International Journal of Soft Computing 2 (4): 531-537, 2007

A Tabu Search Algorithm for Job Shop Scheduling Problem with Industrial Scheduling Case Study

[1]P. Senthil Velmurugan and [2]V. Selladurai
[1]Department of Mechanical Engineering, Kongu Engineering College,
Erode-638 052, India
[2]Department of Mechanical Engineering, Coimbatore Institute of Technology,
Coimbatore-641 014, India

Abstract: The general Job Shop Scheduling (JSS) problem is of combinatorial in nature wherein it is difficult to find the optimal solution conventionally. In recent years, much attention has been made to solve these type optimisation problems using heuristic techniques such as Genetic Algorithm, Ant Colony Optimisation, Tabu Search, Simulated Annealing. This study presents a Tabu Search (TS) approach to minimize makespan for the JSS problem. The method uses dispatching rules to obtain an initial solution and searches for new solutions based on new neighbourhood based first-last strategy with dynamic tabu length. Several benchmark problems are tested using this algorithm for the best makespan and the obtained results are compared with benchmark values. Finally the TS algorithm has been tested for scheduling problem in automobile parts manufacturing industry.

Key words: Combinatorial optimisation, job shop scheduling, tabu search, algorithm, manufacturing industry

INTRODUCTION

The manufacturing industry is considered to be the prime contributor to the growth of the competitive market driven industrialised society. This industry must be efficient in all respect, otherwise there will be no sales for the products. In such industries, there is a need for well designed and efficient manufacturing systems which incorporate global business and product strategies as well as process and technological developments. Such manufacturing system relies on efficient, effective and accurate scheduling, which is a complex operation. Scheduling allocates the time when a particular task or activity is to be processed by a given resource in order to optimise the requirements set by the customer. Job Shop Scheduling (JSS) problem is one such important model in scheduling theory and because of its combinatorial nature it is difficult to find the optimal solution conventionally. In JSS problem, n jobs are processed to completion on m unrelated machines. The objective is to minimize the maximum completion time (makespan). This study presents a Tabu Search (TS) approach to minimize the makespan C_{max} for the JSS problem. TS is a meta-heuristic approach designed for finding an optimal or near optimal solution of combinatorial optimisations problems. It works in a deterministic way trying to model human memory processes.

Job shop scheduling problem: In JSS problem, jobs are to be processed on the machines with the objective of minimizing some functions of the completion times of the jobs subject to the constraints. The sequence of the machines for each job is prescribed, each machine can process only one job at a time and pre-emption is not allowed. A set of jobs has to be processed on a collection of machines and each job needs several consecutive operations (routing) before being completed. The condition of equal number of operations for all jobs is also ensured.

Representation models: A schedule is defined by a complete and feasible ordering of operations to be processed on each machine and in a job shop there are two main ways of graphically representing such an ordering

- Disjunctive graph
- Gantt chart

Disjunctive graph model: JSS scheduling problem can be represented by a disjunctive graph(Laarhoven *et al*., 1992). In this disjunctive graph a vertex represents an operation. The conjunctive arcs which are directed lines connect two consecutive operations of the same job. The

Corresponding Author: P. Senthil Velmurugan, Department of Mechanical Engineering, Kongu Engineering College, Erode-638 052, India

Table 1: Processing time and operation sequence for 4×3 instance

Job	Operation sequence (processing time)		
1	1 (2)	2 (3)	3 (4)
2	3 (4)	2 (4)	1 (1)
3	2 (2)	3 (2)	1 (3)
4	1 (3)	3 (3)	2 (1)

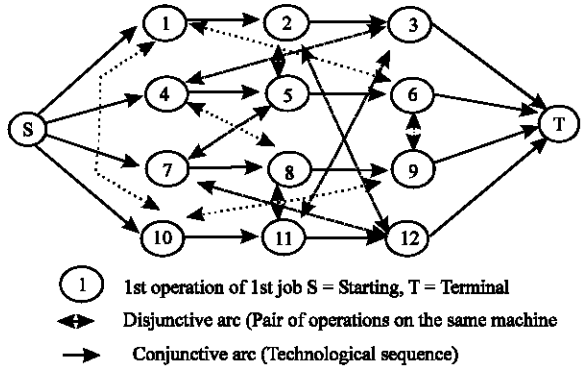

Fig. 1: Disjunctive graph representation for 4×3 problem in Table 1

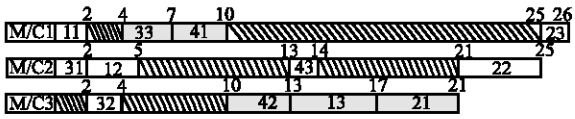

Fig. 2: Feasible Gantt Chart for the 4×3 problem with blocks

disjunctive arcs which are pairs of opposite directed lines connect a pair of operations belonging to different jobs to be processed on the same machine. Two additional vertices are drawn to represent the start and the end of a schedule. Let us consider an example of JSS problem with four jobs and three machines, the data are given in Table 1. The disjunctive graph representation for the above example problem is shown in Fig. 1.

