Int J Adv Manuf Technol (2004) 23: 383–388
DOI 10.1007/s00170-003-1716-9

ORIGINAL ARTICLE

Po-Chieng Hu

Minimising total tardiness for the worker assignment scheduling problem in identical parallel-machine models

Received: 1 March 2003 / Accepted: 4 April 2003 / Published online: 5 November 2003

Abstract The worker assignment scheduling problem involves both the decisions of job scheduling and worker assignment. In this research, only the performance measure of total tardiness is investigated in the model of identical parallel machines with nonpreemptive jobs. Since the worker assignment scheduling problem in the selected model can be shown to be NP-complete, heuristics have been developed for minimising the total tardiness. The worker assignment scheduling problem is solved in two phases of job scheduling and worker assignment. The SES (SPT, EDD, SLACK) heuristic is used for the phase of job scheduling. For the phase of worker assignment, the largest marginal contribution (LMC) procedure is used to minimise the total tardiness. From the simulation conducted, 88 out of 100 simulated problems yielded optimal solutions while the others also obtained very good results. In conclusion, the heuristics developed have shown very impressive results in both effectiveness and efficiency aspects.

Keywords Worker assignment scheduling problem · Tardiness · NP-complete · SES heuristic · LMC procedure

Notation

n	Total number of jobs waiting to be processed
m	Total number of machines available to process the above jobs
J_i	A set of n jobs are to be processed, $i = 1, ..., n$
M_j	A set of m machines are used to process these n jobs, $j = 1, ..., m$
$t_i(W_j)$	The processing time needed for job J_i processed on machine M_j, where W_j workers have been assigned to M_j
$t_{j,[k]}(W_j)$	The processing time function of the k^{th} job assigned to machine M_j, where W_j workers have been assigned to M_j
d_i	The due date of job J_i
$d_{j,[k]}$	The due date of the k^{th} job assigned to machine M_j
r_i	the ready time of job J_i
C_i	The completion time of job J_i
$C_{j,[k]}(W_j)$	The completion time function of the k^{th} job assigned to machine M_j, where W_j workers have been assigned to M_j; $C_{j,[k]}(W_j) = \sum_{p=1}^{k} t_{j,[p]}(W_j)$
F_i	The flow time of job J_i, $F_i = C_i - r_i$
L_i	The lateness of job J_i, $L_i = C_i - d_i$
T_i	The tardiness of job J_i, $T_i = \max\{0, L_i\}$
$T_{j,[k]}$	The tardiness of the k^{th} job assigned to machine M_j, $T_{j,[k]} = \max\{0, C_{j,[k]}(W_j) - d_{j,[k]}\}$
N_j	The number of jobs assigned to machine M_j
W_j	The number of workers assigned to machine M_j
W	The total number of workers

P.-C. Hu (✉)
Department of Industrial Management,
National Huwei Institute of Technology,
632 Yunlin, Taiwan, R.O.C.
E-mail: pchu@sunws.nhit.edu.tw

1 Introduction

1.1 Scheduling problem in the model of identical, parallel machines

In the real world, there are many examples of parallel-machine processing, e.g. a gas station can serve several cars simultaneously; bank counters can serve several customer at the same time; and automotive assembly lines can assemble several identical or nonidentical car models. In the classic parallel-machine processing model, there are two different conditions for the machines that are used: all the machines are identical or the machines may be nonidentical.

Many studies have been presented in the area of parallel-job scheduling. Some of these studies investigated ways of solving more classic problems with different performance measures [1, 2, 3, 4, 5, 6, 7]. Others studied such problems with some additional constraints [8, 9]. Yamada and Nakano [10] presented a genetic algorithm for the job-shop scheduling problem. Sule and Vijayasundaram [11] specifically dealt with a different scheduling problem that involved n jobs in m machine centres where each machine centre may have k number of identical processors. Iwamoto [12], Yamada and Nakeno [13], and Jain and Meeran [14] used different methods, such as fuzzy decision-making, deterministic local search, and neural networks, to solve specific problems.

In parallel-machine processing situations, minimising the total tardiness of the schedule is a very complicated and difficult thing. Gupta and Rothkopf [15, 16] suggested that the dynamic programming method could be used to get an optimal solution. However, even a very small problem (i.e. the numbers of machines and jobs are small), substantial computing is needed to get the solution. Therefore, Dogramaci and Sukis [17] proposed a heuristic to solve this kind of problem. Even this does not promise an optimal solution; however, the solution obtained will be the optimal solution or is not far away from it. In addition, the time consumed by using the heuristic is negligible compared to the dynamic programming or exhaustive methods.

