New scheduling problems with interfering and independent jobs

Nguyen Huynh Tuong, Ameur Soukhal, Jean-Charles Billaut

Submitted to *Journal of Scheduling*
manuscript (Please, provide the mansucript number!)

New scheduling problems with interfering and independent jobs

N. HUYNH TUONG, A. SOUKHAL and J-C. BILLAUT
Laboratoire d'Informatique, Université François Rabelais, 64 Avenue Jean Portalis, 37200 Tours (France)
nguyen.huynh@etu.univ-tours.fr,{ameur.soukhal, jean-charles.billaut}@univ-tours.fr

We consider the problems of scheduling independent jobs, when a subset of jobs has its own objective function to minimize. The performance of this subset of jobs is in competition with the performance of the whole set of jobs and compromise solutions have to be found. Such a problem arises for some practical applications like ball bearing production problems. This new scheduling problem is positioned within the literature and the differences with the problems with competing agents or with interfering job set problems are presented. Classical and regular scheduling objective functions are considered and ε-constraint approach and linear combination of criteria approach are used for finding compromise solutions. The study focus on single machine and identical parallel machine environments and for each environment, the complexity of several problems is established and some dynamic programming algorithms are proposed.

Key words: scheduling, independent jobs, interfering jobs, 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 as equivalent and that 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 (makespan), the total flow time of jobs or a measure related to the tardiness like maximum tardiness, total number of tardy jobs, etc. Introducing distinctions between jobs is generally done by the means of weights. However, in this case the same measure is still applied to all the jobs in order to quantify the quality of a schedule. For instance it can be the total weighted completion time, the total weighted tardiness or the weighted number of tardy jobs.

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, both the mean flow time (equivalent to the total completion time) and the respect of due dates can be of similar importance for the decision maker. In such cases, more than one objective function is defined and the scheduling problem enters the field of multicriteria scheduling T'kindt and Billaut (2006) and Hoogeveen (2005). But again, each objective function is applied to the whole set of jobs.

In some cases, it may happen that the jobs are not equivalent and that applying the same measure to all the jobs is not relevant. For instance, it is possible to consider a workshop where jobs have the following particularities: some jobs may have a soft due date with allowed tardiness (to be minimized); whereas some other jobs may have hard due dates (that must be respected) and other jobs may have no due date (production for stock). For the first type of job, the decision maker wants to minimize the maximum delay, for the second type he imposes that there must be no delayed jobs and for the last type of job he wants to minimize 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 is a multicriteria scheduling problem where a new type of compromise has to be obtained. These problems are called in the literature "*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, each subset having its own objective function to optimize.

We consider in this paper a different problem where the performance of the whole set of jobs has to be minimized, subject to a given performance for a subset of jobs on another objective function. 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,b)). The objective is related to the minimization of the flow time criterion (maximizing the number of items produced) and concerns the whole set of jobs, denoted by $\mathcal{N}$. Generally, the jobs to produce daily exceed the production capacity. In order to impose the production of the remaining jobs (say $\mathcal{N}_1 \subset \mathcal{N}$) during the next day, another performance measure has to be applied, which is the minimization of the number of tardy jobs (or any other due date related measure).

The measure concerning $\mathcal{N}$ is the total completion time minimization but the number of tardy jobs among $\mathcal{N}_1$ cannot exceed a given threshold. Another example can be found in shampoo packing systems (Mocquillon *et al.* (2006)). Shampoo are delivered daily and stored in dedicated storage area with limited capacity. The problem is to pack shampoo of different types into bottles. A global objective is to maximize the production (reducing the setup times). At the same time, future deliveries are known in advance. Thus, each type of product has to be produced daily so that its quantity never exceeds its storage area. This is a typical problem where the global objective concerns all the products and where the subset of products are evaluated with another objective.

The rest of the paper is organized as follows. In Section 2 the problem is defined and the notations are introduced. The state-of-the-art survey is presented and the interest of the study in comparison with the other models of interfering jobs is proved. Section 3 deals with the single machine problems: some polynomially solvable cases and some NP-hard problems are identified. The section terminates with some open problems. Section 4 deals with parallel machine problems. Some complexity results are given and a general dynamic programming formulation offering optimal problem solutions is given. This general DP algorithm is presented for some two-parallel machine problems. The paper is concluded in Section 5.

2. Preliminaries

2.1. Problem definition and notations

A set $\mathcal{N}$ of n jobs has to be scheduled on a single machine or on m identical parallel machines ($m \geq 2$). We assume that all the jobs are available at time 0; preemption is not allowed; processing times are known, deterministic and integer, p_j denotes the processing time of job j, $1 \leq j \leq n$; machines are always available and can process only one job at a time.

$\mathcal{N}_1$ denotes a subset of $\mathcal{N}$. We denote by n_1 the number of jobs in $\mathcal{N}_1$. These jobs are numbered from 1 to n_1. The remaining jobs of $\mathcal{N}$ are numbered from $n_1 + 1$ to n. One objective function is associated to $\mathcal{N}$ and the other one is associated to $\mathcal{N}_1$. We denote by C_j the completion time of job j. $\sum C_j$ is the total flow time and $\sum w_j C_j$ is the total weighted flow time. $C_{\max}$ denotes the maximum completion time (makespan) and $L_{\max}$ denotes the maximum lateness $L_{max} = \max_{1 \le j \le n}(C_j - d_j)$. In the following, U_j is equal to 1 if job j is tardy, and 0 otherwise. $\sum U_j$ denotes the number of tardy jobs and $\sum w_j U_j$ the weighted number of tardy jobs.

We denote by $Z(\mathcal{S})$ the measure Z applied to the set $\mathcal{S}$ of jobs. Referring to the three-field notation for multicriteria scheduling in T'kindt and Billaut (2006) we consider the following two types of objective functions:

- $\varepsilon\big(Z_1(\mathcal{S})/Z_2(\mathcal{S}')\big)$ denotes the ε-constraint approach, i.e. the minimization of $Z_1(\mathcal{S})$ subject to the constraint that $Z_2(\mathcal{S}') \le \varepsilon$ (with $(\mathcal{S}, \mathcal{S}') \in \{(\mathcal{N}, \mathcal{N}_1), (\mathcal{N}_1, \mathcal{N})\}$). In the following, we consider that the objective is to minimize an objective function on $\mathcal{N}$ subject to a bound on the objective function on $\mathcal{N}_1$: $\varepsilon\big(Z_1(\mathcal{N})/Z_2(\mathcal{N}_1)\big)$. Notice that if the two objective functions are bounded, we are in the case of goal programming approaches. The decision problem associated to the ε-constraint version of an optimization problem and the decision problem associated to the goal programming version are the same.
- $F_\ell\big(Z_1(\mathcal{S}), Z_2(\mathcal{S}')\big)$ denotes the linear combination of $Z_1(\mathcal{S})$ and $Z_2(\mathcal{S}')$.

2.2. State-of-the-art survey

The literature contains very few results on these scheduling problems. In HuynhTuong *et al.* (2008) the authors consider a two-machine flow shop with interfering jobs. The objective is the minimization of the makespan subject to the constraint that the completion time of the last job of $\mathcal{N}_1$ does not exceed a given bound. The problems are denoted by $F2||\varepsilon\big(C_{\max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ and $F2||\varepsilon\big(C_{\max}(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$. The problem is proved ordinary NP-hard and the authors propose a pseudo-polynomial time dynamic programming algorithm for the determination of a non dominated solution. Notice that this problem is more general than the multi-agent scheduling problem presented in Agnetis *et al.* (2004) and that their algorithm can also solve this problem. Scheduling interfering jobs on parallel machines is presented in Soukhal *et al.* (2008). In this paper, the authors minimize the total completion time of the jobs of $\mathcal{N}$ subject to a bound on the total completion time of the jobs of $\mathcal{N}_1$. The authors show that the problem is ordinary NP-hard and propose a pseudo-polynomial time dynamic programming algorithm for finding a non-dominated solution. In Agnetis *et al.* (2000), the authors consider a two-job job shop scheduling problem with two subsets of jobs $\mathcal{N}_1$ and $\mathcal{N}_2$, one job per subset ($\mathcal{N}_1 \cup \mathcal{N}_2 = \mathcal{N}$, $\mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$ in the following). They give a polynomial time algorithm for finding compromise schedules to simplify negotiations between agents. In Agnetis *et al.* (2004) the authors 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 minimization of an objective function for one subset of jobs subject to a bound for the other subset of jobs. 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). The authors consider several regular objective functions ($C_{max}, \sum w_j C_j, L_{max}$) and propose an algorithm for the minimization of a linear combination of the objective functions. Complexity results are given and some polynomially solvable cases are identified. Yuan *et al.* (2005) propose some complementary results on these problems. Figure 1 summarizes the results presented in Baker and Smith (2003) and Yuan *et al.* (2005).

In Cheng *et al.* (2006) the authors consider a single machine problem with m disjoint subsets of jobs $\mathcal{N}_1, \ldots, \mathcal{N}_m$ ($\cup_{i=1}^m \mathcal{N}_i = \mathcal{N}$). To each job is associated a deadline. Each subset is measured by the total number of tardy jobs. The authors consider a goal programming problem that can be denoted by $1||GP\big(\sum w_j U_j(\mathcal{N}_1), \ldots, \sum w_j U_j(\mathcal{N}_m)\big)$ or $1|\sum w_j U_j(\mathcal{N}_1) \leq \varepsilon_1, \ldots, \sum w_j U_j(\mathcal{N}_m) \leq \varepsilon_m|-$. The authors prove that the problem is strongly NP-hard. When the number of agents (m) is fixed they show that the problem can be solved in pseudo-polynomial time and give a fully polynomial approximation scheme. If additionally the weights are equal to 1, the problem can be solved in polynomial time. In Cheng *et al.* (2008) the authors consider the single machine multi-agent scheduling problem with m objective functions of type min-max. The authors consider the same goal programming approach and prove that the feasibility problem can be solved in polynomial time, even if jobs are subject to precedence constraints. The authors show that the problems $1||\sum_{i=1}^m \big(L_{\max}(\mathcal{N}_i)\big)$, $1||\sum_{i=1}^m \big(T_{\max}(\mathcal{N}_i)\big)$ and $1||\sum_{i=1}^m \big(\sum w_j C_j(\mathcal{N}_i)\big)$ are NP-hard. Some polynomially solvable cases are identified. In Agnetis *et al.* (2007), the authors consider the single machine two-agent scheduling problems indicated in Figure 1. Two approaches are considered: (1) the "decision problem" to find a solution such that all the criteria are bounded (goal programming approach denoted by GP) and (2) the "Pareto-optimization problem" where the aim is to find the set of all non-dominated solutions (denoted by "#" in T'kindt and Billaut (2006)). Some results are also given for some single machine multi-agent scheduling problems.

