A new algorithm for minimizing makespan on identical parallel machines

Abey Kuruvilla* and Giuseppe Paletta†

Abstract

A heuristic algorithm is developed for the classical multiprocessor scheduling problem in which independent jobs are nonpreemptively scheduled on identical parallel machines with the objective of minimizing the makespan. The algorithm uses iteratively the *LPT* procedure followed by the *MultiFit* procedure on different job and machine sets. The effectiveness of our algorithm is evaluated through computational comparisons with other methods from the literature for different classes of benchmark instances.

Keywords. Identical parallel machines, minimizing makespan, heuristic algorithm, empirical results.

1 Introduction

In the classical multiprocessor scheduling problem, a set $J = \{1, ..., j, ..., n\}$ of n simultaneously available independent jobs, with positive processing times $p_j > 0$, $j \in J$, is scheduled on a set $M = \{1, ..., i, ..., m\}$ of m identical parallel machines. Each machine can process at the most one job at a time, and each job must be processed without interruption by exactly one of the m machines. The paper considers the problem of finding the schedule that minimizes the maximum job completion time (makespan). This problem is denoted in the literature as $P \mid\mid C_{max}$, see Graham et al. [1979]. It is one on the most studied

*School of Business&Technology, University of Wisconsin-Parkside, USA. E-mail addresse: abeykuruvilla@gmail.com.

†Dipartimento di Economia e Statistica, Università della Calabria, 87036 Arcavacata di Rende (CS), Italy. E-mail addresse: g.paletta@unical.it.

problem in combinatorial optimization both for its theoretical interest and for its practical interest in many real world applications (see Blazewicz et al. [1988], Cheng and Sin [1990], and Lee et al. [1997]).

A feasible solution is represented by an m-partition $S = \{S_1, ..., S_i, ..., S_m\}$ of the set J, where each S_i represents the subset of jobs assigned to the machine i, $i \in M$. For each feasible solution S, the work-loads of the machines are represented by the m-set $C(S) = \{C(S_1), ..., C(S_i), ..., C(S_m)\}$, where $C(S_i) = \sum_{j \in S_i} p_j$ is the work-load of machine $i \in M$. In other words, $C(S_i)$ represents the completion time of the latest job assigned to the machine $i \in M$. For each schedule S, $C_{max}(S) = \max_{i \in M}\{C(S_i)\}$ denotes the makespan associated with the solution S.

$P \mid\mid C_{max}$ is strongly NP-Hard for an arbitrary $m \geq 2$ (see Garey and Johnson [1979]). It is supposed that $n > m \geq 2$ to avoid trivialities. Due to the NP-hardness of the problem many polynomial time algorithms, having a known worst-case performance ratio, have been developed to solve $P \mid\mid C_{max}$ (see Cheng and Sin [1990], Lawler et al. [1993], Hoogeveen et al. [1997], Chen et al. [1998], and Mokotoff [2001] for an overview). The most important constructive approaches to solve $P \mid\mid C_{max}$ are:

- *List Scheduling family* (LS) (Graham [1966] and [1969]): a list of all the jobs is made and then iteratively the first job is taken out of the list and assigned to the least loaded machine. The process continues until the list is empty. There are different kinds of LS algorithms, depending on how the list of jobs is made. The best known LS algorithm is the *Longest Processing Time* (LPT) that first sorts all jobs into a list in non-increasing order with respect to their processing times, and then it iteratively assigns the uppermost job from the list to the least loaded machine, until the list is exhausted. The LPT algorithm runs in $O(nlogn + nm)$-time, where the first term corresponds to ordering jobs, and the second to the assignment of each job to a machine. Its worst-case ratio is equal to $4/3 - 1/(3m)$, which is tight.

- *MultiFit* (MF) (Coffman et al. [1978]): a number of times a Bin Packing Problem (BPP) with an appropriate capacity is iteratively solved. BPP can be seen as a "dual" of $P \mid\mid C_{max}$ in which the jobs must be assigned to an unlimited number of identical machines, without exceeding a prefixed capacity (makespan) c, with the objective to minimize the number of machines used. The idea behind MF is to find (by binary search) the smallest prefixed machine capacity c such that no more than m machines need to accommodate all jobs (that are taken in non-increasing order of p_j) and each job is placed into the first machine into which it will fit. Coffman et al. [1978] show that MF runs in

$O(nlogn + tnlogm)$-time and its worst-case ratio is equal to $1.22 + 2^{-t}$, where t represents the number of times that a BPP is solved. It is recommended that t be at least 7. Friesen [1984] improves this bound from 1.22 to 1.2, and Yue [1990] improves it to 13/11, which is tight.

