Advanced Materials Research Vols. 488-489 (2012) pp 1708-1712
Online available since 2012/Mar/15 at www.scientific.net

doi:10.4028/www.scientific.net/AMR.488-489.1708

Comparison of Heuristics for Identical Parallel Machine Scheduling

Dhiren Kumar Behera and Dipak Laha[1]

Mechanical Engineering Department

Jadavpur University, Kolkata 700032, India

[1]Corresponding author, email: dipaklaha_jume@yahoo.com

Keywords: Scheduling, parallel machine, makespan, optimization, heuristics, time complexity

Abstract. This paper addresses the problem of scheduling *n* independent jobs processed nonpreemtively on *m* identical parallel machines with the objective of minimizing makespan. Since these scheduling problems are well known to be NP-hard, among various solution methodologies, heuristics are preferred most. They guarantee near-optimal solutions and due to their polynomial time algorithms require reasonable computational effort, especially for solving large problem sizes. We consider three popular heuristics, multifit, combine and listfit since the seminal work of McNaughton in 1959. We present different experimental frameworks to investigate these heuristics for a comprehensive comparative performance evaluation. We show, through computational experimentation, that listfit outperforms the multifit and combine heuristics in most of the problem instances, however, at the cost of increased time complexity. The computational results also reveal that the combine heuristic performs better than the multifit heuristic, while requiring almost similar computational effort.

Introduction

The problem of scheduling a set of *n* independent jobs to a given set of *m* identical parallel machines with the objective of minimizing makespan has been intensely studied since the publication of the pioneering work of McNaughton [1]. The parallel machine scheduling problems are routinely encountered in many real-life problems such as production lines, shipping docks, university, hospitals, and computer systems [2, 3]. A comprehensive review of these problems has been given by Mokotoff [4] and Chen and Sin [5]. Due to the NP-hardness [6] of this scheduling problem, computation of complete enumeration of all possible schedules is prohibitively large and optimization-based heuristic approaches owing to their polynomial time complexity are extensively applied to obtain near-optimal schedules especially, for problems involving a large number of jobs.

The problem is to obtain an assignment of *n* jobs for processing nonpreemptively on *m* identical parallel machines such that the makespan objective is minimized. Several heuristics have been proposed for its solutions and some popular heuristics are LPT (longest processing time) [7], multifit [8], combine [9], and listfit [10]. Gupta and Ruiz-Torres [10] presented the listfit heuristic and showed with empirical results that it outperforms the LPT, multifit, and combine heuristics at the expense of higher computational complexity. Paletta and Pietramala [11] proposed an approximation algorithm and showed their worst-case performance ratio outperforms the LPT and multifit heuristics. Apart from heuristics, development of lower and upper bounds strategies and exact optimization algorithms for this problem have also been studied. Haouari et al. [12] presented lower and upper bounds for this fundamental problem. Dell'Amico and Martello [13] derived a lower bound for the one-dimensional bin packing problem as well as $P||C_{max}$ problem. Ghomi and Ghazvini [14] proposed a pairwise interchange algorithm for this problem. Theoretical aspects of the problem have been given in [15-18]. In recent years, metaheuristics have been applied to identical parallel machine scheduling problems [19-22].

The objective of this study is to compare the currently best listfit heuristic with other heuristics by reporting improvement in solution quality by percentage interval considering each problem instance rather than just reporting the average improvement for each problem size and the number

of times a heuristic is better than another as given in the previous study. The heuristics also will be compared with their CPU times. Section 2 presents the parallel machine problem formulation. In Section 3, a brief contribution of the existing heuristics is discussed; Section 4 presents the results of computational experimentation and finally, Section 5 provides conclusions.

The Scheduling Problem

The parallel machine scheduling is a production scheduling problem in which a set of *n* independent jobs is to be processed on *m* nonpreemptive identical parallel machines with the objective of optimizing certain performance measure such as makespan and total tardiness [23]. We assume that a machine processes one job at a time and once a job starts processing on either of the *m* machines it must be completed without preemption. It is also assumed that ready times of all jobs are zeros. Let us denote a set of *n* independent jobs to be scheduled to a set of *m* identical machines, p_i (i=1, ...,n) the processing time (including any setup time required) of job i, S_j the subset of jobs to machine M_j in a schedule of jobs with completion time $C(S_j) = \sum_{i \in s_j} p_i$. The makespan of *n* jobs, $C_{max}(S)$ of the schedule S is given by $C_{max}(S) = \max C(S_j)$ with corresponding schedule of jobs given by the subset of jobs S_j. The above parallel machine scheduling problem with the objective of minimizing makespan is designated as $P||C_{max}$ problem where P represents the identical parallel machines and C_{max} the makespan or maximum completion time of all *n* jobs.

