exist two nondominated schedules σ and $\tilde{\sigma}$ (corresponding to (y^A, y^B) and $(\tilde{y}^A, \tilde{y}^B)$, respectively) such that if J^B_k precedes J^A_h in σ, then J^B_k precedes J^A_h also in $\tilde{\sigma}$.*

PROOF. Consider two nondominated schedules σ' and σ'', corresponding to (y^A, y^B) and $(\tilde{y}^A, \tilde{y}^B)$, respectively. We next prove that, if J^B_k precedes J^A_h in σ' and J^B_k follows J^A_h in σ'', it is possible to build two nondominated schedules σ and $\tilde{\sigma}$ for which the lemma holds. This is done in two different ways. In the first subcase (i), we show that there exist two nondominated schedules with J^B_k preceding J^A_h in both. In the second subcase (ii), we show that there exist two nondominated schedules with J^A_h preceding J^B_k in both.

(i) $C^B_k(\sigma'') \leqslant C^A_h(\sigma')$. Call α the sequence of jobs between J^A_h and J^B_k in σ''. We let $\sigma := \sigma'$ and let $\tilde{\sigma}$ be the schedule obtained from σ'' by moving J^A_h after job J^B_k, i.e., replacing the sequence $\ldots J^A_h, \alpha, J^B_k \ldots$ with $\ldots \alpha, J^B_k, J^A_h \ldots$. Note that in $\tilde{\sigma}$, job J^A_h completes at time $C^B_k(\sigma'')$, and therefore its associated cost $f^A_h(C^B_k(\sigma''))$ is

not greater than $f_h^A(C_h^A(\sigma')) \leqslant y^A < \tilde{y}^A$. However, no job other than J_h^A is delayed in $\tilde{\sigma}$ as compared to σ''. Hence, $\tilde{\sigma}$ is another nondominated schedule for the pair $(\tilde{y}^A, \tilde{y}^B)$.

(ii) $C_k^B(\sigma'') > C_h^A(\sigma')$. Call ϕ the sequence of jobs between J_k^B and J_h^A in σ'. We let $\tilde{\sigma} := \sigma''$ and let σ be obtained from σ' by moving J_k^B after job J_h^A, i.e., replacing the sequence $\ldots J_k^B, \phi, J_h^A \ldots$ with $\ldots \phi, J_h^A, J_k^B \ldots$. Note that in σ, job J_k^B completes at time $C_h^A(\sigma')$, and therefore its associated cost $f_k^B(C_h^A(\sigma'))$ is not greater than $f_k^B(C_k^B(\sigma'')) \leqslant \tilde{y}^B < y^B$. However, no job other than J_k^B is delayed in σ as compared to σ'. Hence, σ is another nondominated schedule for the pair (y^A, y^B). □

LEMMA 11.2. *Let (y^A, y^B) and $(\tilde{y}^A, \tilde{y}^B)$ be two nondominated pairs, with $y^A < \tilde{y}^A$ and $y^B > \tilde{y}^B$. There exist two nondominated schedules σ and $\tilde{\sigma}$ (corresponding to (y^A, y^B) and $(\tilde{y}^A, \tilde{y}^B)$, respectively) such that if J_k^B precedes J_h^A in σ, then J_k^B precedes J_h^A also in $\tilde{\sigma}$ for any $J_h^A \in J^A$ and $J_k^B \in J^B$.*

PROOF. Consider two nondominated schedules σ' and σ'', corresponding to (y^A, y^B) and $(\tilde{y}^A, \tilde{y}^B)$, respectively, and any two jobs $J_h^A \in J^A$ and $J_k^B \in J^B$. If J_h^A precedes J_k^B in σ' or J_k^B precedes J_h^A in σ'', the lemma holds. So, we only need to take care of the *contrary* job pairs J_h^A, J_k^B such that J_h^A follows J_k^B in σ' and J_h^A precedes J_k^B in σ''. Consider one contrary job pair. By applying the constructive proof of Lemma 11.1, we can enforce the condition of Lemma 11.1 for J_h^A, J_k^B, and thus eliminate such a contrary pair. This is done by either delaying J_k^B in σ' or by delaying J_h^A in σ''. Delaying a B-job in σ' can only increase the number of A-jobs preceding that B-job. Similarly, delaying an A-job in σ'' can only increase the number of B-jobs preceding that A-job. Hence, in both cases we cannot create any new contrary job pair. By repeatedly applying Lemma 11.1, we eventually find two nondominated schedules σ and $\tilde{\sigma}$ having no contrary job pairs. □

