

Minimum Deviation Algorithm for Two-Stage No-Wait Flowshops with Parallel Machines^{*}

JINXING XIE, WENXUN XING, ZHIXIN LIU AND JIEFANG DONG

Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China
jxie@math.tsinghua.edu.cn

Abstract: The scheduling problems studied in this paper concern the two-stage no-wait flowshops with parallel machines under the objective function of the minimization of the maximum completion time. A new heuristic algorithm, i.e. the minimum deviation algorithm, is developed to solve the problems. In order to evaluate the average case performance of the algorithm, we design numerical experiments to compare the effectiveness of the algorithm with that of the other approximation algorithms. Extensive simulations are conducted under different shop conditions, and the results statistically show that the minimum deviation algorithm performs well under most of the situations.

Keywords: Scheduling, No-wait, Flowshop, Parallel machine, Heuristic

Corresponding author and address:

Dr. Jinxing Xie
Department of Mathematical Sciences
Tsinghua University
Beijing 100084, China
E-mail: jxie@math.tsinghua.edu.cn
Tel.: (8610) 6278 7812
Fax : (8610) 6278 1785

^{*} This research has been supported by The National Natural Science Foundation of China.

Minimum Deviation Algorithm for Two-Stage No-Wait Flowshops with Parallel Machines

Abstract: The scheduling problems studied in this paper concern the two-stage no-wait flowshops with parallel machines under the objective function of the minimization of the maximum completion time. A new heuristic algorithm, i.e. the minimum deviation algorithm, is developed to solve the problems. In order to evaluate the average case performance of the algorithm, we design numerical experiments to compare the effectiveness of the algorithm with that of the other approximation algorithms. Extensive simulations are conducted under different shop conditions, and the results statistically show that the minimum deviation algorithm performs well under most of the situations.

Keywords: Scheduling, No-wait, Flowshop, Parallel machine, Heuristic

I. INTRODUCTION

This paper deals with the two-stage no-wait flowshop scheduling problems with parallel machines. The objective function is the minimization of the maximum completion time C_{max} (also called the makespan, or the schedule length).

No-wait flowshop scheduling problems arise in situations where there must be no delay between the time a product finishes processing on one stage and the time at which it begins processing on the immediately followed stage. This kind of no-wait environment often occurs in the steel production, just-in-time manufacturing process, chemical industry, and service industry, etc. For example, in food processing industry, the canning operation must follow cooking operation with no delay to ensure freshness. Since the no-wait scheduling problems have a variety of industrial applications, they have drawn great interests of both the academicians and the practitioners. For example, the first paper dealing with the no-wait scheduling can be

dated back to 1970s [1-2]. A comprehensive review on the machine scheduling problems with no-wait in processes can be found in [3].

The flowshop scheduling problems with parallel machines are also called flexible flowshop scheduling or hybrid flowshop scheduling, and have been studied by many scholars (for example, see [4-9]). However, the flowshop scheduling problems with parallel machines under a no-wait constraint has received relatively less attention. With the makespan objective, the two-stage flowshop scheduling problem with only one machine at each stage is polynomially solvable, no matter whether there are no-wait constraints or not [10-11]. However, in the real world applications of the no-wait scheduling problems, there are usually many parallel machines available in a stage [12]. When parallel machines are taken into account, the problems become more difficult. The two-stage no-wait flowshop scheduling problems with parallel machines are also NP-hard in the strong sense [13]. Therefore all exact algorithms for even the simple flowshop and simple parallel machines will most likely have running times that increase exponentially with the problem size.

The problem considered here is modeled as follows. A set of n jobs $J = \{J_1, J_2, \dots, J_n\}$ is to be processed. Each job requires two processing operations, which must be processed in two consecutive stages Z_1 and Z_2 with no delay. Stage Z_1 has m_1 identical machines and stage Z_2 has m_2 identical machines. The first and the second operation of a job J_j ($j=1, 2, \dots, n$) must be processed on stage Z_1 and Z_2 for the time p_{1j} and p_{2j} , respectively. The problem can be denoted by $F2(m_1, m_2)/\text{no-wait}/C_{max}$. When the numbers of the parallel machines in the stages are not input parameters, the problem can be denoted by $F2(P)/\text{no-wait}/C_{max}$.

