ELSEVIER

Operations Research Letters 18 (1996) 223–232

operations research letters

Ordinal algorithms for parallel machine scheduling

Wei-Ping Liu[a,1], Jeffrey B. Sidney[b,*,1], André van Vliet[c]

[a] *Bell-Northern Research, P.O. Box 3511, Station C, Ottawa, Ontario, Canada K1Y 4H7*
[b] *Faculty of Administration, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5*
[c] *Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, Netherlands*

Received 1 February 1995; revised 1 September 1995

Abstract

The minimization of maximum completion time for scheduling n jobs on m identical parallel machines is an NP-hard problem for which many excellent heuristic algorithms have been developed. In this paper, the problem is investigated under the assumption that only limited information about the jobs is available. Specifically, processing times are not known for the jobs; rather, the ordering of the jobs by processing time is known.

For the cases of two and three parallel machines, algorithms which cannot be improved upon with respect to worst case performance ratio are developed. For the case of four parallel machines, an algorithm which is near optimal with respect to worst case performance ratio is developed. For arbitrary m, an algorithm which produces solutions whose value is at most five-thirds times the optimal value is presented. Finally, it is shown that as the number of machines gets arbitrarily large, the best possible ordinal algorithm has worst case performance ratio of at least 3/2.

Keywords: Heuristic scheduling; Parallel machine scheduling

1. Introduction

The problem of minimizing the maximum completion time for scheduling n jobs on m identical parallel machines (which is denoted by $P \| C_{\max}$ as in [5]), and its well-known relatives the partition problem and the subset-sum problem, have fascinated many researchers for several decades. Many excellent algorithms have been developed, and almost any (pragmatically) reasonable level of performance may be obtained in modest computer time using various approximation schemes. An excellent review may be found in [5].

The problem $P \| C_{\max}$ is defined as follows:

Given a set $J = \{1, \ldots, n\}$ of n jobs, where job i has non-negative processing time a_i, partition the job set into m subsets $J_1, \ldots, J_m$ so as to minimize the maximum sum of processing times of the jobs in any of the subsets.

* Corresponding author.

[1] This work was supported under Natural Sciences and Engineering Research Council (NSERC) of Canada operating grant number A2507.

SSDI 0167-6377(95)00058-5

In the context of machine scheduling, subset J_i contains the jobs which are assigned for processing on machine j.

In the current research, it is assumed that the values of the processing times a_i are unknown, but that the order of the jobs by non-increasing processing time is known, i.e., without loss of generality that $a_1 \geqslant a_2 \geqslant \cdots \geqslant a_n$. Algorithms will be developed which depend only upon this rank order data, and the quality of the algorithms will be evaluated by their worst case performance ratios with respect to the actual (unknown) data.

For example, suppose $m = 2$, $J = \{1, 2\}$ and by assumption $a_1 \geqslant a_2$. The optimal value is a_1, and this is achieved by the algorithm which places job i on machine i. The algorithm which places both jobs on machine 1 has a worst case performance ratio of 2, and this occurs for the case of $a_1 = a_2$.

More generally, algorithms which utilize only ordinal (rank) data rather than actual magnitudes will be called ordinal algorithms.

Optimal ordinal algorithms exist for a number of well-known problems. The shortest processing time rule optimally solves the single machine and the identical parallel machine scheduling problems with minimization of mean completion time as the objective [2], and the greedy algorithm solves the maximum weight sum problem over independent sets in a matroid [4]. A polynomial asymptotically exact algorithm for the two machine no wait flow shop problem which uses only ordinal data has also been developed [1]. Finally, Liu and Sidney use an ordinal data model similar to the one used in the current work for the bin packing problem [6] and for a packing problem with a target center of gravity [7].

Our main results (Section 2) are summarized in Table 1.

2. General upper bound

Let H denote the value of the heuristic solution given by any of the algorithms for some problem, and let OPT denote the corresponding optimal value. Thus, for any problem the measure of the quality of an algorithm will be given by the worst case performance ratio $\sup\{H/\mathrm{OPT}\}$, where the supremum is taken over all instances of the problem. Let α_i denote the sum of the processing times assigned to machine i (denoted by M_i), let n_i = the number of jobs assigned to machine i, and let $A_i^k = \sum_{j=i}^{k} a_j$. Finally, let $b = A_1^n = \sum_{i=1}^{n} a_i$. The following lemma is trivial to prove.

