A dominant class of schedules for malleable jobs in the problem to minimize the total weighted completion time

Ruslan Sadykov[*]

INRIA Bordeaux — Sud-Ouest, France

Institut de Mathématique de Bordeaux,

351 cours de la Libération, 33405 Talence, France

[*]Corresponding author. Fax: +33 5 40 00 21 17. E-mail: `Ruslan.Sadykov@inria.fr`

Abstract

This paper is about scheduling parallel jobs, i.e. which can be executed on more than one machine at the same time. Malleable jobs is a special class of parallel jobs. The number of machines a malleable job is executed on may change during its execution.

In this work, we consider the NP-hard problem of scheduling malleable jobs to minimize the total weighted completion time (or mean weighted flow time). For this problem, we introduce the class of "ascending" schedules in which, for each job, the number of machines assigned to it cannot decrease over time while this job is being processed.

We prove that, under a natural assumption on the processing time functions of jobs, the set of ascending schedules is dominant for the problem. This result can be used to reduce the search space while looking for an optimal solution.

Keywords: combinatorial optimization; parallel scheduling; total completion time; malleable jobs.

1 Introduction

With the emergence of new production, communication and parallel computing system, the usual scheduling requirement that a job is executed only on one processor has become, in many cases, obsolete and unfounded. Therefore, parallel jobs scheduling is becoming more and more widespread.

The malleable jobs is a special class of the parallel jobs. The number of processors assigned to a malleable job may change during its execution. For an overview of malleable jobs scheduling, we refer to [5], [6], and [9], for an application in parallel optimization, to [1], and for an application in textile industry to [13]. Rapine et al. [12] considered on-line scheduling of malleable jobs.

Recently, Blazewicz et al. [2] studied the problem of scheduling malleable jobs to minimize the makespan. They presented the procedure which converts an optimal solution for the relaxed problem, in which the number of processors allocated to a task is not required to be integer, into an optimal solution for the original problem in $O(n)$ time. Caramia and Drozdowski [4] considered the problem of scheduling malleable jobs with special processing time functions to minimize the total completion time. They showed that, once the order in which jobs are completed is fixed, an optimal schedule can be determined in polynomial time by solving a linear program. Another polynomial algorithm was presented for a restricted case with agreeable jobs. In the general, not agreeable case, this algorithm becomes an approximation algorithm with performance ratio at most 2.

We now define the problem which we consider in this paper. A set $N = \{1, \ldots, n\}$ of preemptive jobs should be processed on a set $M = \{1, \ldots, m\}$ of identical machines. $p_j(q)$ is the processing time of a job j if executed continuously on q machines, $1 \le q \le m$. For presentation purposes, we set $p_j(0) = \infty$.

δ_j is the maximum number of machines job j can occupy at any time moment.

Any schedule π is a sequence of a set $K(\pi)$ of intervals in which the number of machines assigned to each job does not change. For an interval $I_l \in K(\pi)$ we denote as $d_l(\pi)$ its length, and as $q_{lj}(\pi)$ the number of machines occupied by job j in I_l, $q_{lj}(\pi) \leq \delta_j$. Let $H_j(\pi)$ be the *processed part* of job j in this schedule:

$$H_j(\pi) = \sum_{I_l \in K(\pi)} \frac{d_l(\pi)}{p_j(q_{lj}(\pi))}, \tag{1}$$

where $d_l(\pi)/p_j(0) = 0$ for any $I_l \in K(\pi)$ and any $j \in N$. In a feasible schedule π, we should have $H_j(\pi) = 1$ for any job j.

We denote by $C_j(\pi)$ the completion time of job j in schedule π. The objective is to find a schedule π which minimizes the total weighted completion time

$$F(\pi) = \sum_{j \in N} w_j C_j(\pi). \tag{2}$$

Note that function (2) is equivalent to the mean weighted flow time, as all the jobs available for processing from time 0. In the standard scheduling notation, the problem is denoted as $P|var,\ \delta_j|\sum w_j C_j$, see [5].

The problem we consider is a generalisation of the problem to schedule preemptive non-parallel jobs ($\delta_j = 1,\ \forall j \in N$) on identical parallel machines to minimize the total weighted completion time, denoted as $P|pmtn|\sum w_j C_j$. McNaughton [10] showed that there is always an optimal schedule without preemption, i.e. this problem is equivalent to $P||\sum w_j C_j$. As the latter problem is NP-hard in the strong sense, see problem SS13 in Garey and Johnson [7], our problem is also NP-hard in the strong sense.

The contribution of this paper is an important dominance rule for the problem. We will show that, under a natural assumption, there exists an optimal schedule in which, for each job, the number of processors assigned to it does not decrease over time while this job is being processed. We say that such schedules have the *ascending property.* This result can be used to reduce the search for an optimal schedule of the problem to this class of dominant schedules.

The structure of the paper is the following. The result is proved in Section 2. In subsection 2.1, we present the construction of a family of schedules for a given schedule. This construction will be used to prove the dominance rule. In subsection 2.2, the assumption is introduced. Subsection 2.3 contains the main theorem which proves the result. Conclusions are drawn in Section 3.

2 Ascending property

We first introduce some definitions.

Definition 1. A *piece* of a job within a given schedule is a maximal non-preemptive part of it processed on one machine.

The completion and starting time of a piece u of a job j in a schedule π will be denoted as $C_j^u(\pi)$ and $S_j^u(\pi)$. When there is no ambiguity, we will use just C_j^u and S_j^u. The completion time $C_j(\pi)$ of a job j in a schedule π is equal to the maximum completion time for all pieces of job j.

Definition 2. A piece of a job in a schedule is *terminal* if its completion time is equal to the completion time of the job it belongs to. A non-terminal piece is *early.*

Definition 3. A schedule is *ascending* (or has the ascending property) if it does not contain early pieces (all its pieces are terminal). In other worlds, an ascending schedule does not use preemption.

An illustration of the definitions is presented in Figure 1, where pieces s and v are early and all other pieces are terminal. Thus, the schedule in Figure 1 is not ascending.

