第 16 卷第 6 期
1999 年 12 月

控制理论与应用
CONTROL THEORY AND APPLICATIONS

Vol.16, No.6
Dec., 1999

Schedule Grouped Jobs on Parallel Machines with Single Mold Constraint*

Gao Lin
(Department of Automation, Tsinghua University·Beijing, 100084, P.R.China)
Wang Dingwei
(College of Information Science & Engineering, Northeastern University·Shenyang, 110006, P.R.China)
Wang Shuning
(Department of Automation, Tsinghua University·Beijing, 100084, P.R.China)

Abstract: This paper addresses the problem of scheduling n grouped jobs on m identical parallel machines to minimize the total tardiness, subject to single mold constraint. For this problem, there is an optimal solution without machine idle. Thus, the scale of searching for optimal solution is reduced. Branch and bound algorithm, run-based heuristic, multi-stage tabu search and a combined algorithm are proposed and compared in simulation experiments. Some practically useful results are obtained.

Key words: production scheduling; parallel machine; group technology

单一模具约束的平行机台成组工作调度方法

高　林　　汪定伟
(清华大学自动化系·北京,100084)　(东北大学信息科学与工程学院·沈阳,110006)
王书宁
(清华大学自动化系·北京,100084)

摘要: 本文处理在平行机台上调度具有单一模具约束的成组工作,以最小化总拖期量的问题.研究了最优解的性质,并提出了分枝定界法、启发式算法、多阶段 tabu search 算法及组合方法.利用随机问题对各算法进行了对比和分析,获得有实践指导意义的结果.

关键词: 生产调度; 平行机台; 成组技术

1 Introduction

In this paper, we address the problem to schedule grouped jobs in identical parallel machines with single mold constraint. The problem can be stated as follows: there are m identical parallel machines and a set of n jobs associated with known processing times and duedates. These jobs belong to G groups ($G > m$). The processes of jobs belonging to one group need the same mold, and for each group only one mold is available, so any two jobs of the same group can not be processed on different machines simultaneously. A mold-dependent setup time is necessary between processes using different molds. The objective is to find an optimal job schedule to minimize the sum of tardiness.

Up to now, there is no published literature to deal with such a problem, but in the field of parallel machine scheduling and group technology, rich literature can be found.

For the parallel machine scheduling, minimized total tardiness scheduling problem is proved to be NP-complete even for two-machine system[1]; optimal algorithms based on dynamic programming are available[2], but the dimensionality problem is acute, and thus can handle problems with fewer jobs. Some heurisitic methods are also available, where the idea of list scheduling is applied, according to which, the jobs are sorted with

* This research was partially supported by the National Natural Science Foundation under Grants (69684005), Chinese Postdoctoral Science Foundation, and by TSANNKUEN Enterprise Company Limited.
Manuscript received Jul. 16, 1998, revised Jul. 7, 1999.

a priority rule to form a list, and then are assigned to machines with some rules one by one. The most often used priority rules for sorting are SPT, EDD, and Minimum Slack[3]. In the field of group technology, feasible algorithms are dynamic programming[4] and heuristic. Also, some literatures deal with scheduling preemptive batch on parallel machines.

So our problem is obviously a NP-hard. We develop several algorithms respectively, to solve problems with different size, and test them with simulation experiments. Satisfactory results have been obtained.

2 Model description

In the model description, we adopt the commonly used notation: S denotes a schedule, and $f(S)$ the objective of schedule S, n_k the number of jobs scheduled in machine k $(\sum_{k=1}^{m} n_k = n)$, (k, i) the ith job scheduled in machine k, $M(j)$ the mold used by job j, $M0(k, i)$ the mold used by job (k, i), p_j the processing time of job j, d_j the due-date of job j, C_j the completion time of job j, T_j the tardiness of job j, $T_j = \max\{0, C_j - d_j\}$, S_p the setup time of mold p, and $c_{k,i}$ the completion time of the ith job scheduled on machine k.

Our task is to find an optimal schedule S^* to minimize the total tardiness, namely,

$$f(S^*) = \min\Big(f(S) = \sum_{j=1}^{n} T_j\Big).$$

For the problem of minimum total tardiness scheduling of grouped jobs on parallel machines without mold constraint, it is obvious that an optimal schedule without machine idle exists. For the problem with single mold constraint, it is possible that some machine idles are necessary. So we proved the following lemma and theorem:

Lemma 1 If the feasible schedule $S^{(1)}$ can be generated from schedule S by moving forward the process of one job and not shifting that of others, then

$$f(S^{(1)}) \leqslant f(S).$$

Proof This lemma is obvious. It is impossible for this kind of moving to worsen the scheme.