THEOREM 11.3. *There are at most $n_A n_B$ nondominated schedules in $1\|f_{\max}^A \circ f_{\max}^B$.*

PROOF. Starting from $Q = +\infty$, the scheme *PP* generates a succession of nondominated schedules $\sigma^1, \sigma^2, \ldots$. Let σ and $\tilde{\sigma}$ denote the two nondominated schedules obtained at consecutive iterations of the scheme for decreasing values of Q, corresponding to nondominated pairs (y^A, y^B) and $(\tilde{y}^A, \tilde{y}^B)$, respectively. Let $J_{k^*}^B$ be the job attaining the cost value y^B in σ. Note that $C_{k^*}^B(\tilde{\sigma}) < C_{k^*}^B(\sigma)$, because $\tilde{y}^B < y^B$. So, some jobs preceding $J_{k^*}^B$ in σ must follow it in $\tilde{\sigma}$. Among them, there must be at least one A-job. If not, then at least one B-job preceding $J_{k^*}^B$ in σ would complete at time $C_{k^*}^B(\sigma)$ or later, thus attaining a cost value not smaller than y^B, a contradiction. Thus, there must be at least one pair of jobs $J_h^A \in J^A$, $J_k^B \in J^B$, which exchange their relative ordering when switching from σ to $\tilde{\sigma}$. Lemma 11.2 guarantees that these two jobs will not reverse their relative ordering when Q is further decreased. Hence, throughout the execution of the scheme *PP*, each B-job overtakes each A-job at most once. Even supposing that two consecutive nondominated schedules only differ for one such pair of jobs, there can be no more than $n_A n_B$ nondominated schedules. □

We already observed that the scheme *PP* in Figure 4 must be applied with a sufficiently small $\epsilon > 0$ in order not to miss any nondominated solution. In this case, the ϵ to be used depends on the actual shape of the f functions. If their slope is small, small values of ϵ may be needed.

11.2. $1\|\sum C_i^A \circ f_{\max}^B$

We next address only the situation in which one of the two agents has $\sum C_i$ as his or her objective function. The more general case of $\sum w_i C_i$ is open.

Without loss of generality, let A be the agent with $\sum C_i$ objective. From Lemma 5.4 we know that in any nondominated schedule, the jobs of J^A are SPT ordered. As Q decreases, the optimal schedule for $1\|\sum C_i^A : f_{\max}^B \leqslant Q$ changes. The next lemma shows that when the constraint on the objective function of the agent B becomes tighter, the completion time of no A-job can decrease.

LEMMA 11.4. *Let σ be an optimal schedule for $1\|\sum C_i^A : f_{\max}^B \leqslant Q$, and consider job $j \in J^A$. Let σ' be an optimal schedule for $1\|\sum C_i^A : f_{\max}^B \leqslant Q'$, with $Q' < Q$. Then, $C_j(\sigma') \geqslant C_j(\sigma)$.*

PROOF. Suppose that the opposite holds, that is, $C_j(\sigma') < C_j(\sigma)$. Because the A-jobs are always SPT ordered, the A-jobs preceding j in σ' are the same as in σ. Therefore, there must be some B-jobs preceding j in σ and following j in σ', whose cumulative processing time is at least $C_j(\sigma) - C_j(\sigma')$. Among these B-jobs, let u be the one which is completed last in σ'. Hence, $C_u(\sigma') \geqslant C_j(\sigma)$. Because $f_u(C_u(\sigma')) \leqslant Q'$ and $Q' < Q$, then also $f_u(C_j(\sigma)) < Q$. The latter inequality implies that if, in σ, we move u from its current position to the position immediately after j, we obtain a new schedule which is certainly feasible for $1\|\sum C_i^A : f_{\max}^B \leqslant Q$, and is strictly better than σ, because the completion time of j has decreased by p_u. This is a contradiction. □

A straightforward consequence of the above lemma is the following.

