We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Concurrent Openshop Problem to Minimize the Weighted Number of Late Jobs[1]

H.L. Huang and B.M.T. Lin
National Chiao Tung University
Taiwan, R.O.C.

1. Introduction

Concurrent open shop problem can be viewed as the two-stage assemble-type flow shop (Lee et al., 1993) ignoring the second-stage assembling operation. Consider a set of jobs $J = \{1, 2, \ldots, n\}$ and a set of machines $M = \{1, 2, \ldots, m\}$. Each job J_i is composed of tasks t_{ik} which have to be processed on specific machine k with processing time p_{ik}. Each job J_i has a weight w_i and due date d_i. The major difference of this problem from the traditional open shop problem is that all tasks belong to the same job could be processed concurrently. Let C_{ik} be the completion time of task of job i on machine k. The completion time of a job, C_i, is the greatest completion time among all of its tasks, i.e. $C_i = \max_{1 \leq k \leq m}\{C_{ik}\}$. There are several variant applications in the production field (Roemer, 2006). The objective of minimizing the number of tardy jobs has been discussed extensively in different applications. In this chapter, we consider the concurrent open shop problem of minimizing the weighted number of tardy jobs. Following the three-field notation of Graham et al. (1979), we denote this problem by $PD||\Sigma w_i U_i$.

To best of our knowledge, this problem was first proposed by Ahmadi and Bagchi (1990). Most works consider the objective ΣC_i or $\Sigma w_i C_i$. Roemer (2006) has done an extensive review on different objectives on this problem. Because this chapter discusses only the objective $\Sigma w_i U_i$, we simply review the due date related results. The complexity result was first given by Wagneur and Sriskandarajah (1993). They have shown that even if there are only two machines, this problem is NP-hard. Leung *et al.* (2006) has shown that the $PD|d_i=d|\Sigma U_i$ is NP-hard and they proposed a Revised Hodgson-Moore algorithm to solve the $PD||\Sigma U_i$ problem with agreeable conditions. Ng *et al.* (2003) introduced a negative approximation result of the $PD|d_i=d, p_{ik} \in \{0,1\}|\Sigma U_i$ problem. They also designed an LP-rounding algorithm with an error ratio of $d+1$ for the $PD|d_i=d, p_{ik} \in \{0,1\}|\Sigma w_i U_i$ problem. Ahmadi and Bagchi (1997) and Cheng *et al.* developed dynamic programming algorithms independently for $PDm||\Sigma w_i U_i$. Lin and Kononov (2006) have shown some negative approximation results for $PD2|d_i=d|\Sigma U_i$ and proposed an LP-based approximation algorithm for unweighted and weighted cases.

The problem to minimize the weighted number of tardy jobs subject to a common due date is equivalent to the multiple-dimensional 1-0 knapsack problem. The number of machines could be

viewed as the number of dimensions of the knapsack. The objective function can be transformed to selecting the items (jobs) to maximize the profits.

This chapter is organized as the following. The formulation of the studied problem is proposed in Section 2. In Section 3, we introduce a branch and bound algorithm. Approximation algorithms are presented in Section 4. The computational experiments will be shown in Section 5 and the conclusion remarks will be given in Section 6.

2. Problem Formulation

The mathematical formulation could help us define the problem more precisely. In this section, we propose a mathematical formulation to describe this problem. In 1992, Lasserre and Queyranne (1992) introduced an orignal formulation using positional variables , u_i^j. The positional variables could be used in most scheduling problems. If job i is scheduled in the j-th position of a sequence, then u_j^i = 1; otherwise, u_j^i = 0. In 1995, Dauzere-Peres (1995) used positional variables to formulate single machine scheduling to minimize the number of late jobs. Dauzere-Peres and Sevaux (1997) modified the previous formulation to remove the big-M variable so as to achieve a better effieicncy. The new formulation could deal with problems with 50 jobs. After some modifications, their formulation could be adpated to formulate the concurrent open shop problem with multiple machines. As we know, for the concurrent open shop problem, the sequence of orders is identical on every machine in at least one optimal solution. By this property, this formulation only requires n^2 positional variables, regardless of the number of machines invloved. The notations and formulation are described as follows:

t_{jk} : the time that the j-th job starts to be processed on machine k;

p_{ik} : the processing time of job i on machine k;

d_i : the due date of job i;

U_i : if job i is late, U_i = 1; otherwise, U_i = 0.

