European Journal of Operational Research 120 (2000) 297–310

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/orms

A tabu search method guided by shifting bottleneck for the job shop scheduling problem

Ferdinando Pezzella, Emanuela Merelli *

Istituto di Informatica, Università degli Studi di Ancona, via Brecce Bianche, 60131 Ancona, Italy

Received 1 October 1997; accepted 1 September 1998

Abstract

A computationally effective heuristic method for solving the minimum makespan problem of job shop scheduling is presented. The proposed local search method is based on a tabu search technique and on the shifting bottleneck procedure used to generate the initial solution and to refine the next-current solutions. Computational experiments on a standard set of problem instances show that, in several cases, our approach, in a reasonable amount of computer time, yields better results than the other heuristic procedures discussed in the literature. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Production scheduling; Jobshop; Heuristics; Tabu search

1. Introduction

The job shop problem studied in the present paper consists in scheduling a set of jobs on a set of machines with the objective to minimize the *makespan*, i.e., the maximum of completion times needed for processing all jobs, subject to the constraints that each job has a specified processing order through the machines and that each machine can process at most one job at a time.

This problem, described as $J//C_{\max}$ using the three field notation of Graham et al. [17], is *NP-hard* [11]. As a matter of fact, only small size instances of the problem can be solved with a reduced computational time by exact algorithms as shown by Carlier and Pison [7] and Lenstra [20]. Contrarily for large instances, important results have been recently achieved by heuristic algorithms, some of them are based on a local search method. A detailed overview was given by French [10], Morton and Pentico [22], Vaessen et al. [26] and Aarts and Lenstra [1]. Starting from an initial feasible solution, a local search method iteratively searches the best solution among those in the neighbourhood, i.e., in the set of feasible solutions "near" to the current solution. Several authors, Matsuo et al. [21], Van Laarhoven et al. [27], Dell'Amico and Trubian [8] and Nowicki and Smutnicki [23], observe that the choice of a

* Corresponding author. Present address: Dipartimento di Matematica e Fisica, Università degli Studi di Camerino, Camerino, Italy. E-mail: (pezzella,merelli)@inform.unian.it

PII: S 0 3 7 7 - 2 2 1 7 (9 9) 0 0 1 5 8 - 7

good-initial solution is an important aspect of algorithms' performance in terms of solution quality and computational time. Nevertheless, most of the initial solutions in the above algorithms are found by heuristics based on simple priority rules.

In the present paper, a new heuristic algorithm based on a tabu search (TS) method [12–14] and on the shifting bottleneck procedure (SBP) [2] is proposed. It aims to improve the quality of the initial solution and of the next-best ones. As a matter of fact, the SBP is used to find a good-initial feasible solution, and a local reoptimization, based on the same procedure, is used to improve each current solution determined by a TS method.

The job shop scheduling problem is formalized in terms of a mathematical model and is represented via disjunctive graph; subsequently, the TS technique and the SBP are analysed and the new algorithm is described. Finally, computational results on several test problems are described and the new heuristic procedure is compared with some best-performing ones for job shop scheduling.

2. Problem definition and notation

Each instance of the problem $J//C_{\max}$ is defined by a set of jobs, a set of machines and a set of operations. Each job consists of a sequence of operations, each of which has to be performed on a given machine for a given time. A *schedule* is an allocation of the operations to time intervals on the machines. The problem is to find the schedule that minimizes the makespan subject to the following constraints: (i) the precedences of operations given by each job are to be respected, (ii) each machine can perform at most one operation at a time and (iii) the operations cannot be interrupted.

Let:
- $J = \{1, \dots, n\}$ denote the set of jobs;
- $M = \{1, \dots, m\}$ denote the set of machines;
- $V = \{0, 1, \dots, \tilde{n}+1\}$ denote the set of operations, where 0 and $\tilde{n}+1$ represent the dummy *start* and *finish* operations, respectively;
- A be the set of pair of operations constrained by the precedence relations, as in (i);
- V_k be the set of operations to be performed by the machine k;
- $E_k \subset V_k \times V_k$ be the set of pair of operations to be performed on the machine k and which therefore have to be sequenced, as specified in (ii);
- p_v and t_v denote the (fixed) processing time and the (variable) start time of the operation v, respectively. The processing time of the 0 and $\tilde{n}+1$ operations is equal to zero, i.e., $p_0 = p_{\tilde{n}+1} = 0$.

Given the above assumptions, the problem can be stated as [5]

$$
\begin{array}{ll}
\text{minimize} \quad t_{\tilde{n}+1} & \\
\text{subject to} & \\
t_j - t_i \geqslant p_i, & (i,j) \in A, \\
t_j - t_i \geqslant p_i \vee t_i - t_j \geqslant p_j, & (i,j) \in E_k, \quad k \in M, \\
t_i \geqslant 0, & i \in V.
\end{array}
\tag{1}
$$

The first set of constraints represents the precedence relations among the operations of the same job, whereas the second set of constraints describes the sequencing of the operations on the same machine. These constraints impose that either $t_j - t_i \geqslant p_i$ or $t_i - t_j \geqslant p_j$. Any feasible solution of the problem (1) is called schedule.

In this framework, it is useful to represent the job shop scheduling problem in terms of a disjunctive graph $G := (V, A, E)$ [4,24], where V is the set of nodes, A the set of ordinary arcs (conjunctive) and E the set of disjunctive arcs. The nodes of G correspond to operations, the directed arcs to precedence relation, and the disjunctive arcs to operations to be performed on the same machine. More precisely, $E = \cup_{k=1}^{m} E_k$, where E_k is the subset of disjunctive arcs related to a machine k; each disjunctive arc of E can be considered as a pair of opposite directed arcs. The length of an arc $(i,j) \in A$ is p_i, the length of an disjunctive arc $(i,j) \in E$ is either p_i or p_j depending on its orientation. The selection of a processing order on each machine involves the orientation of the disjunctive arcs, in order to produce an acyclic directed graph. A schedule on a disjunctive graph G consists in finding a set of orientations that minimizes the length of the longest path (*critical path*) in the resulting acyclic directed graph.

