JOB SHOP SCHEDULING USING METAHEURESTIC SEARCH TECHNIC – TABU SEARCH

K.Anandapadmanabhan MCA., M.Phil., Head of Computer Science, Sri Vasavi College,(Self-FinanceWing),Erode.Mob: 9842895257 E.Mail : apn1975@hotmail.com

Research Scholor, Category – B, BharathiarUniversity Coimbatore.

----------ABSTRACT----------

ABSTRACT

Basic Job Shop Scheduling Problem (JSSP) is a static optimization problem, since all information about the production program is known in advance. General Job shop problem is the probably most studied one by academic research during the last three decades and is notoriously difficult problem to solve. The JSSP is an NP (Nondeterministic Polynomial) hard problem and among those optimization problems, it is one of the least tractable known problems (Garey and Johnson 1979). It is purely deterministic, since processing time and constraints are fixed and no stochastic events occur. The JSSP also illustrates some of the demands required by a wide array of real world problems. In a shop floor, machines process jobs and each job contains a certain number of operations. Each operation has its own processing time and has to be processed on a dedicated machine. Each job has its own machine order and no relation exists between machine orders of any two jobs. For each job, the machine order of operations is prescribed and is known as technological production recipe or technological constraints, which are static to a problem instance. Operations to be processed on one machine form an operation sequence for this machine. Minimum make span feasible schedules are obtained by permuting the processing order of operations on machines without violating the technological constraints.

Keywords - **Disjunctive graph , Job shop scheduling, Optimization, Tabu Search, Tabu Length.**

INTRODUCTION

Basic Job Shop Scheduling Problem (JSSP) is a static optimization problem, since all information about the production program is known in advance. General Job shop problem is the probably most studied one by academic research during the last three decades and is notoriously difficult problem to solve. The JSSP is an NP (Nondeterministic Polynomial) hard problem and among those optimization problems, it is one of the least tractable known problems (Garey and Johnson 1979). It is purely deterministic, since processing time and

Constraints are fixed and no stochastic events occur. The JSSP also illustrates some of the demands required by a wide array of real world problems. In a shop floor, machines process jobs and each job contains a certain number of operations. Each operation has its own processing time and has to be processed on a dedicated machine. Each job has its own machine order and no relation exists between machine orders of any two jobs. For each job, the machine order of operations is prescribed and is known as technological production recipe or technological constraints, which are static to a problem instance. Operations to be processed on one machine form an operation sequence for this machine. Minimum make span feasible schedules are obtained by permuting the processing order of operations on machines without violating the technological constraints.

This paper presented the application of the Tabu search to solve the job shop scheduling problems. The goal of the work was to gain some insight into the influence of dynamic Tabu length strategies. The Tabu length was changed dynamically during the construction of the solution. The new dynamic tabu length strategies were used to prevent the neighbors which keep the solution in local minima and also used to avoid cycling.

PROBLEM DEFINITION

Job shop scheduling problem consists of a set of jobs $J = \{1 \ldots n\}$, a set of machines $M = \{1 \ldots m\}$, where J_i denotes i^{th} job ($1 \leq i \leq n$) and M_j denotes j^{th} machine ($1 \leq j \leq m$). On the machines $M_1, M_2 \ldots M_m$, the jobs $J_1, J_2 \ldots J_n$ are to be scheduled. Let V be the set of all operations in all jobs. Each job J_i has a set of operations $o_{i1}, o_{i2}, \ldots o_{ik,}$ where k is total number of operations in the job J_i. Operation's precedence constraints are associated with each job and ensure that operation o_{ij} will be processed only after the processing of operation o_{ij-1} in a particular job i.

Generally, standard model of n jobs, m machines job shop is denoted by $n/m/\varphi/P/C_{max}$. The parameter φ is technological matrix denoting the processing order of machines for different jobs. The machine order for i^{th} job is given by φ_{ij} ($1 \leq j \leq m$), where j denotes j^{th} operation in i^{th} job. An example of the technological matrix φ can be represented as follows:

$$\varphi = \begin{pmatrix} M_2 & M_3 & M_1 \\ M_1 & M_2 & M_3 \\ M_3 & M_1 & M_2 \end{pmatrix}$$

Each row of the above matrix represents a job. For first job, first operation is performed on machine M_2, second operation is performed on machine M_3 and third operation is performed on machine M_1. Similarly, other jobs are executed on different machines.