Lemma 1. *Suppose that* $a_1 \geqslant a_2 \geqslant \cdots \geqslant a_i$, $\sum_{j=1}^{i} \lambda_j \geqslant 1$ *and* $0 \leqslant \lambda_j$ $(1 \leqslant j \leqslant i)$. *Then* $a_i \leqslant \sum_{j=1}^{i} \lambda_j a_j$.

Lemmas 2 and 3 establish some general properties which provide direction in the search for "good" algorithms for $P \| C_{\max}$.

Table 1

Problem name	Number of machines	Worst case performance ratio	Ratio tight?
$P2 \| C_{\max}$	2	4/3	Yes
$P3 \| C_{\max}$	3	7/5	Yes
$P4 \| C_{\max}$	4	101/70	?
$Pm \| C_{\max}$	m	$1 + \dfrac{m-1}{m + \lceil m/2 \rceil}$	No
$Pm \| C_{\max}$	As $m \to \inf$	$\geqslant 3/2$	?

Lemma 2. *Suppose that an algorithm σ for $Pm \| C_{\max}$ does not have the following two properties:*
(i) *jobs 1 to m are assigned to different machines;*
(ii) *jobs i and $2m + 1 - i$ are assigned to the same machine.*
Then the algorithm has a worst case performance ratio of at least 3/2.

Proof. First, if $1 \leqslant i < k \leqslant m$ and jobs i and k are assigned to the same machine, then algorithm σ will give a performance ratio of at least 2 for the problem given by the data $a_j = 1$ $(1 \leqslant j \leqslant m)$ and $a_j = 0$ otherwise.

Suppose that condition (i) holds, but not condition (ii). Either of two cases must hold.

Case 1: If at least three jobs on the set $\{1, \ldots, 2m\}$ are assigned to one machine, then σ gives a performance ratio of at least 3/2 for the data given by $a_j = 1$ $(1 \leqslant j \leqslant 2m)$ and $a_j = 0$ otherwise (OPT = 2 in this case).

Case 2: If case 1 does not hold (nor condition (ii)), then there exist a machine h and two jobs i and k, $1 \leqslant i < k \leqslant 2m$, such that $i + k < 2m + 1$, and both i and k are assigned to machine h. These conditions imply that $i \leqslant m - 1$ and $k < 2m$. For the problem given by the data

$$a_j = \begin{cases} 1, & 1 \leqslant j \leqslant i, \\ 1/2, & i + 1 \leqslant j \leqslant 2m - i, \\ 0, & 2m - i + 1 = j, \end{cases}$$

algorithm σ gives a performance ratio of at least 3/2 (OPT = 1 in this case). □

Lemma 3. *Suppose that an algorithm σ for $Pm \| C_{\max}$ with n jobs schedules n_i jobs on machine i. Then the worst case performance ratio is at least* $\max_i \{n_i\} / \lceil n/m \rceil$.

Proof. The stated worst case performance ratio is achieved when all a_i's are equal. □

Note that $\mathrm{LB} = \max\{a_1, a_m + a_{m+1}, b/m\}$ is an obvious lower bound for problem size m.

Next we provide an algorithm for the m machine case, whose worst case performance ratio will be computed in Theorem 1, and then used in the analysis of the two machine case.

Algorithm P_m

Jobs are assigned to machines as follows: For $1 \leqslant i \leqslant \lfloor m/2 \rfloor$

$$\{a_i\} \cup \{a_{2m+1-i+k(m+\lceil m/2 \rceil)} | k \geqslant 0\}.$$

For $\lfloor m/2 \rfloor + 1 \leqslant i \leqslant m$

$$\{a_i\} \cup \{a_{2m+1-i+k(m+\lceil m/2 \rceil)} | k \geqslant 0\} \cup \{a_{3m+1-i+k(m+\lceil m/2 \rceil)} | k \geqslant 0\}.$$

The assignments given by algorithm P_m for the case of $m = 7$ are illustrated in Fig. 1.

Given any machine i, the notation $[h]$ denotes the index of the hth job to be assigned to machine i.

