Journal of Heuristics, 7: 47–58 (2000)

On the Convergence of Tabu Search

SAÏD HANAFI
LAMIH-Recherche Operationnelle et Informatique UMR CNRS 8530, Universite de Valenciennes, LeMont Houy, F-59313 Valenciennes Cedex 9-France
email: said.hanafi@univ-valenciennes.fr

Abstract

The Tabu Search (TS) meta-heuristic has proved highly successful for solving combinatorial and nonlinear problems. A key aspect of TS consists of using adaptive forms of memory to forbid the search process to revisit solutions already examined unless the trajectory to reach it is different. In Glover (*ORSA Journal on Computing*, 1990, 2, 4–32) a special memory design was proposed together with a choice rule for handling the situation where the method was compelled to revisit solutions already encountered. This proposal, which specified the exploration should resume from the earliest solution visited in the past, as accompanied by the conjecture that such a choice has implications for finiteness in the zero-one integer program and optimal set membership examples. Up to now numerous applications of TS in various areas of research are available, however, we are aware of only a few results concerning the convergence of TS. In this paper, we prove that Glover's conjecture is true if the neighborhood employed is strongly connected, yielding a "reversible" path from each solution to every other solution.

Key Words: tabu search, convergence, graph

1. Introduction

We consider a combinatorial optimization problem stated in the following way:

$$(P) \qquad \text{Minimize } c(x) \text{ subject to } x \in X \subseteq E$$

where E is the space of potential solutions and X is the set of feasible solutions that meet certain constraints defined by the problem application. The solution set E is often defined by a set of decision variables whose values are directly related to the model that is used to formulate the problem. The objective function c is a linear or nonlinear mapping that assigns a real cost value $c(x)$ to each solution x. The problem is to find a globally optimal solution $x^* \in X$ such that $c(x^*) \leq c(x)$ for all $x \in X$.

Most optimization techniques (heuristic and exact methods), for solving the problem are iterative procedures which start with an initial solution (feasible or infeasible). Next, the algorithm tries repeatedly to construct a new solution from a current solution by searching neighborhoods. The process continues to generate neighboring solutions until a certain stopping criterion is satisfied. Each solution $x \in E$ has an associated neighborhood $N(x)$, which is a subset of E, and each solution $x' \in N(x)$ reached from the solution x is called a move.

A basic version of a neighborhood search method is the descent method, also called iterative improvement, in which the current solution is replaced by its neighboring solution with lower cost. A classical descent algorithm where all solutions generated are feasible is presented schematically as follows:

Descent Method

Step 0 : Choose an initial solution $x \in X$; set the iteration counter $k = 1$; the current solution $x^k := x$;

Step 1 : Let $N^+(x^k)$ be the improving feasible neighborhood set of x^k, i.e.,

$$N^+(x^k) := \{x' \in N(x^k) \cap X \text{ such that } c(x') < c(x^k)\}$$

If $N^+(x^k)$ is empty, then x^k is the local optimum and the method stops.

Step 2 : Otherwise, select a solution $x^{k+1} \in N^+(x^k)$; set $k = k + 1$ and go to *Step 1*.

The final solution x^* produced by a descent method is called local optimum, since it is at least as good or better than all feasible solutions in its neighborhood.

For several settings in which problems have a particular structure, the descent method assures that the final solution x^* is globally optimal. Particularly, in the context of convex analysis, the convexity properties of the function and the domain assure that any local optimum is also a global optimum. A collection of descent algorithms has been developed to solve combinatorial problems exactly (i.e., with a guarantee of optimality) of exploiting the properties and information concerning the data (c, X, E).

Unfortunately, many problems are NP-Complete (Garey and Johnson, 1979), and the procedures that are exact for specially structured instances often yield a poor local optimum. In the last decade, several meta-heuristics for guiding the search to overcome local optimality have been proposed. Some of the more popular ones (and associated general references) include Tabu Search (TS) (Glover and Laguna, 1997), Simulated Annealing (SA) (Kirkpatrick et al., 1983), Genetic Algorithms (GA) (Goldberg, 1989), Greedy Randomized Adaptive Search Procedures (GRASP) (Feo et al., 1994), which extend the descent method's scheme in various ways. Also see (Reeves, 1993; Osman and Laporte, 1996) for further references to different applications.