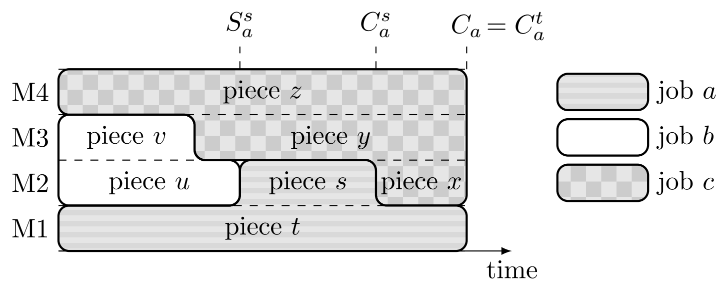


Figure 1: Illustration of the definitions

2.1 A family of schedules

We consider a non-ascending feasible schedule π. Then, there is an early piece in π. Let s be an early piece with the latest completion time among all early pieces of π, and let a be the corresponding job. See an example in Figure 1. Also, for a job $j \in N$, let V_j be the set of pieces of job j completed strictly after $C_a^s(\pi)$ in π, and let U_j be the set of pieces of j started at $C_a^s(\pi)$ or after and completed strictly after $C_a^s(\pi)$ in π, $U_j \subseteq V_j$. Note that, by the choice of piece s, the pieces in V_j are terminal in π for any job $j \in N$. We denote as $|V_j|$ and $|U_j|$ the number of jobs in V_j and U_j.

We will now construct a family of schedules $\pi(\varepsilon)$. Basically, in each schedule $\pi(\varepsilon)$, the length of piece s of job a is changed by $\varepsilon \in \mathbb{R}$, and this change is "compensated" by a change of the completion and starting times of other pieces. This transformation is carried out in such a way that

(i) the starting time of each piece is not larger than its completion time,

(ii) if two pieces of a same job do not overlap in π then, in $\pi(\varepsilon)$, they also do not overlap or one piece immediately precedes the other,

(iii) if two pieces of a same job overlap in π then, in $\pi(\varepsilon)$, they also overlap or one piece immediately precedes the other,

(iv) terminal pieces in π are also terminal in $\pi(\varepsilon)$.

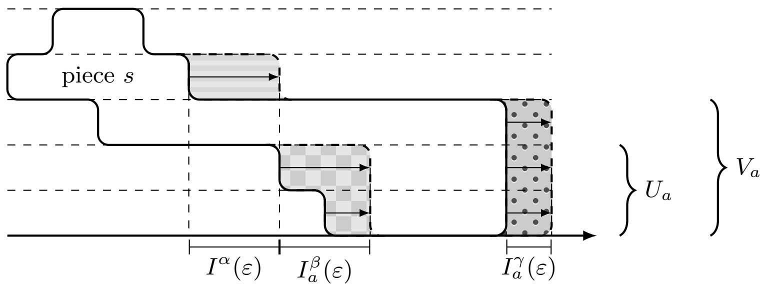

piece s
U_a
V_a
$I^{\alpha}(\varepsilon)$
$I_a^{\beta}(\varepsilon)$
$I_a^{\gamma}(\varepsilon)$

Figure 2: Job a in schedule π and its changes in schedule $\pi(\varepsilon)$

A schedule $\pi(\varepsilon)$ which satisfies these conditions will be called *admissible*.

A formal construction of admissible schedules $\pi(\varepsilon)$ will be now presented. In $\pi(\varepsilon)$, the starting and completion times of each piece u of each job j are changed by $\Delta S_j^u(\varepsilon)$ and $\Delta C_j^u(\varepsilon)$ with respect to schedule π. In $\pi(\varepsilon)$, the completion time of each job j are changed by $\Delta C_j(\varepsilon)$ with respect to schedule π. Note that these functions can take negative values.

For a fixed ε, values $\Delta S_j^u(\varepsilon)$ and $\Delta C_j^u(\varepsilon)$ are computed in the following way. Value $\Delta C_a^s(\varepsilon)$ is equal to ε. For each job j and $u \notin U_j$, $\Delta S_j^u(\varepsilon) = 0$. For each job j and $u \notin V_j \cup \{s\}$, $\Delta C_j^u(\varepsilon) = 0$. For each job j and each piece $u \in U_j$, change $\Delta S_j^u(\varepsilon)$ of its starting time is equal to change $\Delta C_i^v(\varepsilon)$ of the completion time of the piece v of job i which immediately precedes u in π (such piece always exist, as $S_j^u(\pi) \geq C_a^s(\pi)$).

For each job j and each $v \in V_j$, values $\Delta C_j^v(\varepsilon)$ are equal to $\Delta C_j(\varepsilon)$ in order to keep pieces in V_j terminal. For each job j, value $\Delta C_j(\varepsilon)$ should be taken in such a way that the processed part $H_j(\pi(\varepsilon))$ defined in (1) remains equal to one. Informally speaking, $\Delta C_j(\varepsilon)$ should "compensate" changes ΔS_j^u, $u \in U_j$, and change ΔC_s^a if $j = a$. See an illustration for job a in Figure 2.

We will now introduce additional notations which will allow us to determine values $\Delta C_j(\varepsilon)$. For a job $j \in N$ such that $U_j \neq \emptyset$ or $j = a$, let

$$I^{\alpha}(\varepsilon) = [\alpha^-(\varepsilon), \alpha^+(\varepsilon)] = \Big[\min_{t \in A(\varepsilon)} t,\ \max_{t \in A(\varepsilon)} t\Big],\ \text{where } A(\varepsilon) = \big\{C_s^a(\pi), C_s^a(\pi) + \varepsilon\big\}.$$

$$I_j^{\beta}(\varepsilon) = [\beta_j^-(\varepsilon), \beta_j^+(\varepsilon)] = \Big[\min_{t \in B_j(\varepsilon)} t,\ \max_{t \in B_j(\varepsilon)} t\Big],\ \text{where } B_j(\varepsilon) = \bigcup_{u \in U_j} \big\{S_j^u(\pi), S_j^u(\pi) + \Delta S_j^u(\varepsilon)\big\}.$$

$$I_j^{\gamma}(\varepsilon) = [\gamma_j^-(\varepsilon), \gamma_j^+(\varepsilon)] = \Big[\min_{t \in \Gamma_j(\varepsilon)} t,\ \max_{t \in \Gamma_j(\varepsilon)} t\Big],\ \text{where } \Gamma_j(\varepsilon) = \big\{C_j(\pi), C_j(\pi) + \Delta C_j(\varepsilon)\big\}.$$

See an illustration of these intervals in Figure 2. If $U_a = \emptyset$ then $I_a^{\beta}(\varepsilon) = \emptyset$.