According to the Adams et al. [2] method, the graph G can be decomposed into one direct subgraph $D = (V, A)$, by deleting disjunctive arcs, and in m cliques $G_k = (V_k, E_k)$, obtained from G by deleting both the conjunctive arcs and the dummy nodes 0 and $\tilde{n}+1$. A selection S_k in E_k contains exactly one arc between each pair of opposite arcs in E_k. A selection is acyclic since it does not contain any directed cycle. Moreover, sequencing the operations on the machine k is equivalent to choosing an acyclic selection in E_k. A complete selection S is the union of selections S_k, one for each E_k, $k \in M$. S generates the directed graph $D_S = (V, A \cup S)$; S is acyclic if the associated directed graph D_S is acyclic. An acyclic complete selection S infers a schedule, i.e., a feasible solution of Problem (1).

In order to solve the job shop scheduling problem the best acyclic complete selection S^* that minimizes the value of the length of the longest critical path in the direct graph $D_{S^*} = (V, A \cup S^*)$ must be determined.

The neighbourhood of the current solution can be formed by the solutions generated by inverting the direction of a disjunctive arc in the critical path of D_S. To this end, as stated by other authors [16], it is useful to decompose the critical path into a sequence of r blocks $(B_1, B_2, \ldots, B_r)$. Each block contains the operations processed on the same machine; for each pair of consecutive blocks B_j, B_{j+1} with $1 \leqslant j \leqslant r$ the last operation of B_j and the first of B_{j+1} belong to the same job but are performed on different machines. In order to illustrate the introduced notions, let us consider the following example:

Job	Processing cycle
J_1	$(1,10),\ (2,5),\ (3,6)$
J_2	$(2,5),\ (1,8)$
J_3	$(1,2),\ (3,10),\ (2,4)$

The job shop scheduling problem has three jobs, three machines and eight operations. Job first consists of a sequence of three operations, job second consists of a sequence of two operations and job third consists of a sequence of three operations. The processing cycle for each job is a sequence of items (m_v, p_v) where m_v denotes the machine kth and p_v the processing time on machine kth for the operation v, respectively.

The disjunctive graph of the above problem is shown in Fig. 1.

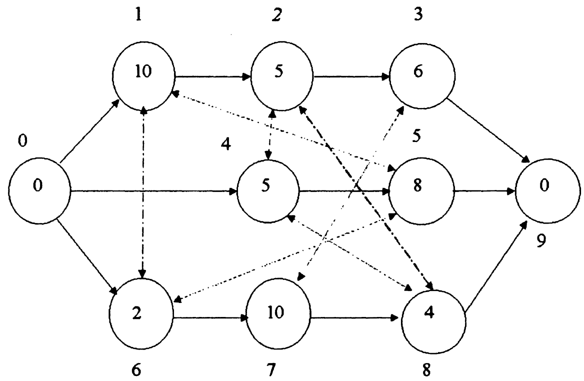

Fig. 1. Disjunctive graph for a job shop scheduling problem with $n=3$, $m=3$, $\tilde{n}=8$.

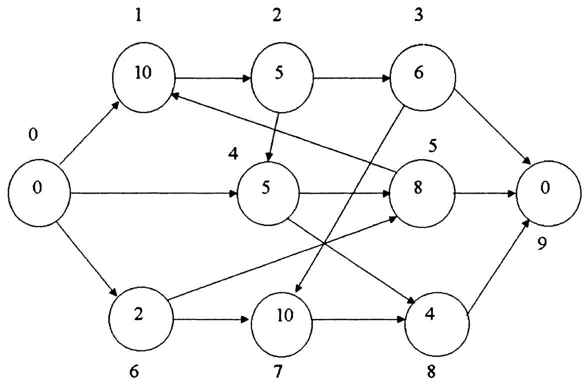

Fig. 2. A feasible solution via a direct graph.

The number outside a node represents the number of operation $v \in V$, whereas the number inside a node is the processing time p_v. A feasible solution of the problem is represented by the directed graph shown in Fig. 2. The associated critical path is $\{(0,4,5,1,2,3,7,8,9)\}$ with the length equal to 48 and $r=5$ blocks are $B_1=\{4\}$, $B_2=\{5,1\}$, $B_3=\{2\}$, $B_4=\{3,7\}$, $B_5=\{8\}$, as illustrated by Fig. 3.

3. The strategies

The implementation of the proposed algorithm is based on two well-known techniques: the TS method and the SBP.

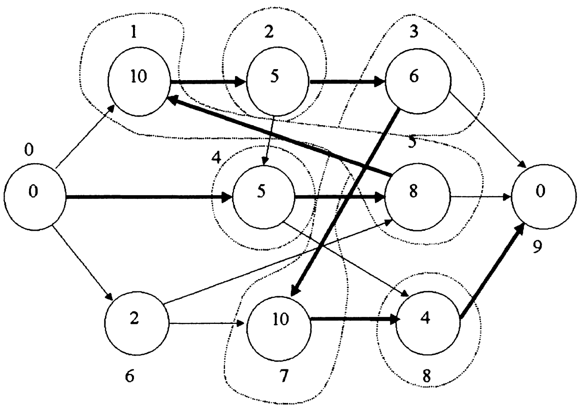

Fig. 3. Critical path and blocks.

3.1. The tabu search

The Tabu Search (TS) is a metaheuristic approach mainly used to find a near-optimal solution of combinatorial optimization problems. It was proposed and formalized by Glover [12–14].

The TS technique is based on an iterative procedure "neighbourhood search method" for finding, in a finite set T of feasible solutions, a solution $t^* \in T$ that minimizes a real-valued objective function f. Neighbourhood search methods are iterative procedures in which a neighbourhood $N(t)$ is defined for each solution $t \in T$ and the next solution is searched among the solutions in $N(t)$, obtained by a predefined partial modification of t, usually called *move*. Starting from an initial feasible solution, the neighbourhood $N(t^c)$ of the current solution t^c is examined and the solution t' with usually the best objective function is chosen as the next solution, i.e., $t' | f(t') \leqslant f(t''),\ t', t'' \in N(t^c)$. The fact that movement from a t^c to a $t' \in N(t^c)$ is allowed even if $f(t') > f(t^c)$ helps escape from local optima.

However, with the above scheme cycling is possible. To prevent cycling, a structure called *tabu list* L of length l (fixed or variable) is introduced to prevent returning to a solution visited in the last l iterations. Recently, the theory and practice of TS was extensively improved by Glover and Laguna [15] and by Hertz et al. [18] by aspiration criteria and intensification and diversification schemes. The TS process stops when the solution is close enough to the lower bound of the objective function value f, if known. Otherwise, it stops when no improvement occurs over the best solution for a given number of iterations or the time limit runs out.

