J Sched (2011) 14:471–481
DOI 10.1007/s10951-010-0201-1

Scheduling two agents on uniform parallel machines with makespan and cost functions

Donatas Elvikis · Horst W. Hamacher · Vincent T'kindt

Published online: 6 November 2010

Abstract We consider the problem of scheduling two jobs A and B on a set of m uniform parallel machines. Each job is assumed to be independent from the other: job A and job B are made up of n_A and n_B operations, respectively. Each operation is defined by its processing time and possibly additional data such as a due date, a weight, etc., and must be processed on a single machine. All machines are uniform, i.e. each machine has its own processing speed. Notice that we consider the special case of equal-size operations, i.e. all operations have the same processing time. The scheduling of operations of job A must be achieved to minimize a general cost function F^A, whereas it is the makespan that must be minimized when scheduling the operations of job B. These kind of problems are called multiple agent scheduling problems. As we are dealing with two conflicting criteria, we focus on the calculation of strict Pareto optima for F^A and $C^B_{\max}$ criteria. In this paper we consider different min-max and min-sum versions of function F^A and provide special properties as well as polynomial time algorithms.

Keywords Parallel machines · Muti-agent problems · Bicriteria scheduling · Pareto optimum

D. Elvikis (✉) · H.W. Hamacher
University of Kaiserslautern, Kaiserslautern, Germany
e-mail: elvikis@mathematik.uni-kl.de

H.W. Hamacher
e-mail: hamacher@mathematik.uni-kl.de

V. T'kindt
LI, Université François Rabelais de Tours, Tours, France
e-mail: vincent.tkindt@univ-tours.fr

1 Introduction

In this paper we consider m uniform parallel machines whose processing speeds are denoted by $v_1, \ldots, v_m$. Two jobs A and B, each consisting of n_A and n_B operations, respectively, have to be scheduled. We denote by $\mathcal{I} = A \cup B$ the set of all operations and $n = |\mathcal{I}| = n_A + n_B$ its cardinality. All operations i in $\mathcal{I}$ have an equal processing time p. However, since machines work with different speeds the actual processing time of an operation scheduled on machine j is equal to $p_j = \frac{p}{v_j}$ for all $j = 1, \ldots, m$.

With job A we associate a general cost function F^A, which is assumed to be minimized. More precisely, we consider in this paper two cases for F^A: either $F^A_{\sum} := \sum_{i \in A} f_i(C_i)$ or $F^A_{\max} := \max_{i \in A} f_i(C_i)$, with C_i the completion time of operation i and f_i a non-decreasing function. With job B we associate criterion $C^B_{\max} := \max_{i \in B}(C_i)$, which is to be minimized. Therefore we are faced with a two-agent scheduling problem, i.e. the scheduling of each job is done according to its own cost function while sharing common resources with the other job. We assume that jobs are selfish and criteria are conflicting, which leads to a bicriteria scheduling problem. Note that if criteria are not conflicting we have a trivial case where minimizing one objective also minimizes the other and therefore there is no need to consider the problem from a multi-objective point of view. From now on we refer to our problem as $Q|p_i = p|\#(F^A, C^B_{\max})$ according to the notation introduced by Graham et al. (1979) and extended by T'kindt and Billaut (2006). In the remainder we make extensive use of this notation to refer to scheduling problems.

As usual, when dealing with multiple conflicting criteria, we focus on the calculation of strict Pareto optima for F^A and $C^B_{\max}$ criteria. A schedule σ is a *strict Pareto optimum* if and only if there does not exist another schedule σ' such that

$F^A(\sigma') \leq F^A(\sigma)$ and $C^B_{\max}(\sigma') \leq C^B_{\max}(\sigma)$ with at least one strict inequality. Such optima have the property of being also weak Pareto optima: a schedule σ is a *weak Pareto optimum* if and only if there does not exist another schedule σ' such that $F^A(\sigma') < F^A(\sigma)$ and $C^B_{\max}(\sigma') < C^B_{\max}(\sigma)$. Notice that a weak Pareto optimum may not be a strict one. This distinction is important regarding the algorithms to be developed in this paper. With each strict Pareto optimum a strictly non-dominated criteria vector $(F^A, C^B_{\max})$ is associated. Henceforth we say that σ' is a *succeeding strictly non-dominated* solution of σ if and only if the inequalities $F^A(\sigma') < F^A(\sigma)$ and $C^B_{\max}(\sigma') > C^B_{\max}(\sigma)$ hold and there does not exist another solution σ'' with $F^A(\sigma') < F^A(\sigma'') < F^A(\sigma)$ and $C^B_{\max}(\sigma') > C^B_{\max}(\sigma'') > C^B_{\max}(\sigma)$. In other words, σ and σ' are neighbors on the Pareto front.

There are two most widely used methods to find a single Pareto optimal solution: one is a convex combination of the criteria functions, also called a weighted sum, and further in this paper denoted by $F_\ell(F^A, F^B) = F^A + \ell F^B$, with $\ell > 0$; the other one is the ε-constraint method denoted by $\varepsilon(F^A/F^B)$ where we minimize criteria F^A subject to $F^B \leq \varepsilon$ and $\varepsilon > 0$ is some a priori fixed value. In this paper we will develop efficient algorithms based on an indirect ε-constraint approach, however, first we give some literature overview related to the problem we tackle.

Scheduling of independent job sets with common resources is a quite recent research area. First, Agnetis et al. (2000) introduced a jobshop scheduling problem with two competing players, each one having their own optimization goal: this was the starting point dealing with multi-agent scheduling problems. For the tackled jobshop problem they focused on the enumeration of Pareto optima for two general quasiconvex functions of the operation completion times. Moreover, the authors also presented some real world applications like runway scheduling which can be used by multiple airline companies to land their aircrafts and where costs are issued for the delays. Another example is telecommunications with different service types, e.g. voice and file transfer, that share the same connection line and there are strict non-delay requirements for voice data packets, thus probably some will be dropped due to delay, whereas no packets can be lost during file transfer even if delivered with some delay.

The problem considered in this paper can be applied to a manufacturing environment. Consider a production company at a given time point having a number of orders that build a set of n_A operations which all belong to job A. The costs of job A are calculated *w.r.t.* function F^A. Then another customer comes in with a new job B requiring that this new job, consisting of n_B operations, must be processed as soon as possible, i.e. makespan $C^B_{\max}$ has to be minimized. The producer has only a limited number m of resources that have to be shared between the existing job A and the new job B operations, thus the manufacturer is in the position when a decision has to be made with respect to the changed situation in which both criteria functions have to be minimized.

Baker and Smith (2003) studied several single machine problems with two independent job sets where scheduling is done by minimizing a convex combination of two criteria among $C_{\max}$, $L_{\max}$ and $\sum w_i C_i$. They provided polynomial time algorithms to calculate the optimal solution (a Pareto optimum) for problems $1|d_i|F_\ell(C^A_{\max}, L^B_{\max})$ and $1|d_i|F_\ell(C^A_{\max}, \sum w_i C^B_i)$. They also proved that problem $1|d_i|F_\ell(L^A_{\max}, \sum w_i C^B_i)$ is $\mathcal{NP}$-hard except when $w_i = 1, \forall i = 1, \ldots, n$. Later, Yuan et al. (2005) proposed an update for some of Baker and Smith's results, notably for the $1|d_i|F_\ell(L^A_{\max}, L^B_{\max})$ and $1|d_i|F_\ell(\sum C^A_i, L^B_{\max})$ problems for which they proposed a dynamic program requiring $\mathcal{O}(n_A n_B(n_A + n_B))$ time.