An obvious alternative to a descent method is to accept a nonimproving move to escape from a local minimum. This is done in both SA and TS, for example. In SA, the probability of acceptance is based only on the change of the objective function between the current solution and its neighbor and the number of iterations performed so far. By contrast, TS employs strategic memory-based evaluation criteria to escape local optima. That is, the evaluation of a move incorporates penalties and/or inducements determined by the history of the search.

Another simple option to obtain good solution consists of enlarging the neighborhood while its structure remains the same along the search. For example, a common strategy in the traveling salesman problem (TSP) is to pass from a 2-exchange heuristic to a 3-exchange (or higher k-exchange) neighborhood. Unfortunately, if the best solution in the whole neighborhood is sought at each step, without exploiting the structure of the problem, this approach evidently proves very expensive.

Another option is to extend the local search to consider more than one solution at each iteration, such as in GA and Scatter Search (Glover and Laguna, 1997). A fourth alternative is to modify the neighborhood's structure during the search. Variable Neighborhood Search (VNS) method (Mladenovic and Hansen, 1996), for example, proceeds to a systematic change of the neighborhood's structure. VNS explores increasingly distant neighborhoods of the current solution, and jumps from there to a new one if and only if an improvement has been made. A component strategy of TS, called strategy oscillation, similarly provides for exploring domains progressively more distant from the current solution, commonly cycling back through closer domains. This approach dynamically changes the neighborhood's structure by excluding some solutions and incorporating others, depending on a selective history of the states meet during the search.

The main components of TS are the introduction of adaptive memory and responsive exploration, as a basis for solving combinatorial and nonlinear problems. Flexible memory is subdivided into short-term and longer-term memory and typically incorporates recency-based memory and frequency-based memory. The short-term memory usually relies on recency-based memory to keep track of the most recent solution attributes that have been modified during the search processes. The responsive exploration is based on the supposition that a bad strategic choice can yield more information than a good random choice. A key concept of TS consists of forbidding the search process to revisit solutions already encountered unless the trajectory to reach them is different. The goal of TS is not to forbid cycling, but to assure that different attractive search paths are explored. Indeed, "good choices" for searching a neighborhood may compel preceding solutions to be revisited before exploring the whole domain.

In Tabu Search Part 2, Glover (1990), proposed a chronological order for revisiting solutions already encountered, restarting the exploration from the earliest solution visited in the past. He conjectures that such a choice has implications for finiteness in the zero-one integer program and optimal set membership examples. In this paper, we prove that Glover's conjecture is true if the definition of the neighborhood allows us to explore the whole search space.

The outline of this paper is as follows. Section 2 reviews the simplest form of tabu search. In Section 3, the finite algorithm to avoid cycling is given. Finally, the remainder of the paper is devoted to the proof of the convergence of a form of TS as conjectured by Glover.

2. Simple tabu search algorithm

An important distinction in TS is to differentiate between short term and longer-term memory. In some applications, using short-term memory structures is sufficient to produce very good solutions. However, in general, including longer-term memory and its associated strategies (intensification and diversification) makes TS more efficient. In the following, to illustrate our topic, we consider a simple TS algorithm incorporating only rigid memory, whose main goal is to prevent cycling.

We shall use the symbols "+" and "−" respectively as union and difference operators of two sets. A rudimentary version of tabu search may be expressed in the following manner:

Simple Tabu Search algorithm

Step 0 : Choose an initial solution $x \in N$ to be feasible or infeasible, and set

- the iteration counter $k = 1$;
- the current solution $x^k = x$;
- the best solution $x^* = x^1$;
- $S^k = \{x^1\}$ the set of solutions on the itinerary upon the iteration k;

Step 1 : Select an *unvisited* solution $x^{k+1} \in (N(x^k) - S^k)$ with respect to certain criterion;
Step 2 : Update the best solution x^* and the trajectory $S^{k+1} = S^k + \{x^{k+1}\}$;
Step 3 : If stopping criterion is satisfied, the method stops.
Step 4 : Otherwise, set $k = k + 1$; go to *Step 1*.