Minimize

$$\sum_{i=1}^{n} w_i U_i$$

Subject to

$$t_{j+1,k} - t_{jk} - \sum_{i=1}^{n} p_{ik} u_j^i \geq 0 , \; \forall j \tag{1}$$

$$t_{jk} + \sum_{i=1}^{n} (p_{ik} - d_i) u_j^i \leq 0 , \; \forall j \tag{2}$$

$$\sum_{i=1}^{n} u_j^i \leq 1 , \; \forall j \tag{3}$$

$$\sum_{i=1}^{n} u_j^i + U_i = 1 , \; \forall i \tag{4}$$

$$\sum_{i=1}^{n} d_i u_{j+1}^i \sum_{i=1}^{n} d_i u_j^i \leq 1 , \; \forall j \tag{5}$$

$$u_j^i \in \{0,1\} , \; \forall i,j$$

$$U_i \in \{0,1\}, \; \forall i$$

The objective function is to minimize the weighted number of late jobs. Constraints (1) enforce that the job scheduled in the (j+1)-th position cannot start before its preceding job is finished. Constraints (2) guarantee that the scheduled jobs must be completed before their due dates. Constraints (3) ensure that every job can only be scheduled at most once. Constraints (4) state that jobs are either scheduled early or late. Since there is at least one optimal solution that all on-

time jobs are scheduled by thier due dates in non-decreasing order, constraints (5) are added to reduce the solution space.

This formulation can be used to solve small-scale problems. However, for large-scale problems, this formulation might take a exceedingly long time to get the optimal solution.

3. Branch and Bound Algorithms

In this section, we introduce the branching scheme and a lower bound that will be used to develop branch-and-bound algorithms. For different problems, based on their properties, we may suggest different branching schemes. Sometimes, the selection of branching scheme is critical in branch and bound algorithms because different branching schemes may present different performances.

For forward sequential branching scheme, the accumulation of objective value is lingering. At the beginning, the objective value of each node could be identical. One of the crucial properties in this problem is that no matter what the sequence is the makespan is fixed. Since the makespan is fixed and the due dates of each job are known, using backward sequential branching scheme, the exact current objective value could be calculated. The accumulation of objective value is faster than the former one in the earlier stage.

Another observation is that the early jobs should be scheduled, without idle time inserted, in the non-decreasing order of their due date before the tardy jobs. Suppose there are two jobs i and j, $d_i > d_j$ but job i precedes job j. it is clear that we can interchange the two jobs without increasing the objective value. Therefore, using the backward branching scheme, once an early job occurs, we could assume the rest jobs are scheduled early by the non-decreasing order of their due date. Ford forward branching scheme, this property could be view as a dominance rule.

Dominance: For any two early jobs i and j, if $d_i < d_j$, we say job i dominates job j and will precede job j.

To minimize the number of tardy jobs, the order of late jobs could be ignored. However, in branch and bound algorithms, each node represents an ordered partial solution. Therefore, there can be several solutions that are identical by the definition. To avoid such a situation, we only consider the situations that the tardy jobs are scheduled in the increasing order of their indices.

Dominance: For any two tardy jobs i and j, if $i < j$, job i dominates job j.

Each node in the branch-and bound tree represents a partial solution. The lower bound method is applied on each node to estimate the possible cost. If the existing and estimated cost is greater than the current best solution, the branching would be bounded.

Denote the partial schedule by P with the set of late jobs $L(P)$.. The current weighted number of late jobs is $\sum_{i \in L(P)} w_i$.

To minimize the number of late jobs on a single machine without considering release dates, we can schedule the jobs by the EDD rule. The rule arranges the jobs according to non-decreasing order of their due dates. Once a job is late, we identifies the scheduled job with the longest processing time and discards that. After considering all jobs, we could get the optimal solution.

If we apply the EDD rule on each machine for unscheduled jobs, we can get the minimum number of late orders on each machine. The maximum number among all machines is a lower bound on the number of late orders. Assume that the maximum number is l. Then, the sum of smallest l weights of unscheduled jobs is a lower bound of weighted tardy jobs. To prevent over-estimation of the weighted number of late jobs of rest jobs, the smallest I weights are used to calculate the lower bound.

4. Approximation Algorithms

Since this problem is NP-hard, it is unlikely to find an efficient algorithm. It might be acceptable to get an approximate solution in a reasonable time. In this section, we propose heuristic and tabu search algorithms. An efficient initial solution can reduce the time a meta-heuristic requires to converge. The heuristic method we propose is not only used to find an approximate solution but also to produce an initial solution for the tabu search algorithm.

4.1 Heuristic Method

We use the concept of the Hodgson-Moore algorithm (1968) to design our heuritic method. This algorithm could produce an optimal sequence for $1||\Sigma U_i$. The jobs are considered by the order of non-decreasing due dates. Once tardiness occurs, the scheduled job with longest processing time would be dropped. All early jobs precede tardy jobs.

