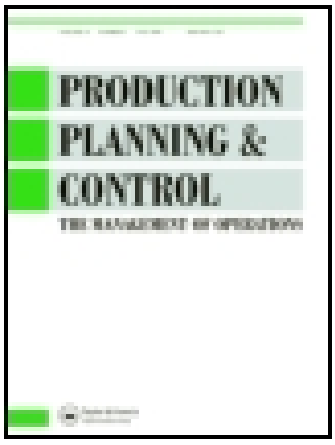


Production Planning & Control: The Management of Operations
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tppc20

A LISTFIT heuristic for minimizing makespan on identical parallel machines

J. N. D. Gupta & A. J. Ruiz-Torres

Published online: 15 Nov 2010.

To cite this article: J. N. D. Gupta & A. J. Ruiz-Torres (2001) A LISTFIT heuristic for minimizing makespan on identical parallel machines, Production Planning & Control: The Management of Operations, 12:1, 28-36, DOI: 10.1080/09537280150203951

To link to this article: http://dx.doi.org/10.1080/09537280150203951

PRODUCTION PLANNING & CONTROL, 2001, VOL. 12, NO. 1, 28–36

A LISTFIT heuristic for minimizing makespan on identical parallel machines

J. N. D. GUPTA and J. RUIZ-TORRES

Keywords parallel machine scheduling, minimizing makespan, listfit heuristic, empirical results.

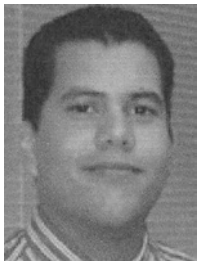
Abstract. This paper presents a new heuristic for minimizing makespan on identical parallel machines. The new heuristic, called LISTFIT, is based on bin-packing and list scheduling, two methods commonly used to solve this problem. Its worst case performance bound is no worse than that of the commonly used MULTIFIT heuristic. However, the average performance of the proposed listfit heuristic is considerably better than that of the multifit heuristic. The proposed heuristic may be of particular relevance to practitioners who are interested in solving this or related problems with real world constraints of computational time and effectiveness.

1. Introduction

Scheduling involves the allocation of machines over time to process jobs, where a 'job' consists of one or more activities, and a 'machine' is a resource that can perform at most one activity at a time. Multiple-machine scheduling problems require the construction of machine processing schedules for a given set of jobs to optimize a given measure of performance. Real life examples of scheduling problems involving multiple parallel machines where each job consists of only one activity include the scheduling of computer systems on one hand and the scheduling of bottleneck workstations involving multiple machines on the other (Pinedo and

Authors: J. N. D. Gupta, Department of Management, Ball State University, Muncie. IN 47306, USA, and A. J. Ruiz-Torres, College of Business Administration, University of Texas at El Paso, Texas 79968-0544, USA.

JATINDER N. D. GUPTA is Professor of Management, Information and Communication Sciences, and Industry and Technology at the Ball State University, Muncie, Indiana, USA. He holds a PhD in Industrial Engineering (with specialization in Production Management and Information Systems) from Texas Tech University. Co-author of a textbook in Operations Research, Dr Gupta serves on the editorial boards of several national and international journals. Recipient of an Outstanding Researcher award from Ball State University, he has published numerous research and technical papers in such journals as *Mathematics of Operations Research, Production and Operations Management, International Journal of Information Management, Annals of Operations Research, Journal of Management Information Systems, Operations Research, IIE Transactions, Naval Research Logistics, European Journal of Operational Research*, etc. His current research interests include information technology, scheduling, planning and control, organizational learning and effectiveness, systems education, and knowledge management. He is a member of several academic and professional societies including the Production and Operations Management Society (POMS), the Decision Sciences Institute (DSI), and Information Resources Management Association (IRMA).



ALEX J. RUIZ-TORRES is an Assistant Professor of Information and Decision Sciences, College of Business Administration, University of Texas at El Paso, El Paso, Texas, USA. Dr Ruiz-Torres obtained industrial engineering degrees from The Georgia Institute of Technology (BIE), Stanford University (MSIE) and The Pennsylvania State University (PhD). He is an author and co-author of several papers on production planning, logistics, simulation, and space systems. His articles have appeared in *Computers and Industrial Engineering, International Journal of Physical Distribution & Logistics Management*, and *European Journal of Operations Research*. He was awarded a NASA/ASEE Summer Fellowship position at the Kennedy Space Center during the summers of 1998 and 1999. He is member of the Production and Operations Management Society (POMS), The Institute for Operations Research, and the Management Science (INFORMS), and the Decision Sciences Institute (DSI).