Problem	Complexity	Reference
$1\|\|F_l\big(C_{\max}(\mathcal{N}_1), C_{\max}(\mathcal{N}_2)\big)$	Polynomial	Baker and Smith (2003)
$1\|\|F_l\big(L_{\max}(\mathcal{N}_1), L_{\max}(\mathcal{N}_2)\big)$	Polynomial	Baker and Smith (2003),Yuan *et al.* (2005)
$1\|\|F_l\big(\sum w_jC_j(\mathcal{N}_1), \sum w_jC_j(\mathcal{N}_2)\big)$	Polynomial	Baker and Smith (2003)
$1\|\|F_l\big(C_{\max}(\mathcal{N}_1), L_{max}(\mathcal{N}_2)\big)$	Polynomial	Baker and Smith (2003)
$1\|\|F_l\big(C_{\max}(\mathcal{N}_1), \sum w_jC_j(\mathcal{N}_2)\big)$	Polynomial	Baker and Smith (2003)
$1\|\|F_l\big(\sum C_j(\mathcal{N}_1), L_{\max}(\mathcal{N}_2)\big)$	Polynomial	Yuan *et al.* (2005)
$1\|\|F_l\big(\sum w_jC_j(\mathcal{N}_1), L_{\max}(\mathcal{N}_2)\big)$	Strongly NP-hard	Baker and Smith (2003)
$1\|\|F_l\big(C_{\max}(\mathcal{N}_1), L_{\max}(\mathcal{N}_2), \sum w_jC_j(\mathcal{N}_3)\big)$	Strongly NP-hard	Baker and Smith (2003)
$1\|\|\sum_i\big(L_{\max}(\mathcal{N}_i)\big)$	Binary NP-Hard	Cheng *et al.* (2008)
$1\|\|\sum_i\big(T_{\max}(\mathcal{N}_i)\big)$	Binary NP-Hard	Cheng *et al.* (2008)
$1\|\|\sum_i\big(\max w_jC_j(\mathcal{N}_i)\big)$	Strongly NP-Hard	Cheng *et al.* (2008)
$1\|\|\varepsilon\big(f_{\max}(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2004)
$1\|\|\varepsilon\big(\sum w_jC_j(\mathcal{N}_1)/C_{\max}(\mathcal{N}_2)\big)$	Binary NP-Hard	Agnetis *et al.* (2004)
$1\|\|\varepsilon\big(\sum w_jC_j(\mathcal{N}_1)/L_{\max}(\mathcal{N}_2)\big)$	Strongly NP-Hard	Ng *et al.* (2006)
$1\|\|\varepsilon\big(\sum w_jC_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2)\big)$	Strongly NP-Hard	Ng *et al.* (2006)
$1\|\|\varepsilon\big(\sum C_j(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2004)
$1\|\|\varepsilon\big(\sum U_j(\mathcal{N}_1)/f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2004)
$1\|\|\varepsilon\big(\sum U_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2004)
$1\|\|\varepsilon\big(\sum C_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2)\big)$	Open*	
$1\|\|\varepsilon\big(\sum w_jC_j(\mathcal{N}_1)/\sum U_j(\mathcal{N}_2)\big)$	Binary NP-Hard	Agnetis *et al.* (2004)
$1\|\|\varepsilon\big(\sum C_j(\mathcal{N}_1)/\sum C_j(\mathcal{N}_2)\big)$	Binary NP-Hard	Agnetis *et al.* (2004)
$F2\|\|\varepsilon\big(C\max(\mathcal{N}_1)/C\max(\mathcal{N}_2)\big)$	Binary NP-Hard	Agnetis *et al.* (2004),HuynhTuong *et al.* (2008)
$O2\|\|\varepsilon\big(C\max(\mathcal{N}_1)/C\max(\mathcal{N}_2)\big)$	Binary NP-Hard	Agnetis *et al.* (2004)
$1\|\|GP\big(f_{\max}(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|GP\big(\sum w_jC_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Binary NP-hard	Agnetis *et al.* (2007)
$1\|\|GP\big(\sum C_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|GP\big(\sum U_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|GP\big(\sum U_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|GP\big(\sum C_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Open	
$1\|\|GP\big(\sum w_jC_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Binary NP-hard	Agnetis *et al.* (2007)
$1\|\|GP\big(\sum C_j(\mathcal{N}_1), \sum C_j(\mathcal{N}_2)\big)$	Binary NP-hard	Agnetis *et al.* (2007)
$1\|\|GP\big(\sum U_j(\mathcal{N}_1), \ldots, \sum U_j(\mathcal{N}_m)\big)$	Polynomial	Cheng *et al.* (2006)
$1\|\|GP\big(\sum w_jU_j(\mathcal{N}_1), \ldots, \sum w_jU_j(\mathcal{N}_m)\big)$	Binary NP-hard	Cheng *et al.* (2006)
$1\|\|GP\big(\sum w_jU_j(\mathcal{N}_1), \ldots, \sum w_jU_j(\mathcal{N}_i), \ldots\big)$	Strongly NP-hard	Cheng *et al.* (2006)
$1\|\|\#\big(f_{\max}(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|\#\big(\sum w_jC_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Exponential	Agnetis *et al.* (2007)
$1\|\|\#\big(\sum C_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|\#\big(\sum U_j(\mathcal{N}_1), f_{\max}(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|\#\big(\sum U_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|\#\big(\sum C_j(\mathcal{N}_1), \sum U_j(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|\#\big(\sum w_jC_j(\mathcal{N}1), \sum U_j(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2007)
$1\|\|\#\big(\sum C_j(\mathcal{N}_1), \sum C_j(\mathcal{N}_2)\big)$	Exponential	Agnetis *et al.* (2007)
$J\|n=2\|\#\big(g(\mathcal{N}_1), g(\mathcal{N}_2)\big)$	Polynomial	Agnetis *et al.* (2000)

- 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)

Figure 1 Some complexity results on multi-agent scheduling problems

2.3. Interest of the study

We denote by F the optimization problem with two objective functions $f_1(\mathcal{N}_1)$ and $f_2(\mathcal{N}_2)$ concerning two disjoint job sets $\mathcal{N}_1$ and $\mathcal{N}_2$ ($\mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$ and $\mathcal{N}_1 \cup \mathcal{N}_2 = \mathcal{N}$). We denote by G the optimization problem with two objective functions $g_1(\mathcal{N})$ on the whole set of jobs and $g_2(\mathcal{N}_1)$ on $\mathcal{N}_1$ ($\mathcal{N}_1 \subset \mathcal{N}$). If f_1 is of the type min-sum it is denoted by sf_1 and by mf_1 if it is of the type min-max (same notation for f_2, g_1 and g_2). We have $sf_1 \in \{\sum C_j, \sum w_jC_j, \sum T_j, \sum w_jT_j, \sum U_j, \sum w_jU_j\}$ and $mf_1 \in \{C_{max}, L_{max}, T_{max}\}$ (the same for f_2, g_1 and g_2).

We are going to explain the difference between problems F and G. We distinguish in this section two possible approaches: the minimization of a linear combination and a goal programming approach (same decision problem as for the ε-constraint approach).

Notice that we have $sg_1(\mathcal{N}) = sg_1(\mathcal{N}_1) + sg_1(\mathcal{N}_2)$ and $mg_1(\mathcal{N}) = \max(mg_1(\mathcal{N}_1), mg_1(\mathcal{N}_2))$.

We have the following (simple) preliminary results: if $f_1 \not\equiv g_2$ ('f_1 not similar to g_2', i.e. $sf_1 \neq sg_2$ if they are of type min-sum and $mf_1 \neq mg_2$ if they are of type min-max) or $f_2 \not\equiv g_1$ then problems F and G are not comparable. Furthermore, if f_2 is of the type mf_2, then F and G are not comparable. In the following, we assume that $f_1 \equiv g_2$ ($sf_1 = sg_2$ or $mf_1 = mg_2$) and $sf_2 = sg_1$.

There remain only two cases to consider:

1. $sf_1 = sg_2$ and $sf_2 = sg_1$
2. $mf_1 = mg_2$ and $sf_2 = sg_1$

We distinguish the linear combination of criteria and the goal programming approach.

- Case of a linear combination approach for $sf_1 = sg_2$ and $sf_2 = sg_1$.

The objective function of problem F is Min $Z = \alpha \sum f_1(\mathcal{N}_1) + \beta \sum f_2(\mathcal{N}_2)$ and for problem G Min $Z' = \alpha' \sum g_1(\mathcal{N}) + \beta' \sum g_2(\mathcal{N}_1) = \alpha' \sum f_2(\mathcal{N}) + \beta' \sum f_1(\mathcal{N}_1)$. Thus, we have $Z' = \alpha' \sum f_2(\mathcal{N}_2) + \alpha' \sum f_2(\mathcal{N}_1) + \beta' \sum f_1(\mathcal{N}_1)$. If sf_1 and sf_2 are not identical, it is not possible to compare the objective functions and problems are not comparable. However, if $sf_1 = sf_2$ (and thus $= sg_2 = sg_1$), then $Z' = \alpha' \sum f_1(\mathcal{N}_2) + (\alpha' + \beta') \sum f_1(\mathcal{N}_1)$. In this particular case, the problems are equivalent, a procedure for solving F or G can solve the other problem.

- Case of a goal programming approach for $sf_1 = sg_2$ and $sf_2 = sg_1$.

Problem F is to find a solution S such that $\sum f_1(\mathcal{N}_1) \leq \varepsilon_1$ and $\sum f_2(\mathcal{N}_2) \leq \varepsilon_2$. Problem G is to find a solution S' such that $\sum f_2(\mathcal{N}_1) + \sum f_2(\mathcal{N}_2) \leq \varepsilon'_1$ and $\sum f_1(\mathcal{N}_1) \leq \varepsilon'_2$. We show that the problems are never comparable. If we have $sf_2 \neq sf_1$, then $\sum f_2(\mathcal{N}_1) + \sum f_2(\mathcal{N}_2)$ is related to $\sum f_2(\mathcal{N}_1)$, which is not taken into account in problem F.

Then, F and G are not comparable. We consider now the case where $sf_2 = sf_1 = sg_2 = sg_1$. It is clear that if $\varepsilon_1 \neq \varepsilon'_2$ or $\varepsilon'_1 \neq \varepsilon_1 + \varepsilon_2$, then F and G are not comparable. When $\varepsilon_1 = \varepsilon'_2$ and $\varepsilon'_1 = \varepsilon_1 + \varepsilon_2$, if S is a solution to problem F, it is also a solution to problem G. But the reverse is not true and is proved by the following instances:

—if $sf = \sum_j C_j$: $\mathcal{N} = 1,2,3$, $\mathcal{N}_1 = 1,2$, $\mathcal{N}_2 = 3$, $p_1 = 1$, $p_2 = 2$, $p_3 = 3$, $\varepsilon_1 = 8$, $\varepsilon_2 = 3$ ($\varepsilon_1 + \varepsilon_2 = 11$). G has a feasible solution $S' = (1,3,2)$ with $\sum_j C_j(\mathcal{N}_1) = 7$ and $\sum_j C_j(\mathcal{N}) = 11$. However, F has no feasible solution. As a consequence, the reverse is not true for any $sf \in \{\sum w_j C_j, \sum T_j, \sum w_j T_j\}$.

—if $sf = \sum_j U_j$: $\mathcal{N} = 1,2,3$, $\mathcal{N}_1 = 3$, $\mathcal{N}_2 = 1,2$, $p_1 = 1$, $p_2 = 2$, $p_3 = 3$, $d_1 = d_2 = d_3 = 1$, $\varepsilon_1 = 2$, $\varepsilon_2 = 0$ ($\varepsilon_1 + \varepsilon_2 = 2$). G has a feasible solution $S' = (1,3,2)$ but F has no feasible solution. As a consequence, the reverse is not true for $sf = \sum w_j U_j$.

So, these problems are not comparable.

- Case of a linear combination approach for $mf_1 = mg_2$ and $sf_2 = sg_1$.

The objective function of F is Min $Z = \alpha \max f_1(\mathcal{N}_1) + \beta \sum f_2(\mathcal{N}_2)$ and for problem G Min $Z' = \alpha' \sum f_2(\mathcal{N}) + \beta' \max f_1(\mathcal{N}_1) = (\alpha' \sum f_2(\mathcal{N}_2) + \beta' \max f_1(\mathcal{N}_1)) + \alpha' \sum f_2(\mathcal{N}_1)$. The term $\sum f_2(\mathcal{N}_1)$ is not considered in the objective function of problem F, thus problems are not comparable.