Theorem 1 For the problem of scheduling grouped jobs in identical parallel machines with single mold constraint to minimize total tardiness, there exists at least one optimal schedule without machine idle.

Proof To save the space, here we only give the outline of the proof. Readers may refer to [5] for detail. For any optimal schedule S with machine idles, we design a general procedure, which, based on the method of exchanging job series, can move an idle to the back of its adjacent latter job and will not worsen the performance of S. For each idle starting earliest, repeat this procedure, we will move idles on each machine to the back of jobs and get a schedule S^* without machine idle, and $f(S^*) \leqslant f(S)$. Since S is an optimal schedule, there must be $f(S^*) = f(S)$, S^* is an optimal solution without machine idle. This completes the proof.

Based upon this theorem, the search space for optimal solution may be reduced to a solution set without machine idle. Thus the model can be described as an Integer Programming Model[5].

3 Optimal solution (BB)

There is certainly a need to gain the true optimal solutions for problems over a wide range of parameters. However, the computational requirement to obtain the true optimal solution by complete enumeration is prohibitive even for a small problem. Complete enumeration, in the worst case, requires

$$\sum\nolimits_{n_1+n_2+\cdots+n_m=n} P_n^{n_1} \cdot P_{n-n_1}^{n_2} \cdots P_{n-n_1-\cdots-n_{m-1}}^{n_m}$$

number of enumerations.

Here we develop a Branch and Bound algorithm. At first, we use a list of job symbol and partitioning symbol (*) as the coding scheme for multiple machine scheduling problem. For example,

$$[5 \quad 2 \quad 3 \quad * \quad 4 \quad 1]$$

means the schedule:

$$M1:\ J_5 + J_2 + J_3,$$

$$M2:\ J_4 + J_1.$$

Generally, for a problem of scheduling n jobs on m machines, a complete code contains n job symbols and $m - 1$ partitioning symbols, resulting in a total length of $(n + m - 1)$. We define it as the Size of the problem.

Due to the complexity of the constraints in our problem, we employed simulation to evaluate any schedule generated during branch procedure. The Branch and Bound algorithm uses the depth first search strategy. Because machines are identical in our problem, there must exist an optimal solution with a non-increasing order of the number of jobs scheduled on each machine. So in the Branch and Bound algorithm, we search only for this solution in the solution set that satisfies

$$n_{i-1} \geqslant n_i \geqslant \frac{n - \sum_{j=1}^{i-1} n_j}{m - i + 1}.$$

This algorithm is suitalble for the small-size problem. The computational time for solving the problem of Size = 12 on a microcomputer with Pentium 200 processor is about 25 minutes.

4 A run-based heuristic(HR)

In the research on group technology, a schedule can be expressed as a series of runs[6]. A run consists of jobs that are processed between two adjacent setups on one machine. For a given schedule, let $\beta(r_i)$ denote the number of jobs in run r_i, and identify each run r_i with its first job i.

The heuristic proposed in this paper, named run-based heuristic, is an improved list scheduling. At first we group the jobs with some characters into a run, then the runs are sorted by the nondecreasing order of slack and assigned to the first available machine one by one.

The principle of the grouping procedure can be described as follows:

In the common sense, we take it for granted that jobs with earlier due-date should be processed earlier, so for each group, we sequence jobs belonging to it by EDD order, and label them. Let $[b, i]$ denote the ith job of group b, and $|b|$ the number of jobs in group b.

In order to complete all latter jobs of $[b, i]$(i.e. $[b, i+1]$ to $[b, |b|]$) on time, due to the constraint of mold resource, there must be $C_{[b,i]} \leqslant \min\limits_{i<j\leqslant |b|} \left(d_{[b,j]} - \sum_{l=i+1}^{j} p_{[b,l]} \right)$, therefore the family-adjusted due-date of job $[b, i]$ is defined as

$$d'_{[b,i]} = \min\left(d_{[b,i]}, \min_{i<j\leqslant |b|} \left(d_{[b,j]} - \sum_{l=i+1}^{j} p_{[b,l]} \right) \right).$$

This is a procedure of resource requirement balance. Then we group jobs $[b, i], [b, i+1], \cdots, [b, j]$ into a run, where

$$j = \max_{i\leqslant k\leqslant |b|} \left(k : \left(d_{[b,k]} - \sum_{l=i+1}^{k} p_{[b,l]} \right) \leqslant d_{[b,i]} + S_b \right).$$

The procedure of our run-based heuristic can be described as follows:

Step 1 Grouping.

