Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 3229–3242
computers & operations research
www.elsevier.com/locate/cor

A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem

ChaoYong Zhang*, PeiGen Li, ZaiLin Guan, YunQing Rao

School of Mechanical Science & Engineering, Huazhong University of Science & Technology, Wuhan, 430074, PR China

Available online 2 February 2006

Abstract

Tabu search (TS) algorithms are among the most effective approaches for solving the job shop scheduling problem (JSP) which is one of the most difficult NP-complete problems. However, neighborhood structures and move evaluation strategies play the central role in the effectiveness and efficiency of the tabu search for the JSP. In this paper, a new enhanced neighborhood structure is proposed and applied to solving the job shop scheduling problem by TS approach. Using this new neighborhood structure combined with the appropriate move evaluation strategy and parameters, we tested the TS approach on a set of standard benchmark instances and found a large number of better upper bounds among the unsolved instances. The computational results show that for the rectangular problem our approach dominates all others in terms of both solution quality and performance.

Keywords: Job shop scheduling problem; Makespan; Heuristic; Tabu search

1. Introduction

The job shop scheduling problem with objective of minimizing makespan of the schedule, $J\|C_{\max}$, has a very wide engineering background. The problem can be briefly described as follows. There are a set of jobs and a set of machines. Each job consists of a sequence of operations, and each of the operations uses one of the machines for a fixed duration. Each machine processes at most one operation at one time. Once the operation started, no preemption is permitted. A scheduling is an assignment of operations to time intervals on the machines. The objective of the problem is to find a schedule which minimizes the makespan ($C_{\max}$), that is, the finish time of the last operation completed.

The JSP is among the hardest combinatorial optimization problems. Since it is an important practical problem, JSP has been studied by a significant number of researchers, and many optimization algorithms and approximation algorithms have been proposed. The optimization algorithms, which are mainly based on the B&B scheme such as Carlier and Pinson [1] and Brucker et al. [2], have been successfully applied in solving small instances. However, they could not solve instances larger than 250 operations in a reasonable time. On the other hand, approximation algorithms, which include priority dispatch, shifting bottleneck approach, meta-heuristic methods and so on, provide a quite good alternative for the JSP. Approximation algorithms were firstly developed on the basis of dispatching rules, which are very fast, but the quality of solutions that they provide usually leaves plenty of room for improvement. A more elaborate algorithm, which could produce considerably better approximations at a higher computational cost,

* Corresponding author. Tel.: +86 27 87556924; fax: +86 27 87543074.
E-mail address: zcyhust@sohu.com (C. Zhang).

0305-0548/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.12.002

is the shifting bottleneck approach proposed by Adams et al. [3]. More recently, the meta-heuristic methods, such as genetic algorithm (GA) [4], simulated annealing (SA) [5], tabu search (TS) [6,7], could provide the high-quality solutions with reasonable computing times and have captured the attention of many researchers. The relevant surveys can be seen from Vaessens et al. [8], Blażewicz et al. [9] and Jain and Meeran [10].

Within the class of meta-heuristic methods, Tabu search, initially proposed by Glover [11], seems to be one of the most promising methods for the job shop scheduling problem with the makespan criterion. Tabu search was firstly applied to the JSP by Taillard [12], whose main contribution was the use of the neighborhood structure introduced by Van Laarhoven et al. [5] and proposed a fast estimation strategy. Taillard observed that this algorithm has a higher efficiency for rectangular instances. Since then, researchers have introduced numerous improvements to Taillard's original algorithm, and the most important contributions include Nowicki and Smutnicki [7], Dell'Amico and Trubian [13], Barnes and Chambers [14] and Chambers and Barnes [15]. Among these individual TS methods, algorithm TSAB designed by Nowicki and Smutnicki [7] introduces the real breakthrough in both efficiency and effectiveness for the JSP. For example, it finds the optimal solution for the notorious instance FT10 within only 30 s on a now-dated personal computer. The *i*-TSAB technique of Nowicki and Smutnicki [16], which is an extension of their earlier TSAB algorithm, represents the current state-of-the-art approximation algorithm for the JSP and improves the majority of upper bounds of the unsolved instances.

With tabu search algorithms for the JSP being improved, different neighborhood structures and evaluation strategies have been proposed in order to make the search more effective and efficient. In this paper, by thoroughly studying the previous neighborhood strategies, we propose a new enhanced neighborhood structure to solve job shop scheduling problem by TS approach. Our approach was tested on a set of standard benchmark instances and found 54 better upper bounds for the unsolved instances. The remainder of this paper is organized as follows. Section 2 gives the representation of the job shop scheduling problem. In Section 3, the new neighborhood structure and the implementation of tabu search approach are provided. In Section 4, we firstly present the comparison of the different neighborhood structures and comparison of move evaluation strategies respectively, and then give the computational study on the benchmark problems. Conclusion is presented in Section 5.

2. Representation of the JSP

The job shop scheduling problem can be represented with a disjunctive graph introduced by Balas [17]. Let $J = \{1, 2, \ldots, n\}$ be the set of jobs, $M = \{1, 2, \ldots, m\}$ the set of machines. A disjunctive graph $G := (V, A, E)$ is defined as follows: V is $\{0, 1, 2, \ldots, \tilde{n}\}$ the set of nodes representing all operations where 0 and $\tilde{n}$ represent the dummy start and finish operations, respectively. A is the set of conjunctive (directed) arcs connecting consecutive operations of the same job, and E is the set of disjunctive arcs connecting operations to be processed by the same machine k. More precisely, $E = \bigcup_{k=1}^{m} E_k$, where E_k is the subset of disjunctive pair-arcs corresponding to machine k; each disjunctive arc of E can be considered as a pair of oppositely directed arc. The length of an arc $(i, j) \in A$ is p_i that denotes the processing time. The length of each arc$(i, j) \in E$ is either p_i or p_j depending on its orientation. Let us consider an example of the three jobs and three machines given in Table 1. This problem can be represented by a disjunctive graph shown in Fig. 1.

According to the Adams et al. [3] method, the graph G can be decomposed into the direct sub-graph $D = (V, A)$, by removing disjunctive arcs, and into m cliques $G_k = (V_k, E_k)$, obtained from G by deleting both the conjunctive arcs and the dummy nodes 0 and $\tilde{n}$. A selection S_k in E_k contains exactly one directed arc between each pair of oppositely directed arcs in E_k. A selection is acyclic if it does not contain any directed cycle. Moreover, sequencing machine k

Table 1
An example of three jobs and three machines

Job	(Machine sequence, Processing time)		
j1	(1,3)	(2,2)	(3,5)
j2	(1,3)	(3,5)	(2,1)
j3	(2,2)	(1,5)	(3,3)

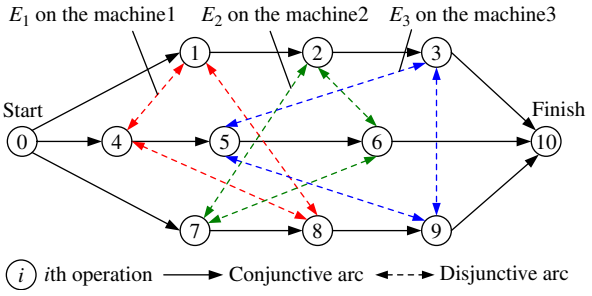

Fig. 1. The disjunctive graph of an instance with $n = 3$, $m = 3$, and $\tilde{n} = 10$.

