SIAM J. Discrete Math.
Vol. 21, No. 2, pp. 313–328

A NEW APPROXIMATION ALGORITHM FOR THE NONPREEMPTIVE SCHEDULING OF INDEPENDENT JOBS ON IDENTICAL PARALLEL PROCESSORS[*]

GIUSEPPE PALETTA[†] AND PAOLAMARIA PIETRAMALA[‡]

Abstract. We consider the scheduling problem of n independent jobs on m identical parallel processors in order to minimize makespan, the completion time of the last job. We propose a new approximation algorithm that iteratively combines partial solutions to the problem. The worst-case performance ratio of the algorithm is $\frac{z+1}{z} - \frac{1}{mz}$, where z is the number of initial partial solutions that are obtained by partitioning the set of jobs into z families of subsets which satisfy suitable properties. The computational behavior of our worst-case performance ratio provided encouraging results on three families of instances taken from the literature.

Key words. optimization, scheduling, approximation algorithm, worst-case performance ratio

AMS subject classifications. 68W25, 90C59

DOI. 10.1137/050634657

1. Introduction.

In this paper, we consider the scheduling problem of n independent jobs on m parallel machines. Each job i must be processed without interruption by only one of the m machines (nonpreemptive environment); the machines are identical, and thus the processing time a_i of the job i is independent of the machine processing it (identical parallel processor environment). The objective is to minimize the makespan, i.e., the total time required to complete all the jobs. Using the standard three field classification scheme (Graham et al. (1979)), this problem is usually denoted as $P||C_{max}$.

The problem is well known to be NP-hard in strong sense for an arbitrary $m \geq 2$ (Garey and Johnson (1979) and Ullman (1976)). It is unlikely that there exists a polynomial-time algorithm for producing a minimal makespan, and so we consider heuristic algorithms in the hope of providing near-optimal results. If we look at the approximation algorithms for this problem, we can refer to the list scheduling family of Graham (1966, 1969), which includes the largest processing time (LPT), and to the multifit decreasing (MFD) scheduling algorithm of Coffman, Garey, and Johnson (1978). The LPT algorithm runs in $O(n\log(n) + nm)$-time and has the worst-case ratio equal to $\frac{4}{3} - \frac{1}{3m}$, whereas the MFD algorithm runs in $O(n\log(n) + knm)$-time and has a worst-case ratio equal to $\frac{13}{11} + \frac{1}{2^k}$ (Friesen (1984) and Yue (1990)), where k represents the number of times that a bin packing problem is solved by using the lowest fit decreasing algorithm (Coffman, Garey, and Johnson (1978)).

The literature of parallel machine scheduling problems, on the heuristic algorithms, has been extensively reviewed by Cheng and Sin (1990), Lawler et al. (1993), and Mokotoff (2001). An overview of existing results and of recent research areas is presented in the handbook edited by Leung (2004).

[*]Received by the editors June 28, 2005; accepted for publication (in revised form) November 15, 2006; published electronically April 27, 2007.
http://www.siam.org/journals/sidma/21-2/63465.html

[†]Dipartimento di Economia e Statistica, Università della Calabria, 87036 Arcavacata di Rende, Cosenza, Italy (g.paletta@unical.it).

[‡]Dipartimento di Matematica, Università della Calabria, 87036 Arcavacata di Rende, Cosenza, Italy (pietramala@mat.unical.it).

The algorithm proposed in this paper is based on the idea of combining iteratively partial solutions until a feasible solution for the scheduling problem is obtained. The initial partial solutions are obtained by partitioning the set of jobs into z families of subsets satisfying suitable properties that will be described below.

The algorithm runs in $O(n\log(n)+nm)$-time as the *LPT* algorithm and produces a solution with a worst-case performance ratio equal to $\frac{z+1}{z} - \frac{1}{mz}$ if $z > 1$, where z is the number of initial partial solutions, whereas, if $z = 1$, the algorithm produces an optimal solution. This bound of the algorithm is very poor whenever z is less than 3. However, as it will be described below, z is at least equal to $\lceil \frac{n}{m\rho} \rceil$ (where ρ is the ratio between $\max_{i=1,\dots,n}\{a_i\}$ and $\min_{i=1,\dots,n}\{a_i\}$ and $\lceil x \rceil$ denotes the smallest integer not less than x) so that when $n > 6m\rho$, our algorithm works very well compared to *LPT* and *MFD* algorithms. Also, our bound is comparable with $\frac{t+1}{t} - \frac{1}{mt}$ (where t is the least number of jobs on any processor), that is, the worst-case performance bound given by Coffman and Sethi (1976) for the *LPT* approach.

The paper is organized as follows. Section 2 presents the definitions and the properties of the partitions that are used to design the algorithm. Section 3 contains the description of the algorithm. Section 4 includes the statements of the inductive assertions about the efficiency and performance of our algorithm. Finally, the computational results obtained from three families of instances taken from the literature are presented in section 5.

2. Definitions and preliminary results. Let $I = \{1,\dots,i,\dots,n\}$ be the set of n independent jobs, $M = \{1,\dots,j,\dots,m\}$ be the set of m identical parallel processors, and $A = \{a_1,\dots,a_i,\dots,a_n\}$ be the set of processing times of the jobs. Let us start with a precise definition of the partitions to be studied.

Let $\mathcal{I} = \{I_1^1,\dots,I_j^1,\dots,I_m^1,\dots,I_1^r,\dots,I_j^r,\dots,I_m^r,\dots,I_1^z,\dots,I_j^z,\dots,I_m^z\}$ be a partition of the set I. Let $a_j^r := \sum_{i\in I_j^r} a_i$ be the sum of the processing times of the jobs belonging to I_j^r, $r = 1,\dots,z$ and $j = 1,\dots,m$.

DEFINITION 1. *A partition*

$$\mathcal{I} = \{I_1^1,\dots,I_j^1,\dots,I_m^1,\dots,I_1^r,\dots,I_j^r,\dots,I_m^r,\dots,I_1^z,\dots,I_j^z,\dots,I_m^z\}$$

*of the set I is called an α-*partition *if the following properties are satisfied:*

(a) $a_1^1 \ge \dots \ge a_j^1 \ge \dots \ge a_m^1 \ge \dots \ge a_1^r \ge \dots \ge a_m^r \ge \dots \ge a_1^z \ge \dots \ge a_m^z$.

(b) $a_1^r \le 2a_m^r$, $r = 1,\dots,z$.

(c) $a_m^r + a_m^z > a_1^r$, $r = 1,\dots,z-1$.

(d) a_j^r, $j = 1,\dots,m$ *and* $r = 1,\dots,z-1$, *is equal to the sum of at least an* $a_i \in A$ *so that* $a_i \ge a_1^z$.

We associate the *α-partition* $\mathcal{I}$ of I with the family $\mathcal{P} = \{\mathcal{I}^1,\dots,\mathcal{I}^r,\dots,\mathcal{I}^z\}$ of z partial solutions, where each $\mathcal{I}^r = \{I_1^r,\dots,I_j^r,\dots,I_m^r\}$, $r = 1,\dots,z$, represents the rth partial solution of the scheduling problem. In particular, with respect to the partial solution $\mathcal{I}^r$, $r = 1,\dots,z$, each I_j^r, $j = 1,\dots,m$, represents the set of jobs that must be performed by the machine j. Each $\mathcal{I}^r$, $r = 1,\dots,z$, is associated with the m-set $G^r = \{a_1^r,\dots,a_j^r,\dots,a_m^r\}$ that is referred to the set of total processing times of the job-sets of $\mathcal{I}^r$.

In view of property (a), the m elements of each G^r, $r = 1,\dots,z$, are sorted in nonincreasing order with respect to their size.

In what follows, let us denote by $\Delta^r := a_1^r - a_m^r$, $r = 1,\dots,z$, the gap between the maximum and the minimum element of the m-set G^r.