- Case of a goal programming approach for $mf_1 = mg_2$ and $sf_2 = sg_1$.

Problem F is to find a solution S such that $\max f_1(\mathcal{N}_1) \leq \varepsilon_1$ and $\sum f_2(\mathcal{N}_2) \leq \varepsilon_2$. Problem G is to find a solution S' such that $\sum f_2(\mathcal{N}_1) + \sum f_2(\mathcal{N}_2) \leq \varepsilon'_1$ and $\max f_1(\mathcal{N}_1) \leq \varepsilon'_2$. For the same reason, problems F and G are not comparable.

Hence, problems F and G are equivalent if and only if Z_1, Z_2, Z_3 and Z_4 are the same and of the type min-sum and if the multicriteria approach is a linear combination of criteria.

Furthermore, if $\mathcal{N}_1 = \mathcal{N}$, the problem is a classical multicriteria scheduling problem, which implies that this model is more general. In all the other cases, the problems are not comparable.

3. Single machine problems

In this section, we consider the case of a single machine environment. We first present some polynomially solvable problems, and then some NP-hard problems.

3.1. Polynomially solvable problems

Proposition 1. *The following problems can be solved in polynomial time:*

1. *Problems* $1||F_\ell\big(C_{\max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ *and* $1||\varepsilon\big(C_{\max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$
2. *Problem* $1||F_\ell\big(C_{\max}(\mathcal{N}), L_{max}(\mathcal{N}_1)\big)$ *and* $1||\varepsilon\big(C_{\max}(\mathcal{N})/L_{max}(\mathcal{N}_1)\big)$
3. *Problem* $1||F_\ell\big(C_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$ *and* $1||\varepsilon\big(C_{\max}(\mathcal{N})/\sum w_j C_j(\mathcal{N}_1)\big)$
4. *Problem* $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), \sum w'_j C_j(\mathcal{N}_1)\big)$

Proof. Problems 1 are trivial since it is sufficient to schedule the jobs of $\mathcal{N}_1$ first in an arbitrary order and the remaining jobs arbitrarily. Similarly, problems 2 are trivial since it is sufficient to schedule the jobs of $\mathcal{N}_1$ first in EDD order and problems 3 are trivial since it is sufficient to schedule the jobs of $\mathcal{N}_1$ first in WSPT order. Remember that problems $1||F_\ell\big(C_{\max}(\mathcal{N}_1), \sum w_j C_j(\mathcal{N}_2)\big)$ and $1||\varepsilon\big(C_{\max}(\mathcal{N}_1)/\sum w_j C_j(\mathcal{N}_2)\big)$ are NP-hard (Baker and Smith (2003), Yuan *et al.* (2005), Agnetis *et al.* (2004)). Problem 4 is also trivial (see Baker and Smith (2003) and Section 2.3). □.

According to the results presented in Baker and Smith (2003), Yuan *et al.* (2005) and Cheng *et al.* (2008), we can deduce the complexity of the following scheduling problems.

Proposition 2. *The following problems can be solved in polynomial time:*

- *Problem* $1||F_\ell\big(\sum C_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ *and* $1||\varepsilon\big(\sum C_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$
- *Problem* $1||F_\ell\big(L_{max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ *and* $1||\varepsilon\big(L_{max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$
- *Problem* $1||F_\ell\big(L_{\max}(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$ *and* $1||\varepsilon\big(L_{\max}(\mathcal{N})/L_{\max}(\mathcal{N}_1)\big)$

Proofs.

- Problem $1||F_\ell\big(\sum C_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ is polynomial.

An optimal solution always exists with the jobs in $\mathcal{N}_1$ and the jobs in $\mathcal{N} \setminus \mathcal{N}_1$ sequenced in WSPT 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 they belong to $\mathcal{N}_1$ or not. Let us suppose that the jobs of $\mathcal{N}_1$ in SPT are numbered as follows: $\{1, 2, \ldots, n_1\}$ and 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 $j \in \{n_1+1, n_1+2, \ldots, n\}$ ($a//b$ stands for the concatenation of a and b). The best sequence is the optimal solution of the problem. This algorithm can be implemented in $O(n\log(n))$.

- Problem $1||\varepsilon\big(\sum C_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ 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 $\mathcal{N}_1$ on the left so that the new solution satisfies the ε-constraint.

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

- Problem $1||F_\ell\big(L_{max}(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$ and $1||\varepsilon\big(L_{max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ are polynomial.

In the same way, the jobs are sorted in EDD order and all the sequences $EDD(\mathcal{N}_1 \cup \{n_1 + 1, \ldots, j\})//EDD(\{j+1, j+2, \ldots, n\})$ for all $j \in \{n_1+1, n_1+2, \ldots, n\}$ are tested. The best sequence gives an optimal solution.

- Problems $1||F_\ell\big(L_{\max}(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$ and $1||\varepsilon\big(L_{\max}(\mathcal{N})/L_{\max}(\mathcal{N}_1)\big)$ are polynomial.

There exists an optimal solution such that the jobs of $\mathcal{N}_1$ are sorted in EDD order and the jobs of $\mathcal{N} \setminus \mathcal{N}_1$ are sorted in EDD order.

We introduce the following notations: $P_{(i,j)} = \sum_{k=1}^{i} p_k + \sum_{k=n_1+1}^{j} p_k$. We consider that L corresponds to $L_{\max}(\mathcal{N}_1)$. Furthermore, we assume that the jobs are numbered according to EDD rule, that is: $d_1 \le d_2 \le \ldots \le d_{n_1}$ on the one hand and $d_{n_1+1} \le \ldots \le d_n$ on the other hand.

The objective function to minimize is $F(i,j,L) = L_{\max}(\mathcal{N})$. In the following, $\mathcal{L}$ denotes the set of possible $L_{max}(\mathcal{N}_1)$ values ($|\mathcal{L}| \le n_1 \times n_2 = n^2$, Yuan *et al.* (2005)). We define $F(0, n_1, 0) = 0$,

$F(i,j,L) = +\infty$, $\forall (i,j,L) \neq (0,n_1,0)$. Let consider now the triplet $F(i,j,L)$:

—if job $i+1$ is inserted, triplet $F(i+1,j,\max(L,P_{(i,j)})$ is then updated:

$$F(i+1,j,\max(L,P_{(i,j)})) \leftarrow \min\big(F(i+1,j,\max(L,P_{(i,j)}));\max(F(i,j,L);P_{(i,j)}-d_i)\big)$$

—if job $j+1$ is inserted, triplet $F(i,j+1,\max(L,P_{(i,j)})$ is then updated:

$$F(i+1,j,L)) \leftarrow \min\big(F(i,j+1,L);\max(F(i,j+1,L);P_{(i,j)}-d_j)\big)$$

For the problem of minimizing $\alpha L_{\max}(\mathcal{N}) + \beta L_{\max}(\mathcal{N}_1)\big)$, the optimal solution corresponds to

$$\min_{L\in\mathcal{L}} \alpha F(n_1,n,L) + \beta L$$

For the problem of minimizing $L_{\max}(\mathcal{N})$ with $L_{\max}(\mathcal{N}_1) \leq \varepsilon$, the optimal solution corresponds to

$$\min_{L\in\mathcal{L},L\leq\varepsilon} F(n_1,n,L)$$

The overall running time to find an optimal solution is bounded by $O(n^4)$. $\square$

3.2. NP-hard problems

Proposition 3. *The following problem is ordinary NP-hard:* $1||\varepsilon(\sum w_jC_j(\mathcal{N})/C_{\max}(\mathcal{N}_1))$.

Proof. We denote by WCCM the decision problem associated to $1||\varepsilon\big(\sum w_jC_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$. This problem is defined by:

WCCM

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

Question: Is there a single machine schedule σ for $\mathcal{N}$ such that $\sum_{j\in\mathcal{N}} w_jC_j \leq Y$ and $\max_{j\in\mathcal{N}_1} C_j \leq Y_1$?

We prove that PARTITION $\propto$ WCCM.

PARTITION

Data: Finite set $\mathcal{A}$ of r elements $a_1,a_2,\ldots,a_r$, with integer sizes $s(a_i)$, $\forall i, 1\leq i\leq r$, $\sum_{i=1}^r s(a_i) = 2B$.

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

Given an arbitrary instance of PARTITION, we construct an instance of WCCM as follows:

- $\mathcal{N} = \{1,2,...,r+1\}$, $\mathcal{N}_1 = \{r+1\}$,
- for $j \in \{1,2,...,r\}$: $p_j = w_j = s(a_j)$; $p_{r+1} = 2B$, $w_{r+1} = 1$,

- $Y = 6B^2 + 4B - 1$ and $Y_1 = 3B$,

($\Rightarrow$) Given a feasible solution to PARTITION, we can define a solution to WCCM by sequencing a subset of jobs corresponding to $\mathcal{A}_1$ before job $r+1$, the jobs corresponding to $\mathcal{A} \setminus \mathcal{A}_1$ are scheduled after job $r+1$. This schedule satisfies the conditions and the answer to problem WCCM is 'yes'.

($\Leftarrow$) If there exists a feasible solution to problem WCCM, then:

1. $\max_{j \in \mathcal{N}_1} C_j \leq Y_1 \Leftrightarrow \sum_{j \in \mathcal{A}_1} p_j + p_{r+1} \leq Y_1 \Leftrightarrow \sum_{j \in \mathcal{A}_1} s(a_j) \leq B$
2. $\sum_{j \in \mathcal{N}} w_j C_j \leq Y$

$\Leftrightarrow (\sum_{j \in \mathcal{N} \setminus \mathcal{N}_1} p_j)(\sum_{j \in \mathcal{N} \setminus \mathcal{N}_1} w_j) + w_{r+1}(\sum_{j \in \mathcal{A}_1} p_j + p_{r+1}) + p_{r+1}(\sum_{j \in \mathcal{N} \setminus (\mathcal{N}_1 \cup \mathcal{A}_1)} w_j) \leq Y$

$\Leftrightarrow 4B^2 + w_{r+1}(2B - \sum_{j \in \mathcal{A}_1} p_j) + p_{r+1} w_{r+1} + p_{r+1}(\sum_{j \in \mathcal{N} \setminus (\mathcal{N}_1 \cup \mathcal{A}_1)} w_j) \leq Y$

$\Leftrightarrow (p_{r+1} - w_{r+1}))(\sum_{j \in \mathcal{A} \setminus \mathcal{A}_1} s(a_j)) \leq Y - 4B^2 - 2B w_{r+1} - p_{r+1} w_{r+1} = Y - 4B^2 - 3B - 1$

$\Leftrightarrow \sum_{j \in \mathcal{A} \setminus \mathcal{A}_1} s(a_j) \leq B$

Because $\sum_{j \in \mathcal{A}} s(a_j) = 2B$, we have $\sum_{j \in \mathcal{A}_1} s(a_j) = B$ and $\sum_{j \in \mathcal{A} \setminus \mathcal{A}_1} s(a_j) = B$ and the answer to PARTITION is 'yes'. $\square$

PROPOSITION 4. *The following problems are strongly NP-hards:* $1||\varepsilon\big(\sum w_j C_j(\mathcal{N}) / L_{\max}(\mathcal{N}_1)\big)$; $1||\varepsilon\big(L_{\max}(\mathcal{N}) / \sum w_j C_j(\mathcal{N}_1)\big)$; $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$; $1||F_\ell\big(L_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$.

Proof. For the two first problems, the result can be obtained by using the instance defined by Lawler Lawler (1977) for problem $1||\sum w_j T_j$ and the sketch of the proof for Theorem 2.2 in Ng *et al.* (2006).