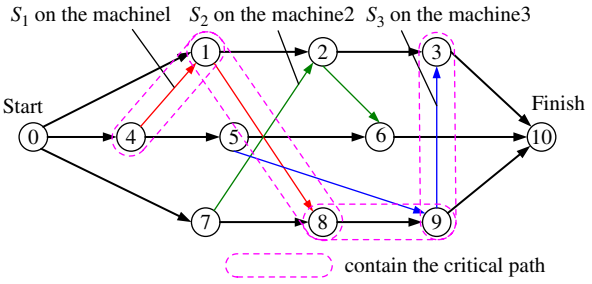

Fig. 2. A feasible solution for the disjunctive graph in Fig. 1.

means choosing an acyclic selection in E_k. A complete selection S consists of the union of selections S_k, one of each E_k, $k \in M$. A complete selection S, i.e., replacing the disjunctive arc set E with the conjunctive arc set S, gives rise to directed graph $D_s = (V, A \cup S)$; A complete selection S is acyclic if the digraph D_s is acyclic. An acyclic selection S defines a schedule, i.e., a feasible solution of problem. Fig. 2 represents a feasible solution for the disjunctive graph in Fig. 1. Furthermore, if $L(u, v)$ denotes the length of a longest path from u to v in D_s, then the makespan $L(0, \tilde{n})$ of the schedule is equal to the length of a longest path in D_s. Therefore, in the language of disjunctive graphs, to solve the job shop scheduling problem is to find an acyclic complete selection $S \subset E$ that minimizes the length of the longest (critical) path in the directed graph D_s.

A key component of a feasible solution is the critical path, which is the longest route from start to end in directed graph $D_s = (V, A \cup S)$ and whose length represents the makespan $C_{\max}$. Any operation on the critical path is called a critical operation. In Fig. 2 the length of the critical path is 19 and the critical path is $\{0, 4, 1, 8, 9, 3, 10\}$. It is also possible to decompose the critical path into a number of blocks. A block is a maximal sequence of adjacent critical operations that is processed on the same machine. In Fig. 2 the critical path is divided into two blocks, $B_1 = \{4, 1, 8\}$ and $B_2 = \{9, 3\}$. Any operation, u, has two immediate predecessors and successors, its job predecessor and successor denoted by $JP[u]$ and $JS[u]$ and its machine predecessor and successor denoted by $MP[u]$ and $MS[u]$. In other words, $(JP[u], u)$ and $(u, JS[u])$ are arcs of the conjunctive graph D_s, $(MP[u], u)$ and $(u, MS[u])$ (if they exist) are arcs of S. At the core of any local search algorithm such as tabu search is neighborhood, which briefly speaking, is a set of feasible solutions obtained by applying small perturbations to a given feasible solution. In the JSP, small perturbations are generally produced by re-ordering the sequence of operations on a critical path, and only through such re-ordering is it possible to produce a neighbor with a makespan better than that of the current solution.

3. Tabu search algorithm for the JSP

The tabu search, defined and developed primarily by Glover [18–20], has been successfully applied to a large number of combinatorial optimization problems, especially in production scheduling domain. TS is an enhancement of the well-known hill climbing heuristic, which use a memory function to avoid being trapped at a local minimum.

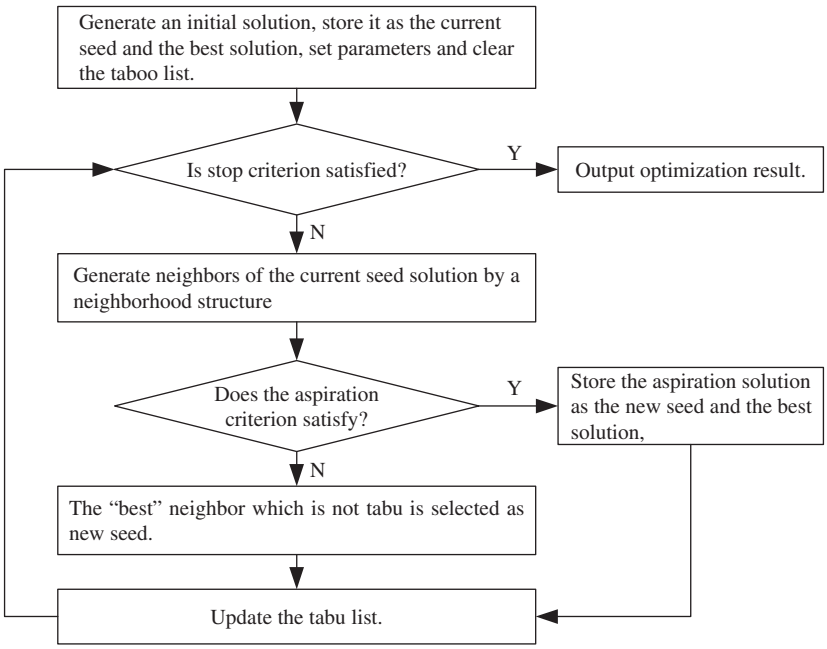

Fig. 3. Generic flowchart of tabu search algorithm.

The TS procedure is generally simple. The procedure starts with a feasible initial solution and stores it as the current seed and the best solution. The neighbors of the current seed are then produced by a neighborhood structure. These are candidate solutions. They are evaluated by an objective function, and a candidate which is the best not tabu or satisfies the aspiration criterion is selected as new seed solution. This selection is called a move and added to tabu list, another (the oldest one) move is removed from tabu list if it is overloaded. If the new seed solution is better than the current best solution, it is stored as new best solution. Iterations are repeated until a stop criterion is satisfied. Fig. 3 shows the procedure of tabu search algorithm. In the remainder of this section, a detailed description of our enhanced dynamic TS (TS_{ED}) is given.

3.1. Initial solution

The initial solution can be obtained by various methods such as the priority dispatching rules, the diverse insertion, the shifting bottleneck procedure and random methods. In general the initial solution methods have little influence on the solution quality provided by our algorithm, but they affect the running time. In this paper the search is initiated from the active solution randomly generated.