First of all, we renumber the indices of all jobs in non-decreasing order of their due dates such that for any two jobs i, j if $i < j$, $d_i < d_j$. We schedule the jobs by their indices. Denote the partial schedule by P for j jobs being considered. There is no job late in P and the set of late jobs is $L(P)$. Next, we add job J_{j+1} into P and get a new schedule called P'. If there is no late job in P', we accept P' as our current partial schedule with the set of late job $L(P)$ and consider the next job J_{j+2}. If tardiness occurs on machine k in P', mark the job with smallest p_{ik}/w_i. Since this problem could be viewed as maximizing the weighted number of early jobs, the job with smallest p_{ik}/w_i contributes the least unit profits on machine k. If we remove the marked job in P' and J_{j+1} still remains late, mark the job with second smallest p_{ik}/w_i. Following the same procedure, there might be more than one job having to be marked. If the sum of weights of the marked jobs is less than w_{j+1}, then the former will be removed from P' to $L(P')$ and this schedule will be accepted as the current partial schedule; otherwise, P will be accepted and J_{j+1} will be included in $L(P)$. Following the same procedur, we could get a sequence.

4.2 Tabu Search

Tabu search method was first proposed by Glover (1989). It is a simple idea with excellent computational efficiency. The word, tabu, means the things that cannot be touched. It is a single agent meta-heuristic which gets one solution at each iteration. In each iteration, based on the current solution, it generates several neighborhood solutions. The agent selects the best solution among them and follows the same procedure. Tabu search method keeps track of the recent accepted solutions and will not accept such solutions in regulative iterations which is controlled by the tabu list size. The agent will not stop until the stopping criteria is satisfied. The stopping criteria can be the improvement ratio of the initial solution or the number of iterations in which the solution is not improved.

We set the tabu list size as 7 and 20 neighborhood solutions are generated randomly for each iteration. We use the heuristic method developed above to find the initial solution. If the current best solution is updated, we do the hill climbing procedure that is to find the order with largest weight among all late orders and interchange it and the order with least weight among all early jobs. If the weight of the late one is less than that of the early one or the solution has not been improved, we terminate the procedure. If the best solution is not updated within 1000 consecutive iterations, tabu search stops.

5. Computational Experiments

The experiment framework was designed from Fisher's experiment. The platform is personal computer with an Intel i586 CPU of 2.4GHz running Microsoft XP. The program is coded in C. The detailed information of experimental data is described below.

a). $p_{ik} \in [1,10]$;

b). $w_i \in [1,10]$;

c). $d_i \in [T \times (1-\tau-R/2), T \times (1-\tau+R/2)]$, where $T = \max_{m} \sum_{i=1}^{n} p_{im}$ and τ and R are the factors of due date.

The instances are generated from uniform distribution. For each problem size, we generated 20 instances randomly. The experiments consist of two parts. One is for small-scale problems solved by two different branching schemes, heuristic and tabu search method. The other is for large-scale problems solved by heuristic and tabu search method.

Due to the exponential growth of the solution space, the scale of the problem that can be solved by the branch and bound algorithms is quite limited. Table 1 summarizes the numerical results of small-scale test instances. Two branching schemes are compared by the number of nodes visited and the elapsed run time. From Table 1, we can see that the the backward branching scheme performs much better than the forward counterparyt. Numerical results also suggests that the lower bound is not tight, with deviations of 60% to 70% from the optimal solutions. Looking at the results of approximation algorithms, we know that tabu search performs very well in the small scale problems. The column entitled #_Opt contains the number of instances that have been optimally solved. For all test instances, the tabu search algorithm can find optiml solutions.

n	Forward B&B		Backward B&B		Lower Bound		Heuristic			Tabu Search		
	Time	Node	Time	Node	# of opt	Error (%)	# of Opt	Error (%)	Time	# of opt	Error (%)	
10	0.194	7.6E05	0.000	2342	4	61.250	6	65.00	0.016	20	0	
12	28.019	7.8E07	0.009	5.2E04	1	71.277	2	100.00	0.018	20	0	
14			0.100	8.4E05	5	57.333	2	133.33	0.021	20	0	
16			0.141	1.1E06	6	57.143	2	121.43	0.024	20	0	
18			1.327	1.3E07	4	68.932	2	133.01	0.026	20	0	