Sriskandarajah [14] established a tight worst case bound of $3-1/m$ using an arbitrary sequence for the list scheduling algorithm when there is only one machine in one of the two stages and there are m machines in the other. This bound for the list scheduling algorithm can be improved to 2 when the jobs are ordered according to the jobs' processing times. For the problem with m identical machines at both stages,

he also described a heuristic with the worst case bound of $3-1/m$.

In the next section, we will propose a new heuristic algorithm, i.e. the minimum deviation algorithm, to solve the problem. In order to evaluate the average case performance of the algorithm, we design numerical experiments to compare the effectiveness of the algorithm with some other heuristics.

II. MINIMUM DEVIATION ALGORITHM

We first outline the basic two-step approach used in the heuristics for the general network flowshop. The framework for these heuristics consists of the following two steps: sequence generation and sequence evaluation. In the sequence generation step, the algorithm calculates a processing sequence for the jobs to be processed. This sequence is actually a priority list of the processing order for the jobs in the final schedule. In the sequence evaluation step, each job in the priority list is assigned to a machine to process and its completion time is calculated, governing the no-wait constraint and all other flowshop scheduling constraints.

Most of the existing heuristics in the literature follow these two steps to design the heuristics. In the two-stage no-wait flowshop scheduling problems with parallel machines, the sequence evaluation step is quite simple. When a priority list is given, one can easily assign the jobs to the machines using the strategy of ‘one job at a time’. The heuristics differ only in the sequence generation step, which result in different performance of the heuristics.

The heuristic put forward in this paper uses a different logic to develop the schedule. This algorithm generates and evaluates the sequence simultaneously, *i.e.*, the sequence is obtained in a dynamic manner. The basic idea of the algorithm is to reduce the idle time of the machines due to no-wait constraints when we arrange the jobs to process on the machines. We name this approach as LDA (least deviation algorithm) [15].

Denote the m_1 parallel machines in the first center Z_1 by $\{M_{1,1}, M_{1,2}, \dots, M_{1,m_1}\}$ and the m_2 parallel machines in the second center Z_2 by $\{M_{2,1}, M_{2,2}, \dots, M_{2,m_2}\}$. LDA is a greedy algorithm, *i.e.*, whenever a job is scheduled on machines and a partial schedule is obtained, this partial schedule will never be changed anymore. When the algorithm starts to run (assume the time be zero), all the machines in both machine centers are considered to be idle. After a partial schedule is already obtained, we should make our decision on which job should be processed next. Let M_{1,k_1} be the machine which is the first idle machine in Z_1 from now on, and let t_1 be the time point when M_{1,k_1} becomes idle. Similarly, let M_{2,k_2} be the machine which is the first idle machine in Z_2 from now on, and let t_2 be the time point when M_{2,k_2} becomes idle. Then we find an unprocessed job J_j with p_{1j} , the processing time of J_j on machine center Z_1 , being closest to $\max(0, t_2 - t_1)$. In order to reduce the idle time of the machines, J_j is processed next, on machine M_{1,k_1} in the first center and then on machine M_{2,k_2} in the second center. That is to say, if $p_{1j} \geq t_2 - t_1$, job J_j starts to process on machine M_{1,k_1} immediately; otherwise J_j has to start to process on machine M_{1,k_1} after $(t_2 - t_1 - p_{1j})$ waiting time. This way we generated a new partial schedule with one more job arranged to machines to process.

Specifically, the algorithm LDA can be described as follows.

Step 0. Set $a_i=0$ ($i=1, 2, \dots, m_1$) and $b_i=0$ ($i=1, 2, \dots, m_2$). Set $K=\{1, 2, \dots, n\}$ and the list L to empty.

