Available online at www.sciencedirect.com
ScienceDirect
Omega 37 (2009) 734–739
omega
The International Journal of Management Science
www.elsevier.com/locate/omega

Technical note

A heuristic to minimize total flow time in permutation flow shop☆

Dipak Laha[a], Subhash C. Sarin[b,*]

[a]Department of Mechanical Engineering, Jadavpur University, Kolkata 700032, India
[b]Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA 24061, USA

Received 1 December 2007; accepted 11 May 2008
Available online 16 May 2008

Abstract

In this paper, we address an n-job, m-machine permutation flow shop scheduling problem for the objective of minimizing the total flow time. We propose a modification of the best-known method of Framinan and Leisten [An efficient constructive heuristic for flowtime minimization in permutation flow shops. Omega 2003;31:311–7] for this problem. We show, through computational experimentation, that this modification significantly improves its performance while not affecting its time-complexity.

Keywords: Flow shop scheduling; Total flow time; Heuristic procedure

1. Introduction

A general flow shop is a manufacturing system where n jobs are processed by m machines in the same order to optimize a certain performance measure [1,2]. One important performance measure is total flow time or, equivalently, mean flow time, which leads to rapid turn-around of jobs and minimization of in-process inventory [2].

Since flow shop as well as job shop problems with few exceptions have been proved to be NP-hard [3], heuristic procedures are the most suitable ones for their solution, especially for large-size instances. Some noteworthy constructive heuristics for the total flow time criterion have been developed by Rajendran and Chaudhuri [4], Rajendran [5], Rajendran and Ziegler [6], Woo and Yim [7], Framinan and Leisten [1], and Liu and Reeves [8]. Flow shop scheduling with the makespan objective has been investigated by Kalczynski and Kamburowski [9], Kalir and Sarin [10], Sarin and Lefoka [11], Osman and Potts [12], Rad et al. [13], Simons [14], and Huq et al. [15]. More recently, another class of heuristics, called composite heuristics, has been studied by Li et al. [16], Allahverdi and Aldowaisan [17], and Framinan et al. [18] that rely on a combination of good constructive heuristics.

The method proposed by Framinan and Leisten [1] relies on the idea of optimizing partial schedules contained in the heuristic procedure proposed by Nawaz et al. [19]. Framinan et al. [18] have presented a review and comparative evaluation of different existing heuristic procedures in permutation flow shops for total flow time criterion. Based on this and other studies, the method proposed by Framinan and Leisten [1] is considered the best procedure for the minimization of total flow time. In this paper, we propose a modification of this method and experimentally demonstrate that

☆This manuscript was processed by Area Editor B. Lev.

*Corresponding author. Tel.: +1 540 231 6656; fax: +1 540 231 3322.
E-mail addresses: dipaklaha_jume@yahoo.com (D. Laha), sarins@vt.edu (S.C. Sarin).

0305-0483/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.omega.2008.05.002

this modification significantly improves its performance while not affecting its time-complexity.

The remainder of this paper is organized as follows. The heuristic of Framinan and Leisten [1] and its modification are presented in Section 2. Results of our experimental investigation are presented in Section 3. Finally, concluding remarks are made in Section 4.

2. The method of Framinan and Leisten and its modification

The method proposed by Framinan and Leisten [1] for the total flow time criterion is based on the idea of optimizing partial schedules. This concept is similar to that presented in the NEH heuristic method of Nawaz et al. [19] for the makespan criterion. The pseudocode of this heuristic is given below:

Step 1: For each job i, find the total processing time T_i which is given by

$$T_i = \sum_{j=1}^{m} t_{ij} \quad \text{for all } i = 1, 2, \ldots, n.$$

Step 2: Sort the jobs in ascending order of the sum of their processing times on all machines.

Step 3: Set $k = 2$. Select the first two jobs from the sorted list and select the better between the two possible sequences.

Step 4: Increment k, $k = k + 1$. Select the kth job from the sorted list and insert it into k possible positions of the best partial sequence obtained so far. Among the k sequences, the best k-job partial sequence is selected based on minimum total flow time. Next, determine all possible sequences by interchanging jobs in positions i and j of the above partial sequence for all i, j $1 \leqslant i < k$, $i < j \leqslant k$). Select the best partial sequence among $k(k-1)/2$ sequences having minimum total flow time.

Step 5: If $k = n$, then STOP; else, go to Step 4.

Note that Step 4 dictates the time-complexity of this heuristic. In this step, first, k schedules are generated as a result of the insertion operation, which is followed by the generation of $k(k-1)/2$ schedules because of the pairwise interchanges performed on the best among the k schedules. This gives rise to a total $k(k+1)/2$ schedules. The complexity of total flow time calculation for each schedule of k jobs on m machines is $O(km)$. Since Step 4 is executed for every $k = 2, \ldots, n$, the overall time-complexity of the method of Framinan and Leisten [1] is $O(kmkk(k+1)/2)$ or $O(n^4m)$.