Agnetis et al. (2004) tackled several single machine scheduling problems including the $1 \parallel \varepsilon(\sum w_i C^A_i / F^B_{\max})$ and $1 \parallel \varepsilon(F^A_{\max}/F^B_{\max})$ problem. They showed that the former is $\mathcal{NP}$-hard (by reduction from the knapsack problem) except when $w_i = 1, \forall i = 1, \ldots, n_A$, and that the latter can be solved in $\mathcal{O}(n_A^2 + n_B \log n_B)$ time. They also focused on the cardinality of the set of strictly non-dominated criteria vectors for the $1 \parallel \#(F^A_{\max}, F^B_{\max})$ problem and showed that this set contains at most $n_A n_B + 1$ vectors. Agnetis et al. (2004) also considered two two-machine shop problems which were shown to be $\mathcal{NP}$-hard. The first problem with more than two agents was dealt with by Cheng et al. (2006) who considered the $1|d_i, \sum w_i U^{(\ell)}_i \leq \varepsilon^{(\ell)}|$— problem, i.e. for each independent job set we impose a bound constraint on its weighted number of tardy operations. They showed that this problem is strongly $\mathcal{NP}$-hard.

Later Ng et al. (2006) proved that $1 \parallel \varepsilon(\sum w_i C_i / L_{\max})$ and $1 \parallel \varepsilon(\sum w_i C_i / \sum U_i)$ problems are strongly $\mathcal{NP}$-hard and that the $1 \parallel \varepsilon(\sum C_i / \sum U_i)$ problem is $\mathcal{NP}$-hard under the id-encoding scheme. Only recently Leung et al. (2010) proved that the latter problem, which was left open by Agnetis et al. (2004), is binary $\mathcal{NP}$-hard. In the same paper authors considered different problem settings, e.g., with preemption, release dates on both or one of the job sets, and established relationships between two-agent scheduling and scheduling subject to availability constraints.

Balasubramanian et al. (2009) were the first researchers to deal with parallel machines environment, namely identical parallel machines. They developed an iterative heuristic as well as a bicriteria genetic algorithm to solve the $P \parallel \#(\sum C_i{}^A, C^B_{\max})$ problem and concluded that the proposed heuristics provide a high quality solution faster by comparison with a time-indexed integer program for small and large instances. Later Leung et al. (2010) considered several identical parallel machine problems where jobs can be preempted and may have release dates. They showed that the problem $P_m|r_i, pmtn|\varepsilon(F^A_{\max}/F^B_{\max})$ can be solved

in pseudo-polynomial time and problem $P_2|pmtn|\varepsilon(\sum C_i^A / F_{\max}^B)$ in polynomial time. Moreover, they proved that problems with release dates and one of the criteria $\sum C_i^A$ or $\sum U_i^A$ in combination with maximum criteria are binary $\mathcal{NP}$-hard.

Recently Huynh Tuong and Soukhal (2009) studied a two-agent scheduling problem on a three-machine flow shop. They considered a case where the first set of jobs has to be processed on the first two machines and the second set of jobs has to be processed on the last two machines, thus the second machine is shared between the jobs of two agents. Authors considered that the makespan has to be minimized for both agents and applied an ε-constraint approach to solve this $\mathcal{NP}$-hard problem in pseudo-polynomial time.

A new approach to multi-agent scheduling was introduced by Huynh Tuong et al. (2009). Authors considered bicriteria single and identical parallel machines scheduling problems with one objective associated to the whole set of jobs $\mathcal{I}$ and another associated to a subset $\mathcal{I}_1 \subset \mathcal{I}$. They called this new class of problems "interfering job scheduling" and showed the differences existing between these and multi-agent scheduling problems.

One other feature of the problems tackled in this paper is that all operations are of equal size. This case has been the matter of numerous studies during the last decade, but none on multi-agent scheduling problems. Regarding single machine problems, Baptiste studied the $1|r_i, p_i = p, |\sum w_i U_i$ problem (Baptiste 1999) and the $1|r_i, p_i = p, |\sum T_i$ problem (Baptiste 2000) for which he proposed polynomial time algorithms. Later on, Chrobak et al. (2006) considered the $1|r_i, p_i = p, |\sum U_i$ problem for which they proposed a modification of Baptiste's algorithm with improved time complexity. Other single machine problems were dealt with by Baptiste and Brucker (2004).

Parallel machine problems have also been tackled, mainly when machines are identical. Baptiste (2000) focused on the two $Pm|r_i, p_i = p|\sum w_i C_i$ and $Pm|r_i, p_i = p|\sum T_i$ problems and showed that these can be solved in polynomial time. Later on Baptiste (2003), proved that the case in which each job requires a given number of machines, referred to as $Pm|r_i, p_i = p, \text{size}_i|\sum C_i$, is still polynomially solvable. Brucker and Kravchenko (2008) focused on some problems including the $P|r_i, p_i = p|\sum w_i C_i$ problem that was shown to be solvable in polynomial time. Other problems, notably some where preemption is allowed, can be found in the literature (Baptiste and Brucker 2004; Baptiste et al. 2004; Brucker and Kravchenko 2008). We will now review several publications which are more related to the problems tackled in this paper.

Dessouky et al. (1990) studied a set of simple but important scheduling problems on uniform parallel machines with unit processing time. They showed that the $Q|p_i = 1|F_{\sum}$ and $Q|p_i = 1|F_{\max}$ problems can be solved in $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ time, respectively. The first one was solved by means of an assignment problem, while the second one is solved by a simple constructive algorithm. They also provided specializations to particular classic max and sum scheduling criteria.

Another important contribution is due to Tuzikov et al. (1998) who developed two polynomial time algorithms for the $Q|p_i = p|\varepsilon(F_{\max}^A / F_{\max}^B)$ and $Q|p_i = p|\varepsilon(F_{\sum}^A / F_{\max}^B)$ problems which are then iteratively solved to calculate the strict Pareto set. Even though these problems may seem similar to the ones tackled in this paper, they are thoroughly different since we consider the scheduling of jobs owned by two independent agents. Moreover, Tuzikov et al. (1998) showed that the size of the set of strictly non-dominated criteria vectors is bounded by at most n^2 solutions for both problems, while we refine this bound for our specific problem and show that there are no more than $n_A + 1$ strictly non-dominated points in the criteria space.

The paper is organized as follows. We first focus in Sect. 2 on general results on equal-size operations and minimization of criterion $C_{\max}^B$. In Sect. 3 we consider the particular case of $F_{\sum}^A$ and $C_{\max}^B$ criteria and provide a polynomial time algorithm for enumerating the set of strictly non-dominated criteria vectors. In Sect. 4 we do the same for $F_{\max}^A$ and $C_{\max}^B$ criteria.

