Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 2840–2848
computers & operations research
www.elsevier.com/locate/cor

An exact method for $Pm/sds, r_i/\sum_{i=1}^{n} C_i$ problem

Rabia Nessah*, Chengbin Chu, Farouk Yalaoui

ISTIT-OSI (CNRS FRE 2732), Université de Technologie de Troyes, 12 rue Marie Curie-BP 2060, 10010 Troyes Cedex, France

Available online 13 December 2005

Abstract

This paper addresses an identical parallel machine scheduling problem, with sequence-dependent setup times and release dates to minimize total completion time. This problem is known to be strongly NP-hard. We prove a sufficient and necessary condition for local optimality which can also be considered as a priority rule. We then define a dominant subset based on this condition. We present efficient heuristic algorithms using this condition to build a schedule belonging to this subset. We also prove dominance theorem, and develop a lower bound that can be computed in polynomial time. We construct a branch-and-bound algorithm in which the heuristic, the lower bound and the dominance properties are incorporated. Computational experiments suggest that the algorithm can handle test problems with 40 jobs and 2 machines.

Keywords: Identical parallel machine; Scheduling; Setup time; Release dates; Lower bound; Dominance properties; Heuristic

1. Introduction

The identical parallel machine scheduling problem with sequence-dependent setup times and release dates (IPMSPS), often arises in manufacturing systems where n non-preemptive jobs must be executed on m identical parallel machines. Sequence-dependent setup times are non-negligible. The aim is to minimize the total completion time. More precisely, denote $J = \{1, \ldots, n\}$ as the set of jobs and $M = \{1, \ldots, m\}$ the set of machines. Each job i has a processing time p_i and a release date r_i. A setup time s_{ij} is required if job j is the immediate successor of job i on a machine. The value of s_{ii} is set to zero for all i without loss of generality. We assume that the setup times satisfy the triangle inequality; i.e., $s_{ij} + s_{jk} \geqslant s_{ik}, \forall i, j, k$. Note that this assumption is realistic and without loss of generality $s_{0i} = 0, \forall i$. We also assume that $m \leqslant n - 1$ in order to avoid trivial cases. A machine can process at most one job at a time. The IPMSPS consists of simultaneously determining an assignment of jobs to machines and a sequence of jobs on each machine. The objective is to minimize the sum of completion time ($\sum_{i=1}^{n} C_i$) where C_i denotes the completion time of job i. According to the standard machine scheduling classification, the IPMSPS is referred to $Pm/sds, r_i/\sum_{i=1}^{n} C_i$ [1].

The single machine counterpart (usually referred to as $1/r_i/\sum_{i=1}^{n} C_i$) is known to be *NP-hard* in the strong sense [2]. This allows to conclude that the problem $Pm/sds, r_i/\sum_{i=1}^{n} C_i$ is also NP-hard in the strong sense.

For parallel machine scheduling with setup times, the most studied criterion is the makespan. Frederickson et al. [3] worked in the context of the minmax m-TSP. They showed that if m job sequences are constructed simultaneously, using a least cost insertion or a nearest neighborhood method, and the instance matrix satisfies the triangle inequality,

* Corresponding author.
E-mail addresses: rabia.nessah@utt.fr (R. Nessah), chengbin.chu@utt.fr (C. Chu), farouk.yalaoui@utt.fr (F. Yalaoui).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.10.017

then the heuristic has a worst case performance ratio of 2. França et al. [4] applied two versions of tabu search. More recently, Gendreau et al. [5] elaborated a divide and merge heuristic and a lower bound for the problem $Pm/sds/C_{\max}$. Brucker and Thiele [6] proposed a branch-and-bound method for the general-shop problem with sequence-dependent setup times to minimize the makespan. Baptiste and Le Pape [7] elaborated a branch-and-bound algorithm for a single machine scheduling problem to minimize a regular objective function under setup constraints. Yalaoui and Chu [8] proposed a heuristic for parallel machine scheduling with job splitting and sequence-dependent setup times. Chen and Powell [9] solved the following problems ($Pm/s_u/\sum_{j=1}^{n}(w_jC_j)$, $Pm/s_{uv}/\sum_{j=1}^{n}(w_jC_j)$, $Pm/s_u/\sum_{j=1}^{n}(w_jU_j)$ and $Pm/s_{uv}/\sum_{j=1}^{n}(w_jU_j)$) optimally with the column generation method, where s_{uv} is the setup time if a job from family v is processed immediately after a job from family u. Allahverdi and Aldowaisan [10,11] studied a two-machine flowshop with sequence independent setup times problem. Rajendran and Ziegler [12] proposed a heuristic for the problem $F/sds/\sum_{i=1}^{n}(C_i+T_i)$ where C_i, and T_i are, respectively, the completion time and the tardiness of job i. On the other hand, the problem without setup times was extensively studied in the literature.