- Problem $1||F_\ell\big(L_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$ is strongly NP-Hard.

The decision version of this problem is given by the following problem, denoted LMWC.

LMWC

Data: A set $\mathcal{N}$ of n jobs; a subset $\mathcal{N}_1 \subset \mathcal{N}$; processing times p_j for each job j, $1 \leq j \leq n$ and due dates d_j if $j \in \mathcal{N}_1$; a, b and y are real values.

Question: Does a schedule σ exist such that $aL_{\max}(\mathcal{N}) + b\sum w_j C_j(\mathcal{N}_1) \leq y$?

It is clear that problem LMWC is in the class NP. We next prove that LMWC is NP-complete in the strong sense by a reduction from 3-PARTITION (Garey and Johnson (1979)).

Given an instance of 3-PARTITION, we construct an instance of the LMWC problem as follows:

—$\mathcal{N} = \{1, 2, \ldots, 4r+1\}$, $\mathcal{N}_1 = \{1, 2, \ldots, 3r\}$, $P_n = \sum_{j=1}^{4r+1} p_j$

—for $j \in \{1, 2, \ldots, 3r\}$: $p_j = w_j = a_j$ and $d_j = P_n$,

—for $j \in \{3r+1, 3r+2, \ldots, 4r+1\}$: $p_j = B$ and $d_j = (2(j-3r)-1)B$,

—$y = \frac{1}{2}B^2 r(r+1) + \sum_{i=1}^{3n} \sum_{j=1}^{i} a_i a_j$,

—$a = 2y$ and $b = 1$.

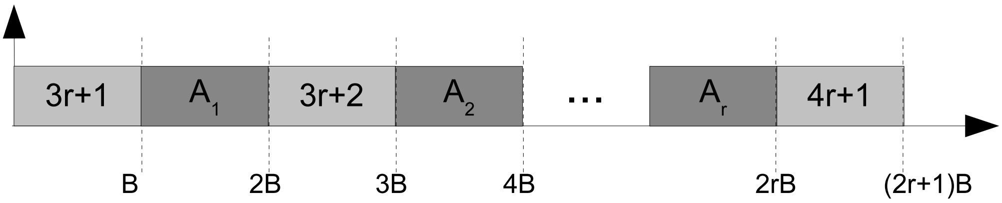

Figure 2 An optimal sequence σ of $1||F_\ell\big(L_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1)\big)$

Because of the definition of coefficient a, $L_{\max}(\mathcal{N})$ has to be smaller than or equal to 0. Because job $3r+1$ has a duration and a due date equal to B, this job has to start at time 0. Thus, in what follows, $L_{\max}(\mathcal{N}) = 0$ and the jobs $j \in \{3r+1, 3r+2, \ldots, 4r+1\}$ cannot be tardy.

($\Rightarrow$) Suppose that 3-PARTITION has a 'yes' answer which partition the set A into r disjoint subsets A_j $(1 \le j \le r)$. Then we construct the following solution to the LMWC problem. We form r blocks, where jth block contains job $3r+j$ followed by the jobs corresponding to the elements in A_j, which we process contiguously in this order. The last job of the sequence is job $4r+1$, which completes at time P_n.

It can easily be established that the maximum lateness is equal to 0 and that the weighted completion time of jobs in $\mathcal{N}_1$ is equal to $\sum_{k=0}^{r-1} B^2(r-k) + \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j = y$. Thus, $aL_{\max}(\mathcal{N}) + b\sum w_j C_j(\mathcal{N}_1) = y$ which means that LMWC has also a 'yes' answer.

($\Leftarrow$) Suppose now that LMWC has a 'yes' answer, we denote by σ a feasible solution. The maximum lateness of σ is equal to 0. Each job $j \in \mathcal{N} \setminus \mathcal{N}_1$ is early, we are going to prove that all these jobs complete at their due date.

Define H_j $(j = 1, \ldots, r)$ as the jobs in $\mathcal{N}_1$ that are processed between jobs $3r+j$ and $3r+j+1$. We use $p(H_j)$ and $w(H_j)$ as a short-hand notation for the total processing time and the total weight of the jobs in H_j, respectively. The total weighted completion time of the jobs in $\mathcal{N}_1$ according to σ, $\sum w_j C_j(\mathcal{N}_1)$, is then equal to $\sum_{i=1}^{3r}\sum_{j=1}^{i} a_i a_j + B\sum_{j=1}^{r} jw(H_j)$.

Since job $3r+2$ is completed by its due date, $p(H_1) \le B$. Similarly, since job $3r+3$ is completed by its due date, we have $p(H_1)+p(H_2) \le 2B$. Extending this reasoning, we find that

$rp(H_1)+(r-1)p(H_2)+\ldots+p(H_r) \le \sum_{i=1}^{r} iB = Br(r+1)/2$

$\Leftrightarrow (r+1)[p(H_1)+p(H_2)+\ldots+p(H_r)] - \sum_{j=1}^{r} jp(H_j) \le Br(r+1)/2$

$\Leftrightarrow (r+1)[rB] - \sum_{j=1}^{r} jp(H_j) \le Br(r+1)/2$

$\Leftrightarrow Br(r+1) - Br(r+1)/2 \le \sum_{j=1}^{r} jp(H_j)$

$\Leftrightarrow Br(r+1)/2 \le \sum_{j=1}^{r} jp(H_j) \Leftrightarrow Br(r+1)/2 \le \sum_{j=1}^{r} jw(H_j)$ (1) since $w(H_j)=p(H_j)$.

On the other hand, we have:

$aL_{\max}(\mathcal{N}) + b\sum w_j C_j(\mathcal{N}_1) \le y$

$\Leftrightarrow \sum w_j C_j(\mathcal{N}_1) \le \frac{1}{2}B^2 r(r+1) + \sum_{i=1}^{3n}\sum_{j=1}^{i} a_i a_j$

$\Leftrightarrow \sum_{i=1}^{3r}\sum_{j=1}^{i} a_i a_j + B\sum_{j=2}^{r} jw(H_j) \le \sum_{i=1}^{3n}\sum_{j=1}^{i} a_i a_j + \frac{1}{2}B^2 r(r+1)$

$\Leftrightarrow \sum_{j=1}^{r} jw(H_j) \le Br(r+1)/2$ (2)

From (1) and (2) we deduce that:

$\Leftrightarrow \sum_{j=1}^{r} jp(H_j) = Br(r+1)/2$

Because $\forall j \in \{1,\ldots,r\}$, $p(H_1)+\ldots+p(H_j) \le jB$ we can deduce that $\forall j \in \{1,\ldots,r\}$, $p(H_j)=B$. Hence, the partitioning of A into $H_1,\ldots,H_r$ yields a yes-instance to 3-PARTITION. This completes the proof. $\square$

- Problem $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$ is strongly NP-hard.

The decision version of this problem is given by the following problem, denoted WCLM.

WCLM

Data: a set $\mathcal{N}$ of n jobs; a subset $\mathcal{N}_1 \subset \mathcal{N}$, processing times p_j for each job j, $1 \le j \le n$ and due dates d_j if $j \in \mathcal{N}_1$; a, b and y real numbers.

Question: Does a schedule σ exist such that $a\sum w_j C_j(\mathcal{N}) + bL_{\max}(\mathcal{N}_1) \le y$?

It is clear that problem WCLM is in NP. We next prove that WCLM is NP-complete in the strong sense by a reduction from 3-PARTITION, which is known to be NP-complete in the strong sense (Garey and Johnson (1979)).

3-PARTITION

Data: an integer B and a set $A = \{a_1, .., a_{3r}\}$ of $3r$ positive integers with $B/4 < a_k < B/2$ ($k = 1, .., 3r$) and $\sum_{k=1}^{3r} a_k = rB$.

Question: Is there a partition of A into r mutually disjoint subsets $A_1, \ldots, A_r$ such that the elements in A_k sum up to B for each $k = 1, .., r$?

Given an arbitrary instance of 3-PARTITION, we construct an instance of WCLM as follows:

—$\mathcal{N} = \{1, 2, \ldots, 4r\}$, $\mathcal{N}_1 = \{1, 2, \ldots, r\}$,

—for $j \in \{1, 2, \ldots, r\}$: $p_j = w_j = 1$ and $d_j = 1 + (j-1)(B+1)$, let $D = \sum_{i=1}^{r} d_j$,

—for $j \in \{r+1, r+2, \ldots, 4r\}$: $p_j = a_{j-r}$, $w_j = D^2 a_{j-r}$ (all jobs will have the same ratio $\frac{p_j}{w_j} = 1/D^2$),

—$a = 1$ and $b = y = D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + D^2 Br(r+1)/2 + D$,

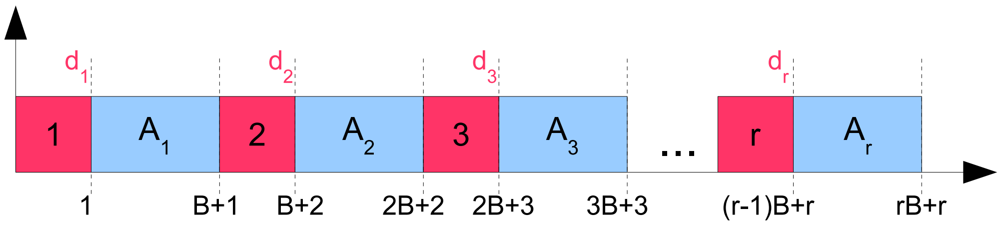

Figure 3 An optimal sequence σ of $1||F_\ell(\sum w_j C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1))$

($\Rightarrow$) Suppose that 3-PARTITION has a 'yes' answer. Then we construct the following solution to the WCLM problem. We form r blocks, where jth block contains job j followed by the jobs corresponding to the elements in A_j, which we process contiguously in this order. It can be easily verified that each job of $\mathcal{N}_1$ finishes at its due date. Thus, $bL_{\max}(\mathcal{N}_1) = 0$. Furthermore, it can be easily established that the weighted completion time of jobs in $\mathcal{N}$ is equal to $D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + Br(r+1)/2 + D$. Thus, WCLM has also a 'yes' answer.

($\Leftarrow$) Suppose now that WCLM has a 'yes' answer. We denote by σ a feasible solution. Because the delay of job 1 is greater than or equal to 0, $L_{\max}(\mathcal{N}_1) \geq 0$. Because $\sum w_j C_j(\mathcal{N}) > 0$ and because

$y = b$, we have $L_{\max}(\mathcal{N}_1) < 1$ and thus $L_{\max}(\mathcal{N}_1) = 0$ and job 1 starts at time 0. The total weighted completion time of the jobs of $\mathcal{N}_1$ is then smaller than or equal to D.