3.1.1. Neighbourhood and moves

In order to improve the efficiency of the exploration process TS keeps track not only of local information (the current value of the objective function) but also of some information related to the exploration process. This systematic use of the stored information is the essential feature of TS. The search process uses this information to avoid cycling and to explore new directions in the neighbourhood. The stored information allows to exclude some solutions from those in the neighbourhood $N(t^c)$ reducing the set of choices, then the structure of $N(t^c)$ will depend upon the itinerary and hence upon the iteration K; so we may refer to the neighbourhood as $N(t^c, K)$ instead of $N(t^c)$. The definition of $N(t^c, K)$ implies that some recently visited solutions are removed from $N(t^c)$. They are considered *tabu solutions*, which should be avoided in the next iteration. The stored information, represented by a data structure named tabu list, are based on the last events and will partly prevent cycling. The exploration process in the set of feasible solutions T is described in terms of *moves* from one solution to the next. For each solution $t \in T$, $M(t)$ is defined as the set of moves $\mathtt{m}$ that can be applied to t in order to obtain a new solution s; let $s = t \oplus \mathtt{m}$ be a notation then $N(t) = \{s | \exists \mathtt{m} \in M(t) \text{ with } s = t \oplus \mathtt{m}\}$. In general the moves are reversable: for each $\mathtt{m}$ there exists a move $\mathtt{m}^{-1}$ such that $(t \oplus \mathtt{m}) \oplus \mathtt{m}^{-1} = t$. So instead of storing the complete solution in the tabu list and testing if a candidate solution belongs to the list, that is impractible, the tabu list stores only the move or the reverse of the move associated with the move actually performed. The restrictions due to the tabu list sometimes are too strong, and prevent a good search of the algorithm. This situation can be overcome by using a so-called *aspiration criterion*, which allows to choose among the forbidden solutions: this corresponds to surmounting a specific tabu status. A tabu move $\mathtt{m}$ applied to a current solution t may be promising since it gives a better solution than the best one so far found. The current move m can be feasible in spite of its status if at least an aspiration criterion is satisfied.

Writing t^* for the best solution found so far and K for the iteration counter, the main steps of TS algorithm are:

1. Choose an initial solution $t \in T$; Set $t^* = t$ and $K = 0$.
2. $K = K + 1$ and generates the subset $\hat{N}$ of solutions in $N(t, K)$ so that either the applied move does not belong to the tabu list or at least one of the aspiration criteria holds.
3. Choose a best solution $t \in \hat{N}$ according to the objective function $f(\cdot)$ of the problem.
4. If $f(t) < f(t^*)$ then set $t^* = t$.
5. Update the tabu list and the aspiration criteria.
6. If stopping criterion is met, then stop. Otherwise go to Step 2.

Some of the stopping criteria are as follows: (i) $N(t, K + 1) = \emptyset$, (ii) K is larger than the maximum number of iteration allowed, (iii) the number of iterations since the last improvement of the best solution is larger than a specific number and (iv) the optimal value is obtained.

3.2. The shifting bottleneck

The Shifting Bottleneck Procedure (SBP) proposed by Adams et al. [2] is an effective heuristic method for the jobshop scheduling problem. It is based on an empirical idea: the performance of the system for industrial application depends on the correct use of scarce resources. In the framework of the job shop scheduling problem, the scarce resource is represented by a bottleneck machine, i.e., the machine that mostly affects the value of the makespan. The algorithm produces the sequences of operations on the set of the machines so as to determine every time the bottleneck machine (primary, secondary and so on) among those not yet sequenced. Therefore, for each machine not yet sequenced, the algorithm solves exactly a one-machine sequencing problem. Whenever a new machine is sequenced, all the already sequenced ones are locally reoptimized by solving again for each of them a one-machine sequencing problem, keeping the sequence constant on the other machines. The main steps of an iteration of the SBP are:

1. *identification* of kth bottleneck machine (solving $m - k + 1$ one machine problems);
2. *local reoptimization* of the sequence of each critical machine (solving a one-machine problem and cycling until some improvement is found).

The procedure iterates over each machine and finishes when no improvement is found. At the last step, after the last machine has been sequenced, the procedure continues to local reoptimization until there is no improvement for the full cycle.

4. Tabu search with shifting bottleneck

In this section, the new procedure based on a TS method and the Shifting Bottleneck procedure (TSSB) is presented. The integration of the SBP in the TS framework aims to use qualitative information extensively during the local search. The proposed heuristic method based on the TS technique uses the SBP to determine the initial solution, and subsequently to reoptimize locally the sequence of each machine belonging to the critical path.

The TSSB algorithm is composed of three fundamental modules; the first (*SB_init*) implements the SBP and then generates the initial solution, the second (*TS_proc*) implements the local search based on the TS technique and the third (*SB_reopt*) reoptimizes locally the sequence of each machine whenever a better solution is provided by *TS_proc*. The integration of the three modules is shown in Fig. 4.

4.1. Initial solution

As previously observed, in most of the algorithms based on TS methods [8,23] the availability of a good initial solution is fundamental for the computational performance of the algorithm itself. The choice generation of the initial feasible solution by the SBP allows to obtain better solutions in comparable computational times or the same solution in shorter computational times. The *SB_init*

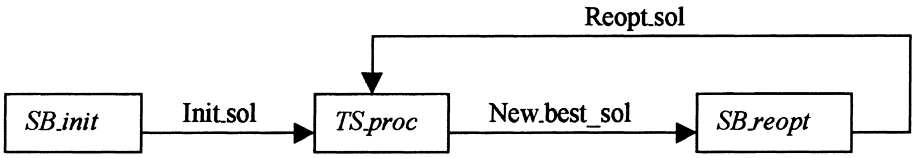

Fig. 4. Main modules for the TSSB algorithm.

also provides a good lower bound to quantify the error in the solution obtained by the *TS_proc* procedure in the next step of the algorithm.

4.2. Neighbourhood structure

The implementation of TSSB is based on three different kinds of functions denoted *d* by $N1(\cdot)$, $N2(\cdot)$, $N3(\cdot)$. The above functions are applied to pairs of operations i, j which are assigned to the same machines so that the arc (i, j) is on the critical path. Since the proposed algorithm operates on a directed acyclic graph $D_S = (V, A \cup S)$ where the critical path is represented in terms of blocks, the functions $N1(\cdot)$, $N2(\cdot)$ and $N3(\cdot)$ are applied to the operations of each single block.