Lee and Massey [1988] subsequently propose the $COMBINE$ heuristic which utilizes the solution of LPT as the incumbent and then applies MF with fewer iterations. $COMBINE$ requires the same computational time and has the same worst-case ratio as MF. Gupta and Ruiz-Torres [2001] propose the $LISTFIT$ algorithm that is based on the combination of multiple list of jobs with the bin packing approach. $LISTFIT$ runs in $O(n^2logn + tn^2logm)$-time and its worst-case ratio performance is like MF.

In this paper, an $O(nlogn + mtnlogm)$ algorithm is designed to solve $P \mid\mid C_{max}$. It applies at most m times LPT approach followed by MF approach on different job and machine sets that are constructed by using the solution obtained at the iteration before.

The plan of the paper is the following. The algorithm is described in Section 1 and the results of a wide computational experimentation on a large number of benchmark instances are reported in Section 2. Finally, some concluding remarks are provided.

2 Algorithm

The algorithm, called *Different Job and Machine Sets* ($DJMS$), solves $P \mid\mid C_{max}$ by using LPT and MF approaches, many times in sequence, on different job and machine sets. At each iteration, it considers two sets of machine: M^a (active machine set) and M^c (closed machine set), and two job sets: J^a (active job set) and J^c (closed job set). The machine set M^c includes the machines already loaded, whereas the machine set M^a includes the machines that are available for a new schedule. The job set J^c includes the jobs that are assigned to the closed machine set, whereas the job set J^a includes the jobs that must be assigned to the active machine set. At the start, J^a and M^a are set to J and M respectively, whereas J^c and M^c are both empty. At each iteration, $P \mid\mid C_{max}$ is solved by using LPT and MF approaches on the instance defined by the job set J^a and the machine set M^a. Once a solution $S^a = \{S^a_h, \forall h \in M^a\}$ with work-loads of the machines $C(S^a) = \{C(S^a_h), \forall h \in M^a\}$ is obtained for the instance defined by M^a and J^a, then the least load machines, among those with work-load not less not less a lower bound, are considered closed machines, and the jobs assigned to

them are considered closed jobs. To select the closed machine and job sets, this algorithm uses the well-known lower bound

$$L_2 = \max\left\{\left\lceil \frac{1}{m}\sum_{j=1,\ldots,n} p_j \right\rceil, p_1, p_m + p_{m+1}\right\},$$

where the jobs are sorted so that $p_1 \geq ... \geq p_j \geq ... \geq p_n$ (see Dell'Amico and Martello [1995]).

At each iteration, the current solution for the instance defined by J and M is given by $S^a \cup S^c$, where $S^c = \{S^c_h, \forall h \in M^c\}$ is the solution related to job set J^c and machine set M^c. The incumbent makespan $C_{max}(S^i)$ is updated each time it is greater than the current makespan ($max\{C_{max}(S^a), C_{max}(S^c)\}$).

The algorithm ends either when all the machines are closed ($mac_c = m$, where mac_c is the number of closed machines) or when the makespan of the closed machines $C_{max}(S^c)$ is not less than the makespan of the active machines $C_{max}(S^a)$. Formally, the algorithm can be described as follows.

DJMS Algorithm

Step 0. Consider the set of jobs J sorted in non-increasing order with respect to the processing times, and the set of machines M. Set $mac_c = 0$, $J^a = J$, $M^a = M$, $J^c = M^c = \emptyset$, $C_{max}(S^c) = L_2$ (denote the makespan of the closed machines), and $C_{max}(S^i) = \infty$ (denote the makespan of the incumbent solution).

Step 1. Apply $LPT(J^a, M^a, S^a, C(S^a))$ algorithm to solve $P \mid\mid C_{max}$ on job set J^a and machine set M^a.

Step 2. Apply $MF(J^a, M^a, S^a, C(S^a))$, by starting from solution given by LPT, to solve $P \mid\mid C_{max}$ on job set J^a and machine set M^a. Compute $C_{max}(S^a) = max_{h \in M^a}\{C(S^a_h)\}$.

Step 3. If $C_{max}(S^i) > max\{C_{max}(S^a), C_{max}(S^c)\}$ then set $S^i_h = S^a_h$, $h \in M^a$, and $S^i_h = S^c_h$, $h \in M^c$.