Starting with an initial solution (feasible or infeasible), at each iteration a new solution is examined according to some criterion, which is generally not restricted to the objective function. Note that TS attaches a great importance to *Step 1*, of judiciously selecting a next solution via candidate list strategies. The purpose is to yield a good trade-off between the quality of solution and the amount of computer time and memory requirements to find it. See, for example, Glover and Laguna (1997), for more elaborate details.

To prevent cycling, the rigid memory records the complete solutions encountered along the search up to iteration k, as denoted S^k in the above algorithm. (A mechanism called the Reverse Elimination method (REM) introduced in Glover (1990) makes it possible to implicitly maintain such a record for various types of neighborhoods without requiring complete solutions to be stored in memory.) Regardless of the implementation details, short-term memory functions provide one of the important foundations of the TS methodology, constituting one of the primary means for continuing beyond local optima, by allowing the execution of nonimproving moves coupled with the modification of the structure of subsequent solutions. Instead of storing complete solutions as suggested by the simple version of TS sketched above, however, these structures are based on recording attributes.

In one type of memory structure, attributes of the prohibited moves are stored on a so-called Tabu List (TL), where they stay for a specified number of iterations and then are eliminated by releasing them from their tabu status. The rules for managing TL are critical. TL can be defined and handled in various ways depending on the problem to be solved. A naive static approach forbids moves by reference to attributes that are retained on TL for a certain number of iterations t, fixed through the search processes. This way of managing TL can be implemented by means of a circular list, eliminating, at each iteration, the oldest "tabu-active" attribute to make room for the new one. For such a simple static approach, the size of TL must be neither too small if it is to escape from basins of attraction, nor too large if it is to provide flexibility in choosing moves.

By contrast, a dynamic approach makes it possible to vary the tenure of the tabu status according to the attributes considered, thus implicitly or explicitly modulating the size of TL according to the state of the search. Some of the dynamic management methods determine a tabu status based on sequential relationships between the selected moves to avoid certain types of cycling. This occurs, for example, in the Cancellation Sequence method and Reverse Elimination Method proposed in Glover (1990) and implemented in Dammeyer and Voss

(1993). Another form a systematic dynamic tabu list consists of subdividing the list into a static part and a dynamic part, as in the moving gap approach implemented in Hübscher and Glover (1994).

The usual stopping criterion used in TS algorithms is based on the maximum number of iterations, or the maximum number of iterations between two improvements, or the specified amount of computer time allowed. In the remainder of this paper, we assume that the optimization problem (P) is unconstrained, i.e. $X = E$. In addition, we suppose the set of solution E is finite and the stopping criterion in CTS algorithm is the condition $(S^k = E)$.

It is well known that the design of the neighborhood structure is crucial for the efficiency of an iterative method. An iterative solution method can be interpreted from a graph perspective. Given a neighborhood structure N, let $G_N = (V, A)$ denote a graph induced by N, where the node set V is the set of solutions E. Associated with this is an edge set A defined by $e = (x, x') \in A$ if $x \in N(x')$. The sequence of trial solutions obtained by executing an iterative procedure starting from an initial solution x^r and proceeding to a terminal solution x^s, not necessarily the best one, corresponds to a path in G_N, noted $P(x^r, x^s)$, defined by the ordered pairs (x^i, x^{i+l}) for $i = r, \ldots, s-1$. The value $(s-r)$ identifies the "length" of the path.

Therefore, the iterative solution search method can be viewed as a walk in G_N. One step of an iterative procedure consists of moving from the current vertex to one of its adjacent vertexes. Generally, the imprint of the trajectory in graph G_N is a path in the case of local methods, while for certain meta-heuristics (such as GA, TS, GRASP, VNS, Scatter Search, ...) the itinerary constitutes a chain or (in "parallel" or "implicit-parallel" processes) a collection of chains.

