Single machine multiagent scheduling problems with a global objective function

Nguyen Huynh Tuong, Ameur Soukhal, Jean-Charles Billaut

J Sched
DOI 10.1007/s10951-011-0252-y

Single-machine multi-agent scheduling problems with a global objective function

N. Huynh Tuong · A. Soukhal · J.-C. Billaut

Abstract In this paper, we consider the problem of scheduling independent jobs when several agents compete to perform their jobs on a common single processing machine. Each agent wants to minimise its cost function, which depends exclusively on its jobs and we assume that a global cost function concerning the whole set of jobs has to be minimised. This cost function may correspond to the global performance of the workshop or to the global objective of the company, independent of the objectives of the agents. Classical regular objective functions are considered and both the ε-constraint and a linear combination of criteria are used for finding compromise solutions. This new multi-agent scheduling problem is introduced into the literature and simple reductions with multicriteria scheduling and multi-agent scheduling problems are established. In addition, the complexity results of several problems are proposed and a dynamic programming algorithm is given.

Keywords Scheduling · Multi-agent · Single machine · Complexity · Dynamic programming

1 Introduction

Generally in scheduling literature, the quality of a schedule is given by a measure applied to the whole set of jobs. Indeed, classical models consider all jobs to be equivalent and the quality of the global schedule is given by applying the same measure to all jobs without distinction. For instance, the measure may be the maximum completion time of jobs (the makespan), the total flow time of jobs or a measure related to the jobs' tardiness, e.g. maximum tardiness and the total number of tardy jobs. Introducing distinctions between jobs is generally done by the means of weights. For instance, the total weighted completion time, the total weighted tardiness or the weighted number of tardy jobs introduce such distinctions. However, in this case the same measure is still applied to all the jobs to quantify the quality of a schedule.

In a real context, these models are not always reliable. In some practical situations, it can be necessary to consider several aspects of the schedule. For instance, the mean flow time (which is equivalent to the total completion time) and the respect for due dates can be of similar importance for the decision maker. In such cases, more than one objective functions is defined and the scheduling problem enters the field of multicriteria scheduling (T'kindt and Billaut 2006 and Hoogeveen 2005 present a complete state-of-the-art survey).

In some cases, it may happen that the jobs are not equivalent and that applying the same measure to all the jobs is not useful. For instance, it is possible to consider a workshop where jobs have the following particularities: whereas some jobs may have a soft due date with allowed tardiness (which must be minimised); other jobs may have hard due dates (i.e. due dates that must be respected) and still other jobs may have no due date (e.g. production for stock). For the first type of job, the decision maker wants to minimise the maximum delay. For the second type of job he imposes that there must be no delayed jobs and for the last type of job he wants to minimise the total flow time. These jobs are assessed according to different objectives, but these jobs are in competition for the use of the machines. This problem is a multicriteria scheduling problem, where a new type

N. Huynh Tuong · A. Soukhal · J.-C. Billaut (✉)
Laboratoire d'Informatique, Université François Rabelais,
64 Avenue Jean Portalis, 37200 Tours, France
e-mail: jean-charles.billaut@univ-tours.fr

A. Soukhal
e-mail: ameur.soukhal@univ-tours.fr

Published online: 21 September 2011

of compromise has to be obtained. In the literature, these problems are called "*interfering job sets*" (Balasubramanian et al. 2009), "*multi-agent scheduling*" (Agnetis et al. 2000; Cheng et al. 2006) or "*scheduling with competing agents*" (Agnetis et al. 2004). In all these studies, the authors consider a partition of the set of jobs in competition for the use of resources, with each subset having its own objective function to optimise.

In this paper, we consider a different version of this problem in which the performance of the whole set of jobs has to be minimised. Such a problem may appear in real life situations. For instance, SKF MDGBB (Medium Deep Groove Ball Bearings) factories are workshops composed of parallel machines (see Pessan et al. 2008a, 2008b). The objective is related to the minimisation of the flow time criterion (i.e. maximising the number of items produced) and concerns the whole set of jobs, denoted by $\mathcal{N}_0$. Generally, the jobs that ideally, should be produced daily exceed the production capacity. To impose the production of the remaining jobs (say $\mathcal{N}_1 \subset \mathcal{N}_0$) during the next day, another performance measure has to be applied, which is the minimisation of the number of tardy jobs (or any other due date-related measure). The measure concerning $\mathcal{N}_0$ is the total completion time minimisation, but the number of tardy jobs among $\mathcal{N}_1$ cannot exceed a given threshold. This is the field of *rescheduling problems* (Wu et al. 1993; Hall and Potts 2004). Another example can be found in shampoo packing systems (Mocquillon et al. 2006). Shampoo is delivered daily and stored in a dedicated storage area with a limited capacity. The problem is to find the most efficient way to pack shampoo of different types into bottles. A global objective is to maximise the production, thus reducing the setup times. At the same time, the times and quantities of future deliveries are known in advance. Thus, each type of product has to be produced daily so that its quantity never exceeds its allotted storage area. This is a typical problem where the global objective concerns all the products and where the subset of products is evaluated with another objective.

The rest of the paper is organised as follows: in Sect. 2 the problem is defined, the notations are introduced and a state-of-the-art survey is presented. The first simple reductions on existing multi-agent or multicriteria scheduling problems are also given. Section 3 deals with polynomially solvable single-machine problems. The application of simple reductions from known NP-hard problems is detailed in Sect. 4. Section 5 is devoted to the total completion time case and a pseudo-polynomial time dynamic programming is also presented.

2 Preliminaries

2.1 Problem definition and notations

A set $\mathcal{N}_0$ of n independent jobs has to be scheduled on a single machine. We assume that all the jobs are available at time 0, that preemption is not allowed, and that the processing times are deterministic and integers. In this paper, p_j denotes the processing time of job j and d_j denotes its due date, where $1 \leq j \leq n$; the single machine is always available and it can process only one job at a time.

$\mathcal{N}_k$ denotes the kth subset of jobs of $\mathcal{N}_0$. We assume that $\bigcup_{k=1}^{K} \mathcal{N}_k = \mathcal{N}_0$ and that $\bigcap_{k=1}^{K} \mathcal{N}_k = \emptyset$, with K being the number of subsets. We denote by n_k the number of jobs in $\mathcal{N}_k$, for $1 \leq k \leq K$.

We denote the completion time of job j by C_j. $\sum(w_j)C_j$ is the total (weighted) completion time. Additionally, $C_{\max}$ denotes the maximum completion time (makespan) and $L_{\max}$ denotes the maximum lateness, which is defined by $L_{\max} = \max_{1 \leq j \leq n}(C_j - d_j)$. We denote the total (weighted) tardiness by $\sum(w_j)T_j$, where $T_j = \max(0, C_j - d_j)$. In addition, U_j is equal to 1 if job j is tardy, and U_j is equal to 0 otherwise. $\sum(w_j)U_j$ denotes the (weighted) number of tardy jobs.

$f^k(\mathcal{N}_k)$ is the objective function associated with agent k ($1 \leq k \leq K$), i.e. to the subset of jobs $\mathcal{N}_k$ and this function is simply denoted f^k when there is no ambiguity. $f^0(\mathcal{N}_0)$ is the objective function associated with the whole set of jobs. The problem is to find the completion times of the jobs that minimise the functions f^k, $\forall k \in \{0, 1, \ldots, K\}$. Notice that in the case of only one agent, the problem already has two objective functions. According to the three-field notation proposed by Graham et al. (1979) and extended to multicriteria scheduling problems in T'kindt and Billaut (2006), we consider the following functions:

- $\varepsilon(f^0/f^1, f^2, \ldots, f^K)$ in the case of K agents. This denotes the ε-constraint approach, i.e. the minimisation of f^0, subject to $f^1 \leq \varepsilon_1$, $f^2 \leq \varepsilon_2, \ldots, f^K \leq \varepsilon_K$.
- $F_\ell(f^0, f^1, f^2, \ldots, f^K)$ in the case of K agents. This formulation indicates a linear combination of criteria, i.e. $F_\ell(f^0, f^1, f^2, \ldots, f^K) = \sum_{k=0}^{K} \lambda_k f^k$.