LEMMA 11.5. *Let σ be an optimal schedule for $1\|\sum C_i^A : f_{\max}^B \leqslant Q$, and suppose that $k \in J^B$ precedes $j \in J^A$ in σ. Then, if σ' is an optimal schedule for $1\|\sum C_i^A : f_{\max}^B \leqslant Q'$, with $Q' < Q$, k precedes j also in σ'.*

PROOF. Suppose by contradiction that $k \in J^B$ follows $j \in J^A$ in σ'. From Lemma 11.4, $C_j(\sigma') \geqslant C_j(\sigma)$. So, $C_k(\sigma') > C_j(\sigma)$. Therefore, moving k to follow right after j in σ results in a schedule, say σ'', with $C_k(\sigma'') = C_j(\sigma) < C_k(\sigma')$, and hence, is feasible for $1\|\sum C_i^A : f_{\max}^B \leqslant Q$. Because A-jobs between k and j in σ are done earlier in σ'', $\sum C_i^A(\sigma'') < \sum C_i^A(\sigma)$. This is a contradiction. □

Lemma 11.5 shows that, once a B-job *overtakes* (i.e., it is done before) an A-job, as Q is decreased, no reverse overtake can occur when Q decreases further.

THEOREM 11.6. *There are at most $n_A n_B$ nondominated schedules in $1\|\sum C_i^A \circ f_{\max}^B$.*

PROOF. Starting from $Q = +\infty$, the scheme *PP* generates a succession of nondominated schedules $\sigma^1, \sigma^2, \ldots$. Let σ and σ' denote the two nondominated schedules obtained at consecutive iterations of the scheme. Because the two schedules must be different, there must be at least one pair of jobs $h \in J^B$, $j \in J^A$, which exchange their relative ordering when switching from σ to σ', i.e., h follows j in σ and h precedes j in σ' (otherwise the value of $\sum C_i$ would be the same in both schedules). Lemma 11.5 guarantees that these two jobs will not reverse their relative ordering when Q is further decreased. Hence, throughout the execution of the scheme *PP*, each B-job overtakes each A-job at most once. Even supposing that two consecutive nondominated schedules only differ for one such pair of jobs, there can be no more than $n_A n_B$ nondominated schedules. □

The scheme given in Figure 4 works for any $0 < \epsilon \leqslant 1$.

11.3. $1\|f_{\max}^A \circ \sum U_i^B$, $1\|\sum w_i C_i^A \circ \sum U_i^B$, and $1\|\sum U_i^A \circ \sum U_i^B$

When at least one of the agents has $\sum U_i$ as an objective function, the number of nondominated schedules is obviously linear. The scheme *PP* in Figure 4 can be applied with $0 < \epsilon \leqslant 1$.

11.4. $1\|\sum C_i^A \circ \sum C_i^B$

We have seen in §9 that finding one nondominated solution for $1\|\sum C_i^A \circ \sum C_i^B$ is binary NP-hard. We next show that the number of nondominated solutions may be exponential with respect to the instance size.

EXAMPLE 11.7. Consider an instance in which the sets J^A and J^B are identical. Each set consists of k jobs of size $p_0 = 1$, $p_1 = 2$, $p_2 = 4$, $p_3 = 8, \ldots, p_{k-1} = 2^{k-1}$. Now consider a subset of all possible schedules, namely those in which the two jobs of length p_0 are scheduled first, then the two jobs of length p_1, the two of length p_2, etc. Let σ be one such schedule. In σ, for each pair of jobs having equal length, either A's or B's job is scheduled first. Call $\bar{J}^A$ the set of job pair indices in which the A-job precedes the B-job having the same length in σ, and $\bar{J}^B$ the set of pair indices in which the opposite holds in σ. Consider job J_h^A. If it is scheduled before J_h^B, its contribution to the cost function is $2^h + 2(2^h - 1)$, otherwise it is $2(2^h) + 2(2^h - 1)$. Hence, the total completion time for the agent A is given by

$$\sum_{h\in J^A} C_h^A = \sum_{h\in \bar{J}^A} (2^{h+1} + 2^h - 2) + \sum_{h\in \bar{J}^B} (2^{h+2} - 2) \tag{8}$$

