Yugoslav Journal of Operations Research
6 (1996), Number 1, 5-17

TABU SEARCH: A BRIEF SURVEY AND SOME REAL-LIFE APPLICATIONS

Mirjana M. ČANGALOVIĆ *), Vera V. KOVAČEVIĆ-VUJČIĆ *)
Laboratory for Operations Research, Faculty of Organizational Sciences, University of Belgrade, Yugoslavia

Lav IVANOVIĆ, Milan DRAŽIĆ
Faculty of Mathematics, University of Belgrade, Yugoslavia

Miroslav D. AŠIĆ
Department of Mathematics, The Ohio State University, USA

Abstract: This paper gives a brief survey of Tabu search methodology - one of the widely used modern general heuristics, originally designed to solve combinatorial optimization problems. The power of this methodology is illustrated by applications to two different types of real-life large dimensional problems: a combinatorial assignment problem and a continuous optimal design problem.

Keywords: Tabu search, heuristic methods, combinatorial optimization, global optimization

1. INTRODUCTION

Tabu search (TS) methodology belongs to a class of so-called modern heuristics which are currently intensively used to solve a large variety of combinatorial optimization problems. The basic concepts of this methodology were proposed by Glover [6], while similar views were independently developed by Hansen et al. [15] (under the name Steepest ascent mildest descent heuristic). TS represents a general heuristic strategy for solving any combinatorial optimization problem of the form

$$\min_{x \in X} f(x),$$

where X is a finite set of feasible solutions, called the solution space, and f is an arbitrary function such that $f: X \to R$. This strategy has already been successfully ap-

*) This research was supported by the Science Fund of Serbia, grant number 0401F, through the Mathematical Institute.

plied to a wide range of various problems and is rapidly spreading to many new fields of application. Such success in implementation could be explained by the fact that the general concepts of TS are flexible enough to incorporate some elements of Artificial Intelligence and, thus, they can be easily adapted to different kinds of problem structures. Therefore, TS is one of the most popular and the most promising tools (together with Simulated annealing and Genetic algorithms) for solving hard combinatorial problems, especially practical ones.

The purpose of this paper is to give a short review of TS methodology and to illustrate its power in applications to two different types of real-life problems. The first one is a combinatorial assignment problem: assigning students to exams at the universities. The second one is a continuous optimal design problem: a global optimization problem arising from the spread spectrum radar polyphase code design in telecommunications. Although TS was originally proposed to solve discrete optimization problems, here we show that it can be successfully applied to a continuous case.

The paper is organized as follows: In Section 2 we briefly describe the basic concepts of the methodology as well as some components of its more sophisticated versions. Sections 3 and 4 contain applications to the assignment problem and the optimal design problem, respectively. In both cases some numerical experiments are reported.

2. TABU SEARCH METHODOLOGY

TS is a general iterative procedure based on the well-known local search principle:

A neighborhood structure is introduced to the solution space X in the following way: Each $x \in X$ has an associated set $N(x) \subset X$, $x \notin N(x)$, called the **neighborhood** of x. $N(x)$ is defined as the set of all $y \in X$ that can be obtained directly from x by a modification called a **move** $m(x,y)$ from x to y.

The procedure starts from an initial feasible solution and at each step moves from the current solution to another one in its neighborhood , trying to reach an optimal solution. At step k a subset $N'(x^k)$ of the neighborhood $N(x^k)$ of the current solution x^k is constructed and the best solution in $N'(x^k)$ is chosen as the next solution x^{k+1} (even if $f(x^{k+1}) \geq f(x^k)$). In this way TS allows ascent moves and, consequently, avoids being trapped in local minima (which is the main weakness of standard descent algorithms).

In order to guide the search process in an intelligent manner, TS procedure incorporates a **flexible memory structure** as its essential component. Generally speaking, at each step k the procedure maintains a selected history H of the previous search, i.e. a record which memorizes the characteristics of some previously generated solutions. Then, instead of $N(x^k)$, a modified neighbourhood $N(x^k,H)$ is defined according to history H, and a set $N'(x^k)$ is selected such that $N'(x^k) \subseteq N(x^k,H)$.

The next solution x^{k+1} is usually obtained by minimizing the objective function $f(x)$ over $N'(x^k)$. But in general x^{k+1} can be chosen to minimize an evaluation function $f(x,H)$ which also depends on history H.

Summarizing the above considerations, the TS procedure can be expressed in the most general way as follows:

Tabu search algorithm

Initialization: Select an initial solution $x^1 \in X$.
$x^*=x^1$, $f^*=f(x^1)$
The history record H is empty.