Step 4. For each $h \in M^a$ so that $C(S^a_h) = min_{l \in M^a}\{C(S^a_l) : C(S^a_l) \geq L_2\}$ set $S^c_h = S^a_h$, $C(S^c_h) = C(S^a_h)$, $mac_c = mac_c + 1$, $J^a = J^a \backslash S^c_h$, $M^a = M^a \backslash \{h\}$, $J^c = J^c \cup S^c_h$, $M^c = M^c \cup \{h\}$, and $C_{max}(S^c) = max_{l \in M^c}\{C(S^c_l)\}$.

Step 5. If $C_{max}(S^a) > C_{max}(S^c)$ and $mac_c < m$ then return to Step 1.

Step 6. Return S^i, $C(S^i)$, and $C_{max}(S^i)$.

By making the paper comprehensive, the LPT and MF procedures are reported in the following.

$LPT(J^a, M^a, S^a, C(S^a))$ *Procedure*

Step 0. Consider the set of jobs J^a sorted in non-increasing order with respect to the processing times, and the set of machines M^a. For each machine $h \in M^a$ set $S_h^a = \emptyset$ and $C(S_h^a) = 0$.

Step 1. For each job $j \in J^a$ select the machine $h \in M^a$ such that $C(S_h^a)$ is as small as possible, and assign j to the machine h by setting $S_h^a = S_h^a \cup \{j\}$ and $C(S_h^a) = C(S_h^a) + p_j$.

Step 2. Return S^a and $C(S^a)$.

$MF(J^a, M^a, S^a, C(S^a))$ *Procedure*

Step 0. Consider the current feasible solution S^a and $C(S^a)$ obtained by using $LPT(J^a, M^a, S^a, C(S^a))$, the set of jobs J^a sorted in non-increasing order with respect to the processing times, and the set of machines M^a. Set $UB = C_{max}(S^a)$ (upper bound to the recommended machine work-load), $LB = L_2$ (lower bound to the recommended machine work-load) and $C = (UB + LB)/2$ (recommended machine work-load). Let t the maximal number of iterations without improvement. Set $iter = 0$ (number of iterations without improvement).

Step 1. For each machine $h \in M^a$ set $S'_h = \emptyset$ and $C(S'_h) = 0$.

Step 2. For each job $j \in J^a$ find the first machine $h \in M^a$ such that $C(S'_h) + p_j \leq C$, and assign j to the machine h by setting $S'_h = S'_h \cup \{j\}$ and $C(S'_h) = C(S'_h) + p_j$. If such machine does not exist then go to Step 4.

Step 3. Set $S^a = S'$, $C(S^a) = C(S')$. If $C_{max}(S^a) = L_2$ then go to Step 5. Moreover set $UB = C_{max}(\bar{S}^a)$, $C = (UB + LB)/2$, $iter = 0$ (an improvement has been obtained), and return to Step 1.

Step 4. If $iter < t$ then set $iter = iter + 1$, $LB = C$, $C = (UB + LB)/2$ (no improvement has been obtained) and return to Step 1.

Step 5. Return S^a and $C(S^a)$.

With regard to the efficiency, $DJMS$ runs in $O(nlogn+mtnlogm)$, where the first term corresponds to ordering jobs (just a time the jobs must be ordered), and the second term corresponds to solving m times a BPP by using MF.

3 Computational results

$DJMS$ has been implemented in Fortran on an Intel Core i5 processor (310 GHz, 6 GbRAM) under Microsoft Windows 7 Professional operative system, and compared with the LPT, MF, $COMBINE$, and $LISTFIT$ algorithms. Four families (E1, E2, E3, and E4) of instances taken from the literature (see Kedia [1971], Lee and Massey [1988], and Gupta and Ruiz-Torres [2001]) have been used for the comparisons.

Each family of instances considers three parameters: the number of machines (m), the number of jobs (n), and the interval (U) used by a uniform distribution to generate randomly the integer processing times. Table 1 presents a summary of all four families with respect to the different combinations of m, n and U. For each choice of m, n, and U, 100 instances have been taken. This gives a total of 9800 instances.

The performances of the heuristics have been evaluated with respect to the lower bound L_2 for the families E1, E2 and E3. Family E4 has been used to evaluate the performances of the heuristics with respect to the optimum solutions.