Step 1. Compute $t_1 = \min\{a_i\}$ and $k_1 = \operatorname{argmin}\{a_i\}$ (break ties arbitrarily). Compute $t_2 = \min\{b_i\}$ and $k_2 = \operatorname{argmin}\{b_i\}$ (break ties arbitrarily). Let $t = \max(0, t_2 - t_1)$.

Step 2. Find $j = \arg \min\{p_{1j} - t \mid j \in K\}$ (break ties arbitrarily). Set $K = K \setminus \{j\}$ and add j to the list L .

Step 3. If $p_{1j} \geq t$, set $a_{k_1} = a_{k_1} + p_{1j}$ and then set $b_{k_2} = a_{k_1} + p_{2j}$. Otherwise, set $a_{k_1} = b_{k_2}$ and $b_{k_2} = b_{k_2} + p_{2j}$.

Step 4. If $K = \emptyset$, stop. Otherwise, go to Step 2.

In the algorithm LDA presented above, a_i is the time point when the machine M_{1i} becomes idle ($i=1, 2, \dots, m_1$) and b_i is the time point when the machine M_{2i} becomes idle ($i=1, 2, \dots, m_2$); L represents the ordered processing list of the jobs and K represents the indices set of the jobs not scheduled any more.

Let's see an example with $n=8$, $m_1 = 2$, $m_2 = 5$. The processing times of the jobs are listed in Table 1. The corresponding schedule generated by LDA for the example is shown in Figure 1. In the figure, the first number in a rectangle stands for the processing time, and the second number (in brackets) stands for the index of the corresponding job. Rectangles with shadows in the figure mean the machines are waiting to process jobs. When the algorithm begins to run, $t_1 = 0, t_2 = 0, t = 0$, thus job 3 should be processed first. Assume that this job is assigned to M_{11} and M_{21} . In the second iteration, we still have $t_1 = 0, t_2 = 0, t = 0$, thus job 8 should be processed next and we can assign it to M_{12} and M_{22} . In the third iteration, $t_1 = 3, t_2 = 10, t = 7$, thus job 5 should be processed next and we can assign it to M_{11} and M_{22} . In the fourth iteration, $t_1 = 4, t_2 = 19, t = 15$, thus job 2 should be processed next and we can assign it to M_{12} and M_{22} . In the fifth iteration, $t_1 = 10, t_2 = 23, t = 13$, thus job 6 should be processed next and we can assign it to M_{11} and M_{21} . In the sixth iteration, $t_1 = 19, t_2 = 23, t = 4$, thus job 4 should be processed next and we can assign it to M_{12} and M_{22} . In the seventh

iteration, $t_1 = 23, t_2 = 32, t = 9$, thus job 7 should be processed next and we can assign it to M_{11} and M_{21} . In the last iteration, $t_1 = 32, t_2 = 45, t = 23$, only job 2 should be processed next and we can assign it to M_{12} and M_{21} . Finally we get the schedule with $C_{\max} = 54$ (cf. Figure 1).

(*****insert Table 1 about here*****)

(*****insert Figure 1 about here*****)

III. NUMERICAL EXPERIMENTS

In order to evaluate the average case performance of the LDA algorithm, we design numerical experiments to compare the effectiveness of the algorithm with that of the other heuristics under different shop conditions.

3.1 Independent Variables

Seven independent variables are included in the numerical experiments. They are listed in Table 2 and will be described in detail below.

(*****insert Table 2 about here*****)

1. Number of the machines (NM): Since we think the performance of the algorithm might be influenced by the number of machines, we test two types of the combinations of the machines. Under $NM = S$ (small number of machines), we set $m_1=3$ and $m_2=4$; Under $NM = L$ (large number of machines), we set $m_1=8$ and $m_2=10$.

2. Number of the jobs (n): Similarly to the number of machines, we test two types of the number of the jobs. In this paper we will test $n=10(m_1+m_2)$ for the small number of jobs and $n=100(m_1+m_2)$ for the large number of jobs.