2 General properties of the $Q|p_i = p|\#(F^A, C_{\max}^B)$ problem

In this section we consider a rather general objective function F^A and develop some properties of the $Q|p_i = p|\#(F^A, C_{\max}^B)$ problem. First, observe that we only need to restrict the search for Pareto optima to the set of active schedules if F^A is regular, i.e. a non-decreasing function of the completion times (T'kindt and Billaut 2006). Consequently, due to the fact that operations are of equal size, the completion time of the operation scheduled in k^{th} position on a given machine j is equal to $t_{j,k} = kp_j = k\frac{p}{v_j}$. Let us define the sequence of completion times for machine j by $\mathcal{T}_j = \{t_{j,k} \mid k \in \mathbb{N}\}$. Assume that all timeslots in $\bigcup_{j=1,\ldots,m} \mathcal{T}_j$ are numbered in increasing order of their completion times and, in case of ties, slots on a faster machine get lower indexes, i.e. we build a sequence $\mathcal{T} = \{t_1 \le t_2 \le \cdots \le t_n\} \subseteq \bigcup_{j=1,\ldots,m} \mathcal{T}_j$ of the n smallest timeslots (see Fig. 1). Therefore $\mathcal{T}$ is the sequence of all possible completion times for any active schedule σ such that minimizing criteria F^A and $C_{\max}^B$ is equivalent to assigning operations to completion times. In other words, the uniform parallel machines problem can be seen as a particular instance of a single machine problem for which the sequence of completion times is fixed and we only need to

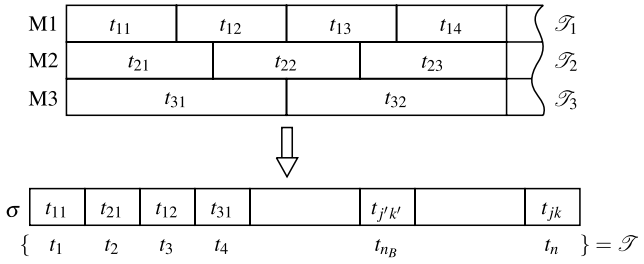

Fig. 1 Machine timeslots transformation to a sequence $\mathcal{T}$

assign jobs to the timeslots. This makes the problem more particular than those tackled by Agnetis et al. (2004).

We now turn to the situation in which F^A is minimized under the constraint that $C^B_{\max} \leq \varepsilon$, where $\varepsilon > 0$ is given. This ε-constraint problem, referred to as $Q|p_i = p|\varepsilon(F^A/C^B_{\max})$, will be solved later on to compute a Pareto optimum. Notice that theoretically an optimal solution to this problem is a weak Pareto optimum. However, it is well known that to each strict Pareto optimum we can assign such ε value that it is an optimal solution to the associated ε-constraint problem (Ehrgott 2005). This implies that when solving such a problem we must take care of discarding all weak but not strict Pareto optima. Assume that the last operation of job B completes at timeslot t, then it is obvious that the assignment of the other operations of B to timeslots $t_j < t$ does not matter. We now provide a similar result to the Property 1 of Baker and Smith (2003) for the single machine case.

Lemma 1 *There exists an optimal schedule σ to the $Q|p_i = p|\varepsilon(F^A/C^B_{\max})$ problem such that all operations of job B are sequenced in a single block completing at time $C^B_{\max} \leq \varepsilon$, i.e. no operation of job A is scheduled between two operations of job B.*

Proof Assume that σ does not satisfy this property, i.e. there exists a timeslot k such that an operation i of job A completes at time $t_k \leq C^B_{\max}$ and some operation j of job B completes at time t_{k-1}. By swapping i and j we obtain a schedule with $C^B_{\max}$ unchanged and since F^A is a regular criterion, its value is not increased. The iterative application of such swaps on an optimal schedule builds another optimal schedule for which all operations of B form a block. □

Let us denote by n_ε the number of timeslots which complete not later than ε, i.e.

$$n_\varepsilon = |\{t_k \in \mathcal{T} \mid t_k \leq \varepsilon\}|. \tag{1}$$

Since F^A is regular, minimizing F^A is, due to Lemma 1, equivalent to finding an optimal assignment of job A operations to the $t_1, \ldots, t_{n_\varepsilon - n_B}, t_{n_\varepsilon+1}, \ldots, t_n$ timeslots. This can be done in $\mathcal{O}(n_A^3)$ or $\mathcal{O}(n_A^2)$ time with sum and maximum criteria, respectively, by applying the algorithms proposed by Dessouky et al. (1990). Consequently, we have a result on the cardinality of the set of strict Pareto optima.

Lemma 2 *There are at most $n_A + 1$ strictly non-dominated criteria vectors* (*strict Pareto optima in criteria space*).

Proof This follows directly from the fact that there are at most $n_A + 1$ possible completion times for job B. □

We next focus on the calculation of strict Pareto optima for the F^A and $C^B_{\max}$ criteria. More precisely, we provide algorithms, based on the ε-constraint approach, which enumerate one strict Pareto optimum per strictly non-dominated criteria vector. The enumeration of this set $\mathcal{P}$ of solutions can be achieved by iteratively solving, with different ε values, the $Q|p_i = p|\varepsilon(F^A/C^B_{\max})$ problem (see Algorithm 1). This generic algorithm can be implemented in $\mathcal{O}(n \log m + n_A \Phi)$ time, where $n \log m$ is the time needed to generate n timeslot sequence $\mathcal{T}$ and we need Φ time to optimally assign operations of job A to the timeslots in $\mathcal{T}$ for each of at most $n_A + 1$ non-dominated solutions. However, later in this paper we will see that Algorithm 1 can be improved depending on the definition of criterion F^A.

Algorithm 1: Solution of the $Q|p_i = p|\#(F^A, C^B_{\max})$ problem.

Data: An instance of the $Q|p_i = p|\#(F^A, C^B_{\max})$ problem.
Result: Strict Pareto set $\mathcal{P}$.

begin
  $\mathcal{P} \leftarrow \emptyset$; /* Initialize Pareto set. */
  $\varepsilon \leftarrow t_{n_B}$, $n_\varepsilon \leftarrow n_B$; /* Set initial ε and n_ε values. */
  repeat
    $n_\varepsilon \leftarrow \text{argmax}_{k=n_\varepsilon,\ldots,n}\{t_k \leq \varepsilon\}$; /* Index of the last feasible timeslot for the operation in job B. */
    Put the whole job B into $t_{n_\varepsilon - n_B + 1}, \ldots, t_{n_\varepsilon}$ timeslots;
    Sequence operations of A to the $t_1, \ldots, t_{n_\varepsilon - n_B}, t_{n_\varepsilon+1}, \ldots, t_n$ timeslots;
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\sigma_\varepsilon, F^A, C^B_{\max} = \varepsilon)\}$; /* Append solution to the Pareto set. */
    $\varepsilon \leftarrow t_{n_\varepsilon+1}$; /* Increase ε value. */
  until $n_\varepsilon < n$;
  Remove weak but not strict non-dominated solutions from $\mathcal{P}$;
  return $\mathcal{P}$;
end

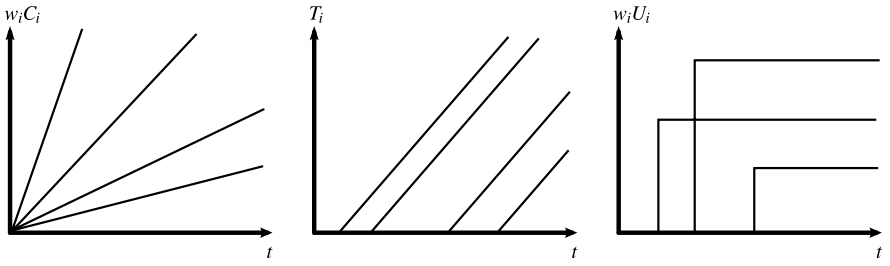

Fig. 2 Some cost functions illustrating Assumption 1

3 Solving the $Q|p_i = p|\#(F_{\sum}^{A}, C_{\max}^{B})$ problem

In this section we focus on the case $F^A = F_{\sum}^{A} = \sum_{i\in A} f_i$, where f_i is for all i a non-decreasing function of the completion times. First, observe that sequencing job A in Step 7 of Algorithm 1 can be achieved by solving an assignment problem in $\mathcal{O}(n_A^3)$ time, thus leading to an overall $\mathcal{O}(n\log m + n_A^4)$ time complexity for Algorithm 1. We show that this complexity can be reduced if the following assumption holds.