Table 1. Summary of the experimental frameworks.

	m	n	U
E1	3, 4, 5	*2m, 3m, 5m*	[1,20], [20,50]
E2	2, 3	10, 30, 50, 100	[100,800]
	4,6,8,10	30, 50, 100	[100,800]
E3	3,5,8,10	*3m+1,3m+2,4m+1,4m+2,5m+1,5m+2*	[1,100], [100,200]
E4	2	10	[1,20],[20,50],[1,100],[50,100],[100,200],[100,800]
	3	9	[1,20],[20,50],[1,100],[50,100],[100,200],[100,800]

Computational results are presented in Tables 2-5. Columns "LPT", "MF", "COMBINE", "LISTFIT", and "DJMS" summarize the results obtained by using the respective algorithms. In Tables 2, 3, and 4, which refer to E1, E2, and E3 families respectively, the sub-columns "gap" report the average relative errors with respect to the lower bound L_2. Whereas, in Table 5, which refer to E4 family, the sub-columns "gap" report the average relative errors with respect to the optimal solution. The relative errors are averaged for a group of 100 instances. The Tables also report the number of instances (out of 100) that have been solved to optimality (sub-columns labeled "o"). This number, for the families E1, E2, and E3, was determined by a comparison to the lower bound value L_2, whereas for the family E4 it was determined by a comparison to the optimum value. In Tables 2-5, the last rows report the overall average percentage errors on all instances (columns "gap"), and the

total number of instances that have been solved to optimality (columns "o"). Finally, computational times are not reported given these were fractions of a second for all the algorithms.

Table 2. Computational results for E1

			LPT		MF		COMBINE		LISTFIT		DJMS	
n	m	U	gap	o	gap	o	gap	o	gap	o	gap	o
6	3	[1,20]	2.644E-02	62	2.478E-02	63	2.478E-02	63	2.478E-02	63	2.478E-02	63
9	3	[1,20]	3.074E-02	35	1.511E-02	65	1.147E-02	68	7.077E-03	79	7.099E-03	79
15	3	[1,20]	1.152E-02	56	6.169E-03	71	3.438E-03	83	3.390E-04	99	1.852E-04	99
6	3	[20,50]	2.827E-02	45	2.741E-02	45	2.741E-02	45	2.741E-02	45	2.741E-02	45
9	3	[20,50]	2.846E-02	11	4.783E-02	2	2.221E-02	13	1.845E-02	18	1.756E-02	21
15	3	[20,50]	7.128E-03	26	1.965E-02	14	5.887E-03	35	4.047E-03	44	3.604E-03	48
8	4	[1,20]	2.276E-02	66	1.990E-02	68	1.990E-02	68	1.990E-02	68	1.990E-02	68
12	4	[1,20]	2.892E-02	34	1.433E-02	60	1.199E-02	64	7.689E-03	76	7.774E-03	76
20	4	[1,20]	1.038E-02	56	4.523E-03	76	2.958E-03	84	3.154E-04	98	4.921E-04	97
8	4	[20,50]	3.302E-02	40	3.204E-02	40	3.204E-02	40	3.204E-02	40	3.204E-02	40
12	4	[20,50]	2.331E-02	10	5.166E-02	1	2.049E-02	11	1.924E-02	11	1.701E-02	14
20	4	[20,50]	7.440E-03	24	2.311E-02	10	6.583E-03	29	5.438E-03	32	3.796E-03	45
10	5	[1,20]	3.723E-02	48	3.168E-02	52	3.168E-02	52	3.118E-02	53	3.118E-02	53
15	5	[1,20]	3.649E-02	23	1.362E-02	55	1.248E-02	59	7.668E-03	74	6.080E-03	79
25	5	[1,20]	1.231E-02	47	3.167E-03	82	2.479E-03	86	8.715E-04	95	0.000E+00	100
10	5	[20,50]	3.319E-02	27	3.142E-02	27	3.142E-02	27	3.142E-02	27	3.142E-02	27
15	5	[20,50]	2.366E-02	9	5.278E-02	0	2.095E-02	9	2.024E-02	9	1.673E-02	11
25	5	[20,50]	7.609E-03	17	2.723E-02	0	7.161E-03	17	6.554E-03	18	4.943E-03	31
			2.272E-02	636	2.480E-02	731	1.641E-02	853	1.470E-02	949	1.400E-02	996