1.2 Worker assignment scheduling problems

In the classic scheduling problem [18, 19, 20], no matter how many machines (work stations, processors, etc.) are involved, the number of operators (workers) at each machine may be ignored or assumed to be fixed and is not taken into consideration. However, in some cases, assigning more workers to work on the same job will decrease job completion time. Hence, ignoring worker assignment decision in these cases may cause managerial problems. In order to solve those problems and optimise the overall performance, decisions about job scheduling and worker assignment need to be resolved together.

In Hu's [21] research, the scope of classic scheduling problems had been extended in that the assumption of constant job processing time was no longer necessary. Under the original assumption, job processing time was not affected by the number of workers work on the job. But Hu's research suggested that there exist certain relationships between job processing time and the number of workers assigned to work on the job. Hence, job processing time is no longer a constant but related to the number of workers assigned to work on the job. Therefore, the worker assignment scheduling problem needs to solve two subproblems: how to assign jobs to machines and how to assign workers to machines.

1.3 Purposes of this study

This research focusses on the model of identical, parallel machines and uses total tardiness as the performance measure, that is, how to effectively assign the jobs and workers to machines to minimise the total tardiness in the model of identical, parallel machines.

This research has two purposes: one is to establish a mathematical model to describe the specific problem presented; the other is to develop efficient and effective heuristics to solve these two subproblems. In addition, hopefully, these heuristics can be adopted by industries to solve related problems.

2 Identical, parallel-machine processing worker assignment scheduling problem

The only difference between the worker assignment scheduling problem and the classic scheduling problem is that the former has an additional worker assignment problem. Hence, the processing times of jobs are no longer constants but related to the number of workers assigned to work on the job. Assumptions of the identical, parallel-machine processing worker assignment scheduling problem are as follows [21]:

1. Each job needs only one operation.
2. Each job only needs to be processed by one machine.
3. Any machine can process any job.
4. No machine may process more than one job at a time.
5. Machines will never break down and are available throughout the scheduling period.
6. Job processing times are independent of the job sequence.
7. Machine set-up time is negligible.
8. Transportation time between machines is negligible.
9. Number of jobs is fixed.
10. Number of machines is fixed.
11. There is a group of W workers that have the same abilities to perform the duties assigned to them.
12. W is within reasonable bounds, i.e. $m \leq W \leq W_c$. W_c is the maximum capacity of the resource.
13. These m machines, n jobs, and W workers are available at time zero.
14. The processing time function has a simplified form of $t_i(W_j) = A_i + B_i/(E_i \times W_j)$, where $t_i(W_j)$ is the processing time of job i that is processed on machine j in which the number of W_j workers are assigned to it, A_i is a fixed constant and not affected by the number of workers, $B_i/(E_i \times W_j)$ is the variable part and affected by the number of workers. In addition, $t_i(W_j) = A_i + B_i/(E_i \times W_j)$ is assumed to be a decreasing function, i.e. $A_i \geq 0$, $B_i > 0$, $E_i \geq 1$.
15. If any machine has work waiting to be processed, workers must be assigned to this machine to process the job. The machine cannot be kept idle.

16. The number of workers assigned to each machine needs to be decided before any job can be processed and they will not be reassigned until all the jobs have been completed.

2.1 The mathematical model for minimising the total tardiness in the model of the identical, parallel-machine processing worker assignment scheduling problem

Before solving this specific identical, parallel-machine processing worker assignment scheduling problem, its mathematical model should be constructed to understand the problem's characteristics. The performance measure used for this specific problem is total tardiness. If the tardiness of the k^{th} job assigned to machine M_j is $T_{j,[k]}$ and $T_{j,[k]}=\max\{0,\ C_{j,[k]}(W_j)-d_{j,[k]}\}$, then the total tardiness will be $\sum_{j=1}^{m}\sum_{k=1}^{Nj}T_{j[k]}=\sum_{j=1}^{m}\sum_{k=1}^{Nj}\ \max\{0, C_{j,[k]}(W_j)-d_{j,[k]}\}$. In addition, let $x_{i,j,[k]}=1$ when job J_i is the k^{th} job assigned to machine M_j, and $x_{i,j,[k]}=0$, otherwise. And let $Y=\sum_{j=1}^{m}\sum_{k=1}^{Nj}\ T_{j,[k]}$. Then, the mathematical model of minimising total tardiness of the identical, parallel-machine processing worker assignment scheduling problem is as follows:

min $\quad Y$

subject to:

$$\sum_{j=1}^{m} Nj = n$$

– The sum of the jobs assigned to all machines is n.

$$\sum_{j=1}^{m} Wj = W$$

– All the W workers must be assigned.

$$\sum_{j=1}^{m}\sum_{k=1}^{Nj} x_{i,j,[k]} = 1 \qquad i=1,\ \ldots,\ n$$

– Each job can only be assigned to one machine and its processing order is k^{th} on that machine.

$$\sum_{i=1}^{n}\sum_{k=1}^{Nj} x_{i,j,[k]} = N_j \qquad j=1,\ \ldots,\ m$$

– There are N_j jobs that are assigned to machine M_j.

