A Concrete Model for Job Shop Scheduling

Abstract

A concrete model, which is called wood plate model, for solving the minimum make span problem of job shop scheduling is presented. Based on this model, a new basic algorithm is proposed. Some definitions and theorems about this basic algorithm are given in this paper.

Introduction

The job shop scheduling problem, a famous NP-hard problem, is firstly presented by Bowman (Bowman 1959). The definition of the job shop scheduling problem discussed in this paper is as following. Some jobs are to be processed by some machines. Every job is to be processed by every machine exactly once. The processing of a job on a machine is called an operation, which can not be interrupted, and whose duration is a given constant. The problem is to determine the start time of all the operations (i.e. to generate a schedule) with the objective of minimizing the time span (called makespan) from the start time of the operation whose start time is the first one to the end time of the operation whose end time is the last one, subject to two constraints that: (i) each job can be processed by at most one machine at any time. The processing of each job is composed of a prescribed sequence of operations. The operations of the same job have to be performed one by one. (ii) Each machine can process at most one job at any time. The first constraint is job constraint. The second constraint is machine constraint.

The formalized definition of the job shop scheduling problem is as following. Let: $J = \{J_1, \cdots, J_n\}$ denote the set of jobs; $M = \{M_1, \cdots, M_m\}$ denote the set of machines; $V = \{0,1, \cdots, r+1\}$ denote the set of operations, where 0 and $r+1$ represent the dummy "start" and "finish" operations, respectively; A the set pairs of operations constrainted by the precedence relations presenting condition (i) above; V_k the set of operations to be performed by machine M_k; $E_k \subset V_k \times V_k$ the set of pairs of operations to be performed on machine M_k and therefore cannot overlap in time; p_j the (fixed) duration or processing time; t_j the (variable) start time of operation j; We have $p_j > 0$ for $1 \le j \le r$, and $p_0 = p_{r+1} = 0$. The problem can be stated as: minimize t_{r+1}, subject to

$$\begin{array}{ll} t_j - t_i \ge p_i & (i,j) \in A, \\ t_j - t_i \ge p_i \vee t_i - t_j \ge p_j & (i,j) \in E_k,\ M_k \in M, \\ t_i \ge 0 & i \in V. \end{array} \quad (1)$$

Any feasible solution to (1) is called a schedule(Balas and Vazacopoulos 1998).

Job shop scheduling is known to be a difficult, strongly NP-complete problem (Garey and Johnson 1979). In fact, practical experience shows that it is among the hardest combinatorial optimization problems.

Several optimization algorithms and approximation algorithms have been proposed to solve job shop scheduling problem. The optimization algorithms usually proceed by branch and bound. For literatures on optimization algorithms for scheduling, see Lageweg et al. (Lageweg, Lenstra, and Rinnooy Kan 1977), Carlier and Pinson (Carlier and Pinson 1989), Applegate and Cook (Applegate and Cook 1991). While considerable progress has been made in this approach, however, practitioners still find such algorithms unattractive. They are time-consuming, and the size of problems that can be solved within a reasonable time limit is small (up to 100 operations). Moreover, their implementation requires a certain level of programmer sophistication.

On the other hand, approximation algorithms are a quite good alternative. Approximation algorithms may give optimal or near optimal solutions in a reasonable amount of computer time. Approximation algorithms, or heuristics, were first developed on the basis of dispatching rules (French 1982), or priority rules for choosing the next job to be "dispatched" (scheduled) in the process of generating a so-called active schedule. These procedures are very fast, but the quality of the solutions that they produce usually leaves plenty of room for improvement.

A more elaborate approach, which at a higher computational cost tends to produce considerably better approximations, is the Shifting Bottleneck Procedure of Adams et al. (Adams, Balas, and Zawack 1988), based on repeatedly optimizing the sequence on each individual machine, while keeping the sequence on all other machines fixed. If pursued until no more improvements can be obtained, this procedure yields a local optimum over the neighborhood defined by all those schedules obtainable from a given one by changing the sequence on any single machine.