Lemma 4. $\alpha_i - a_i \leqslant x A^n_{i+1}$ *if, for each* $h \geqslant 2$, $[h] - [1] = [h] - i \geqslant (h - 1)(1/x)$,

Proof. Follows from Lemma 1. □

Theorem 1. *For $m \geqslant 2$, algorithm P_m gives a worst case performance ratio no greater than* $1 + (m - 1)/(m + \lceil m/2 \rceil)$ *and for all values of $m \geqslant 2$ there are instances of P_m which achieve this ratio.*

Machine 1:	1	14		25		36		....
Machine 2:	2	13		24		35		....
Machine 3:	3	12		23		34		....
Machine 4:	4	11	18	22	29	33	40	....
Machine 5:	5	10	17	21	28	32	39	....
Machine 6:	6	9	16	20	27	31	38	....
Machine 7:	7	8	15	19	26	30	37	....

Fig. 1. Illustration of Algorithm P_m for $m = 7$.

Proof. The proof of Theorem 1 begins with the following lemma. The proof of the lemma, which consists of simple algebraic manipulations, may be found in [8].

Lemma 5. *The following relationship holds*:

$$(\alpha_i - a_i) \leqslant \begin{cases} \dfrac{2A_{i+1}^n}{3(m-i+1)} & \text{if } m \text{ is even and } 1 \leqslant i \leqslant m-1, \\ \dfrac{2A_{i+1}^n}{3(m-i+1)+1} & \text{if } m \text{ is odd and } 1 \leqslant i \leqslant (m-1)/2, \\ \dfrac{2A_{i+1}^n}{3(m-i+1)} & \text{if } m \text{ is odd and } (m+1)/2+1 \leqslant i \leqslant m-1. \end{cases} \tag{1}$$

Returning to the proof of Theorem 1, we consider three cases.
Case 1: *m is even and* $1 \leqslant i \leqslant m-1$, *or m is odd and* $(m+1)/2+1 \leqslant i \leqslant m-1$.

$$\alpha_i = a_i + (\alpha_i - a_i) \leqslant a_i + \frac{2A_{i+1}^n}{3(m-i+1)} \quad \text{(by Lemma 5)}$$

$$= a_i - \frac{2A_1^i}{3(m-i+1)} + \frac{2b}{3(m-i+1)}$$

$$= \frac{2}{3(m-i+1)}\left(\frac{3(m-i+1)}{2}a_i - A_1^i\right) + \frac{2b}{3(m-i+1)}$$

$$\leqslant \frac{3m-5i+3}{3(m-i+1)}a_i + \frac{2b}{3(m-i+1)}$$

$$= \frac{3m-5i+3}{3(m-i+1)}a_i + \frac{2m}{3(m-i+1)}\left(\frac{b}{m}\right)$$

$$\leqslant \frac{5m-5i+3}{3(m-i+1)}\max\left\{a_i, \frac{b}{m}\right\}$$

$$\leqslant \begin{cases} \dfrac{5m-2}{3m}\,\mathrm{LB}, & m \text{ is even,} \\ \dfrac{5m-9}{3m-3}\,\mathrm{LB}, & m \text{ is odd and } (m+1)/2+1 \leqslant i \leqslant m-1. \end{cases}$$

Case 2: *m is odd and* $1 \leqslant i \leqslant (m-1)/2$.

$$\alpha_i = a_i + (\alpha_i - a_i) \leqslant a_i + \frac{2A^n_{i+1}}{3(m-i+1)+1} \quad \text{(by Lemma 5)}$$

$$= a_i - \frac{2A^i_1}{3(m-i+1)+1} + \frac{2b}{3(m-i+1)+1}$$

$$= \frac{2}{3(m-i+1)+1}\left(\frac{3(m-i+1)+1}{2}a_i - A^i_1\right) + \frac{2b}{3(m-i+1)+1}$$

$$\leqslant \frac{3m-5i+4}{3(m-i+1)+1}a_i + \frac{2b}{3(m-i+1)+1}$$

$$= \frac{3m-5i+4}{3(m-i+1)+1}a_i + \frac{2m}{3(m-i+1)+1}\left(\frac{b}{m}\right)$$

$$\leqslant \frac{5m-5i+4}{3(m-i+1)+1}\max\left\{a_i, \frac{b}{m}\right\}$$

$$\leqslant \frac{5m-1}{3m+1}\,\mathrm{LB}.$$

Case 3: $i = m$ *(even and odd) and* $i = (m+1)/2$ *for m odd.* The proof for this case, available in [8], is similar to those of cases 1 and 2, and is left as a reference.