3. Distribution of the processing times (DT): we test two types of distributions, *i.e.*, the normal distribution (ND) and the uniform distribution (UD).

4. Average processing times (\mathbf{m}): $\mathbf{m}=50$, $\mathbf{m}=500$ respectively.

5. Variation of processing times (\mathbf{s}): $\mathbf{s}=0.1$, $\mathbf{s}=0.3$ respectively.

6. The relationship of the processing times and the number of machines (RP). We investigate two cases: RP = UR means the processing times of the jobs is unrelated to the number of machines; RP = PR means the processing times of the jobs is proportional to the number of machines. Specifically, when the processing times are normally distributed (DT=ND), we set the processing time

$$p_{ij} = \begin{cases} \mathbf{m}(1 + \mathbf{s}N(0,1)) \times m_i, (i = 1,2), & \text{when RP = PR;} \\ \mathbf{m}(1 + \mathbf{s}N(0,1)), (i = 1,2), & \text{when RP = UR;} \end{cases}$$

where $N(0,1)$ means a random number with standard normal distribution. In order to avoid negative processing times, we ignore the random number which results in a negative processing time.

However, when the processing times are uniformly distributed (DT=UD), we set the processing time

$$p_{ij} = \begin{cases} \mathbf{m}(1 + 2\sqrt{3}\mathbf{s}U(-\frac{1}{2}, \frac{1}{2})) \times m_i, (i = 1,2), & \text{when RP = PR;} \\ \mathbf{m}(1 + 2\sqrt{3}\mathbf{s}U(-\frac{1}{2}, \frac{1}{2})), (i = 1,2), & \text{when RP = UR;} \end{cases}$$

where $U(-0.5,0.5)$ means a random number uniformly distributed in the interval $[-0.5,0.5]$.

Please note that under this kind of design, we can make sure that the variances of the processing times under both normal and uniform distributions are the same.

7. Algorithm (H): Besides LDA algorithm, we test several other heuristic algorithms. The first heuristic we tested in this paper is the H_g algorithm proposed by

Sriskandarajah [14], where the problem is partitioned into m_1 flowshop problems with parallel machines, each having two machine centers with the first center having exactly one machine and the second center having at most $\lceil m_2/m_1 \rceil$ parallel machines. The jobs (with job j having the modified processing time of $p_{1j}+p_{2j}$) are allocated to the sub-flowshops using the algorithm due to Hochbaum and Shmoys [16], where the control parameter for the error bound is set at $1/6$ at this paper. This algorithm is labeled as H_1 in this paper. The second heuristic labeled as H_2 we tested in this paper is a simple version of the H_g algorithm, where the jobs are allocated to the sub-flowshops using the simple LPT (Largest Processing Time) rule, instead of using the algorithm due to Hochbaum and Shmoys [16]. The third heuristic labeled as H_3 we tested in this paper is a list scheduling heuristic with the list of the jobs being generated by the classical Johnson's rule [10]. The fourth heuristic labeled as H_4 we tested in this paper is a simple variation of H_3 , where the classical Johnson's rule [10] is used to the processing times of the jobs that are modified by considering the numbers of the parallel machines, i.e. for job j , p_{1j} is modified to p_{1j}/m_1 and p_{2j} is modified to p_{2j}/m_2 . The last heuristic labeled as H_5 is the LDA algorithm proposed in this paper.

3.2 Dependent Variable

Use f_I^H to stand for the makespan got by an algorithm H for an instance I , and use f_I^o to stand for a lower bound of the optimal makespan for the instance I . In this paper, we use the following formula to calculate the lower bound:

$$f_I^o = \max \left\{ \left(\frac{1}{m_1} \sum_{j=1}^n p_{1j} + \min_j p_{2j} \right), \left(\frac{1}{m_2} \sum_{j=1}^n p_{2j} + \min_j p_{1j} \right) \right\}$$

Then we can use $R = \frac{f_I^H}{f_I^o}$ as the performance measure of the algorithm. This

is the only dependent variable in our paper.