Iteration step: For $k=1,2,\ldots$

- Define a modified neighborhood $N(x^k,H)$ and an evaluation function $f(x,H)$.
- Generate a set $N'(x^k)$ as a subset of $N(x^k,H)$.
- Determine x^{k+1} by minimizing $f(x,H)$ over $N'(x^k)$.
- If $f(x^{k+1}) < f(x^*)$, then $x^*=x^{k+1}$ and $f^*=f(x^{k+1})$
- Update the history record H

End: If the stopping criterion is satisfied, then stop.

A crucial part of the TS approach is the definition of an appropriate flexible memory structure. In its basic version only one such type of structure is applied: the **short-term** (or the **recency-based**) memory structure, where at each step the record H memorizes a selected history of the most recently generated solutions. H is formally defined using so-called tabu lists $T_1, \ldots, T_p$ with lengths $L_1, \ldots, L_p$, where $p \geq 1$. A **tabu list** T_i, $i \in \{1, \ldots, p\}$, represents a list of one or several selected attributes of L_i most recently performed moves or of the corresponding reverse moves (i.e. at step k, T_i contains attributes of $m(x^l,x^{l+1})$ or $m(x^{l+1},x^l)$ for $l = k\text{-}L_i, \ldots, k\text{-}1$). Now, $H = T_1 \cup \ldots \cup T_p$. The tabu lists define the tabu status of any move $m(x^k,x)$ from the current solution x^k to some of its neighbors x. The move $m(x^k,x)$ is **tabu** if at least one of its attributes belongs to H. If $m(x^k,x)$ is **tabu**, then x is forbidden to be a candidate for the next solution x^{k+1}. A tabu list T_i is maintained as follows: At each step k the corresponding attributes of the move $m(x^k,x^{k+1})$ or of the reverse move $m(x^{k+1},x^k)$ enter the list T_i and stay there in the next L_i iterations. The recency-based memory structure is a core component of TS with the main role to prevent cycling back to some previously generated solutions and to diversify the search process, i.e. induce the search of new subregions of the solution space X.

The attribute-based tabu lists can be too restrictive and forbid some moves to solutions that have never been reached before. Therefore, TS offers the possibility of canceling the tabu status of a move when it leads to a "good enough" solution. This is realized using **aspiration criteria**. In the basic version of TS an aspiration criterion usually has the following general form: At step k, let move $m(x^k,x)$, $x \in N(x^k)$, be tabu. Then, two values are determined: **aspiration level** $a(m(x^k,x))$ and **threshold value** $A(m(x^k,x))$. If $a(m(x^k,x)) < A(m(x^k,x))$, then the tabu status of $m(x^k,x)$ is cancelled. The simplest aspiration criterion of such a form is: the tabu status of $m(x^k,x)$ is cancelled if solution x is better than the currently best solution x^* obtained up to step k, i.e. $a(m(x^k,x))=f(x)$ and $A(m(x^k,x))=f(x^*)$.

Applying the tabu list restrictions and an aspiration criterion (if it exists), the set $N(x^k,H)$ is defined as the set of all $x \in N(x^k)$ such that $m(x^k,x)$ is either not tabu or $a(m(x^k,x)) < A(m(x^k,x))$. As $N(x^k,H)$ is usually too large to be examined efficiently, a set $N'(x^k)$ is chosen such that $N'(x^k) \subset N(x^k,H)$ and $|N'(x^k)| << |N(x^k,H)|$. The set $N'(x^k)$ can be obtained by generating randomly a specified number of neighborhood solutions. But in some more intelligent applications, it can represent a set of "preferred" solutions, with the structure depending on the characteristics of the particular problem.

In the case of short-term memory, the **next solution** $x^{k+1} \in N'(x^k)$ is chosen to minimize the objective function $f(x)$ over the set $N'(x^k)$ (i.e. $f(x,H)=f(x)$ at each step of the procedure). This local optimization problem may sometimes be a nontrivial one and a special heuristic could be required to find the solution.

The initial solution x^1 can be randomly chosen or obtained by a special heuristic designed for the particular problem. Applying such heuristics is recommended in problems where "good" solutions have a special structure that cannot be easily reached starting from a random solution.

The stopping criterion can have the following forms: the procedure is terminated if the number of consecutive iterations, performed without any improvement of the currently best objective function value f^*, is greater than a specified number. If the minimum value of $f(x)$ is known in advance, then the process can be interrupted as soon as this value is reached.

Extensive literature exists devoted to the main concepts of TS methodology. A general introduction to TS can be found e.g. in [10], [14], [16], [18], [19], while a more detailed elaboration of its basic elements with a comprehensive bibliography is given e.g. in [7], [8], [13].

One of the main strategic principles of TS is that the search process should ensure an appropriate periodical alternation between local intensification and global diversification phases, i.e. between intensifying the search in a "promising" subregion and driving the search into new subregions. But sometimes the short-memory structure is not sufficient to realize such a principle.Therefore, in refined TS versions two additional kinds of memory structures can be incorporated: the **intermediate** and the **long-term** memory.