Our modification pertains to Step 4 of the above procedure, and it implements another iteration of the insertion step of the NEH heuristic rather than performing pairwise interchanges, thereby generating a more effective neighborhood as demonstrated by Nawaz et al. [19]. Step 4 is, now, executed as follows.

Step 4: For $k = 3$ to n do the following.

Insert the kth job on the sorted list into k possible positions of the $(k-1)$-job current sequence, thereby generating k, k-job partial sequences, and select from these a k-job partial sequence with the best total flow time value. Designate this as a k-job current sequence. Place each job (except for the kth job of the sorted list) of this sequence into its $(k-1)$ positions and select the best k-job sequence having the least total flow time value from among those generated. This becomes the next k-job current sequence.

The modified Step 4, now, requires a total of $\{k + (k-1)^2\}$ calculations to determine flow times, thereby giving rise to a time-complexity of $O(k(k^2 - k + 1)km)$ or $O(n^4m)$, which is the same as that required by the method of Framinan and Leisten [1].

3. Performance evaluation

Both the method of Framinan and Leisten [1] and the proposed heuristic were coded in the C programming language and run on a Pentium 4, 256 MB, 2.8 GHz PC. To compare their performance, we carried out experimentation in two phases. In the first phase, we considered small-size problems with the number of jobs, $n = 6$, 7, and 8, and the number of machines, $m = 5$, 10, 15, and 20. The second phase consisted of large-size problems with $n = 10$, 20, 30, 40, 50, 60, 70, and 80, and $m = 5$, 10, 15, and 20. Twenty instances were considered for each combination of jobs and machines. Hence, a total of 240 problems were considered in the first phase, and 640 problems in the second phase. As is typically used in the literature (see [5,19]), the processing time probability distribution follows a discrete $U(1, 99)$.

In order to compare the performance of these heuristics, we consider two measures, namely, average relative percentage deviation (ARPD) and percentage of optimal solutions based on the best-known solution for a given problem instance. For a set of problems, we define ARPD for a heuristic as follows:

$$\text{ARPD} = \frac{100}{20} \sum_{i=1}^{20} \left(\frac{\text{Heuristic}_i - \text{Best}_i}{\text{Best}_i} \right),$$

where Heuristic_i is the objective function value obtained for the ith instance by a heuristic and Best_i is the best solution value for that instance. For problems in the first phase, the best total flow time is obtained by complete enumeration (and, hence, constitutes the optimal value),

Table 1
Comparison of various heuristics for small problems ($n = 6$, 7, and 8)

n	m	No. of problems	Framinan and Leisten [1]			Proposed method		
			ARPD	Percent optimal	Average CPU time (in seconds)	ARPD	Percent optimal	Average CPU time (in seconds)
6	5	20	0.318	70	0.00015	0.134	85	0.00020
	10	20	0.642	50	0.00025	0.258	75	0.00035
	15	20	0.073	80	0.00035	0.024	90	0.00055
	20	20	0.169	70	0.00050	0.149	80	0.00070
7	5	20	0.687	45	0.00025	0.078	85	0.00040
	10	20	0.724	45	0.00045	0.405	60	0.00065
	15	20	0.456	50	0.00070	0.564	60	0.00100
	20	20	0.387	55	0.00090	0.196	75	0.00130
8	5	20	0.554	45	0.00040	0.365	60	0.00060
	10	20	0.781	45	0.00070	0.266	65	0.00110
	15	20	0.639	35	0.00110	0.550	55	0.00165
	20	20	0.341	40	0.00145	0.216	60	0.00220

Table 2
Comparison of various heuristics for large problems ($n = 10$, 20, 30, 40, 50, 60, 70, and 80)

