Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Scheduling problems with two competing users

A. Agnetis[1], P. B. Mirchandani[2], D. Pacciarelli[3], A. Pacifici[4]

RT-DIA-67-01 **July 2001**

(1) Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy.

(2) Department of Systems and Industrial Engineering, The University of Arizona, Tucson, USA.

(3) Dipartimento di Informatica e Automazione, Università Roma Tre, Italy.

(4) Centro Vito Volterra, Università di Roma "Tor Vergata", Italy.

This work is supported in part by the Italian Ministry of Scientific Research, grants "Models and Algorithms for Operational Planning in Distributed Systems" and "Metodi e Sistemi di Supporto alle decisioni".

ABSTRACT

We consider the scheduling problems arising when two users, each owning a set of nonpreemptive jobs, compete to perform their respective jobs on a common processing resource. Each user wants to minimize a certain objective function, which depends on the completion times of his/her jobs only. The objective functions we consider in this paper are: (*i*) maximum of regular functions (associated with each job), (*ii*) number of tardy jobs, and (*iii*) total weighted completion time. The problem consists in finding nondominated schedules, i.e., such that a better schedule for one of the two users necessarily results in a worse schedule for the other user. We get different scenarios, depending on the objective function of each user, and on the structure of the processing system (single-machine or shop). For each scenario, we address the complexity of the problem of finding single nondominated schedules, and we investigate the problem of determining their overall number.

1 Introduction

In recent years, management problems in which multiple users 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 Multi-Agent 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 decision problems faced by two competing agents who wish to schedule their tasks minimizing their own cost functions.

Literature on quantitative methods for scheduling problems from the multi-agent perspective mainly investigates the development of heuristic approaches for the construction of schedules that are acceptable to the users, with no particular concern on optimality. For instance, Kim et al. [9] 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 [7] 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. [5] propose a number of negotiation protocols for functional agent cooperation in a supply chain context. Brewer and Plott [4] devised a bidding mechanism for the problem of scheduling trains (agents) on a shared single railtrack. In the work by Ling [11], two users compete for a single machine. The users' objective functions are nonlinear, and a branch and bound approach is proposed for enumerating negotiation alternatives.

In this paper we review several classical single-user scheduling models, from a new two-user perspective. The main focus is to determine *nondominated* (or Pareto-optimal) solutions, i.e., such that a better solution for one user necessarily results in a worse solution for the other user. It is important to underline that *two-users problems* differ from *problems with two objective functions* (see for instance [6, 12, 3]), since in the latter case each job in the jobset contributes to all the objective functions, and the nondominated solutions do not pertain to trading one user's schedule with the other user's. In our analysis, we adopt two slightly different viewpoints. The first consists in determining the best solution to one user, given that the other user does not accept schedules having cost greater than a certain value for him/her. The second viewpoint consists in determining the set of all nondominated solutions, which provides the basis for negotiation between the two users.

The paper is organized as follows. In Section 2 we introduce the notation and terminology used throughout the paper. From Section 3 to Section 9 we characterize the complexity of the decision problems arising for different combinations of the two users' objective functions and different system structures (single machine, flow shop, job shop, open shop). In particular, From Section 3 to Section 8 we consider single-machine problems, in Section 9 two machine shop problems. In Section 10, we concentrate on enumerating all nondominated solutions. Conclusions follow in Section 11.

2 Problem definition and notation

In this section we introduce the notation and terminology we use throughout the paper. There are two competing users, called *user A* and *user B*. Each of them has a set of non-preemptive *jobs* to be done on a common processing resource. For the sake of simplicity, we next introduce the notation concerning the case in which the processing resource is a single machine. This notation will be generalized to more complex shop systems in Section 9.2.

User A has to execute the job set $J^A = \{J_1^A, J_2^A, \ldots, J_{n_A}^A\}$, whereas user 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 users will have to schedule his/her jobs on the common single machine complying with the presence of the other user'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 users. The starting times of job J_h^A and J_k^B in σ will be denoted as $S_h^A(\sigma)$ and $S_k^B(\sigma)$ respectively, whereas we use $C_h^A(\sigma)$ and $C_k^B(\sigma)$ for the completion times. We will use the notation J_h, p_h, d_h, $S_h(\sigma)$, $C_h(\sigma)$ 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 user X, which can be either A or B.

Each user has a certain objective function, which depends on the completion times of his/her 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 (for $X = A, B$):

- $f^X(\sigma) = \Gamma_{\max}^X(\sigma) = \max_{i=1,\ldots,n_X}\{\gamma_i^X(C_i^X(\sigma))\}$, where each $\gamma_i^X(\cdot)$ is a nondecreasing function of the completion time of job J_i^X (*maximum of regular functions*).
- $f^X(\sigma) = n_T^X(\sigma) = \sum_{i=1}^{n_X} U_i^X(\sigma)$, where $U_i^X(\sigma) = 1$ if job J_i^X is tardy in σ and zero otherwise (*number of tardy jobs*).
- $f^X(\sigma) = \overline{wC}^X(\sigma) = \sum_{i=1}^{n_X} w_i^X C_i^X(\sigma)$ (*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 $C_{\max}^X(\sigma)$ and the maximum lateness $L_{\max}^X(\sigma)$ as special cases. We use $\bar{C}^X$ for $\overline{wC}^X$ when $w_i^X = 1$ for all i.

We say that a schedule σ is *nondominated* if there is no schedule $\bar{\sigma}$ such that $f^A(\bar{\sigma}) \leq f^A(\sigma)$, $f^B(\bar{\sigma}) \leq f^B(\sigma)$ and at least one of the two inequalities is strict. In other words, a schedule is nondominated if a better schedule for one of the two users necessary results in a worse schedule for the other user. Distinct nondominated schedules $\sigma, \sigma', \ldots$ may yield the same pair of objective function values $(f^A(\sigma), f^B(\sigma)) = (f^A(\sigma'), f^B(\sigma')) = \ldots = (y^A, y^B)$. 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 *any* of them, not all of them.

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

PROBLEM 2.1 **Constrained Optimization Problem** (COP). *Given the job sets J^A and J^B of the two users, the two objective functions $f^A(\cdot)$ and $f^B(\cdot)$, and an integer Q, find a schedule σ^* such that $f^B(\sigma^*) \leq Q$, and $f^A(\sigma^*)$ is minimum.*

PROBLEM 2.2 **Pareto Optimization Problem** (POP). *Given the job sets J^A and J^B of the two users 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.*

Note that the viewpoints are slightly different in the two problems. In the former problem (*COP*), user A wants to find the best solution for him/her, given that user B will accept a schedule of cost up to Q. An instance of *COP* 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 a general bisection search. However, as we will see, in many cases it can be approached in a more efficient and straightforward way.

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

We indicate the two above problems by $COP(f^A|f^B \leq Q)$ and $POP\{f^A, f^B\}$. For instance, $POP\{C_{max}, \bar{C}\}$ indicates the problem of finding all nondominated pairs when the objective function of user A is makespan minimization and that of user B is total (unweighted) completion time.

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

PROBLEM 2.3 **Recognition Problem** REC. *Given two integers Q_A and Q_B, the job sets J^A and J^B of the two users, and the two objective functions $f^A(\cdot)$ and $f^B(\cdot)$, find wether a feasible schedule σ exists such that $f^B(\sigma) \leq Q_B$ and $f^A(\sigma) \leq Q_A$. We refer to this problem as* REC$(f^A \leq Q_A|f^B \leq Q_B)$.

Also, we will often refer to the following well-known NP-complete problem:

PROBLEM 2.4 (PARTITION). *Given a set of nonnegative integers $\{p_1, p_2, \ldots, p_k\}$ with $\sum_{i=1}^{n} p_i = P$, decide whether a subset $S \subseteq \{1, \ldots, k\}$ exists such that $\sum_{i \in S} p_i = P/2$.*

2.1 Symmetric scenarios

It is convenient to observe that some of the different scenarios are indeed equivalent. It is easy to see that $COP(f|g \leq Q)$ and $COP(g|f \leq Q)$ are reducible to each other by means of a binary search. In fact, suppose we want to solve $COP(f|g \leq \bar{Q})$, for a given $\bar{Q}$, and we have an algorithm for solving $COP(g|f \leq Q)$, for any Q. First, note that the optimal solution value of $COP(g|f \leq Q)$ is nonincreasing for increasing values of Q. By iteratively solving $COP(g|f \leq Q)$ for different values of Q, we can find the value f^* such that, for $Q < f^*$, the optimal value of g is strictly greater than $\bar{Q}$. Clearly, f^* is the optimal solution value of $COP(f|g \leq \bar{Q})$. If the starting value Q_0 of Q is an upper bound on the optimal solution value of $COP(f|g \leq \bar{Q})$, f^* can be found by solving $\lceil \log_2 Q_0 \rceil$ instances of $COP(g|f \leq Q)$. Because of this symmetric relationship, we consider only six distinct scenarios. These are addressed in Sections 3 to 8. Table 1 summarizes the main results for the single machine problems.