Define H_j $(j = 1,..,r-1)$ as the jobs in $\mathcal{N} \setminus \mathcal{N}_1$ that are processed between jobs j and $j+1$, and define H_r as the set of jobs in $\mathcal{N} \setminus \mathcal{N}_1$ that are processed after job r in σ. We use $p(H_j)$ and $w(H_j)$ as a short-hand notation for the total processing time and the total weight of the jobs in H_j, respectively. The total weighted completion time of the jobs in $\mathcal{N} \setminus \mathcal{N}_1$ according to σ is then equal to $D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + \sum_{j=1}^{r} j w(H_j)$ (the first part is the total weighted completion time of jobs of $\mathcal{N} \setminus \mathcal{N}_1$ without considering the contribution of the jobs in $\mathcal{N}_1$). Since job 2 is completed by its due date, we know that $p(H_1) \le B$. Similarly, since job 3 is completed by its due date, we know that $p(H_1) + p(H_2) \le 2B$. Extending this reasoning, we find that

$rp(H_1) + (r-1)p(H_2) + \ldots + p(H_r) \le \sum_{i=1}^{r} iB = Br(r+1)/2$

$\Leftrightarrow (r+1)[p(H_1) + p(H_2) + \ldots + p(H_r)] - \sum_{j=1}^{r} jp(H_j) \le Br(r+1)/2$

$\Leftrightarrow (r+1)[rB] - \sum_{j=1}^{r} jp(H_j) \le Br(r+1)/2 \Leftrightarrow Br(r+1) - Br(r+1)/2 \le \sum_{j=1}^{r} jp(H_j)$

$\Leftrightarrow Br(r+1)/2 \le \sum_{j=1}^{r} jp(H_j)$

On the other hand, we have

$\sum w_j C_j(\mathcal{N} \setminus \mathcal{N}_1) + \sum w_j C_j(\mathcal{N}_1) \le D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + D^2 Br(r+1)/2 + D$

$\Leftrightarrow D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + \sum_{j=1}^{r} jw(H_j) + \sum w_j C_j(\mathcal{N}_1) \le D^2 \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + D^2 Br(r+1)/2 + D$

$\Leftrightarrow \sum_{j=1}^{r} jw(H_j) + \sum w_j C_j(\mathcal{N}_1) \le D^2 Br(r+1)/2 + D$

$\Leftrightarrow D^2 \sum_{j=1}^{r} jp(H_j) + \sum w_j C_j(\mathcal{N}_1) \le D^2 Br(r+1)/2 + D$

$\Leftrightarrow \sum_{j=1}^{r} jp(H_j) \le Br(r+1)/2 + (D - \sum w_j C_j(\mathcal{N}_1))/D^2$

Thus, we have:

$$Br(r+1)/2 \le \sum_{j=1}^{r} jp(H_j) \le Br(r+1)/2 + (D - \sum w_j C_j(\mathcal{N}_1))/D^2$$

Since $0 < \sum w_j C_j(\mathcal{N}_1) \le D$, $0 < (D - \sum w_j C_j(\mathcal{N}_1))/D^2 < 1$.

Thus, $\sum_{j=1}^{r} jp(H_j) = Br(r+1)/2$

Because $\forall j \in \{1,\ldots,r\}$, $p(H_1) + \ldots + p(H_j) \le jB$ we can deduce that $\forall j \in \{1,\ldots,r\}$, $p(H_j) = B$. Hence, the partitioning of A into $H_1,\ldots,H_r$ yields a yes-instance to 3-PARTITION. This completes the proof. □

Remark: as a consequence, the classical biobjective scheduling problem $1||F_\ell(\sum w_j C_j(\mathcal{N}), L_{\max}(\mathcal{N}))$ is also strongly NP-hard.

3.3. Total (weighted) completion time for both criteria

We consider in this section two problems for which the two objective functions are the same. The first problem involves $\sum C_j$ objective function and the second one involves $\sum w_j C_j$.

Problem $1||\varepsilon\big(\sum C_j(\mathcal{N})/\sum C_j(\mathcal{N}_1)\big)$

PROPOSITION 5. *There always is an optimal solution that respects the following properties:*

1. *there is no idle time.*
2. *jobs in* $\mathcal{N}1$ *respect the SPT order (*Shortest Processing Time *first).*
3. *jobs in* $(\mathcal{N}\setminus\mathcal{N}1)$ *respect the SPT order.*
4. *if* $p_i \le p_j$*, 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 respect the SPT order cannot decrease the solution quality. The last point is true because the permutation of i and j improves both $\sum_{j\in\mathcal{N}} C_j$ and $\sum_{j\in\mathcal{N}_1} C_j$. Note that point 4 is not true if $(i,j)\in(\mathcal{N}\setminus\mathcal{N}_1)\times\mathcal{N}_1$. □

PROPOSITION 6. *Problem* $1||\varepsilon\big(\sum C_j(\mathcal{N})/\sum C_j(\mathcal{N}_1)\big)$ *is binary NP-hard.*

Let us remember PARTITION problem Garey and Johnson (1979) defined as follows:

PARTITION

Data: Finite set $\mathcal{A}$ of r elements $a_1, a_2, \dots, a_r$, with integer sizes $s(a_i), \forall i, 1\le i\le r$, $\sum_{i=1}^r s(a_i) = 2B$.

Question: Is there a subset $\mathcal{A}_1$ of indices such that $\sum_{i\in\mathcal{A}_1} s(a_i) = \sum_{i\in\{1,2,\dots,r\}\setminus\mathcal{A}_1} s(a_i) = B$?

We define the problem PWDE (PARTITION with distinct elements) by:

PWDE

Data: Finite set $\mathcal{B}$ of t elements $b_1, b_2, \dots, b_t$ with distinct integer sizes $(s(b_i)\neq s(b_j),\ \forall i,j)$, $\sum_{i=1}^t s(b_i) = 2C$.

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

This problem is ordinary NP-hard (HuynhTuong *et al.* (2009)). We denote by INT1m the decision problem associated to $1||\varepsilon\big(\sum C_j(\mathcal{N})/\sum C_j(\mathcal{N}_1)\big)$. This problem is defined by:

INT1M

Data: a set $\mathcal{N}$ of n jobs, a subset $\mathcal{N}_1 \subset \mathcal{N}$, processing times p_j for each job j, $1 \le j \le n$, two integer values Y and Y_1.

Question: Is there a single machine schedule σ for $\mathcal{N}$ such that $\sum_{j\in\mathcal{N}} C_j \le Y$ and $\sum_{j\in\mathcal{N}_1} C_j \le Y_1$?

We prove that PWDE $\propto$ INT1m.

We consider an instance of PWDE and we assume w.l.o.g. that $s(a_1) < s(a_2) < \ldots < s(a_t)$. We have $\min_{i=1}^{t-1} \frac{a_{i+1}}{a_i} > 1$. It is always possible to find α and K such that $1 < \alpha < \min_{i=1}^{t-1} \frac{a_{i+1}}{a_i}$ and $\alpha K \in \mathbb{N}$ (if $\frac{a_{\ell+1}}{a_\ell} = \min_{i=1}^{t-1} \frac{a_{i+1}}{a_i}$ 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 for instance $\alpha = \frac{10\times a_{\ell+1}}{10\times a_\ell + 1}$ and $K = 10 \times a_\ell + 1$ otherwise).

Because of the definition of α and K we have: $Ks(a_i) < \alpha K s(a_i) < K s(a_{i+1}) < \alpha K s(a_{i+1})$.

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

We define an instance of problem INT1m as follows: $n = 2t$ and

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

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 SPT rule (see Figure 4).

We have:

$\sum_{j=1}^{n} C_j(S^0) = Ks(a_1) + (Ks(a_1) + \alpha K s(a_1)) + (Ks(a_1) + \alpha K s(a_1) + Ks(a_2)) + (Ks(a_1) + \alpha K s(a_1) + Ks(a_2) + \alpha K s(a_2)) + \ldots$

$\Rightarrow \sum_{j=1}^{n} C_j(S^0) = 2tKs(a_1) + (2t-1)\alpha K s(a_1) + (2t-2)Ks(a_2) + (2t-3)\alpha K s(a_2) + \ldots$

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

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

In the same way, we obtain:

$\sum_{j\in\mathcal{N}_1} C_j(S^0) = (K\times s(a_1)+\alpha K\times s(a_1))+(K\times s(a_1)+\alpha K\times s(a_1)+K\times s(a_2)+\alpha K\times s(a_2))+\dots$

$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^0) = (K\times s(a_1)+\alpha K\times s(a_1))+(K\times s(a_1)+\alpha K\times s(a_1)+K\times s(a_2)+\alpha K\times s(a_2))+$
$\dots$

$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^0) = t\times(K\times s(a_1)+\alpha K\times s(a_1))+(t-1)\times(K\times s(a_2)+\alpha K\times s(a_2))+\dots$

$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^0) = t\times((1+\alpha)K\times s(a_1))+(t-1)\times((1+\alpha)K\times s(a_2))+\dots$

$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^0) = K(1+\alpha)(t\times s(a_1)+(t-1)\times s(a_2)+\dots$

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

Thus, this solution is not a feasible solution for problem INT1m: $\sum_{j\in\mathcal{N}} C_j(S^0)\le Y$ but $\sum_{j\in\mathcal{N}_1} C_j(S^0) > Y_1$.

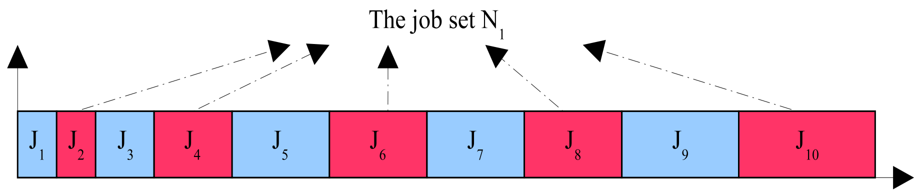

Figure 4 Initial sequence with 10 jobs

Let us suppose that the answer to PWDE is 'yes'. We are going to propose a method for permuting consecutive jobs for decreasing $\sum_{j\in\mathcal{N}_1} C_j$ and increasing $\sum_{j\in\mathcal{N}} C_j$ at the same time. We consider the set of jobs $\mathcal{G}=\{j\in\mathcal{N}/j=2i\wedge i\in\mathcal{B}_1\}$. 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 $\sum_{j\in\mathcal{N}} C_j(S^1)$ 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').

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

Thus, $\sum_{j\in\mathcal{N}} C_j(S^1) = \sum_{j\in\mathcal{N}} C_j(S^0)+\sum_{j\in\mathcal{G}}(p_j-p_{j-1}) = \sum_{j\in\mathcal{N}} C_j(S^0)+\sum_{j\in\mathcal{G}}(\alpha K\times s(a_{j/2}) - K\times s(a_{j/2}))$.

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

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

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

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

$\Rightarrow \sum_{j\in\mathcal{N}_1} C_j(S^1) = \sum_{j\in\mathcal{N}_1} C_j(S^0) - \sum_{j\in\mathcal{G}} K \times s(a_{j/2})$

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

Thus, S^1 is the sequence for which the answer to INT1m is 'yes'.

Suppose now that the answer to INT1m is 'yes' for sequence σ. If σ does not respect the conditions of proposition 5, then we shift all the jobs to the left, we apply the SPT rule to the jobs of $\mathcal{N}_1$, we apply the SPT rule to the jobs of $\mathcal{N}\setminus\mathcal{N}_1$ and each time condition 4 occurs, we permute jobs i and j. We obtain a new sequence σ' so that:

- $\sum_{j\in\mathcal{N}} C_j(\sigma') \le \sum_{j\in\mathcal{N}} C_j(\sigma) \le Y$ **(1)**
- $\sum_{j\in\mathcal{N}_1} C_j(\sigma') \le \sum_{j\in\mathcal{N}_1} C_j(\sigma) \le Y_1$ **(2)**
- and σ' satisfies the conditions of Proposition 5.

We will now compare σ' and S^0.

Let us consider the job number $2i$. This job is in position $2i$ in S^0 and 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 $\mathcal{N}_1$ since the jobs of $\mathcal{N}_1$ in σ' are sorted according to SPT. Thus this job belongs to $\mathcal{N}\setminus\mathcal{N}_1$. This case is not possible because of condition 4 of proposition 5. Thus, $k \le 2i$. Similarly, we can show that job $2i-1$ is in position $2i-1$ in S^0 and in position l in σ' with $l \ge 2i-1$. The case is illustrated in Figure 5.