Du et al. [13] proved that the problem for parallel machine scheduling to minimize total completion time with release dates remains NP-hard even if preemption is allowed. However, the problem is polynomially solvable if all jobs have the same processing time and the number of machines does not exceed two [13]. Yalaoui and Chu developed an exact method based on branch-and-bound for the same problem [14]. Belouadah and Potts [15] elaborated their method by using the dominance properties developed by Elmagraby and Park [16]. They proposed an upper bound and a lower bound based on lagrangian relaxation for the total weighted completion time scheduling problem but with identical release dates ($Pm//\sum_{j=1}^{n}w_jC_j$). Azizoglu and Kirca [17] proposed a branch-and-bound algorithm to solve this problem. They used the results of Elmagraby and Park by bringing new dominance properties. We can also quote the algorithms by Barnes and Brennan [18] and Serin et al. [19]. Baptiste [20] proposed a dynamic programming algorithm which calculates the optimal solution of the problem $Pm/r_j, p_j=p, a_j/\sum_{j=1}^{n}C_j$ where a_j is the number of processors required to execute job j. Azizoglu [21] proved that the problem of scheduling n jobs on m identical parallel machines with deadlines $d_1, d_2, \ldots, d_n$, by assuming that the processing times and deadlines are agreeable, i.e., $p_i \leqslant p_j$ implies $d_i \leqslant d_j$, is polynomially solvable when preemption is allowed.

The development of $1+\varepsilon$-approximate algorithms or *polynomial time approximation schemes* (PTAS) received much attention. They are polynomial algorithms and the performance in the worst case does not exceed $1+\varepsilon$. Leonardi et Raz [22] studied both problems $Pm/pmtn, r_j/\sum_{j=1}^{n}F_j$ and $Pm/r_j/\sum_{j=1}^{n}F_j$ where $F_j=C_j-r_j$. They solved the problem with preemption, using the shortest processing remaining time (SPRT) rule in $O(\lg(\min(n/m, p)))$ time where $p=\max p_j/\min p_j$. Phillips et al. [23] showed that the ratio between the optimal solutions of these two problems is less than or equal to $\frac{7}{3}$. This performance improves the one proposed by Phillips et al. [24] which is equal to $(3-1/m)$.

The purpose of this article is to provide efficient lower bounds for the IPMSPS, as we obtain a condition for local optimality, define FS-active schedules for this problem and dominance criterion. These are described in Section 2. Sections 3 and 4, respectively, are devoted to a lower bound and heuristics. Section 5 presents the branch-and-bound algorithm that we propose. The quality of the proposed lower and upper bounds is showed through a series of computational experiments presented in Section 6. The conclusion follows in Section 7.

2. Local optimality condition and dominance property

2.1. A condition for local optimality

Obtaining a condition for local optimality is equivalent to considering a two job scheduling problem. Let us consider this problem. Two jobs i and j have to be scheduled on a machine available after t. We look for the condition under which scheduling i before j gives an optimal schedule.

Assuming that C_{uv} ($u, v \in \{i, j\}$) is the sum of completion time of jobs u and v obtained by scheduling u before v, the problem is to find the condition under which we have $C_{ij} \leqslant C_{ji}$.

We note by k, the last job scheduled on the considered machine which becomes available at time t. If this job does not exist, we put $k=0$. Consider the following function:

$$f_{ij}(k,t)=\max\{2\max(t+s_{ki}, r_i)+p_i+s_{ij}, \max(t+s_{ki}, r_i)+\max(t+s_{ki}, r_j)\}.$$

It should be noticed that this function depends on the parameters defining jobs i and j. The sufficient and necessary condition for local optimality can be obtained according to this function.

In the remainder, the word "schedule" indifferently refers to a partial schedule or a complete schedule, except where otherwise noted. A partial schedule is a subsequence of the set of jobs.

Theorem 1. *Consider a scheduling problem involving two jobs i and j which have to be scheduled on a machine available after time t, with k being the last job scheduled.* $C_{ij} \leqslant C_{ji}$ *if and only if* $f_{ij}(k,t) \leqslant f_{ji}(k,t)$.