Example 1. Let us focus on the instance $I = \{1,2,\dots,25\}$, $M = \{1,2,3,4,5\}$, and $A = \{47,46,39,33,31,31,31,27,27,26,25,25,25,24,23,19,19,19,18,18,9,6,6,5,2\}$.

The partition $\mathcal{I} = \{I_1^1 = \{1\}, I_2^1 = \{2\}, I_3^1 = \{4, 22, 23\}, I_4^1 = \{3, 24\}, I_5^1 = \{5, 21, 25\}, I_1^2 = \{6\}, I_2^2 = \{7\}, I_3^2 = \{8\}, I_4^2 = \{9\}, I_5^2 = \{10\}, I_1^3 = \{11\}, I_2^3 = \{12\}, I_3^3 = \{13\}, I_4^3 = \{14\}, I_5^3 = \{15\}, I_1^4 = \{16\}, I_2^4 = \{17\}, I_3^4 = \{18\}, I_4^4 = \{19\}, I_5^4 = \{20\}\}$ is an *α-partition* of I because it satisfies all the properties of Definition 1. This *α-partition* is associated with the family $\mathcal{P} = \{\mathcal{I}^1, \mathcal{I}^2, \mathcal{I}^3, \mathcal{I}^4\}$ of 4 partial solutions, where $\mathcal{I}^1 = \{I_1^1, I_2^1, I_3^1, I_4^1, I_5^1\}$, $\mathcal{I}^2 = \{I_1^2, I_2^2, I_3^2, I_4^2, I_5^2\}$, $\mathcal{I}^3 = \{I_1^3, I_2^3, I_3^3, I_4^3, I_5^3\}$, and $\mathcal{I}^4 = \{I_1^4, I_2^4, I_3^4, I_4^4, I_5^4\}$. The 5-set $G^1 = \{47, 46, 33 + 6 + 6, 39 + 5, 31 + 9 + 2\}$ and the gap $\Delta^1 = 5$ are associated with the partial solution $\mathcal{I}^1$, $G^2 = \{31, 31, 27, 27, 26\}$ and the gap $\Delta^2 = 5$ are associated with $\mathcal{I}^2$, $G^3 = \{25, 25, 25, 24, 23\}$ and the gap $\Delta^3 = 2$ are associated with $\mathcal{I}^3$, and $G^4 = \{19, 19, 19, 18, 18\}$ and the gap $\Delta^4 = 1$ are associated with $\mathcal{I}^4$.

Let $\mathcal{I} = \{I_1^1, \dots, I_j^1, \dots, I_m^1, \dots, I_1^r, \dots, I_j^r, \dots, I_m^r, \dots, I_1^{z-1}, \dots, I_m^{z-1}, I_1^z, \dots, I_j^z, \dots, I_p^z\}$ be a partition of the set I with $p \le m$; let $a_j^r := \sum_{i \in I_j^r} a_i$ be the sum of the processing times of the jobs belonging to I_j^r, $r = 1, \dots, z-1$ and $j = 1, \dots, m$; let $u_j := \sum_{i \in I_j^z} a_i$ be the sum of the processing times of the jobs belonging to I_j^z, $j = 1, \dots, p$ and $u_j := 0$, $p < j \le m$.

Definition 2. *A partition*

$$\mathcal{I} = \{I_1^1, \dots, I_j^1, \dots, I_m^1, \dots, I_1^r, \dots, I_j^r, \dots, I_m^r, \dots, I_1^{z-1}, \dots, I_j^{z-1}, \dots,$$
$$I_m^{z-1}, I_1^z, \dots, I_j^z, \dots, I_p^z\}$$

*of the set I is called a β-*partition *if the following properties are satisfied:*

(a) $a_1^1 \ge \dots \ge a_j^1 \ge \dots \ge a_m^1 \ge \dots \ge a_1^r \ge \dots \ge a_m^r \ge \dots \ge a_1^{z-1} \ge \dots \ge a_m^{z-1} \ge u_1 \ge \dots \ge u_p$.

(b) $a_1^r \le 2a_m^r$, $r = 1, \dots, z-1$, *and* $u_1 > 2u_m$.

(c) $a_m^r + u_p > a_1^r$, $r = 1, \dots, z-1$.

(d) a_j^r, $j = 1, \dots, m$ *and* $r = 1, \dots, z-1$, *is equal to the sum of at least an* $a_i \in A$ *so that* $a_i \ge u_1$.

We associate the *β-partition* $\mathcal{I}$ of I with the family $\mathcal{P} = \{\mathcal{I}^1, \dots, \mathcal{I}^r, \dots, \mathcal{I}^{z-1}, \mathbf{I}\}$ of z partial solutions, where each $\mathcal{I}^r = \{I_1^r, \dots, I_j^r, \dots, I_m^r\}$, $r = 1, \dots, z-1$, represents the rth partial solution and $\mathbf{I} = \{I_1^z, \dots, I_j^z, \dots, I_p^z, \emptyset_{p+1}, \dots, \emptyset_m\}$ represents the zth partial solution, where $m-p$ machines do not perform jobs. In particular, with respect to the partial solution $\mathbf{I}$, I_j^z, $j = 1, \dots, p$, represents the set of jobs that are performed by the machine j, and $\emptyset_j$, $j = p+1, \dots, m$, indicates that the machine j does not perform jobs. Each $\mathcal{I}^r$, $r = 1, \dots, z-1$, is associated with the m-set $G^r = \{a_1^r, \dots, a_j^r, \dots, a_m^r\}$, whereas $\mathbf{I}$ is associated with the m-set $U = \{u_1, \dots, u_p, 0, \dots, 0\}$ that is referred to the set of the total processing times of the job-sets of $\mathbf{I}$.

In view of property (a), the elements of each G^r, $r = 1, \dots, z-1$, and the elements of U are sorted in nonincreasing order with respect to their size.

As before, let us denote by $\Delta^r := a_1^r - a_m^r, r = 1, \dots, z-1$, the gap between the maximum and the minimum element of the m-set G^r and by $\Delta^U := u_1 - u_m$ the gap between the maximum and the minimum element of the m-set U.

Example 2. Let us focus on the instance $I = \{1, 2, \dots, 25\}$, $M = \{1, 2, 3, 4, 5\}$, and $A = \{50, 48, 44, 42, 39, 36, 35, 34, 32, 30, 29, 28, 28, 28, 28, 27, 26, 26, 23, 11, 10, 9, 9, 2, 1\}$. The partition $\mathcal{I} = \{I_1^1 = \{1\}, I_2^1 = \{5, 20\}, I_3^1 = \{2\}, I_4^1 = \{3\}, I_5^1 = \{4, 24\}, I_1^2 = \{6\}, I_2^2 = \{7\}, I_3^2 = \{8\}, I_4^2 = \{9\}, I_5^2 = \{10\}, I_1^3 = \{11\}, I_2^3 = \{12\}, I_3^3 = \{13\}, I_4^3 = \{14\}, I_5^3 = \{15\}, I_1^4 = \{16\}, I_2^4 = \{17\}, I_3^4 = \{18\}, I_4^4 = \{19\}, I_5^4 = \{21, 22, 25\}, I_1^5 = \{23\}\}$ is a *β-partition* of I because it satisfies all the properties of Definition 2. This *β-partition* is associated with the family $\mathcal{P} = \{\mathcal{I}^1, \mathcal{I}^2, \mathcal{I}^3, \mathcal{I}^4, \mathbf{I}\}$

of 5 partial solutions, where $\mathcal{I}^1 = \{I_1^1, I_2^1, I_3^1, I_4^1, I_5^1\}$, $\mathcal{I}^2 = \{I_1^2, I_2^2, I_3^2, I_4^2, I_5^2\}$, $\mathcal{I}^3 = \{I_1^3, I_2^3, I_3^3, I_4^3, I_5^3\}$, $\mathcal{I}^4 = \{I_1^4, I_2^4, I_3^4, I_4^4, I_5^4\}$, and $\mathbf{I} = \{I_1^5, \emptyset_2, \emptyset_3, \emptyset_4, \emptyset_5\}$. The 5-set $G^1 = \{50, 39 + 11, 48, 44, 42 + 2\}$ and the gap $\Delta^1 = 6$ are associated with the partial solution $\mathcal{I}^1$, $G^2 = \{36, 35, 34, 32, 30\}$ and the gap $\Delta^2 = 6$ are associated with $\mathcal{I}^2$, $G^3 = \{29, 28, 28, 28, 28\}$ and the gap $\Delta^3 = 1$ are associated with $\mathcal{I}^3$, $G^4 = \{27, 26, 26, 23, 10 + 9 + 1\}$ and the gap $\Delta^4 = 7$ are associated with $\mathcal{I}^4$, and $U = \{9, 0, 0, 0, 0\}$ and the gap $\Delta^U = 9$ are associated with $\mathbf{I}$.