We give below an example of a neighborhood currently used that may be helpful to clarify basic ideas. For pure zero-one integer programming problems (0–1 IP) where all decision variables x must receive zero or one values; the usual neighborhood of a solution x is

$$N_d(x) = \left\{ x' : \sum_{j=1}^{n} |x_j - x'_j| = d \right\}$$

(see Dammeyer and Voss, 1993) and (Hanafi et al., 1995, 1996)). We suppose that the objective function is nonlinear and that the set of feasible solutions corresponds to the set of n-dimensional binary vectors i.e. the unit-hypercube $X = \{0, 1\}^n$. For the case with constraints, the problem can often be transformed in an equivalent one by incorporating a penalty function which assures that constraints such as $Ax \leq b$ are satisfied at optimality.

Figure 1 shows the resulting graph G_{N1} for $n = 4$ and $d = 1$. This graph has $16 = 2^4$ vertices which corresponds to points of the hypercube and each vertex has exactly $n = 4$ adjacent vertices corresponding to the size of the neighborhood.

In the simple version of TS, considering only a static neighborhood structure, if our purpose is to prevent cycling, then even a memory that records all solutions of the trajectory (in complete form) risk the possibility of becoming blocked, in the sense that the method will reach a point where no move can be accepted other than the one that will revisit a previous solution. Consequently, cycling will be necessary to allow the search to explore new solutions. To illustrate this case we return to the 0-1 IP example given in figure 1. In Table 1, we indicate an execution of our simple TS algorithm, starting from the initial

Table 1. Example of blockage.

Iteration	Solution	Iteration	Solution	
x^1	0 0 0 0	x^8	0 0 0 1	
x^2	1 0 0 0	x^9	0 0 1 1	
x^3	1 1 0 0	x^{10}	0 1 1 1	
x^4	1 1 1 0	x^{11}	0 1 1 0	
x^5	1 1 1 1	x^{12}	0 1 0 0	
x^6	1 1 0 1	x^{13}	0 1 0 1	Blockage
x^7	1 0 0 1		x^6 x^8 x^{10} x^{12}	

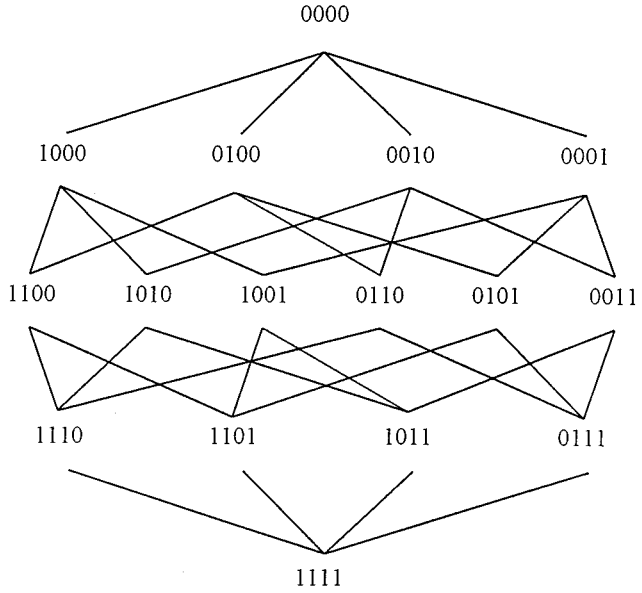

Figure 1. Graph induced by a Hamiltonian neighborhood for 0-1 IP.

solution $x^1 = 0000$, where at each iteration only one elementary drop or add move is used to generate a next solution. At iteration 2, the first component of x^1 is changed into its complement, yielding $x^2 = 1000$, and so on. At iteration 13 the search process becomes blocked if we are unwilling either to change the neighborhood's structure or to duplicate a solution already visited. For example, complementing the first component gives again the solution x^6. Complementing the second yields the solutions x^8, and so on. (See the last line of Table 1).

3. Convergent tabu search algorithm

In this section a Convergent Tabu Search (CTS) algorithm is given based on a suggestion of Glover for remedying the situation where cycling is unavoidable.