With cases 1–3 proved, the first part of the proof, namely that the worst case performance ratio is bounded from above by $1 + (m-1)/(m + \lceil m/2 \rceil)$, is complete.

To show the tightness of the $1 + (m-1)/(m + \lceil m/2 \rceil)$ bound for each $m \geqslant 2$, consider a problem instance consisting of $(m-1)$ jobs of size 1 and $(m-1)(m + \lceil m/2 \rceil)$ jobs of size $1/(m + \lceil m/2 \rceil)$. One can easily verify that $\mathrm{OPT} = 2$ and that $\alpha_m = 2 + 2(m-1)/(m + \lceil m/2 \rceil)$. This gives

$$\frac{\alpha_m}{\mathrm{OPT}} = 1 + \frac{(m-1)}{(m + \lceil m/2 \rceil)}$$

as claimed.

This completes the proof of Theorem 1. □

3. Upper bound for two, three and four machines

While the above algorithm gives us a guarantee for all values of m, it would be desirable to improve upon this bound whenever possible. Procedure CREATE below takes as input the number of machines m and a desired worst case performance ratio r, and attempts to find an ordinal algorithm which achieves r.

Procedure CREATE

1. For $1 \leqslant i \leqslant m/r$

$$x_i = \frac{r-1}{m-i}, \quad \lambda_i = 0.$$

For $m/r < i \leqslant m-1$

$$x_i = \frac{r}{m}, \quad \lambda_i = \frac{ir}{m} - 1.$$

For $i = m$

$$x_m = \frac{2(r-1)}{(m-1)}, \quad \lambda_i = 0.$$

2. For $1 \leqslant i \leqslant m-1$

$$J_i = \{i\} \cup \left\{k > i: \left\lfloor \frac{k+\lambda_i - i}{x_i} \right\rfloor = \left\lfloor \frac{k-1+\lambda_i - i}{x_i} \right\rfloor + 1\right\}.$$

For $i = m$

$$J_m = \{m, m+1\} \cup \left\{k > m+1: \left\lfloor \frac{k-(m+1)}{x_m} \right\rfloor = \left\lfloor \frac{k-1-(m+1)}{x_m} \right\rfloor + 1\right\}.$$

3. If for each $k \geqslant 1$ the relation $\sum_{i=1}^{m} |\{h \in J_i | h \leqslant k\}| \geqslant k$ holds then an ordinal algorithm which achieves r may be obtained by altering J_i through executing steps (a) and (b) below as required:
   (a) eliminate a job from J_i;
   (b) replace a job h in J_i with a job $h' > h$.

Step 1 above generates an inequality which, when satisfied, guarantees that $\alpha_i \leqslant r\mathrm{LB}$. For example, if $1 \leqslant i \leqslant m-1$, the inequality is

$$\begin{aligned}
\alpha_i &\leqslant a_i + x_i\left(\frac{\lambda_i}{x_i} a_i + b - A_1^i\right) \\
&\leqslant a_i(1 + \lambda_i - ix_i) + mx_i\left(\frac{b}{m}\right) \quad (\text{since } ia_i \leqslant A_1^i) \\
&\leqslant (1 + \lambda_i - ix_i + mx_i)\mathrm{LB} \quad (\text{since } 1 + \lambda_i - ix_i \geqslant 0) \\
&= r\mathrm{LB} \quad (\text{by step 1}).
\end{aligned} \tag{2}$$

For $i = m$, similar analysis yields $\alpha_m \leqslant r\mathrm{LB}$ provided $r \leqslant 1 + (m-1)/(m+1)$, a condition which will be satisfied for all "useful" values of r. Thus, provided $r \leqslant 1 + (m-1)/(m+1)$, the inequalities in step 1 guarantee that the bound of r is satisfied.

In step 2, for each $i \leqslant m$ a set J_i which satisfies the inequalities of step 1 is found. For $i \leqslant m-1$, the defining conditions in step 2 are based on the observation that $1 + \lfloor (k + \lambda_i - i)/x_i \rfloor$ represents the maximum number of elements $\leqslant k$ that can be assigned to J_i so that Lemma 1 applied to J_i will yield (3). For $i = m$, the corresponding expression is

$$2 + \left\lfloor \frac{k-(m+1)}{x_m} \right\rfloor.$$

Step 3 states that if for each k the total of these maxima is at least k, then there is a feasible ordinal algorithm which achieves r, and at least one such algorithm is easily generated from the sets given in step 2.