$N1(\cdot)$ is a modification of that proposed by Van Laarhoven et al. [27]. Each move exchanges two subsequent operations of the block with the exception of the first and the last ones.

$N2(\cdot)$ is a modification of the neighbourhood NB proposed by Dell'Amico and Trubian [8]. Each move is a sequence of elementary exchanges, by which an operation that is neither the first nor the last one is put in the second position and in the last position but one of the block.

Finally, $N3(\cdot)$ proposed by Nowicki and Smutnicki [23], exchanges the first two (and the last two) operations in every block.

Each of these functions when applied to a feasible solution $t \in T$ leads to a set of feasible solutions $t' \in T$.

Each exchange that deletes the link between blocks is called an *external exchange*, whereas those that do not delete the links are called *internal exchanges*; an external exchange leads to a different sequence of blocks. The functions $N1(\cdot)$ and $N2(\cdot)$ perform only internal exchanges whereas $N3(\cdot)$ performs external exchanges.

The functions $N2(\cdot)$ and $N3(\cdot)$ are sequentially applied during the diversification phase of the local search, whereas the function $N1(\cdot)$ is used during the intensification phase of the search. Moreover, the function $N1(\cdot)$ is applied after finding a better solution, for a fixed number of iterations.

4.3. Tabu list and implementation strategies

During the iteration, the *TS_proc* procedure selects the best not forbidden solution among those in the neighbourhood set; the choice is limited by the information stored in the tabu list. Therefore, the length of the list plays an important role in the search process. Nevertheless, if the list is too long the search can be inhibited, whereas if it is too short cycling cannot be avoided; the experimental evidence has shown that the average number of explored solutions increases as the length of the tabu list increases.

The proposed *TS_proc* procedure uses a dynamic list. If there is no improvement in the current solution, the length l of the list changes as the iterations *iter* increases. The behaviour of the length is showed in Fig. 5 and described by the next five steps:

1. for the first K iterations, the list has a constant length equal to n,
2. from K to $2K$ iterations, the length of the list linearly decreases until $l_{\min} = \frac{2}{3}n$,
3. from $2K$ to $3K$ iterations, the length of the list remains constant,
4. from $3K$ to $4K$ iterations, the length of the list linearly increases from $l_{\min}$ until $l_{\max} = 2n$, and finally
5. from $4K$ to $5K$ iterations, the length of the list remains constant.

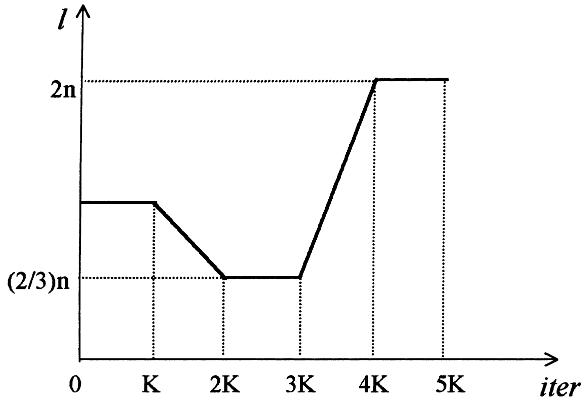

Fig. 5. Dynamic list.

where n is the number of jobs, K the constant fixed to $\frac{1}{5}$*Maxiter*, *Maxiter* the parameter representing the maximum number of iterations with no improvement of the best solution allowed during the local search.

The tabu list contains the information associated to the selected moves. If $N1(\cdot)$ and $N3(\cdot)$ are applied each reverse exchange, i.e., *reverse move* is memorized; if $N2(\cdot)$ is applied the *sequence of the elementary exchanges* is memorized, forbidding their reversion.

For the neighbourhood functions $N1(\cdot)$ and $N3(\cdot)$ a candidate move is tabu if its reverse belongs to the tabu list. For the neighbourhood function $N2(\cdot)$ a candidate move is tabu if the last exchange of its sequence belongs to the tabu list.

The adopted aspiration criteria (i.e., the conditions for which the tabu list can be violated) are the following:

- *by default*: the move with oldest attributes in the tabu list can be selected if all moves are classified to be tabu and no other aspiration criterion holds;
- *by objective*: a move classified as tabu can be selected if the relative solution has a better makespan than the one of the current best solution.

Furthermore, the proposed algorithm stops if the best obtained solution equals the lower bound or if the number of iterations is equal to the maximum number of allowed iterations represented by the Maxiter parameter.

4.4. Local reoptimization

The SBP guides the TS not only by providing the initial solution but also by improving each better solution via a *local reoptimization* as suggested by Adams et al. [2].

In the framework of TSSB algorithm, the *SB_reopt* is the procedure used to local reoptimization the machines in the critical path. Whenever the *TS_proc* procedure selects a new best solution, the *SB_reopt* procedure must try to improve the sequences of operations associated with each machine in the critical path. The reoptimization of each machine is obtained by solving a one-machine sequencing problem while keeping the sequences constant the other machines.

In detail, let t' be the new best feasible solution at a given step of the TSSB algorithm, S_k be a related selection associated with each machine $k \in M$ and $D_{S(t')}$ be the corresponding direct graph with a complete selection $S(t') = \cup_k^m S_k$. Let $\bar{M}$ denote the set of machines belonging to the critical path of the direct graph $D_{S(t')}$, where an arbitrary ordering on $\bar{M}$, i.e., $(\bar{M}(1), \bar{M}(2), \ldots, \bar{M}(p))$ with $p = |\bar{M}|$ and $p \leqslant m$ is imposed, and denote a one-machine sequencing problem by $P(M(k), \bar{M} - \{M(k)\})$. The $P(M(k), \bar{M} - \{M(k)\})$ problem can be stated as

$$\begin{array}{lll}
\text{minimize} & t_{\bar{n}+1} & \\
\text{subject to} & & \\
t_j - t_i \geqslant p_i, & (i,j) \in A \cup (S_p\colon & \\
 & \quad p \in \bar{M}\{k\}), & (2)\\
t_j - t_i \geqslant p_i \vee t_i - t_j \geqslant p_j, & (i,j) \in E_k, & \\
t_i \geqslant 0, & i \in V. &
\end{array}$$

Then, for $k = 1, \ldots, p$ the one-machine sequencing problem $P(M(k), \bar{M} - \{M(k)\})$ is solved. The solution of $P(M(k), \bar{M} - M(k))$ in terms of selection $S_{M(k)}$ replaces the old selection and the next problem is solved. Every time a full cycle is completed, the elements of $\bar{M}$ are reordered according to decreasing values of the solutions of Problem (2). The local reoptimization continues until there is not any improvement for a full cycle.