When an intermediate memory is applied at step k, the history record H maintains attributes of some "good" solutions from the past. Solutions with such attributes are forced to be the next ones by appropriately chosen $N(x^k,H)$ and $f(x,H)$. In this way the search can be implicitly focused on a "good" subregion, i.e. an intensification phase is induced. In the case of long-term memory, H keeps track of attributes that have not appeared for a long time (or whose appearence was less frequent) in a sequence of previously generated solutions. Favoring solutions with attributes from H, the next solution can be significantly different from those previously generated, which causes diversification (for the main ideas of refined TS versions and how they are realized, see [13], [14]).

In order to implement TS methodology in a particular problem, a number of tactical choices have to be made: the specification of the objective function and the solution space (if the problem is not originally formulated as an optimization one), the definition of the basic neighborhood structure, the selection of move attributes for tabu lists and the lengths of tabu lists, the organization of longer-term memories in refined versions, etc. Most of these choices strongly depend on the specific structure of the problem. According to their generality, they provide enough opportunities to create an intelligent search procedure which can imitate human reasoning or to apply learning rules based on Artificial Intelligence principles. Therefore, TS can be viewed as a metaheuristic.

TS (either in a basic or refined form) has been successfully applied to almost all well-known standard combinatorial optimization problems (e.g. traveling salesman, vehicle routing, quadratic assignment, graph coloring, etc.) and to a large number of hard practical problems (job shop, flow shop, machine scheduling, electric circuit design, etc.). In most of these applications the obtained solutions were superior to the best previously found by other existing methods (for a systematic survey of TS applications, with corresponding bibliography, see [13], some parallel implemen- tations are reviewed in [20]). Numerous experiments have shown that until now there have been no general rules for creating the best possible choice of TS components in each particular case. Namely, in most of the cases, this choice was determined by experimentation.

Finally, let us mention that there are a few theoretical results dealing with the convergence of TS, but only in the case of the probabilistic version (see [5]). Also, some papers show that the TS concept can be applied to continuous optimization problems [1], [11], [17] .

3. APPLICATION TO A COMBINATORIAL ASSIGNMENT PROBLEM

The following problem of assigning students to exams during an examination period arose at the Law School of Belgrade University:

In an examination period we are given a set of subjects for which exams should be performed. For each subject we know in advance the exam days, i.e. the days in which the exam can be organized. Each exam day of a given subject has its capacity: the maximal number of students that can be examined on that day. Each student specifies the subjects which he wants to take and should be assigned to one of the exam days for each subject. The assignment of all students should satisfy the following additional requirements: a student should have at least a prescribed number of free days between two consecutive exam days and the capacities of exam days should not be violated for any subject. The problem of finding an assignment of students to exams under the described constraints will be denoted by (P).

Problem (P) belongs to a class of practical combinatorial problems which are very difficult both to model and to solve. The usual way to deal with such problems is to partition the set of requirements into hard and soft ones [4]. Then, soft require-

ments are penalized in the form of the objective function, while hard ones appear as constraints in the corresponding combinatorial optimization problem. Using such an approach, problem (P) is modeled as a combinatorial optimization problem (CP) on an associated dynamically weighted graph in the following way:

Let $S = \{S_1, \ldots, S_m\}$ be the set of all subjects. The set of applications of all students defines combinations $K_1, \ldots, K_p$ of the subjects. The combination K_i, $i \in \{1, \ldots, p\}$, is a subset of subjects from S such that there exists at least one student whose application specifies K_i. Let q_i be the number of subjects in K_i and s_i be the total number of students taking this combination. Each subject S_k has n_k exam days d_{kl}, $l=1, \ldots, n_k$, where d_{kl} is equal to the ordinal number of the exam day in the examination period. The capacity of d_{kl} is denoted by c_{kl}. Tables 1 and 2 give the data for a hypothetical situation with 3 subjects and 4 students taking 3 combinations. For example, exam days of the subject S_2 are the 11th and the 20th day of the exam period with capacities 2 and 4, respectively.

Table 1. Subjects and exam days

Subject S_k	Ordinal numbers of exam days (d_{kl})	Capacities of exam days (c_{kl})
S_1	8, 12, 13	1, 3, 3
S_2	11, 20	2, 4
S_3	10, 14	4, 7

Table 2. Combinations of subjects

Combination K_i	Subjects of K_i	No. of students (s_i)
K_1	S_1, S_2	2
K_2	S_1, S_3	1
K_3	S_1, S_2, S_3	1

We can associate to the problem a weighted graph G as follows: Each d_{kl} is represented as a node of the graph. Two nodes $d_{k_1 l_1}$ and $d_{k_2 l_2}$, $k_1 \neq k_2$, are connected with an edge if:

- there is a K_i such that $S_{k_1}, S_{k_2} \in K_i$ and
- $|d_{k_1 l_1} - d_{k_2 l_2}| > d_{min}$, where d_{min} is the prescribed number of free days between two consecutive student exams.