Application of Procedure CREATE yields algorithms for the two, three and four machine cases as follows:

Two machines: *Algorithm* $P(2)$

With $m = 2$ and $r = 4/3$, Procedure CREATE produces the partition P_2 (the special case of algorithm P_m for $m = 2$), so we take $P(2) = P_2$.

Three machines: *Algorithm* $P(3)$

With $m = 3$ and $r = 7/5$, Procedure CREATE gives $x_1 = 1/5$, $x_2 = x_3 = 2/5$ and, at the end of step 2,

$$J_1 = \{1\} \cup \{6 + 5i | i \geqslant 0\},$$

$$J_2 = \{2\} \cup \{5 + 5i | i \geqslant 0\} \cup \{7 + 5i | i \geqslant 0\},$$

$$J_3 = \{3, 4\} \cup \{7 + 5i | i \geqslant 0\} \cup \{9 + 5i | i \geqslant 0\}.$$

One realization of step 3 yields the algorithm P_m for $m = 3$, so we take $P(3) = P_3$.

Four machines: *Algorithm* $P(4)$

With $m = 4$ and $r = 13/9$, Procedure CREATE allows us a choice in step 3, one possible result of which is

$$J_1 = \{1\} \cup \{8 + 7i | i \geqslant 0\},$$

$$J_2 = \{2\} \cup \{7 + 14i | i \geqslant 0\} \cup \{11 + 14i | i \geqslant 0\} \cup \{16 + 14i | i \geqslant 0\},$$

$$J_3 = \{3\} \cup \{6 + 14i | i \geqslant 0\} \cup \{9 + 14i | i \geqslant 0\} \cup \{12 + 14i | i \geqslant 0\} \cup \{14 + 14i | i \geqslant 0\} \cup \{18 + 14i | i \geqslant 0\},$$

$$J_4 = \{4, 5\} \cup \{10 + 14i | i \geqslant 0\} \cup \{13 + 14i | i \geqslant 0\} \cup \{17 + 14i | i \geqslant 0\} \cup \{19 + 14i | i \geqslant 0\}.$$

Theorem 2. *Algorithms* $P(2)$, $P(3)$, *and* $P(4)$ *yield worst case performance ratios of* 4/3, 7/5, *and* 101/70 *respectively. These bounds are tight for* $m = 2$ *and* 3. *For* $m = 4$, *a lower bound on worst case performance ratio is* 23/16 (*so that the proven worst case performance ratio of* 101/70 *is less than* 0.4% *above the lower bound*).

Proof. The stated upper bounds for $P(2)$ and $P(3)$ are guaranteed by the use of Procedure CREATE to synthesize the algorithms. While CREATE provides a guarantee of 13/9 for $P(4)$, we shall prove the better bound of 101/70.

To prove the lower bounds, we deal with each algorithm separately.

To prove tightness for $P(2)$, we may without loss of generality assume that $n = 4$, and that a_1 is assigned to machine M_1. If an algorithm assigns at least one of a_2, a_3 or a_4 to M_1, then for $a_1 = 3$, $a_2 = a_3 = a_4 = 1$ we have OPT $= 3$ and $H \geqslant 4$. On the other hand, if an algorithm assigns a_2, a_3 and a_4 to M_2, then for $a_1 = a_2 = a_3 = a_4 = 1$ we have OPT $= 2$ and $H = 3$.