2.2 Related literature

The literature contains some results on multi-agent scheduling, but there are very few results when a global objective function is also considered.

Agnetis et al. (2004) consider the single machine, flow shop and open shop problems with two subsets of jobs $\mathcal{N}_1$ and $\mathcal{N}_2$. They consider the minimisation of an objective function for one subset of jobs subject to a bound for the other subset of jobs. In addition, they give some complexity

results and dynamic programming algorithms for the single-machine problem. The single-machine problem is also considered in Baker and Smith (2003). These authors consider several regular objective functions ($C_{\max}$, $\sum w_j C_j$, $L_{\max}$) and they propose an algorithm for the minimisation of a linear combination of the objective functions. Complexity results are given and some polynomially solvable cases are identified. Furthermore, Yuan et al. (2005) propose some complementary results for these problems. Figure 1 summarises the results presented in Baker and Smith (2003) and Yuan et al. (2005). Cheng et al. (2006) consider a single-machine problem with K disjoint subsets of jobs $\mathcal{N}_1, \ldots, \mathcal{N}_K$ ($\bigcup_{i=1}^{K} \mathcal{N}_i = \mathcal{N}_0$). Each job is associated with a deadline, and each subset is measured by the total number of tardy jobs. The authors prove that the decision problem denoted by $1|\sum w_j U_j^1 \leq \varepsilon_1, \ldots, \sum w_j U_j^K \leq \varepsilon_K|-$ is strongly NP-hard. When the number of agents is fixed, they show that the problem can be solved in pseudo-polynomial time and they give a fully polynomial time approximation scheme. Additionally, if the weights are equal to 1, the problem can be solved in polynomial time. Cheng et al. (2008) consider the single-machine multi-agent scheduling problem with K objective functions of min-max type. The authors prove that the feasibility problem can be solved in polynomial time, even if jobs are subject to precedence constraints. Furthermore, the authors show that the problems $1||\sum_{k=1}^{K}(L_{\max}^k)$, $1||\sum_{k=1}^{K}(T_{\max}^k)$ and $1||\sum_{k=1}^{K}(\sum w_j C_j^k)$ are NP-hard and some polynomially solvable cases are identified by them. Agnetis et al. (2007) consider single-machine two-agent scheduling problems (reported in Fig. 1). Two approaches are considered: (1) the "decision problem", which is to find a solution such that all the criteria are bounded and (2) the "Pareto-optimisation problem", where the aim is to find the set of all non-dominated solutions (denoted by "#" in Fig. 1). Some results are also given for some single-machine multi-agent scheduling problems.

2.3 Some simple reductions

If the decision problem P reduces to decision problem P', we use the notation $P \propto P'$. We also use the following notations: $\Gamma^{\max} = \{C_{\max}, L_{\max}\}$, $\Gamma^{\Sigma} = \{\sum C_j, \sum T_j, \sum U_j\}$, $\Gamma^{\Sigma w} = \{\sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$ and $\Gamma = \Gamma^{\max} \cup \Gamma^{\Sigma} \cup \Gamma^{\Sigma w}$. We focus on the objective functions, and f stands for the problem $\alpha|\beta|f$.

Proposition 1 *The following reductions hold*, $\forall f^0, f^1 \in \Gamma$:

1.1 $f^0 \propto F_\ell(f^0, f^1)$ *and* $f^1(\mathcal{N}_0) \propto F_\ell(f^0, f^1(\mathcal{N}_1))$
1.2 $f^0 \propto \varepsilon(f^0/f^1)$
1.3 $f^1(\mathcal{N}_0) \propto \varepsilon(f^0/f^1(\mathcal{N}_1))$

Proof 1.1 Take the linear combination where the coefficient associated with f^1 is equal to zero; take the linear combination where the coefficient associated with f^0 is equal to zero.

1.2 Take a large value for ε_1 associated to f^1.

1.3 Finding a solution which satisfies $f^1(\mathcal{N}_1) \leq \varepsilon$ is equivalent to the decision version of problem $\alpha|\beta|f^1(\mathcal{N}_0)$ with subset $\mathcal{N}_1$. □

Proposition 2 *The following reductions from bicriteria scheduling hold*, $\forall f^0, f^1 \in \Gamma$:

2.1 $F_\ell(f^0, f^1(\mathcal{N}_0)) \propto F_\ell(f^0, f^1(\mathcal{N}_1))$
2.2 $\varepsilon(f^0/f^1(\mathcal{N}_0)) \propto \varepsilon(f^0/f^1(\mathcal{N}_1))$

Proof Take $\mathcal{N}_1 = \mathcal{N}_0$ and the problems are the same. □

The consequence is that if a bicriteria scheduling problem is NP-hard, then the corresponding problem with one agent and a global objective function is NP-hard.

Proposition 3 *The following reductions from multi-agent scheduling hold*:

3.1 $F_\ell(f^1, f^2(\mathcal{N}_2)) \propto F_\ell(f^1, f^2(\mathcal{N}_0))$, $\forall f^1 \in \Gamma$, $\forall f^2 \in \Gamma^{\Sigma w} \cup \{L_{\max}, \sum U_j\}$
3.2 $\varepsilon(f^2(\mathcal{N}_2)/f^1) \propto \varepsilon(f^2(\mathcal{N}_0)/f^1)$, $\forall f^1 \in \Gamma$, $\forall f^2 \in \Gamma^{\Sigma w} \cup \{L_{\max}, \sum U_j\}$

Proof If $f^2 \in \Gamma^{\Sigma w}$, one can build an instance to problem $\alpha|\beta|F_\ell(f^1(\mathcal{N}_1), f^2(\mathcal{N}_0))$ with $w_j = 0$ for the jobs that are not in $\mathcal{N}_1$. Because $f^2 \in \Gamma^{\Sigma w}$, $f^2(\mathcal{N}_0) = f^2(\mathcal{N}_2)$. The proof holds for the ε-constraint approach.

If $f_2 \in \{L_{\max}, \sum U_j, \sum w_j U_j\}$, one can build an instance to problem $\alpha|\beta|F_\ell(f^1(\mathcal{N}_1), f^2(\mathcal{N}_0))$ with $d_j = HV$ for each job $j \notin \mathcal{N}_1$ (HV being an high value). Because $f^2 \in \{L_{\max}, \sum U_j, \sum w_j U_j\}$, the problems are the same. The same reasoning can be applied to the ε-constraint approach. □

Proposition 4 *The following reductions from multi-agent with a global objective function hold*:

4.1 $F_\ell(f^1(\mathcal{N}_1), f^0) \propto F_\ell(f^1(\mathcal{N}_0), f^0)$, $\forall f^1 \in \Gamma^{\Sigma w} \cup \{L_{\max}, \sum U_j\}, \forall f^0 \in \Gamma$.
4.2 $\varepsilon(f^0/f^1(\mathcal{N}_1)) \propto \varepsilon(f^0/f^1(\mathcal{N}_0)), \forall f^1 \in \Gamma^{\Sigma w} \cup \{L_{\max}, \sum U_j\}, \forall f^0 \in \Gamma$.