In this way a subject can be viewed as a set of nonadjacent nodes corresponding to its exam days and, consequently, each combination is presented as a subgraph of G. Let $C_i = \{C_{ij}, j=1, \ldots, r_i\}$ be the set of all cliques (maximal complete subgraphs) of K_i which have exactly q_i nodes. The associated graph G for the hypothetical example is presented in Fig. 1. Here e.g. $C_2 = \{C_{21}, C_{22}\}$, where $C_{21} = \{8, 14\}$, $C_{22} = \{13, 10\}$.

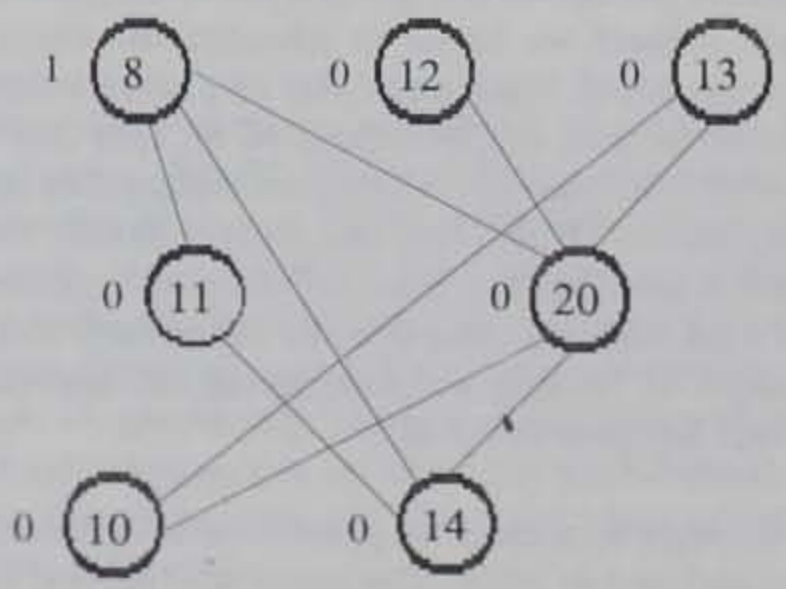

Figure 1. The associated graph

A **feasible assignment** is an assignment $x=\{x_{ij}, i=1,...p, j=1,...r_i\}$ where x_{ij} is a nonnegative integer assigned to clique C_{ij} such that

$$\sum_{j=1}^{r_i} x_{ij} = s_i, \quad i=1, \dots, p$$

In fact, clique C_{ij} represents a feasible schedule of exam days for the subjects of K_i, while x_{ij} is the number of students using this schedule. Given a feasible assignment x, for each node d_{kl} we can calculate the corresponding violation v_{kl} of its capacity c_{kl} as

$$v_{kl} = \max \left\{ \sum_{(i,j):d_{kl} \in C_{ij}} x_{ij} - c_{kl}, 0 \right\}$$

In this way each feasible assignment determines the weights v_{kl} corresponding to the nodes of G, i.e. G has dynamically weighted nodes. Fig. 1 indicates violations v_{kl} corresponding to the feasible assignment given in Table 3 (the only violation $v_{11}=1$ is at the node $d_{11}=8$).

Table 3. A feasible assignment

Combination K_i	Cliques of K_i (C_i)	No. of assigned students (x_{ij})
K_1	{8, 11}	1
	{12, 20}	1
	{8, 20}	0
	{13, 20}	0
K_2	{13, 10}	1
	{8, 14}	0
K_3	{8, 20, 14}	1
	{8, 11, 14}	0
	{13, 20, 10}	0

Now the problem (CP) is formulated as follows: find such a feasible assignment for which the value of the objective function, which is defined as the maximal violation of all nodes, is minimal. Any optimal solution of (CP) is called the optimal assignment, while the corresponding maximal violation is the optimal value. Obviously, if the optimal value is 0, then the optimal assignment is a solution of the problem (P). If the optimal value is greater than 0, then the set of solutions of (P) is empty.

The evaluation of the objective function of problem (P) requires that the sets of cliques C_i, $i=1, \dots, p$ are known. Hence, before the assignment is attempted, clique generation preprocessing is to be done. Since finding all cliques is too time consuming (even constructing a single clique is NP-hard), the preprocessing is not trivial. In [2] we propose a special heuristic which generates "satisfactory" subsets $\overline{C}_i \subset C_i$, $i=1, \dots, p$. A satisfactory $\overline{C}_i$ should have cardinality large enough to prevent the accumula-

tion of too many students at the same clique, but significantly smaller than that of C_i. At the same time, $\overline{C}_i$, $i=1, \ldots, p$, should be structured in such a way that the same node is not contained in too many cliques, in order to avoid the accumulation of students at the same node. The clique generation heuristic simultaneously produces a feasible assignment.