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{Nj}\ x_{i,j,[k]} = n$$

– All the n jobs need to be assigned completely.

– $x_{i,j,[k]}=1$ or 0, N_j and W_j are integers greater than 0.

According to Garey and Johnson [22], the specific problem presented in this research is NP-complete. Since there is no way to obtain the optimal solution other than with a mathematical model, heuristics, a guided random/iterative search such as GA (genetic algorithm), or simulated annealing could be used to solve this problem.

2.2 Heuristic for minimising the total tardiness in the model of identical, parallel-machine processing worker assignment scheduling problem

Since this research has two subproblems, how to assign jobs and how to assign workers, the descriptions of these two subproblems will follow accordingly. First, the job assignment subproblem will be explored. In order to solve this subproblem the heuristic developed by Dogramaci and Surkis [17] can be applied after some modifications.

2.2.1 Procedure 1 (for assigning n jobs)

Step 1. List these n jobs in an order using priority rules SPT, EDD, and SLACK in sequence. Let $W_j=W$ for the processing time functions of all jobs.

Step 2. Among those jobs that have not yet been assigned to a machine choose the first one in the ordered list, assign it to the earliest available machine, then remove it from the ordered list. Repeat this step until all n jobs are assigned.

Step 3. Consider separately each of the m machines with the jobs assigned to it in step 2. Treat each as an individual single-processor tardiness problem, and resequence the assigned jobs to minimise the total tardiness.

Since the procedure presented above is mainly an application of SPT, EDD, and SLACK rules, it is referred to as the SES heuristic. After all n jobs have been assigned, we turn to the worker assignment subproblem. The solving procedures are as follows.

2.2.2 Procedure 2 (for assigning W workers)

Step 1. Assign one worker to every machine that has job on it; therefore, m workers are assigned.

Step 2. Assign the next worker to the machine that can minimise the total tardiness.

Step 3. Repeat step 2 until all these W workers have been assigned completely.

Step 4. Consider separately each of the m machines with the jobs assigned to it in step 3. Treat each as an individual single-processor tardiness problem, and resequence these assigned jobs to minimise the total tardiness.

Because the W workers are assigned according to their marginal contribution (i.e. the amount of total tardiness reduced by adding one more worker.), the procedure can be referred to as the largest marginal contribution procedure (LMC procedure).

2.2.3 Procedure 3 (for choosing the final solution)

Step 1. If the total tardiness of the schedule obtained in procedure 2 reaches zero, the schedule is the final solution. Otherwise, repeat procedures 1 and 2 until all the priority rules have been applied.

Step 2. If none of these schedules yields zero total tardiness, compare the total tardiness generated by these schedules. The schedule with the least total tardiness is the final solution.

3 Performance evaluation of the heuristics developed

In order to evaluate the performance of the heuristics developed in the previous section, the method of evaluation and the result of the evaluation will be described in the following text.

3.1 Method of evaluation

In this research, a simulation procedure is used to evaluate the performance of these heuristics; the procedure is illustrated as follows.

Step 1. Assume that the processing time function of any job J_i is $t_i(W_j)=A_i+B_i/(E_i\times W_j)$, where A_i, B_i, and E_i follow uniform distribution and $0 \le A_i < 100$, $1 \le B_i < 100$, and $1 \le E_i < 100$. Due date $d_i=A_i+B_i/(E_i \times W)+$ an integer between 1 and 99. This is because due date cannot be less than the least possible processing time.

Step 2. All the values of A_i, B_i, E_i and the integer part of d_i are generated randomly.

Step 3. 100 sets of worker assignment scheduling problems have been generated and each of them has the following characteristic: there are 10 workers available, 6 jobs waited to be processed, and 3 machines ready to process the jobs.