More recently, some local search procedures (Aarts and Lenstra 1997) (Spachis and King 1979) were developed and produce better solutions than Shifting Bottleneck Procedure does. Starting from an initial feasible solution, a local search method iteratively searches the best solution among those in the neighborhood, i.e., in the set of feasible solutions "near" to the current solution. Tabu search (Glover, Taillard, and de Werra 1993) (Nowicki and Smutnicki 1996) is a local search procedure that relies on specialized memory structures to avoid entrapment in local optimum and achieve an effective balance of intensification and diversification. Simulated annealing (Laarhoven, Arts, and Lenstra 1992) (Kolonko 1999), a computational effective method for job shop scheduling, is also based on local search framework.

There are other interesting methods like geometric approach (Shmoys, Stein, and Wein 1991), job insertions (Werner and Winkler 1992), genetic approach, and so on.
However, better solution available in the literature is achieved by some hybrid approximation algorithms embed local search or tabu search into a shifting bottleneck framework.
Lastly, in the present paper, a concrete model, wood plate model, is presented. Based on this model, a basic algorithm, which is called dual front greedy algorithm, is proposed. Finally, some theorems about this basic algorithm are given in this paper.

Concrete Model

A concrete model for the job shop scheduling problem is presented in this paper. This concrete model can be stated as following. Given finite rectangular wood plates and finite closed bottom baskets, these wood plates are divided into finite classes. Each class of wood plates are covered with the same pattern. In the middle of each wood plate, there is a number, which is surrounded by a circle, and which is the serial number of the basket in which this wood plate will be exactly put. The problem is to put all the wood plates into the baskets with the objective of minimizing the whole distance from the bottom of the wood plate whose bottom is the lowest one to the top of the wood plate whose top is the highest one, subject to two constraints that: (i) Near the bottom of each wood plate, there is a number, which denotes the serial number of this wood plate in its class. For every two wood plate with the same pattern, in other words, these two wood plates belong to the same class, the top of the wood plate whose serial number is smaller should be lower than or equal to the bottom of the wood plate whose serial number is bigger, after these two wood plates have been put into the baskets. (ii) For every two wood plates which have been put into the same basket, the top of one wood plate should be lower than or equal to the bottom of another wood plate.

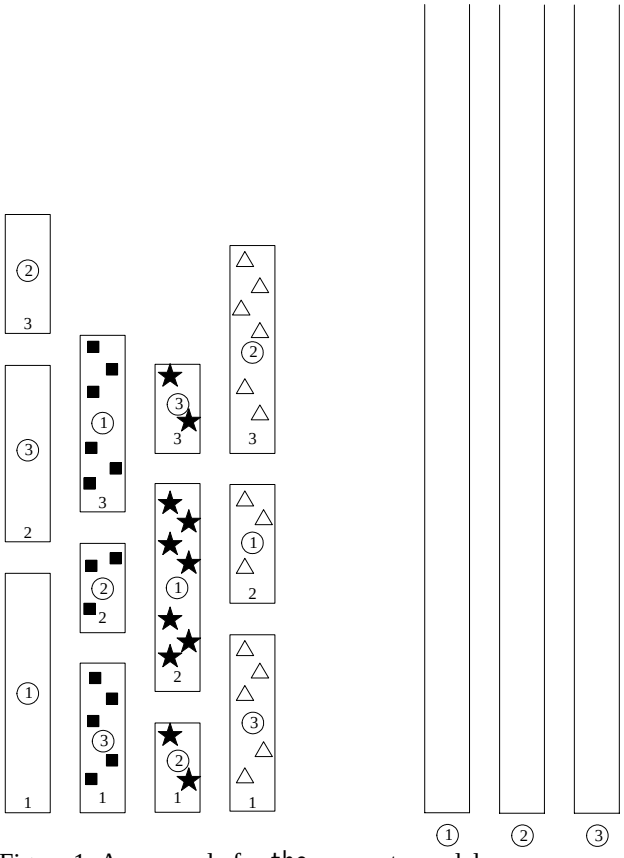

Figure 1: An example for the concrete model

The following example (Example 1) illustrates the quasi-physical model, which is shown in Figure 1.
Example 1 Given a problem instance, there are 12 wood plates which are covered with four patterns and 3 baskets in this problem instance.
A solution to this problem instances is shown in figure 2.