The CTS algorithm operates as follows. At each iteration, a new solution located in the neighborhood of the current solution is examined if and only if the whole neighborhood has not been explored. In the case where all the neighboring solutions have been visited, the search proceeds to the earliest neighboring solution visited in the past (see *Step 1*.) To facilitate pinpointing this solution, a label is attached to each solution, that indicates the iteration at which it was most recently visited. So, the index i of x^i, designates this iteration. When a solution is revisited, its label is changed to identify the current iteration. A schematic description of the CTS algorithm is given below.

Convergent Tabu Search algorithm

Step 0 : Choose an initial solution $x \in X$; set $k = 1$; $x^k = x$;

Step 1 : If all neighbor solutions of x^k are already visited i.e. $N(x^k) \subseteq S^k$ then Select the earliest solution x^{i^*} examined, i.e.

$$x^{k+1} = x^{i^*} \qquad \text{with} \quad i^* = \min\{i : x^i \in N(x^k)\}$$

Update the last time the solution x^{i^*} has been visited

$$S^{k+1} = S^k - \{x^{i^*}\} + \{x^{k+1}\}$$

Step 2 : Otherwise,
Select an unvisited solution $x^{k+1} \in (N(x^k) - S^k)$ with respect to certain criterion;
Update the trajectory $S^{k+1} = S^k + \{x^{k+1}\}$

Step 3 : If the stopping criterion is satisfied, the method stops.

Step 4 : Otherwise, set $k = k + 1$ and go to *Step 1*.

From a graph theory perspective, a simple interpretation of selecting the earliest solution visited in the past, when cycling is unavoidable, is to choose the vertex that forms the cycle of maximal length (in terms of arcs on the trajectory).

Table 2. Trace of CTS algorithm.

Iteration	Solution					State
x^{13}	0	1	0	1	$N(x^{13}) \subseteq S^{13}$	Blockage
x^{14}	1	1	0	1	$x^{14} = x^6$	Repetition
x^{15}	1	1	0	0	$x^{15} = x^3$	Repetition
x^{16}	1	0	0	0	$x^{16} = x^2$	Repetition
x^{17}	1	0	1	0		New solution
x^{18}	0	0	1	0		New solution
x^{19}	0	0	0	0	$x^{19} = x^1$	Repetition
x^{20}	0	0	0	1	$x^{20} = x^8$	Repetition
x^{21}	1	0	0	1	$x^{21} = x^7$	Repetition
x^{22}	1	0	1	1	$\lvert S^{22}\rvert = 2^4$	Stop

Table 3. Execution without repetition.

Iteration	Solution				Iteration	Solution			
x^1	1	0	1	1	x^9	1	1	0	1
x^2	1	0	1	0	x^{10}	1	0	0	1
x^3	0	0	1	0	x^{11}	0	0	0	1
x^4	0	0	0	0	x^{12}	0	0	1	1
x^5	1	0	0	0	x^{13}	0	1	1	1
x^6	1	1	0	0	x^{14}	0	1	1	0
x^7	1	1	1	0	x^{15}	0	1	0	0
x^8	1	1	1	1	x^{16}	0	1	0	1

Consider again the instance of the 0-1 IP problem given above and apply the CTS algorithm to it. In Table 2 the solution visited after the blockage stage are reported. The first line of the Table 2 corresponds to the blockage stage in Table 1. We can see that solution x^6 becomes solutions x^{14}, because of the update provided in *Step 1*. Shaded solutions represent the solution that are repeated. Solutions in light color are the new solutions explored.

The choice of a suitable order for selecting next solutions in a given neighborhood structure is essential to reduce the number of repetitions. Applying the CTS algorithm to the same example, simply changing the order of selection, we obtain no repetition, as disclosed below in Table 3.

For constrained problems, it will also be judicious to consider infeasible solutions in order to reduce the number of repetitions. Therefore, infeasible solutions will be penalized rather being strictly discarded from consideration.

4. Finiteness of CTS algorithm

In this section, we demonstrate that under certain hypotheses the CTS algorithm guarantees the search space will be explored completely, thus assuring the global optimum will be found.