Proof. According to the definitions, we put $\delta_i = t + s_{ki}$, we have

$$\begin{aligned} C_{ij} &= \max(\delta_i, r_i) + p_i + \max(\max(\delta_i, r_i) + p_i + s_{ij}, r_j) + p_j \\ &= \max(\delta_i, r_i) + \max(\max(\delta_i, r_i) + p_i + s_{ij}, \delta_i, r_j) + p_i + p_j \\ &= \max(\delta_i, r_i) + \max(\max(\delta_i, r_i) + p_i + s_{ij}, \max(\delta_i, r_j)) + p_i + p_j \\ &= \max(2\max(\delta_i, r_i) + p_i + s_{ij}, \max(\delta_i, r_i) + \max(\delta_i, r_j)) + p_i + p_j \\ &= f_{ij}(k,t) + p_i + p_j. \end{aligned}$$

We conclude that $C_{ij} \leqslant C_{ji}$ if and only if $f_{ij}(k,t) \leqslant f_{ji}(k,t)$. □

According to Theorem 1, we can consider the f_{ij} function as a dynamic priority measure: the smaller the value compared to f_{ji}, the higher the priority of job i compared to job j. This new rule is incorporated in the construction of heuristics presented in Section 5.

Definition 1. Given an active schedule S, if for any two adjacent and consecutive jobs x, y (x followed by y) scheduled on a machine available at time t and on which the last scheduled job is k, we have $\max(t + s_{kx}, r_x) < \max(t + s_{ky}, r_y) + s_{yx}$ or $f_{xy}(k,t) \leqslant f_{yx}(k,t)$, S is said to be FS-active.

We have the following theorem:

Theorem 2. *All optimal schedules of the identical parallel machines scheduling problem with sequence-dependent setup times and job release dates, to minimize total completion time are FS-active.*

Proof. Let us assume that there exists an optimal schedule S which is not *FS-active*. There is certainly at least one couple of adjacent and consecutive jobs x and y (x followed by y) scheduled on a machine verifying neither of the two conditions described in the definition of *FS-active* schedules, then:

$$\begin{cases} \max(t + s_{kx}, r_x) \geqslant \max(t + s_{ky}, r_y) + s_{yx}, \\ f_{xy}(k,t) > f_{yx}(k,t), \end{cases} \tag{1}$$

where t is the time that the machine becomes available and k is the job preceding immediately x.

Let z the job following y on the same machine μ and let U_h, $h = 1, \ldots, m$, be the set of jobs scheduled on machine h, let B be the set of jobs preceding x and A the set of jobs following y. In this case, the total completion time for S is

$$C = \sum_{i=1}^{n} C_i = \sum_{h=1,\ldots,m, h\neq\mu} \sum_{i\in U_h} C_i + \sum_{i\in B\cup A} C_i + C_{xy}. \tag{2}$$

We construct another schedule S' by interchanging x and y without moving any other jobs. From (1), S' is feasible.

We put $a = \max(t + s_{kx}, r_x) + s_{xy} + s_{yz} + p_x + p_y$ and $b = \max(\max(t + s_{ky}, r_y) + p_y + s_{yx}, r_x) + p_x + s_{xz}$.

- If $\max(\max(\phi_\mu + s_{ky}, r_y) + p_y + s_{yx}, r_x) = r_x$, then $b = r_x + p_x + s_{xz}$, $b - a = r_x - \max(\phi_\mu + s_{kx}, r_x) + (s_{xz} - s_{xy} - s_{yz}) - p_y$. According the triangular inequality, we obtain then $b - a \leqslant -p_y \leqslant 0$.

- If $\max(\max(\phi_\mu+s_{ky}, r_y)+p_y+s_{yx}, r_x)=\max(\phi_\mu+s_{ky}, r_y)+p_y+s_{yx}$, then $b=\max(\phi_\mu+s_{ky}, r_y)+p_y+s_{yx}+p_x+s_{xz}$. $b-a=(\max(\phi_\mu+s_{ky}, r_y)+s_{yx}-\max(\phi_\mu+s_{kx}, r_x))+(s_{xz}-s_{xy}-s_{yz})$. According to the inequalities (1) and the triangular inequality, we obtain then $b-a\leqslant 0$.

For either cases, we have $b\leqslant a$. The earliest beginning time of job z in S and S' is, respectively defined as: $\Delta_z(S)=\max(a, r_z)$ and $\Delta_z(S')=\max(b, r_z)$. Since $b\leqslant a$, we conclude $\Delta_z(S')\leqslant\Delta_z(S)$.

