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),