Assumption 1 Let $f_{i'}$ and $f_{i''}$ be two non-decreasing functions: For all $t_2 \geq t_1 \geq 0$, we have either $(f_{i'} - f_{i''})(t_1) \leq (f_{i'} - f_{i''})(t_2)$ or $(f_{i'} - f_{i''})(t_1) \geq (f_{i'} - f_{i''})(t_2)$, with $(f_{i'} - f_{i''})(t) = f_{i'}(t) - f_{i''}(t)$.

It is interesting to notice that some of the classic sum criteria in scheduling satisfy this assumption. For instance, this is the case for the weighted sum of completion times criterion $\sum w_iC_i$ for which $w_iC_i = f_i(t) = w_it$. This assumption also holds for the total tardiness criterion $\sum T_i$, with $T_i = f_i(t) = \max(0, t - d_i)$ and for the total weighted tardiness $\sum w_iT_i$ if weights w_i and due dates d_i are agreeable, i.e. $d_{i'} \leq d_{i''} \Rightarrow w_{i'} \geq w_{i''}$. Figure 2 illustrates that for criteria $\sum w_iC_i$ and $\sum T_i$ Assumption 1 holds, whilst this is not the case for weighted number of tardy jobs criterion $\sum w_iU_i$ (two cost functions intersect), where $w_iU_i = f_i(t) = w_i$ if $t > d_i$ and $w_iU_i = f_i(t) = 0$ otherwise.

In the remainder of this section we assume that Assumption 1 holds and we use the following notations: $\sigma(k)$ denotes the operation sequenced in position k in sequence σ and $t_k \in \mathcal{T}$ is its completion time, i.e. the completion time of the timeslot in which $\sigma(k)$ is scheduled. We are now ready to present a result which states that the optimal sequence of job A operations does not depend on the value of the timeslots.

Lemma 3 *Assume that we have two sets of timeslots* $\mathcal{T}' = \{t'_1 \leq \cdots \leq t'_n\}$ *and* $\mathcal{T}'' = \{t''_1 \leq \cdots \leq t''_n\}$ *such that* $\mathcal{T}' \neq \mathcal{T}''$. *Let* σ' *and* σ'' *be optimal sequences of operations for criterion* $F_{\sum}^{A}$ *with respect to the timeslots of* $\mathcal{T}'$ *and* $\mathcal{T}''$, *respectively. Then* $\sigma' = \sigma''$.

Proof We will prove this lemma using a contradiction. First we will assume that we are given two different schedules σ' and σ'' and finally show that σ'' can be transformed to be equal to σ' without increasing its objective value.

Suppose $\sigma' \neq \sigma''$ and let $k < n$ be the index such that $\sigma'(v) = \sigma''(v)$, $\forall v = 1, \ldots, k-1$, and $\sigma'(k) \neq \sigma''(k)$. We assume that σ' and σ'' are maximally equal, i.e. there does not exist other optimal sequences σ' and σ'' with a higher k value.

Consider operation $i' = \sigma'(k)$ which is scheduled in σ'' in position $r > k$, and set $i'' = \sigma''(r-1)$. If we denote by ℓ the index such that $\sigma'(\ell) = i''$ we conclude $\ell > k$. As σ' is optimal regarding timeslots $\mathcal{T}'$, swapping i' and i'' in σ' we obtain

$$\delta'_{\sum} = F_{\sum}^{A}\big(\sigma'(i' \leftrightarrow i'')\big) - F_{\sum}^{A}(\sigma') \geq 0,$$

where $\sigma'(i' \leftrightarrow i'')$ is the sequence σ' with operations i' and i'' swapped. The above equation can be rewritten as

$$\begin{aligned}\delta'_{\sum} &= \big(f_{i'}(t_\ell) + f_{i''}(t_k)\big) - \big(f_{i'}(t_k) + f_{i''}(t_\ell)\big)\\ &= (f_{i'} - f_{i''})(t_\ell) - (f_{i'} - f_{i''})(t_k) \geq 0.\end{aligned}$$

Recall that $t_\ell \geq t_k$, thus function $(f_{i'} - f_{i''})(t)$ is non-decreasing. Now, let us turn to σ'' which is optimal regarding timeslots $\mathcal{T}''$. By swapping i' and i'' we obtain

$$\begin{aligned}\delta''_{\sum} &= F_{\sum}\big(\sigma''(i' \leftrightarrow i'')\big) - F_{\sum}(\sigma'')\\ &= \big(f_{i''}(t_r) + f_{i'}(t_{r-1})\big) - \big(f_{i''}(t_{r-1}) + f_{i'}(t_r)\big)\\ &= (f_{i'} - f_{i''})(t_{r-1}) - (f_{i'} - f_{i''})(t_r).\end{aligned}$$

As $(f_{i'} - f_{i''})(t)$ is a non-decreasing function and $t_r \geq t_{r-1}$ we deduce that $\delta''_{\sum} \leq 0$. But as σ'' is assumed to be optimal regarding timeslots $\mathcal{T}''$, we necessarily have $\delta''_{\sum} = 0$ and swapping i' and i'' does not increase the objective function value of $F_{\sum}$ over $\mathcal{T}''$. But this clearly contradicts the fact that σ' and σ'' have been chosen as maximally equal. Consequently, we have $\sigma' = \sigma''$. □

Using Lemma 3 we can improve Algorithm 1 to reduce its time complexity, since operations of job A can be sequenced independently from the position of job B in the

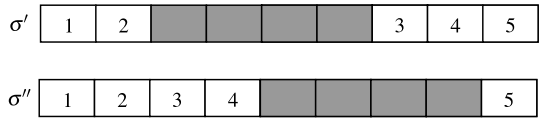

Fig. 3 The sequence of job A operations is independent from job B position (*gray*) in timeslot sequence $\mathcal{T}$

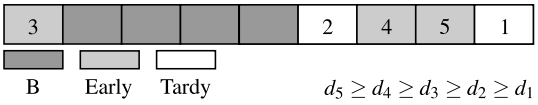

Fig. 4 Adoption of Dessouky et al. (1990) algorithm for $Q|p_i = p|\sum U_i$ problem

timeslot sequence $\mathcal{T}$ (see Fig. 3). Consequently, in the resulting improved Algorithm 1, the $Q|p_i = p|F_{\sum}^{A}$ problem is solved only once to produce an optimal sequence σ^* in Step 3. Then in Step 7 the operations of job A are assigned to the free timeslots in their order in sequence σ^*. The overall complexity for finding all non-dominated criteria vectors is $\mathcal{O}(n \log m + \Phi)$ time, with Φ the time complexity needed to solve the $1|p_i = 1|F_{\sum}^{A}$ problem. Notice that if we want to save one schedule per criteria vector, we need to pay an extra cost which yields in this case $\mathcal{O}(n \log m + \Phi + n_A^2 n_B + n_A n_B^2)$ time complexity.

