Genetic Algorithm with Direction Selection for the Hybrid Flow Shop Scheduling Problem with makespan minimization

Zhixiong Su[1, a *], Junmin Yi[1, b]

[1]School of Management, Xiamen University of Technology, Xiamen 361024, China

[a]z.su@163.com, [b]yijunmin@xmut.edu.cn

Keywords: production scheduling; hybrid flow shop; genetic algorithm; active schedule; reversibility

Abstract. The regular hybrid flow shop scheduling problem, with *m* stages, identical parallel machines and makespan minimization, is investigated and served as the natural starting point for the hybrid flow shop scheduling problems. Some properties of this scheduling problem were discussed, and they lead to the theoretical foundation and instruction for algorithm design. Based on the active scheduling technique and reversibility property, a genetic algorithm with direction selection combined with extended Giffler & Thompson algorithm was developed. The experimental results of benchmark problems and comparisons with two other algorithms indicate the effectiveness of the proposed algorithm.

Introduction

A hybrid flow shop (HFS) scheduling problem is characterized as the processing of *n* jobs through an *s*-stage flow shop, where there exist at least more than one machine at a stage. It is an extension of classical flow shop scheduling and parallel machine scheduling problems. The HFS scheduling problem has important applications in practical production systems, including the petrochemical, steel, electronics, paper, and textile industries [1].

The HFS problem considered in this paper is referred to as $FHs,((PM^{(k)})_{k=1}^{s})\,||\,C_{\max}$ [2]. The minimization of the makespan or the maximum completion time (denoted as $C_{\max}$) is the most common and popular measure of the performance [1]. This kind of problem is NP-hard even if it contains two stages [3]. Many different approaches have been proposed to solve this HFS problem, such as exact algorithms [4,5], heuristics and metaheuristics [6-11]. However, no comprehensive theories are available to guide the algorithm design due to the lack of study on basic properties. Thus, we will also discuss some properties and develop a genetic algorithm with direction selection for this scheduling problem. The effectiveness of the proposed method is tested on benchmark problems by Carlier and Néron [4].

Problem Statement

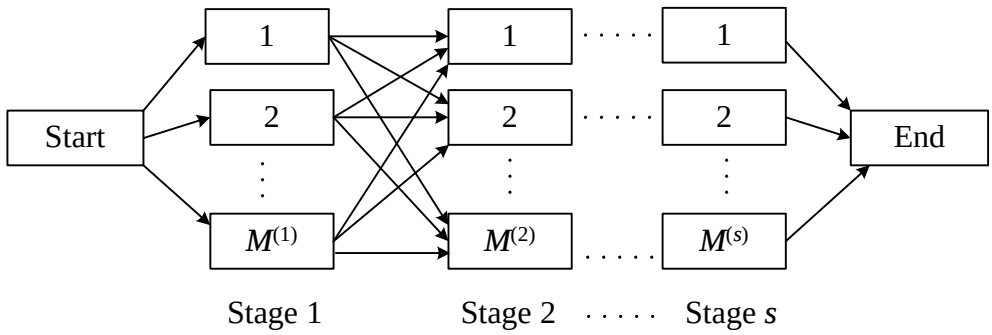

Fig. 1 Layout of the hybrid flow shop environment

Description of the Problem. The HFS scheduling problem is described as follows. As shown in Fig. 1, there are n independent and simultaneously available jobs to be processed through s stages in series. Stage k has $M^{(k)}$ identical machines and has sufficient capacity of buffer

storage for work-in-processes. At least one stage has more than one machine. Each machine can process at most one job at a time. Job j has a processing time p_{jk} and has to be processed without preemption on only one of machines at stage k. The objective is to find a schedule that minimizes the makespan.