We fix a value $\mathcal{M}$ which is greater than the makespan (maximum completion time) of any schedule without idle time. We can set $\mathcal{M} = \sum_{j \in N} \max_{q=1}^{\delta_j} p_j(q) + 1$. Let $\pi^{\alpha}(\varepsilon)$ be the schedule in which the starting and completion times of all pieces are the same as in $\pi(\varepsilon)$ except that

- pieces in U_a are removed,
- the completion time of all pieces in $V_a \setminus U_a$ is equal to $\mathcal{M}$.

Let $\pi^{\beta}(\varepsilon)$ be the schedule in which the starting and completion times of all pieces are the same as in $\pi(\varepsilon)$ except that, for each piece $j \in N$,

- pieces which are not in V_j are removed.
- the completion time of all pieces in V_j is equal to $\mathcal{M}$,

Let $\pi^{\gamma}(\varepsilon)$ be the schedule in which the starting and completion times of all pieces are the same as in $\pi(\varepsilon)$ except that, for each piece $j \in N$,

- pieces which are not in V_j are removed.
- the starting time of all pieces in V_j is equal to 0,

Let $H_j^{[t^-,t^+]}(\pi)$ be the processed part of job $j \in N$ in interval $[t^-, t^+]$ in schedule π. It is formally defined as $H_j(\pi)$ in (1) except that set $K(\pi)$ of intervals is limited to interval $[t^-, t^+]$. Thus, $H_j(\pi) = H_j^{[0,\mathcal{M}]}(\pi)$.

Let also $\Delta H_j(\varepsilon) = H_j(\pi(\varepsilon)) - H_j(\pi)$, and for a job $j \in N$ such that $U_j \neq \emptyset$ or $j = a$, let

$$\Delta Z^{\alpha}(\varepsilon) = H_a^{[\alpha^-(\varepsilon),\alpha^+(\varepsilon)]}(\pi^{\alpha}(\varepsilon)) - H_a^{[\alpha^-(\varepsilon),\alpha^+(\varepsilon)]}(\pi^{\alpha}(0)),$$

$$\Delta Z_j^{\beta}(\varepsilon) = H_j^{[\beta_j^-(\varepsilon),\beta_j^+(\varepsilon)]}(\pi^{\beta}(\varepsilon)) - H_j^{[\beta_j^-(\varepsilon),\beta_j^+(\varepsilon)]}(\pi^{\beta}(0)),$$

$$\Delta Z_j^{\gamma}(\varepsilon) = H_j^{[\gamma_j^-(\varepsilon),\gamma_j^+(\varepsilon)]}(\pi^{\gamma}(\varepsilon)) - H_j^{[\gamma_j^-(\varepsilon),\gamma_j^+(\varepsilon)]}(\pi^{\gamma}(0)).$$

If $U_a = \emptyset$, then $\Delta Z_a^{\beta}(\varepsilon) = 0$. Note that schedule π is equivalent to $\pi(0)$.

There are three types of changes to do with job a in order to pass from schedule $\pi(0)$ to schedule $\pi(\varepsilon)$: change of $C_a^s(\pi)$, changes of $S_a^u(\pi)$, $u \in U_a$, and changes of C_a^v, $v \in V_a$. These three changes can be done consecutively, using two intermediate schedules. Schedule $\pi(\varepsilon)$ is admissible, and thus conditions (i) and (ii) are satisfied. Therefore, there always exists an order in which these three types of changes can be made such that

- when performing change $\Delta C_a^s(\varepsilon)$, in interval $I^{\alpha}(\varepsilon)$, the assignment of job a to machines is the same in $\pi^{\alpha}(0)$ and before the change, and the same in $\pi^{\alpha}(\varepsilon)$ and after the change;
- when performing changes $\Delta S_a^u(\varepsilon)$, $u \in U_a$, in interval $I^{\beta}(\varepsilon)$, the assignment of job a to machines is the same in $\pi^{\beta}(0)$ and before the changes, and the same in $\pi^{\beta}(\varepsilon)$ and after the changes;
- when performing changes $\Delta C_a^v(\varepsilon)$, $v \in V_a$, in interval $I^{\gamma}(\varepsilon)$, the assignment of job a to machines is the same in $\pi^{\gamma}(0)$ and before the changes, and the same in $\pi^{\gamma}(\varepsilon)$ and after the changes.

From this reasoning, it follows that

$$\Delta H_a(\varepsilon) = \Delta Z^{\alpha}(\varepsilon) + \Delta Z_a^{\beta}(\varepsilon) + \Delta Z_a^{\gamma}(\varepsilon). \qquad (3)$$

For jobs $j \in N$, $j \neq a$ and $U_j \neq \emptyset$, the reasoning is the same except that there is no change of $C_a^s(\pi)$. Then,

$$\Delta H_j(\varepsilon) = \Delta Z_j^{\beta}(\varepsilon) + \Delta Z_j^{\gamma}(\varepsilon), \quad \forall j \in N, j \neq a, U_j \neq \emptyset. \tag{4}$$

In order to keep schedule $\pi(\varepsilon)$ feasible, $\Delta H_j(\varepsilon)$ should be equal to 0. From (1), for each $j \in N$, we have $\Delta Z_j^{\gamma}(\varepsilon) = \Delta C_j(\varepsilon)/p_j(|V_j|)$. Therefore, using (3) and (4), we have

$$\Delta C_j^u(\varepsilon) = \begin{cases} -p_a\big(|V_a|\big) \cdot \big(\Delta Z_j^{\beta}(\varepsilon) + \Delta Z^{\alpha}(\varepsilon)\big), & j = a, u \in V_a, \\ \varepsilon, & j = a, u = s, \\ -p_j\big(|V_j|\big) \cdot \Delta Z_j^{\beta}(\varepsilon), & j \neq a, U_j \neq \emptyset, u \in V_j, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

In practice, for a fixed ε, values $\Delta S_j^u(\varepsilon)$ and $\Delta C_j^u(\varepsilon)$ should be computed in a non-decreasing order of values $S_j^u(\pi)$ and $C_j^u(\pi)$ in order to avoid cycles in the computation.