We now turn to special cases of the $F_{\sum}^{A}$ cost function for which Assumption 1 holds, and we first start with $F_{\sum}^{A} = \sum w_i C_i^A$, $w_i \geq 0, \forall i = 1, \ldots, n_A$. It is well known (Dessouky et al. 1990) that $Q|p_i = p|\sum w_i C_i$ can be solved in polynomial time by a straight adaptation of Smith's rule: schedule at any time the job with the greatest weight w_i to the machine which completes it the earliest. Henceforth, sequence σ^* is in Step 3 of Algorithm 1 obtained by sorting job A operations in decreasing order of their weight values. Therefore, the $Q|p_i = p|\#(\sum w_i C_i^A, C_{\max}^B)$ problem can be solved in $\mathcal{O}(n \log m + n_A \log n_A)$ time.

Additionally, both $Q|p_i = p|F_{\sum}^{A}$ problems with $F_{\sum}^{A} = \sum T_i^A$ and $F_{\sum}^{A} = \sum w_i T_i^A$ (with agreeable weights and due dates) can be solved (Dessouky et al. 1990) by a straight adaptation of Jackson's rule: schedule at any time the job with the smallest due date d_i to the machine which completes it the earliest. Again, sequence σ^* is obtained by sorting operations of job A in increasing order of their due dates. This also results in an overall $\mathcal{O}(n \log m + n_A \log n_A)$ time complexity for the improved Algorithm 1.

There are objective functions in the scheduling theory which do not satisfy Assumption 1. One of these functions is the total number of tardy jobs $\sum U_i^A$. However, Dessouky et al. (1990) present an algorithm for solving the $Q|p_i = 1|\sum w_i U_i$ problem which can be simplified for solving the $Q|p_i = p|\sum U_i$ problem. First, sort operations in descending order of the due dates to obtain list $\mathcal{L}$. Operations are iteratively scheduled as follows: Consider the last free timeslot $t_k \in \mathcal{T}$, then the first available operation, $i = \mathcal{L}[1]$, is assigned to this timeslot if $t_k \leq d_i$. Otherwise, consider the previous free timeslot and let t_k be unassigned. Finally, sequentially schedule the remaining, tardy, operations in the order of $\mathcal{L}$ to the unassigned timeslots such that the operation with the latest due date is assigned to the first unassigned timeslot (see Fig. 4). Correctness of the procedure can be easily proved by interchange arguments.

In the remainder of this section we assume that $t_1 \leq d_i$ holds for all $i \in A$. If this is not the case, label such operations always tardy by a simple preprocessing step. We present the results which lead us to an efficient enumeration algorithm for the $Q|p_i = p|\#(\sum U_i^A, C_{\max}^B)$ problem.

Lemma 4 *There exists a strict Pareto optimum σ for the $Q|p_i = p|\#(\sum U_i^A, C_{\max}^B)$ problem with no tardy operation $u \in A$ sequenced before job B.*

Proof Let σ' be a strict Pareto optimum, with criteria values $\sum U'^A$ and $C_{\max}'^B$, such that it contains a tardy operation $u \in A$ sequenced before job B. Let σ be the schedule obtained by moving u right after job B and by left timeshifting all operations scheduled between the old and new timeslot of u. Consequently, the number of tardy A operations and the makespan of job B do not increase. By applying this repeatedly we can build a strict Pareto optimal schedule with no tardy A operation before job B. □

Notice that Lemma 4 is even stronger for a single machine, since in this case a strict Pareto optimal schedule with some tardy operation $u \in A$ sequenced before job B and the same criteria values do not exist. This is true, since there are no two timeslots with equal completion times, therefore swapping tardy operation u with job B decreases makespan criteria without increasing the number of tardy operations of A.

We now provide two results related to the process of enumerating the set of strict Pareto optima: the first one establishes a stopping condition whilst the second one concerns the choice of ε values to generate succeeding non-dominated schedules.

Lemma 5 *Let σ be a strict Pareto optimal solution with objective value $(\sum U^A, C_{\max}^B)$. If there exists a timeslot $t_k < C_{\max}^B$ with $\sigma(k)$ a job A operation such that $t_{k+1} > d_{\sigma(k)}$, then there is no succeeding strict Pareto optimal solution σ' such that $\sum U^A > \sum U'^A$ and $C_{\max}^B < C_{\max}'^B$.*

Proof According to Lemma 4 we assume that no operation i from job A sequenced before job B in σ is tardy. Let σ' be a succeeding strict Pareto optimal schedule of σ such that $\sum U^A > \sum U'^A$ and $C_{\max}^B < C_{\max}'^B$ hold. As $C_{\max}^B < C_{\max}'^B$, it follows that job B is shifted in σ' $r \geq 1$ timeslots to the right.

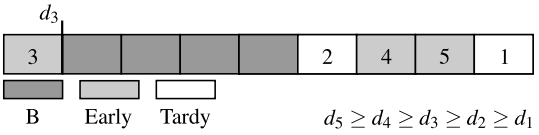

Fig. 5 Illustration of stop condition in Lemma 5

Consequently, job A operations sequenced before job B in σ may also be postponed by r timeslots to the right as long as they stay early. However, we know that operation $\sigma(k) \in \sigma$ is tardy if right timeshifted. Since all operations sequenced before $\sigma(k)$ and all tardy operations in σ have due dates not greater than $d_{\sigma(k)}$ none of them can be sequenced non tardy in a position greater than k in σ'. Therefore $\sum U^A = \sum U'^A$ and σ' is not a strict Pareto optimal solution, which is a contradiction (see Fig. 5). □

Lemma 6 *Let σ be a strict Pareto optimal solution with objective value $(\sum U^A, C^B_{\max})$, and let k be a timeslot with the first tardy operation $\sigma(k) \in A$ following job B. The succeeding strict Pareto optimal solution σ' with $\sum U^A > \sum U'^A$ and $C^B_{\max} < C'^B_{\max}$ is obtained by solving the $Q|p_i = p|\varepsilon(\sum U_i^A / C^B_{\max})$ problem with $\varepsilon = t_k$*

Proof Let σ be such that Lemma 5 does not hold. Now observe that there is no strict Pareto optimal solution with $\varepsilon < t_k$ satisfying the conditions of Lemma 5. This is true, since such schedule only swaps job B with early operations following it and therefore does not change the number of tardy operations, but rather increases $C^B_{\max}$.

Let $\varepsilon = t_k$ (see Fig. 6), then makespan of job B in schedule σ' is $C'^B_{\max} = t_k > C^B_{\max}$ since operation $\sigma(k)$ follows job B in σ and Lemma 5 does not hold. In the new schedule σ' job B is sequenced by $r \geq 1$ timeslots later. Hence, all early operations of A sequenced between B and $\sigma(k)$ in σ can be scheduled immediately before B in σ' and operation $\sigma(k)$ in the first timeslot of σ'. Note that there may be more job A operations scheduled in timeslots $j > k$ such that $t_j = \varepsilon = t_k$. If this is the case we sequence them iteratively as described above. Thus the number of tardy jobs is reduced by at least one with $C'^B_{\max} > C^B_{\max}$ and this change is minimal: therefore schedule σ' is a succeeding strict Pareto optimal solution of σ. □

Due to the lemma above we can easily construct an efficient algorithm to enumerate the set of strict Pareto optima.