	$\Gamma^B_{\max}$	n^B_T	$\overline{wC}^B$
$\Gamma^A_{\max}$	$O(n \log n)$	-	-
n^A_T	$O(n \log n)$	$O(n^3)$	-
$\overline{wC}^A$	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard even if $w^A_i = w^B_i = 1$
	$O(n \log n)$ if $w^A_i = 1$	open if $w^A_i = 1$	

Table 1: Summary of complexity results for *COP* in the single machine case.

3 *COP*$(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$

Here we address the problem of finding an optimal solution to $COP(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$. Notice that, since all functions $\gamma^B_k(C^B_k)$ are regular, for each function $\gamma^B_k(C^B_k)$ it is possible to define a certain deadline D_k such that $\gamma^B_k(C^B_k) \leq Q$ for $C^B_k \leq D_k$ and $\gamma^B_k(C^B_k) > Q$ for $C^B_k > D_k$. (For instance, if $\gamma^B_k(C^B_k)$ is the lateness of job J^B_i and $Q = 0$, D_k coincides with the due date of job J^B_k).

$COP(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$ can be efficiently solved by means of a suitable modification of a well-known algorithm by Lawler [10]. The following lemmas are used to determine the last job of the schedule.

LEMMA 3.1 *Consider a feasible instance of* $\mathrm{COP}(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$*, and let* $\tau = P_A + P_B$*. If there is a B-job* $J^B_{\bar{k}}$ *such that* $\gamma^B_{\bar{k}}(\tau) \leq Q$*, then there is an optimal schedule in which* $J^B_{\bar{k}}$ *is scheduled last.*

Proof. Let σ' be an optimal schedule in which $J^B_{\bar{k}}$ is not scheduled last, and let σ^* be the schedule obtained by moving $J^B_{\bar{k}}$ in the last position. The completion time for $J^B_{\bar{k}}$ is now τ, and since by hypothesis $\gamma^B_{\bar{k}}(\tau) \leq Q$, the σ^* schedule is still feasible. Switching from σ' to σ^*, the completion time of no job other than $J^B_{\bar{k}}$ increases and therefore, since the functions $\gamma^A_h(\cdot)$ are regular, $\Gamma^A_{\max}(\sigma^*) \leq \Gamma^A_{\max}(\sigma')$. Therefore σ^* is optimal. ∎

LEMMA 3.2 *Consider a feasible instance of* $\mathrm{COP}(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$*, let* $\tau = P_A + P_B$*, and let* $J^A_{\bar{h}}$ *be such that* $\gamma^A_{\bar{h}}(\tau) = \min_h\{\gamma^A_h(\tau)\}$*. If* $\gamma^B_k(\tau) > Q$ *for all B-jobs* J^B_k*, then there is an optimal schedule in which* $J^A_{\bar{h}}$ *is scheduled last.*

Proof. Observe that no B-job can be scheduled last, and then use the same interchange argument in the proof of Lemma 3.1, except now we move $J^A_{\bar{h}}$ to last position. ∎

A solution algorithm can be devised which repeatedly uses Lemmas 3.1 and 3.2. 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 $\gamma^B_k(\bar{\tau}) \leq Q$ can be scheduled to end at $\bar{\tau}$ (Lemma 3.1). If there is no such B-job, we schedule the A-job J^A_h for which $\gamma^A_h(\bar{\tau})$ is minimum (Lemma 3.2). 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.

THEOREM 3.3 $\mathrm{COP}(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$ *can be solved in* $O(n^2_A + n_B \log n_B)$.

Proof. The job set J^B can be ordered a priori, in nondecreasing order of deadlines D_k, 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. On the other hand, for each job in J^A, the corresponding $\gamma_h^A(\bar{\tau})$ value must be computed. Supposing that each $\gamma_h^A(\cdot)$ value can be computed in constant time, whenever no B-job can be scheduled, all unscheduled A-jobs may have to be tried out. Since this happens n_A times, the theorem follows. ∎

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.

3.1 Getting a nondominated schedule

Using the above algorithm, we get an optimal solution σ^* to $COP(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$. Let $Q_A = \Gamma^A_{\max}(\sigma^*)$ and $Q_B = \Gamma^B_{\max}(\sigma^*)$. 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 users, and solve an instance of COP in which the cost for user A is bounded by Q_A, i.e., $COP(\Gamma^B_{\max}|\Gamma^A_{\max} \leq Q_A)$. Call $\tilde{\sigma}$ the optimal schedule obtained in this way. Note that $\Gamma^A_{\max}(\tilde{\sigma}) = Q_A$ (since otherwise σ^* is not optimal for the original problem).

THEOREM 3.4 *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 users, since otherwise $\tilde{\sigma}$ would not be optimal for both $COP(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$ and $COP(\Gamma^B_{\max}|\Gamma^A_{\max} \leq Q_A)$. If $\Gamma^A_{\max}(\sigma') = \Gamma^A_{\max}(\tilde{\sigma})$, for σ' to dominate $\tilde{\sigma}$ it must be $\Gamma^B_{\max}(\sigma') < \Gamma^B_{\max}(\tilde{\sigma})$. This is not possible, because $\Gamma^A_{\max}(\tilde{\sigma}) = Q_A$ and $\tilde{\sigma}$ is optimal for $COP(\Gamma^B_{\max}|\Gamma^A_{\max} \leq Q_A)$. On the other hand, if $\Gamma^B_{\max}(\sigma') = \Gamma^B_{\max}(\tilde{\sigma})$, then, for σ' to dominate $\tilde{\sigma}$, it must be $\Gamma^A_{\max}(\sigma') < \Gamma^A_{\max}(\tilde{\sigma})$. This is also not possible, since $Q_B \leq Q$ and hence σ' would be better than σ^* in $COP(\Gamma^A_{\max}|\Gamma^B_{\max} \leq Q)$. ∎

4 $COP(\overline{wC}^A|\Gamma^B_{\max} \leq Q)$

Here we address the problem in which A wants to minimize total weighted flow time, given that B only accepts schedules such that $\max_k\{\gamma_k^B(C_k^B)\}$ does not exceed Q. As in Section 3, since all functions $\gamma_k^B(C_k^B)$ are regular, for each job J_k^B we can define a deadline D_k.

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

4.1 Weighted case

We next show that the recognition form of $COP(\overline{wC}^A|\Gamma^B_{\max} \leq Q)$ is NP-complete, even when the objective function of user B is maximum completion time, i.e., $\Gamma^B_{\max} = C^B_{\max}$. We use the well-known NP-completeness property of the KNAPSACK problem [2]:

PROBLEM 4.1 KNAPSACK. *Given a set of nonnegative integers $\{u_1, u_2, \ldots, u_n\}$, a set of weights $\{w_1, w_2, \ldots, w_n\}$ and two integers b and W, decide whether a subset $S \subseteq \{1, \ldots, n\}$ exists such that $\sum_{i \in S} u_i \leq b$ and $\sum_{i \in S} w_i \geq W$.*

THEOREM 4.2 *Problem* REC($\overline{wC}^A \leq Q_A | C^B_{\max} \leq Q_B$) *is NP-complete.*

Proof. We reduce KNAPSACK to Problem $REC(\overline{wC}^A \leq Q_A | C^B_{\max} \leq Q_B)$. 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 $REC(\overline{wC}^A \leq Q_A | C^B_{\max} \leq Q_B)$. User A has $n_A = n$ jobs, having processing times $p^A_i = u_i$ and weights $w^A_i = w_i$, $i = 1, \ldots, n$. User B has only one job, having processing time $p^B_1 = \hat{w}\hat{u}$. Finally, $Q_B = b + p^B_1$ and $Q_A = \hat{w}\hat{u} + (\hat{w} - W)p^B_1$.

A feasible solution to KNAPSACK corresponds to a solution to $REC(\overline{wC}^A \leq Q_A | C^B_{\max} \leq Q_B)$ in which the A-jobs in the set $\{J^A_i | i \in S\}$ are sequenced before the B-job. Clearly the completion time of the B-job does not exceed Q_B, while the cost of the A-jobs can be computed as follows: $\sum_{i=1,\ldots n} w_i C^A_i = \sum_{i \in S} w_i C^A_i + \sum_{i \notin S} w_i (C^A_i - p^B_1) + p^B_1 \sum_{i \notin S} w_i$. The two former quantities are smaller than $\hat{w}\hat{u}$. Since $\sum_{i \in S} w_i \geq W$, then $\sum_{i \notin S} w_i \leq \hat{w} - W$, and the latter quantity is smaller than $(\hat{w} - W)p^B_1$. Hence, $\sum_{i=1,\ldots n} w_i C^A_i \leq \hat{w}\hat{u} + (\hat{w} - W)p^B_1 = Q_A$.