We will now determine values ε, for which schedule $\pi(\varepsilon)$ is admissible. By condition (i), for each piece, its starting time does not exceed its completion time:

$$\begin{aligned} S_j^u(\pi) + \Delta S_j^u(\varepsilon) \leq C_j^u(\pi) + \Delta C_j^u(\varepsilon), \\ \forall j \in N, \forall u. \end{aligned} \tag{6}$$

Let $\underline{\varepsilon}_1 < 0$ be the smallest value and $\overline{\varepsilon}_1 > 0$ be the largest value such that (6) is satisfied for all $\varepsilon \in \big[\underline{\varepsilon}_1, \overline{\varepsilon}_1\big]$.

By condition (ii), if a piece u of j precedes another piece v of j in π, u still precedes v in $\pi(\varepsilon)$:

$$\begin{gathered} C_j^u(\pi) + \Delta C_j^u(\varepsilon) \le S_j^v(\pi) + \Delta S_j^v(\varepsilon), \\ \forall j \in N, \forall u, v : C_j^u(\pi) \le S_j^v(\pi). \end{gathered} \tag{7}$$

Let $\underline{\varepsilon}_2 \le 0$ be the smallest value and $\overline{\varepsilon}_2 \ge 0$ be the largest value such that (7) is satisfied for all $\varepsilon \in \left[\underline{\varepsilon}_2, \overline{\varepsilon}_2\right]$.

The condition (iii) concerning two overlapping pieces of a same job in π is satisfied as long as

$$\begin{gathered} C_j^u(\pi) + \Delta C_j^u(\varepsilon) \ge S_j^v(\pi) + \Delta S_j^v(\varepsilon), \\ \forall j \in N, \forall u, v : S_j^u(\pi) \le S_j^v(\pi) < C_j^u(\pi). \end{gathered} \tag{8}$$

Let $\underline{\varepsilon}_3 < 0$ be the smallest value and $\overline{\varepsilon}_3 > 0$ be the largest value such that (8) is satisfied for all $\varepsilon \in \left[\underline{\varepsilon}_3, \overline{\varepsilon}_3\right]$.

The condition (iv) is satisfied, i.e. terminal pieces in π remain terminal in $\pi(\varepsilon)$, as long as

$$\begin{gathered} C_j^u(\pi) + \Delta C_j^u(\varepsilon) \le C_j^v(\pi) + \Delta C_j^v(\varepsilon), \\ \forall j \in N, \forall u, v : \ u \notin V_j, \ v \in V_j. \end{gathered} \tag{9}$$

Let $\underline{\varepsilon}_4 < 0$ be the smallest value and $\overline{\varepsilon}_4 > 0$ be the largest value such that (9) is satisfied for all $\varepsilon \in \left[\underline{\varepsilon}_4, \overline{\varepsilon}_4\right]$.

Note that $\underline{\varepsilon}_2$ and $\overline{\varepsilon}_2$ can be equal to zero. In this situation, the completion time of on piece and the starting time of another piece of the same job coincide in π. Other values ε_1, ε_3 and ε_4 are non-zero because of (6), (8) and (9) correspondingly and the fact that functions $\Delta S_j^u(\varepsilon)$ and $\Delta C_j^u(\varepsilon)$ are continuous. The continuity follows from (5) and the definition of functions $\Delta H_j^{[t^-, t^+]}$ and ΔZ.

Let us denote

$$\underline{\varepsilon} = \max\{\underline{\varepsilon}_1, \underline{\varepsilon}_2, \underline{\varepsilon}_3, \underline{\varepsilon}_4\} \leq 0 \quad \text{and}$$
$$\overline{\varepsilon} = \min\{\overline{\varepsilon}_1, \overline{\varepsilon}_2, \overline{\varepsilon}_3, \overline{\varepsilon}_4\} \geq 0.$$

We can conclude that, as long as $\varepsilon \in [\underline{\varepsilon}, \overline{\varepsilon}]$, schedule $\pi(\varepsilon)$ is admissible. Interval $[\underline{\varepsilon}, \overline{\varepsilon}]$ is not empty, as schedule $\pi(0)$ is equivalent to π, which is trivially admissible.

Also, if $\varepsilon \in [\underline{\varepsilon}, \overline{\varepsilon}]$, schedule $\pi(\varepsilon)$ is feasible, as it satisfies (5), and conditions (i) and (ii). The latter condition suffices to see that the number of pieces of job $j \in N$ executed simultaneously does not exceed δ_j, as $\pi(0)$ is feasible.

Example. To illustrate the family of schedules introduced, we consider an example for the *work preserving case* [5], in which, the processing time function for every job $j \in N$ is $p_j(q) = p_j(1)/q$. In this case, in any feasible schedule, the "surface" of each job remains the same, and functions $\Delta C_j(\varepsilon)$ are computed as

$$\Delta C_j^v(\varepsilon) = \begin{cases} \dfrac{\sum_{u \in U_a} \Delta S_j^u(\varepsilon) - \varepsilon}{|V_a|}, & j = a \text{ and } v \in V_a, \\ \varepsilon, & j = a \text{ and } v = s, \\ \dfrac{\sum_{u \in U_j} \Delta S_j^u(\varepsilon)}{|V_j|}, & j \neq a \text{ and } v \in V_j, \\ 0, & \text{otherwise.} \end{cases}$$

An example of changes $\Delta S(\varepsilon)$ and $\Delta C(\varepsilon)$ for a fixed $\varepsilon = \underline{\varepsilon} = \underline{\varepsilon}_1$ is illustrated in Figure 3. There are one early piece of job a and three early pieces of job f in schedule $\pi = \pi(0)$. As the completion time of the early piece of job a is the largest, this piece is chosen as piece s. Note that, schedule $\pi(-4)$ has one early piece less than schedule $\pi(0)$.