Definition 3. *Let $\mathcal{I}^r$ and $\mathcal{I}^q$ be two partial solutions related to an α-*partition *(β-*partition*) of I. Let us define "*combination*" among $\mathcal{I}^r$ and $\mathcal{I}^q$ ($\mathcal{I}^r \uplus \mathcal{I}^q$) as the m-family*

$$\mathcal{I}^r \uplus \mathcal{I}^q = \{I_1^r \cup I_m^q, \ldots, I_j^r \cup I_{m-j+1}^q, \ldots, I_m^r \cup I_1^q\}.$$

This new family corresponds to a new partial solution on the jobs belonging to $\mathcal{I}^r$ and $\mathcal{I}^q$. In particular, the set $I_j^r \cup I_{m-j+1}^q$, for each $j = 1, \ldots, m$, represents the jobs performed by the machine j. The total processing time needed for the machines to perform all the jobs belonging to $\mathcal{I}^r \uplus \mathcal{I}^q$ is computed by using the following definition.

Definition 4. *Let G^r and G^q be the sets of processing times of the partial solutions $\mathcal{I}^r$ and $\mathcal{I}^q$ related to an α-*partition *(β-*partition*) of I. Let us define "*sum*" among G^r and G^q ($G^r \oplus G^q$) as the m-set (not necessarily ordered)*

$$G^r \oplus G^q = \{a_1^r + a_m^q, \ldots, a_j^r + a_{m-j+1}^q, \ldots, a_m^r + a_1^q\}.$$

Notice that $a_j^r + a_{m-j+1}^q$ represents the total processing time needed for machine j to perform all the jobs belonging to $I_j^r \cup I_{m-j+1}^q$, and $\mathcal{I}^r \uplus \mathcal{I}^q$ is a partial solution that is not related to an *α-partition* or to a *β-partition* of I because the elements of $G^r \oplus G^q$ are not sorted in decreasing order with respect to their size.

Example 3. Let G^1 and G^2 be the sets of processing times of the partial solutions $\mathcal{I}^1$ and $\mathcal{I}^2$ which are related to the *α-partition* of Example 1; then

$$\begin{aligned} G^1 \oplus G^2 &= \{a_1^1 + a_5^2, a_2^1 + a_4^2, a_3^1 + a_3^2, a_4^1 + a_2^2, a_5^1 + a_1^2\} \\ &= \{47 + 26, 46 + 27, 45 + 27, 44 + 31, 42 + 31\} = \{73, 73, 72, 75, 73\} \end{aligned}$$

and

$$\begin{aligned} \mathcal{I}^1 \uplus \mathcal{I}^2 &= \{I_1^1 \cup I_5^2, I_2^1 \cup I_4^2, I_3^1 \cup I_3^2, I_4^1 \cup I_2^2, I_5^1 \cup I_1^2\} \\ &= \{\{1, 10\}, \{2, 9\}, \{4, 22, 23, 8\}, \{3, 24, 7\}, \{5, 21, 25, 6\}\}, \end{aligned}$$

where, for example, $a_5^1 + a_1^2 = 73$ represents the total processing time needed for machine 5 to perform the jobs belonging to $I_5^1 \cup I_1^2 = \{5, 21, 25, 6\}$.

The "*sum*" operator satisfies the properties indicated in Lemmas 1 and 3.

Lemma 1. *Let G^r and G^q be the sets of processing times of the partial solutions $\mathcal{I}^r$ and $\mathcal{I}^q$ which are relative to an α-partition (β-partition) of I. Put $S = G^r \oplus G^q$ and $\Delta^S = \max\{S\} - \min\{S\}$. Then*

1. $\max\{S\} \leq 2 \min\{S\}$;
2. $\Delta^S \leq \max\{\Delta^r, \Delta^q\}$;
3. $\Delta^S < a_m^z$ *($\Delta^S < u_p$).*

Proof.

Statement 1.

Let

$$\max\{S\} = a^r_k + a^q_{m-k+1} \text{ for some } k, \quad 1 \leq k \leq m,$$

and

$$\min\{S\} = a^r_l + a^q_{m-l+1} \text{ for some } l, \quad 1 \leq l \leq m.$$

Then

$$\max\{S\} = a^r_k + a^q_{m-k+1} \leq \text{(property (a))} \leq a^r_1 + a^q_1 \leq \text{(property (b))}$$

$$\leq 2a^r_m + 2a^q_m \leq \text{(property (a))} \leq 2(a^r_l + a^q_{m-l+1}) = 2\min\{S\}.$$

Statement 2.
Let

$$\Delta^S = (a^r_k + a^q_{m-k+1}) - (a^r_l + a^q_{m-l+1}) = (a^r_k - a^r_l) + (a^q_{m-k+1} - a^q_{m-l+1}).$$

It can happen that $a^r_k - a^r_l \geq 0$ or not. We examine both cases separately.
First case: $a^r_k - a^r_l \geq 0$.
An immediate consequence of the definition of S and of property (a) gives rise to

$$a^q_{m-k+1} - a^q_{m-l+1} \leq 0.$$

It follows that

$$\Delta^S = (a^r_k - a^r_l) + (a^q_{m-k+1} - a^q_{m-l+1}) \leq a^r_k - a^r_l \leq \text{(property (a))} \leq a^r_1 - a^r_m = \Delta^r.$$

Second case: $a^r_k - a^r_l \leq 0$.
An immediate consequence of the definition of S and of property (a) gives rise to

$$a^q_{m-k+1} - a^q_{m-l+1} \geq 0.$$

Then

$$\Delta^S = (a^r_k - a^r_l) + (a^q_{m-k+1} - a^q_{m-l+1}) \leq a^q_{m-k+1} - a^q_{m-l+1}$$

$$\leq \text{(property (a))} \leq a^q_1 - a^q_m = \Delta^q.$$

Consequently,

$$\Delta^S \leq \max\{\Delta^r, \Delta^q\}.$$

Statement 3.
Property (c) ensures that

$$\Delta^r < a^z_m \quad \text{and} \quad \Delta^q < a^z_m (\Delta^r < u_p \quad \text{and} \quad \Delta^q < u_p).$$

From Statement 2, we deduce that

$$\Delta^S < a^z_m (\Delta^S < u_p). \qquad \square$$

LEMMA 2. *Let $G^r = \{a^r_1, \ldots, a^r_j, \ldots, a^r_m\}$ and $U = \{u_1, \ldots, u_p, 0, \ldots, 0\}$, $p < m$, be the sets of processing times of the partial solutions $\mathcal{I}^r$ and $\mathbf{I}$ related to a β-partition of I. Set $S = U \oplus G^r$. Then*

1. $\min\{S\} = a^r_{m-p}$;
2. $\max\{S\} = u_k + a^r_{m-k+1}$ for some $k, 1 \le k \le p$.