Production Planning & Control ISSN 0953–7287 print/ISSN 1366–5871 online
http://www.tandf.co.uk/journals

Chao 1999). The parallel-machine scheduling problem is also an important generalization of the single-machine problem and can be a subproblem in many complex multiple-machine problems (Hax and Candea 1984, Pinedo and Chao 1999). The parallel machines may be identical as each machine requires the same amount of time to process a given job. If parallel machines have different ages or technology, these machines may require different amounts of time to process a given job.

A common measure of performance used in scheduling theory related to shop efficiency and utilization is the maximum completion time, or 'makespan'. This is defined as the elapsed time in which a given set of jobs completes processing in the shop. In practical terms, a schedule that minimizes the makespan also minimizes the time the shop is operating, which could relate to minimizing support cost and maximizing the use of resources. The minimization of the makespan could also have practical use in the scheduling of batches where a 'batch' is defined as a set of jobs. Creating a schedule that minimizes the makespan for the set of jobs in a batch will get the next batch started sooner, resulting in higher efficiency and resource utilization. In the parallel-machine scheduling problem, minimization of makespan also identifies a bottleneck machine requiring maximum attention of the shop management.

This paper considers the identical parallel-machine scheduling problem to minimize makespan and proposes a new heuristic that is based on bin-packing and list scheduling; two methods commonly used to solve this problem. The heuristic, called LISTFIT, has a worst case performance bound that is equal to the commonly used MULTIFIT or COMBINE heuristic, but has an average performance that is considerably better.

The rest of this paper is organized as follows. Problem formulation and available solution algorithms are briefly reviewed in section 2. Section 3 briefly describes three existing and commonly used heuristics: LPT, MULTIFIT and COMBINE . The proposed LISTFIT heuristic is described in section 4. Section 5 describes the experimental framework and presents the analysis of results. Finally, section 6 concludes the paper with some fruitful directions for future work.

2. Problem formulation and solution algorithms

Specifically, this paper considers the following scheduling problem: a set $N = \{1, 2, \ldots, n\}$ of n simultaneously available jobs is to be processed by m identical parallel machines. Each job requires one operation which can be processed by either one of the m machines. The processing time of each job $i \in N$ is p_i and includes any setup time required. A job once started is processed to completion without interruption. There is no precedence relationship between the jobs and each machine can process only one job at a time. For each machine $j \leq m$, let S_j be the subset of jobs assigned to machine j with completion time $C(S_j) = \sum_{i \in S_j} p_i$. The makespan, $C_{\max}(S)$, of this schedule is calculated as follows:

$$C_{\max}(S) = \max_{1 \leq j \leq m} C(S_j) = \max_{1 \leq j \leq m} \left\{ \sum_{i \in S_j} p_i \right\}. \qquad (1)$$

Then, the problem considered in this paper is one of finding the subsets $\{S_1, S_2, \ldots, S_m\}$ of the given n jobs such that the makespan, $C_{\max}(S)$ given by equation (1) is minimum.

Following the standard three-field notation of scheduling problems, the above identical parallel-machine problem to minimize makespan is designated as a $P\|C_{\max}$ problem where P designates the identical parallel machines and $C_{\max}$ denotes the maximum completion time or makespan. The $P\|C_{\max}$ problem has been shown to be NP-hard in the strong sense for an arbitrary number of machines m (Garey and Johnson 1979).

In view of the NP-hard nature of the above $P\|C_{\max}$ problem, several polynomial time algorithms have been proposed for its solution. The well-known LPT rule (Graham 1969) is part of a family of algorithms known as list-scheduling algorithms which iteratively assigns a list of jobs to the least loaded machine. The LPT rule has received extensive attention and has been shown to perform very well for the makespan single criteria problem (Kedia 1971, Blazewicz *et al.* 1996). The MULTIFIT heuristic proposed by Coffman *et al.* (1978) utilizes the close relation between the bin-packing problem and the maximum completion time problem. Although MULTIFIT is not guaranteed to perform better than LPT, it has been shown to have a worst case performance bound that is better than LPT (Friesen 1984, Yue 1990). Lee and Massey (1988) propose the COMBINE heuristic which utilizes the LPT result as an initial solution for the MULTIFIT algorithm so the error bound is never worse than MULTIFIT, and a performance as good as the best of either LPT or MULTIFIT. Using pairwise interchanges, Riera *et al.* (1996) proposed two improvements of the LPT (Longest Processing Time) schedule and empirically demonstrated that their algorithms perform similar to the MULTIFIT algorithm, but require less CPU time than MULTIFIT.

3. Three popular heuristic algorithms

In view of the NP-hard nature of the $P\|C_{\max}$ problem, several polynomially bounded heuristic algorithms have been proposed to find an approximate solution to the

problem. Three of the most popular heuristics are Graham's (1969) LPT algorithm, the MULTIFIT heuristic algorithm developed by Coffman *et al.* (1978), and the COMBINE heuristic developed by Lee and Massey (1988). Each of these is described below.