On the other hand, a feasible solution to $REC(\overline{wC}^A \leq Q_A | C^B_{\max} \leq Q_B)$ corresponds to a feasible solution to KNAPSACK in which the set S corresponds to those jobs that are sequenced before the B-job. In fact, in a feasible solution to $REC(\overline{wC}^A \leq Q_A | C^B_{\max} \leq Q_B)$ the total weight of the A-jobs sequenced after the B-job must be smaller of equal than $\hat{w} - W$, thus ensuring that the weight of the jobs sequenced before the B-job is greater or equal than W. ∎

4.2 Unweighted case

In this section we show that $COP(\bar{C}^A | \Gamma^B_{\max} \leq Q)$is polynomially solvable. Two lemmas allow us to devise the solution algorithm for this problem. The first is very similar to Lemma 3.1, so we omit the proof.

LEMMA 4.3 *Consider a feasible instance of* COP($\bar{C}^A | \Gamma^B_{\max} \leq Q$)*, and let* $\tau = P_A + P_B$*. If there is a* B*-job* J^B_k *such that* $\gamma^B_k(\tau) \leq Q$*, then there is an optimal schedule in which* J^B_k *is scheduled last.*

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

LEMMA 4.4 *Consider a feasible instance of* COP($\bar{C}^A | \Gamma^B_{\max} \leq Q$)*, let* $\tau = P_A + P_B$*, and let* $J^A_{\bar{h}}$ *be the longest* A*-job. If for all* B*-jobs* J^B_k*,* $\gamma^B_k(\tau) > Q$*, then there is an optimal schedule in which* $J^A_{\bar{h}}$ *is scheduled last.*

Proof. The result is established by a simple interchange argument. Let σ' be an optimal schedule in which $J^A_{\bar{h}}$ is not scheduled last. By hypothesis, the last job in σ' is an A-job, call it J^A_l, such that $p^A_l \leq p^A_{\bar{h}}$. Let σ^* be the schedule obtained by swapping J^A_l and $J^A_{\bar{h}}$. For any job preceding $J^A_{\bar{h}}$ in σ' nothing has changed, while $C^X_i(\sigma^*) \leq C^X_i(\sigma')$ for any job J^X_i between $J^A_{\bar{h}}$ and J^A_l. Since the functions $\gamma^B_h(\cdot)$ are regular, $\Gamma^B_{\max}(\sigma^*) \leq \Gamma^B_{\max}(\sigma')$, while also $\bar{C}^A(\sigma^*) \leq \bar{C}^A(\sigma')$, the equality holding if and only if $p^A_l = p^A_{\bar{h}}$. ∎

The solution algorithm is similar to the one in Section 3. At each step, the algorithm selects a job to be scheduled last among unscheduled jobs. If possible, a B-job is selected. Else, 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.

THEOREM 4.5 $\mathrm{COP}(\bar{C}^A|\Gamma^B_{\max} \leq Q)$ *can be solved in* $O(n_A \log n_A + n_B \log n_B)$.

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 D_k 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., $O(n_A \log n_A + n_B \log n_B)$. ∎

4.2.1 Getting a nondominated schedule

The optimal solution obtained by the above algorithm may not be nondominated. To get an optimal nondominated schedule, it is sufficient to slightly modify the above algorithm. Let $\bar{\tau}$ be the current scheduling time, i.e., the sum of the lengths of the unscheduled jobs (at the beginning of the algorithm, $\bar{\tau} = P_A + P_B$). The algorithm in the previous section would schedule *any* B-job J^B_k such that $\gamma^B_k(\bar{\tau}) \leq Q$. Instead of doing so, let us schedule the job J^B_l such that $\gamma^B_l(t) = \min_{k \in U^B}\{\gamma^B_k(t)\}$ (where U^B stands for the unscheduled B-jobs). Ties are broken arbitrarily. Let $\tilde{\sigma}$ be the schedule produced in this way.

THEOREM 4.6 *The schedule* $\tilde{\sigma}$ *is nondominated.*

Proof. Suppose a schedule σ' exists which is optimal for $COP(\bar{C}^A|\Gamma^B_{\max} \leq Q)$ and which dominates $\tilde{\sigma}$. The schedule σ' cannot be strictly better than $\tilde{\sigma}$ for both users, since otherwise $\tilde{\sigma}$ would not be optimal. Also, if $\Gamma^B_{\max}(\sigma') = \Gamma^B_{\max}(\tilde{\sigma})$, it should be $\bar{C}^A(\sigma') < \bar{C}^A(\tilde{\sigma})$, which is not possible since $\tilde{\sigma}$ is optimal for $COP(\bar{C}^A|\Gamma^B_{\max} \leq Q)$. The only case left is $\bar{C}^A(\sigma') = \bar{C}^A(\tilde{\sigma})$ and $\Gamma^B_{\max}(\sigma') < \Gamma^B_{\max}(\tilde{\sigma})$. Let $Q_B = \Gamma^B_{\max}(\tilde{\sigma})$. Call $J^B_{\tilde{k}}$ the job in $\tilde{\sigma}$ attaining the cost value Q_B and let $\tilde{t}$ its completion time in $\tilde{\sigma}$, i.e., $\gamma_{\tilde{k}}(\tilde{t}) = Q_B$. Since $\Gamma^B_{\max}(\sigma') < Q_B$, in σ' the completion time of $J^B_{\tilde{k}}$ must be earlier than in $\tilde{\sigma}$. On the other hand, in $\tilde{\sigma}$ the B-jobs that are scheduled after $\tilde{t}$, are scheduled after $\tilde{t}$ also in σ', since otherwise the room left should be filled with A-jobs, and this would lead to a nonoptimal schedule for $COP(\bar{C}^A|\Gamma^B_{\max} \leq Q)$ (recall that the A-jobs are totally ordered). So, from $\tilde{t}$ onwards, σ' and $\tilde{\sigma}$ are identical. Let us therefore consider which job completes at time $\tilde{t}$ in σ'. It cannot be any B-job, since by definition job $J^B_{\tilde{k}}$ is the one attaining the minimum $\gamma^B_k(\tilde{t})$, and it cannot end in $\tilde{t}$. So, it has to be an A-job, say $J^A_{\tilde{h}}$. But if we extract $J^B_{\tilde{k}}$ from its current position in σ' and reinsert it after $J^A_{\tilde{h}}$, the schedule improves for user A, which means that it was not optimal for $COP(\bar{C}^A|\Gamma^B_{\max} \leq Q)$, a contradiction. ∎

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

THEOREM 4.7 *A nondominated optimal solution to* $\mathrm{COP}(\bar{C}^A|\Gamma^B_{\max} \leq Q)$ *can be computed in* $O(n_A \log n_A + n_B^2)$.

5 $\boldsymbol{COP}(n^A_T|\Gamma^B_{\max} \leq Q)$

Let us turn to the problem in which A wants to minimize the number of tardy jobs, given that B only accepts schedules in which $\max_k\{\gamma^B_k(C^B_k)\} \leq Q$. As in the previous sections,

it is possible to define a certain deadline D_k^B for each B-job J_k^B such that $\gamma_k^B(C_k^B) \leq Q$ for $C_k^B \leq D_k^B$ and $\gamma_k^B(C_k^B) > Q$ 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 $C_k^B \leq D_k^B$ 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 $\gamma_k^B(C_k^B) > Q$.

Consider now, for each B-job J_k^B, the latest processing interval $[LS_k, D_k^B]$. Let $I = \cup_{k=1}^{n_B}[LS_k, D_k^B]$. Set I consists of a number $\beta \leq n_B$ of intervals, $I_{1,h_1}, I_{h_1,h_2}, \ldots, I_{h_{\beta-1},n_B}$, call them *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* $I_{u,v}$.

Consider the preemptive variant of problem $COP(n_T^A|\Gamma_{\max}^B \leq Q)$, call it $COP_{prmt}(n_T^A|\Gamma_{\max}^B \leq Q)$, and denote by n_T^* and $n_{T,prmt}^*$ the two optimal values.