We then have the total completion time for S':

$$C'=\sum_{h=1,\ldots,m,h\neq\mu}\sum_{i\in U_h}C'_i+\sum_{i\in B\cup A}C'_i+C_{yx}\leqslant\sum_{h=1,\ldots,m,h\neq\mu}\sum_{i\in U_h}C_i+\sum_{i\in B\cup A}C_i+C_{yx}. \tag{3}$$

From (1) and Theorem 1, we have $C_{yx}<C_{xy}$. Then, we conclude $C>C'$. This contradicts the assumption that S is an optimal schedule. □

2.2. Dominance property

In the branch-and-bound method, a node of the search tree represents a partial schedule from time 0 containing a subset of jobs, called scheduled jobs. If we can show that a partial schedule cannot lead to an optimal complete schedule, the corresponding node of the search tree can be pruned, hence to reduce the compilation time. We note by $C(\sigma)$ the total completion time of jobs in the partial schedule σ, $\phi_\mu(\sigma)$ the completion time of the last job in σ on machine μ and $\Omega(\sigma)$ the schedule obtained by appending to σ the optimal partial schedule of the remaining jobs.

Definition 2. Let σ_1 and σ_2 be two partial schedules. We say that σ_1 dominates σ_2 or σ_2 is dominated, if

$$C(\Omega(\sigma_1))\leqslant C(\Omega(\sigma_2)).$$

The following theorem develops conditions so that a partial schedule is dominated.

Theorem 3. *Let σ_1 and σ_2 be two partial schedules for the same set of jobs K, and let $s=|K|$. If $C(\sigma_1)-C(\sigma_2)\geqslant(n-s)\Delta$, then σ_2 dominates σ_1. Where $\Delta=\max_{\mu\in\{1,\ldots,m\}}\{[\phi_\mu(\sigma_2)-\phi_\mu(\sigma_1)]+\max_{j\in N-K}(s_{k_\mu(\sigma_2)j}-s_{k_\mu(\sigma_1)j})\}$, $k_\mu(\sigma_1)$ and $k_\mu(\sigma_2)$ are, respectively, the last job scheduled on machine μ in schedule σ_1, and in schedule σ_2.*

Proof. Without loss of generality, the machines are renumbered in increasing order of their finishing time in σ_1 and σ_2.

Let us consider $\Omega(\sigma_1)$. Construct another schedule S from $\Omega(\sigma_1)$ so that for any machine μ, the partial schedule after $\phi_\mu(\sigma_1)$ in $\Omega(\sigma_1)$ is replaced by the partial schedule after $\phi_\mu(\sigma_2)$ in $\Omega(\sigma_2)$.

Let j_1 be the first job scheduled after $\phi_\mu(\sigma_1)$ on the machine μ in $\Omega(\sigma_1)$.

Case 1: If $\Delta\leqslant 0$, then $\forall j\in N-K$, we have $\phi_\mu(\sigma_2)+s_{k_\mu(\sigma_2)j_1}\leqslant\phi_\mu(\sigma_1)+s_{k_\mu(\sigma_1)j_1}$. Hence, $C(\Omega(\sigma_2))-C(\Omega(\sigma_1))\leqslant C(S)-C(\Omega(\sigma_1))\leqslant C(\sigma_2)-C(\sigma_1)$. Then, we conclude σ_2 dominates σ_1.

Case 2: If $\Delta>0$, let Q_μ be the number of jobs scheduled on μ after $\phi_\mu(\sigma_2)$ in S. Let $\alpha_\mu=\phi_\mu(\sigma_2)-\phi_\mu(\sigma_1)+s_{k_\mu(\sigma_2)j_1}-s_{k_\mu(\sigma_1)j_1}$.

- If $\alpha_\mu>0$, then each job scheduled after $\phi_\mu(\sigma_1)$ on μ in S, is delayed at most α_μ, i.e. $C_i(S)-C_i(\Omega(\sigma_1))\leqslant\alpha_\mu$, for all job i scheduled after $\phi_\mu(\sigma_1)$ in S.
- If $\alpha_\mu\leqslant 0$, then all jobs scheduled after $\phi_\mu(\sigma_1)$ on μ, will be delayed, i.e. $C_i(S)\leqslant C_i(\Omega(\sigma_1))$. We put $L=\{\mu$ such that $\alpha_\mu>0\}$.