Gantt charts: Gantt chart is the graphical representation of position of jobs and operations on the respective machines. It also represents idle times, starting and completion times of machines. In gantt chart, the various operation blocks are moved to the left as much as possible on each machine and this will help to have a compact schedule which will generally minimize the makespan measure. Figure 2 represents the gantt chart of the feasible solution for the problem in Table 1.

TABU SEARCH FOR JOB SHOP SCHEDULING

Glover (1989, 1990) presented the fundamental principles of TS as a strategy for combinatorial optimization problems. This study indicated the basic principles, ranging from the short-term memory process at the core of the search to the intermediate and long term memory process for intensifying and diversifying the search. This study also, introduced new dynamic strategies for managing Tabu lists, allowing complete exploitation of underlying evaluation functions.

Number of researchers adopted this TS technique for solving JSS. Taillard (1994) proposed parallel TS algorithm which is more efficient than the shifting of bottleneck procedure and simulated annealing for minimizing the makespan. Barnes and Chambers (1995) used dispatching rules to find the initial solution and introduced the concept of historical generators and search restart in conjunction with a contiguous spectrum of short term memory values to enhance the overall exploration strategy. The TS algorithm based on specific neighbourhood technique which employs a critical path and blocks of operations notions is proposed by Nowicki and Smutnicki (1996, 2005). They also provided a new approximation algorithm based on the big valley phenomenon that uses path relinking technique as well as new theoretical properties of neighbourhoods. Jain *et al.* (2000) used Nowicki and Smutnicki concept and developed a multi level hybrid system called core and shell based on the principle of scatter search and path relinking and concluded that the choice of strategy to generate a critical path has a negligible influence on the best solution. Ponnambalam *et al.* (2000) used the adjacent pair wise interchange method to generate neighbourhoods. The performance measure considered is makespan time and results are compared with simulated annealing and genetic algorithms. Pezzella and Merelli (2000) proposed the shifting bottleneck procedure to generate the initial solution and a dynamic list is introduced to refine the solutions. Armentano and Scrich (2000) presented a TS approach to minimize total tardiness for the JSS problem. The method uses dispatching rules to obtain an initial solution. Diversification and intensification strategies were incorporated to the short-term memory TS.

Tabu search procedure: TS Watson *et al.* (2003) is a local search algorithm that requires a starting solution and a neighbourhood structure. It proceeds by transiting from solution to solution using the transitions (or moves) defined by the neighbourhood structure in a systematic way until some stopping criterion is met. TS examine all the neighbours of the current solution and select the best admissible move. An admissible move is a move which is not on the tabu list, where the tabu list is a list containing forbidden moves. Here an initial solution is obtained

from any priority dispatching schedules. It is observed that, given a feasible solution the makespan can be improved by the exchange of two adjacent critical operations on the same machine. These insights are used to define a search neighbourhood. At each move the effect of each possible exchange is estimated. The reversal of an operation pair exchanged during the extent of short term memory is forbidden unless the resulting makespan would be better than the current one.

The TS (Geyik and Cedimoglu, 2004) implementation in connection with JSS problem is explained as follows.

Initial solution: The first step in the TS is to create the initial solution. An initial solution is obtained by selecting from any one priority dispatching solutions. The initial solution affects the scheduling solution quality. In this paper we have followed the Shortest Processing Time (SPT) rule.

Neighbourhood structure: A detailed historical sketch of neighbourhood was provided by Jain *et al.* (2000). The first major contribution was provided by Laarhooven *et al.* (1992). A neighbourhood structure is a mechanism, which contains new set of neighbour solutions by applying a simple modification to given solutions. Each neighbour solution is reached from a given solution by move. A sequencing move is defined by the exchange of certain adjacent critical operation pairs within the block and then considered the exchange of every adjacent critical operation pair on every machine. A block is a maximal sequence of adjacent critical operations on the same machine. Neighbourhood structure is directly effective on the efficiency of TS because TS proceeds iteratively from one neighbour to another in problem solution space. Therefore, a neighbourhood structure must eliminate unnecessary and infeasible moves if it is possible.

