An O(n^4m) algorithm to minimize the general flow shop scheduling problem

SAWAT PARARACH

Department of Industrial Engineering, Thammasat University,THAILAND

psawattu@hotmail.com

Abstract

On a general shop scheduling problem, n jobs are arranged on m machines in the series and must follow the same routing. In this paper, we investigate the objective of minimizing the total flow time or the differ between completion times and release times and the makespan minimization. The flowtime and makespan minimization have been proved to be NP-hard, therefore we consider the polynomial-time heuristic for the time-complexity of O(n4m) to obtain the solution. From experimental results of 640 instances, the proposed procedure has better in the flowtime and makespan values than the well-known method while remaining its time complexity.

Introduction

A general flow shop problem is a multi-operation processing involves a set of *n* jobs and *m* machines by all the jobs must follow the same routing and machines are arrange in series. For each machine we know the order to perform it and its processing time. The obtained solution is the schedule that minimized the total flow time of all jobs. The total flow time is not makespan objective, but it represents the differ between completion and release time for each job. The advantage of total flow time objective is minimization of work-in-process, but not restricted to none intermediate storage inventory. In the case of the makespan objective, it considers the routing that minimizing the completion time of the last job in the last machine, but not determine the waiting time for each job in the process. The sequence that minimizing total flow time may be not the same as the minimized-makespan sequence.

Solution for turn-around of n jobs in flow shop scheduling rely on a set of restrictive assumptions as follows[1]:

-Single part or batch of parts are always treated as a single job.
-Preemption and job cancellation is not allowed.
-Processing times are independent of the schedule.
-Work-in-process is allowed
-Machines are able to process one job at a time.
-Each job visits all machines exactly once.
-Machines are always available and the only resource modeled.
-Jobs are all known in advance.
-The scheduling is purely deterministic.

Flow shop scheduling have been proved to be NP-hard[2]. In several years, many heuristics for solving these problem have been considered. Averbakh[3] studies the flow-shop problem with 2 jobs and *m* machines and uncertainly interval processing times of operations. Soukhal et.al[4] investigate two-machine flow shop scheduling problems taking transportation into account. Koulamas and Kypasiris[5] study the two-stage assembly flow shop scheduling with concurrent operations in the first stage and a single assembly operation in the second stage. Yokoyama[6] considers a flow shop scheduling model with partitions machining ,setup and assembly operations into blocks. On the improved-heuristics for total flow time minimization, Agarwal et al. [7] develop the non-polynomial time heuristic based on the adaptive learning approach. For polynomial time methods, Framinan and Leisten[8] develop a constructive heuristic based on pairwise interchanges approach, Laha and Sarin[9] improve its performance by node-insertion procedures.
The proposed procedure in this paper is improving the method of Laha and Sarin[8] while not affecting its time-

complexity of $O(n^4m)$ algorithm. The investigation presents in Section 2. Results of the experimentation are demonstrated in Section 3. Finally, concluding remarks are made in Section 4.

The $O(n^4m)$ polynomial time algorithm

From the literature, the effective heuristic has been solves many general flow shop instances in a time-complexity of $O(n^4m)$ is considered by Laha and Sarin[9]. This concept based on node insertions that modified from the pairwise interchanges of Framinan and Leisten[8]. The steps of this concept can be given as:

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: 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.

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

From this concept, the modification in this research performs by adding the step as Step 6. The Step 6 determines the pairwise interchanges on the n-job sequence obtaind from STOP mode in Step 5 by interchanging jobs in position i and j for all i,j ,$1\leq i \leq n$, $i < j \leq n$. Select the best sequence obtained among the $n(n-1)/2$ sequences having minimum total flow time.

The examples of the step of the method of Laha and Sarin and the proposed procedure in Step 6 can be shown in figure 1-5.

Note that Step 6 dictates the time-complexity of $O(nm)$ for the schedule of n jobs on m machine , multiplied the $O(nm)$ of the schedule to the $O(n2)$ operations of changing the sequence, the overall executed time in Step 6 is $O(n3m)$.

From the working paper[9], Step 1-5 perform $O(n4m)$ operations.

By Step 6 adding to Step 1-5, the procedure performs $O(n4m)+ O(n3m) \approx O(n4m)$ operations , since the proposed procedure in Step 6 is not increasing the time-

Figure 1. The example of sorting jobs in step 1 and 2.