Then, we have $C(\Omega(\sigma_2))-C(\Omega(\sigma_1))\leqslant C(S)-C(\Omega(\sigma_1))\leqslant C(\sigma_2)-C(\sigma_1)+\sum_{\mu\in L}\alpha_\mu Q_\mu\leqslant C(\sigma_2)-C(\sigma_1)+(n-s)\Delta$. We also conclude σ_2 dominates σ_1. □

The following corollary applies to the special case of consecutive jobs.

Corollary 1. *Let σ be a partial schedule for the set of jobs K, let $s=|K|$, and let i and j be distinct jobs in $N-K$. If*

$$C(\sigma ij)-C(\sigma ji)\geqslant(n-s-2)\Delta, \tag{4}$$

where

$$\Delta = \max_{\mu\in\{1,\ldots,m\}}\{[\phi_\mu(\sigma ji) - \phi_\mu(\sigma ij)] + \max_{h\in N-(K\cup\{i,j\})}(s_{k_\mu(\sigma ji)h} - s_{k_\mu(\sigma ij)h})\}$$

and σij the schedule obtained by adding jobs i after j to a partial schedule σ, then σji dominates σij.

3. A lower bound for the problem $Pm/sds, r_i/\sum_{i=1}^{n} C_i$

To obtain a lower bound, we consider a relaxation of the considered problem. A usual relaxation consists of considering job preemption. However, even with this relaxation, the problem remains *NP-hard*. Our lower bound, called lb_{NCY}, is derived as follows.

We note by P, the following problem: $Pm/sds, r_i/\sum_{i=1}^{n} C_i$, $s_{i.} = \min_{j\neq i} s_{ij}$ and $s_i^* = s_{i.}/m$.

Let us consider the following three relaxed problems of problem P:

P_1: $Pm/s_{i.}, r_i/\sum_{i=1}^{n} C_i$.

$P2$: $1/s_i^*, r_i, \frac{p_i}{m}/\sum_{i=1}^{n} C_i$

subject to : $C_i \geqslant r_i + p_i$

P_3: $1/r_i, \left(\frac{p_i}{m} + s_i^*\right), pmtn/\sum_{i=1}^{n} \max(C_i^* - s_i^*, r_i + p_i)$.

We have: the problem P_{k+1} is a relaxation of problem P_k, $k = 0, \ldots, 3$ and $P_0 = P$.

The lb_{NCY} is based on the so-called modified SPRT rule on a single machine to solve P_3.

Modified SPRT: Each time the machine becomes available, the job with the shortest remaining duration $f_i = \rho_i(t) + s_i^*$ among all the available but uncompleted ones is scheduled. The job in process is interrupted if another job becomes available with a f_i strictly shorter than that of the job i in process.

Let S_* be the schedule obtained with the modified SPRT rule.

Let $C_j^*(S_*)$ be the modified completion time of the job j with the processing time of each job j is $p_i + s_i^*$.

Let us quote first of all the following lemma due to Chu [25].

Lemma 1. *Let two series of numbers* $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ *and an ordered series* $(x'_1, x'_2, \ldots, x'_n)$ *be so that* $x'_1 \leqslant x'_2 \leqslant \cdots \leqslant x'_n$ *and* $x'_j \leqslant x_j, \forall j = 1, \ldots, n$.

If $(y'_1, y'_2, \ldots, y'_n)$ *is the series obtained by sorting the series* $(y_1, y_2, \ldots, y_n)$ *in nondecreasing order, the following relation holds*:

$$\sum_{i=1}^{n} \max(x'_i, y'_i) \leqslant \sum_{i=1}^{n} \max(x_i, y_i).$$

We have the following theorem.

Theorem 4. *For each feasible schedule* S_0 *for the problem* P_3, *we have*

$$\forall j = 1, \ldots, n, \quad C^*_{[j]}(S_*) \leqslant C^*_{[j]}(S_0). \tag{5}$$

This theorem can be proved in a similar way to Theorem 6 of Chu [25] by replacing the processing time of each job i with $p_i + s_i^*$.

We have the following theorem.

Theorem 5. *Let* $a_i = p_i + r_i + s_i^*$. *Let* $(a_{[1]}, a_{[2]}, \ldots, a_{[n]})$ *be the series obtained by sorting the series* $(a_1, a_2, \ldots, a_n)$ *in nondecreasing order. Then*

$$lb_{NCY} = \sum_{i=1}^{n} \max[C^*_{[i]}(S_*), a_{[i]}] - \sum_{i=1}^{n} s_i^*$$

is a lower bound for problem P.