The LPT (longest processing time) algorithm assigns an available job with maximum processing time to the first machine. The steps of this algorithm are as follows.

Algorithm LPT: Longest processing time procedure

Input : n, m, p_i for $i = 1, \ldots, n$.

Step 1. Let $\sigma = (\sigma(1), \ldots, \sigma(n))$ be the job ordering such that $p_{\sigma(1)} \geq p_{\sigma(2)} \geq \ldots \geq p_{\sigma(n)}$. Set $T_1 = \cdots = T_m = 0$ and set $k = 1$. Further, set $S_1 = \cdots = S_m = \emptyset$.

Step 2. Select machine M_h so that T_h is as small as possible. Schedule job $\sigma(k)$ on machine M_h, set $T_h = T_h + p_{\sigma(j)}$ and $S_h = (S_h\sigma(k))$.

Step 3. If $k < n$, then set $k = k + 1$ and return to step 2. Otherwise, stop. The schedule where jobs in S_h are processed on machine h is the heuristic solution of the $P\|C_{\max}$ problem makespan $\alpha = \max_{1 \leq h \leq m}\{T_h\}$.

The computational time required for algorithm LPT is $O(n \log n)$. Graham (1969) shows that the worst-case performance ratio of algorithm LPT is $(4/3 - 1/3m)$.

By performing some more computations, it may be possible to improve the makespan. To do this, we use the first-fit decreasing heuristic to pack a set of items into the least number of bins of capacity C. Coffman *et al.* (1978) used this idea to develop a multifit algorithm which assigns an available job to one machine at a time to pack as much of the capacity C on a machine as possible. In doing so, it attempts to pack the jobs with longest processing time first. If all jobs can be processed within capacity C, then, an attempt is made to reduce the makespan by decreasing capacity C. To start with, capacity C is found as the average of some known lower and upper bounds. To describe this algorithm, let $P = \sum_{i=1}^{n} p_i$. Then, the initial lower bound on the makespan, LB is calculated as follows:

$$LB = \max\left\{\max_{1 \leq i \leq n} p_i; P/m\right\}. \tag{2}$$

The initial upper bound, UB can be either the value of a known schedule or can be obtained by the following equation:

$$UB = \max\left\{\max_{1 \leq i \leq n} p_i; 2P/m\right\}. \tag{2}$$

The steps of the multifit heuristic, then, are as follows.

Algorithm MULTIFIT: The multifit procedure

Input: n, m, p_i for $i = 1, \ldots, n$, and b.

Step 1. Let $\sigma = (\sigma(1), \ldots, \sigma(n))$ be an ordering of jobs such that $p_{\sigma(1)} \geq p_{\sigma(2)} \geq \cdots \geq p_{\sigma(n)}$. Using equations (2) and (3), calculate LB and UB. Set $t = 1$ and go to step 2.

Step 2. Let $\rho = (\rho(1), \ldots, \rho(n)) = \sigma = (\sigma(1), \ldots, \sigma(n))$, and $C = (UB + LB)/2$. Set $T_1 = \cdots = T_m = 0$, and set $S_1 = \cdots = S_m = \emptyset$. Let $k = h = 1$, and go to step 3.

Step 3. If $T_h + p_{\sigma(k)} \leq C$, set $T_h = T_h + p_{\sigma(k)}$ and $S_h = S_h\sigma(k)$. Set $\rho = \{\rho - \sigma(k)\}$. Go to step 4.

Step 4. If $k < n$, set $k = k + 1$ and go to step 5; otherwise go to step 6.

Step 5. If $\sigma(k) \in \rho$, return to step 3; otherwise return to step 4.

Step 6. If $\rho = \emptyset$, set $\pi_h = S_h$ for $h = 1, 2, \ldots, m$, $\alpha = \max_{1 \leq h \leq m}\{T_h\}$, $UB = C$, and go to step 8; otherwise go to step 7.

Step 7. If $h < m$, set $h = h + 1$ and return to step 3; otherwise set $LB = C$ and go to step 8.

Step 8. If $t < b$, set $t = t + 1$ and return to step 2; otherwise, stop. The schedule where jobs in π_h are processed on machine h is an approximate solution of the $P\|C_{\max}$ problem with makespan α.

Algorithm MULTIFIT requires $O(n \log n + bm \log m)$ computational time, where b is the number of iterations. Friesen (1984) and Yue (1990) show that the worst-case performance ratio of algorithm MULTIFIT is $13/11 + 2^{-b}$. In our implementation of algorithm MULTIFIT, we fixed the number of iterations, $b = 7$, as recommended by Coffman *et al.* (1978).