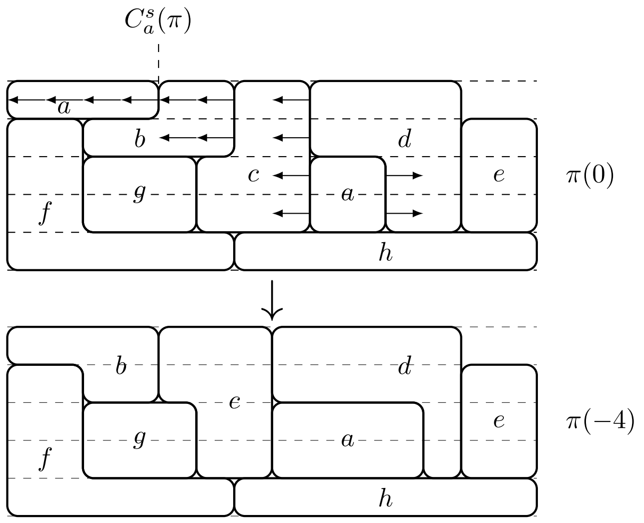


Figure 3: An example of transformation ($\varepsilon = \underline{\varepsilon} = \underline{\varepsilon}_1 = -4$) for the work preserving case.

2.2 A natural assumption

Suppose that, the *processing speed* of any job is an *non-decreasing* and *concave* function of the number of processors allocated to this job:

$$\forall j \in N,\ \forall q \in \{1, \ldots, \delta_j - 1\}, \quad \frac{1}{p_j(q)} - \frac{1}{p_j(q-1)} \geq \frac{1}{p_j(q+1)} - \frac{1}{p_j(q)} \geq 0, \quad (10)$$

where $1/p_j(0) = 0$. This assumption can be interpreted as "the more machines are allocated to a job, the less is the gain in the processing speed of this job". Note that the work preserving processing time function mentioned above satisfies inequalities (10).

The assumption on the concavity of the processing speed functions is quite natural and can be encountered in the literature, for example in [3]. Note also that assumption (10) is related to the monotonous penalty assumption [6, 11].

The latter assumes that, for any job $j \in N$ and for all $q \in \{1, \ldots, \delta_j - 1\}$,

$$p_j(q) \geq p_j(q+1) \text{ and } q \cdot p_j(q) \leq (q+1) \cdot p_j(q+1). \tag{11}$$

The first inequality of (11) is equivalent to the second inequality of (10). Multiplying the first inequality of (10) by $p(q+1) \cdot p(q)$, we have

$$p_j(q) \leq 2 \cdot p_j(q+1) - \frac{p_j(q+1) \cdot p_j(q)}{p_j(q-1)}. \tag{12}$$

We will now show by induction that the second inequality of (11) follows from (12). If $q = 1$ then, from (12), $p_j(1) \leq 2 \cdot p_j(2)$, as $1/p_j(0) = 0$. If $q > 1$ then, by induction, $(q-1) \cdot p_j(q-1) \leq q \cdot p_j(q)$. Using this and (12), we have

$$p_j(q) \leq 2 \cdot p_j(q+1) - \frac{p_j(q+1) \cdot p_j(q)}{\frac{q}{q-1} \cdot p_j(q)} = \frac{q+1}{q} \cdot p(q+1).$$

We have just shown that (10) implies (11). The reverse is not true. For example, the processing time function such that $p_j(1) = 8$, $p_j(2) = 6$, $p_j(3) = 4$, satisfies (11) but not (10). Therefore, (10) is a restriction of the monotonous penalty assumption, but a generalization of the work preserving assumption.

2.3 The main result

Under assumption (10), functions $\Delta C_j^u(\varepsilon)$ have a useful property, as shown in the next lemmas.

We will need the following notations. By definition, $\Delta Z^\alpha(\varepsilon)$, $\Delta Z_j^\beta(\varepsilon)$, and $\Delta S_j^u(\varepsilon)$ are continuous piecewise linear functions of ε. Let now $(\Delta Z^\alpha)'(\varepsilon)$ be a subderivative of function $\Delta Z^\alpha(\varepsilon)$, $(\Delta Z_j^\beta)'(\varepsilon)$ be a subderivative of function $\Delta Z_j^\beta(\varepsilon)$, and $(\Delta S_j^u)'(\varepsilon)$ be a subderivative of function $\Delta S_j^u(\varepsilon)$.

Lemma 1. *If inequalities (10) hold for $j = a$ then function $\Delta Z^\alpha(\varepsilon)$ is convex*

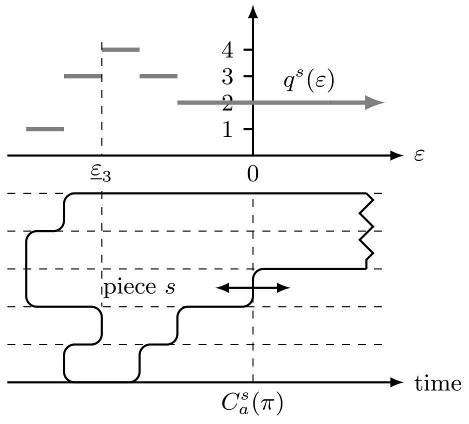


Figure 4: Illustration for Lemma 1

in interval $[\underline{\varepsilon}, \overline{\varepsilon}]$.

Proof. Let $q^s(\varepsilon)$ be the number of machines assigned to job a at time moment $C_a^s(\pi) + \varepsilon$ in schedule π^α without taking into account the machine where piece s is processed. See an illustration in Figure 4.

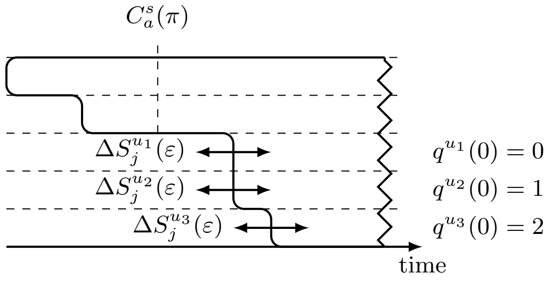
By the definition of $\Delta Z^\alpha(\varepsilon)$, we have

$$(\Delta Z^\alpha)'(\varepsilon) = \frac{1}{p_a\big(q^s(\varepsilon)+1\big)} - \frac{1}{p_a\big(q^s(\varepsilon)\big)}.$$

When $\varepsilon \in [\underline{\varepsilon}_3, \mathcal{M}]$, function $q^s(\varepsilon)$ is non-increasing, as in π^α, there are no pieces of job a started in $(\underline{\varepsilon}_3, \mathcal{M}]$. Thus, in this interval, $(\Delta Z^\alpha)'(\varepsilon)$ is non-decreasing by (10), and therefore, $\Delta Z^\alpha(\varepsilon)$ is convex in $[\underline{\varepsilon}, \overline{\varepsilon}] \subset [\underline{\varepsilon}_3, \mathcal{M}]$. □

Lemma 2. *If inequalities (10) hold then functions* $\Delta C_j^v(\varepsilon)$ *defined in (5) are concave in interval* $[\underline{\varepsilon}, \overline{\varepsilon}]$.