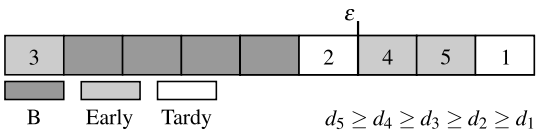

Fig. 6 Optimal choice of ε value in Lemma 6

Algorithm 2: Solution of the $Q|p_i = p|\#(\sum U_i^A, C^B_{\max})$ problem.

Data: An instance of the $Q|p_i = p|\#(\sum U_i^A, C^B_{\max})$ problem.
Result: Strict Pareto set $\mathcal{P}$.

begin
  $\mathcal{P} \leftarrow \emptyset$; `/* Initialize Pareto set. */`
  Find σ an initial solution minimizing $Lex(C^B_{\max}, \sum U^A)$;
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\sigma, \sum U^A, C^B_{\max})\}$; `/* Append initial solution to P. */`
  $n_T \leftarrow \texttt{argmax}_{l=n_B,\ldots,n}\{t_l \leq C^B_{\max}\}$;
  while (*Lemma 5 is not answered*) **do**
    $\varepsilon \leftarrow C^B_{\max}$, $n_\varepsilon \leftarrow n_T$;
    $k \leftarrow \texttt{argmin}_{l=n_{\varepsilon+1},\ldots,n}\{t_l > d_{\sigma(l)}\}$; `/* Index of the first tardy operation following job B. */`
    $n_T \leftarrow k$; `/* Index of the last feasible timeslot for the operations of job B. */`
    $E \leftarrow \{\sigma(l) \mid n_\varepsilon < l \leq n_T \text{ and } t_l \leq d_{\sigma(l)}\}$; `/* Sequence of early operations following B, but before its new completion time. */`
    $T \leftarrow \{\sigma(l) \mid n_\varepsilon < l \leq n_T \text{ and } t_l > d_{\sigma(l)}\}$; `/* Sequence of tardy operations following B, but before its new completion time. */`
    Build non-dominated schedule σ_ε with $\varepsilon = t_k$ by the following steps:
    Put operations in T to $t_1, \ldots, t_{|T|}$ timeslots in increasing order of their due dates;
    Put operations in E to $t_{n_T - n_B - |E|+1}, \ldots, t_{n_T - n_B}$ timeslots in increasing order of their due dates;
    Put the whole job B to $t_{n_T - n_B+1}, \ldots, t_{n_T}$ timeslots;
    Sequence all other operations to the rest timeslots in the same order as in σ;
    $\sigma \leftarrow \sigma_\varepsilon$, $\sum U^A \leftarrow \sum U^A - |T|$, $C^B_{\max} \leftarrow \varepsilon$;
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\sigma, \sum U^A, C^B_{\max})\}$; `/* Append solution to the Pareto set. */`
  return $\mathcal{P}$;
end

First, schedule job B and sequence operations of job A, i.e. build an initial strict Pareto optimal solution σ, such that $(C^B_{\max}, \sum U_i^A)$ is minimized with respect to the lexicographical ordering. This can be done by assigning op-

erations of job B to the first timeslots and next sequence operations of job A. Then use results of Lemma 6 and its constructive proof to build succeeding strict Pareto optimal solutions until the stopping condition of Lemma 5 holds. The details which are given in Algorithm 2 can be implemented in $\mathcal{O}(n \log m + n_A \log n_A)$ time if only strictly non-dominated criteria vectors are of interest. Otherwise we get an $\mathcal{O}(n \log m + n_A^2)$ time complexity.

To complete this section, notice that in the case where $F_{\sum}^A$ is equal to the weighted number of late jobs $\sum w_i U_i^A$, we have to go back to Algorithm 1. Henceforth, the strict Pareto optima of the $Q|p_i = p|\#(\sum^A w_i U_i, C_{\max}^B)$ problem can be calculated in $\mathcal{O}(n \log m + n_A^2 \log n_A)$ time using the procedure of Dessouky et al. (1990) in Step 7.

4 Solving the $Q|p_i = p|\#(F_{\max}^A, C_{\max}^B)$ problem

In this section we focus on the case where $F^A = F_{\max}^A = \max_{i \in A} f_i$, where f_i is for all i a non-decreasing function of the completion times. This type of criterion has often been considered in the literature since it generalizes classic scheduling criteria such as makespan, maximum tardiness or maximum lateness. A straightforward implementation of Algorithm 1 can be derived for the $Q|p_i = p|\#(F_{\max}^A, C_{\max}^B)$ problem using Lawler's algorithm (Lawler 1973) when applying Step 7. Consequently, this step can be solved in $\mathcal{O}(n_A^2)$ time and the overall complexity of the enumeration algorithm becomes $\mathcal{O}(n \log m + n_A^3)$. In the remainder, we focus on two particular and widely used scheduling criteria, namely the makespan $C_{\max}^A = \max_{i \in A} C_i$ and the maximum lateness $L_{\max}^A = \max_{i \in A}(C_i - d_i)$ for which we provide dedicated algorithms.

Firstly, it is obvious that the $Q|p_i = p|\#(C_{\max}^A, C_{\max}^B)$ problem admits only two strictly non-dominated criteria vectors: $(C_{\max}^A = t_{n_A}, C_{\max}^B = t_n)$ and $(C_{\max}^A = t_n, C_{\max}^B = t_{n_B})$.

Now, we turn to the maximum lateness criterion. Even though Lemma 3 does in general not apply to $F_{\max}^A$ functions, we show that in case of $L_{\max}^A$ a similar result holds.

Lemma 7 *Whatever the timeslots $\mathcal{T} = \{t_1 \leq \cdots \leq t_n\}$ are, the $Q|p_i = p|L_{\max}^A$ problem can be solved by sorting operations of job A in ascending order of their due dates (EDD rule) and by assigning them in this order to the timeslots.*

Proof This result can be shown by simple interchange arguments. □

As a consequence of the above lemma it is possible to derive a generic algorithm which follows the same lines as Algorithm 1 for the $F_{\sum}^A$ criterion. The main difference is in Step 3 where schedule σ^* is built: for the $F_{\max}^A$ criterion we only have to sequence operations of job A following EDD order. This general algorithm has an overall $\mathcal{O}(n \log m + n_A \log n_A)$ time complexity to find strictly non-dominated criteria vectors and $\mathcal{O}(n \log m + n n_A)$ if corresponding schedules are of interest. However, it is still possible to reduce its practical time complexity by proposing an enumeration algorithm which eliminates from computation the weak, but not strict, non-dominated criteria vectors. The idea is to avoid building the non-dominated solutions from scratch for each ε value.

Let us assume that operations of job A are divided into two sets A' and A'', where A' contains all operations that are sequenced before job B in a strict Pareto optimal schedule σ and A'' is defined by $A'' = A \setminus A'$. The idea is to build the succeeding strict Pareto optimal schedule σ' starting from σ. The next result introduces a stopping criteria for the enumeration process.