The implementation of *SB_reop* uses the Carlier's algorithm [6] to solve the one-machine sequencing problem. Fig. 6 shows a flow chart of the TSSB algorithm.

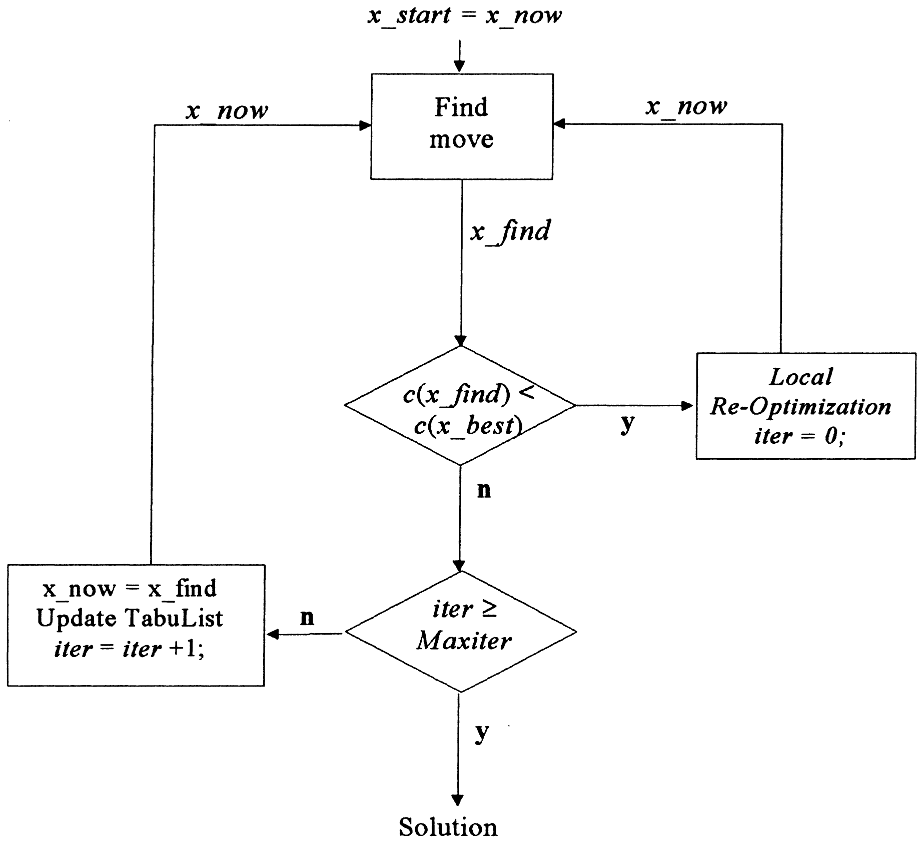

Fig. 6. Flow chart of the TSSB algorithm.

Table 1
Results by TSSB for the problem instances of class (a)

Problem	n	m	Opt (LB UB)	TSSB	RE_{TSSB}	T_{av}
ORB1	10	10	1059	1064	0.47	82
ORB2	10	10	888	890	0.23	75
ORB3	10	10	1005	1013	0.80	87
ORB4	10	10	1005	1013	0.80	75
ORB5	10	10	887	887	0.00	81
FT6	6	6	55	55	0.00	–
FT10	10	10	930	930	0.00	80
FT20	20	5	1165	1165	0.00	115
ABZ5	10	10	1234	1234	0.00	75
ABZ6	10	10	943	943	0.00	80
ABZ7	20	15	656	666	1.52	200
ABZ8	20	15	(645 669)	678	5.12	205
ABZ9	20	15	(661 679)	693	4.84	195
MRE					1.06	

5. Computational results

The TSSB algorithm implemented in C language on personal computer Pentium 133 MHz.

In order to reduce the running time the Maxiter parameter, i.e., the maximum number of iterations, has been set to $100n$, where n is the number of jobs.

The algorithm has been tested on 133 problem instances of various sizes and hardness level provided by OR-Library (`http://mscmga.ms.ic.ac.uk/info.html`) collected in the following classes:

(a) Five instances denoted as (ORB1-ORB5) due to Applegate and Cook [3], three instances (FT6, FT10, FT20) due to Fisher and Thomp-

Table 2
Computational results by TSSB for the problem instances of class (b)

Problem	n	m	Opt (LB UB)	TSSB	RE_{TSSB}
LA01	10	5	666	666	0.00
LA02	10	5	655	655	0.00
LA03	10	5	597	597	0.00
LA04	10	5	590	590	0.00
LA05	10	5	593	593	0.00
LA06	15	5	926	926	0.00
LA07	15	5	890	890	0.00
LA08	15	5	863	863	0.00
LA09	15	5	951	951	0.00
LA10	15	5	958	958	0.00
LA11	20	5	1222	1222	0.00
LA12	20	5	1039	1039	0.00
LA13	20	5	1150	1150	0.00
LA14	20	5	1292	1292	0.00
LA15	20	5	1207	1207	0.00
LA16	10	10	945	945	0.00
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
LA21	15	10	1046	1046	0.00
LA22	15	10	927	927	0.00
LA23	15	10	1032	1032	0.00
LA24	15	10	935	938	0.32
LA25	15	10	977	979	0.20
LA26	15	10	1218	1218	0.00
LA27	20	10	1235	1235	0.00
LA28	20	10	1216	1216	0.00
LA29	20	10	(1142 1153)	1168	2.28
LA30	20	10	1355	1355	0.00
LA31	30	10	1784	1784	0.00
LA32	30	10	1850	1850	0.00
LA33	30	10	1719	1719	0.00
LA34	30	10	1721	1721	0.00
LA35	30	10	1888	1888	0.00
LA36	15	15	1268	1268	0.00
LA37	15	15	1397	1411	1.00
LA38	15	15	1196	1201	0.42
LA39	15	15	1233	1240	0.57
LA40	15	15	1222	1233	0.90
MRE					0.14

Table 3
Comparison with Balas and Vazacopoulos [5]