Neighbourhood strategy: In this proposed method only one critical path is generated. In the given blocks, swap the first two operations and last two operations on each block. In the study where the block contains only two operations, then these operations are swapped.

Move: The best neighbour which is not in Tabu or satisfies a given aspiration criterion is selected as a new seed solution. The best neighbour is one whose objective function C_{max} is minimum. If the entire neighbour is in Tabu or no neighbour satisfies the aspiration criterion then the oldest neighbour entering the tabu list at first is selected as new seed solution.

Tabu list: The purpose of the tabu list is to prevent the search from degenerating by starting to cycle between the same solutions. In tabu list, the elements added on the list are attributive to save computer memory. The tabu list is started with empty. The length of tabu list is important for selection of best solution. If the list is long more diversification is allowed due to which getting best solution is difficult. Contrary, if the list is too short cyclic occurs. The length of the tabu list is problem dependent and varies dynamically. It is evident that, the average number of explored solutions increases as the length of the tabu list increases (Pezzella and Merelli, 2000).

Aspiration criterion: The aim of the aspiration criterion, when it is necessary, is to override the Tabu status of a neighbour. The aspiration criterion is used as follows: if the move yields a solution better than the best obtained so far, then the move is performed even if it is in Tabu.

Termination criterion: When the number of disimproving moves reaches to a maximum set value and tabu length criteria is met or the best obtained solution equals the lower bound, then the TS algorithm is terminated.

Let us consider the example problem in Table 1. The TS algorithm is explained with this example problem. The initial solution is obtained by SPT rule. A feasible schedule of this problem is represented in Fig. 3.

The critical path for the above solution is {7→8→9→10→11→3→4→5→6} with makespan length of 26. The obtained solution is only one of the possible ones and its optimality for makespan is not guaranteed. Here, the obtained solution is stored as the current seed and the best solution. The critical path can be decomposed into a number of blocks. A block is a maximal sequence of adjacent critical operations that require the same machine. In Fig. 3, the critical path is divided into

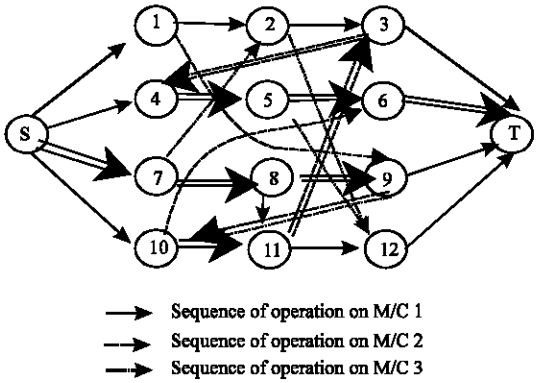

Fig. 3: Feasible sequence for the problem described in Table 1

Table 2: Tabu Search for problem in Table 1

Iter. No.	Neighbours	C_{max}	Move	Tabu list
1	10-9, 3-11, 4-3	21, 22, 22	10-9	9-10
2	4-3, 3-11, 10-1	17, 21, 21	4-3	9-10, 3-4
3	10-1, 4-11	16,17	10-1	9-10, 3-4, 1-10
4	11-8, 4-11	17, 17	11-8	9-10, 3-4, 1-10, 8-11
5	8-11, 4-8	16, 16	4-8	9-10, 3-4, 1-10, 8-11, 8-4
6	4-11, 6-9, 3-8	14, 18, 20	4-11	9-10, 3-4, 1-10, 8-11, 8-4, 11-4
7	5-12, 1-10, 12-2	13, 14, 16	5-12	9-10, 3-4, 1-10, 8-11, 8-4, 11-4, 12-5

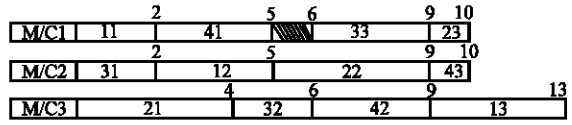

Fig. 4: Gantt chart for the optimum sequence

two blocks B1 and B2 that are processed on machines 1 and 3, respectively. Block B1 consists of the operations in nodes 9 (33), 10 (41) while B2 is made up of the operations 11(42), 3 (13), 4 (21). The sequence of operations on machines, are as follows: M_1 = {1→9→10→6}, M_2 = {7→2→12→5} and M_3 = {8→11→3→4}.