3.2. The new neighborhood structure

A neighborhood structure is a mechanism which can obtain a new set of neighbor solutions by applying a small perturbation to a given solution. Each neighbor solution is reached immediately from a given solution by a move [20]. Neighborhood structure is directly effective on the efficiency of TS algorithm. Therefore, unnecessary and infeasible moves must be eliminated if it is possible. The first successful neighborhood structure for the JSP was introduced by Van Laarhoven et al. [5], and is often denoted by N1[1] [9]. The N1 neighborhood is generated by swapping any adjacent pair of critical operations on the same machine, and based on the following properties:

- Given a feasible solution, the exchange of two adjacent critical operations cannot yield an infeasible solution.
- The permutation of non-critical operations cannot improve the objective function and even may create an infeasible solution.

[1] N1 is first named by Blazewicz et al. [9], and the following N4, N5 and N6 are similar.

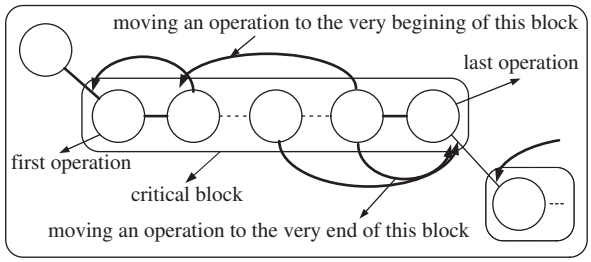

Fig. 4. The N4 neighborhood of moves.

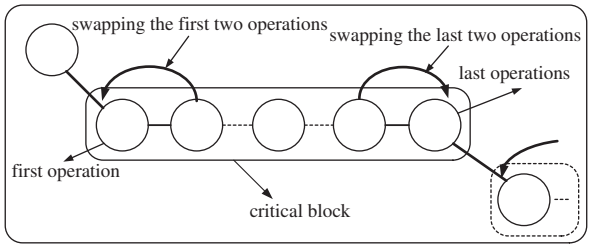

Fig. 5. The N5 neighborhood of moves.

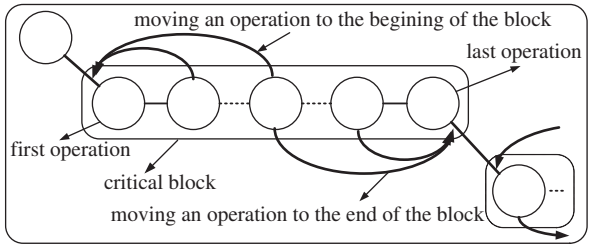

Fig. 6. The N6 neighborhood of moves.

However, the size of the neighborhood N1 is quite large and includes a great number of unimproved moves. An important observation, due to Matsuo et al. [21], is that unless the job-predecessor of u or the job-successor of v is on the critical path $P(0, n)$, the interchange containing u and v cannot reduce the makespan, i.e. swapping internal operations within a block never gives an immediate improvement on the makespan. Therefore, the further refined neighborhoods have been proposed by Dell'Amico and Trubian (N4)[2] [13], Nowicki and Smutnicki (N5) [7] and Balas and Vazacopoulos (N6) [22]. The neighborhoods N4, N5 and N6 are illustrated in Figs. 4–6, respectively. Those recently well-known neighborhoods are mainly based on the concept of block, in which a move is defined by inserting an operation to either the front or the rear of the critical block. The neighborhood N5 involves the reversal of a single border arc of a critical block[3] and is substantially smaller than the other neighborhoods, whereas the neighborhoods N4 and N6 involve the reversal of more than one disjunctive arc at a time and thus could nvestigate a considerably larger neighborhood. The neighborhood N6, which is also considered as an extension of the neighborhood N5, is more constrained than the neighborhood N4 and is currently one of the most effective and efficient neighborhood structures (see Table 2 in Section 4.1).

[2] Dell'Amico and Trubian [13] introduced two neighborhood structures N3 and N4; in this paper we only discuss the neighborhood N4.

[3] In this neighborhood only one critical path is generated. A move is defined by the interchange of two successive operations i and j, where i or j is the first or last operation in a block that belongs to a critical path. In the first block only the last two operations and symmetrically in the last block of the critical path only the first two operations are swapped.

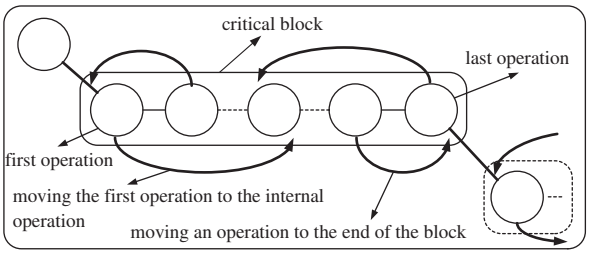

Fig. 7. The further extended neighborhood structure.

According to Matsuo et al. [21], in order to achieve an improvement by an interchange on u and v (assume that u is processed before v), either $JP[u]$ or $JS[v]$ must be contained on the critical path $P(0, n)$, that is, either u or v must be the first or last operation of a critical block. Therefore, a further extended neighborhood structure is proposed, in which a move is defined by inserting an operation to either the beginning or the end of the block, or moving either the first or the last operation of the critical block into the internal operation within the block (illustrated in Fig. 7). This leads to a considerably larger neighborhood and investigates a much wider space. However, the questions to be explored are under which conditions an interchange on critical operation u and v is guaranteed not to create a cycle and how to reduce the neighborhood size. Next we give the two theorems and proof for our neighborhood structure. Consider a feasible solution s:

Theorem 1. *If two operations u and v to be performed on the same machine are both on the critical path* $P(0, n)$, *and* $L(v, n) \geqslant L(JS[u], n)$, *then moving u right after v yields an acyclic complete selection.*

Proof. By contradiction: suppose moving u right after v create a cycle C. Then C contains either $(u, JS[u])$ or $(u, MS[v])$. If $(u, JS[u]) \in$ C, there is a path from $JS[u]$ to v in D_s (the cycle is $JS[u] \rightarrow v \rightarrow u \rightarrow JS[u]$), hence $L(JS[u], n) > L(v, n)$, contrary to assumption. if $(u, MS[v]) \in$ C, there is a path from $MS[v]$ to v in D_s, contrary to the assumption that D_s is acyclic. This completes the proof.

This Theorem 1 derives the idea that moving a critical operation u right after a critical operation v will not create a cycle if there is no directed path from $JS[u]$ to v in D_s. □

By analogy, we have Theorem 2.

Theorem 2. *If two operations u and v to be performed on the same machine are both on the critical path* $P(0, n)$, *and* $L(0, u) + p_\mu \geqslant L(0, JP[v]) + p_{JP[v]}$), *then moving v right before u yields an acyclic complete selection.*

Proof. Parallels that of Theorem 1.