Dual front greedy algorithm

In this paper, the process of generating a schedule is as following. Firstly, the start times of all operations are not assigned. Then the start times of the operations are assigned one by one. Finally, the start times of all operations are assigned so as to generate a schedule.
To illustrate the process of generating a schedule clearly, the definition of "situation" is given.
Definition 1 situation The start times of some operations are assigned, while the start times of other operations are not assigned. This is called a situation.
The process of generating a schedule can be illustrate by the evolution of situations. ① Initial situation. In the initial situation, the start times of all the operations are not assigned. ② When an old situation is developing into a new situation, firstly an operation is chosen out, then the start time of this operation is assigned. By this means the start times of all operations are assigned one by one, so as to generate a schedule and evolve into the final situation. ③ Final situation. In the final situation, the start times of all operations are assigned.
According to the definition of the job shop scheduling problem, there are prescribed precedence relations among the operations which belong to the same job. So the definition of immediate job predecessor and immediate job successor is given as following.
Definition 2 immediate job predecessor and immediate job successor Suppose $(i, j) \in A$, operation i is called the immediate job predecessor of j, j the immediate job successor of i.
For each job, all operations have exactly one immediate job predecessor except for the first operation which has not any immediate job predecessor, and all operations have

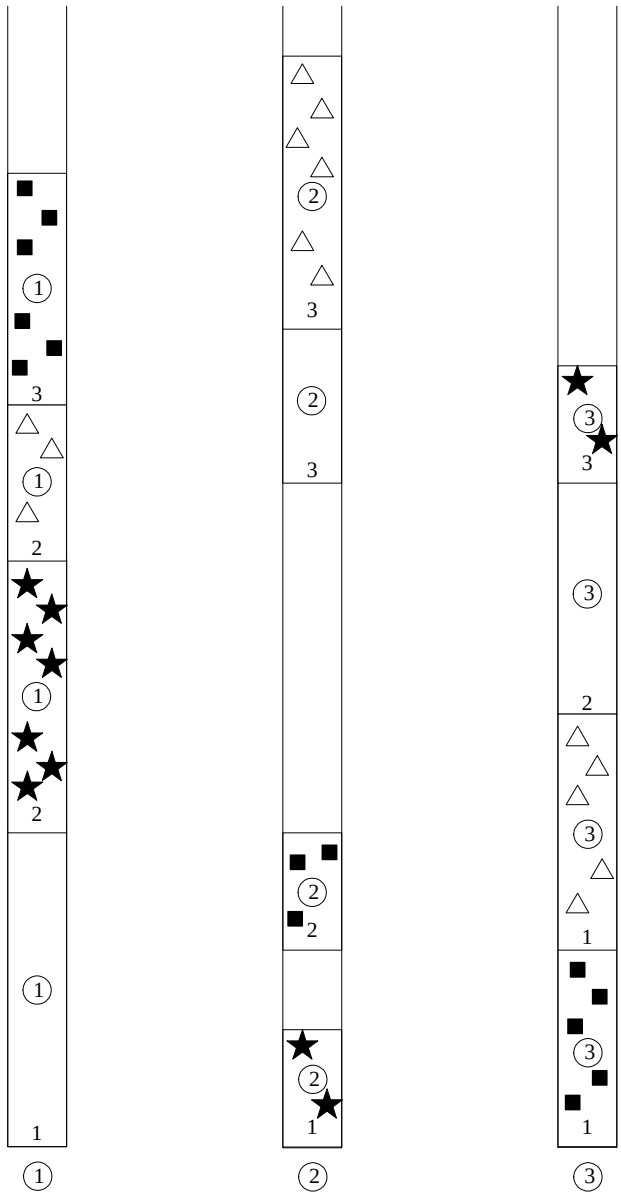

Figure 2: A solution

exactly one immediate job successor except for the last operation which has not any immediate job successor.