Proof. Functions $\Delta C_j^v(\varepsilon)$, $v \notin V_j$, are concave by definition (5). The concavity of other functions we will prove by induction. We consider jobs $j \in N$, $V_j \neq \emptyset$, in a non-decreasing order of their completion times in π. Let j' be is the first such job. Then for each $v \in V_{j'}$, either $\Delta S_{j'}^v(\varepsilon) = 0$ or $\Delta S_{j'}^v(\varepsilon) = \varepsilon$ by definition.



$C^s_a(\pi)$
$\Delta S^{u_1}_j(\varepsilon)$
$\Delta S^{u_2}_j(\varepsilon)$
$\Delta S^{u_3}_j(\varepsilon)$
$q^{u_1}(0) = 0$
$q^{u_2}(0) = 1$
$q^{u_3}(0) = 2$
time

Figure 5: Illustration for Lemma 2: job j in schedule π^β

Therefore, by (5), for each $v \in V_{j'}$, $\Delta C^v_{j'}(\varepsilon)$ is linear and thus concave function for any ε.

Let now j be a non-first job in the order defined in the previous paragraph. Consider any piece $v \in V_j$. If $S^v_j(\pi) < C^s_a(\pi)$ then $\Delta S^v_j(\varepsilon) = 0$. Otherwise $\Delta S^v_j(\varepsilon)$ is concave in interval $[\underline{\varepsilon}, \overline{\varepsilon}]$, as all functions $\Delta C^u_i(\varepsilon)$, $C^u_i(\pi) \le S^v_j(\pi) < C_j(\pi)$, are concave in this interval by induction.

Let $t^u_j(\varepsilon)$ be the number of machines assigned to pieces in $V_j \setminus U_j$ of job j at time moment $S^u_j(\pi^\beta) + \Delta S^u_j(\varepsilon)$ in schedule π^β.

We now fix an arbitrary order O for pieces in U_j. For each piece $u \in U_j$, let $q^u(\varepsilon)$ be the number of pieces $u \in V_j$ such that

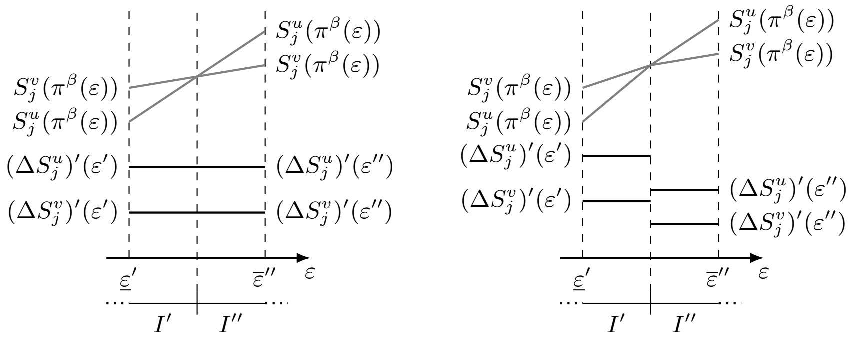
$$S^v_j(\pi^\beta) + \Delta S^v_j(\varepsilon) < S^u_j(\pi^\beta) + \Delta S^u_j(\varepsilon) \quad \text{or}$$
$$S^v_j(\pi^\beta) + \Delta S^v_j(\varepsilon) = S^u_j(\pi^\beta) + \Delta S^u_j(\varepsilon) \text{ and } v \text{ precedes } u \text{ in order } O.$$

See illustration of values $q^u(\varepsilon)$ for fixed $\varepsilon = 0$ in Figure 5.

By the definition of $\Delta Z^\beta_j(\varepsilon)$, and (1), we have

$$(\Delta Z^\beta_j)'(\varepsilon) = \sum_{u \in U_j} (\Delta S^u_j)'(\varepsilon) \cdot \underbrace{\left(\frac{1}{p_j\big(t^u_j(\varepsilon) + q^u(\varepsilon)\big)} - \frac{1}{p_j\big(t^u_j(\varepsilon) + q^u(\varepsilon) + 1\big)} \right)}_{\le 0 \text{ by } (10)}.$$

Remember that $\Delta S^u_j(\varepsilon)$ are concave functions in $[\underline{\varepsilon}, \overline{\varepsilon}]$, and thus subderiva-



$S_j^u(\pi^\beta(\varepsilon))$
$S_j^v(\pi^\beta(\varepsilon))$
$S_j^v(\pi^\beta(\varepsilon))$
$S_j^u(\pi^\beta(\varepsilon))$
$(\Delta S_j^u)'(\varepsilon')$
$(\Delta S_j^u)'(\varepsilon'')$
$(\Delta S_j^v)'(\varepsilon')$
$(\Delta S_j^v)'(\varepsilon'')$
$\underline{\varepsilon}'$
$\overline{\varepsilon}''$
ε
I'
I''
$S_j^u(\pi^\beta(\varepsilon))$
$S_j^v(\pi^\beta(\varepsilon))$
$S_j^v(\pi^\beta(\varepsilon))$
$S_j^u(\pi^\beta(\varepsilon))$
$(\Delta S_j^u)'(\varepsilon')$
$(\Delta S_j^v)'(\varepsilon')$
$(\Delta S_j^u)'(\varepsilon'')$
$(\Delta S_j^v)'(\varepsilon'')$
$\underline{\varepsilon}'$
$\overline{\varepsilon}''$
ε
I'
I''

Figure 6: Functions $\Delta S_j^u(\varepsilon')$ and their subderivatives around the border between intervals I' and I'' (possible variants)

tives $(\Delta S_j^u)'(\varepsilon)$ are non-increasing functions in it. Moreover, functions $t_j^u(\varepsilon)$ are non-decreasing, as in π^β, there are no pieces of job j completed before $\mathcal{M}$. Therefore, by (10), in intervals where functions $q^u(\varepsilon)$ are constant, subderivative $(\Delta Z_j^\beta)'(\varepsilon)$ is a non-decreasing function.