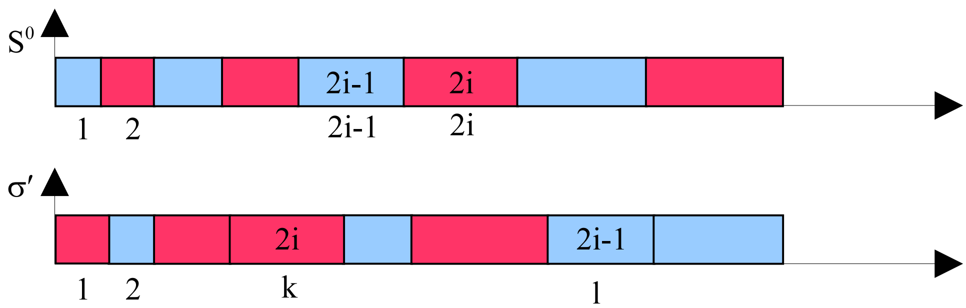

Figure 5 Sequences S^0 and σ' and position of job $2i$

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

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

$\Rightarrow \sum_{j \in \mathcal{N}_1} C_j(\sigma') = \sum_{j \in \mathcal{N}_1} C_j(S^0) - \sum_{j \in \mathcal{N}_1} \sum_{k \in \mathcal{H}_j} p_k$ **(3)**

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

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

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

$\Rightarrow KC \leq \sum_{j \in \mathcal{N}_1} \sum_{k \in \mathcal{H}_j} Ks(a_{(k+1)/2}) \Rightarrow C \leq \sum_{j \in \mathcal{N}_1} \sum_{k \in \mathcal{H}_j} s(a_{(k+1)/2})$ **(4)**

Due to Proposition 5.4, 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 $j \in \mathcal{N} \setminus \mathcal{N}_1$ in σ' would be unchanged or moved to the right. The deviation of the completion time of a job $j \in \mathcal{N} \setminus \mathcal{N}_1$ 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 scheduled before j in σ'. For instance, in Figure 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:

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

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

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

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

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

Due to (3), we have: $\sum_{j \in \mathcal{N}_1} C_j(\sigma') - \sum_{j \in \mathcal{N}_1} C_j(S^0) = \sum_{j \in \mathcal{N}_1} \sum_{k \in \mathcal{H}_j} p_k$

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

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

Since $p_j > p_k$ where $j \in \mathcal{N}_1, k \in \mathcal{H}_j$, we have $p_j \geq p_{k+1}$ with $j, k+1 \in \mathcal{N}_1$ and $k \in \mathcal{H}_j$ **(5)**

$\Rightarrow \sum_{j\in\mathcal{N}} C_j(\sigma') - \sum_{j\in\mathcal{N}} C_j(S^0) \geq \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j}(p_{k+1} - p_k) = \sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j}\big(\alpha K s(a_{(k+1)/2}) - K s(a_{(k+1)/2})\big)$

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

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

According to (2) that $\sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} s(a_{(k+1)/2}) \geq C$, we have then:

$\sum_{j\in\mathcal{N}} C_j(\sigma') \geq \sum_{j\in\mathcal{N}} C_j(S^0) + \beta K C = Y$ **(6)**

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

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

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

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

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

$\Rightarrow$ The equality $\sum_{j\in\mathcal{N}_1}\sum_{k\in\mathcal{H}_j} s(a_{(k+1)/2}) = C$ defines the subset B_1 of PWDE.

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

Problem $1||\varepsilon\big(\sum w_j C_j(\mathcal{N}) / \sum w'_j C_j(\mathcal{N}_1)\big)$

PROPOSITION 7. *Problem* $1||\varepsilon\big(\sum w_j C_j(\mathcal{N}) / \sum w'_j C_j(\mathcal{N}_1)\big)$ *is strongly NP-hard.*

Proof. We denote by INT1MWC the decision problem associated to $1||\varepsilon\big(\sum w_j C_j(\mathcal{N}) / \sum w'_j C_j(\mathcal{N}_1)\big)$. This problem is defined by:

INT1MWC

Data: A set $\mathcal{N}$ of n jobs, a subset $\mathcal{N}_1 \subset \mathcal{N}$, processing times p_j and weights w_j, w'_j for each job j, $1 \leq j \leq n$, two integer values Y and Y_1.

Question: Is there a single machine schedule σ for $\mathcal{N}$ such that $\sum_{j\in\mathcal{N}} w_j C_j \leq Y$ and $\sum_{j\in\mathcal{N}_1} w'_j C_j \leq Y_1$?

We show that the answer to problem 3-PARTITION is 'yes' if and only if the answer to problem INT1MWC is 'yes'. Given an instance of 3-PARTITION, we construct an instance of INT1MWC as follows:

- $\mathcal{N} = \{1, 2, \ldots, 4r\}$, $\mathcal{N}_1 = \{1, 2, \ldots, r\}$,
- for the job $j \in \{1, 2, \ldots, r\}$: $p_j = B$, $w_j = 1$ and $w'_j = B^{r-j}$,
- for the job $j \in \{r+1, r+2, \ldots, 4r\}$: $p_j = a_{j-r}$, $w_j = a_{j-r}$ (all jobs will have the same ratio $\frac{p_j}{w_j} = 1$),
- $Y_1 = \sum_{i=1}^{3r} \sum_{j=1}^{i} a_i a_j + Bn(B+1)/2$ and $Y_2 = \sum_{i=1}^{r} (2i-1) B^{r-i+1}$.

We let the reader complete the proof, similarly as for problem $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$. □

3.4. Open problems

PROPOSITION 8. *The following problems remain open:* $1||\varepsilon\big(\sum w_j C_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ *and* $1||F_\ell\big(\sum w_j C_j(\mathcal{N}), C_{\max}(\mathcal{N}_1)\big)$; $1||\varepsilon\big(L_{\max}(\mathcal{N})/\sum C_j(\mathcal{N}_1)\big)$ *and* $1||F_\ell\big(L_{\max}(\mathcal{N}), \sum C_j(\mathcal{N}_1)\big)$; $1||\varepsilon\big(\sum C_j(\mathcal{N})/L_{\max}(\mathcal{N}_1)\big)$ *and* $1||F_\ell\big(\sum C_j(\mathcal{N}), L_{\max}(\mathcal{N}_1)\big)$.

4. Parallel machine scheduling problems

We consider now that the workshop is composed by identical parallel machines. We assume that the number of machines is known and equal to m.

4.1. Corollaries

PROPOSITION 9. *The following problems are binary NP-hard:* $Pm||\varepsilon\big(\sum C_j(\mathcal{N})/\sum C_j(\mathcal{N}_1)\big)$ *and* $Pm||\varepsilon\big(\sum C_j(\mathcal{N}_1)/\sum C_j(\mathcal{N})\big)$; $Pm||\varepsilon\big(C_{\max}(\mathcal{N})/\sum w_j C_j(\mathcal{N}_1)\big)$ *and* $Pm||\varepsilon\big(\sum w_j C_j(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$; $Pm||\varepsilon\big(\sum w_j C_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ *and* $Pm||\varepsilon\big(C_{\max}(\mathcal{N}_1)/\sum w_j C_j(\mathcal{N})\big)$; $Pm||\varepsilon\big(C_{\max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ *and* $Pm||\varepsilon\big(C_{\max}(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$.

Proof: Since the scheduling problems $Pm||C_{\max}$ and $Pm||\sum w_j C_j$ are NP-Hard (see Lenstra *et al.* (1977) and Bruno et al. (1974), the proof is straightforward. □

Notice that the same problems with goal programming or linear combination of these objective functions are also binary NP-hard.

4.2. General dynamic programming formulation

We first consider the case of two-parallel machines. The problem is denoted by $P2||\varepsilon(Z_1(\mathcal{A})/Z_2(\mathcal{B}))$. In the following, the two cases are possible: $(\mathcal{A}=\mathcal{N})\wedge(\mathcal{B}=\mathcal{N}_1)$ or $(\mathcal{A}=\mathcal{N}_1)\wedge(\mathcal{B}=\mathcal{N})$ and Z_1 and Z_2 belong to $\{C_{\max}, \sum C_j, \sum w_jC_j\}$.

We assume that the jobs in $\mathcal{N}_1$ are numbered from 1 to $n_1 = |\mathcal{N}_1|$ and that the jobs in $\mathcal{N}\setminus\mathcal{N}_1$ are numbered from n_1+1 to n.

We denote by $F(i, j, P_1, Q_1, Q_2)$ the minimum cost of scheduling jobs $\{1, 2, \dots, i\} \in \mathcal{N}_1$ and jobs $\{n_1+1, n_1+2, \dots, j\} \in \mathcal{N}\setminus\mathcal{N}_1$ so that the sum of processing times of the jobs in $M1$ is equal to P_1. Q_1 and Q_2 depend on the objective functions Z_1 and Z_2. Clearly, the total processing time of all jobs $P = \sum_{j=1}^{n} p_j$ is an upperbound of P_1. Let Q'_1 (respectively Q'_2) be an upperbound of Q_1 (respectively Q_2).

The decision consists in assigning one job of $\mathcal{N}_1$ or of $\mathcal{N}\setminus\mathcal{N}_1$ on $M1$ or on $M2$. We first give a general formulation of the DP algorithm and then present its application to several problems.

The decisions are the following (in the case of two machines):

- assign the next job i in $\mathcal{N}_1$ to $M1$
- assign the next job i in $\mathcal{N}_1$ to $M2$
- assign the next job j in $\mathcal{N}\setminus\mathcal{N}_1$ to $M1$
- assign the next job j in $\mathcal{N}\setminus\mathcal{N}_1$ to $M2$

These decisions can be easily extended to the case of more than two machines (leading to $F(i, j, P_1, P_2, \dots, P_{m-1}, Q_1, Q_2, \dots, Q_m)$).

The general recursive relation (in the case of two machines) is given by $F(i, j, P_1, Q_1, Q_2)$:

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

$$F(i,j,P_1,Q_1,Q_2)=+\infty,\begin{pmatrix}\forall i>n_1,\\ \forall j\le n_1,\\ \forall(P_1,Q_1,Q_2)\end{pmatrix}$$

$$F(i,j,P_1,Q_1,Q_2)=+\infty,\begin{pmatrix}\forall i\in\{0,1,\dots,n_1\},\\ \forall j\in\{n_1,n_1+1,\dots,n\},\\ \forall((P_1,Q_1,Q_2)<(0,0,0)\vee(P_1,Q_1,Q_2)>(P,Q'_1,Q'_2))\end{pmatrix}$$

$$F(i,j,P_1,Q_1,Q_2)=\min\left\{\begin{array}{l}F(i-1,j,P_1-p_i,Q_{11},Q_{21})+F_1,\\ F(i-1,j,P_1,Q_{12},Q_{22})+F_2,\\ F(i,j-1,P_1-p_j,Q_{13},Q_{23})+F_3,\\ F(i,j-1,P_1,Q_{14},Q_{24})+F_4\end{array}\right\},\begin{pmatrix}\forall i\in\{1,\dots,n_1\}\\ \forall j\in\{n_1+1,\dots,n\}\\ \forall 0\le P_1\le P\\ \forall 0\le Q_1\le Q'_1\\ \forall 0\le Q_2\le Q'_2\end{pmatrix}$$

F_k definition ($1\le k\le 4$) is related to decision k and to the objective function. Q_{1k}, Q_{2k} are related to Q_1 and Q_2 and to decision k.