Lemma 8 *If schedule σ is a strict Pareto optimal solution for the $Q|p_i = p|(L_{\max}^A, C_{\max}^B)$ problem with criteria vector$(L_{\max}^A, C_{\max}^B)$ and the $L_{\max}^A$ value is given by an operation of set A', then there is no succeeding strict Pareto optimal schedule σ' with criteria values $L_{\max}^{'A} < L_{\max}^A$ and $C_{\max}^{'B} > C_{\max}^B$.*

Proof To decrease the $L_{\max}^A$ value we have to increase $C_{\max}^B$, i.e. we right timeshift the block of operations of job B. Consequently, some operations of set A'' are moved to set A' but, due to Lemma 7, operations in A' still follow the EDD order. Thus we have not decreased the $L_{\max}^A$ value, i.e. $L_{\max}^{'A} = L_{\max}^A$. □

From Lemma 8 we conclude that the enumeration, which starts by scheduling first the operations of job B, must stop when the $L_{\max}^A$ value is given by an operation sequenced in set A'. At each iteration, a new ε value is set and the operations of job B are therefore right timeshifted. However, not all ε values lead to strictly non-dominated criteria vectors. Lemma 9 states a condition how much operations of job B must be right timeshifted in order to get a new succeeding strictly non-dominated criteria vector.

Lemma 9 *Let σ be a strict Pareto optimal schedule with criteria vector $(L_{\max}^A, C_{\max}^B)$, where the $L_{\max}^A$ value is given by an operation $i \in A''$ (break ties by choosing the latest operation in A'' which gives the $L_{\max}^A$ value). The succeeding strict Pareto optimal schedule σ' with $L_{\max}^{'A} < L_{\max}^A$ is obtained by moving all operations scheduled before operation i and including i from set A'' to set A'.*

Proof Given a schedule σ' such that $L_{\max}^{'A} < L_{\max}^A$, operation i must be completed earlier in σ' than in σ. This, however, means that i is swapped with at least one operation u

Algorithm 3: Solution of the $Q|p_i = p|\#(L^A_{\max}, C^B_{\max})$ problem.

Data: An instance of the $Q|p_i = p|(L^A_{\max}, C^B_{\max})$ problem.
Result: Strict Pareto set $\mathcal{P}$.

```
begin
    Sort operations of job A in ascending order of their due dates;
    P ← ∅; /* Initialize Pareto set. */
    ε ← t_{n_B}, n_ε ← n_B; /* Set initial ε and n_ε values. */
    repeat
        n_ε ← argmax_{k=n_ε,...,n}{t_k ≤ ε}; /* Index of the last feasible timeslot for the operations of job B. */
        A' ← {First n_ε − n_B operations of A}; /* Early operations of job A. */
        A'' ← A \ A'; /* Late operations of job A. */
        Sequence operations of job A' to the t_1, ..., t_{n_ε − n_B} timeslots of σ_ε;
        Put the whole job B into t_{n_ε − n_B + 1}, ..., t_{n_ε} timeslots;
        Sequence operations of job A'' to the rest t_{n_ε + 1}, ..., t_n slots of σ_ε;
        This results to the schedule σ_ε with criteria L^A_max and C^B_max = t_{n_ε};
        P ← P ∪ {σ_ε}; /* Append σ_ε to the Pareto set. */
        I' ← {i ∈ A'' | C_i − d_i = L^A_max}; /* Determine indexes of jobs issuing L^A_max value. */
        ε ← max_{i∈I'} C_i; /* Update ε-constraint value. */
    until I' ≠ ∅ ;
    return P;
end
```

sequenced before i in σ. First suppose that u is an operation of job A. As u is scheduled before i in σ, the inequality $d_u \le d_i$ implies by Lemma 7 two cases:

(i) either $d_u = d_i \Rightarrow L^{'A}_{\max} = L^A_{\max}$,
(ii) or $d_u < d_i \Rightarrow L^{'A}_{\max} > L^A_{\max}$,

which both contradict the assumption $L^{'A}_{\max} < L^A_{\max}$. Hence operation u must belong to job B. Furthermore, due to Lemma 1 operations in B can be considered as a single block, thus i and all operations j of job A scheduled before i in σ must be sequenced before job B in σ'. □

Using the two previous lemmas we derive an improved enumeration algorithm (Algorithm 3) for finding the set of strict Pareto optima of the $Q|p_i = p|\#(L^A_{\max}, C^B_{\max})$ problem. After the first iteration, we get an initial solution σ_ε which actually is a solution of the lexicographical $Q|p_i = p|Lex(C^B_{\max}, L^A_{\max})$ problem where $C^B_{\max}$ is minimized first. Then, the succeeding strict Pareto schedule σ'_ε is obtained by identifying the last operation i of job A with lateness equal to $L^A_{\max}(\sigma_\varepsilon)$ and by moving all operations j of job A, such that $d_j \le d_i$, to set A' in schedule σ'_ε. This way, a succeeding schedule σ'_ε with $L^{'A}_{\max} < L^A_{\max}$ and $\varepsilon = C^{'B}_{\max} > C^B_{\max}$ is built. The operations of job A still follow the EDD order. This process is iterated until the stopping condition stated in Lemma 8 is met.

In the worst case the complexity of Algorithm 3 is the same as that of Algorithm 1, i.e. $\mathcal{O}(n \log m + n_A \log n_A)$ time. We perform, however, only as many iterations as strictly non-dominated solutions exist. This means that the practical time complexity will in general be reduced.

We close this section by strengthening the result of Lemma 2.

Lemma 10 *For the $Q|p_i = p|\#(L^A_{\max}, C^B_{\max})$ problem there are at most $(n_A + 1)$ strictly non-dominated criteria vectors and the bound is tight.*

Proof We show that the bound is tight by means of the following example.
Consider a single machine problem, i.e. $m = 1$, for which all operations of jobs A and B have $p = 1$. The due dates for operations in A are:

$$d_i = i + \frac{(i-1)n_B}{n_A}, \quad 1 \le i \le n_A.$$

The timeslots are defined by $t_k = k$ for all $1 \le k \le n$, which implies that if job B is scheduled first we get schedule σ_1 with the following maximum lateness for job A:

$$L^A_{\max}(\sigma_1) = \max\left\{n_B + 1 - 1, n_B + 2 - 2 - \frac{n_B}{n_A}, \ldots, n - n_A - \frac{(n_A - 1)n_B}{n_A}\right\} = n_B$$

and makespan $C^B_{\max}(\sigma_1) = n_B$. When right timeshifting job B by one timeslot we obtain the second strict Pareto optimal solution σ_2 with criteria values given by

$$L^A_{\max}(\sigma_2) = \max\left\{1 - 1, n_B + 2 - 2 - \frac{n_B}{n_A}, \ldots, n - n_A - \frac{(n_A - 1)n_B}{n_A}\right\} = n_B - \frac{n_B}{n_A}$$