LEMMA 5.1 *Given an optimal solution to* $\mathrm{COP}_{prmt}(n_T^A|\Gamma_{\max}^B \leq Q)$*, there exists an optimal solution to* $\mathrm{COP}(n_T^A|\Gamma_{\max}^B \leq Q)$ *with the same number of tardy A-jobs.*

Proof. Just observe that if in the optimal solution to COP_{prmt} there is a job J_i (of any user), 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 other job. Repeating this for each preempted job, we eventually get a nonpreemptive solution. ∎

LEMMA 5.2 *There exists an optimal solution to* $\mathrm{COP}_{prmt}(n_T^A|\Gamma_{\max}^B \leq Q)$ *in which each B-job is nonpreemptively scheduled in the reserved interval it is associated with.*

Proof. In an optimal solution to COP_{prmt}, 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 get a solution in which the completion time of no A-job has increased, since we only moved pieces of A-jobs backwards. ∎

Lemma 5.2 tells us the position of the B-jobs in an optimal solution to $COP_{prmt}(n_T^A|\Gamma_{\max}^B \leq Q)$. The position of the A-jobs can be found by solving an auxiliary instance of the well-known single-user (nonpreemptive) problem $1||n_T$ (solvable by Moore's algorithm). Given an instance of $COP_{prmt}(n_T^A|\Gamma_{\max}^B \leq Q)$, such 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 of 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 $\mathcal{D}_h^A$ as $\mathcal{D}_h^A = d_h^A - \sum_{u,v:D_v^B \leq d_h^A} ||I_{u,v}||$. 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 $\mathcal{D}_h^A = LS_p - \sum_{u,v:D_v^B < d_h^A} ||I_{u,v}||$.

THEOREM 5.3 $\mathrm{COP}(n_T^A|\Gamma_{\max}^B \leq Q)$ *can be solved in* $O(n_A \log n_A + n_B \log n_B)$.

Proof. Given a schedule σ for the auxiliary instance of $1||n_T$, it is possible to define a solution σ' to $COP_{prmt}(n_T^A|\Gamma_{\max}^B \leq Q)$ by re-inserting 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 get an optimal solution to $COP_{prmt}(n_T^A|\Gamma_{\max}^B \leq Q)$. Applying Lemma 5.1, we get an optimal solution to $COP(n_T^A|\Gamma_{\max}^B \leq Q)$, obtained 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 $O(n_A)$ and solved in $O(n_A \log n_A)$ by Moore's algorithm. The optimal solution to $COP_{prmt}(n_T^A|\Gamma_{\max}^B \leq Q)$ can be reconstructed in $O(n_A + n_B)$. Finally, the optimal solution to $COP(n_T^A|\Gamma_{\max}^B \leq Q)$ is obtained in $O(n_A + n_B)$. The overall complexity is therefore dominated by the ordering steps and the theorem follows. ∎

6 $\boldsymbol{COP}(n_T^A|n_T^B \leq Q)$

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

LEMMA 6.1 *There is an optimal schedule σ^* of* COP$(n_T^A|n_T^B \leq Q)$ *in which all the tardy 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. We consider each part in turn. Consider an optimal schedule σ^* and move all the tardy jobs to the end of the schedule, thus obtaining a new schedule σ'. Clearly, $n_T^A(\sigma') \leq n_T^A(\sigma^*)$, 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 jobs in $J^A \cup J^B$ are numbered from J_1 to $J_{n_A+n_B}$ according to the EDD order. We next describe a dynamic programming algorithm for problem $COP(n_T^A|n_T^B \leq Q)$. In the description of Algorithm n_T^A, n_T^B that follows, we let $C(i,h,k)$ be the shortest 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 tardy A-jobs and at most k tardy B-jobs. By definition, we set $C(i,h,k) = +\infty$ if no such schedule exists.

Algorithm n_T^A, n_T^B

Boundary Conditions

$C(0,0,0) = 0$
$C(i,h,k) = +\infty$ if $(i < 0$ *or* $h < 0$ *or* $k < 0)$.

Recurrence Relation

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

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

Optimal Solution Value

$\min\{h : C(n_A+n_B,h,Q) < +\infty\}$

We next prove that Algorithm n_T^A, n_T^B provides an optimal solution to problem $COP(n_T^A|$ $n_T^B \leq Q)$.

LEMMA 6.2 *If the value $C(i,h,k)$ provided by Algorithm n_T^A, n_T^B is finite, it is the smallest completion time among all the feasible schedules of the job set $\{J_1,\ldots,J_i\}$, with at most h tardy A-jobs and k tardy B-jobs. If $C(i,h,k) = +\infty$, then no feasible schedule of the job set $\{J_1,\ldots,J_i\}$ exists with at most h tardy A-jobs and k tardy B-jobs.*

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 tardy A-jobs and k tardy B-jobs. Assume first that J_i is an A-job. If J_i is tardy 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 Algorithm n_T^A, n_T^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 none of the two holds, there can be no feasible schedule of $\{J_1,\ldots,J_i\}$ with at most h tardy A-jobs and k tardy 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 tardy 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 6.3 *Algorithm n_T^A, n_T^B produces an optimal solution to problem* $\mathrm{COP}(n_T^A|n_T^B \leq Q)$ *in $O(n_A^2 n_B + n_A n_B^2)$ time.*

Proof. Let $h^* = \min\{h : C(n,h,Q) < \infty\}$. Suppose that an optimal schedule σ for problem $COP(n_T^A|n_T^B \leq Q)$ exists in which $n_T^A(\sigma) < h^*$. W.l.o.g. we can assume that σ has the structure of Lemma 6.1. Note that, by definition of h^*, $C(n_A+n_B, n_T^A(\sigma), Q) = +\infty$. Let now J_j be last early job in σ. From Lemma 6.2, $C_j(\sigma) \geq C(n_A+n_B, n_T^A(\sigma), Q)$. This implies $C(n_A+n_B, n_T^A(\sigma), Q) < +\infty$, a contradiction. Therefore h^* is the optimal value for $COP(n_T^A|n_T^B \leq Q)$.

Let us 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 of Algorithm n_T^A, n_T^B is therefore dominated by the former quantity. ∎

7 $\boldsymbol{COP}(\overline{wC}^A | n_T^B \leq Q)$

Here we show that the problem in which A wants to minimize the weighted sum of flow times, and B accepts that up to Q of his/her own jobs are tardy is $\mathcal{NP}$-complete. For this, we make use of the following restricted version of PARTITION:

PROBLEM 7.1 (CONSTRAINED-PARTITION). *Given a set of nonnegative rationals* $\{u_1, u_2, \ldots, u_n\}$ *with* $\sum_{i=1}^n u_i = 1$, *and* $\frac{1}{n+2} < u_i < \frac{1}{n-2}$, *decide whether a subset* $S \subseteq \{1, \ldots, n\}$ *exists such that* $\sum_{i \in S} u_i = \frac{1}{2}$.

Observe that such restrictions on the u_i's imply that the cardinality of S must be $\frac{n}{2}$. The next lemma shows that despite this restriction, the problem remains hard.

LEMMA 7.2 *Problem* CONSTRAINED-PARTITION *is* $\mathcal{NP}$*-complete.*

Proof. Membership in $\mathcal{NP}$ is trivial. We first observe that Problem PARTITION remains hard even with the additional constraint that $|S|$ must be equal to $\frac{n}{2}$ (see, e.g. [2]). Let us call EQUI-PARTITION the version of PARTITION with this additional constraint. To show hardness, we reduce EQUI-PARTITION to CONSTRAINED-PARTITION as follows. Consider an instance of EQUI-PARTITION with set of nonnegative integers $\{u_1, u_2, \ldots, u_n\}$. Let $U = \sum_{i=1}^n u_i$ and set $u'_i := u_i + C$ where C is a suitably large constant ensuring that

$$\frac{U + nC}{n+2} \leq u'_i \leq \frac{U + nC}{n-2} \text{ for all } i = 1, \ldots, n$$

(it suffices $C \geq \frac{nU}{2}$.) Now, after normalizing $u'_k := \frac{u'_k}{U+nC}$, we get an instance of Problem CONSTRAINED-PARTITION which is a yes-instance if and only if the original instance of EQUI-PARTITION is a yes instance. Note that, after normalization, the u_k's are rational values. Though C depends on n, we only require $O(\log n)$ memory space to store all the data: therefore we have reduced EQUI-PARTITION to CONSTRAINED-PARTITION in polynomial time and proved that the latter problem is $\mathcal{NP}$-hard. ∎

We are now in the position of reducing Problem CONSTRAINED-PARTITION to Problem $REC(\overline{wC}^A \leq Q_A | n_T^B \leq Q_B)$. We next show that by solving a suitable instance of $REC(\overline{wC}^A \leq Q_A | n_T^B \leq Q_B)$ it is possible to solve any instance of CONSTRAINED-PARTITION.