Proof. Let S be any feasible solution of the problem P_3. We have according to the Theorem 5, $\forall j = 1, \ldots, n$, $C^*_{[j]}(S_*) \leqslant C^*_{[j]}(S)$. According to Lemma 1, we deduce

$$\sum_{j=1}^{n} \max[C^*_{[j]}(S_*), a_{[j]}] \leqslant \sum_{i=1}^{n} \max[C^*_i(S), a_i]. \tag{6}$$

Then, we obtain

$$\sum_{j=1}^{n} \max(C^*_{[j]}(S^*), a_{[j]}) - \sum_{j=1}^{n} s^*_j \leqslant \sum_{i=1}^{n} \max(C^*_i(S) - s^*_i, r_i + p_i).$$

As the problem P_3 is a relaxation of the problem P, we conclude then $lb_{NCY} = \sum_{i=1}^{n} \max[C^*_{[i]}(S_*), a_{[i]}] - \sum_{i=1}^{n} s^*_i$ is a lower bound for the problem P. □

Remark 1. This lower bound lb_{NCY} is a generalization of the lower bound developed by Baptiste and Le Pape [7] for a single machine problem, to the identical parallel machine problem.

Remark 2. The complexity of lb_{NCY} is $O(n \log n)$, the same complexity as SPRT.

4. Heuristics

In this section, we build some heuristic algorithms: earliest completion time (ECT), priority remaining total flow time (PRTF) and priority remaining total setup time (PRTS).

All the algorithms definitively schedule a job behind the partial schedule composed of scheduled jobs. At each iteration, the machine μ begins to be available at time ϕ_μ, and there is a set of unscheduled jobs A. All the ϕ_μ, $\mu = 1, \ldots, m$ and A are initialized to 0 and $\{1, \ldots, n\}$, respectively. We note by k_μ, the last job scheduled on machine μ, if this job does not exist, we put $k_\mu = 0$, and $s_{0i} = 0$, $\forall i$. When a job x is chosen to be scheduled on a machine μ, ϕ_μ and A are updated such that $\phi_\mu := C_x(k_\mu, \phi_\mu)$ and $A := A - \{x\}$. The algorithms continue until $A = \emptyset$.

Algorithm PRTF. Select job x and a machine μ with $\min_\mu \min_{x \in A}(2 \times R_x(k_\mu, \phi_\mu) + p_x)$. Break ties by choosing x which can start the earliest.

Algorithm ECT. Select job x and a machine μ with $\min_\mu \min_{x \in A} C_x(k_\mu, \phi_\mu)$. Break ties by choosing x which can start the earliest.

Algorithm PRTS. *Step* 1: Select job i and a machine μ with $\min_\mu \min_{i \in A} (2 \times R_i(k_\mu, \phi_\mu) + p_i)$. Break ties by choosing i which can start the earliest.

Step 2: Select job j with $\min_{j \in A} C_j(k_\mu, \phi_\mu)$. Break ties by choosing j which can start the earliest.

Step 3: IF $f_{ij}(k_\mu, \phi_\mu) \leqslant f_{ji}(k_\mu, \phi_\mu)$,

THEN $x = i$
ELSE $x = j$.

5. A branch-and-bound algorithm

The proposed branch-and-bound algorithm uses a usual scheme. During the computation, we keep a list of unexplored nodes arranged in increasing order according to the lower bounds of nodes, with ties broken by a nonincreasing number of scheduled jobs. Each node represents a partial schedule which is also a partial list. The algorithms always try to develop the head of the list. The branching from a node consists of creating son nodes by adding an unscheduled job to the end of the partial list which consists of assigning this unscheduled job to a machine so that it can be completed the earliest [26]. Before any new node is created, a dominance property (Theorem 3) is checked. For each node of

the search tree, which cannot be eliminated by dominance property, a lower bound is calculated. If the lower bound is greater than or equal to the total completion time of a known complete solution, this node is also eliminated.

The lower bound used is lb_{NCY}. The initial solution is given by the best solution found by PRTF, ECT and PRTS. For each node, we also compute an upper bound using the PRTS heuristic, because it seems the most efficient among the heuristics presented so far.

6. Computational experiments

This section describes the computational results on randomly generated instances. The branch-and-bound algorithm was programmed and tested with C compiler on a biprocessor G 5000 HP Workstation, 440 MHz and 1 Giga RAM.