The results for $E1$ (Table 2) show that $DJMS$ and $LISTFIT$ algorithms always outperform all other approaches in terms of the considered measures of accuracy (number of optimal solutions and average percentage errors). By comparing $DJMS$ and $LISTFIT$, Table 2 shows that $DJMS$ finds the optimal solution in 996 out of 1800 test problems, whereas the $LISTFIT$ algorithm finds the optimum in 949. Moreover, the overall average percentage error of the new algorithm is practically identical to that of $LISTFIT$, but it is smaller than those of other algorithms. With respect to the average percentage errors, $DJMS$ outperforms $LISTFIT$ in 9 out of 18 experimental points, $LISTFIT$ outperforms $DJMS$ in 3 points, while in the other 6 both methods perform identically. With respect to the number of optimal solutions, $DJMS$ outperforms $LISTFIT$ in 8 out of 18 experimental points, $LISTFIT$ outperforms $DJMS$ in 1 points, while in the other 9 both methods perform identically.

The results for experiment E2 (Table 3) show that $LISTFIT$ and $DJMS$ always outperform all other algorithms with respect to the number of optimal solutions and the average percentage errors. $DJMS$ finds the optimal

solution in 122 out of 2000 test problems, whereas the $LISTFIT$ algorithm finds the optimum in 373. Moreover, the total average percentage error of the new algorithm is smaller than those of $LISTFIT$. With low values of m ($m = 2$, 3, 4, and 6), $LISTFIT$ outperforms $DJMS$ in term of the considered measures of accuracy. Whereas, with high values of m ($m = 8$, and 10), $DJMS$ outperforms $LISTFIT$ with respect the average relative errors. In the last case, $DJMS$ and $LISTFIT$ do not find optimal solutions.

Table 3. Computational results for E2

			LPT		MF		COMBINE		LISTFIT		DJMS	
n	m	U	gap	o	gap	o	gap	o	gap	o	gap	o
10	2	[100,800]	1.206E-02	2	1.244E-02	4	6.750E-03	6	2.243E-03	11	2.826E-03	12
30	2	[100,800]	1.785E-03	3	2.646E-03	4	1.085E-03	7	6.339E-05	63	5.644E-04	12
50	2	[100,800]	5.529E-04	9	9.735E-04	5	2.990E-04	19	6.277E-06	93	1.876E-04	26
100	2	[100,800]	1.100E-04	26	7.831E-04	3	6.597E-05	43	0.000E+00	100	3.930E-05	52
10	3	[100,800]	4.730E-02	0	2.450E-02	0	2.239E-02	0	1.545E-02	0	1.376E-02	0
30	3	[100,800]	4.376E-03	1	4.809E-03	0	2.926E-03	1	8.030E-04	7	1.658E-03	3
50	3	[100,800]	5.440E-03	0	2.022E-03	0	1.869E-03	3	2.081E-04	23	9.296E-04	4
100	3	[100,800]	4.275E-03	0	1.102E-03	0	7.092E-04	6	2.587E-05	65	3.420E-04	10
30	4	[100,800]	1.862E-02	0	7.048E-03	0	6.920E-03	0	3.022E-03	0	3.821E-03	0
50	4	[100,800]	9.889E-03	0	3.576E-03	0	3.389E-03	0	9.428E-04	0	1.629E-03	0
100	4	[100,800]	7.253E-04	0	1.493E-03	0	4.372E-04	2	1.162E-04	11	3.104E-04	3
30	6	[100,800]	1.680E-02	0	1.263E-02	0	1.047E-02	0	7.717E-03	0	7.176E-03	0
50	6	[100,800]	1.834E-02	0	5.392E-03	0	5.240E-03	0	3.257E-03	0	3.411E-03	0
100	6	[100,800]	5.510E-03	0	2.035E-03	0	1.728E-03	0	8.360E-04	0	1.118E-03	0
30	8	[100,800]	3.961E-02	0	1.642E-02	0	1.602E-02	0	1.413E-02	0	1.177E-02	0
50	8	[100,800]	2.460E-02	0	7.593E-03	0	7.494E-03	0	5.876E-03	0	5.212E-03	0
100	8	[100,800]	1.015E-02	0	3.199E-03	0	2.864E-03	0	1.937E-03	0	1.703E-03	0
30	10	[100,800]	4.565E-02	0	2.286E-02	0	2.085E-02	0	1.930E-02	0	1.719E-02	0
50	10	[100,800]	1.675E-02	0	1.072E-02	0	9.302E-03	0	8.106E-03	0	6.912E-03	0
100	10	[100,800]	3.904E-03	0	3.822E-03	0	2.629E-03	0	2.180E-03	0	1.885E-03	0
			1.432E-02	41	7.303E-03	16	6.172E-03	87	4.311E-03	373	4.122E-03	122