Matrix P, denoting the processing time of different operations, is represented as follows:

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix}$$

where p_{ij} represents time of j^{th} operation of i^{th} job.

The technological matrix φ and the processing time matrix P are given as problem data. The processing order (machine sequence) for machine M_i is given by Π_{ik} ($1 \leq k \leq n$), where k denotes k^{th} operation to be processed on machine M_i. A solution to JSSP can be represented by a matrix Π denoting processing orders of all machines. For instance, one solution of the above problem is considered as follows:

$$\Pi = \begin{array}{c} \\ M_1 \\ M_2 \\ M_3 \end{array} \begin{array}{c} \begin{array}{ccc} o_1 & o_2 & o_3 \end{array} \\ \begin{pmatrix} J_2 & J_3 & J_1 \\ J_1 & J_2 & J_3 \\ J_3 & J_1 & J_2 \end{pmatrix} \end{array}$$

According to the above schedule, first operation of second job is scheduled on machine M_1, followed by second operation of third job and third operation of first job. Similarly other machines have schedules represented in second and third rows. Generally, subscript values denoting machine numbers in φ and job numbers in Π are given to formulate technological matrix and matrix representing a solution respectively. Processing unit of j^{th} operation of i^{th} job on a machine is denoted as o_{ij}. Each operation o has at most two direct predecessor operations, a job predecessor PJ_o and a machine predecessor PM_o. First operation of a machine sequence has no PM_o, and first operation of a job has no PJ_o. Similarly each operation has at most two direct successor operations, a job successor SJ_o and a machine successor SM_o. Last operation of a machine sequence has no SM_o and last operation of a job has no SJ_o. An operation o is called schedulable, if both, PJ_o and PM_o are already scheduled.

Let $r_{o_{ij}}$ be the starting time of j^{th} operation of i^{th} job. Completion time $C_{o_{ij}}$ for o_{ij} is calculated as in Equation (1.1)

$$C_{o_{ij}} = r_{o_{ij}} + P_{ij},$$
$$r_{o_{ij}} = \max(C_{PJ_{o_{ij}}}, C_{PM_{o_{ij}}}) \quad (1.1)$$

$r_{o_{ij}}$ are assigned by zero values for undefined $PJ_{o_{ij}}$ and $PM_{o_{ij}}$. After scheduling all operations, the make span C_{max} representing completion time of all operations is calculated as in Equation (1.2).

$$C_{max} = \max(C_{o_{ij}}) \quad \text{for all } o_{ij} \in \mathcal{V} \quad (1.2)$$

Main objective is to minimize C_{max} value with certain restrictions listed as follows:

- No two operations of one job may be processed simultaneously.
- A machine performs only one job at a time.
- Once an operation is initiated for processing, it will not be interrupted until its completion.
- An operation of a job can not be started until its previous operations of the same job are completed
- More than one operations of a job cannot be processed on a single machine.
- Jobs must wait for the next machine to be available.
- Machines may be idle within the schedule period.
- One job is independent with other jobs.

TABU SEARCH

Memory Concepts of Tabu Search

TS is based on the premise that problem solving must incorporate adaptive memory and responsive exploration. The adaptive memory feature enables TS approaches to the solution space economically and effectively. The emphasis on responsive exploration in TS derives from the fact that a bad strategic choice can yield more information than a good random choice. Hence, in a system that uses memory, a bad choice can provide useful clues about how the strategy, in which the choice is based, may profitably be changed. Responsive exploration integrates exploiting good solution features while exploring new promising regions. TS finds new and more effective ways of taking advantage of the mechanisms associated with both adaptive memory and responsive exploration. TS also employ memory structures to operate in four dimensions, which are called as recency, frequency, quality and influence.