The Existing Heuristics

Since the parallel machine scheduling problems belong to the class of NP-hard, several heuristics with polynomial complexity have been developed for obtaining near-optimal solutions. The well-known LPT heuristic by Graham [6] has long received the attention of researchers because it is very simple, fast, and yields considerably good solution quality. The LPT heuristic always finds a schedule having makespan within $(4/3 - 1/3m)C_{opt}$, where C_{opt} is the optimal makespan of the problem, i.e., its worst-case performance ratio is$(4/3 - 1/3m)$. The LPT continued to be best until the development of multifit heuristic by Coffman et al. [8]. The multifit provides better error bound than LPT rule at the expense of increased time complexity. However, it has been observed that LPT rule even performs better than multifit in many instances. The worst-case performance ratio of the multifit is $(13/11 + 2^{-b})$ as shown by Friesen [24] and Yue [25]. Next, Lee and Massey [9] proposed the combine heuristic using the LPT as an incumbent and then applying multifit with a small number of iterations, resulting in always better error bound than those of LPT and multifit. Recently, Gupta and Ruiz-Torres [10] presented the listfit heuristic based on the bin packing problem and the list scheduling. They have shown that listfit heuristic outperforms the LPT, the multifit, and the combine heuristics. However, its time complexity is higher than that of the multifit. The comparative performance measures of these heuristics are presented in Table1. The parameter *b* in Table 1 denotes the number of iterations used in the heuristics.

Table 1. Comparison of heuristics

Performance measure	LPT	multifit	combine	listfit
Worst-case performance ratio	$(4/3 - 1/3m)$	$13/11 + 2^{-b}$	$13/11 + 2^{-b}$	$13/11 + 2^{-b}$
Algorithmic complexity	O ($n \log n + n\, m$)	O ($n \log n + b\, n \log m$)	O ($n \log n + b\, n \log m$)	O ($n^2 \log n + n^2 b \log m$)

Computational Experimentation

In this section, we investigate a comparative performance of multifit, combine and listfit heuristics. All these heuristics have been coded in C and run on a Pentium 4, 2.66 GHz processor. The experimentation is presented in two phases. The first phase considers small-sizes problems with 8, 10, 15 and 20 jobs and 2, 3, and 4 machines, while the second phase comprises large-sized

problems with 70, and 100 jobs and 2, 3, 6, 8 machines and 300 jobs with 2, 3, 8, and 10 machines. Hundred problem instances were considered for each combination of jobs and machines. Hence, a total of 1200 problems were taken in the first phase and 1200 problems in the second phase. The problem instances were generated by setting the processing times of jobs to random integers (1 to 100) following a discrete uniform distribution.

In order to compare the performance of the heuristics, we consider three performance measures, namely, percentage improvement in makespan (Δ) for each problem instance, mean and standard deviation of ratio of makespan and lower bound (LB). The LB is defined as follows: $LB = max\{\max_{1\le i\le n} p_i\ ;\ \sum_{i=1}^{n} {}^{p_i}/_{m}\}$. For a problem instance comparing Δ is listfit given by: $\Delta = \left(\frac{LPT - listfit}{LPT}\right) x\ 100$, and similarly for other two heuristics. Tables 2 and 3 display comparative results for the small-sized problems in phase 1. For the large-sized problems in phase 2, comparative evaluations are reported in Tables 4 and 5.

Table 2 compares the results of listfit and multifit and indicates the improvement of one method over the other in terms of number of problem instances by percentage interval and the average Δ for each problem size. listfit shows the improvement over the multifit in approximately 41% cases and the average Δ is 0.44%. In approximately 36% cases, the improvement of the listfit is less than 2% and for 5% cases the improvement equals or exceeds 2%. As an illustration shown * mark in Table 2, there are 15 cases out of 100 problems where listfit makespan is 1-2% less than the multifit makespan. Table 3 presents the results of comparison of listfit and combine heuristics. Listfit improves over combine in nearly 22% cases and the mean Δ in this comparison is found to be 0.21% .Tables 4 and 5 shows the comparative results of listfit versus multifit and listfit versus combine for large problem sizes. In Table 4, it is found that listfit is better than combine in 93% occasions in the range of 0-1% improvement. Also the average Δ in this comparison is 0.56%. Similarly, the results of table 5 depict that the improvement of listfit over combine is nearly 36% in the interval of 0-1%. Table 6 displays the average CPU times (in 10^{-6} seconds) of the heuristics. The results reveal that listfit requires a considerably higher computational time compared to those required by the multifit and combine.