First, suppose that the job set for user A is composed by n jobs $J^A = \{J_1^A, J_2^A, \ldots, J_n^A\}$, all having processing time equal to 1, and user A's objective function is the minimization of weighted total completion times $\overline{wC}^A = \sum_{h=1}^n w_h C_i^A$, with weights $w_h = 2^{n-h}$, $h = 1, \ldots, n$.

User B must process the jobs in the set $J^B = \{J_1^B, \ldots, J_{2n+1}^B\}$. We distinguish

- n *short* jobs J_k^B, $k = 1, \ldots, n$, with processing times $s_k = 2^{n+k-1} + u_k$, and due date $d_k^B = d_k^s = k - 1 + 2^n(2^k - 1) + \sum_{t=1}^{k-1} u_t$;
- n *long* jobs J_{n+k}^B, $k = 1, \ldots, n$, with processing times $l_k = 2^{n+k-1} + 2u_k$ and due date $d_{n+k}^B = d_k^l = k + 2^n(2^k - 1) + 2\sum_{t=1}^{k} u_t$; and

- a *stopper* job J^B_{2n+1}, with processing time $M \gg 1$ and due date $d^B_{2n+1} = n + 2^n(2^n - 1) + \frac{3}{2} + M$. In particular, M is large enough to ensure that any solution $\hat{\sigma}$ with one job of J^A being scheduled after the stopper job, achieves a value $\overline{wC}^A(\hat{\sigma}) > Q_A$. (For instance, it holds for $M \geq n2^{3n}$.)

Note that (i) $l_k = s_k + u_k$, and (ii) $d^l_k = d^s_k + 1 + 2u_k + \sum_{t=1}^{k-1} u_t$. User B accepts that at most Q_B of his/her jobs are tardy. We also set

$$Q_A := \sum_{h=1}^{n} \left(w_h(h + 2^n(2^h - 1) + \sum_{t=1}^{h} u_t) \right) - \frac{1}{2} - n2^{2n-2} \quad (1)$$

$$Q_B := n. \quad (2)$$

We now need some preliminary lemmas presenting the particular structure of any nondominated solution for $REC(\overline{wC}^A \leq Q_A | n^B_T \leq Q_B)$ and afterwards prove the main result of this section.

REMARK 7.3 *Since $w_{h+1} > w_h$ and $d^s_{k+1} > d^l_k > d^s_k$, we restrict ourselves to consider the jobs of J^A sequenced in the same order of their indices and the jobs of J^B sequenced in non decreasing order of their due dates.*

LEMMA 7.4 *In any nondominated solution for* REC($\overline{wC}^A \leq Q_A | n^B_T \leq Q_B$), *for all pairs of jobs J^B_k and J^B_{n+k}, $i = 1, \ldots, n$, at least one job must be tardy.*

Proof. We only need to verify that $s_k + l_k > d^l_k (> d^s_k)$. This always holds, as long as $n > 2$. ∎

Lemma 7.4 establishes that the stopper job J^B_{2n+1} plus *exactly one* job for each pair J^B_k, J^B_{n+k} must complete on time to achieve feasibility. As noticed above, to achieve feasibility, we must schedule all the n jobs of J^A before the stopper job J^B_{2n+1}: otherwise the value of M is large enough to make $\overline{wC}^A$ larger than Q_A. It is in fact possible to find feasible schedules with that property: for instance, it is easy to verify that there is room enough to schedule the n short jobs of J^B and all the jobs in J^A before the stopper job. We are therefore interested in finding the most convenient (for user A) schedule such that all the jobs of J^A and n jobs of $J^B \setminus \{J^B_{2n+1}\}$ are scheduled before the stopper job J^B_{2n+1}.

LEMMA 7.5 *In any nondominated solution for* REC($\overline{wC}^A \leq Q_A | n^B_T \leq Q_B$), *blocks may assume two alternative forms only:*

$$\mathcal{B}_h = \begin{cases} (J^B_h \prec J^A_h) \\ (J^A_h \prec J^B_{n+h}) \end{cases} \qquad h = 1, \ldots, n$$

Proof. From equation $(iii)'$ in the proof of Lemma 7.5, $\tau_{h-1} + 1 + s_h > d^s_h$ and therefore block $\mathcal{B}_h$ cannot be of the form $(J^A_h \prec J^B_h)$, i.e., h-th job of J^A followed by the h-th short job of J^B.

Moreover, $\mathcal{B}_h = (J^A_h \prec J^B_{n+h})$ is feasible. In fact, we get the latest starting time for block $\mathcal{B}_h$ if we have scheduled, in the previous $h-1$ blocks, the long jobs of J^B $J^B_{n+1}, \ldots, J^B_{n+h-1}$. Till there, the length of such a schedule is therefore:

$$\tau'_{h-1} = (h-1) + \sum_{t=1}^{h-1} l_t = (h-1) + 2^n(2^{h-1} - 1) + 2\sum_{t=1}^{h-1} u_t.$$

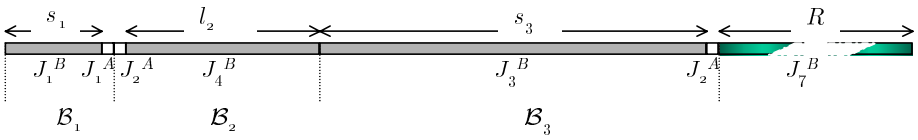

Figure 1: Structure of a nondominated schedule

It is easy to verify that this partial schedule fits the due date, i.e., $\tau'_{h-1} + 1 + l_h = d^l_h$. Therefore all the blocks $\mathcal{B}_h = (J^B_{n+h} \prec J^A_h)$ are obviously dominated by $\mathcal{B}_h = (J^A_h \prec J^B_{n+h})$. □

Figure 1 illustrates the structure for a nondominated schedule in an example where $n = 3$.

In view of the previous Lemmas, all we need, to completely recover all the informations about any nondominated solution, is the form of all its blocks. For this purpose we introduce a set of binary variables y_h, for $h = 1, \ldots, n$. Each variable y_h is 0 if block $\mathcal{B}_h = (J^B_h \prec J^A_h)$, i.e., it contains the short h-th job of J^B, and $y_h = 1$ if block $\mathcal{B}_h = (J^A_h \prec J^B_{n+h})$, i.e., it contains the long h-th job of J^B. We denote by $\bar{\sigma}(y)$, $y \in \{0,1\}^n$, the nondominated schedule:

$$\bar{\sigma}(y) = \left\{ \begin{array}{l} (J^B_1 \prec J^A_1) \Leftrightarrow y_1 = 0 \\ (J^A_1 \prec J^B_{n+1}) \Leftrightarrow y_1 = 1 \end{array} \right\} \prec \ldots \prec \left\{ \begin{array}{l} (J^B_n \prec J^A_n) \Leftrightarrow y_n = 0 \\ (J^A_n \prec J^B_{2n}) \Leftrightarrow y_n = 1 \end{array} \right\} \prec J^B_{2n+1}$$

We call *basic schedule* the schedule having the short jobs of J^B in all blocks and denote it by $\bar{\sigma}(0_n)$. The length of a such a schedule, up to the end of blo ck $\mathcal{B}_h$, is equal to $\tau_h(0_n) = h + \sum_{t=1}^h s_t = h + 2^n(2^h - 1) + \sum_{t=1}^h u_t$. Hence, we can easily compute the value of the weighted completion time corresponding to the basic schedule:

$$\overline{wC}^A(\bar{\sigma}(0_n)) = \sum_{h=1}^n w_h \tau_h(0_n).$$

Similarly, the length of a generic schedule $\bar{\sigma}(y)$, up to the end of block $\mathcal{B}_h$, is $\tau_h(y) = \tau_h(0_n) + \sum_{t=1}^h u_t y_t$, and hence the weighted completion time for such a schedule becomes:

$$\overline{wC}^A(\bar{\sigma}(y)) = \overline{wC}^A(\bar{\sigma}(0_n)) - \sum_{h=1}^n \left(w_h \left(s_h y_h - \sum_{t=1}^{h-1} u_t y_t \right) \right).$$

User A may only accept that $\overline{wC}^A(\bar{\sigma}(y)) \leq Q_A$ and therefore he/she wants to schedule his/her own jobs as soon as possible. On the other hand, if we choose to anticipate job J^A_i in the i-th block, the schedule length increases by u_i. In order to maintain feasibility, the length $\tau_n(y)$ of $\bar{\sigma}(y)$ should allow the stopper job to complete on time, which is equivalent to set $\tau_n(y) + M \leq d^B_{2n+1}$. Recalling the expression of d^B_{2n+1} and noticing that $\tau_n(y) - \tau_n(0_n) = \sum_{i=1}^n u_i y_i$, this is in turn equivalent to:

$$\tau_n(y) + M \leq d^B_{2n+1} \Leftrightarrow \sum_{h=1}^n u_h y_h \leq \frac{1}{2}. \tag{3}$$