$$= \sum_{h\in \bar{J}^A} (3(2^h) - 2) + \sum_{h\in \bar{J}^B} (4(2^h) - 2) \tag{9}$$

$$= \sum_{h=0}^{k-1} 3(2^h) - 2k + \sum_{h\in \bar{J}^B} 2^h. \tag{10}$$

Note that only the last term in expression (10) depends on the actual schedule σ. This expression shows that the quantity $\sum_{h\in J^A} C_h^A$ may attain 2^k different values, one for each possible set $\bar{J}^B$. Symmetrically, the same analysis for the agent B yields

$$\sum_{h\in J^B} C_h^B = \sum_{h=0}^{k-1} 3(2^h) - 2k + \sum_{h\in \bar{J}^A} 2^h. \tag{11}$$

Because obviously $\sum_{h\in \bar{J}^B} 2^h + \sum_{h\in \bar{J}^A} 2^h = 2^k - 1$, for each choice of the set $\bar{J}^A$ we find a nondominated solution.

The scheme *PP* in Figure 4 works for any $0 < \epsilon \leqslant 1$.

12. Conclusions

In this paper, we propose a novel approach for modeling scheduling problems in a multiagent environment. In particular, we address the problem in which two agents compete for the usage of shared processing resources, and each agent has his or her own criterion to optimize. Our approach complements other contributions arising in different research areas, such as game theory, probabilistic scheduling, multiagent systems, and market mechanisms. Rather than analyzing negotiation protocols, we focus directly on schedule generation. In *PP*, we tackle the problem of generating a minimum set of alternative schedules to provide an operational basis for negotiation. In this case, the negotiation process takes place *after* a set of (Pareto-optimal) schedules has been generated, and consists of reaching an agreement over one of them. In the *CP* model, negotiation takes place *before* computing a schedule, and consists of specifying minimum performance values for one of the two agents, i.e., the value Q.

Several different directions for future research may be foreseen.

- Generalization of the problems to more than two agents. Note that in some cases such extension is fairly straightforward. For instance, the polynomial solution approaches of §§4 and 7 can be easily extended to the k-agents problems $1\|f_{\max}^1 : f_{\max}^2 \leqslant Q^2, \ldots, f_{\max}^k \leqslant Q^k$ and $1\|\sum U_i^1 : \sum U_i^2 \leqslant Q^2, \ldots, \sum U_i^k \leqslant Q^k$. On the contrary, the complexity of "mixed" situations such as, for instance, $1\|\sum C_i^1 : f_{\max}^2 \leqslant Q^2, \sum U_i^3 \leqslant Q^3$ is less obvious.
- Design of effective enumeration algorithms for computationally hard cases.
- Design of approximation algorithms and polynomial approximation schemes for hard cases.
- Extension to different resource usage modes (concurrent usage, preemption, etc.) and/or different system structures (e.g., parallel machines).
- Analysis of online scenarios or semi-online scenarios.

Acknowledgments

This work was partially inspired by Mario Lucertini, who unexpectedly passed away on March 16, 2002. The authors

dedicate this work to the generosity and enthusiasm of Mario, who taught three of the authors so much in operations research and life. The authors gratefully acknowledge the helpful comments and constructive suggestions of the associate editor and two anonymous referees.

References

Agnetis, A., P. B. Mirchandani, D. Pacciarelli, A. Pacifici. 2000. Nondominated schedules for a job-shop with two competing agents. *Comput. Math. Organ. Theory* **6**(2) 191–217.

Brewer, P. J., C. R. Plott. 1996. A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks. *Internat. J. Indust. Organ.* **14** 857–886.

Chen, C. L., R. L. Bulfin. 1990. Scheduling unit processing times jobs on a single machine with multiple criteria. *Comput. Oper. Res.* **17**(1) 1–7.

Chen, C. L., R. L. Bulfin. 1993. Complexity of single machine, multicriteria scheduling problems. *Eur. J. Oper. Res.* **70**(1) 115–125.

Chen, Y., Y. Peng, T. Finin, Y. Labrou, S. Cost, B. Chu, J. Yao, R. Sun, B. Wilhelm. 1999. A negotiation-based multi-agent system for supply chain management. *Proc. Agents'99 Workshop Agent-Based Decision-Support for Managing Internet-Enabled Supply-Chain.* Seattle, WA, 15–20.