Proof See the proof for Proposition 3.1. □

These simple reductions are shown in Figs. 2 and 3.

Tables 1 and 2 summarise the complexity results that are established in the rest of this paper for single-machine problems ('H' indicates an NP-hard problem, 'P' indicates a polynomial problem and 'o' indicates an open problem).

Fig. 1 Some complexity results on multi-agent scheduling problems

Problem	Complexity	Reference
$1\|\|F_l(C_{\max}(\mathcal{N}_1), C_{\max}(\mathcal{N}_2))$	Polynomial	Baker and Smith (2003)
$1\|\|F_l(L_{\max}(\mathcal{N}_1), L_{\max}(\mathcal{N}_2))$	Polynomial	Baker and Smith (2003), Yuan et al. (2005)
$1\|\|F_l(\sum w_jC_j(\mathcal{N}_1), \sum w_jC_j(\mathcal{N}_2))$	Polynomial	Baker and Smith (2003)
$1\|\|F_l(C_{\max}(\mathcal{N}_1), L_{\max}(\mathcal{N}_2))$	Polynomial	Baker and Smith (2003)
$1\|\|F_l(C_{\max}(\mathcal{N}_1), \sum w_jC_j(\mathcal{N}_2))$	Polynomial	Baker and Smith (2003)
$1\|\|F_l(\sum C_j(\mathcal{N}_1), L_{\max}(\mathcal{N}_2))$	Polynomial	Yuan et al. (2005)
$1\|\|F_l(\sum w_jC_j(\mathcal{N}_1), L_{\max}(\mathcal{N}_2))$	Strongly NP-hard	Baker and Smith (2003)
$1\|\|F_l(C_{\max}(\mathcal{N}_1), L_{\max}(\mathcal{N}_2), \sum w_jC_j(\mathcal{N}_3))$	Strongly NP-hard	Baker and Smith (2003)
$1\|\|\sum_k(L_{\max}(\mathcal{N}_k))$	Ordinary NP-Hard	Cheng et al. (2008)
$1\|\|\sum_k(T_{\max}(\mathcal{N}_k))$	Ordinary NP-Hard	Cheng et al. (2008)
$1\|\|\sum_k(\max w_jC_j(\mathcal{N}_k))$	Strongly NP-Hard	Cheng et al. (2008)
$1\|\|\varepsilon(f_{\max}(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2004)
$1\|\|\varepsilon(\sum w_jC_j(\mathcal{N}_1)/C_{\max}(\mathcal{N}_2))$	Ordinary NP-Hard	Agnetis et al. (2004)
$1\|\|\varepsilon(\sum w_jC_j(\mathcal{N}_1)/L_{\max}(\mathcal{N}_2))$	Strongly NP-Hard	Ng et al. (2006)
$1\|\|\varepsilon(\sum w_jC_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2))$	Strongly NP-Hard	Ng et al. (2006)
$1\|\|\varepsilon(\sum C_j(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2004)
$1\|\|\varepsilon(\sum U_j(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2004)
$1\|\|\varepsilon(\sum U_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2004)
$1\|\|\varepsilon(\sum C_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2))$	Open*	
$1\|\|\varepsilon(\sum w_jC_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2))$	Ordinary NP-Hard	Agnetis et al. (2004)
$1\|\|\varepsilon(\sum C_j(\mathcal{N}_1)/\sum C_j(\mathcal{N}_2))$	Ordinary NP-Hard	Agnetis et al. (2004)
$1\|\|\#(f_{\max}(\mathcal{N}_1), f_{\max}(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2007)
$1\|\|\#(\sum w_jC_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2))$	Exponential	Agnetis et al. (2007)
$1\|\|\#(\sum C_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2007)
$1\|\|\#(\sum U_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2007)
$1\|\|\#(\sum U_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2007)
$1\|\|\#(\sum C_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2007)
$1\|\|\#(\sum w_jC_j(\mathcal{N}1), \sum U_j(\mathcal{N}_2))$	Polynomial	Agnetis et al. (2007)
$1\|\|\#(\sum C_j(\mathcal{N}_1), \sum C_j(\mathcal{N}_2))$	Exponential	Agnetis et al. (2007)

– with $f_{\max}$ a regular function of type $\max_j(f_j(C_j))$
– with g a non-regular function
(*) The problem is NP-Hard under high multiplicity encoding (Ng et al. 2006)

Fig. 2 Simple reductions between objective functions from single objective and bicriteria scheduling

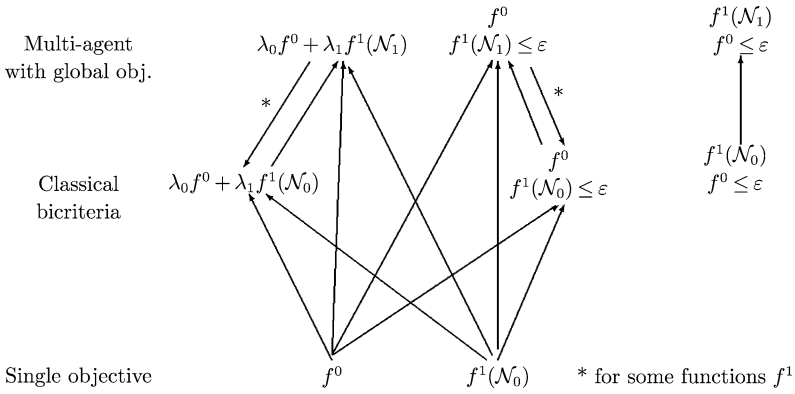
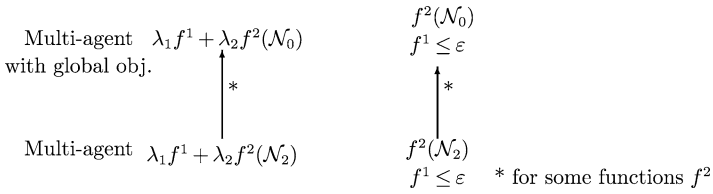


Fig. 3 Simple reductions between objective functions from multi-agent scheduling



3 Polynomially solvable problems

In this section, we present some polynomially solvable problems. Note that because the makespan is a constant, single-machine problems involving $C^0_{\max}$ as a global objective function are single objective problems. Thus, problems of type $1||F_\ell(f^1, C^0_{\max})$, $1||\varepsilon(f^1/C^0_{\max})$ and $1||\varepsilon(C^0_{\max}/f^1)$ have the same complexity as problems $1||f^1(\mathcal{N}_0), \forall f^1 \in \Gamma$.