For each job j, an integer processing times p_j from the uniform distribution $[1, 100]$ was generated. After the processing time are generated, the setup times s_{ij} are set to $a \times \min(p_i, p_j)$, where a is a coefficient randomly generated in $[A, B]$, where $[A, B] \in \{[0.01, 0.1], [0.05, 0.1], [0.1, 0.2], [0.2, 0.5], [0.1, 0.5]\}$. Since the range of release dates is likely to influence the effectiveness of the lower bounds, release dates were generated from the uniform distribution $[0, 50.5 \times N \times \alpha/M]$. Five α values $\{0.6, 0.8, 1.50, 2.0, 3.0\}$ were considered.

Problems with 5, 10, 15, 20, 25, 30 or 40 jobs were generated with 2, 3 or 5 machines. Ten instances for each of the 10 α values were solved for a total of 250 instances for each problem size.

The computational result of the branch-and-bound method are given in Table 1, where "Ngene", "NoRs", "Time", "gapheur" and "gapopt" represent, respectively, the average number of nodes generated, the number of problems unsolved among 100 instances, the average computational time in CPU seconds, the average deviation of the heuristic solution from the optimal solution and the average deviation of the optimal solution from the lower bound lb_{NCY}, i.e.,

$$gapheur = 100 \times \left(\sum C_j(heuristic) - \sum C_j(optimal)\right) \Big/ \sum C_j(optimal) \ ,$$

$$gapopt = 100 \times \left(\sum C_j(optimal) - lb_{NCY}\right) \Big/ \sum C_j(optimal) \ .$$

Table 1
Average of computational results of branch-and-bound

n	*m*	*Ngene*	*NoRs*	*Time*	*gapheur*	*gaplb*
5	2	3.50	0	0.60	0.72	4.01
	3	3.75	0	0.60	0.43	2.33
	5	0	0	0	0	0
10	2	20.76	0	677.96	2.23	6.38
	3	48.60	0	1007.52	0.92	4.83
	5	103.90	0	1363.12	0.57	2.73
15	2	458	0	1303.60	2.33	4.32
	3	308	0	1975	1.98	4.22
	5	1289	0	11 789	0.88	4.46
20	2	4613.84	0	17 305	1.27	3.73
	3	1712.88	19	32214.56	4.10	4.93
	5	11100.92	22	618 374	3.86	5.55
25	2	13775.32	4	35 305	1.23	3.45
	3	6712.88	19	233214.56	3.87	4.43
	5	78687.92	22	618 374	4.56	4.78
30	2	19 784	10	188 305	2.03	3.12
35	2	47235.2	17	345 305	1.87	3.63
40	2	123428.56	23	765 305	1.54	4.03

Table 2
Average of nodes considered

[A, B]	*m*	2	3	5
[0.01, 0.1]	*Ngene*	25162.32	865.29	935.27
	NoRs	0	0	0
[0.05, 0.1]	*Ngene*	25236.63	920.79	986.38
	NoRs	0	0	0
[0.1, 0.2]	*Ngene*	25625.31	965.39	935.98
	NoRs	0	0	0
[0.2, 0.5]	*Ngene*	28257.54	1312.45	1385.53
	NoRs%	4	8	6
[0.1, 0.5]	*Ngene*	29348.87	1523.13	1409.06
	NoRs%	10	10	12

Through the numerical tests, we can note that the number of generated nodes, the computation time and the number of unsolved problems increase with the problems size (i.e. the number of jobs and the number of machines). The unsolved problems are essentially those generated with the values $\{0.6, 0.8, 1, 1.25\}$ of α and $a \in [0.1, 0.5] \cup [0.2, 0.5]$. For large values of α, the release dates of the jobs are largely dispersed which reduces the number of active schedules (Table 2).

The average number of problems unsolved accounts for 6% for 2 machines, 9% for 3 machines and 9% for 5 machines on the whole of the tested problems. The difference between the average of the upper bounds compared to the average of the optimal solutions is 1.65% for 2 machines, 1.41% for 3 machines and 1.23% for 5 machines. The difference between the average of the optimal solutions compared to the average of the lower bounds is 4.08% for 2 machines, 2.59% for 3 machines and 2.19% for 5 machines. The developed method has efficiently solved problems with 40 jobs in the case of 2, 3, 5 machines.

7. Conclusions

We presented a necessary and sufficient condition that can be considered as a priority rule and a new dominant subset, called *FS*-active subset defined on the basis of this condition. We also presented good heuristic algorithms which construct *FS*-active schedules using the new priority rule.