In order to construct our neighborhood structure, we in fact extend the scope of the Propositions 2.2 and 2.3 proposed by Balas and Vazacopoulos and present the Theorems 1 and 2. The Theorems 1 and 2 could be applied to an interchange on any two critical operations u and v to be performed on the same machine, whether or not it contains either the $JS[u]$ or $JP[v]$ on the critical path. In the experiments, we observe that the neighborhood constructed by Theorems 1 and 2 is simpler and more constrained than the similar neighborhood N4. Therefore, our search neighborhood could be obtained from a feasible solution s by performing an interchange on two critical operation u and v as follows: (1) If a critical path $P(0, n)$ containing u and v also contains $JS[v]$, then insert u right after v; (2) If a critical path $P(0, n)$ containing u and v also contains $JS[v]$, then move v right before the first successive internal operations; (3) If a critical path $P(0, n)$ containing u and v also contains $JP[u]$, then insert v right before u; (4) If a critical path $P(0, n)$ containing u and v also contains $JP[u]$, then move u right after the first successive internal operations (see Fig. 5). Assume that an interchange on two operation u and v results in a makespan increase of the new schedule compared to the old one. Then it is obvious that every critical path in new schedule contains the arc (v, u) [22]. We could make use of the fact and reduce the neighborhood size further. □

3.3. Move evaluation

The run-time of TS algorithm for the JSP is typically dominated by the cost of computing each move. Therefore, in order to make the algorithm more efficiently, a number of neighborhood evaluation strategies, such as exact methods and estimation methods, have been proposed. The procedure suggested by Ten Eikelder et al. [23], called *bowtie*, is the most efficient exact method for the JSP, which only recalculates the head and the tail values of the operations that need to be updated after the move. However, this exact method still takes a very long computational time for the larger instances. In order to accelerate the search process, the estimation methods which can quickly filter out moves that have a high probability of directing the search to new elite solution have been proposed. A fast estimation approach has been presented by Taillard [6]. A further accurate approach of this strategy has also been proposed by Nowicki and Smutnicki [16], which is employed in the famous *i*-TSAB algorithm. But these estimation strategies are only adapted to swap the adjacent operations of the critical block. A significant estimation strategy proposed by Balas and Vazacopoulos [22] could be applied to reverse more than one disjunctive at a time, and the procedure of Balas and Vazacopoulos is also one of the most efficient implementations to solve the JSP.

In this paper, we compare the exact approach of Ten Eikelder et al. with the estimation approach of Balas and Vazacopoulos using our algorithm (see Section 4.2). The empirical testing in Section 4.2 shows that the estimation approach reduces the computational effort by 20 to 40% in comparison with the exact approach, and the efficiency of the search increases as the instances become larger. Intuition suggests that exact approach might perform better than the estimation approach in the solution quality, but empirical results indicate that the estimation approach is also able to achieve the lower MRE in comparison with the exact approach, such that this intuition remains unconfirmed.

3.4. Tabu list and tabu status of move

The purpose of the tabu list is to avoid the search process turning back to the solutions visited in the previous steps. The elements stored in the tabu list are the attributes of moves, rather than the attributes of solutions. The main aim of using this attributive is to save computer memory. With our neighborhood structure, the move selected at each step may reverse more than one disjunctive arc and involve a sequence of operations. In this paper, unlike the previous TS algorithm, we memorize in the tabu list not only the sequence of the operations, but also their positions on the machine. This approach could better represent the attributes of moves. More precisely, if the move consists of the exchange on u and v, then a move achieved the same sequence of operations and positions (namely, the operations from u to v $(u, \ldots, w, \ldots, v)$ and their positions on the machine from u to v $(p_u, \ldots, p_w, \ldots, p_v)$) is not permitted for the duration that the move is labeled as "tabu". Moreover, it must be pointed out that the definition of tabu status is based on attributes of moves. A side effect of implementing the "a partial attribute" tabu list is that it may lead to giving a tabu status to unvisited solution or even an interesting solution. One way of avoiding this problem is to use an aspiration criterion which accepts the move provided that its makespan is lower than that of the current best solution found so far.

The length of tabu list determines the time limit of remaining on memory for the elements, which is discouraged at the current iterations. Therefore, it has an influence on the course of the search. If the length of list is too short, cycling cannot be avoided; conversely, a too long size creates too many restrictions. According to our experimental study, the suitably "small" length of tabu list is convenient for intensifying search in the visited region and might find a better solution, whereas the long tabu list size is adapted for the diversification of the search, which directs the search to explore the larger region. However, how to set the length of tabu list efficiently in JSP still remains an open problem, and setting the parameters often suffers from tedious trial and error. An empirical study suggests it may be possible to find good quality solutions when the length of tabu list is allowed to vary dynamically during the course of the search. For example, Taillard [6] suggests the length of tabu list be randomly chosen from a range between given minimal and maximal values and changed each time a number of iterations. Another example is the somewhat more sophisticated approach of Dell'Amico and Trubian [13].

Our preliminary experiments show the length of tabu list depends strongly on the number of jobs and machines, and the suitable length of tabu list increases as the ratio of the number of jobs (n) to the number of machines (m) becomes larger. The smallest length of tabu list could be set to $L = 10 + n/m$ for getting good results. In order to avoid cycling more effectively, dynamic tabu list is applied and the length of tabu list is randomly chosen between two extreme values $L_{\min}$ and $L_{\max}$. In a great number of experiments, we found that the range between two extreme values

is not too large and increases as the ratio of the number of jobs to the number of machines becomes larger. $L_{\min} = [L]$ and $L_{\max} = [1.4–1.5L]$ are appropriate values for the considered problems. If $n \leqslant 2m$, then the maximal extreme value $L_{\max} = [1.4L]$. Otherwise, $L_{\max} = [1.5L]$. However, in our experiments the method that the tabu list size changes every number of iterations does not always show the distinct superiority. Hence, in our approach, the length of tabu list is randomly selected between the given minimal and maximal values simply.

3.5. Move selection

The choice rule of the TS method is to select the move which is non-tabu with the lowest makespan or satisfies the aspiration criterion. However, a situation can arise where all possible moves are tabu and none of them are able to satisfy the aspiration criterion. In such a case, our procedure randomly selects a move among the possible moves.

3.6. Termination criterion

The algorithm stops when the number of iterations or the number of disimproving moves reaches to a maximum value, or the solution is proved to be optimal. If one of the following conditions is satisfied: (1) all critical operations are processed on the same machine (i.e. only one critical block is generated) or belong to the same job (i.e. each block consists of only one operation), (2) the makespan is equal to the known lower bound, then the solution is optimal and the algorithm terminates.