Here we shall develop a TS procedure to find an optimal assignment corresponding to the cliques $C_{ij} \in \overline{C}_i$, $i=1, \ldots, p$, $j=1, \ldots, \bar{r}_i$. The **solution space** X is the set of all feasible assignments to the cliques from $\overline{C}_i$, $i=1, \ldots, p$, while the **objective function** $f(x)$ is the maximal violation of all nodes for the assignment $x \in X$. The **initial** feasible solution x^1 is the feasible assignment produced by the special heuristic.

The neighbourhood $N(x)$ is defined as follows: Solution $y=\{y_{ij}\}$ is a **neighbor** of solution $x=\{x_{ij}\}$, $x,y \in X$, if y can be obtained from x by the following modification:

(M) Select a combination K_i for which $\overline{C}_{ij}$ is not singleton, and two cliques C_{ij_1}, $C_{ij_2} \in \overline{C}_{ij}, j_1 \neq j_2$, such that $x_{ij_1} > 0$. Then $y_{ij_1} = x_{ij_1} - 1$, $y_{ij_2} = x_{ij_2} + 1$, while $y_{ij} = x_{ij}$ for all $j \neq j_1, j_2$.

The modification (M) defines a **move** $m(x,y)$ in the neighborhood of x. In fact, y is obtained from x by moving only one student from the clique C_{ij_1} (called the **from-clique**) to the clique C_{ij_2} (called the **to-clique**) of the same combination.

In order to prevent cycling back to some recently generated solutions, two **tabu lists** T_1 and T_2 with lengths L_1 and L_2 are introduced. At step k the lists T_1 and T_2 are updated in the following way: Let C_{ij_1} be the from-clique of the move $m(x^k, x^{k+1})$. Now a node d_{kl} of C_{ij_1}, which has the maximal weight, is entered in both lists. T_1 disallows decreasing the weight of d_{kl} in the next L_1 iterations, while T_2 forbids increasing this weight in the next L_2 iterations. A move is considered tabu if either its from-clique contains a member from T_1 or its to-clique contains a node from T_2. As T_1 could forbid some good moves which decrease the objective function value, L_1 should be smaller than L_2.

Now, generating a subset $N'(x^k)$ of the neighborhood $N(x^k)$ of the current solution x^k and moving to the next solution x^{k+1} are performed in a four-step procedure:

Step 1: Order all nodes according to decreasing weights corresponding to the solution x^k;

Step 2: Passing through the ordered nodes, select the first node d_{kl} for which at least one **admissible** move exists, i.e. such a move that it is not tabu and its from-clique contains d_{kl}, but its to-clique does not contain any of the already passed nodes.

Step 3: Among all admissible moves of d_{kl} find the **best** one, i.e. such a move for which the maximal weight of the nodes in its to-clique is minimal.

Step 4: Perform the best move and thus obtain x^{k+1}. Update current weights of the corresponding nodes and put d_{kl} to tabu lists.

The TS procedure terminates if either no improvement of the objective function value has been made during a given number of iterations or the optimal assignment x^* with $f(x^*)=0$ was obtained.

The power of the described TS procedure was tested on the examination period in September 1994: The length of the examination period was 20 days. There were 45 subjects, with 290 exam days, 5823 students and 715 combinations. The number of free days between two consecutive exam days for any student was at least 2.

In this example 33 combinations contained only one subject and involved a total number of 1135 students. As such students can be trivially assigned to exam days with free capacities, left after any assignment of all other students, TS was applied to the remaining 682 combinations with 4688 students. Input data statistics, presented in Table 4, can give insight into some aspects of the complexity of the problem and the associated graph G. For example, there were 207 combinations (subgraphs) with 3 subjects, where the average number of exam days (nodes) was 25 per combination, while the total number of students involved was 1370.

Table 4. Input data statistics

No. of subjects	2	3	4	5	6	7	8	9	10
No. of combinations	108	207	168	98	59	29	7	5	1
Average No. of exam days	17	25	34	42	48	57	72	79	71
Total No. of students	1820	1370	838	425	177	38	14	5	1

The special heuristic from [2] generated a total of 6107 cliques and a feasible assignment which violated given capacities at 17 nodes for 78 students. The distribution of violations is given in Table 5. Starting from this feasible solution, i.e. with $f(x^1)=19$, TS procedure with tabu lists lengths $L_1=5$, $L_2=9$ was applied. The procedure needed 279 iterations to reach the optimal assignment x^* with $f(x^*)=0$.