1.1) Group jobs, then sequence the jobs of each group in EDD order. For jobs with the same due-date, SPT order is applied;

1.2) For each group b, calculate the family-adjusted due-date of each job. The iteration function is

$$d'_{[b,|b|]} = d_{[b,|b|]},$$
$$d'_{[b,i]} = \min(d_{[b,i]}, d'_{[b,i+1]} - p_{[b,i+1]}).$$

If $d'_{[b,i+1]} - p_{[b,i+1]} - S_b \leqslant d_{[b,i]}$, mark the relationship between $[b, i]$ and $[b, i+1]$ as connected;

1.3) Group the jobs connected with each other into a run, and thus form a set of n_r runs. Mark $r_{[b,i]}$ with the label of the first job $[b, i]$ in it.

Step 2 Listing.

2.1) Calculate the Slack of each run,

$$\text{Slack}_{r_{[b,i]}} = d'_{[b,i]} - S_b - p_{[b,i]};$$

2.2) Sequence runs in the nondecreasing order of their Slack to gain a list of runs.

Step 3 Assigning.

3.1) Initialize the available time of each machine $k: A_k = 0, k = 1, \cdots, m$; current loaded mold of each machine $k: M_k = 0, k = 1, \cdots, m$ and available time of each mold $b: B_b = 0, b = 1, \cdots, G$;

3.2) For the run on the head of run list, $r_{[b,i]}$, select a machine k for assignment,

$$k = \arg\min\{A_k : A_k \geqslant B_b, \quad k = 1, \cdots, m\};$$

3.3) Modify the machine k's available time:

$$A_k = A_k + \sum_{i\leqslant l< i+\beta(r_{[b,i]})} p_{[b,l]} + \begin{cases} 0, & \text{if} \quad b = M_k, \\ S_b, & \text{otherwise}, \end{cases}$$

mold's available time: $B_b = A_k$, and

machine's current onloading mold: $M_k = b$.

Delete run $r_{[b,i]}$ from the run list. If the run list is not null, goto Step 3.2), otherwise stop.

This heuristic requires the implementing time $O(m \times n + n\log n)$ and very little storage space. The time for solving the problem of Size = 400 is about 5 minutes.

5 Multi-stage tabu search(TS)

Tabu search, first introduced by Glover, as a technique for solving combinatorial optimization problems, is basically a strategy to overcome local optimality. Since then, tabu search has been successfully applied to a wide range of problems. Traditional TS starts with one feasible initial solution. To further enhance the searching ability, in our algorithm, TS starts with a group of initial solutions, and chooses the best move according to the hill-climbing scheme respectively. So we call it a multi-stage tabu search.

We use the same scheme as the one employed in Branch and Bound. The solution of HR together with other 19 solutions that are generated randomly is used as initial solution. For problems with Size$\leqslant$25, all pair-interchanges are utilized as the switching strategy, whose total number is $\text{Size}(\text{Size}-1)/2-(m-1)(m-2)/2$; for problems with Size > 25, 250 random swapping is used. A tabu list size of 7 is selected, and the same simulation procedure as the one used in BB is employed to evaluate any schedule. Aspiration criterion is that any move that improves the performance measurement is selected, even if the move is tabu. A long period list of size 20 records the initial solution or best solution of the 20 stages to avoid repeating among stages and cycling in any stage.