Table 4 presents the results for E3. As in the previous two experiment sets, $DJMS$ and $LISTFIT$ always outperform all other approaches. $DJMS$ outperforms $LISTFIT$ in 19 out of 48 experimental points with respect to the number of optimal solutions found, and in 39 out of 48 experimental points with respect to the average percentage errors. $LISTFIT$ outperforms $DJMS$ in 11 out of 48 experimental points with respect to the number of optimal solutions found, and in 9 of the 48 experimental points with respect to the average percentage error. The performance of both $LISTFIT$ and $DJMS$ improves as m decreases, as n increases, and as the interval U changes from $[100, 200]$ to $[1, 100]$.

Table 4. Computational results for E3

			LPT		MF		COMBINE		LISTFIT		DJMS	
n	m	U	gap	o	gap	o	gap	o	gap	o	gap	o
10	3	[1,100]	3.436E-02	3	2.413E-02	4	1.945E-02	6	1.404E-02	7	1.145E-02	13
11	3	[1,100]	3.042E-02	0	1.880E-02	7	1.426E-02	7	9.678E-03	11	8.423E-03	19
13	3	[1,100]	2.261E-02	4	1.190E-02	12	9.459E-03	16	5.190E-03	30	4.301E-03	36
14	3	[1,100]	1.913E-02	2	1.005E-02	15	8.484E-03	16	3.829E-03	39	3.965E-03	36
16	3	[1,100]	1.559E-02	4	8.278E-03	15	7.193E-03	16	3.105E-03	38	1.967E-03	54
17	3	[1,100]	1.344E-02	6	7.252E-03	21	5.634E-03	23	2.388E-03	42	1.871E-03	54
10	3	[100,200]	1.288E-01	0	2.591E-02	0	2.591E-02	0	1.805E-02	1	1.820E-02	0
11	3	[100,200]	6.849E-02	0	5.887E-02	0	5.500E-02	0	1.842E-02	1	2.739E-02	0
13	3	[100,200]	1.017E-01	0	3.366E-02	0	3.366E-02	0	1.305E-02	2	9.205E-03	4
14	3	[100,200]	5.192E-02	0	3.756E-02	0	3.663E-02	0	1.065E-02	3	1.603E-02	0
16	3	[100,200]	8.171E-02	0	4.733E-02	0	4.731E-02	0	1.259E-02	1	1.326E-02	0
17	3	[100,200]	4.198E-02	0	2.976E-02	0	2.847E-02	0	8.518E-03	2	1.156E-02	1
16	5	[1,100]	3.698E-02	4	1.774E-02	11	1.540E-02	12	1.249E-02	16	1.072E-02	17
17	5	[1,100]	3.411E-02	1	1.520E-02	9	1.380E-02	9	1.091E-02	13	9.318E-03	19
21	5	[1,100]	2.663E-02	2	1.057E-02	7	8.596E-03	9	6.889E-03	12	5.290E-03	20
22	5	[1,100]	2.331E-02	1	9.373E-03	3	8.351E-03	4	5.966E-03	9	4.668E-03	17
26	5	[1,100]	1.484E-02	2	6.962E-03	12	6.322E-03	14	4.576E-03	18	3.353E-03	32
27	5	[1,100]	1.341E-02	3	6.286E-03	18	5.833E-03	20	3.740E-03	27	2.860E-03	39
16	5	[100,200]	1.565E-01	0	3.433E-02	0	3.433E-02	0	3.017E-02	0	1.791E-02	0