Proof.

Statement 1.

Let

$$S = \{u_1 + a^r_m, \dots, u_p + a^r_{m-p+1}, \dots, a^r_{m-p}, \dots, a^r_1\}.$$

Property (a) yields $a^r_1 \ge \cdots \ge a^r_{m-p}$. Moreover, for each $j = 1, \dots, p$,

$$a^r_{m-p} \le a^r_1 < (\text{property (c)}) < u_p + a^r_m \le (\text{property (a)}) \le u_j + a^r_{m-j+1},$$

and hence we deduce that $\min\{S\} = a^r_{m-p}$.

Statement 2.

It follows from property (c) that, for all $j = 1, \dots, p$,

$$u_j + a^r_{m-j+1} \ge a^r_1 \ge (\text{property (a)}) \ge a^r_2 \ge \cdots \ge a^r_{m-p},$$

from which we deduce that

$$\max\{S\} = \max_{j=1,\dots,p}\{u_j + a^r_{m-j+1}\}. \qquad \square$$

Lemma 3. *Let G^r and U be the sets of processing times of the partial solutions $\mathcal{I}^r$ and $\mathbf{I}$ related to a β-partition of I. Set $S = U \oplus G^r$ and $\Delta^S = \max\{S\} - \min\{S\}$. Then*

1. $\max\{S\} \le 2\min\{S\}$;
2. $\Delta^S \le \max\{\Delta^U, \Delta^r\} = \Delta^U$;
3. $\Delta^S \le u_1$.

Proof.

Statement 1.

We distinguish two cases.

First case: $p < m$.

Lemma 2 states that $\min\{S\} = a^r_{m-p}$ and $\max\{S\} = u_k + a^r_{m-k+1}$ for some $k, 1 \le k \le p$. Then

$$\max\{S\} = u_k + a^r_{m-k+1} \le (\text{property (a)}) \le a^r_{m-p} + a^r_{m-p} = 2a^r_{m-p} = 2\min\{S\}.$$

Second case: $p = m$.

First,

$$a^r_1 < (\text{property (c)}) < u_m + a^r_m \le (\text{property (a)}) \le u_j + a^r_{m-j+1},\ j = 1, \dots, m.$$

So, we obtain

$$a^r_1 \le \min_{j=1,\dots,m}\{u_j + a^r_{m-j+1}\}. \qquad \text{(i)}$$

Now

$$\max\{S\} = \max_{j=1,\dots,m}\{u_j + a^r_{m-j+1}\} \le (\text{property (a)}) \le u_1 + a^r_1$$

$$\le (\text{property (a)}) \le 2a^r_1 \le \text{from (i)} \le 2\min\{S\}.$$

Statement 2.

We distinguish two cases.

First case: $p < m$.

First, we show that $\max\{\Delta^U, \Delta^r\} = \Delta^U$. In fact, we have

$$\Delta^r = a_1^r - a_m^r < (\text{property (c)}) < u_p \le (\text{property (a)}) \le u_1 = \Delta^U, \quad r = 1, \dots, z-1.$$

Moreover, Lemma 2 ensures that, for some k, $1 \le k \le p$, one has

$$\Delta^S = u_k + a_{m-k+1}^r - a_{m-p}^r \le (\text{because } a_{m-k+1}^r - a_{m-p}^r \le 0)$$

$$\le u_k \le (\text{property (a)}) \le u_1 = \Delta^U.$$

Second case: $p = m$.

First, we show that $\max\{\Delta^U, \Delta^r\} = \Delta^U$. In this case, we obtain

$$\Delta^r - \Delta^U = a_1^r - a_m^r - (u_1 - u_m) < (\text{property (c)})$$

$$< a_m^r + u_m - a_m^r - u_1 + u_m = 2u_m - u_1 < (\text{property (b)}) < 0.$$

Now let $\max\{S\} = u_k + a_{m-k+1}^r$ for some $k, 1 \le k \le m$, and $\min\{S\} = u_l + a_{m-l+1}^r$ for some $l, 1 \le l \le m$. So,

$$\Delta^S = (u_k - u_l) + (a_{m-k+1}^r - a_{m-l+1}^r).$$

If $u_k - u_l \ge 0$ (hence $a_{m-k+1}^r - a_{m-l+1}^r \le 0$), then

$$\Delta^S \le u_k - u_l \le (\text{property (a)}) \le u_1 - u_m = \Delta^U.$$

If $u_k - u_l < 0$ (hence $a_{m-k+1}^r - a_{m-l+1}^r \ge 0$), then

$$\Delta^S \le a_{m-k+1}^r - a_{m-l+1}^r \le (\text{property (a)}) \le a_1^r - a_m^r = \Delta^r \le \Delta^U.$$

Summarizing, we derive

$$\Delta^S \le \Delta^U.$$

Statement 3.

From Statement 2, we deduce that

$$\Delta^S \le \Delta^U = u_1 - u_m \le u_1. \qquad \square$$

DEFINITION 5. *Let G^r and G^q be the sets of processing times of the partial solutions $\mathcal{I}^r$ and $\mathcal{I}^q$ related to an α*-partition *(β*-partition*) of I. Let us define* "ordered sum" *among G^r and G^q as the ordered m-set* Ord*($G^r \oplus G^q$) whose elements are the elements of $G^r \oplus G^q$ sorted in nonincreasing order with respect to their size.*

DEFINITION 6. *Let $\mathcal{I}^r$ and $\mathcal{I}^q$ be two partial solutions related to an α*-partition *(β*-partition*) of I. Let us define* "ordered combination" *among $\mathcal{I}^r$ and $\mathcal{I}^q$ as the m-family* Ord*($\mathcal{I}^r \uplus \mathcal{I}^q$) whose sets are those of $\mathcal{I}^r \uplus \mathcal{I}^q$ sorted so that the jth element of* Ord*($G^r \oplus G^q$) represents the total processing time of the jth job-set of* Ord*($\mathcal{I}^r \uplus \mathcal{I}^q$).*

Thus, we have $Ord(G^r \oplus G^q) = Ord(G^q \oplus G^r)$ and $Ord(\mathcal{I}^r \uplus \mathcal{I}^q) = Ord(\mathcal{I}^q \uplus \mathcal{I}^r)$. In the following, the partial solution $Ord(\mathcal{I}^r \uplus \mathcal{I}^q)$ is called "*combined partial solution*" to distinguish it from the initial partial solutions obtained by the procedure described in the next section.

Example 4. With reference to Example 3, we have

$$Ord(G^1 \oplus G^2) = Ord(73, 73, 72, 75, 73) = \{75, 73, 73, 73, 72\}$$

and

$$Ord(\mathcal{I}^1 \uplus \mathcal{I}^2) = Ord(\{1, 10\}, \{2, 9\}, \{4, 22, 23, 8\}, \{3, 24, 7\}, \{5, 21, 25, 6\})$$
$$= \{\{3, 24, 7\}, \{5, 21, 25, 6\}, \{1, 10\}, \{2, 9\}, \{4, 22, 23, 8\}\}.$$

Lemma 4. *Let $\mathcal{P} = \{\mathcal{I}^1, \dots, \mathcal{I}^r, \dots, \mathcal{I}^z\}$ be a family of z partial solutions related to an α-partition of I. Then* $\{\mathrm{Ord}(\dots(\mathrm{Ord}(\mathrm{Ord}(\mathcal{I}^1 \uplus \mathcal{I}^2) \uplus \mathcal{I}^3) \uplus \dots) \uplus \mathcal{I}^r), \mathcal{I}^{r+1}, \dots, \mathcal{I}^z\}$, *$r = 2, \dots, z$, is a family of $z - r + 1$ partial solutions that is related to an α-partition of I.*

Proof. Property (a) is guaranteed because the "*ordered combination*" operator is iteratively performed among the first two partial solutions related to an *α-partition.* Properties (b) and (c) are guaranteed, respectively, by Statements 1 and 3 of Lemma 1. Property (d) is guaranteed because it was guaranteed by the partial solutions in $\mathcal{P}$. □

By using the same arguments in the proof of Lemma 4, we have the following lemma.