Problem	n	m	TSSB		SB-GLS1		SB-GLS2	
			MRE	T_{av}	MRE	T_{av}	MRE	T_{av}
LA01-05	10	5	0.00	9.8	0.44	1.2	0.00	5.9
LA06-10	15	5	0.00	–	0.00	–	0.00	–
LA11-15	20	5	0.00	–	0.00	–	0.00	–
LA16-20	10	10	0.00	61.5	0.50	9.7	0.00	47
LA21-25	15	10	0.10	115	0.60	21.3	0.15	139.6
LA26-30	15	10	0.46	105	0.83	20.6	0.47	121.6
LA31-35	30	10	0.00	–	0.00	–	0.00	–
LA36-40	15	15	0.58	141	0.78	0.4	0.23	278
MRE				0.14		0.63		0.17

Table 4
Comparison with other algorithms

Problem	n	m	Opt (LB UB)	DT		TSAB		SB-GLS1		SB-RGLS5		TSSB	
				UB	RE	UB	RE	UB	RE	UB	RE	UB	RE
FT10	10	10	930	935	0.538	930	0.000	930	0.000	930	0.000	930	0.000
LA2	10	5	655	655	0.000	655	0.000	666	1.679	655	0.000	655	0.000
LA19	10	10	842	842	0.000	842	0.000	852	1.188	842	0.000	842	0.000
LA21	15	10	1046	1048	0.191	1047	0.096	1048	0.191	1046	0.000	1046	0.000
LA24	15	10	935	941	0.642	939	0.428	941	0.642	935	0.000	938	0.321
LA25	15	10	977	979	0.205	977	0.000	993	1.638	977	0.000	979	0.205
LA27	20	10	1235	1242	0.567	1236	0.081	1243	0.648	1235	0.000	1235	0.000
LA29	20	10	(1142 1153)	1182	3.503	1160	0.607	1182	3.503	1164	1.926	1168	2.277
LA36	15	15	1268	1278	0.789	1268	0.000	1268	0.000	1268	0.000	1268	0.000
LA37	15	15	1397	1409	0.859	1407	0.716	1397	0.000	1397	0.000	1411	1.002
LA38	15	15	1196	1203	0.585	1196	0.000	1208	1.003	1196	0.000	1201	0.418
LA39	15	15	1233	1242	0.730	1233	0.000	1249	1.298	1233	0.000	1240	0.568
LA40	15	15	1222	1233	0.900	1229	0.573	1242	1.637	1224	0.164	1233	0.900
MRE					0.73		0.19		1.03		0.16		0.43

son [9], and five instances (ABZ5-ABZ9) due to Adams et al. [2]. The optimal solutions of the ABZ8 and ABZ9 instances are still unknown.

(b) Forty instances of eight different sizes ($n \times m = 10 \times 5, 15 \times 5, 20 \times 5, 10 \times 10, 15 \times 10, 20 \times 10, 30 \times 10, 15 \times 15$) denoted as (LA01-LA40) due to Lawrence [19]. The optimal solution of the LA29 instance is still unknown.

(c) Eighty instances of eight different size ($n \times m = 15 \times 15, 20 \times 15, 20 \times 20, 30 \times 15, 30 \times 20, 50 \times 15, 50 \times 20, 100 \times 20$) denoted by (TA1-TA80). This class contains "partially hard" cases selected by Taillard [25] among a large number of randomly generated instances. The optimal solution is known only for 33 out of 80 instances.

The effectiveness of the proposed algorithm was analysed in terms of best solution found by TSSB algorithm (UB) compared to the best value provided by the OR-Library.

The relative error RE(%) was calculated for each instance of problem, as a percentage by which the solution obtained is above the optimum value (Opt) if it is known or the best known lower bound value (LB): $100 * (\mathrm{UB} - \mathrm{Opt})/\mathrm{Opt}$, or $100 * (\mathrm{UB} - \mathrm{LB})/\mathrm{LB}$.

The computational results are given in Tables 1–6; Tables 1, 2 and 5 list the results obtained by applying the proposed algorithm to the classes:

Table 5
Results by TSSB and SB-GLS1 for problem instances of class (c)

Problem	n	m	Opt (LB UB)	TSSB	RE_{TSSB}	SB-GLS1	$RE_{SB\text{-}GLS1}$	BV-*best*	$RE_{BV\text{-}best}$
TA1	15	15	1231	1241	0.812	1244	1.056	1231	0.000
TA2	15	15	1244	1244	0.000	1255	0.884	1244	0.000
TA3	15	15	(1206 1218)	1222	1.327	1225	1.575	1218	0.995
TA4	15	15	(1170 1175)	1175	0.427	1191	1.795	1181	0.940
TA5	15	15	(1210 1228)	1229	1.570	1256	3.802	1233	1.901
TA6	15	15	(1210 1239)	1245	2.893	1247	3.058	1243	2.727
TA7	15	15	(1223 1228)	1228	0.409	1244	1.717	1228	0.409
TA8	15	15	(1187 1217)	1220	2.780	1222	2.949	1217	2.527
TA9	15	15	(1247 1274)	1291	3.528	1291	3.528	1274	2.165
TA10	15	15	1241	1250	0.725	1266	2.015	1241	0.000
TA11	20	15	(1321 1364)	1371	3.785	1402	6.132	1392	5.375
TA12	20	15	(1321 1367)	1379	4.391	1416	7.192	1367	3.482
TA13	20	15	(1271 1350)	1362	7.160	1377	8.340	1350	6.216
TA14	20	15	1345	1345	0.000	1361	1.190	1345	0.000
TA15	20	15	(1293 1342)	1360	5.182	1383	6.961	1353	4.640
TA16	20	15	(1300 1368)	1370	5.385	1418	9.077	1369	5.308
TA17	20	15	(1458 1464)	1481	1.578	1519	4.184	1478	1.372
TA18	20	15	(1369 1396)	1426	4.164	1433	4.675	1396	1.972
TA19	20	15	(1276 1341)	1351	5.878	1376	7.837	1341	5.094
TA20	20	15	(1316 1353)	1366	3.799	1398	6.231	1359	3.267
TA21	20	20	(1539 1647)	1659	7.797	1692	9.942	1659	7.797
TA22	20	20	(1511 1601)	1623	7.412	1638	8.405	1603	6.089
TA23	20	20	(1472 1558)	1573	6.861	1594	8.288	1558	5.842
TA24	20	20	(1602 1651)	1659	3.558	1714	6.991	1659	3.558
TA25	20	20	(1504 1598)	1606	6.782	1631	8.444	1615	7.380
TA26	20	20	(1539 1655)	1666	8.252	1698	10.331	1659	7.797
TA27	20	20	(1616 1689)	1697	5.012	1722	6.559	1689	4.517
TA28	20	20	(1591 1615)	1622	1.948	1653	3.897	1615	1.508
TA29	20	20	(1514 1625)	1635	7.992	1639	8.256	1629	7.596
TA30	20	20	(1473 1596)	1614	9.572	1621	10.048	1604	8.893
TA31	30	15	(1764 1766)	1771	0.397	1809	2.551	1766	0.113
TA32	30	15	(1774 1803)	1840	3.720	1840	3.720	1803	1.635
TA33	30	15	(1778 1796)	1833	3.093	1844	3.712	1796	1.012
TA34	30	15	(1828 1832)	1846	0.985	1898	3.829	1832	0.219
TA35	30	15	2007	2007	0.000	2010	0.149	2007	0.000
TA36	30	15	(1819 1823)	1825	0.330	1874	3.024	1823	0.220
TA37	30	15	(1771 1784)	1813	2.372	1846	4.235	1784	0.734
TA38	30	15	(1673 1681)	1697	1.435	1762	5.320	1681	0.478
TA39	30	15	(1795 1798)	1815	1.114	1822	1.504	1798	0.167
TA40	30	15	(1631 1686)	1725	5.763	1749	7.235	1686	3.372
TA41	30	20	(1859 2023)	2045	10.005	2106	13.287	2026	8.983
TA42	30	20	(1867 1961)	1979	5.999	2018	8.088	1967	5.356
TA43	30	20	(1809 1879)	1898	4.920	1946	7.573	1881	3.980
TA44	30	20	(1927 2003)	2036	5.656	2069	7.369	2004	3.996
TA45	30	20	(1997 2005)	2021	1.202	2049	2.604	2008	0.551
TA46	30	20	(1940 2033)	2047	5.515	2115	9.021	2040	5.155
TA47	30	20	(1789 1920)	1938	8.329	1973	10.285	1921	7.378
TA48	30	20	(1912 1973)	1996	4.393	2080	8.787	1982	3.661
TA49	30	20	(1915 1991)	2013	5.117	2046	6.841	1994	4.125
TA50	30	20	(1807 1951)	1975	9.297	2009	11.179	1967	8.854
TA51	50	15	2760	2760	0.000	2760	0.000	2760	0.000
TA52	50	15	2756	2756	0.000	2756	0.000	2756	0.000
TA53	50	15	2717	2717	0.000	2717	0.000	2717	0.000