Table 5. Distribution of violations in the initial solution

Violations	1	2	3	7	9	10	11	16	19	$\Sigma=78$
N° nodes	4	3	1	1	2	2	1	2	1	$\Sigma=17$

4. APPLICATION TO A CONTINUOUS OPTIMAL DESIGN PROBLEM

In this section we shall consider a global optimization problem arising in the spread spectrum radar polyphase code design [3]. The problem is formulated as follows:

$$\underset{x\in X}{global\ \min}\ f(x)=\max\{\varphi_1(x), \ldots, \varphi_{2m}(x)\}$$
$$X=\{(x_1, \ldots, x_n)\in R^n \mid 0\le x_j \le 2\pi,\ j=1, \ldots, n\}, \qquad (1)$$

where $m=2n-1$ and

$$\varphi_{2i-1}(x)=\sum_{j=i}^{n}\cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right),\quad i=1,\ldots,n$$

$$\varphi_{2i}(x)=0.5+\sum_{j=i+1}^{n}\cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right),\quad i=1,\ldots,n-1$$

$$\varphi_{m+i}(x)=-\varphi_i(x),\quad i=1, \ldots, m$$

It is proved in [17] that problems of type (1) are special instances of a class of NP-hard problems. The attempt to solve (1) by an exact implicit enumeration technique failed for $n>5$. The difficulties are caused by the fact that it is not possible to get good upper and lower bounds of the minimum of $f(x)$ on subregions of X and, consequently, the number of considered subregions grows rapidly. This motivates the use of heuristic methods. Here we applied to (1) a general multi-level TS technique developed in [17] for global optimization problems. The basic components of this technique are specified below:

The **neighborhood** $N(x^k)$, $x^k\in X$, is defined in the usual Euclidean way:

$$N(x^k)=\{x\in X \mid 0<\|x-x^k\|\le\alpha_k\},$$

where $\alpha_k > 0$ is a radius which depends on the iteration number k, (i.e. the move $m(x^k,x)$ is here $x=x^k+\alpha d$, $0<\alpha\le\alpha_k$, $d\in R^n\setminus\{0\}$).

Several **tabu lists** are introduced, but in each iteration only one of them is active. Namely, when x^k is an interior point of X, after performing a move from x^k, the pair (x^k, β_k) enters the currently active tabu list. This pair defines a tabu cube

$$C(x^k, \beta_k)=\{x\in X \mid |x_j - x_j^k|\le\beta_k,\ j=1, \ldots, n\},$$

where β_k, $0<\beta_k<\alpha_k$, is the cube size which again depends on k. A tabu cube contains points which are forbidden to be visited in a prescribed number of future iterations with the same active tabu list. This number represents the length of the list. A point is tabu in an iteration if it belongs to the tabu region which is the union of all tabu cubes from the currently active tabu list.

The subneighbourhood $N'(x^k)$ is a finite subset of $N(x^k)$ which is generated using a finite set $D(x^k)$ of "good" move directions. In our case the definition of a good move direction is motivated by the structure of the minmax problem. Let for a given $x\in X$, $A(x)=\{i\in\{1, \ldots, m\} \mid f(x)=\varphi_i(x)\}$. If $i\in A(x)$ we shall say that φ_i is active at x. Generally speaking, it could be expected that at the optimal solution $A(x)$ is not a singleton. Therefore, during TS we shall try to generate points such that $A(x^k)\neq A(x^{k+1})$, i.e. to make moves which change active functions. We can then expect

that the segment $[x^k, x^{k+1}]$ contains "valley" points with low objective function values. Such points can be obtained by a suitable discretization of the segment $[x^k, x^{k+1}]$. Having this in mind, a search direction at x^k can be considered good if it decreases active and "almost active" functions and/or increases functions which are not active. For example, suppose that at the point x^k the function φ_1 is active, functions $\varphi_2, \dots, \varphi_s$ are almost active (i.e. $\varphi_1(x^k) - \varphi_i(x^k) \le \varepsilon$, $i=2,\dots, s$) and $\varphi_{s+1}, \dots, \varphi_m$ are not active (i.e. $\varphi_1(x^k) - \varphi_i(x^k) > \varepsilon$, $i=s+1, \dots, m$) , where $\varepsilon > 0$ is a given parameter.