The average-case performance bound of algorithm MULTIFIT can be improved by using the schedule obtained from algorithm L as the initial upper bound. Thus, if $C_{\max}(LPT)$ is the makespan of the schedule from algorithm LPT, the upper bound, UB used in the combine heuristic is given by:

$$UB = C_{\max}(LPT). \tag{4}$$

Further, we use an updated lower bound in algorithm MULTIFIT as follows:

$$LB = \left\{C_{\max}(LPT)/(4/3 - 1/3m), \max_{i \leq i \leq n} p_i; \sum_{i \in N}\{p_i/m\}\right\}. \tag{5}$$

With the above modifications, the steps of the COMBINE procedure, proposed by Lee and Massey (1988), are as follows.

Algorithm COMBINE: Combined LPT and MULTIFIT algorithm

Step 1. Let $C_{\max}(\text{LPT})$ be the makespan obtained by using algorithm L (the LPT rule) and $\pi_1, \ldots, \pi_m$ be the subsets of jobs assigned to machines $1, \ldots, m$ respectively. If $\alpha = C_{\max}(\text{LPT}) \geq 1.5 * \{\sum_{i \in N}\{p_i/m\}\}$ go to step 3, otherwise go to step 2.

Step 2. Let $\alpha = C_{\max}(M)$ be the makespan obtained by using algorithm MULTIFIT where UB and LB are calculated using equations (4) and (5). Go to step 3.

Step 3. Accept the schedule with makespan α as the solution of the $P\|C_{\max}$ problem.

Algorithm COMBINE requires $O(n \log n + bm \log m)$ computational time, where b is the number of iterations used in the multifit algorithm. As for algorithm MULTIFIT, its worst-case performance ratio is $(13/11 + 2^{-b})$.

4. The LISTFIT heuristic

The proposed LISTFIT heuristic combines the bin-packing method used in the MULTIFIT heuristic with multiple lists of jobs. The proposed LISTFIT heuristic divides the jobs into two sets and iteratively creates lists that are then assigned to the least loaded machine. By generating multiple combinations of the load list, there are opportunities for better packing combinations in the bin-packing process.

A list of jobs is created by combining two sub-lists, A and B. The jobs assigned to the two sub-lists change iteratively, as well as the ordering method for each sub-list. Jobs in list A and B are scheduled in an SPT or LPT order of processing times.

Let ω_r and ϕ_r be the ordering of jobs in lists A and B, respectively, according to an ordering rule r where $r = 1$ implies SPT order and $r = 2$ means an LPT order. Then, the steps of the proposed LISTFIT algorithm are as follows.

Algorithm LISTFIT: The SPT/LPT and MULTIFIT procedure

Input: n, m, p_i for $i = 1, \ldots, n$.

Step 1. Let $r = 1$, $q = 1$, and $C_{\max} = C_{\max}(LPT)$, the makespan obtained by the LPT algorithm. Go to step 2.

Step 2. Let $\phi = \emptyset$, $A = \{1, \ldots, n\}$; and $B = \emptyset$. Let ω_r be the sequence of jobs in job-list A ordered according to ordering r and go to step 3.

Step 3. Let $\alpha = C_{\max}(M)$ be the makespan obtained by using algorithm MULTIFIT with $\sigma = \phi_q \omega_r$ in step 1 of algorithm M. If $C_{\max} > \alpha$, set $C_{\max} = \alpha$, $\gamma_h = \pi_h$ for $h = 1, 2, \ldots, m$. If $A \neq \emptyset$, go to step 4; otherwise go to step 5.

Step 4. Remove the last job of ω_r and place it into B. Update A, ϕ_q and ω_r. Let $\sigma = \phi_q \omega_r$ and return to step 3.

Step 5. If $r < 2$, set $r = r + 1$ and return to step 2; otherwise, go to step 6.

Step 6. If $q < 2$, set $q = q + 1$, $r = 1$, and return to step 2; otherwise stop. The schedule where jobs in γ_h are processed on machine h is an approximate solution of the $P\|C_{\max}$ problem with makespan $C_{\max}$.

Algorithm LISTFIT sorts a list of jobs n times repeating this for four cycles, thus creating a total of $4n$ lists of all jobs. Taking all the steps, the proposed heuristic algorithm LISTFIT has a computational complexity of $O(n^2 \log n + n^2 b \log m)$, where b is the number of iterations used in the multifit algorithm. As for algorithm MULTIFIT, its worst-case performance ratio is $(13/11 + 2^{-b})$.

5. Empirical evaluation of algorithms

The effectiveness and efficiency of heuristic algorithms in finding minimum makespan schedules was empirically evaluated by solving a large number of problems. In this section, we describe the experimental framework used and the computational results.