We now prove tightness for $P(3)$. By Lemma 2, we may assume that jobs i and $(7 - i)$ are assigned to machine i, for $1 \leqslant i \leqslant 3$. Choose $n = 11$. Then (i) $n_1 \leqslant 2$, for otherwise the data set $a_1 = 1$, $a_2 = \cdots = a_{11} = \frac{1}{5}$ gives $H \geqslant \frac{7}{5}$ and OPT $= 1$; (ii) $n_2 \leqslant 4$, for otherwise the data set $a_1 = a_2 = 1$ and $a_3 = \cdots = a_{11} = \frac{1}{9}$ gives $H \geqslant \frac{13}{9}$ and OPT $= 1$; (iii) job 7 must be assigned to M_3, for otherwise the data set $a_1 = a_2 = 1$, $a_3 = \cdots = a_7 = \frac{1}{5}$, $a_8 = \cdots = a_n = 0$ gives $H \geqslant \frac{7}{5}$ and OPT $= 1$; (iv) $n_3 \leqslant 4$, for otherwise the data set $a_1 = \cdots = a_4 = 1$, $a_5 = \cdots = a_7 = \frac{1}{3}$ and $a_8 = \cdots = a_{11} = \frac{1}{4}$ gives $H \geqslant 2 + \frac{1}{3} + \frac{1}{4}(2) = \frac{17}{6}$ and OPT $= 2$. Obviously one of (i), (ii) and (iv) cannot be true since $n = 11$. Therefore it is impossible to improve the ratio 7/5.

The proof of the lower bound of 23/16 is conceptually straightforward but somewhat lengthy, and is left as a reference [8].

In order to show that the bound will be achieved we can take the following example:

$p_1 = p_2 = 70,\ p_3 = \cdots = p_{11} = 14,\ p_{12} = \cdots = p_{25} = 1.$ □

4. Lower bound

For m machines and n jobs, the set of ordinal algorithms is finite. Let T_{mn} denote this set, and let τ_{mn} give the smallest worst case performance ratio P_{mn} among the algorithms in T_{mn}. Then $\sup\{P_{mn} | n \geqslant 1\} \equiv 1 + e_m$ gives the smallest worst case performance ratio over all m machine problems. We show below that $e_m \geqslant \frac{1}{2}$ for sufficiently large m.

Theorem 3. *Let $n \geqslant 2m$. Then $m \geqslant 18$ implies that $e_m \geqslant 1/2$, i.e., the worst case performance ratio is $\geqslant 3/2$ for all algorithms in T_{mn}.*

Proof. Assume that $1 + e_m < 3/2$. By Lemma 2, we may without loss of generality assume that jobs i and $2m + 1 - i$ are assigned to machine i for $1 \leqslant i \leqslant m$.

For each i, $1 \leqslant i \leqslant \lceil 2m/3 \rceil$, consider the problem instance given by $a_1 = \cdots = a_i = 1$ and $a_{i+1} = \cdots = a_n = (m - i)/(n - i)$. For this problem instance, it is easy to see that $\mathrm{OPT} < 1 + (m - i)/(n - i)$. In order for $1 + e_m$ to be an upper bound on the worst case performance ratio, we need

$$\frac{1 + (n_i - 1)((m - i)/(n - i))}{(1 + (m - i)/(n - i))} \leqslant 1 + e_m,$$

which is written as

$$n_i \leqslant 2 + e_m + e_m\left(\frac{n - i}{m - i}\right). \tag{3}$$

For $\lceil 2m/3 \rceil + 1 \leqslant i \leqslant m$ consider the problem instance given by $a_1 = \cdots = a_{2m+1-i} = 1$ and $a_{2m+2-i} = \cdots = a_n = (i - 1)/(n - (2m + 1 - i))$. For this problem instance, it is easy to see that $\mathrm{OPT} < 2 + ((i + 1)/(n - (2m + 1 - i)))$. In order for $1 + e_m$ to be an upper bound on the worst case performance ratio, we need

$$\frac{2 + (n_i - 2)((i - 1)/(n - (2m + 1 - i)))}{(2 + ((i - 1)/(n - (2m + 1 - i))))} \leqslant 1 + e_m,$$

which is rewritten as

$$n_i \leqslant 3 + e_m + 2e_m\left(\frac{n - (2m + 1 - i)}{i - 1}\right). \tag{4}$$

By (3) and (4), and letting $\theta = \lceil 2m/3 \rceil$,

$$n = \sum_{i=1}^{m} n_i \leqslant \sum_{i=1}^{\theta}\left(2 + e_m + e_m\left(\frac{n - i}{m - i}\right)\right) + \sum_{i=\theta+1}^{m}\left(3 + e_m + 2e_m\left(\frac{n - 2m - 1 + i}{i - 1}\right)\right). \tag{5}$$

Dividing (5) by n and letting n approach infinity yields

$$1 \leqslant e_m \sum_{i=1}^{\theta} \frac{1}{m - i} + 2e_m \sum_{i=\theta+1}^{m} \frac{1}{i - 1} = e_m \sum_{i=m-\theta}^{m-1} \frac{1}{i} + 2e_m \sum_{i=\theta}^{m-1} \frac{1}{i} \equiv e_m \phi(m). \tag{6}$$