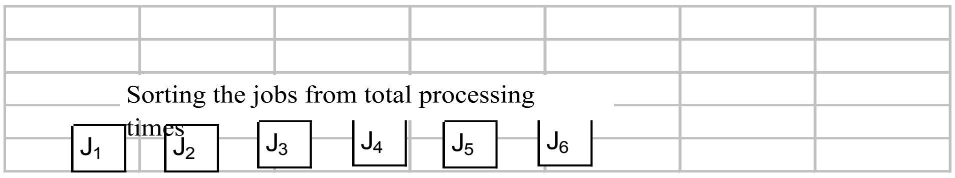


Figure 2. The example of selecting the starting sequence in the step 3

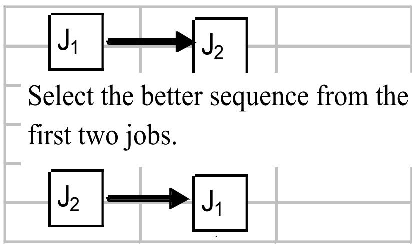
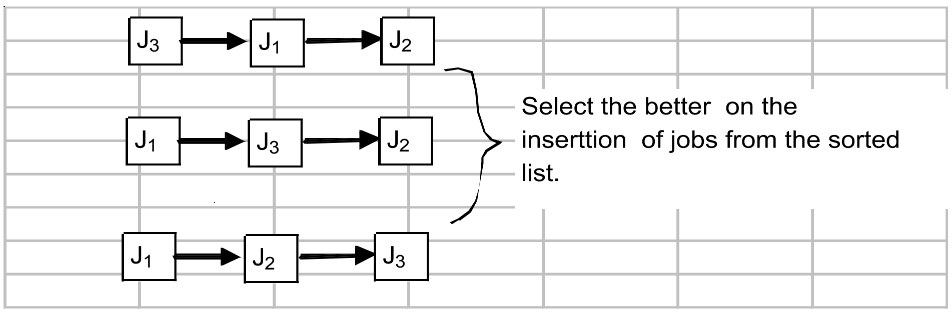


Figure 3. The example of selecting job into the current sequence.



J_3
J_1
J_2
J_1
J_3
J_2
J_1
J_2
J_3
Select the better on the insserttion of jobs from the sorted list.

Figure 4. The example of sequencing in Step 4.

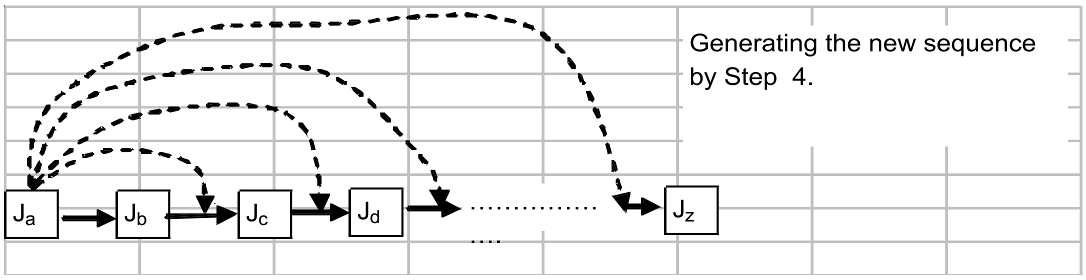
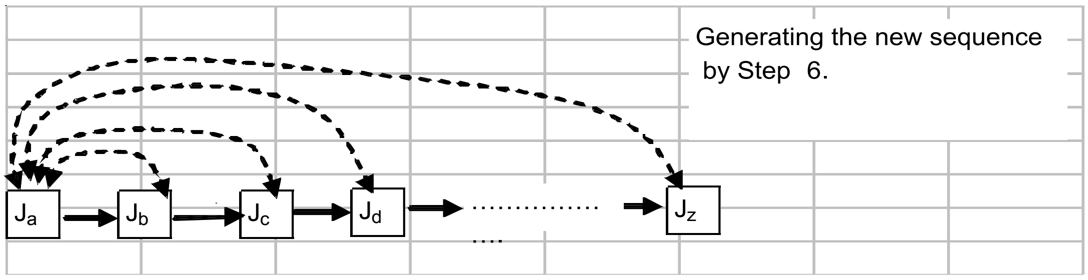


Figure 5. The example of pairwise interchanges in Step 6.



complexity of Laha and Sarin method and advances in searching the minimized total flow time value.