5.1. *Experimental framework*

The relative performance of various heuristic algorithms was investigated under four experimental frameworks. Each experiment investigates three variables: the number of machines (m); the number of jobs (n); and the minimum and maximum values of a uniform distribution used to generate processing times (p_{gen}). Two of these experiments, E1 and E2, used the parameters presented by Kedia (1971), and Lee and Massey (1988), respectively. The parameters of experiment E3 were selected to generate 'difficult' problems as identified in our pilot experiments. Finally, experiment E4 evaluates the relative performance of the heuristics compared to the true

optimum found by full enumeration on two-machine nine-job problems, and three-machine 10-job problems.

Table 1 presents a summary of all four experiments. For experiment E1, the number of machines is set at three levels: 3, 4 and 5. The number of jobs is set at three levels: $2m$, $3m$, and $5m$. The processing times are integers generated from a uniform distribution. Thus, the p_{gen} variable is set at two levels: uniform $(1, 20)$ and $U(20, 50)$. For experiment E2, the number of machines is set at six levels: 2, 3, 4, 6, 8 and 10. The number of jobs is set at four levels: 10, 30, 50 and 100 (when n equals 10, m is only considered at 2 and 3), and p_{gen} is fixed at $U(100, 800)$, which approximates the truncated normal with range $(100, 800)$ used by Lee and Massey (1988). For experiment E3, the number of machines is considered at four levels, 3, 5, 8 and 10. The number of jobs is set at six levels: $3m + 1$, $3m + 2$, $4m + 1$, $4m + 2$, $5m + 1$ and $5m + 2$, and p_{gen} is set at two levels: $U(1, 100)$ and $U(100, 200)$. For experiment E4, the m and n variables are fixed, while p_{gen} is investigated at the combined five levels used in the three previous experiments. One hundred replications are taken at each experimental point for all experiments. This gives a total of 9600 problems.

The heuristics were implemented in Delphi 3 and ran on a Pentium II processor. The makespan for each problem instance was normalized by dividing it by its lower bound, LB in equation (2). Thus, the normalized makespan from heuristic h, denoted by r_h is calculated as follows:

$$r_h = C_{max}(h)/LB \tag{6}$$

To compare the performance of LISTFIT and COMBINE heuristics, the percentage of time $C_{\max}(\text{LISTFIT}) < C_{\max}(\text{COMBINE})$, q was calculated. A higher value of q indicates that the performance of LISTFIT is better than that of the COMBINE heuristic.

An analysis of variance was conducted (at a significance level of 0.01) for each experimental framework in order to determine the effect of the investigated factors (the number of jobs n, number of machines m, the range of processing times as determined by p_{gen}) in the performance of the heuristics. The analysis of variance was appropriate in this case given equal samples and a large number of replications. Levene's test was used to validate the assumption of homogeneous variance. Duncan's *post hoc* test was used to rank the heuristics.

5.2. *Effectiveness of heuristic algorithms*

The results are presented in tables 2–7. Each table presents the experimental parameters followed by the average r for 100 replications for the LPT, MULTIFIT, COMBINE and LISTFIT heuristics. While the performance of the LPT and MULTIFIT heuristics is not expected to be better than the COMBINE and LISTFIT heuristics, they were included as a point of reference and to provide an independent test of their performance. The last column of tables 2–7 provides the percentage of cases where the LISTFIT heuristic outperformed the COMBINE heuristic (q).

In all the experiments, the ratio n/m had a significant effect on the relative performance of LISTFIT versus COMBINE; performance increased as n/m increased. In relation to the CPU times, while the LISTFIT heuristic required more computational time than the COMBINE heuristic, the CPU time for a problem instance was still less than 1 s even for relatively large problems (10 machines and 100 jobs).

As expected, the COMBINE, and LISTFIT heuristics always outperformed the LPT and MULTIFIT heuristics. However, there are cases where the LPT algorithm outperformed the MULTIFIT heuristic. Details of the relative performance of the COMBINE and LISTFIT heuristics for each experiment are discussed below.