To prove convergence, we assume that the neighborhood structure N satisfies the following hypotheses:

A1. The neighborhood relation is symmetric, i.e.

$$x \in N(x') \Leftrightarrow x' \in N(x) \text{ for all } x, x' \in E.$$

A2. The graph G_N induced by the neighborhood structure N is strongly connected i.e. for every pair of solutions $x, x' \in E$, there exists a path from x to x'.

These conditions are not restrictive, from the standpoint that several types of neighborhood structures frequently used in different techniques for solving large classes optimization

problems satisfy these assumptions. Two well-known examples are provided by the local neighborhood search approaches proposed by Lin (1965) and Lin and Kernighan (1973) for the traveling salesman problem, where a k-opt neighborhood is used. Other approaches satisfying $A1$ and $A2$ are those that incorporate the pivoting notion of the simplex method of linear programming to solve nonconvex and mixed integer linear programming problems. These approaches exploit the fact that a "good" solution may be found at or near an extreme point of the LP-relaxation feasible set (i.e. ignoring integer constraints). The pivoting move corresponds to a move from one extreme point to an adjacent one. Let

$$N_d(x) = \{x' \text{ obtained from } x \text{ by applying } d \text{ times the pivoting operation without repetition}\}.$$

Several fruitful procedures use such a type of neighborhood structure generally with $d = 1$ (see for example, Balas and Martin, 1980; Aboudi and Jörnsten, 1994; Lokketangen and Glover, 1996). These types of neighborhoods as well as the one for pure zero-one integer programming problems defined in Section 2, satisfy our assumptions.

Trivially, if the CTS algorithm terminates then it explores all solutions in space E. From now on, we assume that the CTS algorithm does not terminate. We precede the main result by a series of technical lemmas that analyze the algorithm's behavior. If the algorithm does not stop, then for a large enough iteration k the algorithm will cycle infinitely, i.e. no more new solutions will be visited. This result is stated by the following observation.

Lemma 1. *Let S^k be the set of solutions generated by the CTS algorithm upon the iteration k. The sequence $\{S^k\}$ $(k \geq 1)$, is nondecreasing and converges to a limit S^*.*

Proof: Obviously, by construction, it follows that the sequence $\{S^k\}$ $(k \geq 1)$ is nondecreasing since we have $S^k \subseteq S^{k+1}$. Since it is bounded above by the set E, the sequence must tend to a limit S^*. □

The next lemma gives a necessary condition so that the CTS algorithm will visit all solutions of E. This stopping criterion stipulates that it is not necessary to continue the exploration if all neighbor solutions of the already visited solutions have also been inspected.

Lemma 2. *Let N be a neighborhood structure such that assumptions A1 and A2 are satisfied. If there exits an iteration k^* such that*

$$\text{For all } x \in S^{k^*}, \text{ we have } N(x) \subseteq S^{k^*}$$

Then $S^{k^*} = E$, i.e. the CTS algorithm stops.

Proof: We shall use a proof by contradiction to show that $S^{k^*} = E$. Suppose that $S^{k^*} \neq E$. According to our assumption, there is at least one path which connects a solution in $E - S^{k^*}$, namely y, to a solution in S^{k^*}, namely x. Let the path from y to x be $P(y, x) = (y = y^1, y^2, \ldots, y^p = x)$ with $y^{i+1} \in N(y^i)$ for $i = 1, \ldots, p-1$. Let us consider the first solution

y^{i^*} along the path $P(x, y)$ such that $y^{i^*} \in E - S^{k^*}$ and $y^{i^*+1} \in S^{k^*}$ with $y^{i^*+1} \in N(y^{i^*})$. However, the symmetric property of the neighborhood structure assure that $y^{i^*} \in N(y^{i^*+1})$. But, since $y^{i^*+1} \in S^{k^*}$, we would have $N(y^{i^*+1}) \subseteq S^{k^*}$, and $y^{i^*} \in S^{k^*}$. This contradiction completes the proof. □