Proposition 5 *The following problems can be solved in polynomial time*:

5.1 $1||F_\ell(L^0_{\max}, C^1_{\max})$ *and* $1||\varepsilon(L^0_{\max}/C^1_{\max})$
5.2 $1||F_\ell(L^0_{\max}, L^1_{\max})$ *and* $1||\varepsilon(L^0_{\max}/L^1_{\max})$

Proof 5.1 Suppose that jobs are numbered in EDD order. All the sequences $EDD(\mathcal{N}_1 \cup \{n_1+1, \ldots, j\})//EDD(\{j+$

Table 1 Complexity results in the case of a linear combination of criteria. Notice that this matrix is a priori not symmetric

$\lambda_1 f^1(\mathcal{N}_1)$	$\lambda f^0(\mathcal{N}_0)$							
	$C^0_{\max}$	$L^0_{\max}$	$\sum C^0_j$	$\sum T^0_j$	$\sum U^0_j$	$\sum w_j C^0_j$	$\sum w_j T^0_j$	$\sum w_j U^0_j$
$C^1_{\max}$	P	P	P	H	o	P	H	H
$L^1_{\max}$	P	P	P	H	o	H	H	H
$\sum C^1_j$	P	o	P	H	H	P	H	H
$\sum T^1_j$	H	H	H	H	H	H	H	H
$\sum U^1_j$	P	o	H	H	o	H	H	H
$\sum w_j C^1_j$	P	H	P	H	H	P	H	H
$\sum w_j T^1_j$	H	H	H	H	H	H	H	H
$\sum w_j U^1_j$	H	H	H	H	H	H	H	H

Table 2 Complexity results after simple reductions in the case of the ε-constraint

$f^1(\mathcal{N}_1) \leq \varepsilon_1$	Min $f^0(\mathcal{N}_0)$							
	$C^0_{\max}$	$L^0_{\max}$	$\sum C^0_j$	$\sum T^0_j$	$\sum U^0_j$	$\sum w_j C^0_j$	$\sum w_j T^0_j$	$\sum w_j U^0_j$
$C^1_{\max}$	P	P	P	H	o	H	H	H
$L^1_{\max}$	P	P	P	H	o	H	H	H
$\sum C^1_j$	P	o	H	H	H	H	H	H
$\sum T^1_j$	H	H	H	H	H	H	H	H
$\sum U^1_j$	P	o	H	H	o	H	H	H
$\sum w_j C^1_j$	P	H	H	H	H	H	H	H
$\sum w_j T^1_j$	H	H	H	H	H	H	H	H
$\sum w_j U^1_j$	H	H	H	H	H	H	H	H

$1, j+2, \ldots, n\}$) for all $j \in \{n_1+1, n_1+2, \ldots, n\}$ are tested. The best sequence with the linear combination gives an optimal solution for the function $F_\ell(L^0_{\max}, C^1_{\max})$ and the best sequence that satisfies $C_{\max}(\mathcal{N}_1) \leq \varepsilon$ gives an optimal solution for function $L^0_{\max}$. The problem can be solved in $O(n^2)$ time. The formal proof can be obtained by pairwise interchange arguments.

5.2 There exists an optimal solution such that the jobs of $\mathcal{N}_1$ and the jobs of $\mathcal{N} \setminus \mathcal{N}_1$ are sorted in EDD order. Of course, the global sequence obtained may not follow the EDD order.

Let us consider the problem $1||\varepsilon(L^0_{\max}/L^1_{\max})$. Let *edd* denote the global EDD sequence and let edd^1 denote the EDD sequence for the jobs of $\mathcal{N}_1$. An optimal algorithm is given in Table 3. This algorithm ensures, firstly, that the epsilon-constraint is verified. Then, the jobs are scheduled according to *edd* order to the greatest extent possible. This algorithm finds the optimal solution in $O(n \log n)$ time.

Now let us consider the $1||F_\ell(L^0_{\max}, L^1_{\max})$ problem. The polynomial time dynamic programming algorithm proposed by Yuan et al. (2005) in the multi-agent case can be applied to solve this problem. □

Notice that problems $1||F_\ell(L^0_{\max}, \sum C^1_j)$ and $1||\varepsilon(L^0_{\max}/\sum C^1_j)$ remain open.

Proposition 6 *The following problems can be solved in polynomial time*:

6.1 $1||F_\ell(\sum C^0_j, C^1_{\max})$ *and* $1||\varepsilon(\sum C^0_j/C^1_{\max})$
6.2 $1||F_\ell(\sum C^0_j, L^1_{\max})$ *and* $1||\varepsilon(\sum C^0_j/L^1_{\max})$

Proof 6.1 An optimal solution always exists with the jobs in $\mathcal{N}_1$ and the jobs in $\mathcal{N} \setminus \mathcal{N}_1$ being sequenced in the shortest processing time (SPT) order. Furthermore, for the makespan objective, only the completion time of the last job of $\mathcal{N}_1$ has to be considered. Thus, the jobs before the last job of $\mathcal{N}_1$ are sequenced in SPT order, whether or not they belong to $\mathcal{N}_1$. Let us suppose that the jobs of $\mathcal{N}_1$ in SPT are numbered as follows: $\{1, 2, \ldots, n_1\}$. Let us also suppose that the jobs of $\mathcal{N} \setminus \mathcal{N}_1$ are $\{n_1+1, n_1+2, \ldots, n\}$. We evaluate the sequences $SPT(\mathcal{N}_1 \cup \{n_1+1, \ldots, j\})//SPT(\{j+1, j+2, \ldots, n\})$ for all $j \in \{n_1+1, n_1+2, \ldots, n\}$, where $a//b$ stands for the concatenation of a and b. The best sequence is the optimal solution of the problem. This algorithm can be implemented in $O(n \log n)$ time.

6.2 Problem $1||\varepsilon(\sum C^0_j/C^1_{\max})$ is polynomial.

If $P_{\mathcal{N}1} = \sum_{J_j \in \mathcal{N}_1} p_j < \varepsilon$, there is no feasible solution. Otherwise, an optimal solution can be obtained by the following two-step algorithm:

1. determine the initial solution by ordering the jobs of $\mathcal{N}$ in SPT order
2. move the last jobs in $\mathcal{N}_1$ to the left in such a way that the new solution satisfies the ε-constraint.

The complexity is bounded by $O(n \log n)$.

6.2 Problem $1||F_\ell(\sum C^0_j, L^1_{\max})$ has the same complexity as problem $1||F_\ell(L^0_{\max}, \sum C^0_j)$, which is solvable in $O(n^2)$ (see Hoogeveen 1992 and Proposition 4). Problem $1||\varepsilon(\sum C^0_j/L^1_{\max})$ is equivalent to the $1|\tilde{d}_j|\sum C_j$ problem, which can be solved in polynomial time (Hoogeveen and Van de Velde 1995). □

Proposition 7 *The following problems can be solved in polynomial time*:

7.1 $1||F_\ell(\sum w_j C^0_j, \sum w'_j C^1_j)$
7.2 $1||F_\ell(\sum w_j C^0_j, C^1_{\max})$

Proof 7.1 We set $w''_j = \lambda_0 \times w_j + \lambda_1 \times w'_j$ and solve problem $1||\sum w''_j C^0_j$ (see Baker and Smith 2003). We deduce that problems $1||F_\ell(\sum w_j C^0_j, \sum C^1_j)$ and $1||F_\ell(\sum C^0_j, \sum w_j C^1_j)$ are also polynomially solvable.