5.2.1. *Experiment E1*

Table 2 presents the results for experiment E1 . The performance of the LPT heuristic was similar to that reported by Kedia (1971) for the same experimental conditions; although in some parameter combinations, the ratio r for LPT found in our experiments was somewhat

Table 1. Summary of the experimental frameworks.

	m	n	p_{gen}
E1	3, 4, 5	$2m$, $3m$, $5m$	$U(1, 20)$, $U(20, 50)$
E2	2, 3, 4, 6, 8, 10	10, 30, 50, 100	$U(100, 800)$
E3	3, 5, 8, 10	$3m + 1$, $3m + 2$, $4m + 1$, $4m + 2$, $5m + 1$, and $5m + 2$	$U(1, 100)$, $U(100, 200)$
E4	2 3	9 10	$U(1, 20)$, $U(20, 50)$, $U(50, 100)$, $U(100, 200)$, $U(100, 800)$

Table 2. Results for experiment E1.

m	p_{gen}	n	LPT	MULTIFIT	COMBINE	LISTFIT	q
3	U(1, 20)	6	1.066	1.064	1.064	1.064	0
		9	1.028	1.018	1.015	1.011	4
		15	1.011	1.006	1.004	1.000	16
	U(20, 50)	6	1.043	1.043	1.043	1.043	0
		9	1.019	1.053	1.016	1.013	20
		15	1.007	1.016	1.005	1.003	35
4	U(1, 20)	8	1.069	1.065	1.064	1.064	0
		12	1.032	1.013	1.012	1.008	15
		20	1.011	1.006	1.004	1.000	13
	U(20, 50)	8	1.055	1.055	1.055	1.055	0
		12	1.022	1.050	1.021	1.018	19
		20	1.007	1.025	1.006	1.005	31
5	U(1, 20)	10	1.070	1.069	1.068	1.068	0
		15	1.037	1.015	1.014	1.010	14
		25	1.011	1.004	1.002	1.001	14
	U(20, 50)	10	1.049	1.049	1.049	1.049	0
		15	1.022	1.053	1.020	1.018	23
		25	1.007	1.028	1.007	1.006	25

Table 3. Results for experiment E2.

m	n	LPT	MULTIFIT	COMBINE	LISTFIT	q
2	10	1.013	1.012	1.008	1.002	67
3		1.042	1.024	1.021	1.014	54
2	30	1.002	1.002	1.001	1.000	83
3		1.004	1.005	1.003	1.001	79
4		1.018	1.007	1.007	1.002	91
6		1.017	1.013	1.010	1.006	79
8		1.037	1.018	1.017	1.013	57
10		1.046	1.023	1.020	1.017	57
2	50	1.001	1.001	1.000	1.000	81
3		1.005	1.002	1.002	1.000	94
4		1.010	1.003	1.003	1.001	95
6		1.018	1.006	1.006	1.002	92
8		1.024	1.007	1.007	1.004	80
10		1.018	1.010	1.009	1.007	65
2	100	1.000	1.001	1.000	1.000	70
3		1.005	1.001	1.001	1.000	97
4		1.001	1.001	1.001	1.000	89
6		1.006	1.002	1.002	1.001	96
8		1.011	1.003	1.003	1.001	90
10		1.004	1.004	1.003	1.002	73

higher. In this experiment, LISTFIT did not perform significantly better than COMBINE for most parameter combinations. LISTFIT only found a better solution than COMBINE in ~12% of the problems. However, as the number of jobs increases, the number of better solutions found by the LISTFIT heuristic is somewhat higher. Of all the experimental factors, only the number of jobs had a significant effect. The Duncan *post hoc* test resulted in statistical difference between LPT, MULTIFIT and COMBINE, but no difference between COMBINE and LISTFIT. The average performance for COMBINE is 1.024 and for LISTFIT is 1.022.

5.2.2. *Experiment E2*

Table 3 presents the results for experiment E2 . The performance of COMBINE in this experiment is similar

Table 4. Results for experiment E3: $p_{gen} = U(1, 100)$.

m	n	LPT	MULTIFIT	COMBINE	LISTFIT	q
3	10	1.126	1.023	1.023	1.016	35
	11	1.068	1.058	1.053	1.014	97
	13	1.102	1.035	1.034	1.008	82
	14	1.052	1.039	1.037	1.005	98
	16	1.082	1.045	1.044	1.005	95
	17	1.042	1.029	1.027	1.003	92
5	16	1.160	1.036	1.035	1.026	53
	17	1.120	1.043	1.042	1.021	92
	21	1.123	1.055	1.053	1.027	96
	22	1.092	1.041	1.039	1.020	97
	26	1.098	1.041	1.039	1.022	86
	27	1.075	1.032	1.030	1.021	69
8	25	1.175	1.049	1.047	1.039	64
	26	1.150	1.040	1.038	1.029	75
	33	1.133	1.055	1.053	1.039	89
	34	1.113	1.049	1.047	1.036	84
	41	1.108	1.042	1.039	1.028	80
	42	1.094	1.042	1.040	1.029	66
10	31	1.180	1.053	1.051	1.044	68
	32	1.159	1.042	1.040	1.033	65
	41	1.138	1.056	1.054	1.044	84
	42	1.123	1.052	1.049	1.042	77
	51	1.111	1.042	1.040	1.031	69
	52	1.099	1.046	1.043	1.030	83