n	m	No. of problems	Framinan and Leisten [1]			Proposed method		
			ARPD	Percent best	Average CPU time (in seconds)	ARPD	Percent best	Average CPU time (in seconds)
10	5	20	0.3673	55	0.001	0.1345	80	0.002
	10	20	0.5982	35	0.002	0.0569	80	0.003
	15	20	0.3389	70	0.003	0.4745	70	0.004
	20	20	0.2922	40	0.004	0.0590	90	0.006
20	5	20	1.3661	10	0.014	0.0899	90	0.021
	10	20	1.2024	20	0.025	0.1401	80	0.041
	15	20	0.7713	30	0.037	0.1167	70	0.063
	20	20	0.8113	5	0.047	0.0296	95	0.081
30	5	20	0.7944	40	0.06	0.1663	60	0.011
	10	20	1.1997	5	0.12	0.0046	95	0.020
	15	20	0.8213	10	0.18	0.0976	90	0.30
	20	20	0.8072	10	0.24	0.0231	90	0.41
40	5	20	0.6962	20	0.20	0.1335	80	0.34
	10	20	1.0451	20	0.36	0.0841	85	0.66
	15	20	1.2110	15	0.56	0.0637	85	0.93
	20	20	1.1152	10	0.73	0.0725	90	1.33
50	5	20	0.8103	25	0.45	0.0600	75	0.85
	10	20	1.0917	5	0.87	0.0152	95	1.61
	15	20	0.9725	10	1.30	0.1050	90	2.42
	20	20	1.1352	5	1.76	0.0190	95	3.26
60	5	20	0.6172	45	0.93	0.2138	55	1.78
	10	20	0.9417	20	1.78	0.0353	80	3.37
	15	20	1.1156	15	2.72	0.0397	85	5.05
	20	20	1.2669	0	3.60	0.0000	100	6.81
70	5	20	0.5585	15	1.72	0.0774	85	3.30
	10	20	0.6305	45	3.32	0.3517	55	6.29
	15	20	0.8713	15	4.96	0.0385	85	9.41
	20	20	1.1630	5	6.66	0.0293	95	12.70
80	5	20	0.5456	25	2.88	0.1169	75	5.65
	10	20	1.0178	5	5.62	0.0420	95	10.82
	15	20	0.9063	15	8.42	0.0427	85	16.13
	20	20	0.8994	5	11.31	0.0196	95	21.74

Table 3
Results of statistical tests (Framinan and Leisten [1] versus the proposed method)

n	m	No. of problems	Framinan and Leisten [1] versus proposed method		
			Total flow time difference		t
			Mean	Std. dev.	
6	5	30	3.13	18.20	0.942
	10	30	20.80	32.35	3.522
	15	30	8.80	25.97	1.856
	20	30	1.00	9.40	0.583
7	5	30	10.47	31.00	1.850
	10	30	20.90	37.74	3.033
	15	30	−2.07	60.28	−0.188
	20	30	13.40	41.89	1.752
8	5	30	3.57	46.18	0.423
	10	30	29.17	59.04	2.706
	15	30	9.27	87.60	0.580
	20	30	12.17	53.79	1.239
10	5	30	17.9	38.55	2.543
	10	30	40.4	74.08	2.987
	15	30	−12.0	165.47	−0.397
	20	30	43.0	84.55	2.786
20	5	30	157.83	192.62	4.488
	10	30	193.77	265.94	3.991
	15	30	207.00	272.10	4.167
	20	30	259.27	236.29	6.010
30	5	30	135.53	275.94	2.690
	10	30	464.83	391.39	6.505
	15	30	516.20	567.13	4.985
	20	30	540.47	567.30	5.218
40	5	30	268.90	432.33	3.407
	10	30	637.93	706.12	4.948
	15	30	1002.20	1064.20	5.158
	20	30	947.73	1053.50	4.927
50	5	30	450.53	559.06	4.414
	10	30	965.60	763.14	6.930
	15	30	887.57	1096.70	4.433
	20	30	1596.10	1235.60	7.075
60	5	30	276.03	838.39	1.803
	10	30	994.83	1011.40	5.387
	15	30	1740.70	1349.10	7.067
	20	30	2053.60	1469.20	7.656
70	5	30	491.77	792.58	3.398
	10	30	910.27	1804.0	2.764
	15	30	1866.20	1908.20	5.357
	20	30	2408.30	1939.90	6.800
80	5	30	636.63	1052.20	3.314
	10	30	1997.30	1643.60	6.656
	15	30	2424.20	2205.80	6.020
	20	30	2823.70	2004.30	7.716

while for the problems in the second phase, the best solution is taken as the better of the two generated by the heuristics.

Tables 1 and 2 display comparative evaluations of the proposed method and the method of Framinan and Leisten [1] for the problems in Phase 1 and Phase 2, respectively. Also, the average computing times (in seconds) required for solving every problem instance by both the heuristics are reported in Tables 1 and 2.

It is evident from the results of Table 1 that the proposed heuristic gives better results than those obtained by the method of Framinan and Leisten [1] for all problem instances. The ARPD values for the proposed heuristic range from 0.024% to 0.56%, while the number of times an optimal solution is obtained ranges from 55% to 90%, whereas the numbers in these categories for the method of Framinan and Leisten [1] are 0.073–0.78% (for ARPD), and 35–80% for the number of times an optimal solution is obtained.

As depicted in Table 2, an identical level of performance of the proposed heuristic carries over to the large-size problems as well. The ARPD for the proposed heuristic ranges from 0% to 0.47% and the number of times the best solution is obtained varies from 55% to 100% compared with the corresponding values of 0.29–1.37% and 0–70% for the method of Framinan and Leisten [1]. Clearly, the proposed heuristic outperforms the method of Framinan and Leisten [1]. Although the computational complexity of both the heuristics is the same, the average CPU time values reported in Tables 1 and 2 show that the proposed heuristic takes slightly more CPU time than that required by the method of Framinan and Leisten [1].