7.2 Suppose that the jobs of $\mathcal{N}_1$ are numbered according to the weighted shortest processing time (WSPT) order. Apply the algorithm for 7.1 with $w'_j = 0, \forall j \neq n_1$ and $w'_{n_1} = 1$. □

Table 3 Algorithm for problem $1||\varepsilon(L^0_{\max}/L^1_{\max})$

For $i = 1$ to n_1 do
  $\tilde{d}_i = d_i + \varepsilon$
Endfor
Schedule the jobs of $\mathcal{N}_1$ in edd^1 order as late as possible and so that the deadline is respected.
$t = 0$
For $i = 1$ to n_1 do
  Consider $\mathcal{S}_i = \{j \in \mathcal{N} \setminus \mathcal{N}_1, d_j \leq d_i\}$, in *edd* order
  Schedule jobs in $\mathcal{S}_i$ from date t with preemption if necessary (without changing the start time of the jobs of $\mathcal{N}_1$)
  If there is an idle time before job i. Then
    Shift job i to the left.
    $t = C_i$
  Else
    $t =$ completion time of the last job of $\mathcal{S}_i$
  Endif
Endfor
For all the preempted jobs do
  In order to obtain a non-preemptive solution, shift the parts of the job to the right so that its completion time is not changed and the jobs of $\mathcal{N}_1$ complete earlier.
Endfor

4 Application of simple reductions from known NP-hard problems

Proposition 8 *The following problems are NP-hard*:

8.1 $1||F_\ell(f^0, f^1)$ *and* $1||\varepsilon(f^0/f^1)$, $\forall f^0 \in \{\sum T_j, \sum w_j T_j, \sum w_j U_j\}, \forall f^1 \in \Gamma$
8.2 $1||F_\ell(f^0, f^1)$ *and* $1||\varepsilon(f^0/f^1)$, $\forall f^0 \in \Gamma$, $\forall f^1 \in \{\sum T_j, \sum w_j T_j, \sum w_j U_j\}$
8.3 $1||F_\ell(L^0_{\max}, \sum w_j C^1_j)$ *and* $1||F_\ell(\sum w_j C^0_j, L^1_{\max})$
8.4 $1||F_\ell(\sum C^0_j, \sum U^1_j)$ *and* $1||\varepsilon(\sum C^0_j/\sum U^1_j)$
8.5 $1||F_\ell(\sum U^0_j, \sum C^1_j)$ *and* $1||\varepsilon(\sum U^0_j/\sum C^1_j)$
8.6 $1||F_\ell(\sum U^0_j, \sum w_j C^1_j)$ *and* $1||\varepsilon(\sum U^0_j/\sum w_j C^1_j)$

Proof 8.1 Deduced from prop. 1.1 and 1.2.

8.2 Deduced from prop. 1.1 and 1.3.

8.3 This item is true, because $1||F_\ell(L^0_{\max}, \sum w_j C^0_j)$ has been proved to be NP-hard in Hoogeveen (1992) (see Proposition 2.1).

8.4 Problem $1||Lex(\sum U^0_j, \sum C^0_j)$ is proved to be NP-hard in Huo et al. (2007) (with Lex being the lexicographic minimisation, i.e., Min $\sum C^0_j$ is subject to $\sum U^0_j$ being optimal). With an appropriate choice of weights, we can show that $1||Lex(\sum U^0_j, \sum C^0_j) \propto 1||F_\ell(\sum U^0_j, \sum C^0_j)$. As a consequence, this last problem is NP-hard. Thus, $1||F_\ell(\sum C^0_j, \sum U^1_j)$ is also NP-hard (see Proposition 2.1). We deduce that problem $1||F_\ell(\sum w_j C^0_j, \sum U^1_j)$ is also NP-hard. The second property is true because $1||\varepsilon(\sum C^0_j/\sum U^0_j)$ has been proved NP-hard in Nelson et al. (1986) (see Proposition 2.1).

8.5 For $1||F_\ell(\sum U^0_j, \sum C^1_j)$, see the proof 8.4.

8.6 This is an immediate reduction from 8.5. □

Proposition 9 *The following problems are NP-hard*:

9.1 $1||\varepsilon(L^0_{\max}/\sum w_j C^1_j)$
9.2 $1||\varepsilon(\sum w_j C^0_j/C^1_{\max})$ *and* $1||\varepsilon(\sum w_j C^0_j/L^1_{\max})$
9.3 $1||\varepsilon(\sum w_j C^0_j/\sum U^1_j)$

Proof 9.1 The decision version of this problem is to find a sequence so that $\sum w_j C^1_j \leq \varepsilon_1$ and $L^0_{\max} \leq \varepsilon_0$. The proposition comes because problem $1||\varepsilon(\sum w_j C^1_j/L^2_{\max})$ is proved NP-hard in Agnetis et al. (2004) (see Proposition 3.2).

9.2 This proposition has the same proof as 9.1.

9.3 The proof of this proposition has the same reasoning as the proof for 9.1 because problem $1||\varepsilon(\sum w_j C^1_j/\sum U^2_j)$ is proved to be NP-hard in Agnetis et al. (2004). □

5 The case of total completion time criteria

First we prove the NP-hardness of the $1||\varepsilon(\sum C^0_j/\sum C^1_j)$ problem in the case of one agent. Then, we provide a pseudo-polynomial time dynamic programming algorithm for problem $1||\varepsilon(\sum C^0_j/\sum C^1_j)$.

5.1 Establishing the complexity

In this section, we consider the epsilon-constraint problem for which the two objective functions are the total comple-

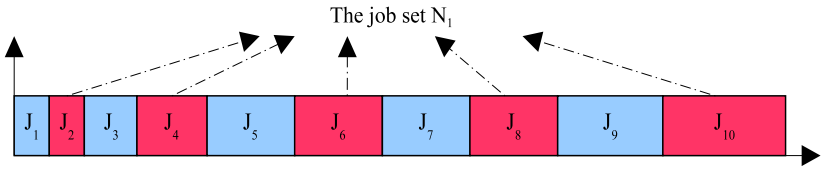


Fig. 4 Initial sequence with 10 jobs

tion time, denoted by $1||\varepsilon(\sum C_j^0/\sum C_j^1)$. No simple reduction can be used for deriving a complexity result. We show that the problem is NP-hard.

Proposition 10 *There is always an optimal solution that respects the following properties*:

10.1 *there is no idle time.*
10.2 *jobs in* $\mathcal{N}_1$ *follow the SPT order* (Shortest Processing Time *first*).
10.3 *jobs in* $(\mathcal{N} \setminus \mathcal{N}_1)$ *follow the SPT order.*
10.4 *if* $p_i \leq p_j$, *then* i *must be scheduled before* $j, \forall (i, j) \in \mathcal{N}_1 \times (\mathcal{N} \setminus \mathcal{N}_1)$.

Proof The first point is true because we consider regular criteria. The two next points are true because an interchange of jobs that do not follow the SPT order cannot decrease the solution's quality. The last point is true because otherwise, the permutation of i and j improves both $\sum C_j^0$ and $\sum C_j^1$. Note that point 4 is not true if $(i, j) \in (\mathcal{N} \setminus \mathcal{N}_1) \times \mathcal{N}_1$. □

Proposition 11 *Problem* $1||\varepsilon(\sum C_j^0/\sum C_j^1)$ *is NP-hard.*

Proof We define the problem PWDE (PARTITION with distinct elements) below. Note that this problem has been proved NP-hard in Huynh Tuong et al. (2009):

PWDE:
Data: Finite set $\mathcal{B}$ of t integer elements $a_1, a_2, \ldots, a_t$ with distinct sizes $(a_i \neq a_j, \forall i, j)$, $\sum_{i=1}^{t} a_i = 2C$.

Question: Is there a subset $\mathcal{B}_1$ of indices such that $\sum_{i \in \mathcal{B}_1} a_i = \sum_{i \in \{1,2,\ldots,t\} \setminus \mathcal{B}_1} a_i = C$?

We denote by 1mCC the decision version of problem $1||\varepsilon(\sum C_j^0/\sum C_j^1)$. This problem is defined by

1mCC:
Data: A set $\mathcal{N}_0$ of n jobs, a subset $\mathcal{N}_1 \subset \mathcal{N}_0$, processing times p_j for each job j, $1 \leq j \leq n$, two integer values Y_0 and Y_1

Question: Is there a one-machine schedule σ for $\mathcal{N}_0$ so that $\sum C_j^0 \leq Y_0$ and $\sum C_j^1 \leq Y_1$?

We must prove that PWDE $\propto$ 1mCC.