Table 5. Results for experiment E3: $p_{gen} = U(100, 200)$.

m	n	LPT	MULTIFIT	COMBINE	LISTFIT	q
3	10	1.131	1.029	1.029	1.021	43
	11	1.069	1.061	1.055	1.017	96
	13	1.100	1.033	1.033	1.008	80
	14	1.052	1.043	1.040	1.007	100
	16	1.081	1.046	1.045	1.005	95
	17	1.042	1.030	1.028	1.003	97
5	16	1.158	1.037	1.036	1.027	54
	17	1.120	1.045	1.043	1.024	91
	21	1.123	1.052	1.050	1.024	98
	22	1.092	1.043	1.041	1.020	97
	26	1.099	1.041	1.039	1.022	90
	27	1.074	1.031	1.029	1.020	63
8	25	1.175	1.049	1.047	1.040	65
	26	1.153	1.044	1.042	1.032	76
	33	1.134	1.056	1.053	1.038	94
	34	1.117	1.047	1.045	1.034	93
	41	1.110	1.044	1.041	1.030	71
	42	1.094	1.043	1.040	1.030	73
10	31	1.180	1.052	1.050	1.044	68
	32	1.161	1.043	1.040	1.033	75
	41	1.139	1.057	1.055	1.045	86
	42	1.125	1.055	1.052	1.042	86
	51	1.112	1.044	1.041	1.032	71
	52	1.100	1.047	1.044	1.032	87

Table 6. Results for experiment E4.

m	n	LPT	MULTIFIT	COMBINE	LISTFIT	q
2, 9	$U(1,20)$	1.015	1.009	1.007	1.000	25
	$U(20,50)$	1.058	1.012	1.012	1.002	50
	$U(1,100)$	1.016	1.011	1.007	1.001	46
	$U(50,100)$	1.069	1.006	1.005	1.001	35
	$U(100,200)$	1.069	1.005	1.004	1.001	29
	$U(100,800)$	1.025	1.016	1.013	1.002	74
3, 10	$U(1,20)$	1.026	1.014	1.010	1.006	17
	$U(20,50)$	1.098	1.017	1.017	1.006	42
	$U(1,100)$	1.028	1.014	1.010	1.004	41
	$U(50,100)$	1.113	1.010	1.009	1.003	43
	$U(100,200)$	1.116	1.009	1.008	1.003	41
	$U(100,800)$	1.044	1.019	1.016	1.006	60

Table 7. Results for experiment E4: number of optimal solutions found out of 100.

m, n	p_{gen}	COMBINE	LISTFIT
2, 9	$U(1,20)$	75	100
	$U(20,50)$	35	73
	$U(1,100)$	42	79
	$U(50,100)$	49	77
	$U(100,200)$	59	75
	$U(100,800)$	13	47
3, 10	$U(1,20)$	64	79
	$U(20,50)$	32	52
	$U(1,100)$	40	60
	$U(50,100)$	38	67
	$U(100,200)$	39	64
	$U(100,800)$	12	28

to that reported by Lee and Massey (1988) for most parameter combinations. LISTFIT performed very well in this experiment when compared to COMBINE: out of a total of 2000 problem instances, LISTFIT found a better solution for 70.2% of the problems. The Duncan *post hoc* test resulted in statistical difference between all four heuristics, with LISTFIT being the best performer. In this experimental framework, none of the experimental factors had a significant effect on performance. The average performance for COMBINE is 1.006 and for LISTFIT is 1.004.

5.2.3. *Experiment E3*

Tables 4 and 5 present the results for experiment E3 when the processing times are generated by $U(1,100)$ and $U(100,200)$ respectively. In this experiment, the number of jobs and the number of machines factors were significant. As the number of jobs increased, the performance of all heuristics improved. The Duncan *post hoc* test resulted in statistical difference between all four heuristics, with LISTFIT being the best performer.

The LISTFIT heuristic performed very well in this experiment, finding better solutions that COMBINE in ~80% of the problems (total of 4800 instances). The average performance for COMBINE is 1.042 and for LISTFIT is 1.026.

5.2.4. *Experiment E4*

The results of E4 are presented in table 6 and demonstrate how close each heuristic is to the true optimal makespan value found by full enumeration. The average deviation from the true optimal makespan $(r-1)$ are 1% and 0.3%, respectively for COMBINE and LISTFIT.

Table 7 presents the number of times, out of 100 replications, that COMBINE and LISTFIT found the optimal makespan. LISTFIT finds the optimal solution for 66.8% of the problems compared to 41.5% for COMBINE. The Duncan *post hoc* test resulted in statistical difference between all four heuristics, with LISTFIT being the best performer. Finally, the p_{gen}, n, and, m variables had a significant effect on the ability of both algorithms to find optimal solutions.