We have also proved some dominance properties and a polynomial complexity lower bound, which were proved to be very efficient to prune the search tree. With the *FS*-active subset, the proved dominance properties, the proposed heuristic and also the lower bound, we succeeded in conceiving a branch-and-bound algorithm capable of solving problems with up to 40 jobs and 2 machines.

References

[1] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 1979;5:287–326.
[2] Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB. Sequencing and scheduling: algorithms and complexity. In: Graves SC, Rinnooy Kan AHG, Zipkin P, editors. Handbooks in operations research and management science, vol. 4: logistics of production and inventory. Amsterdam: North-Holland; 1993.
[3] Frederickson GN, Hecht MS, Kim CE. Approximation algorithms for some routing problems. SIAM Journal on Computing 1978. p. 178–93.
[4] França PM, Gendreau M, Laporte G, Müller FM. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. International Journal of Production Economics 1996;43:79–89.
[5] Gendreau M, Laporte G, Guimarães EM. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. European Journal of Operational Research 2001;133:183–9.
[6] Brucker P, Thiele O. A branch and bound method for the general-shop problem with sequence dependent setup-times. OR Spectrum 1996;18: 145–61.
[7] Baptiste P, Le Pape C. Scheduling a single machine to minimize a regular objective function under setup constraints. Discrete Optimization 2005;2:83–99.

[8] Yalaoui F, Chu C. An efficient heuristic approach for parallel machine scheduling with job splitting and sequence dependent setup times. IIE Transactions 2003;35(2):183–90.
[9] Chen ZL, Powell WB. Exact algorithms for scheduling multiple families of jobs on parallel machines. Naval Research Logistics 2003;50: 823–40.
[10] Allahverdi A, Aldowaisan T. Minimizing mean flowtime in a two-machine flowshop with sequence independent setup times. Computers & Operations Research 2000;27:111–27.
[11] Allahverdi A, Aldowaisan T. Two-machine flowshop scheduling to minimize total completion time with separate setup and removal times. International Journal of Industrial Engineering 2002;9(3):275–86.
[12] Rajendran C, Ziegler H. Scheduling to minimize the sum of weighted fowtime and weighted tardiness of jobs in a fowshop with sequence-dependent setup times. European Journal of Operational Research 2003;149:513–22.
[13] Du J, Leung JY-T, Young GH. Scheduling chain structured tasks to minimize makespan and mean flowtime. Information and Computing 1991;92:219–36.
[14] Yalaoui F, Chu C. A new exact method to solve the $Pm/r_i/\sum c_i$ schedule problem. International Journal of Production Economics 2006; 100(1): 168–79.
[15] Belouadah H, Potts CN. Scheduling of a single machine to minimize total weighted completion time. Discrete Applied Mathematics 1994;48: 201–18.
[16] Elmaghraby SE, Park SH. Scheduling jobs on a number of identical machines. AIIE Transactions 1974;6:1–13.
[17] Azizoglu M, Kirca O. On the minimization of total weighted flow time with identical and uniform parallel machines. European Journal of Operational Research 1999;113:91–100.
[18] Barnes JW, Brennan JJ. An improved algorithm for scheduling jobs on identical machines. AIIE Transactions 1977;9:23–31.
[19] Serin SC, Ahn S, Bishop AB. An improved branching scheme for the branch and bound procedure of scheduling n jobs on m machines to minimize total weighted flow time. International Journal of Production Research 1988;26:1183–91.
[20] Baptiste P. A note on scheduling multiprocessor tasks with identical processing times. Computers & Operations Research 2003;30:2071–8.
[21] Azizoglu M. Preemptive scheduling on identical parallel machines subjects to deadlines. European Journal of Operational Research 2003;148:205–10.
[22] Leonardi S, Raz D. Approximation total flow time on parallel machines. In: Proceedings of the 29th annual ACM symposium on the theory of computing. 1997. p. 110–9.
[23] Phillips CA, Stein C, Wein J. Scheduling jobs that arrive over time. In: Proceedings of the fourth workshop on algorithm and data structures. Lecture notes in computer science, vol. 95. 1995. p. 86–97.
[24] Phillips CA, Schulz AS, Shmoys DB, Stein C, Wein J. Improved bounds on relaxations of a parallel machine scheduling problem. Journal of Combinatorial Optimization 1998;1:413–26.
[25] Chu C. A branch and bound algorithm to minimize total tardiness with different release dates. Naval Research Logistics 1992;39:265–83.
[26] Schutten JMJ. List scheduling revisited. Operations Research Letters 1996;18:167–70.