We consider an arbitrary instance of PWDE and we assume without loss of generality that $a_1 < \cdots < a_t$. We know that $\min_{1 \leq i \leq t-1} \frac{a_{i+1}}{a_i} > 1$. It is always possible to find α and K such that $1 < \alpha < \min_{1 \leq i \leq t-1} \frac{a_{i+1}}{a_i}$ and $\alpha K \in \mathbb{N}$ (if $\frac{a_{\ell+1}}{a_\ell} = \min_{1 \leq i \leq t-1} \frac{a_{i+1}}{a_i}$, take, for instance, $\alpha = \frac{a_{\ell+1}}{a_\ell+1}$ and $K = a_\ell + 1$ if $a_{\ell+1} \neq a_\ell + 1$ or take $\alpha = \frac{10 \times a_{\ell+1}}{10 \times a_\ell + 1}$ and $K = 10 \times a_\ell + 1$ otherwise).

Because of the definitions of α and K we have $Ka_i < \alpha K a_i < K a_{i+1} < \alpha K a_{i+1}$.

Let $\beta = \alpha - 1$, $\beta > 0$ and $X = K \sum_{i=1}^{t} (2(t - i + 1) + (2t - 2i + 1)\alpha) \times a_i$.

We define an instance of problem 1mCC as follows: $n = 2t$ and

- $p_{(2i-1)} = Ka_i$, $\forall i = 1, 2, \ldots, t$; $p_{(2i)} = \alpha K a_i$, $\forall i = 1, 2, \ldots, t$;
- $Y_1 = K(1 + \alpha)(\sum_{i=1}^{t} (t - i + 1) \times a_i) - KC$; $Y_0 = X + \beta K C$;
- $\mathcal{N}_1 = \{2, 4, 6, \ldots, 2t\}$.

We define an initial solution $S^0 = \{1, 2, 3, \ldots, 2t - 1, 2t\}$, i.e. the sequence where the jobs are sorted according to the SPT rule (see Fig. 4).

We have

$$\sum_{j=1}^{n} C_j(S^0) = Ka_1 + (Ka_1 + \alpha K a_1) + (Ka_1 + \alpha K a_1 + Ka_2) + (Ka_1 + \alpha K a_1 + Ka_2 + \alpha K a_2) + \cdots$$

$$\Rightarrow \sum_{j=1}^{n} C_j(S^0) = 2tKa_1 + (2t-1)\alpha K a_1 + (2t-2)Ka_2 + (2t-3)\alpha K a_2 + \cdots$$

$$\Rightarrow \sum_{j=1}^{n} C_j(S^0) = Ka_1\big(2t + (2t-1)\alpha\big) + Ka_2\big((2t-2) + (2t-3)\alpha\big) + \cdots$$

$$\Rightarrow \sum_{j=1}^{n} C_j(S^0) = K \sum_{i=1}^{t} \big(2(t-i+1) + (2t-2i+1)\alpha\big) \times a_i = X.$$

In the same way, we obtain

$$\sum_{j \in \mathcal{N}_1} C_j(S^0) = (Ka_1 + \alpha K a_1) + (Ka_1 + \alpha K a_1$$

$$+ Ka_2 + \alpha Ka_2) + \cdots$$

$$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^0) = t(Ka_1 + \alpha Ka_1) + (t-1)(Ka_2 + \alpha Ka_2) + \cdots$$

$$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^0) = t\big((1+\alpha)Ka_1\big) + (t-1) \times \big((1+\alpha)Ka_2\big) + \cdots$$

$$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^0) = K(1+\alpha)\big(ta_1 + (t-1)a_2 + \cdots\big)$$

$$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^0) = K(1+\alpha)\sum_{i=1}^{t}(t-i+1)a_i = Y_1 + KC.$$

Thus, this solution is not a feasible solution for problem 1mCC: $\sum_{j\in\mathcal{N}} C_j(S^0) \le Y$ but $\sum_{j\in\mathcal{N}_1} C_j(S^0) > Y_1$.

• Let us suppose that the answer to PWDE is 'yes'. We are going to propose a method for permuting consecutive jobs for decreasing $\sum C_j^1$ (this method will increase $\sum C_j^0$ at the same time). We consider the set of jobs $\mathcal{G} = \{j \in \mathcal{N}_0 \mid j = 2i \text{ with } i \in \mathcal{B}_1\}$. Note that $\mathcal{G} \subseteq \mathcal{N}_1$. We define the sequence S^1 by the permutation in S^0 of each job of $\mathcal{G}$ with its predecessor: $S^1[j] = S^0[j-1]$, $S^1[j-1] = S^0[j]$ for $j \in \mathcal{G}$ and $S^1[j] = S^0[j]$ for the other jobs.

We have to compute $\sum_{j\in\mathcal{N}_0} C_j(S^1)$ and $\sum_{j\in\mathcal{N}_1} C_j(S^1)$. We first compute these values after the permutation of only two jobs (sequence S').

$$\sum_{j\in\mathcal{N}_0} C_j(S') = \sum_{j\in\mathcal{N}_0} C_j\big(S^0\big) + (p_j - p_{j-1}).$$

Thus,

$$\sum_{j\in\mathcal{N}_0} C_j\big(S^1\big) = \sum_{j\in\mathcal{N}_0} C_j\big(S^0\big) + \sum_{j\in\mathcal{G}}(p_j - p_{j-1}) = \sum_{j\in\mathcal{N}_0} C_j\big(S^0\big) + \sum_{j\in\mathcal{G}}(\alpha Ka_{j/2} - Ka_{j/2}).$$

$$\Rightarrow \sum_{j\in\mathcal{N}_0} C_j\big(S^1\big) = \sum_{j\in\mathcal{N}_0} C_j\big(S^0\big) + \sum_{j\in\mathcal{G}}(\beta Ka_{j/2}) = \sum_{j\in\mathcal{N}_0} C_j\big(S^0\big) + \beta K\sum_{j\in\mathcal{G}} a_{j/2}$$

$$\Rightarrow \sum_{j\in\mathcal{N}_0} C_j\big(S^1\big) = \sum_{j\in\mathcal{N}_0} C_j\big(S^0\big) + \beta K \times C = X + \beta KC = Y.$$

Similarly,

$$\sum_{j\in\mathcal{N}_1} C_j(S') = \sum_{j\in\mathcal{N}_1} C_j\big(S^0\big) - p_{j-1}$$

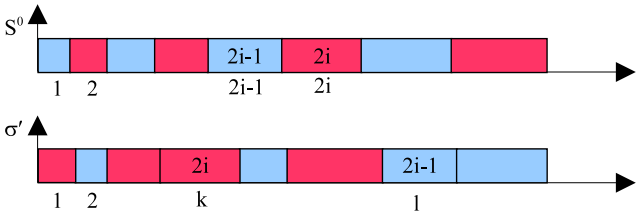


Fig. 5 Sequences S^0 and σ' and position of job $2i$

Thus,

$$\sum_{j\in\mathcal{N}_1} C_j\big(S^1\big) = \sum_{j\in\mathcal{N}_1} C_j\big(S^0\big) - \sum_{j\in\mathcal{G}} p_{j-1}.$$

$$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j\big(S^1\big) = \sum_{j\in\mathcal{N}_1} C_j\big(S^0\big) - \sum_{j\in\mathcal{G}} Ka_{j/2}$$

$$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j\big(S^1\big) = \sum_{j\in\mathcal{N}_1} C_j\big(S^0\big) - KC = Y_1 + KC - KC = Y_1.$$

Thus, S^1 is the sequence for which the answer to 1mCC is 'yes'.