If we let $\Delta(y) = \overline{wC}^A(\bar{\sigma}(0_n)) - \overline{wC}^A(\bar{\sigma}(y))$ (i.e., $\Delta(y) \geq 0$ is the *gain* that we get in the objective function when choosing a generic schedule $\bar{\sigma}(y)$ instead of the basic schedule),

and recalling that $w_i = 2^{n-i}$ and $s_i = 2^{n+i-1} + u_i$, we have that:

$$\Delta(y) = 2^{2n-1} \sum_{h=1}^{n} y_h + \sum_{h=1}^{n} u_h y_h. \tag{4}$$

Now we note that

$$Q_A = \overline{wC}^A(\bar{\sigma}(0_n)) - \frac{1}{2} - n2^{2n-2}$$

and therefore $\overline{wC}^A(\bar{\sigma}(y)) \leq Q_A$ is equivalent to

$$\Delta(y) \geq \frac{1}{2} + n2^{2n-2}.$$

From equation 4, we conclude that:

$$\sum_{h=1}^{n} u_h y_h + 2^{2n-1} \sum_{h=1}^{n} y_h \geq \frac{1}{2} + n2^{2n-2}. \tag{5}$$

Recall now the particular bounds on the values of u_h, $h = 1, \ldots, n$, in the instance of CONSTRAINED-PARTITION. The equality $\sum_{h=1}^{n} y_h = \frac{n}{2}$ must hold in order to get $\sum_{h=1}^{n} u_h y_h = \frac{1}{2}$. From equations 3 and 5 we get eventually that $\bar{\sigma}(y)$ is a feasible schedule for problem $REC(\overline{wC}^A \leq Q_A | n_T^B \leq Q_B)$ if and only if

$$\sum_{h=1}^{n} u_h y_h = \frac{1}{2}.$$

We are now in the position to state the following

THEOREM 7.6 *Problem* REC$(\overline{wC}^A \leq Q_A | n_T^B \leq Q_B)$ *is* $\mathcal{NP}$*-complete.*

Proof. If a schedule σ exists such that $\overline{wC}^A(\sigma) \leq Q_A$ and $n_T^B(\sigma) \leq Q_B)$ then there is a nondominated one $\bar{\sigma}(y)$ such that $\sum_{h=1}^{n} u_h y_h = \frac{1}{2}$ and therefore the instance of CONSTRAINED-PARTITION is a yes-instance.

On the other hand, if $REC(\overline{wC}^A \leq Q_A | n_T^B \leq Q_B)$ is not feasible we must conclude that no $y \in \{0,1\}^n$ exists such that $\sum_{h=1}^{n} u_h y_h = \frac{1}{2}$, since otherwise it would be $\overline{wC}^A(\bar{\sigma}) \leq Q_A$ and $n_T^B(\bar{\sigma}) \leq Q_B)$, and therefore the instance of CONSTRAINED-PARTITION is not feasible.

We have therefore reduced Problem CONSTRAINED-PARTITION in polynomial time and proved that tha latter one is NP-complete. ∎

8 $\boldsymbol{COP}(\bar{C}^A | \bar{C}^B \leq Q)$

Here we consider the problem in which both users wish to minimize their own total flow time. We show that even in the unweighted case, the problem is $\mathcal{NP}$–hard.

First of all, the very same argument of Lemma 4.4 shows that, with no loss of generality, we can suppose that both users order their jobs in SPT order. Hence, for simplicity we number the jobs of each user accordingly. We use again the problem PARTITION, more conveniently restated as follows. Given k integers, $p_1, p_2, \ldots, p_k$, let $P = \sum_{i=1}^{k} p_i$. Is there a bipartition $(S, \bar{S})$ 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., $p_1 \leq p_2 \leq \ldots \leq p_k$. First consider its recognition form. The following theorem establishes the complexity of $REC(\bar{C}^A \leq Q_A | \bar{C}^B \leq Q_B)$.

THEOREM 8.1 REC($\bar{C}^A \leq Q_A | \bar{C}^B \leq Q_B$) *is $\mathcal{NP}$-complete.*

Proof. Membership in $\mathcal{NP}$ is trivial. Given an instance of PARTITION, define an instance of $REC(\bar{C}^A \leq Q_A | \bar{C}^B \leq Q_B)$ 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, ..., 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 user who has the precedence in a pair of jobs having the same length. Also, note that $\bar{C}^A(\sigma) + \bar{C}^B(\sigma)$ is the same for all SPT solutions. Simple arithmetics show 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 let us consider the cost of an SPT schedule for each user. 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 flow time of the jobs in $J[1]$ is p_1 for one user and $2p_1$ for the other, the contribution of the jobs in $J[2]$ is $2p_1 + p_2$ for one user and $2p_1 + 2p_2$ for the other, ..., the contribution of the jobs in $J[h]$ is $2p_1 + 2p_2 + \ldots + 2p_{h-1} + p_h$ for one user and $2p_1 + 2p_2 + \ldots + 2p_{h-1} + 2p_h$ for the other. Hence, in a given SPT schedule, the contribution of $J[h]$ to $\bar{C}^A$ can be obtained by adding to $2p_1 + 2p_2 + \ldots + 2p_{h-1} + p_h$ either 0 or p_h, depending on whether A precedes B in $J[h]$ or viceversa, for each $h = 1, \ldots, k$. For user B the opposite holds, i.e., if $A \prec_i B$, then the value p_h is added. Given a regular solution, 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, the total flow time for user A is

$$\begin{aligned} & p_1 + x(0,p_1) + \\ & +2p_1 + p_2 + x(0,p_2) + \\ & +2p_1 + 2p_2 + p_3 + x(0,p_3) + \\ & \ldots \\ & +2p_1 + 2p_2 + \ldots + 2p_{h-1} + p_h + x(0,p_h) + \\ & \ldots \\ & +2p_1 + 2p_2 + \ldots + 2p_{k-1} + p_k + x(0,p_k) = \\ & P + 2(\sum_{i=1}^{k}(k-i)p_i) + x \end{aligned} \tag{6}$$

whereas for B it is

$$P + 2(\sum_{i=1}^{k}(k-i)p_i) + (P - x) \tag{7}$$

We next prove that if a feasible schedule σ for REC exists (i.e., there exists a solution such that the cost of both users does not exceed $T/2$), σ is SPT.

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 get a new schedule σ^* such that $\bar{C}^A(\sigma^*) = \bar{C}^A(\sigma') + p_j$ and $\bar{C}^B(\sigma^*) = \bar{C}^B(\sigma') - p_i$. Since $p_i > p_j$, one has $\bar{C}^A(\sigma') + \bar{C}^B(\sigma') > \bar{C}^A(\sigma^*) + \bar{C}^B(\sigma^*)$,

i.e., the *overall* total flow time $\bar{C}^A + \bar{C}^B$ has decreased of the amount $p_i - p_j$. So, each B-job following a *longer* A-job and the A-job can be swapped, until no such pair of jobs exists. At each swap, the overall total flow time of the solution decreases. A symmetrical discussion could be done for each A-job following a longer B-job. This time, if we swap them, user A gains p_u and user B loses p_v, and so the overall total flow time of the solution decreases by $p_u - p_v$. By repeatedly applying the above swaps, we eventually get an SPT solution. However, since the solution we started with, σ', was feasible, its overall total flow time cannot exceed $3P + 4\sum_{i=1}^{k}(k-i)p_i$. At each swap, the overall total flow 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 flow time for both A and B must be $T/2$. Recalling the expressions (6) and (7) of the flow times for A and B in an SPT solution, we observe that a feasible solution to *REC* may exist if and only if $x = P/2$, i.e., if and only if there is a solution to the instance of PARTITION. ∎

Although $COP(\bar{C}^A|\bar{C}^B \leq Q)$ is shown to be NP-complete, a pseudopolynomial algorithm can be given for this problem. We next illustrate a dynamic programming approach, which use 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 $COP(\bar{C}^A|\bar{C}^B \leq q)$ 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 flow time of J_j^B is $P(i,j)$. Therefore using the following dynamic programming formula

$$F(i,j,q) = \min\{F(i-1,j,q) + P(i-1,j) + p_i^A; F(i,j-1,q-P(i,j))\} \qquad (8)$$

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

THEOREM 8.2 $\mathrm{COP}(\bar{C}^A|\bar{C}^B \leq Q)$ *can be solved in* $O(n_A n_B Q)$ *time.*

Formula (8) must be suitably initialized, by setting $F(0,0,q) = 0$ for all $q = 1, \ldots, Q$ and $F(i,j,q) = +\infty$ for $q < 0$.