4. Computational experiments

The enhanced dynamic TS (TS_{ED}) algorithm described above was implemented in VC + + language on personal computer Pentium IV1.8G and tested on a large number of standard instances taken from the literature. These job shop scheduling instances include the following classes[4]: three instances denoted as (FT6, FT10, FT20) due to Fisher and Thompson [24], forty instances (LA01–LA40) due to Lawrence [25], five instances (ABZ5–ABZ9) due to Adams et al. [3], twenty instances (SWV01–SW 20) due to Storer et al. [26], four instances (YN1–YN4) due to Yamada and Nakano [27] and eighty instances (DMU01–DMU 80) due to Demirkol et al. [28]. For these benchmark set, the best known upper bounds (UB_{best}) and the best known lower bounds (LB_{best}) are taken from Jain and Meeran [10] and updated with the improved results from Nowicki and Smutnicki [16]. To evaluate the quality of the solutions, the mean relative error (MRE) is calculated from the best known lower bound (LB_{best}), and the upper bound (UB_{solve}) that is the makespan of the best solution solved by our algorithm, using the "relative deviation" formula $RE = 100 \times (UB_{solve} - LB_{best})/LB_{best}$ for each instance of problem.

The TS_{ED} algorithm was initiated from the randomly active solution generated by using the randomized Giffler and Thomson algorithm, the choice of the length of tabu list is indicated in Section 3.4 and the number of iterations is set to 0–50 millions. The number of the disimproving moves is set according to the special conditions. For example, in terms of the comparison of the neighborhood strategies, the number of the disimproving moves is set to 3000; In terms of the comparison of estimation approach and exact approach, it is set to 1 million. In the remainder of this section, the first two sections give the comparison of the neighborhood strategies and comparison of the move evaluation strategies respectively in order to analyze the new neighborhood structure and the estimation strategy used in this paper, and in the last section our algorithm is performed on a large number of the benchmark instances to measure the performance of the algorithm.

4.1. Comparison of the neighborhood strategies

Six neighborhood structures are tested and compared below. They are denoted by NS1 (van Laarhoven et al. 1992, N1), NS2 (Chambers and Barnes, 1994), NS3 (Nowicki and Smutnicki 1996, N5), NS4 (modify the Dell'Amico and Trubian, 1993, N4), NS5 (Balas and Vazacopoulos 1998, N6). Finally, NS6 denotes our new neighborhood structure.

[4] The FT, LA, ABZ, SWV and YN problems are available from the OR Library site http://www.ms.ic.ac.uk/job/pub/jobshop1.txt while the DUM problems are available from http://gilbreth.ecn.purdue.edu/~uzsoy2/benchmark/problems.html

Table 2
The comparison of the six neighborhood structures

	NS6	NS5	NS4	NS3	NS2	NS1
MC_{max}	1416	1420	1415	1462	1473	1600
MRE	3.31	3.51	3.45	5.52	5.84	12.6
NBE	32	33	32	26	24	20
MEN	106 163	70 216	127 140	48 973	48 510	107 649
MNI	5731	5468	5619	6690	6206	4149
CPU-time	298	205	462	121	134	307

Meanwhile, NS1, NS2, NS3 and NS5 are well-known neighborhoods in literature. These neighborhoods are all based on the concept of block except for N1. For further information, see Section 3.2 and Blazewicz et al. [9]. NS4 slightly differs from N4 introduced by Dell'Amico and Trubian. NS4 neighborhood moves the operations in a block to the beginning or the end of this block according to the method of Dell'Amico and Trubian, and the moves do not allow for an interchange on u and v when the critical path containing u and v contains neither $JS[u]$ nor $JP[v]$.

In order to evaluate these neighborhood structures exactly, we use TS_{ED} algorithm as the platform, and the difference only exist in the neighborhood structure. The initial solution is generated by SPT priority dispatch rule, and the length of tabu list is set to 12, suggested by Geyik and Cedimoglu [29] except for NS1 which applies the approach of Taillard. Each move in the neighborhood is evaluated exactly. The algorithm is terminated when the number of disimproving moves reaches to the value 3000. The benchmark set FT, LA, ABZ, SWV and YN containing 72 instances are tested. Moreover, several measures that gain some statistics relating to the comparison are presented. They are the mean makespan $C_{max}(MC_{max})$, the number of the solution found equal to the known best solution (NBE), the mean number of evaluated neighbors (MEN), the mean number of iterations (MNI) and the total CPU time performed in all instances (CPU-time).

Table 2 summarizes the computational results relating to the six neighborhoods. NS6 offers the minimum MRE value among the six neighborhood structures. The MRE provided by NS4 (better than the original N4) is close to that of NS6, but NS4 consumes too much computing times. NS5 is more effective than NS3, NS2 and NS1, and NS3 is better than NS2 and NS1. Overall, it can be seen that NS6 is an effective neighborhood structure for the JSP.

4.2. Comparison of estimation approach and exact approach

The estimation approach proposed by Balas and Vazacopoulos and the exact evaluation approach suggested by Ten Eikelder et al. are compared using our TS_{ED} algorithm as the test platform (see Table 3). The two test platforms are all similar except the move evaluation methods. To make a more detailed performance comparison, we selected the 15 most difficult instances among FT, LA and ABZ benchmark set. The majority of the 15 instances have been viewed as computational challenges, and even the optimal solutions of the instances ABZ8 and ABZ9 still remain unknown. In the experiments, the algorithm was initiated to form the active solution generated randomly and terminated when an optimal solution was found or the number of disimproving moves reaches to 1 million. In Table 3, the column named LB lists the best known lower bounds indicated in Jain and Meeran, the next columns named *Best*, *Avg*, *Worst*, T_{av} (s) show the best makespan, the average makespan, the worst makespan and the average computing time in seconds obtained by the algorithm over 10 runs respectively, the last column named T_{estima}/T_{exact} shows the ratio of the T_{av} of the estimation approach to the T_{av} of the exact approach. The last line indicates the best MRE (b-MRE), the average MRE (avg-MRE) and the average value of T_{estima}/T_{exact}, which could be used for analyzing the performance of the two evaluation approaches.