Table 5 (Continued)

Problem	n	m	Opt (LB UB)	TSSB	RE_{TSSB}	SB-GLS1	$RE_{SB\text{-}GLS1}$	BV-*best*	$RE_{BV\text{-}best}$
TA54	50	15	2839	2839	0.000	2839	0.000	2839	0.000
TA55	50	15	2679	2684	0.187	2683	0.149	2679	0.000
TA56	50	15	2781	2781	0.000	2781	0.000	2781	0.000
TA57	50	15	2943	2943	0.000	2943	0.000	2943	0.000
TA58	50	15	2885	2885	0.000	2885	0.000	2885	0.000
TA59	50	15	2655	2655	0.000	2657	0.075	2655	0.000
TA60	50	15	2723	2723	0.000	2723	0.000	2723	0.000
TA61	50	20	2868	2868	0.000	2891	0.802	2868	0.000
TA62	50	20	(2869 2895)	2942	2.544	2962	3.242	2900	1.081
TA63	50	20	2755	2755	0.000	2796	1.488	2755	0.000
TA64	50	20	2702	2702	0.000	2726	0.888	2702	0.000
TA65	50	20	2725	2725	0.000	2751	0.954	2725	0.000
TA66	50	20	2845	2845	0.000	2845	0.000	2845	0.000
TA67	50	20	(2825 2826)	2865	1.416	2841	0.566	2826	0.035
TA68	50	20	2784	2784	0.000	2785	0.036	2784	0.000
TA69	50	20	3071	3071	0.000	3071	0.000	3071	0.000
TA70	50	20	2995	2995	0.000	3004	0.301	2995	0.000
TA71	100	20	5464	5464	0.000	5464	0.000	5464	0.000
TA72	100	20	5181	5181	0.000	5181	0.000	5181	0.000
TA73	100	20	5568	5568	0.000	5568	0.000	5568	0.000
TA74	100	20	5339	5339	0.000	5339	0.000	5339	0.000
TA75	100	20	5392	5392	0.000	5392	0.000	5392	0.000
TA76	100	20	5342	5342	0.000	5342	0.000	5342	0.000
TA77	100	20	5436	5436	0.000	5436	0.000	5436	0.000
TA78	100	20	5394	5394	0.000	5394	0.000	5394	0.000
TA79	100	20	5358	5358	0.000	5358	0.000	5358	0.000
TA80	100	20	5183	5183	0.000	5183	0.000	5183	0.000
MRE of (TA1-TA80)					2.56		3.68		2.13

(a), (b) and (c) of problem instances. Whereas Table 3 shows the comparison of the results by TSSB algorithm with those obtained by Balas and Vazacopoulos SB-GLS1 and SB-GLS2 procedures [5]. SB-GLS1 and SB-GLS2 has been chosen because the literature provides complete results for the Lawrence's problems. This results have been computed on SunSparc-330). Table 4 shows the comparison of TSSB with well-known algorithms from the literature. The column labelled DT refers to the Dell'Amico and Trubian algorithm [8], next column TSAB is Novicki and Smutnicki algorithm

Table 6
Comparison with Balas and Vazacopoulos [5]

Problem	n	m	TSSB		SB-GLS1		BV-*best*	
			MRE	T_{av}	MRE	T_{av}	MRE	T_{av}
TA01-10	15	15	1.45	2175	2.24	57	1.16	1498
TA11-20	20	15	4.13	2526	6.18	113	3.67	4559
TA21-30	20	20	6.52	34910	8.12	165	6.10	6850
TA31-40	30	15	1.92	14133	3.53	175	0.79	8491
TA41-50	30	20	6.04	11512	8.50	421	5.20	16018
TA51-60	50	15	0.02	421	0.02	152	0.00	196
TA61-70	50	20	0.39	6342	0.83	590	0.11	2689
TA71-80	100	20	0.00	231	0.00	851	0.00	851

[23] and finally the next two columns are the algorithms SB-GLS1 and SB-RGLS5 by Balas and Vazacopoulos [5]. For each algorithm the best value of the makespan and relative error are reported on selected problems set (FT10, LA2, LA19, LA21, LA24-29, LA36-40). All the comparative results were taken from the literature [26].