Lemma 5. *Let $\mathcal{P} = \{\mathcal{I}^1, \dots, \mathcal{I}^r, \dots, \mathcal{I}^{z-1}, \mathbf{I}\}$ be a family of z partial solutions related to a β-partition of I. Then* $\{\mathrm{Ord}(\dots(\mathrm{Ord}(\mathrm{Ord}(\mathcal{I}^1 \uplus \mathcal{I}^2) \uplus \mathcal{I}^3) \uplus \dots) \uplus \mathcal{I}^r), \mathcal{I}^{r+1}, \dots, \mathcal{I}^{z-1}, \mathbf{I}\}$, *$r = 2, \dots, z-1$, is a family of $z - r + 1$ partial solutions that is related to a β-partition of I.*

Example 5. With reference to Example 1, we have $\{Ord(\mathcal{I}^1 \uplus \mathcal{I}^2), \mathcal{I}^3, \mathcal{I}^4\}$, which is a family of three partial solutions, where $Ord(\mathcal{I}^1 \uplus \mathcal{I}^2)$ and $Ord(G^1 \oplus G^2)$ are reported in Example 4.

3. Algorithms.

The proposed algorithm, which uses the procedure named *IPS* (initial partial solutions) described later, partitions the jobs so as to obtain an *α-partition* or a *β-partition* of I, i.e., a family of initial partial solutions to the scheduling problem. Then, as indicated in Lemmas 4 and 5, it iteratively combines, in turn, the initial partial solutions with the current *combined partial solution* by utilizing the *ordered combination* operator. The iterative process is repeated until a solution of the scheduling problem is obtained. The algorithm, named *MPS* (multiprocessor scheduling), can be summarized as follows.

Algorithm *MPS*.

Initialization

- Use the procedure *IPS* to obtain the family $\mathcal{P}$ of z initial partial solutions that are related to an *α-partition* or to a *β-partition* of I. If *IPS* returns only a partial solution, then Stop (the solution is optimal);
- Set $C = \{C_1 = 0, \dots, C_j = 0, \dots, C_m = 0\}$ and $\mathcal{T} = \{\mathcal{T}_1 = \emptyset_1, \dots, \mathcal{T}_j = \emptyset_j, \dots, \mathcal{T}_m = \emptyset_m\}$, where $\mathcal{T}$ represents the current *combined partial solution* and C_j the processing time of the job-set $\mathcal{T}_j$.

Construction

- For $r = 1, \dots, z-1$, compute $C = Ord(C \oplus G^r)$ and $\mathcal{T} = Ord(\mathcal{T} \uplus \mathcal{I}^r)$, where G^r and $\mathcal{I}^r$ have been provided by the *IPS* procedure;
- If *IPS* returns an *α-partition* of I, then compute $C = Ord(C \oplus G^z)$ and $\mathcal{T} = Ord(\mathcal{T} \uplus \mathcal{I}^z)$, where G^z and $\mathcal{I}^z$ have been provided by the *IPS* procedure;

- If *IPS* returns a β-*partition* of I, then compute $C = Ord(C \oplus U)$ and $\mathcal{T} = Ord(\mathcal{T} \uplus \mathbf{I})$, where U and $\mathbf{I}$ have been provided by the *IPS* procedure.

At the end, *MPS* returns $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_j, \ldots, \mathcal{T}_m\}$ and $C = \{C_1, \ldots, C_j, \ldots, C_m\}$, where $\mathcal{T}$ represents a solution of the problem and C the completion times of the machines. In particular, each $\mathcal{T}_j$, $j = 1, \ldots, m$, represents the jobs assigned to machine j and C_j the processing time needed for each machine j to perform the jobs which are assigned to it.

3.1. Determining the initial partial solutions. The procedure *IPS*, which finds the initial partial solutions related to an α-*partition* or a β-*partition* of I, first orders the jobs so that $a_1 \geq \cdots \geq a_i \geq \cdots \geq a_n$. Then it builds an α-*partition* or a β-*partition* of I by processing the jobs in turn, starting with job 1. Now we suppose that, when job i must be inserted, there are either $z-1$ families $\mathcal{I}^r$, $r = 1, \ldots, z-1$, that satisfy $a_1^r \leq 2a_m^r$ and the family $\mathbf{I}$ that satisfies $u_1 > 2u_m$, or there are z families $\mathcal{I}^r$. Also, we suppose that all families are sorted in nonincreasing order with respect to their processing times. Let $\Pi = \{\pi(1), \pi(2), \ldots\}$ be the permutation of the indexes of the current partial solutions of type $\mathcal{I}^r$ so that $\Delta^{\pi(1)} \geq \Delta^{\pi(2)} \geq \cdots$. Then *IPS* inserts job i as follows. First, it selects, among the ordered families of type $\mathcal{I}^r$, the family $\mathcal{I}^q = \mathcal{I}^{\pi(1)}$ with the biggest gap $\Delta^q = \Delta^{\pi(1)}$ between the processing times of the first and the last job-sets. Now if $a_i \leq \Delta^q$, then job i is inserted into the last job-set of $\mathcal{I}^q$, that is, into I_m^q; the job-sets of $\mathcal{I}^q$ are sorted in nonincreasing order with respect to their processing times, and Π is arranged in nonincreasing order with respect to the gaps. If $a_i > \Delta^q$ and all job-sets of $\mathbf{I}$ are not empty, then job i is inserted into the last set of $\mathbf{I}$, that is, into I_m^z, and the job-sets of $\mathbf{I}$ are sorted in nonincreasing order with respect to their processing times. If $a_i > \Delta^q$ and some job-sets of $\mathbf{I}$ are empty, then job i is inserted into the first job-set empty of $\mathbf{I}$. In this case, it is not necessary to sort the job-sets because they are already ordered. If $a_i > \Delta^q$ and all job-sets of $\mathbf{I}$ are empty, then the job i is inserted into the first set of $\mathbf{I}$, and z is increased by one. Also, if job i is inserted into $\mathbf{I}$ and $2u_m \geq u_1$, then $\mathbf{I}$ becomes $\mathcal{I}^z$, index z is inserted into Π, that is again arranged, and $\mathbf{I}$ is placed equal to m empty sets. The procedure can be formally described as follows.

```
    PROCEDURE IPS
    Initialization
      - Order the jobs so that $a_1 \geq \cdots \geq a_i \geq \cdots \geq a_n$. Set $z = 1$ ($z$ = number of
        initial partial solutions);
      - Consider $\mathbf{I} = \{I_1^z = \{1\}, I_2^z = \emptyset_2, \ldots, I_m^z = \emptyset_m\}$, $U = \{u_1 = a_1, u_2 = 0, \ldots,
        u_m = 0\}$;
      - Set $\Delta^U = a_1$, $p = 1$ ($p$ represents the number of elements of $U$ not equal to
        0), and $\Pi = \emptyset$.
Construction
    For each $i = 2, \ldots, n$
      - If $\Pi \neq \emptyset$, then set $q = \pi(1)$, $\Delta^{max} = \Delta^q$, and consider the $m$-set $G^q$ and
        the family $\mathcal{I}^q$, else set $\Delta^{max} = 0$;
      - If $a_i \leq \Delta^{max}$, then
          - $a_m^q = a_m^q + a_i$, $I_m^q = I_m^q \cup \{i\}$, sort the elements of the $m$-set $G^q$
            so that $a_1^q \geq \cdots \geq a_j^q \geq \cdots \geq a_m^q$, and arrange the family $\mathcal{I}^q$ so
            that $a_j^q$ is the total time required by the jobs belonging to $I_j^q$;
          - $\Delta^q = a_1^q - a_m^q$, and arrange the set $\Pi$ so that $\Delta^{\pi(1)} \geq \Delta^{\pi(2)} \geq \cdots$;
      End If $a_i \leq \Delta^{max}$;
      - If $a_i > \Delta^{max}$, then
```

```
            - If all job-sets of $\mathbf{I}$ are empty ($\Delta^U = \infty$), then $z = z + 1$,
              $\mathbf{I} = \{I_1^z = \{i\}, I_2^z = \emptyset_2, \dots, I_m^z = \emptyset_m\}$,
              $U = \{u_1 = a_i, u_2 = 0, \dots, u_m = 0\}$, and $p = 1$;
            - If all job-sets of $\mathbf{I}$ are not empty ($\Delta^U \neq \infty$ and $p = m$), then
              $u_m = u_m + a_i$, $I_m^z = I_m^z \cup \{i\}$, sort the elements of the set $U$ so that
              $u_1 \geq \cdots \geq u_m$, and arrange the family $\mathbf{I}$ so that $u_j$ is the total time
              required by the jobs belonging to $I_j^z$;
            - If some job-sets of $\mathbf{I}$ are empty ($\Delta^U \neq \infty$ and $1 \leq p < m$), then
              $p = p + 1$, $u_p = a_i$, $I_p^z = I_p^z \cup \{i\}$;
            - $\Delta^U = u_1 - u_m$;
            - If $p = m$ and $2u_m \geq u_1$, then $G^z = U$, $\mathcal{I}^z = \mathbf{I}$, $\Delta^z = \Delta^U$, insert $z$
              into $\Pi$, and arrange the set $\Pi$ so that $\Delta^{\pi(1)} \geq \Delta^{\pi(2)} \geq \cdots$,
              $\mathbf{I} = \{\emptyset_1, \dots, \emptyset_j, \dots, \emptyset_m\}$, $U = \{0, , 0, \dots, 0\}$, and $\Delta^U = \infty$;
        End If $a_i > \Delta^{max}$;
    End For $i$.
```