Table 3 shows that the estimation approach of Balas and Vazacopoulos is 2–4 times faster in evaluating moves in comparison to the exact evaluation of Ten Eikelder et al. [23]. In the experiments, the estimation approach performs better than the exact approach, not only by getting to the solutions faster, but also they are able to achieve a slightly lower b-MRE and avg-MRE. Moreover, Table 3 also indicates that the best makespans produced by the estimation approach is better than or equal to those of the exact approach for 14 instances (better than for 4 instances), with the exception of LA40 (on this particular occasion, the best makespan given by the exact approach reaches the optimal solution 1222). A plausible explanation for this is that the estimation strategy enlarges the selected range of the next

Table 3
Comparison of the estimation approach (Balas) with the exact approach (Ten Eikelder)

Problem	Size	LB	Estimation method (Balas)				Exact method (Ten Eikelder)				T_{estima}/T_{exact}
			Best	Avg	Worst	T_{av} (s)	Best	Avg	Worst	T_{av} (s)	
FT10	10 × 10	930	930*	930.4	934	41.1	930*	930.9	935	124.6	0.33
LA19	10 × 10	842	842*	842	842	5.6	842*	842	842	7.7	0.72
LA21	15 × 10	1046	1046*	1046.6	1047	113.3	1046*	1047.4	1049	393.3	0.29
LA24	15 × 10	935	935*	937.1	941	87.5	935*	939.6	943	317.9	0.27
LA25	20 × 10	977	977*	977.1	978	54.7	977*	977.1	978	133.3	0.41
LA27	20 × 10	1235	1235*	1235	1235	16.9	1235*	1235	1235	77	0.22
LA29	20 × 10	1152	1156	1160.9	1164	189.7	1159	1163	1166	612.4	0.31
LA36	15 × 15	1268	1268*	1268.1	1269	59.7	1268*	1268.4	1269	266.1	0.22
LA37	15 × 15	1397	1397*	1406.6	1418	172.8	1397*	1405.6	1410	658.8	0.26
LA38	15 × 15	1196	1196*	1200.9	1202	181.3	1196*	1200.8	1202	480.9	0.38
LA39	15 × 15	1233	1233*	1234.3	1239	145.5	1233*	1238.3	1240	647.8	0.22
LA40	15 × 15	1222	1224	1224.5	1226	167.8	1222*	1224.1	1225	552	0.30
ABZ7	20 × 15	656	657	661.2	667	243.4	658	663.1	669	940.8	0.26
ABZ8	20 × 15	665(645)	669	670.2	672	270.9	670	671.5	673	1185.3	0.23
ABZ9	20 × 15	679(661)	680	684.7	688	263.6	681	684.9	688	1127.6	0.23
MRE			0.48	0.72			0.52	0.81		T_{avg}	0.31

*The best solutions found by our algorithm are equal to the best known lower bound.

iteration, which can make the search explore larger solution space. Therefore, the estimation approach proposed by Balas and Vazacopoulos is implemented to perform these computations of each move in the following section.

4.3. Computational study

Firstly, the fifteen most difficult instances selected above are still considered. In order to make a more detailed performance analysis, we compared our TS_{ED} algorithm with the recently best approximation algorithms and used the following notation for these algorithms: TSSB stands for A tabu search method guided by shifting bottleneck of Pezzella and Merelli [30], BV stands for the guided local search with shifting bottleneck of Balas and Vazacopoulos [22]. Meanwhile, SB-RGLS5 stands for the solution of SB-RGLS5 procedure of Balas and Vazacopoulos, BV-best stands for the best solution obtained by Balas and Vazacopoulos and TSAB stands for the Taboo Search of Nowicki and Smutnicki [7]. Taking account on the big difference of computers speed used in tests, we did not list the computing time in detail. Table 4 shows the makespan performance statistics of these algorithms for the fifteen tough problems.

Due to the stochastic nature of the TS_{ED} algorithm, we compare the mean performance (avg-MRE) of TS_{ED} algorithm with results of the other algorithms. Overall, the solution quality provided by TS_{ED} algorithm is better than those of TSSB and TSAB, and close to that of BV provided that the running time is not taken into account. Furthermore, for the rectangle problems, such as LA27 and LA29, TS_{ED} algorithm outperforms the other algorithms in terms of both solution quality and computing time. For example, TS_{ED} algorithm is able to obtain the optimal solution LA27 of 1235 within several seconds on a personal PC, whereas SB-RGLS1 found the optimal solution in 788 s on CI-CPU time. It should be noted that TS_{ED} algorithm found the solutions for LA29-1156 and ABZ7-657 better than the other algorithms do. These results on these benchmark instances suggest that our TS_{ED} algorithm might be more effective in dealing with the "rectangular" problem.

In order to further analyze TS_{ED} algorithm, some larger instances of size 50 × 10 generated by Storer et al. [26] (SWV11, SWV12, SWV13, SWV14, SWV15) are tested. Table 5 gives the detailed results for the SWV11-15 instances. The results have been selected over three runs of the algorithm. In this table, it lists problem name, problem dimension, the best upper and lower bounds, the number of iterations, the best makespans, the computing time achieved the best makespan and the solution obtained by the other algorithms. Meanwhile, *i*-TSAB stands for the tabu search algorithm of Nowicki and Smutnicki [16], CSSAW stands for the simulated annealing procedure of Steinhöfel et al. [31]. Note that the TS_{ED} algorithm is significantly superior to the other three algorithms in the solution quality. We found 3

Table 4
Computational results of the fifteen tough FT, LA and ABZ problems

Problem	Size	LB	Best	Avg	TSSB	SB-RGLS5	BV-best	TSAB
FT10	10 × 10	930	930*	930.4	930	930	930	930
LA19	10 × 10	842	842*	842	842	842	842	842
LA21	15 × 10	1046	1046*	1046.6	1046	1046	1046	1047
LA24	15 × 10	935	935*	937.1	938	935	935	939
LA25	20 × 10	977	977*	977.1	979	977	977	977
LA27	20 × 10	1235	1235*	1235	1235	1235	1235	1236
LA29	20 × 10	1152	1156	1160.9	1168	1164	1157	1160
LA36	15 × 15	1268	1268*	1268.1	1268	1268	1268	1268
LA37	15 × 15	1397	1397*	1406.6	1411	1397	1397	1407
LA38	15 × 15	1196	1196*	1200.9	1201	1196	1196	1196
LA39	15 × 15	1233	1233*	1234.3	1240	1233	1233	1233
LA40	15 × 15	1222	1224	1224.5	1233	1224	1224	1229
ABZ7	20 × 15	656	657	661.2	666	664	662	670
ABZ8	20 × 15	665(645)	669	670.2	678	671	669	682
ABZ9	20 × 15	679(661)	680	684.7	693	679	679	695
MRE			0.48	0.72	1.08	0.61	0.53	1.0

*The best solutions found by our algorithm equal to the best known lower bound.