17	5	[100,200]	1.183E-01	0	4.407E-02	0	4.403E-02	0	3.181E-02	0	2.447E-02	0
21	5	[100,200]	1.224E-01	0	5.204E-02	0	5.204E-02	0	3.271E-02	0	2.576E-02	0
22	5	[100,200]	9.212E-02	0	3.960E-02	0	3.960E-02	0	2.805E-02	0	1.845E-02	0
26	5	[100,200]	9.796E-02	0	3.940E-02	0	3.940E-02	0	2.430E-02	0	2.626E-02	0
27	5	[100,200]	7.523E-02	0	2.895E-02	0	2.884E-02	0	2.288E-02	0	1.600E-02	0
25	8	[1,100]	4.740E-02	1	1.318E-02	8	1.285E-02	8	1.135E-02	9	1.022E-02	10
26	8	[1,100]	4.435E-02	1	1.411E-02	10	1.383E-02	10	1.177E-02	11	1.029E-02	12
33	8	[1,100]	2.468E-02	0	7.909E-03	7	7.770E-03	7	5.552E-03	17	5.019E-03	20
34	8	[1,100]	2.086E-02	1	7.613E-03	12	7.336E-03	12	5.615E-03	19	4.670E-03	22
41	8	[1,100]	1.764E-02	2	4.721E-03	15	4.531E-03	16	3.329E-03	26	2.798E-03	35
42	8	[1,100]	1.509E-02	1	5.017E-03	17	4.833E-03	17	3.024E-03	36	2.935E-03	33
25	8	[100,200]	1.764E-01	0	4.514E-02	0	4.514E-02	0	4.166E-02	0	2.115E-02	0
26	8	[100,200]	1.497E-01	0	4.234E-02	0	4.234E-02	0	3.841E-02	0	2.662E-02	0
33	8	[100,200]	1.327E-01	0	5.365E-02	0	5.365E-02	0	4.608E-02	0	2.210E-02	0
34	8	[100,200]	1.152E-01	0	4.430E-02	0	4.430E-02	0	3.969E-02	0	3.098E-02	0
41	8	[100,200]	1.073E-01	0	3.872E-02	0	3.872E-02	0	3.346E-02	0	1.689E-02	0
42	8	[100,200]	9.295E-02	0	4.110E-02	0	4.110E-02	0	3.365E-02	0	2.079E-02	0
31	10	[1,100]	5.196E-02	1	1.308E-02	10	1.297E-02	10	1.106E-02	13	9.653E-03	14
32	10	[1,100]	4.028E-02	1	1.235E-02	6	1.184E-02	6	1.024E-02	7	8.709E-03	9
41	10	[1,100]	2.615E-02	0	6.000E-03	12	6.000E-03	12	4.624E-03	20	4.575E-03	17
42	10	[1,100]	2.504E-02	1	6.041E-03	14	5.946E-03	14	4.882E-03	21	4.570E-03	19
51	10	[1,100]	1.926E-02	0	4.069E-03	19	4.069E-03	19	2.748E-03	35	2.868E-03	31
52	10	[1,100]	1.572E-02	0	3.554E-03	22	3.554E-03	22	2.524E-03	38	2.574E-03	37
31	10	[100,200]	1.772E-01	0	4.672E-02	0	4.672E-02	0	4.467E-02	0	2.421E-02	0
32	10	[100,200]	1.620E-01	0	4.224E-02	0	4.224E-02	0	3.982E-02	0	2.957E-02	0
41	10	[100,200]	1.374E-01	0	5.398E-02	0	5.398E-02	0	5.015E-02	0	2.209E-02	0
42	10	[100,200]	1.220E-01	0	4.885E-02	0	4.885E-02	0	4.641E-02	0	3.334E-02	0
51	10	[100,200]	1.118E-01	0	4.051E-02	0	4.051E-02	0	3.680E-02	0	1.680E-02	0
52	10	[100,200]	9.900E-02	0	4.188E-02	0	4.188E-02	0	3.674E-02	0	2.447E-02	0
			6.988E-02	41	2.615E-02	286	2.548E-02	305	1.871E-02	524	1.355E-02	620