Conclusions

In this paper, we have investigated a comparative performance of three most popular heuristics from the literature for the problem of identical parallel machine scheduling with the minimization of makespan objective. The experimental results reveal that the listfit performs best in most of the cases. However, its time complexity is significantly higher. It is also evident that the listfit heuristic performs better than the multifit in about 40% of total problem instances with respect to small-sized problems, whereas for large problems, it is better in about 93% cases. Similarly, the listfit heuristic shows better results than the combine, in about 25% and 30% for small and large problems respectively. It is to be noted that the combine performs better than the listfit in some problem instances for both small and large problems. However, the superiority of the multifit over the listfit has not been observed in a single problem instance for either of the problem sizes. The results of CPU times show that the listfit consumes an excessively very high computational times whereas the multifit and the combine take reasonable time and are comparable to each other.

Table 2. Comparison of listfit to multifit [(multifit - listfit) / multifit] for small – sized problems

n	8	8	8	10	10	10	15	15	15	20	20	20	
m	2	3	4	2	3	4	2	3	4	2	3	4	
% Δ	Number of problem instances in the interval												Total
< 0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	55	81	100	46	65	89	35	45	64	20	50	60	710
0 – 1	16	6	0	27	8	0	56	41	16	79	44	26	319
1 – 2	15*	3	0	17	13	7	6	12	16	1	6	13	109
2 – 3	5	3	0	8	5	2	1	1	4	0	0	1	30
3 – 4	5	3	0	1	2	1	2	1	0	0	0	0	15
4 – 5	1	2	0	1	1	1	0	0	0	0	0	0	15
5 – 6	1	1	0	0	4	0	0	0	0	0	0	0	6
6 – 7	1	1	0	0	1	0	0	0	0	0	0	0	3
7 – 8	0	0	0	0	1	0	0	0	0	0	0	0	1
8-9	1	0	0	0	0	0	0	0	0	0	0	0	1
Avg. Δ	0.87	0.46	0	0.68	0.82	0.23	0.40	0.46	0.44	0.33	0.27	0.34	

Table 3. Comparison of listfit to combine [(combine - listfit) / combine] for small – sized problems

n	8	8	8	10	10	10	15	15	15	20	20	20	
m	2	3	4	2	3	4	2	3	4	2	3	4	
% Δ	Number of problem instances in the interval												Total
< 0	0	0	0	1	1	1	2	3	6	7	16	7	44
0	78	90	100	64	81	89	66	59	71	78	59	64	899
0-1	14	3	0	25	6	0	29	29	11	15	22	25	179
1 – 2	5	2	0	7	7	6	1	8	9	0	3	4	52
	0	0	0	0	0	0	0	0	0	0	0	0	0
2 – 3	2	2	0	3	3	2	2	0	3	0	0	0	17
3 – 4	1	1	0	0	0	1	0	1	0	0	0	0	4
4-5	0	2	0	0	0	1	0	0	0	0	0	0	3
	0	0	0	0	1	0	0	0	0	0	0	0	1
	0	0	0	0	1	0	0	0	0	0	0	0	1
Avg. Δ	0.36	0.22	0	0.31	0.33	0.21	0.16	0.26	0.24	0.03	0.06	0.13	

Table 4. Comparison of listfit to multifit [(multifit - listfit) / multifit] for large – sized problems

n	70	70	70	70	100	100	100	100	300	300	300	300	
m	2	3	6	8	2	3	6	8	2	3	8	10	
% Δ	Number of problem instances in the interval												Total
< 0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	14	56	0	0	1	13	0	0	0	0	84
0 – 1	100	100	86	44	100	100	99	87	100	100	100	100	1116
Avg. Δ	0.69	0.60	0.29	0.12	0.73	0.68	0.48	0.30	0.76	0.75	0.68	0.65	