Mathematical Model. To introduce the HFS model, the required notations are described as follows:

J set of the jobs to be scheduled, $J = \{1, \mathbf{L}, n\}$

j, g, h index of jobs, $j, g, h \in J$

k index of stages, $k \in \{1, \mathbf{L}, s\}$

m index of machines

$M^{(k)}$ number of machines at stage k

p_{jk} processing time of job j at stage k

U very large positive number

x_{jkm} 1, if job j is on machine m at stage k, 0 otherwise

y_{ghk} 1, if job g is before job h at stage k, 0 otherwise

t_{jk} start time of job j at stage k

$C_{\max}$ makespan

Using the above notations, the HFS scheduling problem can be formulated as a 0-1 mixed-integer linear programming as follows [6]:

$$\min\ C_{\max} \tag{1}$$

$$\text{s.t.}\quad t_{js} + p_{js} \le C_{\max},\ \forall j \tag{2}$$

$$t_{jk} + p_{jk} \le t_{j,k+1},\ \forall j, \forall k \in \{1, \mathbf{L}, s-1\} \tag{3}$$

$$t_{gk} + p_{gk} \le t_{hk} + U(3 - x_{gkm} - x_{hkm} - y_{ghk}),\quad \forall g,h(g \ne h), \forall k, \forall m \tag{4}$$

$$y_{ghk} + y_{hgk} \le 1,\ \forall g,h(g \ne h), \forall k \tag{5}$$

$$\sum_{m=1}^{M_k} x_{jkm} = 1,\ \forall j, \forall k \tag{6}$$

$$t_{jk} \ge 0,\ \forall j, \forall k \tag{7}$$

$$x_{jkm}, y_{ghk} \in \{0,1\} \tag{8}$$

Constraint sets (2) indicate that the completion time of the last job at the last stage s is $C_{\max}$. Constraint sets (3) show that it is not possible for job j to be processed at stage $k+1$ before job j at stage k is completed. Constraint sets (4)-(5) define the processing order for jobs g and h on machine m at stage k. In particular, constraint sets (4) guarantee that a machine can process at most one job at a time. Constraint sets (5) reflect that there exist three kinds of order between job g and h: before, after, and simultaneous. Constraint sets (6) ensure that job j at stage k can only be processed on one machine. Constraint sets (7) and (8) define the domains of the decision variables.

Problem Properties and Solving Procedures

Active Scheduling Technique. An active schedule is feasible schedule in which no operation can be completed earlier without delaying other operations [11]. For $FHs, ((PM^{(k)})_{k=1}^{s}) \| C_{\max}$, it is sufficient to consider only active schedules since the optimal schedule is active [12]. In order to employ this property, an extended Giffler & Thompson (EGT) algorithm was proposed to generate all possible active schedules. Furthermore, the genetic algorithm with forward scheduling (GAFS) combined with EGT was developed to solve this scheduling problem, using the active scheduling technique to reduce the search space [6].

Reversibility Property. It is well known that the regular flow shop scheduling problem has the reversibility property: the makespan does not change if the jobs traverse the flow shop in the opposite direction, in reverse order [13]. As a generalization of the regular flow shop, $FHs,((PM^{(k)})_{k=1}^{s})\,||\,C_{\max}$ also meet this property, which can be described as follows: with the same machine assignment, the makespan does not change if the jobs traverse the hybrid flow shop in the opposite direction, and the jobs processed on machine *m* at stage *k* traverse this machine in reverse order [7]. On the basis of GAFS and reversibility property, the genetic algorithm with backward scheduling (GABS) is easy to implement.

Solution Strategy. GAFS employs EGT to generate active schedules. By observing the EGT algorithm, the generation process is controlled by a set of priority rules which resolves conflict situations from stage 1 to stage s. Therefore, the job assignment and processing sequence at stage k will be restricted by the stages before. In general, if the upper stages of its reverse problem have less influence, then solving the reverse problem can significantly improve the computational efficiency without sacrificing scheduling quality. However, it was found that not for all of the problems the backward scheduling approach outperforms the forward scheduling approach.

On the basis of the above two scheduling approaches, a genetic algorithm with direction selection (GADS) was proposed:

Step 1 Select the scheduling direction. 1) Calculate the flow ratio $FR_k = \sum_{j=1}^{n} p_{jk} \Big/ M^{(k)}$ between the total processing time and the number of available machines at each stage *k*. 2) Calculate the influence factor $IF_k = \sum_{i=1}^{k-1} FR_i$ at each stage *k*. 3) Calculate the cumulative sum $CS = \sum_{k=2}^{s} IF_k$, and select the direction with minimum CS.