Table 5 presents the results for E4 with respect to the optimal solutions and shows that $DJMS$ and $LISTFIT$ always outperform all other approaches. The overall average relative error ($2.480E-03$) and the total number of optimal solutions (780 out of 1200) of $DJMS$ are practically equal to those of $LISTFIT$ ($2.499E-03$)and (787). As regards the experimental parameters, only m plays a role in heuristic performances. When $m = 2$, $LISTFIT$ dominates all other approaches, while $DJMS$ dominates all other approaches when $m = 3$.

Table 5. Computational results for E4

			LPT		MF		COMBINE		LISTFIT		DJMS	
n	m	U	gap	o	gap	o	gap	o	gap	o	gap	o
10	2	[1,20]	8.866E-03	61	5.194E-03	74	2.271E-03	88	0.000E+00	100	0.000E+00	100
10	2	[20,50]	6.400E-03	38	1.795E-02	27	4.450E-03	56	1.562E-03	77	2.374E-03	68
10	2	[1,100]	1.037E-02	29	9.278E-03	26	5.386E-03	42	9.601E-04	77	1.598E-03	73
10	2	[50,100]	5.189E-03	25	4.053E-02	3	5.028E-03	27	1.695E-03	59	3.096E-03	43
10	2	[100,200]	4.822E-03	21	4.221E-02	4	4.338E-03	24	1.813E-03	42	2.766E-03	30
10	2	[100,800]	1.144E-02	4	1.086E-02	8	5.514E-03	12	1.735E-03	36	1.939E-03	32
9	3	[1,20]	2.315E-02	50	1.104E-02	69	6.505E-03	80	1.629E-03	95	1.341E-03	95
9	3	[20,50]	1.208E-02	43	3.795E-02	14	8.368E-03	51	5.133E-03	61	4.697E-03	63
9	3	[1,100]	2.519E-02	26	1.011E-02	62	6.970E-03	70	1.962E-03	83	1.508E-03	89
9	3	[50,100]	6.963E-03	45	5.276E-02	4	6.177E-03	49	3.670E-03	59	3.030E-03	64
9	3	[100,200]	9.396E-03	35	5.439E-02	2	8.350E-03	37	4.946E-03	47	3.407E-03	60
9	3	[100,800]	2.335E-02	12	1.674E-02	28	1.013E-02	36	4.889E-03	51	4.010E-03	63
			1.227E-02	389	2.575E-02	321	6.124E-03	572	2.499E-03	787	2.480E-03	780

The four experiments demonstrate that $LISTFIT$ and $DJMS$ always outperform the other evaluated heuristics on all four experiments. The overall average percentage error of $DJMS$ is smaller than those of other algorithms but $LISTFIT$ solves to optimality more instances than other algorithms.

4 Conclusions

A new algorithm for the parallel machine scheduling problem with minimum makespan objective has been proposed. Its effectiveness has been evaluated through a wide computational investigation on different classes of benchmark instances from the literature. The computational results show that $DJMS$ quickly solves all the instances, obtains low relative errors and solves to optimality 26% of instances. Moreover comparisons with some of the best constructive heuristics (LPT algorithm of Graham [1966], MF of Coffman et

al. [1978], $COMBINE$ of Lee and Massey [1988], and $LISTFIT$ of Gupta and Ruiz-Torres [2001]) show that the overall average percentage error of $DJMS$ is smaller than those of other algorithms but $LISTFIT$ solves to optimality more instances (27%) than other algorithms.

References

Blazewicz, J., Finke, G., Haupt, R., Schmidt, G.: New trends in machine scheduling. European Journal of Operational Research 37, 303-317 (1988).

Chen, B., Potts, C.N., Woeginger, G.J.: A review of machine scheduling: complexity, algorithms and approximability. In: Du, D.Z., Pardalos, P. (eds.) Handbook of Combinatorial Optimization, vol. 3, 21-169. Kluwer Academic, Dordrecht (1998).

Cheng, T.C.E., Sin, C.C.S.: A state-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research 47, 271-292 (1990).

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

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

Friesen, D.K.: Tighter bounds for the MultiFit processor scheduling algorithm. SIAM J. Comput. 13, 170-181 (1984).

Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell Syst. Tech. J. 45, 1563-1581 (1966).

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

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

Gupta, J.N.D., Ruiz-Torres, A.J.: LISTFIT heuristic for minimizing makespan on identical parallel machines. Production Planning and Control 12, 28-36 (2001).

Hoogeveen, A., Lenstra, J.K., Van de Velde, S.L.: Sequencing and

Scheduling. In: Dell'Amico, M., Maffioli, F., Martello, S. (eds.) Annotated Bibliographies in Combinatorial Optimization, pp. 181-197. Wiley (1997).

Kedia, S. K.: A job scheduling problem with parallel processors. Unpublished Report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI (1971).

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: algorithms and complexity. In: Graves, S.C., Rinnooy Kan, A.H.G., Zipkin, P.H. (eds.) Handbooks in Operations Research and Management Science 4, pp. 445-522. Elsevier (1993).

Lee, C. Y., Massey, J.D.: Multiprocessor scheduling: combining LPT and MULTIFIT. Discrete Applied Mathematics 20, 233-242 (1988).

Lee, C. Y., Lei, L., and Pinedo, M.: Current trends in deterministic scheduling. Annals of Operations Research 70, 1-41 (1997).

Mokotoff, E.: Parallel machine scheduling problem: A survey. Asia Pac. J. Oper. Res. 18, 193-242 (2001).

Yue, M.: On the exact upper bound for the MultiFit processor scheduling algorithm. Ann. Oper. Res. 24, 233-259 (1990).