Using pair wise exchanges, the neighbourhood is generated for this schedule. For each schedule, makespan times are calculated and schedule which has minimum makespan time is taken as the initial schedule for the next iteration. The neighbours found are 10→9, 3→11 and 4→3. After reaching neighbour solution, sequencing move is done by swapping the critical operation pairs. The completion time of each neighbour is calculated. The makespan obtained for the neighbour 10→9, 3→11 and 4→3 are 21, 22 and 22, respectively. The move selects the neighbour 10→9 because its makespan value is minimum. This neighbour satisfies the aspiration criterion that, this neighbour is not in the tabu list. So, the neighbour 10→9 is added to the tabu list. The schedule for neighbour 10→9 is taken as the initial schedule for the next iteration. The process is repeated until a set termination criterion is met. The Table 2 exhibits the details about Tabu implementation and Fig. 4 represents the gantt chart for the optimum makespan 13.

TS IMPLEMENTATION

The initial schedule is created based on SPT rule, makespan is found and stored as best. Initially the tabu list is started with empty. The blocks are created based on critical path and neighbours are identified using first-last neighbourhood strategy. The pseudo algorithm for the procedure applied to find first and last neighbours is indicated in Fig. 5. Makespan is obtained for each pair with in the block which satisfies the above neighbourhood strategy by swapping its neighbour. Neighbours which have minimum makespan are stored as best which satisfies the aspiration criterion or not in tabu list. This new schedule acts as initial schedule for the next cycle. If the makespan value for new schedule is less than the existing makespan value, the new schedule is the best schedule. The above steps are repeated till it meets the termination criterion. By this way all the schedules are generated and the best result is found. The dynamic tabu length strategy is followed by us. When cycle occurs elite solution sequence was used as a new sequence and proceeds further. Initially the length of tabu list is set as eight and varied dynamically to a maximum of fourteen. The tabu list length is changed randomly at equal intervals of maxiter value, where maxiter represents the maximum number of allowed iterations if no improvement occurs in the current solution and is set as 5000. Number of elite solutions varies between 5 and 8.

```
TS Algorithm  : Find neighbours
Parameters :   k                  = Points the last neighbour
               noofmach           = The number of available machines
               numblock[i]        = The number of blocks in ith machine
               block[i][j].node[k] = Holds the posistion of kth  node in the
                                     jth block of ith  machine
               block[i][j].count  = Holds the total number of nodes in the
                                     jth  block of ith  machine
               neighbourlist[i].start = Holds the start node of ith  neighbour
               neighbourlist[i].end   = Holds the end node of ith neighbour
k=0;
for(m=0;m<noofmach;m++)
{
 for(l=0;l<=numblock[m];l++)
 {
      if(block[m][l].node[0]!=0)
      {
       assign block[m][l].node[0] to neighbourlist[k].start
       assign block[m][l].node[1] to neighbourlist[k].end
       k++;
       if(block[m][l].count>1)
       {
      assign block[m][l].nodes[block[m][l].count-1] to neighbourlist[k].start
      ssign block[m][l].nodes[block[m][l].count] to neighbourlist[k].end
      k++;
       }
      }
 }
}
```

Fig. 5: Pseudo algorithm for the procedure applied to find first and last neighbours

TS ALGORITHM

Step 1: Set cycle number as 0. Initialize tabu length and maxiter. Generate initial solution by using SPT rule and store it as current solution and best solution.

Step 2: Increase cycle number by 1. If the termination criterion is met, go to step 6.

Step 3: Find the critical path and list of neighbours for the current solution and sort neighbours according to the makespan.

Step 4:
1. Select the first neighbour from neighbour list. Repeat the steps 4.2-4.3 until aspiration criterion is met or all neighbours are examined.
2. If aspiration criterion is met, add this neighbour to tabu list and set it as best neighbour. Go to step 5.
3. If aspiration criterion is not met, select te next neighbour and go to step 4.2
4. If none of the neighbour is found, select the oldest neighbour from tabu list and set it as best neighbour.

Step 5: Find the current solution according to current best neighbour. If current solution is lower than the best solution, store current solution as best solution. Go to step 2.

Step 6: Output the best solution.