Suppose $(i, j) \in A$, the algorithm in this paper only assigns t_j when t_i has been assigned. So we give the definition of "job front operation".

Definition 3 job front operation Operation i and operation j are two operations that belong to job l, $(i, j) \in A$. In current situation, t_i has just been assigned, t_j has not yet been assigned, then operation j is called the job front operation of job l in current situation.

Let H_k denote a sequence of the operations on machine M_k. To be more precisely,

$H_1 = o_{11}, o_{12}, \cdots, o_{1n}$ is a sequence of the operations on machine M_1,

……

$H_k = o_{k1}, o_{k2}, \cdots, o_{kn}$ is a sequence of the operations on machine M_k,

……

$H_m = o_{m1}, o_{m2}, \cdots, o_{mn}$ is a sequence of the operations on machine M_m.

$H = (H_1, \cdots, H_k, \cdots H_m)$ is the sequence on all machines.

Definition 4 immediate machine predecessor and immediate machine successor Given $H = (H_1, \cdots, H_k, \cdots H_m)$ the sequence on all machines, $H_k (1 \le k \le m)$ a sequence of the operations on machine M_k, if i and j are two adjacent operations in H_k, and the position of i in H_k is before j, then operation i is called the immediate machine predecessor of j, j the immediate machine successor of i.

Given $H = (H_1, \cdots, H_k, \cdots H_m)$ the sequence on all machines, all operations on machine $M_k (1 \le k \le m)$ have exactly one immediate machine predecessor except for the first operation in H_k which has not any immediate machine predecessor, and all operations have exactly one immediate machine successor except for the last operation in H_k which has not any immediate machine successor.

Symmetrical to the definition of job front operation, there is the definition of machine front operation. Suppose that i is the immediate machine predecessor of j, the algorithm in this paper only assigns t_j when t_i has been assigned. So we give the definition of "machine front operation".

Definition 5 machine front operation Given $H = (H_1, \cdots, H_k, \cdots H_m)$ the sequences on all machines, suppose that i and j are two operations to be performed by machine M_k, and that i is the immediate machine predecessor of j. In current situation, t_i has just been assigned, t_j has not yet been assigned, then operation j is called the machine front operation of machine M_k in current situation.

Definition 6 dual front operation In current situation, if operation i is both job front operation of job $J_l (1 \le l \le n)$ and machine front operation of machine $M_k (1 \le k \le m)$, then i is a dual front operation.

Definition 7 the earliest possible start time of a dual front operation Suppose that operation j is a dual front operation in current situation. The earliest possible start time of j is defined as the smallest possible t_j that satisfies the following three constraints. ① If j has a immediate job predecessor i, then $t_j \ge t_i + p_i$. If j has no immediate job predecessor, then this constraint need no consideration. ② $t_j \ge 0$. ③ If j and i are to be processed on the same machine and if t_i has been assigned, then either $t_j \ge t_i + p_i$ or $t_i \ge t_j + p_j$.

The dual front greedy algorithm can be stated as following. Given $H = (H_1, \cdots, H_k, \cdots H_m)$ the sequence on all machines, the calculating process of dual front greedy algorithm is the evolution of situations. ① Initial situation. In the initial situation, the start times of all the operations are not assigned. ② When an old situation is developing into a new situation, firstly an operation j is chosen uniformly at random from the intersection of the set of job

front operations and the set of machine front operations, i.e. j is a dual front operation, then the earliest possible start time of j is computed out, finally j is assigned to start at its earliest possible start time. (This is a kind of greedy strategy.) ③ Final situation. There are exactly two kinds of final situations. The first kind of final situation is that the start times of all operations have been assigned. This is called final situation of success. The second kind of final situation is there are still some operations whose start times have not been assigned, but the intersection of the set of job front operations and the set of machine front operations is empty set. This is called final situation of failure.

