An algorithm for assembly job shop scheduling problem

Xiao-Qin Wan[1,2,a] and Hong-Sen Yan[1,2,b]

[1]School of Automation, Southeast University, Nanjing 210096, China;

[2]MOE Key Laboratory of Measurement and Control of Complex Systems of Engineering, Southeast University, Nanjing 210096, China.

[a]xqwan87@163.com, [b]hsyan@seu.edu.cn

Keywords: Assembly, job shop, tardiness, earliness.

Abstract. The problem of assembly job shop scheduling (AJSS) is studied with the objective of minimizing the weighted sum of earliness and tardiness penalties. An insertion search algorithm is proposed to solve the problem. The operations of job with tardiness or earliness are shifted left or right to optimize the sequence. Simulation results validate the effectiveness of the proposed model and algorithm.

Introduction

An assembly job shop (AJS) refers to a shop which involves both processing and assembly operations. It is assumed that each job has a tree structure of component and subassemblies that assembly together to build up the end job [1], as shown in Fig.1. These subassemblies in turn have sub-sub-assemblies and so on. These are called as multi level assembly jobs. Higher-level subassembly cannot be assembled until all preceding lower-levels components and/or subassemblies are completed. In an AJS, subassemblies/assemblies undergo operations in a serial fashion as per the precedence constrains and wait for the arrival of its mating components at the assembly station, for the assembly operation to start [2]. As the number of levels increases the complexity of scheduling also increases. This makes the AJS problems quite challenging.

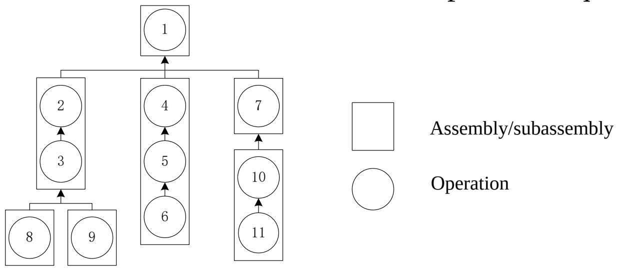

Fig. 1 Tree structure of job

Many previous studies have focused on AJS problems. Some Dispatching rules for AJS have been developed and compared in accordance with several performance measures. Adam, Bertrand and Surkis [3] developed two priority rules called the relative number of remaining operation (RRO) and relative remaining processing time (RRP) rules. Philipoom, Russell and Fry [4] proposed a set of dispatching rules, called important ratio (IR). Adam et al. [5] proposed dynamic due date assignment procedures for multi-level job structures and investigated the interaction between the procedures and two priority rules: earliest job due date (JDD) and earliest operation due date (OPNDD). Reeja and Rajendran [6,7] designed for AJS a class of priority rules termed as developed two priority rules called operation synchronization data (OSD) and a new version of operation due date (ODD). Natarajan et al. [8] proposed several dispatching rules with the consideration of different weights for holding and tardiness of jobs. Some other researches focus on developing efficient heuristics for AJS. Kim and Kim [9] compared the results of SA and GA methods. With respect to the objective of minimizing the weighted sum of tardiness and earliness of items, they found that SA algorithms outperformed GAs in their problems. Wong and Ngan [10] compared the capability of HGA and HPSO in minimizing the makespan under AJSSP environment with lot streaming technique.

The objective of this paper is to develop an effective algorithm for solving the AJS problem. In Section 2, the problem formulation is presented. In Section 3, an insertion search is presented. Computational results are reported and analyzed in Section 4. Conclusions are provided in Section 5.