Step 2 If the direction is forward, use GAFS to solve the problem; otherwise, use GABS.

Computational Results

Benchmark Problems. The test problems used in the experiments are benchmark problems by Carlier and Néron [4], which were also studied in the literature [5-10]. Their 77 problems are classified into 4 types according the machine configurations. The lower bound (LB) was calculated to analyze the performance of the algorithms [5]. In this study, the solution quality is evaluated by the percentage deviation (PD) between the solution $C_{\max}$ and the LB as follows:

$$\mathrm{PD} = (C_{\max} - \mathrm{LB})/\mathrm{LB} \times 100 \tag{9}$$

Experimental Environment. The above three algorithms were implemented in Matlab 8.3 on a PC with Intel core i3 3.4 GHz processor and 4GB memory. The run time of the algorithms was limited to 1600 s or until the LB was reached. If the LB was not found within the time limit, the search was stopped and the best solution was accepted as the final schedule. These three algorithms were independently run ten times in the same environment for each problem to obtain the best PD (denoted as PD^*), best computation time (T^*) from the replications.

Comparison of a & b Type Problems. To make detailed comparisons between the three algorithms, we selected the 47 problems of type a and b from Carlier and Néron's benchmark problems. The computational results are summarized in table 1, in which D, $\overline{\mathrm{PD}}$, SR, and $\overline{\mathrm{T}}$ indicate the scheduling direction, the average PD, the success ratio to reach LB and the average computation time respectively.

It can be seen from table 1 that the overall mean values of PD^*, $\overline{\mathrm{PD}}$ and SR yielded by GADS are equal to 0, 0.03, and 97.87, respectively, which are better than those generated by GAFS and GABS. Besides, the overall mean values of T^* and $\overline{\mathrm{T}}$ of GADS are much shorter than those generated by GAFS and GABS. This means that GADS can converge to the good solutions faster than GAFS and GABS. Also, it can be seen that GADS is more stable than GAFS and GABS.