It is easy to show that the procedure *IPS* produces a partition so that the first job-set of each family is a singleton, and, consequently, the processing time of the last job-set of a family is greater than the processing time of the first job-set of the next family. Also, *IPS* produces an *α-partition* of I if $U = \{u_1 = 0, \dots, u_m = 0\}$ or a *β-partition* of I if $U \neq \{u_1 = 0, \dots, u_m = 0\}$.

4. Efficiency and performance of the algorithm.

4. Efficiency and performance of the algorithm. With regard to efficiency, it is easy to show that the *MPS* algorithm, as the *LPT* algorithm, runs in $O(n\log(n) + nm)$-time, that is, the running time of the procedure *IPS*.

Now we consider the performance of the *MPS* algorithm.

Let us use $C_{max} = \max_{j=1,\dots,m}\{C_j\}$ and C^*_{max} to denote the makespan of the *MPS* solution and the optimal makespan, respectively. Denote by $C_{min} = \min_{j=1,\dots,m}\{C_j\}$ and by $\Delta := C_{max} - C_{min}$.

To obtain the performance ratio of our algorithm, it is necessary to show the following lemmas.

LEMMA 6. *If the procedure* IPS *returns a family of z initial partial solutions relative to an α-partition of I, then $a_m^z \geq \Delta$.*

Proof. Property (c) states that, for $q = 1, \dots, z-1$,

$$\Delta^q = a_1^q - a_m^q < a_m^z.$$

Moreover, we obtain from property (b) that

$$\Delta^z = a_1^z - a_m^z \leq 2a_m^z - a_m^z = a_m^z.$$

Then

$$a_m^z \geq \max_{q=1,\dots,z}\{\Delta^q\}.$$

Using Lemmas 1 and 4, we note that

$$\Delta \leq \max_{q=1,\dots,z}\{\Delta^q\}.$$

Consequently, $a_m^z \geq \Delta$. □

LEMMA 7. *If the procedure* IPS *returns a family of z initial partial solutions relative to a β-partition of I, then $u_1 \geq \Delta$.*

Proof. Using Lemmas 3 and 5, we note that

$$\Delta \leq \max\{\Delta^1, \ldots, \Delta^{z-1}, \Delta^U\} = \Delta^U. \qquad \text{(i)}$$

Since

$$\Delta^U = u_1 - u_m \leq u_1,$$

it follows from (i) that $u_1 \geq \Delta$. ◻

We are now able to find a lower bound on C^*_{max} with respect to the number z of initial partial solutions relative to an *α-partition* or a *β-partition* of I produced by *IPS*.

THEOREM 1. *If the procedure* IPS *returns a family $\mathcal{P}$ of z initial partial solutions relative to an α-partition or a β-partition of I, then*

$$C^*_{max} \geq z\Delta.$$

Proof. We need to distinguish two cases.

First case: $\mathcal{P}$ is relative to an α-partition of I.

Of course,

$$C^*_{max} \geq z a^z_m.$$

Moreover, Lemma 6 ensures that $a^z_m \geq \Delta$.

It follows that

$$C^*_{max} \geq z a^z_m \geq z\Delta.$$

Second case: $\mathcal{P}$ is relative to a β-partition of I.

In view of property (d), the number of elements $a_i \in A$ so that $a_i \geq u_1$ in $a^1_1, \ldots, a^1_m, \ldots, a^{z-1}_1, \ldots, a^{z-1}_m$ is at least $m(z-1)$. Focusing only on these $m(z-1)$ elements $a_i \geq u_1$, together with u_1, we get $m(z-1)+1$ jobs with processing times greater than or equal to u_1 (we can ignore the other jobs). Then

$$C^*_{max} \geq \left\lceil \frac{m(z-1)+1}{m} \right\rceil u_1 = \left\lceil (z-1) + \frac{1}{m} \right\rceil u_1 = z u_1.$$

Moreover, Lemma 7 ensures that $u_1 \geq \Delta$, and so

$$C^*_{max} \geq z\Delta. \qquad ◻$$

We are now in a position to show results about the performance ratio of the proposed algorithm.

THEOREM 2. *If the procedure* IPS *returns a family $\mathcal{P}$ of z initial partial solutions relative to an α-partition (or a β-partition) of I, then*

$$\frac{C_{max}}{C^*_{max}} \leq \frac{z+1}{z} - \frac{1}{mz} \qquad \text{if } z > 1$$

and

$$C_{max} = C^*_{max} \qquad \text{if } z = 1.$$

Proof. If the procedure *IPS* returns only the m-set G^1 (or only the set U), we have an optimal solution. In fact, $a_1^1 = a_1 = \max\{a_i\}$ (respectively, $u_1 = a_1$) and $C^*_{max} \geq a_1$. If $z > 1$, let $C'_{min} = C_{min} + \frac{1}{m}\Delta$. Then

$$C_{max} = C_{min} + \Delta = C'_{min} - \frac{1}{m}\Delta + \Delta = C'_{min} + \left(1 - \frac{1}{m}\right)\Delta.$$

Since

$$C^*_{max} \geq C'_{min},$$

it follows that

$$C_{max} \leq C^*_{max} + \left(1 - \frac{1}{m}\right)\Delta.$$

Moreover, Theorem 1 states that $C^*_{max} \geq z\Delta$. We can conclude that

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{1}{C^*_{max}}\left(1 - \frac{1}{m}\right)\Delta \leq 1 + \frac{1}{z\Delta}\left(1 - \frac{1}{m}\right)\Delta = 1 + \frac{m-1}{mz} = \frac{z+1}{z} - \frac{1}{mz}. \qquad \square$$

We exhibit an example that shows that the worst-case performance ratio of our algorithm cannot be improved.