Now we will see what happens on the borders of these intervals, where some functions $q^u(\varepsilon)$ change their value. Consider two adjacent maximal intervals I' and I'' (I' is on the left of I'') in which functions $q^u(\varepsilon)$ are constant, and functions $t_j^u(\varepsilon)$ are equal to τ. Let function $q^u(\varepsilon)$, $u \in U_j$, take value $q'_u(\varepsilon)$ in I' and value $q''_u(\varepsilon)$ in I''.

Suppose we have $q'_u(\varepsilon)+\tau = q''_v(\varepsilon)+\tau = \rho-1$, $q''_u(\varepsilon)+\tau = q'_v(\varepsilon)+\tau = \rho$, and $q''_x(\varepsilon) = q'_x(\varepsilon)$ for any other pieces $x \in U_j$. Then, on the border between I' and I'', starting time $S_j^u(\pi^\beta(\varepsilon))$ of piece $u \in U_j$ "overtake" starting time $S_j^v(\pi^\beta(\varepsilon))$ of piece $v \in U_j$, i.e. $S_j^u(\pi^\beta(\varepsilon)) \leq S_j^v(\pi^\beta(\varepsilon))$, $\varepsilon \in I'$, and $S_j^u(\pi^\beta(\varepsilon)) \geq S_j^v(\pi^\beta(\varepsilon))$, $\varepsilon \in I''$. By the definition of functions $q^u(\varepsilon)$, there should exist two points $\underline{\varepsilon}' \in I'$ and $\overline{\varepsilon}'' \in I''$ such that, for any $\varepsilon' \geq \underline{\varepsilon}'$, $\varepsilon' \in I'$, and for any $\varepsilon'' \leq \overline{\varepsilon}''$, $\varepsilon'' \in I''$, we have $(\Delta S_j^u)'(\varepsilon') > (\Delta S_j^v)'(\varepsilon')$ and $(\Delta S_j^u)'(\varepsilon'') > (\Delta S_j^v)'(\varepsilon'')$. See illustration in

Figure 6. As subderivatives $(\Delta S_j^u)'(\varepsilon)$ are non-increasing, we have

$$\begin{aligned} \mu &= (\Delta S_j^u)'(\varepsilon') - (\Delta S_j^v)'(\varepsilon'') > 0, \\ \nu &= (\Delta S_j^u)'(\varepsilon'') - (\Delta S_j^v)'(\varepsilon') \leq \mu. \end{aligned} \tag{13}$$

Now, $(\Delta Z_j^\beta)'(\varepsilon'') - (\Delta Z_j^\beta)'(\varepsilon')$ is equal to

$$(\Delta S_j^v)'(\varepsilon'') \cdot \left(\frac{1}{p_j(\rho-1)} - \frac{1}{p_j(\rho)}\right) - (\Delta S_j^v)'(\varepsilon') \cdot \left(\frac{1}{p_j(\rho)} - \frac{1}{p_j(\rho+1)}\right) +$$
$$(\Delta S_j^u)'(\varepsilon'') \cdot \left(\frac{1}{p_j(\rho)} - \frac{1}{p_j(\rho+1)}\right) - (\Delta S_j^u)'(\varepsilon') \cdot \left(\frac{1}{p_j(\rho-1)} - \frac{1}{p_j(\rho)}\right) \overset{(13)}{=}$$
$$\mu \cdot \left(\frac{1}{p_j(\rho)} - \frac{1}{p_j(\rho-1)}\right) - \nu \cdot \left(\frac{1}{p_j(\rho+1)} - \frac{1}{p_j(\rho)}\right),$$

which is a non-negative value by assumption (10) and by $\mu \geq \nu$, $\mu > 0$.

Therefore

$$\max_{\varepsilon \in I'} (\Delta Z_j^\beta)'(\varepsilon) \leq \min_{\varepsilon \in I''} (\Delta Z_j^\beta)'(\varepsilon). \tag{14}$$

The inequality (14) for other adjacent pairs of maximal intervals, in which functions $q^u(\varepsilon)$, $u \in U_j$, are constant, can be shown in the same manner. On borders between these intervals, several functions $q^u(\varepsilon)$ change their value, and/or functions $t_j^u(\varepsilon)$ increase their value.

We have just shown that subderivative $(\Delta Z_j^\beta)'(\varepsilon)$ is non-decreasing in interval $[\underline{\varepsilon}, \overline{\varepsilon}]$. Therefore, function $\Delta Z_j^\beta(\varepsilon)$ is convex. Using Lemma 1 and formulae (5), we can conclude that functions $\Delta C_j^u(\varepsilon)$ are concave in $[\underline{\varepsilon}, \overline{\varepsilon}]$. □

Using Lemma 2, we will now prove the main result of the paper.

Theorem 1. *If the inequalities (10) hold then there exists an optimal ascending schedule.*

Proof. First, there exists an optimal schedule in which each job has at most m pieces, i.e. at most one piece per machine. Otherwise, pieces of the same job assigned to the same machine, could be joined together without increasing the completion time of any job.

To prove the theorem, we will show that it is possible to transform an optimal non-ascending schedule into another optimal schedule in which

- either the total number of pieces is strictly decreased,
- or the number of early pieces is strictly decreased.

Note that, in every feasible schedule, each job contains at least one terminal piece. Therefore, applying this transformation to an optimal schedule at most $(m-1)\cdot n$ times, we can obtain an optimal schedule without early pieces.

Let π be an optimal non-ascending schedule of the problem. For π, we construct the family of admissible schedules $\pi(\varepsilon)$, $\varepsilon \in [\underline{\varepsilon}, \overline{\varepsilon}]$, according to the procedure described above.