Step 4. An improved exhaustive search method is used to find the optimal solution for each of the 100 simulated problems. For example, some of the improvements are: jobs assigned to each machine will be adjusted to follow the EDD rule and once the total tardiness of any schedule reaches zero the searching process is terminated. Therefore, it is sometimes not necessary to exhaust all the possible schedules to get the optimal solution. Then, the heuristics are used to find the solutions of the 100 simulated problems. In order to compare the performance of these two, two performance measures are used:

1. Ratio 1 = total tardiness / least possible total processing time

   $$\text{Ratio 1} = \Sigma t_i/\Sigma[A_i + B_i/(E_i \times W)]$$

2. Ratio 2 = total tardiness / most possible total slack time

   $$\text{Ratio 2} = \Sigma t_i/\Sigma\{d_i - [A_i + B_i/(E_i \times W)]\}$$

These two performance measures allow the results obtained from the improved exhaustive method and heuristics to be compared under the same conditions. In addition, ratio 1 and ratio 2 can show the percentage and severity of the total tardiness compared to the least possible total processing time and the most possible total slack time.

Step 5. Compare the values of ratio 1 and ratio 2 of these 100 sets of simulated problems obtained from the improved exhaustive search method and heuristics.

3.2 Result of the evaluation

The results of first 30 sets of these 100 sets of simulated problems are shown in Tables 1, 2, and 3. According to the results, 88 of the 100 sets of simulated problems obtain the optimal solutions no matter whether the improved exhaustive search method or the heuristics is

Table 1 Results of 1–12 sets of the simulated problems

Data Set		1	2	3	4	5	6	7	8	9	10	11	12
SPT	Ratio 1	0	.074962	.013973	.063293	.008120	0	.051791	.008978	.007745	.082942	0	0
	Ratio 2	0	.084365	.012659	.110908	.007096	0	.061351	.011027	.009442	.090788	0	0
EDD	Ratio 1	0	0	.013973	.063293	0	0	.062874	.064007	0	.050388	0	0
	Ratio 2	0	0	.012659	.110908	0	0	.074479	.078614	0	.055155	0	0
Slack	Ratio 1	0	0	0	.082304	0	0	.047556	.008978	.010193	0	0	0
	Ratio 2	0	0	0	.144221	0	0	.056334	.011027	.012425	0	0	0
Exhaus.	Ratio 1	0	0	0	.037937	0	0	0	.008978	0	0	0	0
Search	Ratio 2	0	0	0	.066478	0	0	0	.011027	0	0	0	0

Table 2 Results of 13-24 sets of the simulated problems

Data Set		13	14	15	16	17	18	19	20	21	22	23	24
SPT	Ratio 1	0	0	.045508	0	0	.066879	0	0	.125413	.080405	0	.024912
	Ratio 2	0	0	.059321	0	0	.067299	0	0	.130697	.094587	0	.029622
EDD	Ratio 1	0	0	.091851	0	.294293	.014092	0	0	.058481	.027577	0	.024912
	Ratio 2	0	0	.119728	0	.286339	.014181	0	0	.060945	.032442	0	.029622
Slack	Ratio 1	0	0	.040650	0	0	0	0	0	0	0	0	.021709
	Ratio 2	0	0	.052987	0	0	0	0	0	0	0	0	.025813
Exhaus.	Ratio 1	0	0	.037551	0	0	0	0	0	0	0	0	.021709
Search	Ratio 2	0	0	.048948	0	0	0	0	0	0	0	0	.025813

Table 3 Results of 25-30 sets of the simulated problems

Data Set		25	26	27	28	29	30
SPT	Ratio 1	0	.048081	0	0	0	.052592
	Ratio 2	0	.044608	0	0	0	.044201
EDD	Ratio 1	0	0	0	0	0	0
	Ratio 2	0	0	0	0	0	0
Slack	Ratio 1	0	0	0	0	0	0
	Ratio 2	0	0	0	0	0	0
Exhaus.	Ratio 1	0	0	0	0	0	0
Search	Ratio 2	0	0	0	0	0	0

used. For the remaining 12 sets, the average difference between the exhaustive search method and the heuristics are 0.0172 for ratio 1 and 0.0254 for ratio 2; all are smaller than 3%. That is, even though the solutions obtained from the heuristics are not optimal, they are not far from it. In addition, by using a PC with Intel Pentium 4 2.00 GHz CPU, it only took 4 s using the heuristics compared to 117 s using the improved exhaustive search method; a 96.58% savings. Therefore, the heuristics developed in this research are efficient and effective.

4 Conclusion

In this research, worker assignment scheduling problems are presented, defined, and formulated. Since only the exhaustive search method can obtain optimal solutions to these problems, heuristics have been developed. After the simulation, results have shown that the heuristics developed in this research can obtain optimal or near optimal solutions for these simulated problems.