Next, we show statistical significance of the better results obtained by our method over those obtained by the method of Framinan and Leisten [1]. The number of problem instances for each problem is taken as 30. Thus, each problem set comprises 30 pairs of total flow time values. For each problem set, the mean and the standard deviation of the 30 differences in total flow times are computed. The difference corresponding to a problem instance is obtained by subtracting the total flow time of the proposed heuristic from that of the method of Framinan and Leisten [1]. We test the null hypothesis, H_0: $\mu = 0$. If it holds true, then, statistically, the difference between the two methods is insignificant. The t-statistic is computed as follows:

$$t = \sqrt{N}\frac{\overline{X} - \mu_0}{S},$$

where N is the sample size, $\overline{X}$ the sample mean, S the sample standard deviation, $\mu_0 = 0$, and degrees of freedom $= N - 1$ (see [20]). The critical value, c, is obtained from the relation

$$\text{Probability}\ (t > c) = \alpha = 0.05.$$

Using the standard tables of t-distribution, we obtain $c = 1.699$ for 29 degrees of freedom.

The results are presented in Table 3. Note that the proposed method is statistically better than the method of Framinan and Leisten [1] in 37 out of the 44 cases. Our method consistently performs better, especially for large-problem-size instances.

4. Concluding remarks

In this paper, we have proposed a modification of the method of Framinan and Leisten [1], for the permutation flow shop problem for the objective of minimizing the total flow time. Our modification significantly enhances the performance of this method, considered to be the best heuristic for this problem, for both small and large problem instances.

References

[1] Framinan JM, Leisten R. An efficient constructive heuristic for flowtime minimization in permutation flow shops. Omega 2003;31:311–7.
[2] Baker KR. Introduction to sequencing and scheduling. New York: Wiley; 1974.
[3] Gonzalez T, Sahni S. Flow shop and job shop scheduling: complexity and approximation. Operations Research 1978;26:36–52.
[4] Rajendran C, Chaudhuri D. An efficient heuristic approach to the scheduling of jobs in a flow shop. European Journal of Operational Research 1991;61:318–25.
[5] Rajendran C. Heuristic algorithm for scheduling in flow shop to minimize total flow time. International Journal of Production Economics 1993;29:65–73.
[6] Rajendran C, Ziegler H. An efficient heuristic for scheduling in a flow shop to minimize total weighted flowtime of jobs. European Journal of Operational Research 1997;103:129–38.
[7] Woo DS, Yim HS. A heuristic algorithm for mean flow time objective in flow shop scheduling. Computers and Operations Research 1998;25:175–82.
[8] Liu J, Reeves CR. Constructive and composite heuristics solution of the $P\|\sum C_i$ scheduling problem. European Journal of Operational Research 2001;132:439–52.
[9] Kalczynski PJ, Kamburowski J. On the NEH heuristic for minimizing the makespan in permutation flow shops. Omega 2007;35:53–60.
[10] Kalir AA, Sarin SC. A near-optimal heuristic for the sequencing problem in multiple-batch flow-shops with small equal sublots. Omega 2001;29:577–84.
[11] Sarin S, Lefoka M. Scheduling heuristic for the n-job, m-machine flow shop. Omega 1993;21:229–34.
[12] Osman IH, Potts CN. Simulated annealing for permutation flow-shop scheduling. Omega 1989;17:551–7.

[13] Rad SF, Ruiz R, Baroojerdian N. New high performing heuristics for minimizing makespan in permutation flowshops. Omega 2009; 37, in press.
[14] Simons JV. Heuristics in flow shop scheduling with sequence dependent setup times. Omega 1992;20:215–25.
[15] Huq F, Cutright K, Martin C. Employee scheduling and makespan minimization in a flow shop with multiprocessor work stations: a case study. Omega 2004;32:121–9.
[16] Li X, Wang Q, Wu C. Efficient composite heuristics for total flowtime minimization in permutation flow shops. Omega 2009; 37, in press.
[17] Allahverdi A, Aldowaisan T. New heuristics to minimize total completion time in *m*-machine flow shop. International Journal of Production Economics 2002;7:71–83.
[18] Framinan J, Leisten M, Ruiz-Usano R. Comparison of heuristics for flow time minimization in permutation flow shop. Computers and Operations Research 2005;32:1237–54.
[19] Nawaz ME, Enscore E, Ham I. A heuristic algorithm for the *m*-machine, *n*-job flow shop sequencing problem. Omega 1983;11:91–5.
[20] Kreyszig E. Advanced engineering mathematics. New York: Wiley; 1972.