Table 5. Comparison of listfit to combine [(combine - listfit) / combine] for large – sized problems

n	70	70	70	70	100	100	100	100	300	300	300	300	
m	2	3	6	8	2	3	6	8	2	3	8	10	
% Δ	Number of problem instances in the interval												Total
< 0	2	0	0	0	2	0	0	0	0	0	0	0	4
0	92	63	31	16	92	80	40	25	100	99	66	58	762
0 – 1	6	37	69	84	6	20	60	75	0	1	34	42	434
Avg. Δ	0.0029	0.045	0.19	0.26	0.0015	0.016	0.121	0.213	0	0.00013	0.03	0.03	

Table 6. Average CPU times (in 10^{-6} seconds) for the heuristics

n	*m*	multifit	combine	listfit
100	2	20	22	1767
	3	24	32	1968
	6	20	44	3293
	8	20	56	3755
200	2	44	66	7269
	3	44	66	8153
	8	64	132	13695
	10	64	154	15422
300	2	88	120	15663
	3	112	132	17671
	6	108	176	25673
	8	132	198	29558

References

[1] R. McNaughton: Management Science Vo.6 (1959), p. 1
[2] E. Mokotoff, J. L. Jimeno, and A. I. Gutierrez: TOP Vol.9 (2001), p. 243
[3] L. H. Su and C. Y. Lien, Int. J. Production Economics Vol. 17 (2009), p. 256
[4] E. Mokotoff: Asia-Pacific Journal of Operational Research Vol.18 (2001), p. 193
[5] T. C. E. Chen and C. C. S. Sin: European Journal of Operational Research Vol. 47 (1990), p. 271
[6] M. R. Garey and J. S. Johnson: *Computers and intractability: A guide to the theory of NP-Completeness,* (San Francisco CA, Freeman 1979).
[7] R. L. Graham: SIAM Journal of Applied Mathematics Vol.17 (1969), p. 416
[8] E. G. Coffman, M. R. Garey and D. S. Johnson: SIAM Journal of Computing Vol. 7 (1978), p.1
[9] C. Y. Lee and J. D. Massey: Discrete Applied Mathematics Vol. 20 (1988), p. 233
[10] J. N. D. Gupta and J. Ruiz-Torres: Production Planning & Control Vol. 12 (2001), p. 28
[11] G. Paletta and P. Pietramala: SIAM Journal of Discrete Math Vol. 21 (2007), p.313
[12] M. Haouari, A. Gharbi and M. Jemmali: Intl. Trans. Op. Res. Vol.13 (2006), p.529
[13] M. Dell'Amico and S. Martello: ORSA Journal on Computing Vol. 7, (1995), p. 191
[14] S. M. T. Fatemi Ghomi and F. Jolai Ghazvini: Production Planning & Control Vol. 9 (1998), p. 685
[15] J. F. Lin and S. J. Chen: Computers Math. Applic Vol. 28 (1994), p. 85
[16] G. Chiaselotti, M. I. Gualtieri and P. Pietramala: J Math Model Algor Vol. 9 (2010), p. 39
[17] J. M. Van Den Akker, J. A. Hoogeveen and S. L. Van De Velde: Operations Research Vol. 47 (1999), p. 862
[18] A. Dogramaci and J. Surkis: Management Science Vol.25 (1979), p. 1208
[19] J. Chen , Q. K. Pan, L. Wang and J. Q. Li, Engineering Optimization (2011), p. 1
[20] A. H. Kashan and B. Karimi, Computers & Industrial Engineering Vol. 56 (2009), p. 216
[21] L. Min and W. Cheng: Artificial Intelligence in Engineering Vol.13 (1999), p. 399
[22] W. C. Lee, C. C. Wu and P. Chen: Int J Adv Manuf Technol Vol. 31 (2006), p. 328
[23] M. Pinedo: *Scheduling: Theory, algorithms, and systems* (Prentice Hall, Upper Saddle River, New Jersey, 2002).
[24] D. K. Friesen: SIAM Journal of Computing Vol. 13 (1984), p. 170
[25] M. Yue: Annals of Operations Research Vol. 24 (1990), p. 233

Key Engineering Materials II
10.4028/www.scientific.net/AMR.488-489

Comparison of Heuristics for Identical Parallel Machine Scheduling

10.4028/www.scientific.net/AMR.488-489.1708