4.3. Applications of the general DP recursion

This DP algorithm can also be applied to the classical $P2||C_{max}$ problem as follows (we have simplified the general formulation):

$$F(0,0)=0 \qquad F(i,P_1)=+\infty,\begin{pmatrix}\forall i\in\{0,1,\dots,n\},\\ \forall(P_1<0\vee P_1>P)\end{pmatrix}$$

$$F(i,P_1)=\min\left\{\begin{array}{l}F(i-1,P_1-p_i)+p_i,\\ F(i-1,P_1),\end{array}\right\},\begin{pmatrix}\forall i\in\{1,\dots,n\}\\ \forall 0\le P_1\le P\end{pmatrix}$$

The value of the optimal solution is equal to $min_{P/2\le P_1\le P}\max(F(n,P_1),P-P_1)$. The solution is built by following a classical backtracking algorithm. The complexity of this DP algorithm is in $O(nP)$ (the same complexity as Rothkopf (1966)).

In the following, we present some implementations of this recursive formulation. We introduce the following notations: $P_{(i,j)}=\sum_{1\le k\le i}p_k+\sum_{n_1+1\le k\le j}p_k$. The quantity $P_{(i,j)}-P_1$ denotes the completion time of the jobs on $M2$. Furthermore, we assume that the jobs are numbered according to the WSPT rule, that is: $p_1/w_1\le p_2/w_2\le\dots\le p_{n_1}/w_{n_1}$ and $p_{n_1+1}/w_{n_1+1}\le p_{n_1+2}/w_{n_1+2}\le\dots\le p_n/w_n$ (we consider $w_j=1$, $j=1..n$, for $\sum C_j$ criterion).

We present in the following the application of the general DP formulation to the problems involving the following objective functions: $C_{\max}(\mathcal{N})$ and $C_{\max}(\mathcal{N}_1)$; $C_{\max}(\mathcal{N})$ and $\sum w_j C_j(\mathcal{N}_1)$; $\sum C_j(\mathcal{N})$ and $\sum C_j(\mathcal{N}_1)$; and $\sum w_j C_j(\mathcal{N})$ and $C_{\max}(\mathcal{N}_1)$.

Problems with $\sum C_j$ objective function

Let us consider for instance problem $P2||\varepsilon\big(\sum C_j(\mathcal{N}_1)/\sum C_j(\mathcal{N})\big)$. We have to minimize $\sum C_j(\mathcal{N}_1)$ and to respect the constraint that $\sum C_j(\mathcal{N}) \le \varepsilon$. We consider that Q_1 corresponds to $\sum C_j(\mathcal{N})$; $Q_2 = 0$ (omitted in the recursive relation). The objective function to minimize is $F(i,j,P_1,Q_1) = \sum C_j(\mathcal{N}_1)$. The general recursive relation becomes:

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

$$F(i,j,P_1,Q_1) = +\infty, \begin{pmatrix} \forall i > n_1, \\ \forall j \le n_1, \\ \forall (P_1,Q_1) \end{pmatrix}$$

$$F(i,j,P_1,Q_1) = +\infty, \begin{pmatrix} \forall i \in \{0,1,\dots,n_1\}, \\ \forall j \in \{n_1,n_1+1,\dots,n\}, \\ \forall ((P_1,Q_1) < (0,0) \vee (P_1,Q_1) > (P,\varepsilon)) \end{pmatrix}$$

$$F(i,j,P_1,Q_1) = \min \left\{ \begin{array}{l} F(i-1,j,P_1-p_i,Q_1-P_1)+P_1, \\ F(i-1,j,P_1,Q_1-w_i(P_{(i,j)}-P_1))+P_{(i,j)}-P_1, \\ F(i,j-1,P_1-p_j,Q_1-P_1), \\ F(i,j-1,P_1,Q_1-(P_{(i,j)}-P_1)) \end{array} \right\}, \begin{pmatrix} \forall i \in \{1,\dots,n_1\} \\ \forall j \in \{n_1+1,\dots,n\} \\ \forall 0 \le P_1 \le P \\ \forall 0 \le Q_1 \le \varepsilon \end{pmatrix}$$

The optimal solution is given by $min_{(0\le P_1 \le P \wedge 0 \le Q_1 \le \varepsilon)} F(n_1,n,P_1,Q_1)$. The running time of this algorithm is in $O(n^2 P\varepsilon)$. This method can be generalized for m machines and we obtain the following proposition.

Proposition 10. *An optimal solution to the problem $Pm||\varepsilon\big(\sum C_j(\mathcal{N}_1)/\sum C_j(\mathcal{N})\big)$ can be determined in $O(n^2 P^{m-1}\varepsilon)$.*

Let us consider now the problem $P2||GP(\sum C_j(\mathcal{N}),\sum C_j(\mathcal{N}_1))$, which is equivalent to finding a solution that respects $\sum C_j(\mathcal{N}) \le \varepsilon$ and $\sum C_j(\mathcal{N}_1) \le \varepsilon_1$. A feasible solution is given by $F(n_1,n,P_1,Q_1) \le \varepsilon_1$, $(0 \le P_1 \le P$ and $0 \le Q_1 \le \varepsilon)$. The running time is in $O(n^2 P\varepsilon)$.

Let us consider now the problem $P2||\varepsilon(\sum C_j(\mathcal{N})/\sum C_j(\mathcal{N}_1))$. We search for the smallest value of Q_1 such that there is a value of P_1 with $F(n_1,n,P_1,Q_1) \le \varepsilon$. We enumerate all possible values of (Q_1, P_1) from $(0,0)$ to $(n \times w_{max} \times P, P)$ where $w_{max} = \max_{j\in\mathcal{N}} w_j$. When (Q_1, P_1) are found so that $F(n_1, P_1, Q_1) \leq \varepsilon$, the algorithm stops and returns the current value of Q_1 which defines the minimum value of $\sum C_j(\mathcal{N}_1)$. If the algorithm does not return any feasible schedule, it returns 'no feasible schedule'. The running time is bounded in $O(n^3P^2w_{max})$. Note that a slight modification of the recursive relation could lead to a more interesting running time in $O(n^2P\varepsilon)$.

Similarly, for solving the problem $P2||F_\ell(\sum C_j(\mathcal{N}), \sum C_j(\mathcal{N}_1))$, we consider the general DP formulation with ε an upper bound to $\sum C_j(\mathcal{N})$. For instance $F_\ell(\sum C_j(\mathcal{N}), \sum C_j(\mathcal{N}_1)) = a\sum C_j(\mathcal{N}) + b\sum C_j(\mathcal{N}_1)$. We enumerate all possible values of (P_1, Q_1) from $(0,0)$ to $(P, nw_{max}P)$ and we return the solution with the minimum value of $a \times Q_1 + b \times F(n_1, n, P_1, Q_1)$. The running time is in $O(n^3P^2w_{max})$. Note that another algorithm can be done with a better running time $(O(n^2P))$ by reformulating $P2||a\sum C_j(\mathcal{N}) + b\sum C_j(\mathcal{N}_1)$ as $P2||\sum w_jC_j(\mathcal{N})$ ($w_j = a$ if $j \in \mathcal{N}\setminus\mathcal{N}_1$, and $w_j = a+b$ if $j \in \mathcal{N}_1$).

Problems with $\sum w_jC_j(\mathcal{N}_1)$ and $C_{\max}(\mathcal{N})$ objective functions

Let us consider the problem $P2||\varepsilon\big(\sum w_jC_j(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$. We have $C_{\max}(\mathcal{N}) \leq \varepsilon$ and we assume $\varepsilon < P$. We consider that Q_1 corresponds to the sum of processing times of the jobs of $\mathcal{N}_1$ on $M1$; and Q_2 is equal to 0 (omitted in the recursive relation). We propose $Q_1' = \sum_{j\in\mathcal{N}_1}$ as an upper bound of Q. Clearly, $Q_1' \leq P$. The objective function to minimize is $F(i,j,P_1) = \sum w_jC_j(\mathcal{N}_1)$. The makespan is given by $\max(P_1, P - P_1)$. The general recursive relation becomes:

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

$$F(i,j,P_1,Q_1) = +\infty, \begin{pmatrix} \forall i > n_1, \\ \forall j \leq n_1, \\ \forall (P_1, Q_1) \end{pmatrix}$$

$$F(i,j,P_1,Q_1) = +\infty, \begin{pmatrix} \forall i \in \{0,1,\ldots,n_1\}, \\ \forall j \in \{n_1, n_1+1, \ldots, n\}, \\ \forall ((P_1,Q_1) < (0,0) \vee (P_1,Q_1) > (\varepsilon, Q_1')) \end{pmatrix}$$

$$F(i,j,P_1,Q_1) = \min \left\{ \begin{array}{l} F(i-1,j,P_1-p_i,Q_1-p_i) + w_iQ_1, \\ F(i-1,j,P_1,Q_1) + w_i(P_{(i,0)} - Q_1), \\ F(i,j-1,P_1-p_j,Q_1), \\ F(i,j-1,P_1,Q_1) \end{array} \right\}, \begin{pmatrix} \forall i \in \{1,\ldots,n_1\}; \\ \forall j \in \{n_1+1,\ldots,n\}; \\ \forall \max(0, P-\varepsilon) \leq P_1 \leq \varepsilon \forall 0 \leq Q_1 \leq Q_1' \end{pmatrix}$$

The optimal solution is given by the smallest value of $F(n_1, n, P_1, Q_1)$ with $\max(P_1, P - P_1) \le \varepsilon$. The running time of this algorithm is in $O(n^2 P\varepsilon)$. This algorithm can be generalized to the case of m machines, we obtain the following proposition.

PROPOSITION 11. *An optimal solution to the problem $Pm||\varepsilon\big(\sum w_j C_j(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$ can be determined in $O(n^2 P^m \varepsilon^{m-1})$.*

By following the same approach as in the previous section, we can deduce the following results: problem $P2||GP(C_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1))$ can be solved in $O(n^2 P\varepsilon)$; problem $Pm||\varepsilon\big(C_{\max}(\mathcal{N})/\sum w_j C_j(\mathcal{N}_1)\big)$ can be solved in $O(n^2 P\varepsilon)$; and problem $P2||F_\ell(C_{\max}(\mathcal{N}), \sum w_j C_j(\mathcal{N}_1))$ can be solved in $O(n^2 P^2)$.

Problems with $\sum w_j C_j(\mathcal{N})$ and $C_{\max}(\mathcal{N}_1)$ objective functions

Let us consider the problem $P2||\varepsilon\big(\sum w_j C_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$. We consider that Q_1 and Q_2 correspond to $C_{\max}(\mathcal{N}_1)$ on $M1$ and $M2$ respectively (remember that P_1 is the makespan on $M1$). The objective function to minimize is $F(i, j, P_1, Q_1, Q_2) = \sum w_j C_j(\mathcal{N})$. The general recursive relation becomes:

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

$$F(i, j, P_1, Q_1, Q_2) = +\infty, \begin{pmatrix} \forall i > n_1, \\ \forall j < n_1, \\ \forall (P_1, Q_1, Q_2) \end{pmatrix}$$

$$F(i, j, P_1, Q_1, Q_2) = +\infty, \begin{pmatrix} \forall i \in \{0, 1, \ldots, n_1\}, \\ \forall j \in \{n_1, n_1 + 1, \ldots, n\}, \\ \forall ((P_1, Q_1, Q_2) < (0, 0, 0) \vee (P_1, Q_1, Q_2) > (P, \varepsilon, \varepsilon)) \end{pmatrix}$$

$$F(i, j, P_1, Q_1, Q_2) = \min \left\{ \begin{array}{l} F(i-1, j, P_1 - p_i, P_1 - p_i, Q_2) + w_i P_1, \\ F(i-1, j, P_1, Q_1, P_{(i,j)} - P_1 - p_i) + w_i(P_{(i,j)} - P_1), \\ F(i, j-1, P_1 - p_j, Q_1, Q_2) + w_j P_1, \\ F(i, j-1, P_1, Q_1, Q_2) + w_j(P_{(i,j)} - P_1) \end{array} \right\}, \begin{pmatrix} \forall i \in \{1, \ldots, n_1\} \\ \forall j \in \{n_1 + 1, \ldots, n\} \\ \forall 0 \le P_1 \le P \\ \forall 0 \le Q_1 \le \varepsilon \\ \forall 0 \le Q_2 \le \varepsilon \end{pmatrix}$$

The optimal solution is given by $min_{(0 \le P_1 \le P \wedge 0 \le Q_1 \le \varepsilon \wedge 0 \le Q_2 \le \varepsilon)} F(n_1, n, P_1, Q)$. The running time of this algorithm is in $O(n^2 P\varepsilon^2)$. In the case of m machines, we obtain the following proposition.