RESULTS AND DISCUSSION

The proposed TS algorithm is coded in C++ language on Linux platform with the configuration of AMD Athlon 2600+, 2GHz and 512 RAM. The performance of TS algorithm is tested on twenty nine JSS instances of different sizes (6×6, 10× 5, 15× 5, 20× 5, 10×10) which are available in the OR-library. The obtained makespan values for different instances are compared with best benchmark values available in OR Library and the results are presented in Table 3. The percentage relative error is also calculated for each instance.

INDUSTRIAL CASE PROBLEM

This case study describes a JSS problem encountered in a brake drum line of M/S sakthi auto components limited, erode located in southern part of India. In this machine shop, brake drums required for light motor vehicle wheels are machined. The brake drum (Fig. 6) is attached to the wheel and provides the rotating surface for the brake linings to rub against to achieve braking action. It is grooved to mate with a lip on the backing plate that provides the rotating seal to keep water and dirt from entering the brake assembly. In this brake drum line, special purpose machines are loaded with various jobs with pre-determined process sequence and time with continuous production. The processing times are functions of the handling time and machining time of the jobs.

Table 3: Comparison of makespan values

Test instances	Size	Best solution known	Obtained solution	RE (%)
LA 01	10×5	666	666	0.00
LA 02	10×5	655	655	0.00
LA 03	10×5	597	597	0.00
LA 04	10×5	590	590	0.00
LA 05	10×5	593	593	0.00
LA 06	15×5	926	926	0.00
LA 07	15×5	890	890	0.00
LA 08	15×5	863	863	0.00
LA 09	15×5	951	951	0.00
LA 10	15×5	958	958	0.00
LA 11	20×5	1222	1222	0.00
LA 12	20×5	1039	1039	0.00
LA 13	20×5	1150	1150	0.00
LA 14	20×5	1292	1292	0.00
LA 15	20×5	1207	1207	0.00
LA16	10×10	945	946	0.10
LA17	10×10	784	784	0.00
LA18	10×10	848	848	0.00
LA19	10×10	842	842	0.00
LA20	10×10	902	902	0.00
ORB01	10×10	1059	1064	0.47
ORB02	10×10	888	888	0.00
ORB03	10×10	1005	1005	0.00
ORB04	10×10	1005	1011	0.59
ORB05	10×10	887	887	0.00
ABZ05	10×10	1234	1236	0.16
ABZ06	10×10	943	943	0.00
FT 06	06×06	55	55	0.00
FT10	10×10	930	935	0.53

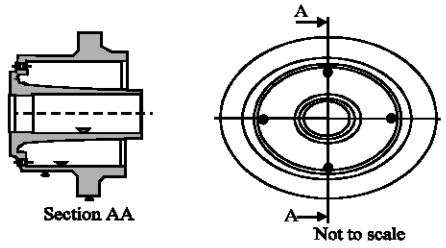

Fig. 6: Rear brake drum

The process sequence and number of operations are not the same for all brake drums. There are 6 different brake drums being machined in 7 different machines. The order in which the machines are required to process a job is called process sequence of that job. Though the operation number in the operation sequence of a job remains the same, the processing times for various jobs on a machine may differ. The operation and machine description to perform these operations are described in Table 4. The job description and sequence of operations are indicated in Table 5. The orders for these brake drums

Table 4: Operation Sequence with machine details

Operation No.	Operation description	Machine No.	Machine name
1	Outer diameter finishing	CN 09	CNC turning machine
2	Internal diameter finishing	CN 23	CNC turning machine
3	Grooving	GL 22	Grooving lathe
4	Drilling	DL 15	Radial drilling machine
5	Chamfering	DL 18	Radial drilling machine
6	Tapping	DL 19	Radial drilling machine
7	Honing	SM 10	Vertical spindle honing machine
8	Final inspection	CMM	Co-ordinate measuring machine
9	Cleaning and packing		

Table 5: Job description with operation sequence

Job no.	Job - break drum	Operation sequence						
A	Honda rear brake drum	1	2	3	4	5	6	7
B	Maruthi Model "B" rear brake drum	1	2	3	4	5	----	----
C	Maruthi Model "C" rear brake drum	1	2	6	3	4	5	7
D	Maruthi Gypsy Van front brake drum	1	2	5	3	4	----	----
E	Maruthi Gypsy Van rear brake drum	----	2	----	4	5	6	----
F	Maruthi 800CC rear brake drum	1	2	3	4	5	----	----

Table 6: Processing times and operation sequence for the Brake drum line

Job	Operation sequence (Processing time in min)						
A	1 (512)	2 (678)	3 (352)	4 (430)	5 (98)	6 (98)	7 (186)
B	1 (585)	2 (720)	3 (560)	4 (550)	5 (420)	(0)	(0)
C	1 (552)	2 (831)	6 (201)	3 (299)	4 (108)	5 (108)	7 (206)
D	1 (765)	2 (765)	5 (249)	3 (254)	4 (132)	(0)	(0)
E	(0)	2 (1170)	(0)	4 (411)	5 (213)	6 (411)	(0)
F	1 (423)	2 (1152)	3 (187)	4 (96)	5 (177)	(0)	(0)

are fluctuating with respect to time and it varies from manufacturer to manufacturer. Hence order data has been collected for six months time and the average batch size is taken to calculate the setup time and operation time.