Remember that

$$\Delta C_j(\varepsilon) = \begin{cases} \Delta C_j^v(\varepsilon) \text{ for some } v \in V_j, & |V_j| > 0, \\ 0, & |V_j| = 0. \end{cases}$$

Then, by Lemma 2, functions $\Delta C_j(\varepsilon)$, $j \in N$, are concave in interval $[\underline{\varepsilon}, \overline{\varepsilon}]$, as well as the objective function $F(\pi(\varepsilon)) = F(\pi) + \sum_{j\in N} w_j \Delta C_j(\varepsilon)$. Therefore, as $\pi = \pi(0)$ is an optimal schedule and $0 \in [\underline{\varepsilon}, \overline{\varepsilon}]$, we should have $F(\pi(\underline{\varepsilon})) = F(\pi)$ or $F(\pi(\overline{\varepsilon})) = F(\pi)$, meaning that at least one of schedules $\pi(\underline{\varepsilon})$ and $\pi(\overline{\varepsilon})$ is optimal.

There are three cases.

1. $\underline{\varepsilon} = \underline{\varepsilon}_1$ (or $\overline{\varepsilon} = \overline{\varepsilon}_1$). Then one of the inequalities (6) is satisfied as an equality, and the corresponding piece u of job j disappears in $\pi(\underline{\varepsilon})$ (or in $\pi(\overline{\varepsilon})$).

2. $\underline{\varepsilon} = \max\{\underline{\varepsilon}_2, \underline{\varepsilon}_3\}$ (or $\overline{\varepsilon} = \min\{\overline{\varepsilon}_2, \overline{\varepsilon}_3\}$). Then one of the inequalities (7)

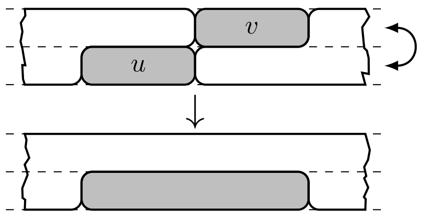


Figure 7: Combining two pieces into one

and (8) is satisfied as an equality, and the corresponding pieces u and v of job j can be combined to one in $\pi(\underline{\varepsilon})$ (or in $\pi(\overline{\varepsilon})$), as illustrated in Figure 7.

3. $\underline{\varepsilon} = \underline{\varepsilon}_4$ (or $\overline{\varepsilon} = \overline{\varepsilon}_4$). Then one of the inequalities (9) is satisfied as an equality, and the corresponding piece u of job j which was early in π becomes terminal in $\pi(\underline{\varepsilon})$ (or in $\pi(\overline{\varepsilon})$).

$\square$

3 Conclusion

In the paper, we have introduced the ascending property for the problem of scheduling malleable jobs to minimize the total weighted completion time. Then we have showed that the set of ascending schedules is dominant given a natural assumption that the gain of assigning an additional machine to a job does not increase with the number of machines assigned to this job. This result can be used to reduce significantly the search space while finding an optimal schedule of the problem.

Note that, in order to find an optimum ascending schedule, the sequence of job completion times must be determined. For the work preserving case, Caramia and Drozdowski [4] showed that, once this sequence is known, the problem can be solved in polynomial time via the linear programming. Therefore, a good alternative for solving the problem is to enumerate the sequences of jobs without using the ascending property. However, our result can be applied in a more general (and more practical) case. Additionally, our dominance rule may be useful if one wants to develop a pure combinatorial algorithm without referring to the linear programming.

As a possible further research direction we can mention that the complexity status of a special case of our problem in which the processing time functions are work preserving and job weights are unitary is still open. It was unknown even for this special case whether the set of ascending schedules is dominant. Therefore, our result potentially can help to determine the complexity status of this important problem which can be denoted as $P|var,\ p_j(q) = p_j/q,\ \delta_j|\sum C_j$. Note that recently Hendel and Kubiak [8] have shown that the latter problem is polynomially solvable when only 2 machines are available.

Acknowledgements

The author is grateful to Yann Hendel who acquainted him with the problem. Also the author would like to thank all anonymous referees for the remarks and suggestions which greatly improved the quality of the paper.

References

[1] E. Blayo, L. Debreu, G. Mounié, and D. Trystram. Dynamic load balancing for adaptive mesh ocean circulation model. *Engineering Simulations*, 22:8–24, 2000.

[2] J. Blazewicz, M.Y. Kovalyov, M. Machowiak, D. Trystram, and J. Weglarz. Preemptable malleable task scheduling problem. *IEEE Transactions on Computers*, 55(4):486–490, 2006.

[3] J. Blazewicz, M. Machowiak, G. Mounié, and D. Trystram. Approximation algorithms for scheduling independent malleable tasks. In Rizos Sakellariou, John Gurd, Len Freeman, and John Keane, editors, *Euro-Par 2001 Parallel Processing*, volume 2150 of *Lecture Notes in Computer Science*, pages 191–197. Springer Berlin / Heidelberg, 2001.

[4] M. Caramia and M. Drozdowski. Scheduling malleable tasks for mean flow time criterion. In *Abstracts of the 10th International Workshop on Project Management and Scheduling*, pages 106–109, 2006.

[5] M. Drozdowski. Scheduling parallel tasks — algorithms and complexity. In J.Y.-T. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis.* Chapman Hall, CRC Press, 2004.

[6] P.-F. Dutot, G. Mounié, and D. Trystram. Scheduling parallel tasks — approximation algorithms. In J.Y.-T. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis.* Chapman Hall, CRC Press., 2004.

[7] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[8] Y. Hendel and W. Kubiak. Minimizing mean flow time for the two machines semi-malleable jobs scheduling problem. In F. Şerifoğlu and Ü. Bilge, editors, *Abstracts of the 11th International Workshop on Project Management and Scheduling*, pages 136–140, 2008.

[9] W.T. Ludwig. *Algorithms for Scheduling Malleable and Nonmalleable Parallel Tasks.* PhD thesis, University of Wisconsin-Madison, 1995.

[10] Robert McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6(1):1–12, 1959.

[11] Gregory Mounie, Christophe Rapine, and Denis Trystram. A $\frac{3}{2}$-approximation algorithm for scheduling independent monotonic malleable tasks. *SIAM Journal on Computing*, 37(2):401–412, 2007.

[12] Ch. Rapine, I. Scherson, and D. Trystram. On-line scheduling of parallelizable jobs. In David Pritchard and Jeff Reeve, editors, *Euro-Par'98 Parallel Processing*, volume 1470 of *Lecture Notes in Computer Science*, pages 322–327. Springer Berlin / Heidelberg, 1998.

[13] P. Serafini. Scheduling jobs on several machines with the jobs splitting property. *Operations Research*, 44(4):617–628, 1996.