Let

$$G = \left\{ -\frac{\nabla\varphi_1(x^k)}{\|\nabla\varphi_1(x^k)\|}, -\frac{\nabla\varphi_1(x^k)}{\|\nabla\varphi_1(x^k)\|} - \frac{\nabla\varphi_2(x^k)}{\|\nabla\varphi_2(x^k)\|}, \dots, -\frac{\nabla\varphi_1}{\|\nabla\varphi_1(x^k)\|} - \frac{\nabla\varphi_s}{\|\nabla\varphi_s(x^k)\|}, \right.$$
$$\left. -\frac{\nabla\varphi_1(x^k)}{\|\nabla\varphi_1(x^k)\|} + \frac{\nabla\varphi_{s+1}(x^k)}{\|\nabla\varphi_{s+1}(x^k)\|}, \dots, -\frac{\nabla\varphi_1(x^k)}{\|\nabla\varphi_1(x^k)\|} + \frac{\nabla\varphi_m(x^k)}{\|\nabla\varphi_m(x^k)\|} \right\}$$

and $G_N = \left\{ \frac{g}{\|g\|} \mid g \in G \right\}$.

Then we can take $D(x^k) = \{d \in G_N \mid A(x^k) \neq A(x^k + \alpha_k d)\}$ (if such $D(x^k) = \varnothing$, we set $D(x^k) = G_N$). Given the set $D(x^k)$, $N'(x^k)$ is generated as follows:

For each $d \in D(x^k)$ generate the point $x(d) = x^k + \alpha_k d$. If $x(d) \notin X$ redefine $x(d)$ as $x(d) = x^k + \lambda d$, where $\lambda > 0$ is such that $x^k + \lambda d \in \partial X$. Set $N'(x^k) = \{x(d), d \in D(x^k) \mid x(d)$ is not tabu$\}$.

The best neighbor $x^{k+1} \in N'(x^k)$ is determined such that $f(x^{k+1}) = \min f(x)$ among all x satisfying $x \in N'(x^k)$. (If either $N'(x^k) = \varnothing$ or x^k is a boundary point of X, i.e. $x^k \in \partial X$, then a random restarting of the TS process is made). Let x_s^{k+1} be the discretization point of the segment $[x^k, x^{k+1}]$ with the minimum objective function value. The currently best solution and the currently best objective function value are updated using both x^{k+1} and x_s^{k+1}.

The step size α_k is chosen to balance two contradictory requirements: It should be sufficiently large in order to provide a search of the entire set X and it should be small enough to attain a satisfactory solution. The balance is achieved using a **multi-level search** strategy based on the following principles: Let $A_1 < \dots < A_w$ be the w possible choices (levels) for the step size during TS procedure. These levels are alternated in a periodical manner, using a given sequence of positive integers $q_1 < \dots < q_w$, where q_i represents the maximal number of consecutive iterations which can be performed with steps at levels $A_1, \dots, A_i$. Let $c_1, \dots, c_w$ be the level counters which are initially set at 0. Then the step size α_k is obtained as follows:
If $c_i \neq q_i$ for all $i=1, \dots, w$, then set $\alpha_k = A_1$ and $c_j = c_j + 1$, $j=1, \dots, w$. Otherwise, find the biggest $i \in \{1, \dots, w\}$ such that $c_i = q_i$. If $i<w$, then set $\alpha_k = A_{i+1}$ and $c_j = 0$, $j=1, \dots, i$, $c_j = c_j + 1$, $j=i+1, \dots, w$. If $i=w$, TS is terminated.

For example, if $q_1=2$, $q_2=3$, $q_3=10$, the pattern of the levels which are used is $A_1A_1A_2A_3A_1A_1A_2A_3A_1A_1$.

According to the described multi-level search strategy, the level A_{i+1} is applied after q_i consecutive iterations with steps from the set $\{A_1, \ldots, A_i\}$. In order to improve the efficiency of the search, this level can be forced earlier, if there was no improvement of the objective function value in the last u_i iterations, where $u_i < q_i$. In this case TS procedure stops after q_w iterations or if there was no improvement in the last u_w iterations.

If a multi-level search strategy is used, it is natural to introduce w tabu lists $T_1, \ldots, T_w$ with lengths $L_1, \ldots, L_w$. Let $B_1 < \ldots < B_w$ be the corresponding tabu cube sizes. If $\alpha_k = A_j$, then the tabu list T_j is active at the k-th iteration and the pair (x^k, β_k) enters the list, where $\beta_k = B_j$. Details of the outlined TS procedure can be found in [17].

We shall illustrate the numerical performance of the multi-level TS procedure on the special instances of (1) obtained for $n=5,10,15$. For $n=5$ we applied a two-level search strategy, while for $n=10$ and $n=15$ we used three levels. The tolerance, used in the definition of the set $D(x^k)$, was $\varepsilon=0.00001$. The discretization of the segment $[x^k, x^{k+1}]$ was uniform, with 10 discretization points. The initial point x^1 was in all cases the center of the feasible set X, i.e. $x_j^1 = \pi$, $j=1, \ldots, n$. The remaining parameters were set up by experimentation.