Crès, H., H. Moulin. 2001. Scheduling with opting out: Improving upon random priority. *Oper. Res.* **49**(4) 565–577.

Curiel, I., G. Pederzoli, S. Tijs. 1989. Sequencing games. *Eur. J. Oper. Res.* **40** 344–351.

Fragnelli, V. 2001. On the balancedness of semi-infinite sequencing games. Preprint no. 442, Dipartimento di Matematica, Università di Genova, Genoa, Italy.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability*. Freeman, New York.

Graham, R. L., E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan. 1979. Optimization and approximation in deterministic machine scheduling: A survey. *Ann. Discrete Math.* **5** 287–326.

Hamers, H., P. Borm, S. Tijs. 1995. On games corresponding to sequencing situations with ready times. *Math. Programming* **70** 1–13.

Hamers, H., J. Suijs, S. Tijs, P. Borm. 1996. The split core for sequencing games. *Games Econom. Behavior* **15** 165–176.

Hoogeveen, J. A. 1992. *Single-Machine Bicriteria Scheduling*. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands.

Hoogeveen, J. A., S. L. van de Velde. 1995. Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time. *Oper. Res. Lett.* **17** 205–208.

Hoogeveen, J. A., S. L. van de Velde. 2001. Scheduling with target start times. *Eur. J. Oper. Res.* **129** 87–94.

Huang, X., J. C. Hallam. 1995. Spring-based negotiation for conflict resolution in AGV scheduling. *Proc. IEEE Internat. Conf. Systems, Man Cybernetics: Intelligent Systems for the 21st Century.* IEEE Press, Vancouver, WA, 789–794.

Karp, R. 1972. Reducibility among combinatorial problems. R. E. Miller, J. W. Thatcher, eds. *Complexity of Computer Computations*. Plenum Press, New York, 85–103.

Kim, K., B. C. Paulson, C. J. Petrie, V. R. Lesser. 1999. Compensatory negotiation for agent-based project schedule coordination. CIFE working paper #55, Stanford University, Stanford, CA.

Lawler, E. L. 1973. Optimal sequencing of a single machine subject to precedence constraints. *Management Sci.* **19** 544–546.

Lawler, E. L. 1983. Scheduling a single machine to minimize the number of late jobs. Working paper #CSD-83-139, Computer Science Division, University of California, Berkeley, CA.

Lenstra, J. K., A. H. G. Rinnooy Kan, P. Brucker. 1977. Complexity of machine scheduling problems. *Ann. Discrete Math.* **1** 343–362.

Livshits, E. M. 1969. Minimizing the maximal penalty in a single machine problem. *Trans. 1st Winter School Math. Programming*. Drogobych, Ukraine, 454–475 (in Russian).

Moore, J. M. 1968. An *n* job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Sci.* **15** 102–109.

Nagar, A., J. Haddock, S. Heragu. 1995. Multiple and bicriteria scheduling: A literature survey. *Eur. J. Oper. Res.* **81** 88–104.

Schultz, D., S.-H. Oh, C. F. Grecas, M. Albani, J. Sanchez, C. Arbib, V. Arvia, M. Servilio, F. Del Sorbo, A. Giralda, G. Lombardi. 2002. A *QoS* concept for packet oriented S-UMTS services. *Proc. 1st Mobile Summit* 2002. Thessaloniki, Greece.

Shen, L. 1998. Logistics with two competing agents. Ph.D. thesis, Faculty of Systems and Industrial Engineering, The University of Arizona, Tucson, AZ.

Smith, W. E. 1956. Various optimizers for single stage production. *Naval Res. Logist. Quart.* **3**(1) 59–66.

Wellman, M. P. 1993. A market-oriented programming environment and its applications to distributed multicommodity flow problems. *J. Artificial Intelligence Res.* **1** 1–23.

Wellman, M. P., W. E. Walsh, P. R. Wurman, J. K. MacKie-Mason. 2001. Auction protocols for decentralized scheduling. *Games Econom. Behavior* **35**(1–2) 271–303.