Problem Description and Formulation

This paper focuses on the problem of AJS scheduling, its main objective being to minimize the weighted sum of earliness and tardiness penalties of jobs. To present the problem formulation, the following notations are used:

n : Number of jobs, $i, j = 1, \ldots, n$;

m : Number of machines $l = 1, \ldots, m$

K_i : Number of operations of job i

o_{ik} : The kth operation of job i

s_{ik} : Start time of the o_{ik}

t_{ik} : Processing time of o_{ik}

c_{ik} : Completion time of o_{ik}

c_i : Completion time of job i

d_i : Due date of job i

E_i : Earliness of job i

T_i : Tardiness of job i

M: Large positive number

α_i : Earliness penalty of job i

β_i : Tardiness penalty of job i

$X_{ikjk'l}$: If o_{ik} precedes $o_{jk'}$ on machine l, then $X_{ikjk'm} = 1$; otherwise $X_{ikjk'm} = 0$

The rescheduling problem for assembly job shop is formulated as follows:

$$\text{Min } \sum_{i=1}^{n} \left(\alpha_i E_i + \beta_i T_i\right) \tag{1}$$

s.t.

$$E_i = \max\left(d_i - c_i, 0\right) \tag{2}$$

$$T_i = \max\left(c_i - d_i, 0\right) \tag{3}$$

$$c_{ik} - s_{ik} = t_{ik}, \ \forall i, k \tag{4}$$

$$s_{i\tilde{k}} - c_{ik} \geq 0, \ \forall i, k \tag{5}$$

$$s_{ik} \geq 0, \ \forall i, k \tag{6}$$

$$c_{jk'} - c_{ik} + M(1 - X_{ik\,jk'l}) \geq t_{jk'}, \ \forall i, j, k, k' \tag{7}$$

$$c_{ik} - c_{jk'} + MX_{ik\,jk'l} \geq t_{ik}, \ \forall i, j, k, k' \tag{8}$$

$$X_{ik\,jk'} \in \{0,1\}, \quad \forall i, j, k, k' \tag{9}$$

Objective (1) is to minimize the total weighted earliness and tardiness penalties. Constraints (2) and (3) compute the earliness and tardiness of job i respectively. Constraint (4) indicates that means that once an operation is started, it cannot be preempted until it is completed. Constraint (5) ensures the precedence relations between operations. Constraint (6) expresses the fact that the start time of each operation is positive. Constraints (7) and (8) ensure that no more than one operation can be processed simultaneously by the same machine. Constraint (9) defines the 0-1 integer variable.

An Insertion Search Algorithm

An insertion search is developed to optimize the operation sequence. For the job with earliness or tardiness, extract its operations and then shift right or left without violating the precedent constraints. The steps of the insertion search are as follows.

Step 1: Select an appropriate dispatching rule to allocate the operations and then the sequence $\boldsymbol{S}$ and the objective function Z are obtained.

Step 2: Initialize the maximum iteration $r_{\max}$. Set $r=1$.

Step 3: Sort the jobs in the order of decreasing penalty and record in $\boldsymbol{GS}$. Set $h=1$.

Step 4: If the penalty of job $GS(h)$ is greater than zero, record its operations in $\boldsymbol{EQ}$ in sequence, and label the total number of operations in $\boldsymbol{EQ}$ as P, set $p=1$, then go to Step 5; otherwise, go to step 15.

Step 5: Operations in $\boldsymbol{S}$ which processed by machine $l\ (l=1,\cdots,m)$ are recorded in vector $\boldsymbol{G}_l$ in sequence. e_l denotes the total number of elements in $\boldsymbol{G}_l$. Let $\boldsymbol{S}_1 \leftarrow \boldsymbol{S}$.

Step 6: Delete the operation $EQ(p)$ in $\boldsymbol{S}_1$. Find the minimum position of its succeeding operations and maximum position of preceding ones in $\boldsymbol{S}_1$, labeled as UP and DW respectively.

Step 7: Find the machine where $EQ(p)$ assigned and labeled as l. Find the position of $EQ(p)$ in $\boldsymbol{G}_l$, labeled as d.

Step 8: If job $GS(h)$ with earliness and $d<e_l$, find the position of the operation (at $d+1$ of $\boldsymbol{G}_l$) in $\boldsymbol{S}_1$, labeled as d^+, and then go to Step 9; otherwise, go to Step 10.