The procedure of the TS is shown as follows:

Step 1 Generate initial solutions for every stage, record them in the long period list, and set $j=1$;

Step 2 Select the jth initial solution as initial sequence;

Step 3 Generate a new sequence by interchanging a pair of jobs in the initial sequence;

Step 4 Check if the new sequence is in the long period list. If it is recorded in the long period list, goto Step 5. Evaluate the new sequence. Apply the aspiration criterion: if the new solution is better than the previous best solution of this stage, then update this solution as the best solution and as the jth solution in the long period list, and record the pair interchange as the best move, goto Step 5. Check the tabu list to determine if this move is forbidden. If it is tabu, goto Step 5; else, determine if it is the best move from initial sequence;

Step 5 Repeat Step 3 to Step 4, until all neighborhood is checked. Interchange the pair of jobs according to the best move, set the solution after moving as the initial sequence, update the tabu list;

Step 6 Repeat Step 3 to Step 5, until the number of the iterations performed in this stage exceeds 5 * Size;

Step 7 Let $j=j+1$, repeat Step 2 to Step 6, until all the stages are performed;

Step 8 Stop and report the best solution in the long period list as the algorithm's result.

This algorithm requires the implementing time $O(m\times n\times \text{Size}+n\log n)$ for solving problems with Size > 25. The computational time for solving the problem of Size = 314 is about 120 minutes.

6 Combined algorithm(CA)

By analyzing the result of HR, we found that the poor performance of most results which are far worse than the optimal solution are mainly caused by the inappropriate sequence of jobs inside the runs. The reason for this is that, in the HR, jobs in each run are sequenced in EDD order, which is the optimal sequence for minimizing the maximum tardiness, rather than for minimizing the total tardiness. So we develop a combined algorithm (CA), in which a single-stage tabu search is employed to improve the performance of the result of HR.

The implementation of CA resembles that of TS except that it starts only with the solution of HR, and the size of the long period list is 1.

This algorithm requires the implementing time $O(m\times n\times \text{Size}+n\log n)$ for solving problems with Size > 25. The computational time for solving the problem of Size = 314 is about 9 minutes.

7 Computational results and analysis

Primary computational experiments are conducted to test the effectiveness of the proposed four algorithms. The simulation results obtained from these algorithms are tabulated in Tables 1 and 2.

The results in Table 1 indicate that the computational time of BB is far longer than the time required by the other three methods. TS can find an optimal solution in all cases, and CA can also reach high probability in reaching optimal, its average deviation from optimal is less than 2%. In contrast to them, the results of HR are not so good—its average deviation from optimal is 46%, and the highest rete of optimal is merely 42%.

According to the data in Table 2, the comparison of the three near-optimal methods is made more apparently with a wide range of problems. In the way of time, HR consumes the shortest time, TS the longest, and CA the modest amount. But the quality of solutions gained by CA and TS is obviously better than HR. The difference between CA and TS are not marked. When the size of problem increases, especially when the size exceeds 30, we can hardly note any difference in the results.

Based upon the analyses above, we can draw the following conclusion: the Branch & bound is available only under the situation where the problem has small size and the limit of computational time is not so strict. The heuristic (HR) fits for solving problems with large size, specially the huge scale problems in real-life application, when the computational speed is required to be high but the quality of sloution just to be satisfying. The multistage tabu search (TS) fits for the situation where the problem has middle size, the quality of solution is important, and the speed is not restricted too much. The combined algorithm (CA) shares the advantages of HR and TS——high speed, high quality——and is the most practicable one for middle-scale problems.

Table 1 Comparison of the performance by the four algorithms in small-size problems

Size (n,m,G)	Computational time(s)				Average deviation from optimal(%)			Rate of optimal(%)		
	BB	HR	TS	CA	HR	TS	CA	HR	TS	CA
8(7,2,4)	2.8	<1	2.5	<1	29.8	0.00	0.41	36	100	98
9(7,3,4)	19.7	<1	2.9	<1	12.8	0.00	0.00	42	100	100
9(8,2,4)	12.3	<1	2.9	<1	46.4	0.00	1.63	36	100	86
10(8,3,4)	81.6	<1	3.1	<1	19.3	0.00	0.02	38	100	98
10(9,2,4)	62.7	<1	3.1	<1	26.2	0.00	1.03	42	100	88

Table 2 Comparison of three near-optimal algorithms

Problem Size (n,m,G)	Computational time(s)			Average deviation from best solution(%)			Rate of the best(%)		
	HR	TS	CA	HR	TS	CA	HR	TS	CA
20(18,3,5)	<1	28	1.6	24.0	0.00	1.86	20	100	70
25(23,3,5)	<1	42	2.1	26.0	0.00	0.46	10	100	86
30(28,3,5)	<1	57	3.4	19.8	0.00	0.00	10	100	100
40(37,4,6)	<1	106	5.5	29.8	0.00	0.00	0	100	100
60(56,5,7)	<1	278	14.5	28.7	0.00	0.00	0	100	100