• Now suppose that the answer to 1mCC is 'yes' for sequence σ. If σ does not respect the conditions of Proposition 10, then all the jobs are shifted to the left, the SPT rule is applied to the jobs of $\mathcal{N}_1$, and with respect to the jobs of $\mathcal{N}_0 \setminus \mathcal{N}_1$ and each time condition 10.4 occurs, jobs i and j are switched. A new sequence σ' is obtained, so that:

- $$\sum_{j\in\mathcal{N}_0} C_j(\sigma') \le \sum_{j\in\mathcal{N}_0} C_j(\sigma) \le Y \quad (1)$$
- $$\sum_{j\in\mathcal{N}_1} C_j(\sigma') \le \sum_{j\in\mathcal{N}_1} C_j(\sigma) \le Y_1 \quad (2)$$

• and σ' satisfies the conditions of Proposition 10.

We now compare σ' and S^0.

Let us consider the job number $2i$. This job is in position $2i$ in S^0 and it is in position k in σ'. Let us suppose that $k > 2i$. In this case, there is at least one job before $2i$ in σ' with a bigger processing time. This job cannot belong to $\mathcal{N}_1$, because the jobs of $\mathcal{N}_1$ in σ' are sorted according to SPT. Thus this job belongs to $\mathcal{N} \setminus \mathcal{N}_1$. However, this case is not possible because of condition 10.4. Therefore, $k \le 2i$. Similarly, we can show that job $2i-1$ is in position $2i-1$ in S^0 and that it is in position l in σ' with $l \ge 2i-1$. This case is illustrated in Fig. 5.

We define the set of jobs $\mathcal{H}_{2i} = \{j/(j \succ_{\sigma'} 2i) \wedge (p_j < p_{2i}) \wedge (j \in \mathcal{N}_0 \setminus \mathcal{N}_1)\}$. For instance, job $2i-1$ belongs to $\mathcal{H}_{2i}$. These jobs are the jobs of $\mathcal{N}_0 \setminus \mathcal{N}_1$ that precede job $2i$ in S^0.

We have $C_{2i}(S^0) = C_{2i}(\sigma') + \sum_{k\in\mathcal{H}_{2i}} p_k$ according to the definition of $\mathcal{H}_{2i}$.

$$\Rightarrow C_{2i}(\sigma') = C_{2i}(S^0) - \sum_{k\in\mathcal{H}_{2i}} p_k$$

$$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(\sigma') = \sum_{j\in\mathcal{N}_1} C_j(S^0) - \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} p_k$$

$$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(\sigma') = Y_1 + KC - \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} p_k. \tag{3}$$

Because (2) that $\sum_{j\in\mathcal{N}_1} C_j(\sigma') \le Y_1$, we have

$$Y_1 + KC - \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} p_k \le Y_1$$

$$\Rightarrow KC \le \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} p_k$$

$$\Rightarrow KC \le \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} Ka_{(k+1)/2}$$

$$\Rightarrow C \le \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} a_{(k+1)/2}. \tag{4}$$

Due to condition 10.4, from the initial solution S^0, the position of jobs $j \in \mathcal{N}_1$ in σ' would be unchanged or moved to the left. Similarly, the position of jobs $j \in \mathcal{N}_0 \setminus \mathcal{N}_1$ in σ' would be unchanged or moved to the right. The deviation of the completion time of a job $j \in \mathcal{N}_0 \setminus \mathcal{N}_1$ between two sequences σ' and S^0 is determined by the total processing times of the jobs of $\mathcal{N}_1$ which are scheduled after j in S^0, and they are also scheduled before j in σ'. For instance, in Fig. 5, the deviation of the completion time of job $2i-1$ between two sequences σ' and S^0 is at least equal to p_{2i}. More generally, we have

$$C_{2i-1}(\sigma') = C_{2i-1}(S^0) + \sum_{k\in\mathcal{N}_1 | 2i-1\in\mathcal{H}_k} p_k$$

$$\Rightarrow \sum_{j\in\mathcal{N}\setminus\mathcal{N}_1} C_j(\sigma') - \sum_{j\in\mathcal{N}\setminus\mathcal{N}_1} C_j(S^0) = \sum_{j\in\mathcal{N}\setminus\mathcal{N}_1}\sum_{k\in\mathcal{N}_1 | j\in\mathcal{H}_k} p_k$$

$$\Rightarrow \sum_{j\in\mathcal{N}_0\setminus\mathcal{N}_1} C_j(\sigma') - \sum_{j\in\mathcal{N}_0\setminus\mathcal{N}_1} C_j(S^0) = \sum_{k\in\mathcal{N}_1 | j\in\mathcal{H}_k}\sum_{j\in\mathcal{N}_0\setminus\mathcal{N}_1} p_k = \sum_{k\in\mathcal{N}_1}\sum_{j\in\mathcal{H}_k} p_k.$$

Thus, the deviation of the total completion times between two sequences σ' and S^0 is defined as follows:

$$\sum_{j\in\mathcal{N}_0} C_j(\sigma') - \sum_{j\in\mathcal{N}_0} C_j(S^0)$$

$$= \left(\sum_{j\in\mathcal{N}_1} C_j(\sigma') - \sum_{j\in\mathcal{N}_1} C_j(S^0)\right) + \left(\sum_{j\in\mathcal{N}_0\setminus\mathcal{N}_1} C_j(\sigma') - \sum_{j\in\mathcal{N}_0\setminus\mathcal{N}_1} C_j(S^0)\right).$$

Due to (3), we have

$$\sum_{j\in\mathcal{N}_1} C_j(\sigma') - \sum_{j\in\mathcal{N}_1} C_j(S^0) = \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} p_k$$

$$\Rightarrow \sum_{j\in\mathcal{N}_0} C_j(\sigma') - \sum_{j\in\mathcal{N}_0} C_j(S^0) = \sum_{k\in\mathcal{N}_1}\sum_{j\in\mathcal{H}_k} p_k - \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} p_k$$

$$\Rightarrow \sum_{j\in\mathcal{N}_0} C_j(\sigma') - \sum_{j\in\mathcal{N}_0} C_j(S^0) = \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} (p_j - p_k).$$

Because $p_j > p_k$ where $j \in \mathcal{N}_1, k \in \mathcal{H}_j$, we have $p_j \ge p_{k+1}$ with $j, k+1 \in \mathcal{N}_1$ and $k \in \mathcal{H}_j$

$$\Rightarrow \sum_{j\in\mathcal{N}_0} C_j(\sigma') - \sum_{j\in\mathcal{N}_0} C_j(S^0) \ge \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} (p_{k+1} - p_k)$$

$$= \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} (\alpha K a_{(k+1)/2} - K a_{(k+1)/2})$$

$$\Rightarrow \sum_{j\in\mathcal{N}_0} C_j(\sigma') - \sum_{j\in\mathcal{N}_0} C_j(S^0) \ge \beta K \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} a_{(k+1)/2}$$

$$\Rightarrow \sum_{j\in\mathcal{N}_0} C_j(\sigma') \ge \sum_{j\in\mathcal{N}_0} C_j(S^0) + \beta K \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} a_{(k+1)/2}. \tag{5}$$

According to (2) and because $\sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} a_{(k+1)/2} \ge C$, we have the following:

$$\sum_{j\in\mathcal{N}_0} C_j(\sigma') \ge \sum_{j\in\mathcal{N}_0} C_j(S^0) + \beta KC = Y. \tag{6}$$

Consequently, thanks to (1) and (6), we deduce that $\sum_{j\in\mathcal{N}_0} C_j(\sigma') = Y$.