Step 9: If $d^+ < UP$, insert $EQ(p)$ into the position $d^+ + 1$ in $\boldsymbol{S}_1$, then go to Step 12; otherwise, go to Step 13.

Step 10: If job $GS(h)$ with tardiness and $d>1$, find the position of the operation (at $d-1$ of $\boldsymbol{G}_l$) in $\boldsymbol{S}_1$, labeled as d^-, and then go to Step 11; otherwise, go to Step 13.

Step 11: If $d^- > DW$, insert $EQ(p)$ into the position d^- in $\boldsymbol{S}_1$, then go to Step 12; otherwise, go to Step 13.

Step 12: Calculate the objective cost Z_1. If $Z_1 \le Z$, then $\boldsymbol{S} \leftarrow \boldsymbol{S}_1$ and $Z_1 \leftarrow Z$.

Step 13: $p \leftarrow p+1$. If $p > P$, go to Step 14; otherwise, go to Step5.

Step 14: $h \leftarrow h+1$. If $h>n$, go to Step 15; otherwise, go to Step 4.

Step 15: $r \leftarrow r+1$. If $r>r_{\max}$, go to Step 16; otherwise, go to Step 3.

Step 16: Stop.

Numerical Tests

To test the effectiveness of the proposed algorithm, a set of instances are generated. The data set covers the two problem size of 10 and 20, for each of which three types of product structures are considered: single-level structures, two-level structures, and three-level structures [6,7]. The number of operations per subassembly is sampled from a discrete uniform distribution in the range [1, 4]. The processing times for the operations are randomly generated from a uniform distribution in the range [1, 10], and the unit earliness cost and the tardiness cost are sampled from in the ranges [1, 4] and [1, 6] respectively. The number of machines $m \in \{8,10\}$. The process routes are independent and generated randomly. The due date of a product is calculated using the following formula:

$$d_i = \left\lfloor f \cdot \sum_{k=1}^{K_i} t_{ik} \right\rfloor$$

where f is a due date tightness factor [11]. Two values of f are considered in the experiment: 1.5 and 2.

The three rules $\mathrm{SEFT}^{\tau}/\mathrm{ECT}$, $\mathrm{TWKR}^{\tau}/\mathrm{OSD}$, and $\mathrm{WTWKR}^{\tau}/\mathrm{LFT}$ [8] are selected to generate the initial scheduling and then followed by the insertion search algorithm. The procedures were referred to as HS, HT and HW respectively. A GA was included in the computational test for comparison. The population size is 30 and the number of generations is 300. The crossover and mutation rates are 0.9 and 0.3 respectively. A random key represent was adopted to represent the chromosome [12]. A linear crossover and swap mutation were employed here. Comparison results are presented in Tables 1 and 2.

As shown, the proposed HS algorithm outperforms those of the others. Compared with HW, the HT algorithm gives better results in 19 of the 24 cases. The selection of the dispatching rules has a significant impact on the quality of the solutions. For the cases of *n*=10, the HS and GA algorithms perform well with respect to single-level structures. As product structure goes complex, the performance of GA gets worse.

Table 1 Computational results for the algorithms

$n \times m$	算法	*f*=1.5			*f*=2		
		S1	S1	S3	S1	S1	S3
10×8	HS	488.6	1316.7	3428.3	285.8	1201.5	3261.9
	HT	572.1	1323.1	3938.1	353.7	1297.5	3524.1
	HW	586.0	1415.3	3479.9	541.8	1470.1	4060.5
	GA	499.8	2162.5	6266.9	431.5	2135.9	6429.5
10×10	HS	347.4	1202.1	2675.9	242.2	736.5	1935.0
	HT	387.4	1433.2	3288.1	307.7	943.6	2895.9
	HW	417.1	1279.7	3852.3	410.3	1097.1	2826.4
	GA	440.6	1867.5	5511.7	297.8	1588.8	4921.2