9 Conclusion

This paper addresses the problem of scheduling n grouped jobs on m identical parallel machines to minimize the total tardiness, subject to the mold constraint. For this problem, we present a theorem that there must exist an optimal solution without machine idle. Based on this theorem, the search for optimal solution is reduced, and a mixed zero-one mathematical programme model is presented. We proposed not only a run-based heuristic algorithm, but also a multi-stage tabu search to find approximate solution of those problems. Furthermore, in consideration of the compromising between computational time and solution quality, we designed a combined algorithm. In the simulation experiments, all of these algorithms are tested using randomly generated problems, and satisfactory results have been gained.

Up to now, the heuristic has been used in a production scheduling system of an electrical appliance plant in south China. It has proved to be efficient in practical application[7].

References

1 Cheng T C E and Sin C C S. A state-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research, 1990,47(2):271-292

2 Rajaraman M K. A parallel sequencing algorithm for minimizing total cost. Naval Research Logistics Quarterly, 1977,24(3):473-481

3 Dogramaci A and Surkis J. Evaluation of a heuristic for scheduling independent jobs on parallel identical processor. Management Science, 1979,23(6):1208-1216

4 Psaraftis H N. A dynamic programming approach for sequencing groups of identical jobs. Operations Research, 1980,28(8):1347-1359

5 Gao L. Research on scheduling group jobs on parallel machines: [Ph. D. degree dissertation]. Shenyang: Northeastern University, 1999

6 Webster S and Baker K R. Scheduling groups of jobs on a single machine. Operations Research, 1995,43(4):692-703

(Continued on page 792)

Switzerland, Birkhauser Verlag, 1993

2 Narendra K S and Taylor J H. Frequency Domain Criteria for Absolute Stability. New York: Academic Press, 1973

3 Savkin A V and Petersen I R. A method for robust stabilization related to the Popov stability criterion. Int. J. Control, 1995, 62(5):1105 - 1115

4 Boyd S, EL Ghaoui L, Feron E and Balakrishnan V. Linear matrix inequality in system and control theory. Philadelphia, SIAM Studies in Applied Mathematics, 15:1994

5 Goh K C, Turan L, Safonov M G et al. Bilinear matrix inequality properties and computational methods. Proc. of Conf. on ACC, Baltimore, 1994, 850 - 855

6 Goh K C, Safonov M G and Papavassilopoulos G P. A global optimization approach for the BMI problem. Proc. of Conf. Decision and Control, Tokyo, 1994, 2009 - 2014

7 Xie L, Fu M and de Souza C E. H^{∞} control and quadratic stabilization of systems with parameter uncertainties via output feedback. IEEE Trans. Automat. Contr., 1992, AC-37:1253 - 1260

本文作者简介

郭　雷　见本刊 1999 年第 4 期第 624 页.

臧明磊　1965 年生. 现为复旦大学数学系研究生. 研究方向为优化方法与计算.

冯纯伯　见本刊 1999 年第 1 期第 126 页.

(Continued from page 787)

7 Gao Lin, Wang Chengyao, Wang Dingwei, Yin Zhisong and Wang Shuning. A production scheduling system for parallel machines in an electrical appliance plant. Computers & Industrial Engineering, 1998, 35(1 - 2):105 - 108

本文作者简介

高　林　1973 年生. 1993 年, 1996 年, 1999 年于东北大学自动控制系本科、硕士、博士毕业. 现为清华大学自动化系博士后. 主要研究方向为生产计划与调度系统的建模、优化与仿真, 智能优化方法, 计算机应用.

汪定伟　1948 年生. 1984 年华中理工大学自动控制系硕士毕业, 1993 年东北大学自动控制系博士毕业. 现为东北大学自动控制系教授、博士生导师. 主要研究方向为生产计划与调度理论, 建模与决策, 智能优化方法.

王书宁　1956 年生. 1982 年湖南大学电气工程系本科毕业, 1984 年, 1988 年华中理工大学自动控制系硕士、博士毕业. 现为清华大学自动化系教授、博士生导师. 主要研究方向为系统辨识, 参数估计, 优化理论和算法以及决策分析.