9 Shop Problems

In this section we address the case in which two users (A and B) share a common processing resource which is a complex shop system.

Here we use a slightly different problem notation. Following the standard classification scheme for scheduling problems $\psi_1|\psi_2|\psi_3$ (Graham *et al.* [8]), we next address the cases in which $\psi_1 = F2$ (two-machine flow shop), $\psi_1 = O2$ (two-machine open shop) and $\psi_1 = Jm$ (m-machine job shop). Also, we denote by $\psi_2 = \{1,1\}$ the case in which each user has one job to do, where each job consists of processing steps that require specific machine.

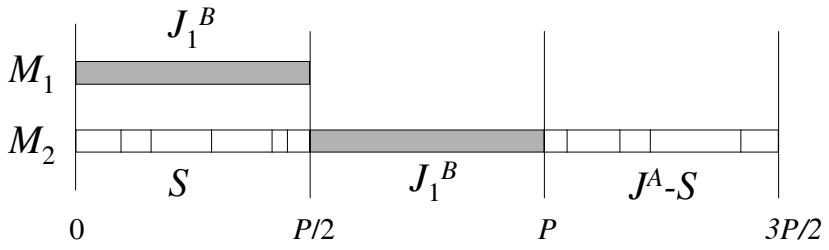

Figure 2: Reduction of PARTITION to $REC(F2||C^A_{\max} \leq Q_A, C^B_{\max} \leq Q_B)$.

9.1 Two-job Job Shop - $\boldsymbol{COP}(Jm|\{1,1\}, f^B \leq Q|f^A)$

This situation arises when each of the two users has one job, requiring to be performed on a sequence of m machines. The performance index of each user only depends on the completion time of the respective job. This situation has been studied in detail by Agnetis et al. [1], for general non-regular, quasi-convex objective functions (note that this includes regular functions as a special case). In particular, consider a nondominated schedule, and the corresponding completion times (C^A, C^B). It can be shown that the set of all pairs of completion times corresponding to nondominated schedules consists of a polynomial number of isolated points and line segments on the (C^A, C^B) plane. This set can be searched in time $O(n_A n_B \log n_A n_B + \log P)$, where P denotes the sum of all the processing times of the two jobs, which therefore is the complexity of $COP(Jm|\{1,1\}, f^B \leq Q|f^A)$.

9.2 Two-machines Flow Shop - $\boldsymbol{COP}(F2|C^B_{\max} \leq Q|C^A_{\max})$

Consider the flow shop with two machines and first limit ourselves to the simplest case, where both users wish to complete their jobs as soon as possible. We show that even this problem is NP-hard. It is easy to show that any other combination of objecive functions (among those considered in this paper) is also NP-hard.

Let p^X_{ih} be the processing time of job J^X_i on machine h ($X = A, B$, $h = 1, 2$). Recall that $REC(F2||C^A_{\max} \leq Q_A, C^B_{\max} \leq Q_B)$ denotes the problem in the recognition form.

THEOREM 9.1 *Problem* REC$(F2||C^A_{\max} \leq Q_A, C^B_{\max} \leq Q_B)$ *is* $\mathcal{NP}$*-complete.*

Proof. We reduce PARTITION to this problem. Consider an instance of $REC(F2||C^A_{\max} \leq Q_A, C^B_{\max} \leq Q_B)$, where user A has $n_A = k$ jobs, and user B has only one job ($n_B = 1$). All the A-jobs have zero processing time on the first machine, while on the second machine the length of the jobs is equal to the integers of PARTITION ($p^A_{i1} = 0$, $p^A_{i2} = p_i$, $i = 1, \ldots, k$). The B-job has processing time $P/2$ on both machines, i.e, $p^B_{11} = p^B_{12} = P/2$. Finally, choose $Q_A = 3P/2$, $Q_B = P$. The constraint $C^B_{\max} \leq Q_B = P$ can be satisfied only if the B-job starts processing on the first machine at time 0 and on the second machine at time $P/2$, thus completing at time P. Hence, in a feasible solution, the total processing time of the A-jobs *preceding* J^B_1 on the second machine (call S this subset of J^A) must be smaller or equal to $P/2$. On the other hand, the constraint $C^A_{\max} \leq Q_A = 3P/2$ can be satisfied only if the total processing time of the A-jobs *following* J^B_1 on the second machine is smaller or equal to $P/2$ (see Figure 2). Hence, the total length of both the

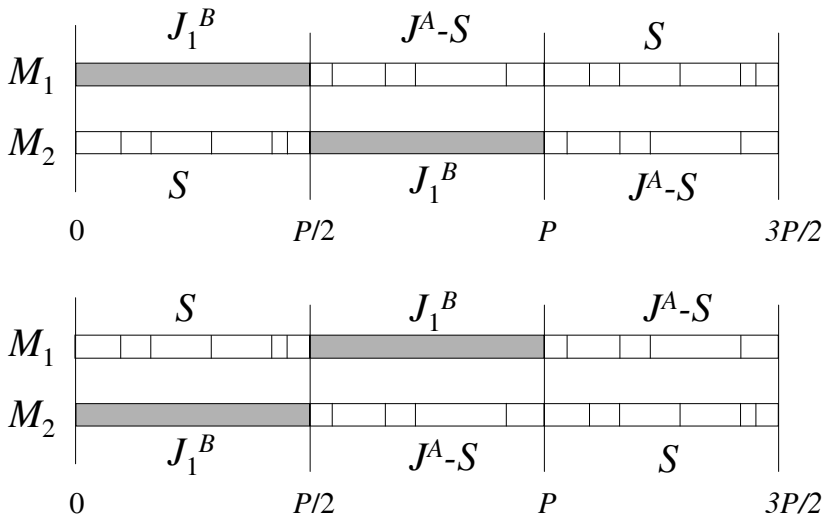

Figure 3: Reduction of PARTITION to $REC(O2||C^A_{\max} \leq Q_A, C^B_{\max} \leq Q_B)$.

A-jobs preceding and following the B-job must equal $P/2$; that is, a feasible schedule exists if and only the corresponding instance of PARTITION is a yes instance. ∎

9.3 Two-machines Open Shop - $\boldsymbol{COP}(O2|C^B_{\max} \leq Q|C^A_{\max})$

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

THEOREM 9.2 *Problem* REC$(O2||C^A_{\max} \leq Q_A, C^B_{\max} \leq Q_B)$ *is $\mathcal{NP}$-complete.*

Proof. We reduce PARTITION to this problem. Given an instance of PARTITION, consider an instance of $REC(O2||C^A_{\max} \leq Q_A, C^B_{\max} \leq Q_B)$, where user A has $n_A = k$ jobs, and user 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 viceversa (see Figure 3). The constraint $C^B_{\max} \leq Q_B = P$ 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, since no A-job completes after $Q_A = 3P/2$, and since $\sum_{i=1}^{n_A} p^A_i + \sum_{i=1}^{n_B} p^B_i = 2P$, 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 the corresponding instance of PARTITION is a yes instance. ∎

10 Pareto-Optimization Problems

In this section we consider the situation in which the two users 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

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

Figure 4: Scheme for enumerating nondominated schedules.

(y_i^A, y_i^B), we associate a nondominated schedule σ^i such that $f^A(\sigma^i) = y_i^A$, $f^B(\sigma^i) = y_i^B$. Let $\mathcal{S} = \{\sigma^1, \ldots, \sigma^k\}$. Problem $POP\{f^A, f^B\}$ consists in determining $\mathcal{S}$.

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

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

10.1 $\boldsymbol{POP}\{\Gamma^A_{max}, \Gamma^B_{max}\}$

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

LEMMA 10.1 *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_h^A and a B-job J_k^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 $\tilde{\sigma}$.*

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_k^B precedes J_h^A in σ' and J_k^B follows J_h^A 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, we show that there exist two nondominated schedules with J_k^B preceding J_h^A in both. In the second subcase, we show that there exist two nondominated schedules with J_h^A preceding J_k^B in both.