In other words, all inequalities (4), (5) should become equalities:

$$\begin{cases} \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} a_{(k+1)/2} = C, \\ p_j = p_{k+1} \quad \text{where } j \in \mathcal{N}_1, k \in \mathcal{H}_j. \end{cases}$$

Let us recall that the processing times of the jobs are all different. Hence, either $p_j = p_{k+1}$ (i.e. $|\mathcal{H}_j| = 1$) or $|\mathcal{H}_j| = 0$ where $j \in \mathcal{N}_1, k \in \mathcal{H}_j$.

$$\Rightarrow |\mathcal{H}_j| \le 1, \quad \forall j \in \mathcal{N}_1$$

$\Rightarrow$ The equality $\sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} a_{(k+1)/2} = C$

defines the subset B_1 of PWDE.

Consequently, the answer for the question of the PWDE problem is 'yes' (i.e., jobs j with $|\mathcal{H}_j| = 1$ give a subset B_1 of PWDE). □

We deduce that problems $1||\varepsilon(\sum C_j^0/\sum w_j C_j^1)$, $1||\varepsilon(\sum w_j C_j^0/\sum C_j^1)$ and $1||\varepsilon(\sum w_j C_j^0/\sum w_j C_j^1)$ are also NP-hard.

5.2 A dynamic programming algorithm for the $1||\varepsilon(\sum C_j^0/\sum C_j^1)$

We have to both minimise $\sum C_j^0$ and respect the constraint that $\sum C_j^1 \leq \varepsilon_1$. We assume that the jobs in $\mathcal{N}_1$ are numbered from 1 to $n_1 = |\mathcal{N}_1|$ with $p_1 \leq p_2 \leq \cdots \leq p_{n_1}$ and that the jobs in $\mathcal{N}\setminus\mathcal{N}_1$ are numbered from n_1+1 to n with $p_{n_1+1} \leq p_{n_1+2} \leq \cdots \leq p_n$. We introduce the following notations: P is the sum of the processing times of all the jobs, $P_{(i,j)} = \sum_{1\leq k\leq i} p_k + \sum_{n_1+1\leq k\leq j} p_k$. We denote by $F(i,j,q) = \sum C_j^0$ the minimum cost that must arise when the jobs $\{1,2,\ldots,i\} \in \mathcal{N}_1$ and the jobs $\{n_1+1, n_1+2,\ldots,j\} \in \mathcal{N}\setminus\mathcal{N}_1$ are scheduled. q corresponds to $\sum C_j^1$.

The general recursive relation is

$$F(0,n_1,0) = 0,$$

$$F(i,j,q) = +\infty, \quad \forall i > n_1, \forall j \leq n_1, \forall q,$$

$$F(i,j,q) = +\infty, \quad \forall i \leq n_1, \forall j \in \{n_1, n_1+1,\ldots,n\},$$

$$\forall q < 0 \vee q > \varepsilon_1,$$

$$F(i,j,q) = \min\left\{\begin{array}{l} F(i-1,j,q-P_{(i,j)}) + P_{(i,j)}, \\ F(i,j-1,q) + P_{(i,j)} \end{array}\right\}$$

$$\left(\begin{array}{l} \forall i \in \{1,\ldots,n_1\} \\ \forall j \in \{n_1+1,\ldots,n\} \\ \forall 0 \leq q \leq \varepsilon_1 \end{array}\right).$$

The optimal solution is given by $\min_{0\leq q\leq\varepsilon_1} F(n_1,n,q)$. The execution time of this algorithm is in $O(n_1(n-n_1)\varepsilon_1)$.

Proposition 12 *An optimal solution to the problem* $1||\varepsilon(\sum C_j^0/\sum C_j^1)$ *can be determined in* $O(n^2\varepsilon_1)$ *time.*

6 Conclusion

In this paper, we consider a new family of scheduling problems at the frontier of multi-agent and multicriteria scheduling. Subsets of jobs are in competition with the whole set of jobs for the use of resources and a compromise solution has to be found. We consider the problem of scheduling independent jobs on a single machine, without additional constraints and the objective functions are of two types: the ε-constraint approach and a linear combination of the criteria. We notice that these results can be extended to the case of goal programming and enumerative approaches.

Some simple reductions from multicriteria scheduling problems are established, and polynomially solvable problems and NP-hard problems are identified. A pseudo-polynomial time dynamic programming algorithm is proposed for solving the case of total completion time.

This category of problems may have a lot of practical applications, and it leads to a wide area of research problems. While some complexity results remain open, some approximation schemes can be constructed for these problems as well as exact and approximated heuristic algorithms. Considering more than one agent is also a challenging perspective, as the problems combine the aspects of multicriteria scheduling and multi-agent scheduling problems, making the problems more complicated to handle. Computing the size of the Pareto set is also an interesting problem.

Acknowledgements This work has been supported by the project ANR-08-BLAN-0331-01, funded by the "National Research Agency" (ANR).

References

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*, 191–217.

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research*, *52*, 229–242.

Agnetis, A., Pacciarelli, D., & Pacifici, A. (2007). Multi-agent single machine scheduling. *Annals of Operations Research*, *150*, 3–15.

Baker, K., & Smith, J. C. (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.

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

Cheng, T. C. E., Ng, C. T., & Yuan, J. J. (2008). Multi-agent scheduling on a single machine with max-form criteria. *European Journal of Operational Research*, *188*, 603–609.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. San Francisco: W.H. Freeman.

Graham, R. L., Lawler, E. L., 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.

Hall, N. G., & Potts, C. N. (2004). Rescheduling for new orders. *Operations Research*, *52*(3), 440–453 +496.

Hoogeveen, H. (1992). *Single machine bicriteria scheduling*. PhD Thesis, Amsterdam.

Hoogeveen, H. (2005). Multicriteria scheduling. *European Journal of Operational Research*, *167*, 592–623.

Hoogeveen, H., & Van de Velde, S. (1995). Minimizing completion time and maximum cost simultaneously is solvable in polynomial time. *Operations Research Letters*, *17*, 205–208.

Huo, Y., Leung, J. Y.-T., & Zhao, H. (2007). Complexity of two dual criteria scheduling problems. *Operations Research Letters*, *35*, 211–220.

Huynh Tuong, N., Soukhal, A., & Billaut, J.-C. (2009). *Complexity of partition problem with distinct elements* (Research report num. 295), University of Tours, France, March.

Mocquillon, C., Lenté, C., & T'Kindt, V. (2006). Solution of a multicriteria shampoo production problem. In *IEEE international conference on service systems and service management (IEEE-SSSM'06)*, Troyes (France) (pp. 907–911).

Nelson, R. T., Sarin, R. K., & Daniels, R. L. (1986). Scheduling with multiple performance measures: the one-machine case. *Management Science*, *32*(4), 464–479.

Ng, C. T., Cheng, T. C. E., & Yuan, J. J. (2006). A note on the complexity of the problem of two-agent scheduling on a single machine. *Journal of Combinatorial Optimization*, *12*, 387–394.

Pessan, C., Bouquard, J.-L., & Neron, E. (2008a). An unrelated parallel machines model for an industrial production resetting problem. *European Journal of Industrial Engineering*, *2*(2), 153–171.

Pessan, C., Bouquard, J.-L., & Neron, E. (2008b). Genetic branch-and-bound or exact genetic algorithm. In *Lecture notes in computer science* (Vol. 4926, pp. 136–147). Berlin: Springer.

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

Wu, S. D., Storer, R. H., & Chang, P.-C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers & Operations Research*, *20*(1), 1–14.

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