Experiments have shown that, independently of the dimensions, the reasonable choices for tabu list lengths are, in the case of a two-level strategy, $L_1=100$, $L_2=20$, and in the case of the three-level strategy $L_1=200$, $L_2=100$, $L_3=20$. The size of the tabu cube was always chosen to be half of the corresponding step size ($B_i=A_i/2$). Different values for A_i and q_i were used in order to determine the best multi-level search strategy. In all cases u_i is taken to be 40% of q_i. Table 6 shows the values of parameters for which the best objective function value is obtained, also this value and the corresponding iteration number. For the sake of comparison, Table 6 also contains the results obtained by the Metropolis algorithm, reported in [3].

Table 6. Tabu search vs. Metropolis

n	A_1	A_2	A_3	q_1	q_2	q_3	The best value	Iteration number	Metropolis
5	0.04	2.0		99	20000		0.3391	15073	0.3403
10	0.02	0.2	3.0	4	999	20000	0.4615	7175	0.5896
15	0.03	0.3	3.0	4	999	20000	0.4383	2913	0.7455

The obtained results are quite satisfactory. Let us note that further improvements could be obtained by a local search. In fact, the one-level TS algorithm with a small step can itself be used as a local search procedure.

5. CONCLUSIONS

The paper presents the main concepts of Tabu search, which is one of the most popular modern heuristic methodologies. It is demostrated that it can be successfully applied not only to large combinatorial problems but also to global optimization minmax problems. Tests were performed on real-life examples arising in two different fields of application.

REFERENCES

[1] Brimberg, J., and Mladenović, N., "Application of Tabu search in the multy source Weber problem", GERAD report G-94-39, McGill University, Montreal, Canada, 1994.

[2] Čangalović, M., Kovačević-Vujčić, V., Ivanović, L., and Dražić, M., "A heuristic approach to the problem of assigning students to exams", in: S.Vujić (ed.), *Proceedings of Yugoslav Symposium on Operations Research*, Donji Milanovac, Yugoslavia, 1995, 313-316.

[3] Dukić, M.L., and Dobrosavljević, Z.S., "A method of spread spectrum radar polyphase code design by nonlinear programming", accepted for publication *in Telecommunications and Related Techniques*.

[4] Eiselt, H.A., and Laporte, G., "Combinatorial optimisation problems with soft and hard requirements", *Journal of Operational Research Society*, 38(1987)785-795.

[5] Faigle, U., and Kern, W., "Some convergence results for probabilistic Tabu search", *ORSA Journal of Computing*, 4(1992) 32-37.

[6] Glover, F., "Future paths for integer programming and links to artificial intelligence", *Computers and Operation Research*, 13(1986) 533-549.

[7] Glover, F., "Tabu search- Part I", *ORSA Journal on Computing*, 1(1989) 190-206.

[8] Glover, F., "Tabu search- Part II", *ORSA Journal on Computing*, 2(1990) 4-32.

[9] Glover, F., "Artificial intelligence, heuristic framework and Tabu search", *Managerial and Decision Economics*, 11(1990) 365-375.

[10] Glover, F., "Tabu search: A tutorial", *Interfaces*, 20(1990)74-94.

[11] Glover, F., "Tabu search for nonlinear and parametric optimization (with links to genetic algoritms)", Technical Report, Graduate School of Business and Administration, University of Colorado, Boulder, 1991.

[12] Glover, F., and Greenberg, H.J., "New approaches for heuristic search: A bilateral linkage with artificial intelligence", *European Journal of Operational Research*, 39(1989) 119-130.

[13] Glover, F., and Laguna, M., "Tabu search", in: C.R.Reevs (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, Oxford, 1993, 70-150.

[14] Glover, F., Taillard, E., and de Werra, D., "A user's guide to Tabu search", *Annals of Operations Research*, 41 (1993) 3-28

[15] Hansen, P., Jaumard, B., "Algorithms for the maximum satisfiability problem", RUTCOR Research Report RR 43-87, Rutgers University, New Brunswick, New Jersey, 1987.

[16] Hertz, A., and de Werra, D., "The Tabu search metaheuristic: How we used it", *Annals of Mathematics and Artificial Intelligence*, 1(1990)111-121.

[17] Kovačević-Vujčić, V., Čangalović, M., Ašić, M., Ivanović, L., and Dražić, M., "Tabu search methodology in global optimization", submitted for publication in *Computers & Mathematics with Applications*.

[18] Pirlot, M., "General local search heuristics in combinatorial optimization: A tutorial", *Belgian Journal of Operations Research*, 32(1992)7-67.

[19] Pirlot, M., "Heuristic search methods", a tutorial paper at EURO XIII/OR 36 Conference, University of Strathclyde, Glasgow, 1994.

[20] Roucairol, C., "Parallel computing and combinatorial optimization", research review paper at EURO XIII/OR 36 Conference, University of Strathclyde, Glasgow, 1994.