Table 2 Computational results for the algorithms

$n \times m$	算法	*f*=1.5			*f*=2		
		S1	S2	S3	S1	S2	S3
20×8	HS	2411.5	6757.5	14266.4	1993.9	5951.6	13323.5
	HT	2666.7	6933.2	14377.0	2393.3	6054.4	14612.7
	HW	3095.5	7619.0	16139.4	3175.4	5794.7	14866.6
	GA	3568.1	1134.8	29713.2	2895.5	11010.8	27399.0
20×10	HS	1832.1	5148.2	11761.5	1413.1	5035.6	10015.6
	HT	1984.8	5870.5	14608.5	1622.3	5624.2	11988.1
	HW	2633.4	5887.4	14043.7	2053.4	6662.0	13293.0
	GA	2591.1	9938.7	26130.0	2101.6	10143.0	23249.8

Conclusion

This paper considers the problem of AJSS with earliness and tardiness penalties. An insertion search algorithm was developed and tested. The results show that the $\mathrm{SEFT}^{\tau}/\mathrm{ECT}$ and the insertion search work well for the measures of performance related to the earliness and tardiness.

For further research, the dominance relations of operations are suggested to be investigated in depth. More efficient heuristic algorithm for the problem is yet to be developed.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grant 60934008; the Fundamental Research Funds for the Central Universities of China under Grant 2242014K10031; the Priority Academic Program Development of Jiangsu Higher Education Institutions and the Scientific Research Foundation of Graduate School of Southeast University under Grant YBJJ1446.

Reference

[1] S. Pathumnakul, P.J. Egbelu. An algorithm for minimizing weighted earliness penalty in assembly job shops. International Journal of Production Economics. 2006, 103(1): 230-245.

[2] M. Omkumar, P. Shahabudeen. Ant colony optimisation for multi-level assembly job shop scheduling. International Journal of Manufacturing Research, 2009, 4(4): 410-427.

[3] N.R. Adam, , J.W.M. Bertrand, J. Surkis. Priority assignment procedures in multi-level assembly job shops. IIE Transactions. 1987, 19(3): 317–328.

[4] P.R. Philipoom, R.S. Russell, T.D. Fry. A preliminary investigation of multi-attribute based sequencing rules for assembly shops. International Journal of Production Research. 1991, 29 (4): 739-753.

[5] N.R. Adam, J.W.M. Bertrand, D.C. Morehead, J. Surkis. Due date assignment procedures with dynamically updated coefficients for multi-level assembly job shops. European Journal of Operational Research. 1993, 68(2): 212–227.

[6] M.K. Reeja, C. Rajendran. Dispatching rules for scheduling in assembly jobshop-part 1. International Journal of Production Research. 2000, 38 (9): 2051-2066.

[7] M.K. Reeja, C. Rajendran. Dispatching rules for scheduling in assembly jobshops-Part 2. International Journal of Production Research. 2000, 38(10): 2349-2360.

[8] K. Natarajan, K.M. Mohanasundaram, B.S. Babu, S. Suresh, K.A.A.D. Raj, C. Rajendran. Performance evaluation of priority dispatching rules in multi-level assembly job shops with jobs having weights for flowtime and tardiness. International Journal of Advanced Manufacturing Technology. 2007, 31(7-8): 751–761.

[9] J.U. Kim, Y.D. Kim. Simulated annealing and genetic algorithms for scheduling products with multi-level product structure. Computers & Operations Research. 1996, 23(9): 857–868.

[10] T.C Wong, S.C. Ngan. A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop. Applied Soft Computing. 2013, 13(3): 1391-1399.

[11] I. Essafi, Y. Mati, S. Dauzère-Pérès. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Computers & Operations Research. 2008, 35(8): 2599-2616.

[12] J.C. Bean. Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing. 1994, 6(2):154–160.