Recency based memory are responsible to keep track of solution attributes that have changed during the recent past. Selected attributes, which occur in solutions recently visited, are labeled as tabu-active and are stored in tabu list, Frequency based memory maintains number of times, in which the selected attribute occurs and maintains a table of frequencies related to that attribute, which may be regarded as tabu, if its frequency is greater than a given threshold value (Scrich et al 2004). Diversification can be achieved using frequency based memory Taillard (1994). The quality based memory refers to the merit of solutions visited during the search. This type of memory identifies elements that are common to good solutions or to paths that lead to such solutions. Quality enforces incentive-based learning, in which inducements are provided to reinforce actions leading to good solutions and penalties are provided to discourage actions leading to poor solutions. These memory structures are flexible that allows the search to be guided in a multi-objective environment, in which more than one function is used to determine the goodness of a

particular search direction. The influence based memory measures the impact of the choices made during the search, not only on quality but also on structure. An additional level of learning is incorporated by recording information about the influence of choices on particular solution elements. However, it is clear that certain decisions have more influence than others as a function of the neighborhood of moves, which are employed and the way, in which this neighborhood is negotiated.

Memories in TS are also classified as explicit memory and attributive memory. Complete solutions including elite solutions visited during the search are recorded in explicit memory. Local search can be expanded by using the memorized elite solutions and their attractive neighbors. The attributive memory enforces a guiding method for TS and records information about solution attributes that change in moving from one solution to another.

Tabu List

TS algorithm uses a short-term memory to escape from local minima. The short-term memory is implemented as a *tabu list*, which is used to store forbidden moves and the use of a tabu list prevents from returning to recently visited solutions. Therefore, it prevents from endless cycling and forces the search to accept even uphill-moves. Length of the tabu list controls the memory of the search process. With a small tabu length, the search will concentrate on small areas of the search space, while a large tabu length forces the search process to explore larger regions. The tabu length can be varied during the search, leading to more robust algorithms.

Implementation of tabu list consisting of complete solutions is not practical, because of inefficiency in managing a list having complete solutions. Therefore, instead of the complete solutions, only the *solution components* involved in moves are stored in the tabu list. Tabu list is usually introduced for each type of solution component, because different types of move corresponding to different types of solution components can be considered. Tabu conditions are defined by different types of solutions together with the corresponding tabu lists and are used to filter the neighborhood of a solution and generate the allowed set.

Aspiration Criteria

Tabus are sometimes too powerful that they may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the searching process. To overcome this problem, *aspiration criteria are* defined. The intend of the aspiration criterion is to avoid bypassing moves, which lead to substantially better solutions. The aspiration mechanism, in some cases, is used to protect the search from the possibility that in a given state, all moves are tabu. In such cases, the aspiration function is set in such a way that at least one move fulfills its criterion, and its tabu status is removed. In other cases, the aspiration mechanism is set in such a way that, if a move with a big impact on the solution is performed, the tabu status of other *lower-influence* moves is dropped. The simplest and most commonly used aspiration criterion allows a move, even if it is tabu, if it results in a solution with an objective value better than that of the current best-known solution or the old one in tabu at the time, when existing moves cannot be carried out. Much more complicated aspiration criteria have been implemented by different research persons and successfully implemented (Hertz and de Werra 1991), but they are rarely used.

Variable Length of Tabu List

Basic role of tabu list is to prevent cycling. Fixed length tabus cannot prevent cycling (Glover 1989, Glover 1990). We can observe that if the length of the list is too small, cycling cannot be prevented and long size tabu creates many restrictions so as to increase the mean value of the visited solutions. An effective way of removing this difficulty is to use a tabu list with variable size according to the current iteration number. Length of the tabu list is initially assigned according to size of the problem and it will be decreased or increased during the construction of solution so as to achieve better exploration of the search space. Typically, if better solutions are discovered, the tabu list is shortened, while tabu list is lengthened when moves leading to worse solutions are taken. Main assumption behind this is that when a good solution is found, there may be more reasonable solutions within a few moves. In this implementation, when the current solution is better than the previous one, the tabu list is shortened by one move and when the current solution is worse than the previous one, the tabu list is lengthened by one move.