Example 6. Let us focus on the instance $I = \{1,2,3,4,5\}, M = \{1,2\}$, and $A = \{3,3,2,2,2\}$. The optimal solution is obtained when jobs 1 and 2 are performed by machine 1 and jobs 3, 4, and 5 are performed by machine 2. This solution is associated with the completion time of the machines $\{3+3, 2+2+2\} = \{6,6\}$, and the related makespan is equal to 6.

The procedure *IPS* returns $\mathcal{I}^1 = \{I_1^1 = \{1\}, I_2^1 = \{2\}\}$, $\mathcal{I}^2 = \{I_1^2 = \{3\}, I_2^2 = \{4\}\}$, $\mathbf{I} = \{I_1^3 = \{5\}, \emptyset\}$, $G^1 = \{3,3\}$, $G^2 = \{2,2\}$, and $U = \{2,0\}$. Altogether, *MPS* returns $\mathcal{T} = Ord(Ord(\mathcal{I}^1 \uplus \mathcal{I}^2) \uplus \mathbf{I}\} = \{\mathcal{T}_1 = \{1,5,4\}, \mathcal{T}_2 = \{2,3\}\}$ and $C = Ord(Ord(G^1 \oplus G^2) \oplus U) = \{2+3+2, 0+3+2\} = \{7,5\}$, and the related makespan is equal to 7. Then

$$\frac{C_{max}}{C^*_{max}} = \frac{7}{6} \quad \text{and} \quad \frac{z+1}{z} - \frac{1}{mz} = \frac{4}{3} - \frac{1}{6} = \frac{7}{6}.$$

4.1. Estimate of the worst-case performance ratio. We estimated the worst-case performance ratio of our algorithm. In the following, it is supposed that $a_1 \geq \cdots \geq a_i \geq \cdots \geq a_n$ and $\rho = \frac{a_1}{a_n}$.

Proposition 1. *The procedure* IPS *returns z initial partial solutions with $z \geq \lceil \frac{n}{m\rho} \rceil \geq 1$.*

Proof. The procedure *IPS* returns z initial partial solutions where each I_1^r, $r = 1,\ldots,z$, is a singleton.

When *IPS* returns an *α-partition* of I, $a_1^r \leq a_1^1 = a_1$, $r = 1,\ldots,z$, it follows that $\sum_{j=1,\ldots,m} a_j^r \leq m a_1^r \leq m a_1$, $r = 1,\ldots,z$. Hence

$$\sum_{i=1,\ldots,n} a_i = \sum_{r=1,\ldots,z} \sum_{j=1,\ldots,m} a_j^r \leq \sum_{r=1,\ldots,z} m a_1 \leq z m a_1.$$

When *IPS* returns a *β-partition* of I, $a_1^r \leq a_1^1 = a_1$, $r = 1,\ldots,z-1$, and $u_1 \leq a_1^1 = a_1$, it follows that $\sum_{j=1,\ldots,m} a_j^r \leq m a_1^r \leq m a_1$, $r = 1,\ldots,z-1$, and $\sum_{j=1,\ldots,p} u_j \leq p a_1^r \leq m a_1^r \leq m a_1$. Hence

$$\sum_{i=1,\ldots,n} a_i = \sum_{r=1,\ldots,z-1} \sum_{j=1,\ldots,m} a_j^r + \sum_{j=1,\ldots,p} u_j \leq \sum_{r=1,\ldots,z-1} m a_1 + m a_1 \leq z m a_1.$$

Summarizing, we obtain

$$z \geq \left\lceil \frac{1}{ma_1} \sum_{i=1,\ldots,n} a_i \right\rceil \geq \left\lceil \frac{na_n}{ma_1} \right\rceil = \left\lceil \frac{n}{m\rho} \right\rceil. \qquad \square$$

Through Proposition 1 and Theorem 2, we immediately obtain the following result.

COROLLARY 1. *The algorithm* MPS *returns a solution with*

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{1}{\left\lceil \frac{n}{\rho m} \right\rceil} - \frac{1}{m \left\lceil \frac{n}{\rho m} \right\rceil}.$$

This estimate is very poor when ρ is very large, whereas, when $\rho \leq \frac{n}{6m}$, the estimate is better than the worst-case ratio of the *LPT* and *MFD* algorithms.

When $a_1 < 2a_n$, we obtain the following results.

PROPOSITION 2. *For all instances so that $a_1 < 2a_n$, the procedure* IPS *returns $z = \lceil \frac{n}{m} \rceil \geq 1$ initial partial solutions.*

Proof. When *IPS* returns an *α-partition* of I, each I_j^r, $r = 1, \ldots, z$ and $j = 1, \ldots, m$, is a singleton, while when *IPS* returns a *β-partition* of I, each I_j^r, $r = 1, \ldots, z-1$ and $j = 1, \ldots, m$, is a singleton, just like each I_j^z, $j = 1, \ldots, p$, is also a singleton.

It follows that $z = \lceil \frac{n}{m} \rceil \geq 1$. $\square$

COROLLARY 2. *For all instances so that $a_1 < 2a_n$, the algorithm* MPS *returns a solution with*

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{1}{\left\lceil \frac{n}{m} \right\rceil} - \frac{1}{m \left\lceil \frac{n}{m} \right\rceil}.$$

This estimate is better than the worst-case ratio of the *LPT* and *MFD* algorithms when $\frac{n}{m} \geq \frac{11}{2}$.

5. Computational results.

First, we analyze the computational behavior of the average worst-case ratio bound $\frac{z+1}{z} - \frac{1}{mz}$ of *MPS* and the average worst-case ratio bound $\frac{t+1}{t} - \frac{1}{mt}$ given by Coffman and Sethi (1976) for the *LPT* approach, and then we compare our bound with the *MFD* bound. The *MPS* and *LPT* algorithms have been coded in Fortran 77 and run on the three families of instances that have been used by Frangioni, Necciari, and Scutellà (2004).

In the first family of instances, denoted NONUNIFORM, the number of machines m were $5, 10$, and 25, the number of jobs n were $100, 500$, and 1000, and the intervals for the integer processing times were $[1, 100], [1, 1000]$, and $[1, 10000]$. Ten instances were randomly generated for each choice of m, n and of the processing time intervals, for a total of 270 instances. The generator, which was presented by Frangioni, Necciari, and Scutellà, when an interval $[a, b]$ of the processing times is given, produces instances where 98% of the processing times are uniformly distributed in the interval $[(b-a)0.9, b]$, while the remaining processing times fall within the interval $[a, (b-a)0.02]$. The generator is available from http://www.di.unipi.it/di/groups/optimize/Data/index.html.

The last two families of instances have been derived from difficult bin packing instances and are available at the OR-Library of J. E. Beasley from http://mscmga.ms.ic.ac.uk/jeb/orlib/binpackinfo.html.

Table 1
Behavior of the average worst-case ratio bounds on BINPACK instances.

		LPT		*MPS*	
m	n	$\frac{t+1}{t}-\frac{1}{mt}$	*sec.*	$\frac{z+1}{z}-\frac{1}{mz}$	*sec.*
[46,52]	120	1.32654	0.1	1.32654	0.1
[97,106]	250	1.33005	0.0	1.33005	0.1
[196,207]	500	1.33167	0.0	1.33167	0.3
[393,411]	1000	1.33250	0.1	1.33250	0.6

Table 2
Behavior of the average worst-case ratio bounds on TRIPLET instances.