Table 1. B&B vs. Tabu Search

	Tabu Search	
n	Time (Sec.)	Improvement (%)
20	0.029	66.464
30	0.056	77.628
40	0.084	81.245
50	0.124	84.136
60	0.176	83.596
70	0.230	81.148
80	0.255	84.703
90	0.360	80.798
100	0.468	81.949

Table 2. Improvement by Tabu Search

Next, we examine the improvement achieved through the deployment of tabu search for large-scale problems. The heuristic is applied first to get an initial solution. The tabu search algorith is activated to start the impeovement phase from the initial solution. The results are
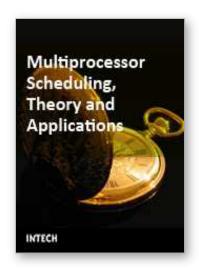
shown in Table 2. The run time required by the heuristic algorithm is niglegible. Although tabu search takes more time, it still remains within the reasonable executive time. However, the performance between this two method is quiet different. As the problem scale increases, the performance deviation between these two methods grows.

6. Concluding Remarks

In this chapter, we discussed the concurrent open shop problem $PD||\Sigma w_i U_i$. A mathematical formulation was given to describe the problem. We proposed a lower bound and studied the performance of different branching schemes for branch-and-bound algorithms. The branching schemes play an important role in this problem. To produce approximate solutions in a reasonable time, we proposed a heuristic and a tabu search algorithm. Computational experiemnts suggest that the tabu search algorithm is efficient and effective in the sense that it can produce quality solutions in an acceptable short time.

For future work, Lagrangian relaxation might be an alternative way to getting a tighter lower bounds and approximate soltuions. Equipping the concurrent open shop model with other constraints, such as precedence relation s, can be an interesting direction.

7. References

R.H. Ahmadi and U. Bagchi (1990), Scheduling of multi-job customer orders in multi-machine environments, *ORSA/TIMS*, Philadelphia.

R.H. Ahmadi and U. Bagchi (1997), Coordinated scheduling of customer orders, *Updated Working Paper,* Anderson School at UCLA.

T.C.E. Cheng, Q. Wang and J. Yuan (2006), Customer order scheduling on multiple facilities, *Working paper*.

S. Dauzere-Peres(1995), Minimizing late jobs in the general one machine scheduling problem, *European Journal of Operational Research*, 81(1), 134-142.

S. Dauzere-Peres and M. Sevaux (1997), An efficient formulation for minimizing the number of late jobs in single-machine scheduling, *ETFA*, LA, USA.

M.L. Fisher (1976), A dual algorithm for the one-machine scheduling problem, *Mathematical Programming*, 11:229-251.

F. Glover (1989), Tabu search – Part I, *ORSA Journal on Computing*, 1(3):190-206.

F. Glover (1989), Tabu search – Part II, *ORSA Journal on Computing*, 2(1):4-32.

R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnoy Kan (1979), Optimization and approximation in deterministic, sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, 5:287-326.

J.B. Lasserre and M. Queyranne, Generic scheduling polyhedral and a new mixed integer formulation for single machine scheduling, *in Proceedings of the Second IPCO Conference*, Carnegie-Mellon University, Pittsburgh, 1992.

C.Y. Lee, T.C.E. Cheng and B.M.T. Lin (1993), Minimizing the makespan in three-machine assembly type flow shop problem, *Management Science*, 39:616-625.

J.Y.-T. Leung, H. Li and M. Pinedo (2006), Scheduling orders for multiple product types with the due date related objectives, *Discrete Optimization*, 168:370-389.

B.M.T. Lin and A.V. Kononov (2007), A note on customer order scheduling to minimize the number of late jobs, *European Journal of Operations Research*, 183:944-948.

Multiprocessor Scheduling, Theory and Applications

Edited by Eugene Levner

ISBN 978-3-902613-02-8

Hard cover, 436 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2007

Published in print edition December, 2007

A major goal of the book is to continue a good tradition - to bring together reputable researchers from different countries in order to provide a comprehensive coverage of advanced and modern topics in scheduling not yet reflected by other books. The virtual consortium of the authors has been created by using electronic exchanges; it comprises 50 authors from 18 different countries who have submitted 23 contributions to this collective product. In this sense, the volume can be added to a bookshelf with similar collective publications in scheduling, started by Coffman (1976) and successfully continued by Chretienne et al. (1995), Gutin and Punnen (2002), and Leung (2004). This volume contains four major parts that cover the following directions: the state of the art in theory and algorithms for classical and non-standard scheduling problems; new exact optimization algorithms, approximation algorithms with performance guarantees, heuristics and metaheuristics; novel models and approaches to scheduling; and, last but least, several real-life applications and case studies.

INTECH
open science | open minds