Table 1 Comparison results for a & b type problems

examples	LB	GADS						GAFS					GABS				
		D	PD^*	$\overline{PD}$	SR	T^*	$\overline{T}$	PD^*	$\overline{PD}$	SR	T^*	$\overline{T}$	PD^*	$\overline{PD}$	SR	T^*	$\overline{T}$
j10c5a2	88	F	0	0	100	0.12	0.20	0	0	100	0.12	0.14	0	0	100	0.15	3.12
j10c5a3	117	F	0	0	100	0.15	0.24	0	0	100	0.17	0.25	0	0	100	0.17	0.62
j10c5a4	121	B	0	0	100	0.46	12.04	0	0	100	0.12	0.15	0	0	100	0.46	12.89
j10c5a5	122	F	0	0	100	0.13	0.17	0	0	100	0.12	0.19	0	0	100	0.15	0.20
j10c5a6	110	B	0	0	100	0.13	0.74	0	0	100	10.18	150.82	0	0	100	0.13	0.64
j10c5b1	130	B	0	0	100	0.12	0.13	0	0	100	0.12	0.13	0	0	100	0.11	0.14
j10c5b2	107	B	0	0	100	0.11	0.13	0	0	100	0.12	0.13	0	0	100	0.11	0.13
j10c5b3	109	B	0	0	100	0.11	0.13	0	0	100	0.18	2.09	0	0	100	0.12	0.14
j10c5b4	122	B	0	0	100	0.12	0.14	0	0	100	5.25	103.52	0	0	100	0.12	0.14
j10c5b5	153	B	0	0	100	0.12	0.12	0	0	100	0.12	0.18	0	0	100	0.11	0.12
j10c5b6	115	B	0	0	100	0.12	0.13	0	0	100	0.11	0.12	0	0	100	0.12	0.13
j10c10a1	139	B	0	0	100	0.18	0.46	0	0	100	0.17	0.44	0	0	100	0.18	0.42
j10c10a2	158	B	0	0.06	90	3.24	250.97	1.27	1.27	0	1599.65	1599.90	0	0.06	90	3.24	329.57
j10c10a3	148	B	0	0	100	2.08	101.01	0	0	100	0.24	0.38	0	0	100	2.08	102.62
j10c10a4	149	F	0	0	100	0.50	179.77	0	0	100	0.50	69.14	0	0	100	0.15	0.24
j10c10a5	148	B	0	0	100	0.17	0.21	0	0	100	0.18	0.39	0	0	100	0.13	0.20
j10c10a6	146	B	0	0	100	0.29	3.89	0	0	100	0.77	6.03	0	0	100	0.60	4.61
j10c10b1	163	B	0	0	100	0.13	0.13	0	0	100	0.12	0.14	0	0	100	0.12	0.13
j10c10b2	157	B	0	0	100	0.35	0.66	0	0	100	1.75	249.45	0	0	100	0.35	0.73
j10c10b3	169	B	0	0	100	0.12	0.13	0	0	100	0.22	0.45	0	0	100	0.12	0.13
j10c10b4	159	B	0	0	100	0.13	0.15	0	0	100	0.12	0.17	0	0	100	0.13	0.15
j10c10b5	165	B	0	0	100	0.13	0.17	0	0	100	0.14	0.18	0	0	100	0.13	0.15
j10c10b6	165	B	0	0	100	0.13	0.16	0	0	100	0.19	0.63	0	0	100	0.14	0.16
j15c5a1	178	B	0	0	100	0.15	0.29	0	0	100	0.29	0.39	0	0	100	0.14	0.27
j15c5a2	165	F	0	0	100	0.12	0.15	0	0	100	0.12	0.14	0	0	100	0.17	0.48
j15c5a3	130	F	0	0	100	0.12	0.15	0	0	100	0.12	0.15	0	0	100	0.14	0.36
j15c5a4	156	B	0	1.15	10	256.44	1464.60	0	0	100	0.16	0.22	0	1.03	20	256.44	1381.63
j15c5a5	164	B	0	0	100	1.57	311.08	0	0	100	0.15	0.28	0	0	100	0.73	213.61
j15c5a6	178	F	0	0	100	0.13	0.18	0	0	100	0.14	0.19	0	0	100	0.19	1.99
j15c5b1	170	B	0	0	100	0.12	0.13	0	0	100	0.12	0.13	0	0	100	0.12	0.13
j15c5b2	152	B	0	0	100	0.12	0.14	0	0	100	0.12	0.13	0	0	100	0.12	0.13
j15c5b3	157	B	0	0	100	0.11	0.13	0	0	100	0.68	3.90	0	0	100	0.11	0.14
j15c5b4	147	B	0	0	100	0.14	0.56	0	0	100	0.15	2.82	0	0	100	0.14	0.59
j15c5b5	166	B	0	0	100	0.14	0.17	0	0	100	1.11	3.65	0	0	100	0.14	0.17
j15c5b6	175	B	0	0	100	0.12	0.13	0	0	100	0.13	0.45	0	0	100	0.12	0.13
j15c10a1	236	B	0	0	100	0.12	0.15	0	0	100	0.13	0.16	0	0	100	0.13	0.15
j15c10a2	200	B	0	0	100	0.47	71.54	0	1.35	10	293.41	1469.30	0	0	100	0.47	148.60
j15c10a3	198	B	0	0	100	0.13	0.19	0	0	100	0.54	1.32	0	0	100	0.13	0.19
j15c10a4	225	B	0	0	100	0.14	0.74	0	0	100	3.72	200.55	0	0	100	0.19	1.71
j15c10a5	182	B	0	0	100	0.21	0.44	0	0	100	0.52	9.90	0	0	100	0.21	0.41
j15c10a6	200	B	0	0	100	0.16	0.23	0	0	100	0.15	0.27	0	0	100	0.16	0.24
j15c10b1	222	F	0	0	100	0.16	0.34	0	0	100	0.16	0.29	0	0.27	40	235.63	1300.81
j15c10b2	187	F	0	0	100	0.17	0.54	0	0	100	0.17	0.62	0	0	100	0.13	0.17
j15c10b3	222	F	0	0	100	0.12	0.16	0	0	100	0.13	0.17	0	0	100	0.13	0.14
j15c10b4	221	F	0	0	100	0.21	0.42	0	0	100	0.16	0.34	0	0	100	0.21	1.78
j15c10b5	200	F	0	0	100	0.28	0.52	0	0	100	0.25	0.43	0	0.05	90	3.94	269.63
j15c10b6	219	F	0	0	100	0.17	1.27	0	0	100	0.19	1.32	0	0.27	40	52.49	1184.30
Mean			0	0.03	97.87	5.76	51.19	0.03	0.06	95.96	40.93	82.60	0	0.04	95.32	11.95	105.64