Table 5
Results for the problem instances of Storer et al. (1992)

Problem	Size	UB(LB)	The number of iterations (in millions)						Our best	Time (s)	*i*-TS AB	BV-best	CS SAW
			0.5	1	3	5	10	50					
SWV11	50 × 10	2983[a]	3016	3001	2984	2983			2983*	823	2983*	3008	3017
SWV12	50 × 10	3003(2972)	3045	3018	2999	2995	2984	2979	**2979**[b]	8631	–	3041	3012
SWV13	50 × 10	3104	3127	3104					3104*	297	–	3163	3122
SWV14	50 × 10	2968	2970	2968					2968*	168	–	2968*	–
SWV15	50 × 10	2904(2885)	2930	2921	2901	2893	2890	2886	**2886**[b]	5672	–	2942	2924
MRE			1.19	0.68	0.30	0.21	0.12	0.05	0.05		0.51	1.41	1.10

[a]The optimal solution has been found by Nowicki and Smutnicki [16].
[b]The makespans our algorithm found are better than the best previously known value.
*The algorithm found the optimal solution.

optimal solutions, and improved the upper bounds for the 2 remaining instances which are remarkably better than the best previously known value and very close to the lower bounds. Moreover, TS_{ED} algorithm on instances SWV11-15 provides MRE = 0.05%, whereas *i*-TSAB algorithm, which is one of the best algorithms recently, only achieves MRE = 0.51% in 50 million iterations. TSED algorithm achieves MRE = 1.19% before 0.5 million of iterations about 1 min on a personal PC, whereas to achieve the similar aim, BV needs approximately 120–180 min on the SUNSPARC-330. CS_{SAW} attains MRE = 1.1% without time bounds ($t \to \infty$). Therefore, we come to the conclusion that TS_{ED} algorithm is more effective and efficient in terms of both the solution quality and computing time when the number of jobs is greater than the number of machines.

Finally, we tested TS_{ED} algorithm on DUM benchmarks. DMU class contains 80 instances ($300 \leqslant$ operations $\leqslant 1000$), and optimal solutions have been known for twenty-two out of them (mainly among the instances DUM01-40). Instances DMU41-80 are considered particularly hard because they satisfy the 2_{SETS} principle, and no optimal solution for these instances has been known. In this paper, we improved the 51 upper bounds among the unsolved 58 DUM instances, see Table 6. Meanwhile, for DUM13 and DUM26 the found results provide the optimal solutions, namely: DUM13-3681 and DUM26-4647. These results have been found over three runs of the algorithm. Table 6 shows all instances which our algorithm found the new upper bounds.

Table 6
New upper bounds found by our algorithm

Problem	Size	Old UB	New UB	Problem	Size	Old UB[a]	New UB	Problem	Size	Old UB	New UB
SWV10	20 × 15	1767	1763	DUM44	20 × 15	3528	3524	DUM63	40 × 15	5446	5371
SWV12	50 × 10	3003	2979	DUM45	20 × 15	3321	3296	DUM64	40 × 15	5443	5330
SWV15	50 × 10	2904	2886	DUM46	20 × 20	4101	4080	DUM65	40 × 15	5271	5201
DUM01	20 × 15	2571	2566	DUM48	20 × 20	3838	3795	DUM66	40 × 20	5911	5797
DUM02	20 × 15	2715	2711	DUM49	20 × 20	3780	3735	DUM67	40 × 20	6016	5872
DUM06	20 × 20	3265	3254	DUM50	20 × 20	3794	3761	DUM68	40 × 20	5936	5834
DUM08	20 × 20	3199	3191	DUM51	30 × 15	4260	4218	DUM69	40 × 20	5891	5794
DUM11	30 × 15	3470	3455	DUM52	30 × 15	4383	4362	DUM70	40 × 20	6096	5954
DUM12	30 × 15	3519	3516	DUM53	30 × 15	4470	4428	DUM71	50 × 15	6359	6278
DUM13	30 × 15	3698	**3681**[*]	DUM54	30 × 15	4425	4405	DUM72	50 × 15	6586	6520
DUM16	30 × 20	3787	3759	DUM55	30 × 15	4332	4308	DUM73	50 × 15	6330	6249
DUM17	30 × 20	3854	3842	DUM56	30 × 20	5079	5025	DUM74	50 × 15	6383	6316
DUM18	30 × 20	3854	3846	DUM57	30 × 20	4785	4698	DUM75	50 × 15	6437	6236
DUM19	30 × 20	3823	3784	DUM58	30 × 20	4834	4796	DUM76	50 × 20	7082	6893
DUM20	30 × 20	3740	3716	DUM59	30 × 20	4696	4667	DUM77	50 × 20	6930	6868
DUM26	40 × 20	4679	**4647**[*]	DUM60	30 × 20	4904	4805	DUM78	50 × 20	7027	6846
DUM42	20 × 15	3448	3416	DUM61	40 × 15	5294	5228	DUM79	50 × 20	7253	7055
DUM43	20 × 15	3473	3459	DUM62	40 × 15	5354	5311	DUM80	50 × 20	6998	6719

[a]Old UB are taken from Nowicki and Smutnicki [16].
[*]The best solutions found by our algorithm are equal to the best known lower bound.

Most of the previous upper bounds for the instances DUM1-80 found by the famous *i*-TSAB algorithm of Nowicki and Smutnicki [16]. In this paper, our TS_{ED} algorithm improved most of the upper bounds found by *i*-TSAB algorithm for the DUM instances, in addition TS_{ED} algorithm found better makespans for all unsolved DUM1-80 instances when $n \geqslant 2m$ (see Table 6). Therefore, it is convinced that for the rectangle instances TS_{ED} algorithm outperforms *i*-TSAB algorithm in the solution quality. Despite the different computers speed used by tests, for the rectangle DUM instances TS_{ED} algorithm generally obtains the upper bounds found by *i*-TSAB algorithm before 5–10 million number of iterations (For $300 \leqslant$ operations $\leqslant 600$ before 5 million iterations, and for $600 <$ operations $\leqslant 1000$ before 10 million iterations), whereas algorithm *i*-TSAB achieves the upper bounds in 50 million of iterations. Overall, our TS_{ED} algorithm outperforms *i*-TSAB algorithm in both solution quality and performance for the rectangle instances.

5. Conclusion

The effectiveness and efficiency of the tabu search for the JSP depend on the neighborhood structures and move evaluation strategies. However, in order for search to be more efficient, the neighborhood structures defined in the JSP are becoming more and more constrained at present, such as the highly restrictive neighborhood of the Nowicki and Smutnicki. Consequently, this TS algorithm is prone to trap at a local minimum, and the recency-based memory with the recovery of elite solutions and cycle checking mechanism are used for overcoming the problem. In this paper, a new neighborhood structure is constructed, which could avoid cycling and investigate much larger solution space. We compare the new neighborhood structure with the other five neighborhood strategies, and confirm that it is an effective neighborhood structure for the JSP. Moreover, the effect of neighborhood evaluation strategies is investigated. Empirical testing discloses that the estimation approach introduced by Balas and Vazacopoulos not only significantly improves the efficiency of the search, but also has no material effect on the solution quality. Finally, we tested our approach on a set of standard benchmark instances and improved a large number of upper bounds, which further validates the effectiveness of the neighborhood structure proposed.