Table 2

m	Lower bound
2	1.3333
3	1.4000
4	1.4375
5	1.3704
6	1.4539
7	1.4542
8	1.4485
9	1.4775
10	1.4776
11	1.4744
12	1.4891
13	1.4892
14	1.4872
15	1.4961
16	1.4961
17	1.4947
$\geqslant 18$	1.5000

To evaluate the sums in (6), let r and s be positive integers with $r \leqslant s$. Thus

$$\sum_{i=r}^{s} \frac{1}{i} < \int_{r-1/2}^{s+1/2} \frac{dx}{x} = \ln\left(\frac{s+1/2}{r-1/2}\right).$$

Applying this inequality to (6) yields

$$e_m > \frac{1}{\ln\left(\dfrac{m-1/2}{m-\lceil 2m/3\rceil - 1/2}\right) + 2\ln\left(\dfrac{m-1/2}{\lceil 2m/3\rceil - 1/2}\right)} \equiv \psi(m) \geqslant \phi^{-1}(m). \tag{7}$$

Straightforward calculations show that $\phi^{-1}(m) > \frac{1}{2}$ for $18 \leqslant m \leqslant 20$ and $\psi(m) > \frac{1}{2}$ for $21 \leqslant m \leqslant 23$. The reader may verify that for $m \geqslant 3$, $\psi(m+3) > \psi(m)$. Hence, for $m \geqslant 18$, the assumption that $1 + e_m < 3/2$ cannot hold. □

Previously, we have shown that 4/3 and 7/5 are tight lower bounds on $1 + e_2$ and $1 + e_3$ respectively, and that 23/16 is a lower bound on $1 + e_4$. For $m = 5$ to 17, the right-hand side of relation (7) is less than 1.5, and hence these right-hand-side values are lower bounds on $1 + e_m$ for these values of m. We may summarize our results in Table 2.

5. Future directions

We have described herein algorithms for parallel machine scheduling with ordinal data which are optimal with respect to worst case performance for two and three machines, near optimal for four machines, and whose worse case performance ratio is bounded by $1 + (m-1)/(m + \lceil m/2 \rceil)$ for all other values of m. However, for m greater than 4, it is probable that algorithms with much better worst case performance than that of P_m exist. The authors would propose as an open problem to find a "generic" algorithm which achieves the best possible worst case performance ratio for each m. Numerical experiments show that the bound of

$1 + (m - 1)/(m + \lceil m/2 \rceil)$ given by P_m is not achievable through procedure CREATE for $m \geqslant 8$. The authors believe that finding the "generic" algorithm above will require some new insights.

There are many other scheduling problems where ordinal, or some weaker variant of ordinal, algorithms could be developed, and we would propose that investigations to find good algorithms for these problems would be of interest to the scheduling community.

Acknowledgements

The authors wish to express their appreciation to Dr. Hans Kelleher of the Institut für Statistik, Ökonometrie und Operations Research, Universität Graz, Austria.

References

[1] A. Agnetis, "No-wait flow shop scheduling with large lot size", Rap. 16.89, Dipartimento di Informatica e Sistemistica, Universita Degli Studi di Roma "La Sapienza", Rome, Italy, 1989.
[2] W. Conway, W.L. Maxwell and L.W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, MA, 1967.
[3] R.L. Graham, "Bounds on multiprocessing timing anomalies", *SIAM J. Appl. Math.* **17**, 416–429 (1969).
[4] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, Toronto, 1976.
[5] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, "Sequencing and scheduling: algorithms and complexity", Centre for Mathematics and Computer Science Report BS-R8909, Stichting Mathematisch Centrum, Amsterdam, 1989.
[6] W.-P. Liu and J.B. Sidney, "Bin packing using semi-ordinal data", *Operations Research Letters* **19** (1996) to appear.
[7] W.-P. Liu and J.B. Sidney, "Ordinal algorithms for packing with target center of gravity", *Order*, to appear.
[8] W.-P. Liu, J.B. Sidney and A. van Vliet, "Ordinal algorithms for parallel machine scheduling", Working Paper 95-57, Faculty of Administration, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5, 1995.