5.2.5. *Summary of results*

As expected, in each case, the LISTFIT and COMBINE heuristics outperformed the LPT and MULTIFIT heuristics. The LISTFIT heuristic outperformed COMBINE for most of the experimental factors investigated. Given the formulation of LISTFIT, it will always generate a result as good as COMBINE, thus the analysis of the results focused on the level of improvement. While there was statistical significance in the

majority of the experimental factor combinations, the question remains about practical significance. The level of improvement ranged from 0.0% in some cases of E1 to an average improvement of 3–4% in some cases of E3, to an average improvement of 1.6% for all 4800 instances of E3, and an average improvement of ~6.4% for the problem instances in experiment E4. These percent improvements in makespan may have a practical effect in some cases as they may result in providing better customer service or decreasing the operating costs. Further, our experience shows that shops running at close to capacity levels greatly benefit from these improvements in machine utilization. The computational experiments reported here used simulated data from uniform distributions. Therefore, the effectiveness of the proposed heuristic could be different for practical problems using real data. However, Amar and Gupta (1986) showed that the scheduling problems using simulated data are harder to solve than those using real data. Therefore, for solving problems involving real-life data, the effectiveness of the proposed LISTFIT heuristic is likely to better than the one reported above.

6. Conclusions

This paper presents a new heuristic for the identical parallel machine makespan problem. The heuristic was compared to LPT, MULTIFIT and COMBINE, heuristics commonly used to solve the identical parallel machine makespan problem. Four experiments demonstrated how the proposed heuristic outperforms all three previous heuristics under a variety of levels of the number of jobs, machines and processing time distribution. These experiments showed that the proposed heuristic is quite robust and provides a better method to solve the makespan problem.

The proposed LISTFIT algorithm uses an efficient approach to generate an optimal or near-optimal schedule for minimizing makespan in insignificant amounts of CPU time. The proposed algorithm, or its modifications, could be implemented in a shop scheduling process and used to generate daily, weekly or monthly schedules. Given that the $P||C_{\max}$ problem has several assumptions, it may not reflect the complexity of a number of production environments. However, in such cases, the proposed algorithm can be used to generate an initial schedule. Based on the specific shop conditions, this initial schedule can then be modified by another heuristic or by a human scheduler using his/her experience and knowledge.

Several issues are worthy of future investigations. First, the development of tighter worse-case performance bounds for the proposed LISTFIT algorithm will be useful. Even a negative answer that the LISTFIT algorithm does not have a better worse-case performance bound than the MULTIFIT heuristic will be beneficial. Second, extension of the proposed LISTFIT heuristic to solve the multiple criteria identical-parallel-machine scheduling problems is both interesting and useful. Finally, extension of our results to more complex machine environments, e.g. multi-stage flowshop and job-shop problems, is important for application in industry.

References

AMAR, A. D., and GUPTA, J. N. D., 1986, Simulated versus real life data in testing the efficiency of scheduling algorithms. *IIE Transactions*, **18**, 16–25.

BLAZEWICZ, J., ECKER, K., PESCH, E., SCHMIDT, G., and WEGLARZ, J., 1996. *Scheduling Computer and Manufacturing Systems*. (Berlin: Springer).

COFFMAN, E. G., GAREY, M. R., and JOHNSON, D. S., 1978. An application of bin-packing to multi-processor scheduling. *SIAM Journal of Computing*, **7**, 1–17

FRIESEN, D. K., 1984. Tighter bounds for the MULTIFIT processor scheduling algorithm. *SIAM Journal of Computing*, **13**, 170–181.

GAREY, M. R., and JOHNSON, J. S., 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (San Francisco, CA: Freeman).

GRAHAM, R. L., 1969. Bounds on multiprocessor timing anomalies. *SIAM Journal of Applied Mathematics*, **17**, 416–429.

HAX, A., and CANDEA, D., 1984, *Production and Inventory Management* (Englewood Cliffs, NJ: Prentice Hall).

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

LEE, C. Y., and MASSEY, J. D., 1988. Multiprocessor scheduling: combining LPT and MULTIFIT. *Discrete Applied Mathematics*, **20**, 233–242.

PINEDO, M., and CHAO, X., 1999, *Operations Scheduling with Applications in Manufacturing and Services*. (Boston, MA: Irwin/McGraw Hill).

RIERA J., ALCAIDE, D., and SICILIA, J., 1996, Approximate algorithms for the $PC_{\max}$ problem. *TOP*, **4**, 345–359.

YUE, M., 1990. On the exact upper bound for the multifit processor algorithm. *Annals of Operations Research*, **24**, 233–259.