Theoretical result

Theorem Given an instance of the job shop scheduling problem, $H=(H_1,\cdots,H_k,\cdots H_m)$ the sequence on all machines, the dual front greedy algorithm can generate a solution. In other words, $H=(H_1,\cdots,H_k,\cdots H_m)$ is corresponding to this solution. If all of the sequences on all machines are enumerated, then each sequence on all machines is corresponding to a solution. The best solution of these solutions is the optimal solution to this instance.
Proof (omitted)

Conclusion

In this paper, a concrete model for the job shop scheduling problem is discussed. Based on this model, a basic algorithm is presented. The theoretical result about this basic algorithm is also proposed.

References

Aarts, E., and Lenstra, J.K. 1997. *Local Search and Combinatorial Optimization*, Wiley: New York.

Adams, J.; Balas E.; and Zawack D. 1988. The shifting bottleneck procedure for job shop scheduling. *Management Science* 34 (3): 391-401.

Applegate, D. and Cook, W. 1991. A computational study of the job shop scheduling problem. *ORSA Journal on Computing* 3 (2): 149-156

Baker, K. 1974. *Introduction to Sequencing and Scheduling*, Wiley: New York.

Balas, E. 1969. Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm. *Operations Research*, 17 (8): 941-957.

Balas, E. and Vazacopoulos, A. 1998. Guided Local Search with Shifting Bottleneck for Job Shop Scheduling, *Management Science*, 44 (2): 262-275.

Bowman, E.H. 1959. The Scheduling Sequencing Problem. *Operations Research*, 7(5): 621-624

Carlier, J. and Pinson, E. 1989. An Algorithm for Solving the Job-Shop Problem, *Management Science* 35 (2): 164-176.

French, S. 1982. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop*, Wiley: New York.

Garey, M.R. and Johnson, D.S. 1979. *Computers and Intractability*, W. H. Freeman and Company: San Francisco.

Glover, F.; Taillard, E.; and de Werra, D. 1993. A Users Guide to Tabu Search. *Annals of Operations Research* 41 (1): 3-28.

Kolonko, M. 1999. Some New Results on Simulated Annealing Applied to the Job Shop Scheduling Problem. *European Journal of Operational Research* 113 (1): 123-136.

Lageweg, B.J.; Lenstra, J.K.; and Rinnooy Kan, A.H.G. 1977. Job-Shop Scheduling by Implicit Enumeration. *Management Science* 24(4): 441-450.

Nowicki, E.; and Smutnicki, C. 1996. A Fast Taboo Search Algorithm for the Job Shop Problem. *Management Science* 42 (6): 797-813.

Pezzella,F.; and Merelli,E. 2000. A Tabu Search Method Guided by Shifting Bottleneck for the Job Shop Scheduling Problem. *European Journal of Operational Research* 120 (2): 297-310.

Shmoys, D.B.; Stein, C.; and Wein, J. 1991. Improved Approximation Algorithms for Shop Scheduling Problems. *SIAM Journal on Computing* 23(3): 617-632.

Spachis, A.S.; and King, J.R. 1979. Job-Shop Scheduling Heuristics with Local Neighborhood Search. *International Journal of Production Research* 17(6): 507-526.

Taillard, E. 1993. Benchmarks for Basic Scheduling Problems. *European Journal of Operational Research* 64(2): 278-285.

Van Laarhoven, P.; Arts, E.; and Lenstra, J. 1992. Job Shop Scheduling by Simulated Annealing. *Operations Research* 40(1):113-125.

Werner, F.; and Winkler, A. 1995. Insertion Techniques for the Heuristic Solution of the Job Shop Problem. Discrete Applied Mathematics 58(2): 191-211.