		LPT		*MPS*	
m	n	$\frac{t+1}{t}-\frac{1}{mt}$	*sec.*	$\frac{z+1}{z}-\frac{1}{mz}$	*sec.*
20	60	1.31666	0.0	1.31666	0.0
40	120	1.32500	0.0	1.32500	0.0
83	249	1.32931	0.0	1.32931	0.1
167	501	1.33133	0.1	1.33133	0.1

In each instance of the last two families, the number of machines m is the number of bins in the best known solution of bin packing instances. In the second family, denoted BINPACK, the number of jobs n were $120, 250, 500, 1000$, and the processing times were uniformly distributed in $[20, 100]$. Twenty instances were generated for each choice of n, for a total of 80 instances. In the third family, denoted TRIPLET, the number of jobs n were $60, 120, 249, 501$, and the processing times were in $[25, 50]$. Twenty instances were generated for each choice of n, for a total of 80 instances.

Tables 1, 2, and 3 compare the behavior of the average worst-case performance bounds in BINPACK, TRIPLET, and NONUNIFORM instances. For each algorithm, the entries give the average worst-case performance bound and the average running time (expressed in seconds, on a Pentium III, 933 MHz, 256 MbRAM, and including the sorting time). The worst-case performance bounds and the running times were averaged for each group of 10 NONUNIFORM instances (Table 3), whereas they were averaged for each group of 20 BINPACK and TRIPLET instances (Tables 1 and 2, respectively).

All the instances of the three families (see Tables 1, 2, and 3) were solved very quickly by both algorithms. The running time of *MPS* is only slightly higher than the running time of *LPT*.

On NONUNIFORM instances (Table 3), the average worst-case ratio bound of *MPS* is slightly better, whereas, on BINPACK and TRIPLET instances (Tables 1 and 2), *MPS* and *LPT* have the same average worst-case ratio bound. Because all TRIPLET instances satisfy the property $a_1 < 2a_n$, the average worst-case performance bound of *MPS* can be computed by using Corollary 2.

The average worst-case ratio bounds of *MPS* and *LPT* decrease as the ratio $\frac{n}{m}$ increase (Table 3), whereas, when the ratio $\frac{n}{m}$ is constant, the average worst-case performance bounds of *MPS* and *LPT* decrease as m decreases (Tables 1 and 2).

As shown in Table 3, the average worst-case ratio bounds were independent from the intervals within which the processing times of the jobs were generated (see each row of the table) because, for each interval $[a, b]$, the generator produced instances where 98% of the processing times were distributed in the interval $[(b - a)0.9, b]$.

TABLE 3
Behavior of the average worst-case ratio bounds on NONUNIFORM instances.

		$\mathbf{a_j} \in [\mathbf{1}, \mathbf{100}]$				$\mathbf{a_j} \in [\mathbf{1}, \mathbf{1000}]$				$\mathbf{a_j} \in [\mathbf{1}, \mathbf{10000}]$			
		LPT		***MPS_A***		***LPT***		***MPS_A***		***LPT***		***MPS_A***	
m	**n**	$\frac{t+1}{t} - \frac{1}{mt}$	**sec.**	$\frac{z+1}{z} - \frac{1}{mz}$	**sec.**	$\frac{t+1}{t} - \frac{1}{mt}$	**sec.**	$\frac{z+1}{z} - \frac{1}{mz}$	**sec.**	$\frac{t+1}{t} - \frac{1}{mt}$	**sec.**	$\frac{z+1}{z} - \frac{1}{mz}$	**sec.**
5	100	1.04000	0.0	1.04000	0.0	1.04000	0.0	1.04000	0.0	1.04000	0.0	1.04000	0.1
	500	1.00808	0.0	1.00802	0.1	1.00808	0.0	1.00807	0.0	1.00808	0.0	1.00807	0.1
	1000	1.00402	0.1	1.00402	0.1	1.00403	0.0	1.00402	0.0	1.00403	0.1	1.00403	0.0
10	100	1.09000	0.0	1.09000	0.0	1.09000	0.0	1.09000	0.0	1.09000	0.0	1.09000	0.0
	500	1.01800	0.0	1.01800	0.0	1.01800	0.0	1.01800	0.0	1.01800	0.0	1.01800	0.0
	1000	1.00909	0.1	1.00900	0.1	1.00909	0.1	1.00900	0.0	1.00909	0.0	1.00907	0.1
25	100	1.24000	0.0	1.24000	0.0	1.24000	0.0	1.24000	0.0	1.24000	0.0	1.24000	0.0
	500	1.04800	0.0	1.04800	0.1	1.04800	0.0	1.04800	0.2	1.04800	0.0	1.04800	0.3
	1000	1.02400	0.0	1.02400	0.0	1.02400	0.0	1.02400	0.0	1.02400	0.0	1.02400	0.0

The average worst-case ratio bounds $\frac{z+1}{z} - \frac{1}{mz}$ and $\frac{t+1}{t} - \frac{1}{mt}$ were always better than the worst-case ratio $\frac{13}{11} + \frac{1}{2^k} \approx 1.1818$ of the *MFD* algorithm, except in instances with $\frac{n}{m} \leq 4$.

The numerical results computed by using Corollary 1 on BINPACK and NONUNIFORM instances were not reported because, as expected ($n \ll 6m\rho$), they were very poor.

6. Conclusions. We have designed a new approximation algorithm, which runs in $O(n\log(n)+nm)$-time as the *LPT* algorithm of Graham, for the scheduling problem of independent jobs on identical parallel processors in order to minimize makespan. The worst-case performance ratio of the algorithm is $\frac{z+1}{z} - \frac{1}{mz}$, where z is the number of initial partial solutions that are obtained by partitioning the set of jobs into z families of subsets which satisfy suitable properties.

The computational results showed that our worst-case performance ratio outperforms the one of the *MFD* algorithm on a few instances taken from the literature, except in instances with $\frac{n}{m} \leq 4$. Also, our worst-case performance ratio bound is comparable to that given by Coffman et al. for the *LPT* algorithm.

An estimate of our bound is also given, and it was quite good when $n > 6m\rho$.

Acknowledgments. We are grateful both to the editor and to the referees for their suggestions, which helped us to improve the presentation of this paper.

REFERENCES

T. Cheng and C. Sin (1990), *A state of the art review of parallel machine scheduling research*, European J. Oper. Res., 47, pp. 271–292.

E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson (1978), *An application of bin-packing to multiprocessor scheduling*, SIAM J. Comput., 7, pp. 1–17.

E. G. Coffman, Jr., and R. Sethi (1976), *A generalized bound on LPT sequencing*, Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Bleue, 10, pp. 17–25.

A. Frangioni, E. Necciari, and M. G. Scutellà (2004), *A multi-exchange neighborhood for minimum makespan machine scheduling problems*, J. Comb. Optim., 8, pp. 195–220.

D. K. Friesen (1984), *Tighter bounds for the Multifit processor scheduling algorithm*, SIAM J. Computing, 13, pp. 170–181.

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

R. L. Graham (1966), *Bounds for certain multiprocessing anomalies*, Bell System Tech. J., 45, pp. 1563–1581.

R. L. Graham (1969), *Bounds on multiprocessing timing anomalies*, SIAM J. Appl. Math., 17, pp. 416–429.

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

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys (1993), *Sequencing and scheduling: Algorithms and complexity in logistics of production and inventory*, in Handbooks in Operation Research and Management Science, Vol. 4, North–Holland, Amsterdam, pp. 445–522.

J. Leung, ed. (2004), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapman and Hall/CRC, Boca Raton, FL.

E. Mokotoff (2001), *Parallel machine scheduling problems: A survey*, Asia-Pacific J. Oper. Res., 18, pp. 193–243.

J. D. Ullman (1976), *Complexity of sequencing problems*, in Computer and Job Shop Scheduling Theory, E. G. Coffman, ed., Wiley, New York, pp. 139–164.

M. Yue (1990), *On the exact upper bound for the multifit processor scheduling algorithm*, Ann. Oper. Res., 24, pp. 233–259.