3.3 Simulation Process

For each combination of the independent variables, the simulation runs with ten replications to eliminate the effects of the random variations. Extensive simulations are conducted under different shop conditions, and the output from the simulation experiments (with totally $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 5 \times 10 = 3200$ observations) was analyzed using the SAS (Statistical Analysis Software) analysis of variance (ANOVA) procedure. The results are presented in the next section.

3.4 Simulation Results

Selected ANOVA results are presented in Table 3. In order to save space, only the results with the R-square being larger than 0.01 are shown in the table. Examinations of the table show that the performance measure is significantly influenced by the factors RP, H, \mathbf{s} and n . It is also shown that the interactions RP*H and \mathbf{s} *H are significant, which mean that the performance of the algorithm (H) is significantly influenced by RP and \mathbf{s} . The other factors (the number of machines, the distribution of the processing times, and the average processing time) seem impact the performance measure to a less extent.

(Insert Tables 3 about here)

According to the ANOVA results, it should be valuable to compare the performance of the heuristics under different combinations of the independent variables RP and \mathbf{s} . These results are presented in Table 4, and they statistically

show that the LDA heuristic (H_5) significantly outperforms the other algorithms at most of the situations.

(Insert Tables 4 about here)

From the table, we can easily see that the performance ratios of the algorithms under $RP=UR$ (the processing times of the jobs are independent of the number of the machines in the corresponding machine centers) are higher than those under $RP=PR$ (the processing times of the jobs are proportional to the number of the machines in the corresponding machine centers). The performance ratios of the algorithms also increase when s (variation of the processing times of the jobs) increases. These observations show that it is more difficult to obtain a good near-optimal schedule for the scheduling problem when the variation of the processing times of the jobs are relatively large and the processing times of the jobs are independent of the number of the machines in the corresponding machine centers. However, even under this case, the average performance ratio of the LDA heuristic is still less than 106%, i.e., the makespan of the schedule generated by the heuristic is within 6% of the optimal makespan.

IV. CONCLUSION

This paper studied the two-stage no-wait flowshop scheduling problems with parallel machines under the objective function of the minimization of the maximum

completion time. A new heuristic algorithm, i.e. the minimum deviation algorithm, is developed to solve the problems. We designed numerical experiments to evaluate the effectiveness of the algorithm. Extensive simulations are conducted under different shop conditions, and the results statistically show that the minimum deviation algorithm significantly outperforms the other algorithms at most of the situations.

REFERENCES

- [1] T.S. Arthanari and K.G. Ramaurthy, An extension of two machines sequencing problem, *Opsearch* **41**, 641~648, (1971).
- [2] T.S. Arthanari, *On Some Problems of Sequencing Grouping*, PhD thesis, Indian Statistical Institute, Calcutta, India, (1974).
- [3] N.G. Hall and C. Sriskandarajah, A Survey of Machine Scheduling Problems with Blocking and No-wait in Process, *Operations Research* **44**, 510-525, (1996).
- [4] B.S. Mittal and P.C. Nagga, Two machines sequencing problem with parallel machines, *Opsearch*, **10**, 50-61, (1973).
- [5] R.J. Wittock, An adaptable scheduling algorithm for flexible flow lines, *Operations Research* **36**, 445-453, (1988).
- [6] J.N.D. Gupta, Two stage hybrid flow shop scheduling problem, *Journal of the Operational Research Society* **38**, 359-364, (1988).
- [7] C. Sriskandarajah and S.P. Sethi, Scheduling algorithms for flexible flow shops: Worst and average case performance, *European Journal of Operational Research* **43**, 143-160, (1989).
- [8] S. Li, A hybrid two-stage flow shop with part family, batch production, major and minor set-ups, *European Journal of Operational Research* **102**, 146-156, (1997).