(*i*) $C_k^B(\sigma'') \leq C_h^A(\sigma')$. Call α the sequence of jobs between J_h^A and J_k^B in σ''. We let $\sigma := \sigma'$ and let $\tilde{\sigma}$ be the schedule obtained from σ'' by moving J_h^A after job J_k^B, i.e., replacing the sequence $\ldots J_h^A, \alpha, J_k^B \ldots$ with $\ldots \alpha, J_k^B, J_h^A \ldots$. Note that in $\tilde{\sigma}$, job J_h^A completes at time $C_k^B(\sigma'')$, and therefore its associated cost $\gamma_h^A(C_k^B(\sigma''))$ is not greater than $\gamma_h^A(C_h^A(\sigma')) \leq y^A < \tilde{y}^A$. On the other hand, 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 $\gamma_k^B(C_h^A(\sigma'))$ is not greater than $\gamma_k^B(C_k^B(\sigma'')) \leq \tilde{y}^B < y^B$. On the other hand, 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 10.2 *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 10.1, we can enforce the condition of Lemma 10.1 for J_h^A, J_k^B, and thus eliminate such 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 10.1, we eventually get two nondominated schedules σ and $\tilde{\sigma}$ having no contrary job pairs. □

THEOREM 10.3 *There are at most $n_A n_B$ nondominated schedules in* POP$\{\bar{\Gamma}_{max}^A, \Gamma_{max}^B\}$.

Proof. Starting from $Q = +\infty$, the scheme *POP* 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 $C_{k^*}^B(\tilde{\sigma}) < C_{k^*}^B(\sigma)$, since $\tilde{y}^B < y^B$. 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 10.2 guarantees that these two jobs will not reverse their relative ordering

when Q is further decreased. Hence, throughout the execution of the scheme POP, 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. ∎

Note that the scheme POP in Figure 4 must be applied with a sufficiently small $\epsilon > 0$ (depending on the actual shape of the γ functions).

10.2 $POP\{\bar{C}^A, \Gamma^B_{max}\}$

Now we address only the situation in which one of the two users has $\bar{C}$ as its objective function. The more general case of $\overline{wC}$ is open.

Without loss of generality, let A be the user with $\bar{C}$ objective. From Lemma 4.4 we know that in any nondominated schedule, the jobs of J^A are SPT-ordered. As Q decreases, the optimal schedule for $COP(\bar{C}^A|\Gamma^B_{max} \leq Q)$ changes. The next lemma shows that when the constraint on the objective function of user B becomes tighter, the completion time of no A-job can decrease.

LEMMA 10.4 *Let σ be an optimal schedule for* $\mathrm{COP}(\bar{C}^A|\Gamma^B_{max} \leq Q)$*, and consider job $j \in J^A$. Let σ' is an optimal schedule for* $\mathrm{COP}(\bar{C}^A|\Gamma^B_{max} \leq Q')$*, with $Q' < Q$. Then $C_j(\sigma') \geq C_j(\sigma)$.*

Proof. Suppose that the opposite holds, that is, $C_j(\sigma') < C_j(\sigma)$. Since 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 $C_u(\sigma') \geq C_j(\sigma)$. Since $\gamma_u(C_u(\sigma')) \leq Q'$, and $Q' < Q$, then also $\gamma_u(C_j(\sigma)) < Q$. The latter inequality implies that if, in σ, we move u from its current position to the position immediately after j, we get a new schedule which is certainly feasible for $COP(\bar{C}^A|\Gamma^B_{max} \leq Q)$, and is strictly better than σ, since 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 10.5 *Let σ be an optimal schedule for* $\mathrm{COP}(\bar{C}^A|\Gamma^B_{max} \leq Q)$*, and suppose that $k \in J^B$ precedes $j \in J^A$ in σ. Then if σ' is an optimal schedule for* $\mathrm{COP}(\bar{C}^A|\Gamma^B_{max} \leq Q')$*, with $Q' < Q$, k precedes j in σ'.*

Proof. Suppose by contradiction that $k \in J^B$ follows $j \in J^A$ in σ'. From Lemma 10.4, $C_j(\sigma') \geq C_j(\sigma)$. 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 feasible for $COP(\bar{C}^A|\Gamma^B_{max} \leq Q)$. Since A-jobs between k and j in σ are done earlier in σ'', $\bar{C}^A(\sigma'') < \bar{C}^A(\sigma)$. This is a contradiction. ∎

Lemma 10.5 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 10.6 *There are at most $n_A n_B$ nondominated schedules in* $\mathrm{POP}\{\bar{C}^A, \Gamma^B_{max}\}$.

Proof. Starting from $Q = +\infty$, the scheme *POP* 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. Since 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 $\bar{C}$ would be the same in both schedules). Lemma 10.5 guarantees that these two jobs will not reverse their relative ordering when Q is further decreased. Hence, throughout the execution of the scheme *POP*, 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 $0 < \epsilon \leq 1$.

10.3 *POP*$\{\Gamma^A_{max}, n^B_T\}$, *POP*$\{\overline{wC}^A, n^B_T\}$ and *POP*$\{n^A_T, n^B_T\}$

When at least one of the users has n_T as objective function, the number of nondominated schedules is obviously linear. The scheme *POP* in Figure 4 can be applied with $0 < \epsilon \leq 1$.

10.4 *POP*$\{\bar{C}^A, \bar{C}^B\}$

We have seen in Section 8 that finding one nondominated solution is $\mathcal{NP}$-hard (in the ordinary sense). We next show that the number of nondominated solutions may be exponential with respect to the instance size.

EXAMPLE 10.7 *Consider an instance in which the sets of jobs that the two users have to execute are identical. Each set consists of k jobs of size $p_0 = 1, p_1 = 2, p_2 = 4, p_3 = 8, \ldots, p_{k-1} = 2^{k-1}$. Consider now 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 pair indices in which the A-job precedes the B-job having the same length, and $\bar{J}^B$ the set of pair indices in which the opposite holds. Consider job J^A_h. If it is scheduled before J^B_h, its contibution to the cost function is $2^h + 2(2^h - 1)$, otherwise it is $2(2^h) + 2(2^h - 1)$. Hence the total flow time for user A is given by*

$$\bar{F}(A) = \sum_{h \in \bar{J}^A} (2^{h+1} + 2^h - 2) + \sum_{h \in \bar{J}^B} (2^{h+2} - 2) \tag{9}$$

$$= \sum_{h \in \bar{J}^A} (3(2^h) - 2) + \sum_{h \in \bar{J}^B} (4(2^h) - 2) \tag{10}$$

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

Note that only the last term in expression (11) depends on the actual schedule σ. This expression shows that the value of $\bar{F}(A)$ may attain 2^k different values, one for each possible set $\bar{J}^B$. Symmetrically, the same analysis for B yields

$$\bar{F}(B) = \sum_{h=0}^{k-1} 3(2^h) - 2k + \sum_{h \in \bar{J}^A} 2^h. \tag{12}$$

Since 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 get a nondominated solution.*

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

11 Conclusions

In this paper we have introduced a new class of scheduling problems, in which two users compete for the usage of a shared processing resource, and each user has an objective funciton to minimize. We have analyzed the complexity of several problems. Several different directions for future research may be foreseen, such as the generalization of the problems to more than two users, the design of effective enumeration and approximation algorithms for computationally hard cases, modeling of different resource usage modes (concurrent usage, preemption etc.).

References

[1] Agnetis,A., Mirchandani, P.B., Pacciarelli, D., Pacifici, A., 2000, Nondominated schedules for a Job-Shop with Two Competing Users, *Computational and Mathematical Organization Theory*, 6(2), 191–217.

[2] Garey, M.R., and D.S. Johnson 1979. *Computers and Intractability*, Freeman, 1979.

[3] Baker, K. R. and Scudder, G. D., Sequencing with Earliness and Lateness Penalties: A Review. *Operations research*, 1990, 38, 22–36.

[4] Brewer, P. J. and Plott, R .P., A Binary Conflict Ascending Price Mechanism for Decentralized Allocation of the Right to Use Railroad Tracks, *Working Paper, Division of Humanities and Social Sciences, California Institute of Technology*, 1994.

[5] Y. Chen, Y. Peng, T. Finin, Y. Labrou, S. Cost, B. Chu, J. Yao', R. Sun, and B. Wilhelm. 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"*, pp. 15–20, Seattle, WA, 1999.

[6] Hall, N. G. and Posner, M. E., Earliness-Tardiness Scheduling Problems. I: Weighted Deviation of Completion Times about a Common Due Date, *Operations Research*, 1991, 5, 836-846.

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

[8] Graham, R.L., E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan. Optimization and approximation in deterministic machine scheduling: a survey. *Annals of Discrete Mathematics*, 1979, **5**, 287–326.

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

[10] Lawler, E. 1973. Optimal sequencing of a single machine subject to precedence constraints, *Management Science*, 19, 544–546.

[11] Ling Shen, Logistics with two competing users, Ph.D. Thesis, Faculty of Systems and Industrial Engineering, The University of Arizona, Tucson, AZ, 1998.

[12] Raghavachari,M. , Scheduling Problems with Non-regular Penalty Functions - A Review, *Opsearch*, 1988, 25, 145-164.