Termination Criteria

In theory, the search could go on forever, unless the optimal value of the problem at hand is known beforehand. In practice, obviously, the search has to be stopped at some point. The most commonly used stopping criteria in TS are given as follows:

(i) A fixed number of iterations.

(ii) After some number of iterations without an improvement in the objective function value.

(iii) When the objective function reaches a pre-specified threshold value. That is, termination procedures are generally used to stop the iteration process, if the quality of the procedures has crossed a certain value.

(iv) A fixed amount of CPU time.

Dynamic Nature of Tabu Search

Difference between short-term memory and long-term memory arises an important distinction in TS. Each type of memory has been accompanied by its own special strategies. However, modifying the neighborhood $N(S)$ of the current solution S is the ultimate effect of both types of memory. The modified neighborhood denoted by $N'(S)$ maintains a selective history of the states encountered during the search. Based on short-term memory, $N'(S)$ is a subset of $N(S)$, and tabu classification is used to serve for the identification of elements in $N'(S)$, which are excluded from $N(S)$. TS based on a short-term memory allows a solution S to be visited more than once, but the corresponding reduced neighborhood $N'(S)$ will be different each time. Based on long-term memory, $N'(S)$ may be

expanded to include solutions not ordinarily found in $N(S)$. Hence, revisiting previous neighborhood solution making repeatedly visiting only a limited subset of S does not arise. The method will identify an optimal or near optimal solution long before a substantial portion of S is examined.

A Template for Simple Tabu Search

A template for simple TS is designed by using elements given in previous sections in this chapter. The general structure of this template is given below:

Initialization
Construct initial solution S^*
Find $f(S^*)$
Set S^* to S
Set $f(S^*)$ to $f(S)$
Searching
While termination criterion not satisfied do
Let k be the total neighbors for S
Find the solution S_i $(1 \leq i \leq k)$
Set appropriate S_i to S
If $f(S) < f(S^*)$ Then
Set S to S^*
Set $f(S)$ to $f(S^*)$
EndIf
Store current S_i to Tabu
Delete oldest entry in tabu if necessary
EndWhile
Output
Print S^* and $f(S^*)$

Initialization phase finds initial solution S^* of the given problem and corresponding makespan $f(S^*)$. This phase also assigns S^* and $f(S^*)$ to the current solution S and $f(S)$ respectively. In the searching phase, the algorithm finds a set of neighbors for the current solution selects suitable neighbor and finds the solution of this neighbor. It stores the best solution and also updates the tabu list. The algorithm repeats the searching phase until a termination criterion is met and finally outputs the best solution found.

REFERENCES

[1] Job shop scheduling Benchmark, OR-Library, http://mscmga.ms.ic.ac.uk/jeb/orlib/jobshopinfo.html.

[2] J. P. Watson and J. Christopher Beck, "Problem Difficulty forTabu Search in Job-Shop Scheduling", Elsevier Science, 2002,vol. 21.

[3] J.B. Chambers and J. Wesley Barnes "Solving the Job Shop Scheduling Problem with Tabu Search" *IIE Transactions*, vol.995.

[4] S. G. Ponnambalam, Aravindan P and Rajesh S V, "A Tabu Search Algorithm for Job Shop Scheduling", International Journal of Advanced Manufacturing Technology, 2000, vol. 16.

[5] F. Geyik and I.H. Cedimoglu, "The Strategies and Parameters of Tabu Search for Job Shop Scheduling", Journal of Intelligent Manufacturing, 2004, vol. 15.

[6] S.Q. Liu, H.L.Ong, and K.M. Ng, "A fast tabu search algorithm for the group shop scheduling problem", Advances in Engineering Software, 2005, vol.36.

[7] Jen-Shiang Chen, Jason Chao-Hsien Pan and Chien-Kuang Wu ,"Hybrid tabu search for re-entrant permutation flow-shopscheduling problem", Expert Systems with Applications, 2008,vol.34.