Table 1 Complexity results on two-agent uniform parallel machines scheduling

Problem	Complexity	Comment/Reference
$Q\|p_i = p\|\#(F_{\sum}^A, C_{\max}^B)$	$\mathcal{O}(n^4)$	$F_{\sum}^A$—regular, Algorithm 1
$Q\|p_i = p\|\#(F_{\max}^A, C_{\max}^B)$	$\mathcal{O}(n^3)$	$F_{\max}^A$—regular, Algorithm 1
$Q\|p_i = p\|\#(F_{\sum}^A, C_{\max}^B)$	$\mathcal{O}(n \log m + \Phi)$	Assumption 1, $\mathcal{O}(\Phi)$—time to solve $1\|p_i = 1\|F_{\sum}^A$
$Q\|p_i = p\|\#(\sum w_i C_i^A, C_{\max}^B)$	$\mathcal{O}(n \log m + n_A \log n_A)$	Lemma 3
$Q\|p_i = p\|\#(\sum T_i^A, C_{\max}^B)$	$\mathcal{O}(n \log m + n_A \log n_A)$	Lemma 3
$Q\|p_i = p\|\#(\sum w_i T_i^A, C_{\max}^B)$	$\mathcal{O}(n \log m + n_A \log n_A)$	Lemma 3, agreeable weights and due dates
$Q\|p_i = p\|\#(\sum U_i^A, C_{\max}^B)$	$\mathcal{O}(n \log m + n_A \log n_A)$	Lemmas 4–6, Algorithm 2
$Q\|p_i = p\|\#(\sum w_i U_i^A, C_{\max}^B)$	$\mathcal{O}(n \log m + n_A^2 \log n_A)$	Algorithm 1
$Q\|p_i = p\|\#(C_{\max}^A, C_{\max}^B)$	$\mathcal{O}(n \log m)$	Lemma 1, Only two Pareto schedules
$Q\|p_i = p\|\#(L_{\max}^A, C_{\max}^B)$	$\mathcal{O}(n \log m + n_A \log n_A)$	Lemmas 7–9, Algorithm 3

and $C_{\max}^B(\sigma_2) = n_B + 1$. By doing this iteratively we build the $(n_A + 1)$ strict Pareto optimal solutions with the last one σ_{n_A+1} having

$$L_{\max}^A(\sigma_{n_A+1}) = \max\left\{1 - 1, 2 - 2 - \frac{n_B}{n_A}, \ldots, n_A - n_A - \frac{(n_A - 1)n_B}{n_A}\right\} = 0$$

and $C_{\max}^B(\sigma_{n_A+1}) = n$ criteria values. □

5 Conclusions

In this paper we have tackled the problem of scheduling jobs owned by two agents on uniform parallel machines. One important feature of our problem is the make-up of jobs by the sets of equal-size operations. We consider the case where one job is evaluated by means of the makespan criterion whilst the other one is evaluated by means of a general cost function. To the best of our knowledge there is no literature on uniform parallel machines and equal job processing times which consider two-agent scheduling problems and in particular this general problem has never been dealt with in the literature. Moreover, only Agnetis et al. (2000) considered efficient ways to fully enumerate the strict Pareto optima while most other works rather concentrated on supplying algorithms to find a single non-dominated solution, mostly based on ε-constraint approach, or showed the $\mathcal{NP}$-completeness of the problems. On the contrary, we have provided polynomial time algorithms for the enumeration of the strict Pareto optima set in the cases where the general cost function is a max-cost function or a sum-cost function. We have also analyzed some specializations to classic scheduling criteria including the maximum lateness, the weighted sum of completion times, the number of tardy jobs and total tardiness. The observed problem properties let us derive efficient polynomial time algorithms to enumerate the strict Pareto optima sets with an improved running time over the naive algorithm implementation using a straight forward ε-constraint approach. Table 1 summarizes the problems considered in this paper and the running times of the proposed algorithms.

There is still a large field of research which needs to be covered regarding multi-agent parallel machine scheduling. This paper is a useful generalization which can be used for other more complicated environments, like jobs with non-equal processing times, unrelated parallel machines problems, flexible flow shop problems and others. It would be interesting to look at the different cases where both agent jobs have release dates or only one of them. Besides, it could be worthy of interest to study problems with non regular criteria, like the earliness of jobs, which to the best of our knowledge has never been done in the literature.

References

Agnetis, A., Mirchandani, P., Pacciarelli, D., & Pacifici, A. (2000). Nondominated schedules for a job-shop with two competing users. *Computational & Mathematical Organization Theory*, *6*(2), 191–217.

Agnetis, A., Mirchandani, P., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research*, *42*(2), 229–242.

Baker, K., & Smith, J. (2003). A multiple-criterion model for machine scheduling. *Journal of Scheduling*, *6*, 7–16.

Balasubramanian, H., Fowler, J., Keha, A., & Pfund, M. (2009). Scheduling interfering job sets on parallel machines. *European Journal of Operational Research*, *199*(1), 55–67. doi:10.1016/j.ejor.2008.10.038.

Baptiste, P. (1999). Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. *Journal of Scheduling*, *2*, 245–252.

Baptiste, P. (2000). Scheduling equal-length jobs on identical parallel machines. *Discrete Applied Mathematics*, *103*(1–3), 21–32. doi:10.1016/S0166-218X(99)00238-3.

Baptiste, P. (2003). A note on scheduling multiprocessor tasks with identical processing times. *Computers and Operations Research*, *30*(13), 2071–2078.

Baptiste, P., & Brucker, P. (2004). Scheduling equal processing time jobs. In J. Y. Leung (Ed.), *Handbook of scheduling: algorithms, models and performance analysis*. London/ Boca Raton: Chapman & Hall / CRC Press.

Baptiste, P., Chrobak, M., Dürr, C., & Sourd, F. (2004). Preemptive multi-machine scheduling of equal-length jobs to minimize the average flow time. arXiv:cs/0412094.

Brucker, P., & Kravchenko, S. (2008). Scheduling jobs with equal processing times and time windows on identical parallel machines. *Journal of Scheduling*, *11*, 229–237.

Cheng, T., Ng, C., & Yuan, J. (2006). Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theoretical Computer Science*, *362*, 273–281.

Chrobak, M., Durr, C., Jawor, W., Kowalik, L., & Kurowski, M. (2006). A note on scheduling equal-length jobs to maximize throughput. *Journal of Scheduling*, *9*, 71–73.

Dessouky, M. I., Lageweg, B. J., Lenstra, J. K., & van de Velde, S. L. (1990). Scheduling identical jobs on uniform parallel machines. *Statistica Neerlandica*, *44*(3), 115–123.

Ehrgott, M. (2005). *Multicriteria optimization* (2nd ed.). Berlin: Springer.

Graham, R. L., EL, Lawler, Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, *5*, 287–326.

Huynh Tuong, N., & Soukhal, A. (2009). Interfering job set scheduling on two-operation three-machine flowshop. In *IEEE—RIVF international conference on computing and communication technologies (RIVF'09)*, Da Nang, Vietnam.

Huynh Tuong, N., Soukhal, A., & Billaut, J. C. (2009). New scheduling problems with interfering and independent jobs, 33 p. Paper submitted to *Journal of Scheduling* the 8 September 2009.

Lawler, E. L. (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, *19*(5), 544–546. doi:10.1287/mnsc.19.5.544.

Leung, J. Y. T., Pinedo, M., & Wan, G. (2010). Competitive Two-Agent Scheduling and Its Applications. *Operations Research*, *58*(2), 458–469. doi:10.1287/opre.1090.0744.

Ng, C., Cheng, T., & Yuan, J. (2006). A note on the complexity of the problem of two-agent scheduling on a single machine. *Journal of Combinatorial Optimization*, *12*(4), 387–394. doi:10.1007/s10878-006-9001-0.

T'kindt, V., & Billaut, J. C. (2006). *Multicriteria scheduling: theory, models and algorithms* (2nd ed.). Berlin: Springer.

Tuzikov, A., Makhaniok, M., & Männer, R. (1998). Bicriterion scheduling of identical processing time jobs by uniform processors. *Computers and Operations Research*, *25*(1), 31–35.

Yuan, J., Shang, W., & Feng, Q. (2005). A note on the scheduling with two families of jobs. *Journal of Scheduling*, *8*, 537–542.