In the future, the conclusions of this research may be extended to scheduling problems in the real world. The task may be highly complicated; nonetheless, it will be an interesting and challenging mission.

5 Appendix A

Table 4 summarizes the data from one of the 100 simulated problems. The final sequence and worker assignment for this problem is given in Table 5.

Table 4 Data of one of the 100 simulated problems; $t_i(W_j) = A_i + B_i/(E_{i\times}W_j)$

Jobs	1	2	3	4	5	6
A	46	29	62	64	26	27
B	83	82	59	98	91	23
E	69	98	25	53	11	99
due date	113	31	119	74	38	107
Due date − *A*	67	2	57	10	11	80

Table 5 Final sequence and worker assignment for the problem data in Table 4

	SPT			EDD			Slack			Exhaus. search		
W_j	1	1	8	6	3	1	8	1	1	1	3	6
M_j	1	2	3	1	2	3	1	2	3	1	2	3
Job sequence	5–1	6–3	2–4	2–1	5–6–3	4	2–1	4–6	5–3	2	5–6–3	4–1
Total tardiness	0			0			0			0		
Ratio 1	0			0			0			0		
Ratio 2	0			0			0			0		

References

1. Bruno J, Coffman EG Jr, Sethi R (1974) Scheduling independent tasks to reduce mean finishing time. Commun ACM 17(7):382–387
2. Rajaraman MK (1975) An algorithm for scheduling parallel processors. Int J Prod Res 13(5)
3. Barnes JW, Brennan JJ (1977) An improved algorithm for scheduling jobs on identical machines. AIIE Trans 9(1)
4. Du J, Leung JYH (1989) Complexity of scheduling parallel task systems. SIAM J Discrete Math 2(4):473–487
5. Ghosal D, Serazzi G, Tripathi SK (1991) The processor working set and its use in scheduling multiprocessor systems. IEEE Trans Softw Eng 17(5):443–453
6. Feitelson DG, Rudolph L (1992) Gang scheduling performance benefits for fine-grain synchronization. J Parallel Distributed Comput 16(4):306–318
7. Wand Q, Cheng KH (1992) A heuristic of scheduling parallel tasks and its analysis. SIAM J Comput 21(2):281–294
8. Hurink J, Knust S (2001) List scheduling in a parallel machine environment with precedence constraints and setup times. Oper Res Lett 29:231–239
9. Sprecher A, Drexl A (1998) Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm. Eur J Oper Res 107: 431–450
10. Yamada T, Nakano R (1997) Genetic algorithms for job-shop scheduling problems. In: Proceedings modern heuristic decision support, UNICOM seminar, 18–19 March 1997, London, pp 67–81
11. Sule DR, Vijayasundaram K (1998) A heuristic procedure for makespan minimisation in job shops with multiple identical processors. Comput Ind Eng 35(3–4):399–402
12. Iwamoto S (2001) Fuzzy decision-making through three dynamic programming approaches. Int J Fuzzy Syst 3(4)
13. Yamada T, Nakeno R (1990) Job-shop scheduling by simulated annealing combined with deterministic local search. Kruwei Academic, MA, pp 237–246
14. Jain AS, Meeran S (1998) Job-shop scheduling using neural networks. Int J Prod Res 36(5):1249–1272
15. Gupta JND, Maykut AR (1973) Scheduling jobs on parallel processors with dynamic programming. Decision Sci 4(4)
16. Rothkopf MH (1966) Scheduling independent tasks on parallel processors. Manage Sci 12(5)
17. Dogramaci A, Surkis J (1979) Evaluation of a heuristic for scheduling independent jobs on parallel identical processors. Manage Sci 25(12):1208–1216
18. Baker KR (1973) Procedures for sequencing tasks with one resource type. Int J Prod Resources 11(2)
19. Baker KR (1974) introduction to sequencing and scheduling. Wiley, New York, pp 10–11
20. French S (1982) Sequencing and scheduling: an introduction to the mathematics of the job shop. Ellis Horwood, Chichester, pp 8–9
21. Hu P (1993) An efficient heuristic for the worker assignment problem in the identical and nonidentical parallel-machine models. PhD dissertation, Department of Industrial and Management Systems Engineering, Pennsylvania State University, University Park, Pennsylvania, p 2
22. Garey MR, Johnson DS (1979) Computers and intractability, a guide to the theory of NP-completeness. W.H. Freeman, New York, p 245