Experimental Results

The experimentation have been carried out in 640 instances with n=10, 20, 30, 40, 50, 60, 70 and 80, and m=5, 10, 15, and 20, and the replication is 20 for each combination of jobs and machines[9]. For the generated random processing times, it follows a discrete uniform distribution between 1 and 99.

The computer programs of the proposed procedure and the method of Laha and Sarin are coded in C++ language and run on a Pentium4, 256 MB, 2.4GHz PC.

Average relative percentage deviation (ARPD) is considered to compare the performance of these method, it defines as[9]:

$$\text{ARPD} = \frac{100}{20}\sum_{i=1}^{20}\frac{A_i - B_i}{B_i}$$

For the ith instance, Ai is the total flow time value obtained from the Laha and Sarin method and Bi is the obtained from the proposed procedure.The solution reports are demonstrated in Table1.

For all instances, the objective of minimizing total flow time and minimizing makespan can be applied to the code program.

From the results, it is evident that the proposed procedure gives solution values better than the obtained from the Laha and Sarin method both the objective of minimizing total flow time and minimizing makespan. The better results from the objective of minimizing total flow time and minimizing makespan have been shown as ARPD in table 1 and 2 , figure 6 and 7 respectively.
Due to the computing times of the proposed procedure, it takes more CPU times than the Laha and Sarin method, but not greater than 0.1 second.

Concluding remarks

The proposed procedure by adding Step 6. into the method of Laha and Sarin enhances the performance measurement of total flow time and makespan value, it represents the time-complexity of *O(n4m)*. The pairwise interchanges in Step 6 are not taken the computer times greater than 0.1 seconds for 640 instances and improved solution from the obtained value by Laha and Sarin method. The proposed procedure is the suitable one for solving the general flow shop scheduling problem in polynomial time.

Table 1. Comparison of the proposed methods for the objective of minimizing total flow time.

N	m	no.instances	ARPD
10	5	1 to 20	0.982219
10	10	21 to 40	0.676018
10	15	41 to 60	0.633287
10	20	61 to 80	0.535631
20	5	81 to 100	1.042264
20	10	101 to 120	0.624009
20	15	121 to 140	0.669467
20	20	141 to 160	0.567634
30	5	161 to 180	0.766617
30	10	181 to 200	0.54204
30	15	201 to 220	0.449736
30	20	221 to 240	0.447026
40	5	241 to 260	0.662283
40	10	261 to 280	0.559147
40	15	281 to 300	0.359461
40	20	301 to 320	0.40412
50	5	321 to 340	0.565697
50	10	341 to 360	0.566107
50	15	361 to 380	0.374524
50	20	381 to 400	0.384409
60	5	401 to 420	0.678612
60	10	421 to 440	0.428656
60	15	441 to 460	0.386263
60	20	461 to 480	0.336179
70	5	481 to 500	0.477944
70	10	501 to 520	0.357064
70	15	521 to 540	0.391865
70	20	541 to 560	0.403442
80	5	561 to 580	0.606884

80	10	581 to 600	0.414618
80	15	601 to 620	0.39081
80	20	621 to 640	0.315106

Figure 6. ARPD analysis for the objective of minimizing total flow time.