PROPOSITION 12. *An optimal solution of problem $Pm||\varepsilon\big(\sum w_j C_j(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ is determined in $O(n^2 P^{m-1} \varepsilon^m)$.*

By following the same approach as in the previous section, we can deduce the following results: problem $P2||GP(C_{\max}(\mathcal{N}_1), \sum w_j C_j(\mathcal{N}))$ can be solved in $O(n^2 P \varepsilon^2)$; problem $Pm||\varepsilon\big(C_{\max}(\mathcal{N}_1)/\sum w_j C_j(\mathcal{N})\big)$ can be solved in $O(n^2 P \varepsilon^2)$; and problem $P2||F_l(C_{\max}(\mathcal{N}_1), \sum w_j C_j(\mathcal{N}))$ can be solved in $O(n^2 P^3)$.

Problem $P2||\varepsilon(C_{\max}(\mathcal{N})/C_{\max}(\mathcal{N}_1))$

Let us consider the problem $P2||\varepsilon\big(C_{\max}(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$. We consider that Q_1 and Q_2 correspond to $C_{\max}(\mathcal{N}_1)$ on $M1$ and $M2$ respectively (remember that P_1 is the makespan on $M1$). The objective function to minimize is $F(i, j, P_1, Q_1, Q_2)$ which corresponds to the makespan on $M2$. The general recursive relation becomes:

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

$$F(i, j, P_1, Q_1, Q_2) = +\infty, \begin{pmatrix} \forall i > n_1, \\ \forall j < n_1, \\ \forall (P_1, Q_1, Q_2) \end{pmatrix}$$

$$F(i, j, P_1, Q_1, Q_2) = +\infty, \begin{pmatrix} \forall i \in \{0, 1, \ldots, n_1\}, \\ \forall j \in \{n_1, n_1 + 1, \ldots, n\}, \\ \forall ((P_1, Q_1, Q_2) < (0, 0, 0) \vee (P_1, Q_1, Q_2) > (P, \varepsilon, \varepsilon)) \end{pmatrix}$$

$$F(i, j, P_1, Q_1, Q_2) = \min \left\{ \begin{array}{l} F(i-1, j, P_1 - p_i, P_1 - p_i, Q_2), \\ F(i-1, j, P_1, Q_1, P_{(i,j)} - P_1 - p_i) + p_i, \\ F(i, j-1, P_1 - p_j, Q_1, Q_2), \\ F(i, j-1, P_1, Q_1, Q_2) + p_j \end{array} \right\}, \begin{pmatrix} \forall i \in \{1, \ldots, n_1\} \\ \forall j \in \{n_1 + 1, \ldots, n\} \\ \forall 0 \le P_1 \le P \\ \forall 0 \le Q_1 \le \varepsilon \\ \forall 0 \le Q_2 \le \varepsilon \end{pmatrix}$$

The optimal solution is given by $min_{(0 \le P_1 \le P \wedge 0 \le Q_1 \le \varepsilon \wedge 0 \le Q_2 \le \varepsilon)} F(n_1, n, P_1, Q)$. The running time of this algorithm is in $O(n^2 P \varepsilon^2)$. Generalizing to m machines, we obtain the following proposition.

Proposition 13. *An optimal solution of problem* $Pm||\varepsilon\big(C_{\max}(\mathcal{N}_1)/C_{\max}(\mathcal{N})\big)$ *is determined in* $O(n^2 P^{m-1} \varepsilon^{2m})$.

By following the same approach as in the previous section, we can deduce the following results: problem $P2||GP(C_{\max}(\mathcal{N}_1), C_{\max}(\mathcal{N}))$ can be solved in $O(n^2 P \varepsilon^2)$; problem $P2||\varepsilon\big(C_{\max}(\mathcal{N})/C_{\max}(\mathcal{N}_1)\big)$ can be solved in $O(n^2 P \varepsilon^2)$; problem $P2||F_l(C_{\max}(\mathcal{N}_1), C_{\max}(\mathcal{N}))$ can be solved in $O(n^2 P^3)$.

5. Conclusion

In this paper, we consider a new family of scheduling problems in the area of interfering job sets or multi-agent scheduling. One subset of jobs ($\mathcal{N}_1 \subseteq \mathcal{N}$) is in competition with the whole set of jobs and a compromise solution has to be found. We first show the difference between these problems and multi-agent scheduling problems and explain why this approach generalizes typical multicriteria scheduling problems. We consider the problem of scheduling independent jobs on a single machine or on identical parallel machines, without additional constraints.

New complexity results are given for single machine problems depending on the objective function (linear combination of criteria or ε-constraint approach). These results can be easily extended for goal programming and enumerative approaches. Concerning identical parallel machine problems, we propose new complexity results and a general dynamic programming formulation. This algorithm is presented for some two-parallel machine problems. This DP formulation can be easily generalized for m identical or uniform parallel machines and for several subsets of jobs. Table 1 summarizes the results presented in the paper for single machine problems and Table 2 for parallel machine problems.

Objective functions	Studied approaches (F_ℓ)	(ε)	(GP)	Reference
$C_{\max}(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Prop. 1
$C_{\max}(\mathcal{N})$, $L_{max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Prop. 1
$C_{\max}(\mathcal{N})$, $\sum w_j C_j(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Prop. 1
$L_{max}(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Prop. 2
$L_{\max}(\mathcal{N})$, $L_{\max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Prop. 2
$L_{\max}(\mathcal{N})$, $\sum C_j(\mathcal{N}_1)$	Open	Open	Open	Prop. 8
$L_{\max}(\mathcal{N})$, $\sum w_j C_j(\mathcal{N}_1)$	Strongly NP-hard	Strongly NP-hard	Strongly NP-hard	Prop. 4
$\sum C_j(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$	Polynomial	Polynomial	Polynomial	Prop. 2
$\sum C_j(\mathcal{N})$, $L_{\max}(\mathcal{N}_1)$	Open	Open	Open	Prop. 8
$\sum C_j(\mathcal{N})$, $\sum C_j(\mathcal{N}_1)$	Polynomial	Binary NP-Hard	Binary NP-Hard	Prop. 6, 1
$\sum w_j C_j(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$	Open	Binary NP-Hard	Binary NP-Hard	Prop. 3
$\sum w_j C_j(\mathcal{N})$, $L_{\max}(\mathcal{N}_1)$	Strongly NP-hard	Strongly NP-hard	Strongly NP-hard	Prop. 4
$\sum w_j C_j(\mathcal{N})$, $\sum w'_j C_j(\mathcal{N}_1)$	Polynomial	Strongly NP-Hard	Strongly NP-Hard	Prop. 1, 7

(F_ℓ): linear combination of criteria
(ε): ε-constraint approach
(GP): goal programming approach

Table 1 Some new complexity results on interfering-jobs single-machine scheduling problems

Objective functions	Studied approaches				Complexity	Reference
	(F_l)	(ε)	(GP)	(DP)		
$\sum C_j(\mathcal{N})$, $\sum C_j(\mathcal{N}_1)$	X	X	X	-	Binary NP-Hard	Prop. 9,10
$\sum w_j C_j(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$	X	X	X	-	Binary NP-Hard	Prop. 9,12
$C_{\max}(\mathcal{N})$, $\sum w_j C_j(\mathcal{N}_1)$	X	X	X	-	Binary NP-Hard	Prop. 9,11
$C_{\max}(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$	X	X	X	-	Binary NP-Hard	Prop. 9,13
$\sum C_j(\mathcal{N})$, $\sum C_j(\mathcal{N}_1)$, $\sum C_j(\mathcal{N}_2)$, ...	X	X	X	-	Binary NP-Hard	Prop. 9,10
$\sum w_j C_j(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$, $C_{\max}(\mathcal{N}_2)$, ...	X	X	X	-	Binary NP-Hard	Prop. 9,12
$\sum w_j C_j(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$, $C_{\max}(\mathcal{N}_2)$, ...	X	X	X	-	Binary NP-Hard	Prop. 9,12
$C_{\max}(\mathcal{N})$, $\sum C_j(\mathcal{N}_1)$, $\sum C_j(\mathcal{N}_2)$, ...	X	X	X	-	Binary NP-Hard	Prop. 9,11
$C_{\max}(\mathcal{N})$, $C_{\max}(\mathcal{N}_1)$, $C_{\max}(\mathcal{N}_2)$, ...	X	X	X	-	Binary NP-Hard	Prop. 9,13

(F_l): linear combination of criteria
(ε): ε-constraint approach
(GP): goal programming approach
(DP): Can be solved by the proposed DP algorithm

Table 2 New complexity results on interfering-jobs parallel-machine scheduling problems

This category of problems leads to a wide area of research problems. Some complexity results remain open; some approximation schemes can be constructed for these problems as well as exact and approximated heuristic algorithms.

References

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

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

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

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

Balasubramanian H., Fowler J., Keha A., Pfund M. (2009). Scheduling interfering job sets on parallel machines. *European Journal of Operational Research*, In Press, Accepted Manuscript, Available online 19 November 2008 .

Bruno J., Coffman E.G., Jr., and R. Sethi. (1974) Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, **17**, pp. 382–387.

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

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

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

Garey M.R. et Johnson D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, San Francisco.

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

Huynh Tuong N., Soukhal A. and Billaut J.-C. (2008) Ordonnancement des travaux interfrant dans un flowshop deux machines (in French). 7e Conférence Francophone de MOdlisation et SIMulation (MOSIM'08), Paris (France), ISBN 978-2-7430-1057-7.

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

E. L. Lawler (1977). A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics* **1**, pp. 331–342.

Lenstra J.K., Rinnooy Kan A.H.G., and Brucker P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics* **1**, pp. 343–362.

Mocquillon C., Lenté C., T'Kindt V. (2006). Solution of a multicriteria shampoo production problem. IEEE International Conference on Service Systems and Service Management (IEEE-SSSM'06), pp. 907–911, Troyes (France).

Ng C. T., Cheng T. C. E., and 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**, pp. 387–394.

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

C. Pessan, J-L. Bouquard, E. Neron (2008). Genetic Branch-and-Bound or Exact Genetic Algorithm? *Lecture Notes in Computer Science*, Volume 4926, Springer Berlin / Heidelberg, pp. 136-147.

Rothkopf M.H. (1966). Scheduling independant tasks on parallel processors *Management Science*, **12**, pp. 438–447.

Soukhal A., Huynh Tuong N. and Dao Z (2008), Parallel Machine Scheduling with Interfering Jobs, 8th Internat. Conference on Multiple Objective and Goal Programming (MOPGP'08), Portsmouth (UK).

T'kindt V. and Billaut J-C. (2006). *Multicriteria Scheduling: Theory, Models and Algorithms.* 2nd edition, Springer.

Vestjens A.P.A., Wennink M. and Woeginger G.J. (2007). Complexity of the job insertion problem in multi–stage scheduling. *Operations Research Letters* **35**, pp. 754–758.

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