The algorithm presented in this paper for the JSP is simple and easily implementable. For the rectangular problem this algorithm even outperforms the intricate *i*-TSAB algorithm in both solution quality and performance. For example, our algorithm improved all upper bounds found by the *i*-TSAB algorithm for the unsolved DUM1-80 instances when $n \geqslant 2m$. The *i*-TSAB algorithm certainly is the current state-of-the-art algorithm for the JSP. However, in this paper

we suspect that i-TSAB algorithm might be a very powerful tool to solve the JSP when $n \leqslant 2m$, but for the rectangle problem, due to the limit of the highly restrictive neighborhood structure, the algorithm i-TSAB does not always perform better. Furthermore, it is our conviction that, for the rectangle problem, tabu search with the powerful neighborhood structures and dynamic tabu list could be more effective than the muti-start TS approach. This is very useful for the real-world applications because the number of jobs is much greater than the number of machines among most of the practical scheduling problems.

In this paper, we focus mainly on the performance of the neighborhood structure and do not mention on the diversification strategy of tabu search. Therefore, it may be a subject of future work. One promising approach for diversification may be the application of multiple neighborhoods or the long-term memory mechanism. In addition, how to efficiently set the tabu list in JSP is still an open problem. A growing research suggests that the length of short term memory varies dynamically in response to the changing conditions of the search, but there is still a broad research for better implementations. Therefore, another study in the future may develop the more efficient tabu list strategies.

Acknowledgements

This paper is supported by the National Basic Research Program of China (under the Grant no. 2005CB724107) and the National Natural Science Foundation, China (under the Grants no. 50305008, 70271053). The authors would like to thank the referees for their helpful comments and suggestions.

References

[1] Carlier J, Pinson E. An algorithm for solving the job shop problem. Management Science 1989;35(29):164–76.
[2] Brucker P, Jurisch B, Sievers B. A branch and bound algorithm for job-shop scheduling problem. Discrete Applied Mathematics 1994;49:105–27.
[3] Adams J, Balas E, Zawack D. The shifting bottleneck procedure for job shop scheduling. Management Science 1988;34:391–401.
[4] Croce FD, Tadei R, Volta G. A genetic algorithm for the job shop problem. Computers & Operations Research 1995;22:15–24.
[5] Van Laarhoven PJM, Aarts EHL, Lenstra JK. Job shop scheduling by simulated annealing. Operations Research 1992;40(1):113–25.
[6] Taillard ÉD. Parallel taboo search techniques for the job-shop scheduling problem. ORSA Journal on Computing 1994;6:108–17.
[7] Nowicki E, Smutnicki C. A fast taboo search algorithm for the job shop scheduling problem. Management Science 1996;42(6):797–813.
[8] Vaessens RJM, Aarts EHL, Lenstra JK. Job shop scheduling by local search. INFORMS Journal on Computing 1996;8:302–17.
[9] Blażewicz J, Domschke W, Pesch E. The job shop scheduling problem: conventional and new solution techniques. European Journal of Operational Research 1996;93:1–33.
[10] Jain AS, Meeran S. Deterministic job shop scheduling: past, present and future. European Journal of Operational Research 1999;113:390–434.
[11] Glover F. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 1986;13:533–49.
[12] Taillard ÉD. Parallel taboo search technique for the job shop scheduling problem. Technical Report ORWP 89/11, DMA, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 1989.
[13] Dell'Amico M, Trubian M. Applying tabu-search to job-shop scheduling problem. Annals of Operations Research 1993;41(1–4):231–52.
[14] Barnes JW, Chambers JB. Solving the job shop scheduling problem using tabu search. IIE Transactions 1995;27:257–63.
[15] Chambers JB, Barnes JW. New tabu search results for the job shop scheduling problem. Technical Report ORP96-10, Graduate Program in Operations Research and Industrial Engineering, The University of Texas at Austin, 1996.
[16] Nowicki E, Smutnicki C. Some new tools to solve the job shop problem. Technical Report 60/2002, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland, 2002.
[17] Balas E. Machine sequencing via disjunctive graph: an implicit numeration algorithm. Operations Research 1969;17:941–57.
[18] Glover F. Tabu search—Part I. ORSA Journal on Computing 1989;1(3):190–206.
[19] Glover F. Tabu search—Part II. ORSA Journal on Computing 1990;2(1):4–32.
[20] Glover F, Laguna M. Tabu search. Dordrecht: Kluwer Academic Publishers; 1997.
[21] Matsuo HD, Suh CJ, Sullivan RS. A controlled search simulated annealing method for general job shop scheduling problem. Working Paper, 03-04-88, Department of Management, The University of Texas at Austin, Austin, TX, 1988.
[22] Balas E, Vazacopoulos A. Guided local search with shifting bottleneck for job shop scheduling. Management Science 1998;44(2):262–75.
[23] Ten Eikelder HMM, Aarts BJM, Verhoeven MGA, Aarts EHL. Sequential and paralled local search algorithms for job shop scheduling. In: Voss S, Martello S, Osman IH, Roucairol C, editors. Meta-Heuristic: Advances and Trends in Local Search Paradigms for Optimization. Massachusetts: Kluwer Academic Publishers; 1999. p. 359–71.
[24] Fisher H, Thompson GL. Probabilistic learning combinations of local job-shop scheduling rules. In: Muth JF, Thompson GL, editors. Industrial scheduling. Prentice-Hall: Englewood Cliffs, NJ; 1963. p. 225–51.
[25] Lawrence S. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement). Graduate School of Industrial Administration, Carnegie–Mellon University, Pittsburgh, Pennsylvania, 1984.
[26] Storer RH, Wu SD, Vaccari R. New search spaces for sequencing problems with applications to job-shop scheduling. Management Science 1992;38(10):1495–509.

[27] Yamada T, Nakano R. A genetic algorithm applicable to large-scale job-shop problems. In: Manner R, Manderick B, editors. Proceedings of the second international workshop on parallel problem solving from nature (PPSN'2). Belgium: Brussels; 1992. p. 281–90.
[28] Demirkol E, Mehta S, Uzsoy R. Benchmarks for shop scheduling problems. European Journal of Operational Research 1998;109:137–41.
[29] Geyik F, Cedimoglu IH. The strategies and parameters of tabu search for job-shop scheduling. Journal of Intelligent Manufacturing 2004;15:439–48.
[30] Pezzella F, Merelli E. A tabu search method guided by shifting bottleneck for job shop scheduling problem. European Journal of Operational Research 2000;120:297–310.
[31] Steinhöfel K, Albrecht A, Wong CK. An experimental analysis of local minima to improve neighbourhood search. Computers & Operations Research 2003;30:2157–73.