After performing the machining operations the inspection for quality work is performed using co-ordinate measuring machine. Dusts and foreign materials are removed in the cleaning section. Before dispatching, the product is oiled to avoid rust and packed with dust free polythene cover.

Based on the above details the problem has been formulated as JSS problem and are shown in Table 6. The TS algorithm has been used to find the optimal machine sequence and total makespan time. The optimum makespan obtained with one set of machine sequence is 5776 min. It is found that the CNC turning machines CN 09 and CN 23 are fully utilised with zero idle time for the operations internal diameter finishing and outer diameter finishing. The optimum makespan is obtained in the 12th iteration itself. The different machine sequences can be optioned by varying TS parameters but with the same makespan value.

CONCLUSION

In this research, a TS algorithm is presented for solving JSS problem. The objective function considered is the minimisation of the makespan time. Schedules are represented using an operation-based representation. The initial solution has been obtained by SPT time rule. The new first-last neighbourhood structure with dynamic tabu length has been proposed. Software has been developed in C under Linux environment for the TS algorithm. The performance of proposed TS algorithm has been tested on different benchmark problems. The optimal solutions are found for all tested instances with less than one percent relative error. This proposed TS algorithm has also been implemented in automobile parts manufacturing industry to find the optimal machine sequence.

REFERENCES

Armentano, V.A. and C.R. Scrich, 2000. Tabu search for minimizing total tardiness in a job shop, Int. J. Prod. Econ., 63: 131-140.

Barnes, W.J. and J.B. Chambers, 1995. Solving the Job shop Scheduling Problem with Tabu Search, IEEE. Trans., 27: 257-263.

Chu, C., J.M. Proth and C. Wang, 1998. Improving job-shop schedules through critical pairwise exchanges, Int. J. Prod. Res., 36: 683- 694.

Dauzere-Peres, S. and Jan J. Paulli, 1997. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search, Ann. Operations Res., 70: 281-306.

Geyik, F. and I.H. Cedimoglu, 2004. The Strategies and Parameters of Tabu Search for Job Shop Scheduling, J. Intelligent Manufacturing, 15: 439-448.

Glover, F., 1989. Tabu Search-Part I, Operations Research Society of America. J. Comput., 1: 190-206.

Glover, F., 1990. Tabu Search-Part II. Operations Research Society of America, J. Comput., 2: 4-32.

Jain, A.S., B. Rangaswamy and S. Meeran, 2000. New and Stronger Job-Shop Neighbourhoods: A focus on the method of Nowicki and Smutnicki, J. Heuristics, 6: 457-480.

Laarhoven, V.P., E. Aarts and J. Lenstra, 1992. Job Shop Scheduling by Simulated Annealing, Operations Res., 40: 113-125.

Nowicki, E. and C. Smutnicki, 1996. A Fast Taboo Search Algorithm for the Job Shop Problem, Manage. Sci., 42: 797-813.

Nowicki, E. and C. Smutnicki, 2005. An Advanced Tabu Search Algorithm for the Job Shop Problem. J. Scheduling, 8: 145-159.

Pezzella, F. and E. Merelli, 2000. A Tabu Search Method guided by Shifting Bottleneck for the Job Shop Scheduling Problem, Eur. J. Operational Res., 120: 297- 310.

Ponnambalam, S.G., P. Aravindan and S.V. Rajesh, 2000. A Tabu Search Algorithm for Job Shop Scheduling, Int. J. Adv. Manufacturing Tech., 16: 765-771.

Taillard, D., 1994. Parallel Taboo Search Techniques for the Job Shop Scheduling Problem, ORSA. J. Comput., 6: 108-116.

Watson, J.P., J.C. Beck, A.E. Howe and L.D. Whitley, 2003. Problem Difficulty for Tabu Search in Job-Shop Scheduling. Artificial Intelligence, 143: 189-217.