Table 5 shows the results for TSSB and SB-GLS1 [5] on a set of 80 problems proposed by Taillard (TA01-TA80). The mean relative error is equal to 2.56% for the TSSB algorithm, while it is equal to 3.67% for SB-GLS1 and to 2.13% if the best solution among all those provided by the algorithms of Balas and Vazacopoulos is chosen (column labeled by BV-*best*). Table 6 shows the comparison between TSSB and Balas and Vazacopoulos algorithms in terms of makespan and computational time on Taillard's problems set, collected for different sizes. The TSSB algorithm, for the problems which optimal solution is known, finds solutions with relative error smaller then one percent with the only exception on instance ABZ7.

6. Conclusions

In this paper, a new heuristics method based on a combination of a TS technique and the SBP has been proposed. The initial solution given by shifting bottleneck, the special structure of neighbourhood, and the proposed dynamic list allow to obtain interesting results.

The algorithm has been tested on several benchmark problem instances. The optimal or near optimal solutions were found for many problem instances in a reasonable amount of computing time.

Acknowledgements

Thanks are due to Filippo Gabrielli and Renato De Leone for useful suggestions that helped improve the paper.

This research activity was performed in the framework of the project on "AAA: Autocoordinamento di Agenti Autonomi" (Self-Coordination of Autonomous Agents) of Italian Ministry of University and Scientific Research, whose financial support is acknowledged.

Appendix A

The noations used in the tables are as follows:

Opt(LB UB)	the optimal value if known, otherwise in brackets the best lower and upper bound, all these values have been taken from OR-Library;
n	number of jobs
m	number of machines
SB-GLS1	best solution of SB-GLS1 procedure of Balas and Vazacopoulos [5]
SB-GLS2	best solution of SB-GLS2 procedure of Balas and Vazacopoulos [5]
SB-RGLS5	best solution of SB-RGLS-5 procedure of Balas and Vazacopoulos [26]
BV-*best*	best solution among of those provided in Balas and Vazacopoulos [5]
TSSB	best solution of TSSB
$RE_{SB\text{-}GLS1}$	percent relative error by SB-GLS1
$RE_{SB\text{-}GLS2}$	percent relative error by SB-GLS2
$RE_{SB\text{-}RGLS5}$	percent relative error by SB-RGLS5
$RE_{BV\text{-}best}$	percent relative error by BV-*best*
RE_{TSSB}	percent relative error by TSSB
T_{av}	average computing time in seconds
DT	Dall'Amico and Trubian results [8]
TSAB	best solution of Nowicki and Smutnicki [26]
RE	relative error in percent
MRE	mean relative error in percent for a set of problems

References

[1] E. Aarts, J.K. Lenstra, Local Search and Combinatorial Optimization, Wiley, New York, 1997.

[2] J. Adams, E. Balas, D. Zawack, The shifting bottleneck procedure for job shop scheduling, Management Science 34 (1988) 391–401.

[3] D. Applegate, W. Cook, A computational study of the job shop scheduling problem, ORSA Journal on Computing 3 (1991) 149–156.
[4] E. Balas, Machine sequencing via disjunctive graph: An implicit enumeration algorithm, Operations Research 17 (1969) 941–957.
[5] E. Balas, A. Vazacopoulos, Guided local search with shifting bottleneck for job shop scheduling, Tech. Rep., Management Science Research Report MSRR-609, GSIA Carnegie Mellon University, Pittsburgh, 1994.
[6] J. Carlier, The one-machine sequencing problem, European Journal of Operational Research 11 (1982) 42–47.
[7] J. Carlier, E. Pison, An algorithm for solving the job-shop problem, Management Science 35 (1989) 164–176.
[8] M. Dell'Amico, M. Trubian, Applying tabu-search to the job-shop scheduling problem, Annals of Operations Research 4 (1993) 231–252.
[9] H. Fisher, G. Thompson, Probabilistic learning combinations of local job-shop scheduling rules, in: J. Muth, G. Thompson (Eds.), Industrial Scheduling, Prentice-Hall, Englewood Cliffs, NJ, 1963.
[10] S. French, Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop, Wiley, New York, 1982.
[11] M. Garey, D. Johnson, R. Sethy, The complexity of flow shop and job shop scheduling, Mathematics of Operations Research 1 (1976) 117–129.
[12] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 13 (1986) 533–549.
[13] F. Glover, Tabu search: Part I, ORSA Journal on Computing 1 (1989) 190–206.
[14] F. Glover, Tabu search: Part II, ORSA Journal on Computing 2 (1990) 4–32.
[15] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, 1997.
[16] J. Grabowski, E. Nowicki, S. Zdrzalka, A block approach for single machine scheduling with release dates and due dates, European Journal of Operational Research 26 (1986) 278–285.
[17] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, Annals of Discrete Mathematics 5 (1979) 287–326.
[18] A. Hertz, E. Taillard, D. De Werra, Local search in combinatorial optimization, in: E. Aarts, J. Lenstra (Eds.), Tabu Search, vol. 5, Wiley, New York, (1997) pp. 121–136.
[19] S. Lawrence, Supplement to, Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques, Tech. Rep., GSIA, Carnegie Mellon University, 1984.
[20] J. Lenstra, Sequencing by enumerative methods, Tech. Rep. Mathematical Centre Tract 69, Mathematisch Centrum, Amsterdam, 1976.
[21] H. Matsuo, C. Suh, R. Sullivan, A controlled search simulated annealing method for the general job-shop scheduling problem, Tech. Rep. 03-04-88, Dept. of Management, The University of Texas, Austin, 1988.
[22] T. Morton, D. Pentico, Heuristic Scheduling Systems, Wiley, New York, 1993.
[23] E. Nowicki, C. Smutnicki, A fast taboo search algorithm for the job shop problem, Management Science 42 (1996) 797–813.
[24] B. Roy, B. Sussman, Le problèmes d'ordonnancement avec contraintes disjonctives, Note DS No. 9 bis, SEMA, Paris, 1964.
[25] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (1993) 278–285.
[26] R. Vaessens, E. Aarts, J. Lenstra, Job shop scheduling by local search, INFORMS Journal on Computing 8 (1996) 302, 317, 2nd revision.
[27] P. Van Laarhoven, E. Aarts, J. Lenstra, Job shop scheduling by simulated annealing, Operations Research 40 (1992) 113–125.