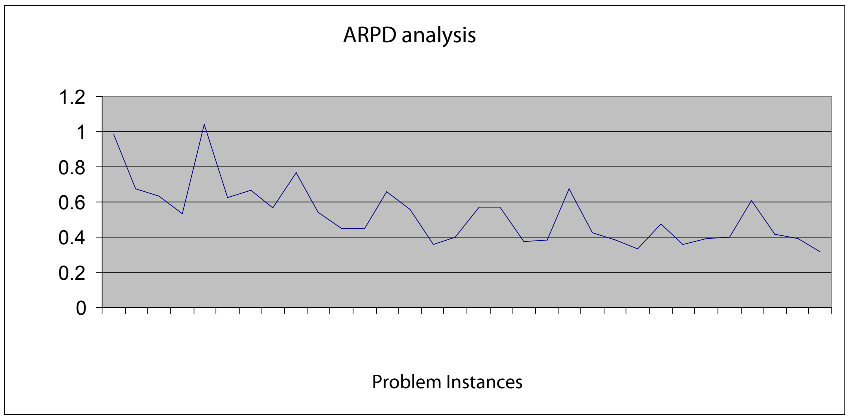
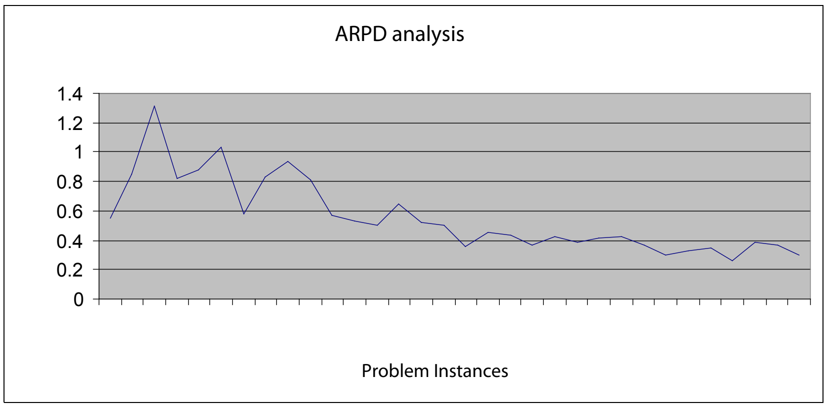


Table 2. Comparison of the proposed methods for the objective of minimizing makespan.

N	m	no.instances	ARPD
10	5	1 to 20	0.550132
10	10	21 to 40	0.851801
10	15	41 to 60	1.311459
10	20	61 to 80	0.823365
20	5	81 to 100	0.874063
20	10	101 to 120	1.037284
20	15	121 to 140	0.575118
20	20	141 to 160	0.830651
30	5	161 to 180	0.934585
30	10	181 to 200	0.806254
30	15	201 to 220	0.565922
30	20	221 to 240	0.527077
40	5	241 to 260	0.503105
40	10	261 to 280	0.650253
40	15	281 to 300	0.521265
40	20	301 to 320	0.497326
50	5	321 to 340	0.359122
50	10	341 to 360	0.456082
50	15	361 to 380	0.432683
50	20	381 to 400	0.368986
60	5	401 to 420	0.421359
60	10	421 to 440	0.38288
60	15	441 to 460	0.412617
60	20	461 to 480	0.425646
70	5	481 to 500	0.371372
70	10	501 to 520	0.302847

70	15	521 to 540	0.330696
70	20	541 to 560	0.347304
80	5	561 to 580	0.260466
80	10	581 to 600	0.384264
80	15	601 to 620	0.37028
80	20	621 to 640	0.301534

Figure 7. ARPD analysis for the objective of minimizing makespan.



References

01. Brandimarte P. and Villa A., (1992), Advanced Models for the Manufacturing Systems Management, U.S.A: CRC Press.

02. Gonzalez T, and Sahni S.,(1978), Flow shop and job shop scheduling: complexity and Approximation, Operations Research, 26, 36-52.

03. Averbakh I., (2006), The minmax regret permutation flow shop problem with two jobs, European Journal of Operational Research, 169, 761-766.

04. Soukhal A., Oulamara A. and Martineau P., (2005), Complexity of flow shop scheduling problems with transportation constraints, European Journal of Operational Research 2005, 161, 32-41.

05. Koulamas C. and Kyparisis G., (2007), A note on the two-stage assembly flow shop scheduling problem with uniform parallel machines, European Journal of Operational Research, 182, 945-951.

06. Yokoyama M., (2008), Flow-shop scheduling with setup and assembly operations.

07. European Journal of Operational Research , 187, 1184-1195. Agarwal A., Colak S., and Eryasoys E. Improvement heuristic for the flow-shop scheduling problem: An adaptive learning approach. European Journal of Operational Research,169, 801-815.

08. Framinan J.M., and Leisten R. An efficient constructive heuristic for flowtime minimization in permutation flowshops. Omega 2003, 31, 311-317.

09. Laha D. and Sarin S.C. A heuristic to minimize total flow time in permutation flow shop. Omega, 37, 734-739.