- [9] L. Richard and W. Zhang, Hybrid flow shop scheduling - a survey, *Computers and Industrial Engineering* **37**, 57-61, (1999).
- [10] S.M. Johnson, Optimal two and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly* **1**, 61-68, (1954).
- [11] P. Gilmore and R. Gomory, Sequencing a one-state variable machine: A solvable case of the traveling salesman problem, *Operations Research* **12**, 665-679, (1964).
- [12] J.R. Callahan, *The No-Wait Delay Problem in the Production of Steel*, PhD Thesis, Department of Industrial Engineering, University of Toronto, Canada, (1971).
- [13] C. Sriskandarajah and P. Ladet, Some no-wait shops scheduling problems: Complexity Aspects, *European Journal of Operational Research* **24**, 424-445, (1986).
- [14] C. Sriskandarajah, Performance of scheduling algorithms for no-wait flow shop with parallel machines, *European Journal of Operational Research* **70**, 365-378, (1993).
- [15] Z. Liu, J. Xie, J. Li, J. Dong, A heuristic for two-stage no-wait hybrid flowshop scheduling with a single machine in either stage, *Tsinghua Science and Technology* **8**, 43-48, (2003).
- [16] D.S. Hochbaum, D.B. Shomys, Using dual approximation algorithms for scheduling problems: Theoretical and practical results, *Journal of the Association for Computing Machinery* **34**, 144-162, (1987).

Table 1. The processing times of the jobs for the example

j	1	2	3	4	5	6	7	8
p_{1j}	10	12	3	5	6	11	9	4
p_{2j}	7	4	20	30	9	9	13	6

Table 2. Independent variables

No.	Factors	# of Levels	levels
1	NM (Number of machines)	2	S - $m_1=3, m_2=4$ L - $m_1=8, m_2=10$
2	n (Number of jobs)	2	$n=10(m_1+m_2), n=100(m_1+m_2)$ respectively
3	DT (Distribution of processing times)	2	ND - Normal distribution UD - Uniform distribution
4	m (Average processing time)	2	$m=50, 500$ respectively
5	s (Variance of processing time)	2	$s=0.1, 0.3$ respectively
6	RP (Relationship between p_{ij} and m_i)	2	UR - Unrelated PR - Proportional related
7	H (Algorithm)	5	H ₁ - Partition method H ₂ - Partition with LPT H ₃ - Johnson's list H ₄ - Modified Johnson's list H ₅ - LDA (this paper)

Table 3. Selected ANOVA results

Source	F Value	Pr > F	R-square
RP	63241	<.0001	0.4012
H	9775.48	<.0001	0.2480
RP*H	8922.81	<.0001	0.2264
s	6050.19	<.0001	0.0384
s *H	765.39	<.0001	0.0194
<i>n</i>	2435.2	<.0001	0.0154

Table 4. Average performance of the heuristics under different conditions

RP	s	H ₁	H ₂	H ₃	H ₄	H ₅
UR	0.1	1.311	1.299	1.071	1.081	1.037
UR	0.3	1.325	1.311	1.153	1.193	1.056
PR	0.1	1.029	1.023	1.015	1.023	1.018
PR	0.3	1.062	1.049	1.065	1.101	1.032

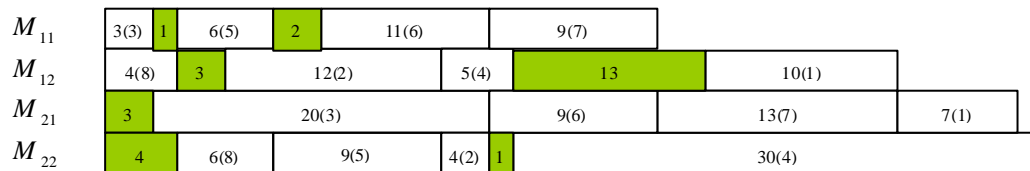


Fig. 1 The schedule generated by LDA for the example