Conclusions

In this paper, we examined the HFS problem with makespan criterion and some properties. A genetic algorithm with direction selection (GADS) was developed to solve this problem. To evaluate the performance of the GADS algorithm, it was tested on the well-known benchmark problems by Carlier and Néron. Computational results show that the proposed GADS algorithm outperforms the other algorithms.

Acknowledgements

This research is partially supported by National Natural Science Foundation of China (No. 71371162), the Natural Science Foundation of Fujian Province, China (No. 2014J01271), and the High-Level Talent Foundation of Xiamen University of Technology (No. YSK10009R).

References

[1] R. Ruiz, J.A. Vázquez-Rodríguez, The hybrid flow shop scheduling problem, Eur. J. Oper. Res. 205 (2010) 1-18.

[2] J.A. Hoogeveen, J.K. Lenstra, B. Veltman, Preemptive scheduling in a two-stage multiprocessor flowshop is NP-hard, Eur. J. Oper. Res. 89 (1996) 172-175.

[3] J. N. D. Gupta, Two-stage hybrid flowshop scheduling problem, J. Oper. Res. Soc. 39 (1988) 359-364.

[4] J. Carlier, E. Néron, An exact method for solving the multi-processor flow-shop, RAIRO- Oper. Res. 34 (2000) 1-25.

[5] E. Néron, P. Baptiste, J. N. D. Gupta, Solving hybrid flow shop problem using energetic reasoning and global operations, Omega- Int. J. Manage. S. 29 (2001) 501-511.

[6] Z.X. Su, T.K. Li, Genetic algorithm for minimizing the makespan in hybrid flow shop scheduling problem, Proc. of International Conference on Management and Service Sciences, IEEE, Beijing, 2009.

[7] Z.X. Su, J.M. Yi, Genetic algorithm with forward-backward scheduling approach for hybrid flow shop problems, Computer Integrated Manufacturing Systems, http://www.cnki.net/kcms/detail/11.3619.TP.20150911.1701.010.html (In Chinese).

[8] Q.K. Pan, L. Wang, J.Q. Li, et al, A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimization, Omega- Int. J. Manage. S. 45 (2014) 42-56.

[9] Z. Cui, X.S. Gu, An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems, Neurocomputing 148 (2015) 248-259.

[10] H. Zini, S. ElBernoussi, A discrete particle swarm optimization with combined priority dispatching rules for hybrid flow shop scheduling problem, Appl. Math. Sci. 9(2015) 1175-1187.

[11] M. Kreutz, D. Hanke, S. Gehlen, Solving extended hybrid-flow-shop problems using active schedule generation and genetic algorithms, Proc. of International Conference on Parallel Problem Solving from Nature, Springer, Paris, 2000, pp. 293-302.

[12] Y.H. He, C.W. Hui, Genetic algorithm for large-size multi-stage batch plant scheduling, Chem. Eng. Sci. 62 (2007) 1504-1523.

[13] M. Pinedo, Scheduling: theory, algorithms, and system, second ed., Prentice Hall, New Jersey, 2002.