Next we claim that the limit S^* satisfies:

$$\text{For all } x \in S^*, \text{ we have } N(x) \subseteq S^* \tag{1}$$

In order to prove (1), subdivide the set S^k of solutions visited up to the iteration k, into two disjoint subsets A^k and B^k, defined as follows:

$$A^k = \{x^i \in S^k : i < h_k\} \text{ and}$$
$$B^k = S^k - A^k = \{x^i \in S^k : h_k \leq i \leq k\} \text{ where}$$
$$h_k = \min\{h : N(x^i) \subseteq S^k \text{ for all } i \leq h \leq k\}$$

If there exists no index h such that $N(x^p) \subseteq S^k$ for all $p \leq h$, let $h_k = 0$. It is clear that h_k is well defined since for a large k, the algorithm executes *Step 1* an infinite number of times to generate solutions. First, sequences $\{A^k\}$ and $\{B^k\}$ possess the following property.

Lemma 3. *The sequences $\{A^k\}$ and $\{B^k\}$ $(k \geq 1)$ are monotone and converge to limits A^* and B^* respectively and two limit sets A^* and B^* constitute a partition of the set S^*.*

Proof: First, by construction, it follows that $\{A^k\}(k \geq 1)$ is a nonincreasing sequence and the sequence $\{B^k\}(k \geq 1)$ is nondecreasing since we have $A^{k+1} \subseteq A^k$ and $B^k \subseteq B^{k+1}$ for all $k \geq 1$. Observe also that the sequence $\{A^k\}$ is bounded below by empty set and $\{B^k\}$ is bounded above by E, thus their limits exist.

Second, clearly for iteration k, the sets $\{A^k\}$ and $\{B^k\}$ constitute a partition of the set S^k, i.e. we have $A^k \cap B^k = \emptyset$ and$A^k \cup B^k = S^k$, so passing to the limits the result is obtained. □

The next result gives information related to the limit sets A^* and B^* which is useful in finding a contradiction if the CTS algorithm does not stop.

Lemma 4. *Let A^* and B^* be limits of the sequences $\{A^k\}$ and $\{B^k\}$ $(k \geq 1)$ respectively. Then*:

i) *For all $x \in B^*$, $N(x) \cap A^* = \emptyset$* (2)
ii) *For all $x \in A^*$, $N(x) \cap B^* = \emptyset$* (3)
iii) *For all $x \in B^*$, $N(x) \subseteq B^*$* (4)
iv) $A^* = \emptyset$ (5)

Proof: i) Suppose that there exists a solution $x^k \in B^*$ such that $N(x^k) \cap A^* \neq \emptyset$. since x^k is one element of B^*, we have $h_k \leq k$ and $N(x^k) \subseteq S^k$. Therefore, at iteration $k+1$, we have:

$$x^{k+1} = x^{i^*} \text{with } i^* = \min\{i : x^i \in N(x^k)\}$$

Since $N(x^k) \cap A^* \neq \emptyset$ and $A^* \subseteq S^k$, i^* can be expressed as follows:

$$i^* = \min\{i : x^i \in N(x^k) \cap A^*\}$$

which implies that $x^{k+1} \in A^* \cap B^*$. Thus we obtain a contradiction with Lemma 3, proving that $A^* \cap B^* = \emptyset$. The proof of i) is complete.

ii) Assume that the result is false, then there exists a solution $x^k \in A^*$, such that $N(x^k) \cap B^* \neq \emptyset$. Let $x^i \in N(x^k) \cap B^*$, hence x^i is belonging to B^*, form the above result (2) of this lemma, we have $N(x^i) \cap A^* = \emptyset$. Moreover, by assumption we have $x^k \in A^*$ and $x^k \in N(x^i)$ which implies that $x^k \in N(x^i) \cap A^*$. Thus, we obtain a contradiction that completes the proof.

iii) Let $x^k \in B^*$, obviously by the definition of sets B^k, it follows that $N(x^k) \subseteq S^k$ with $k > h_k$. On the other hand, according to Lemma 3, we have $S^k \subseteq S^* = A^* \cup B^*$ which implies that $N(x^k) \subseteq A^* \cup B^*$. Consequently $N(x^k) \subseteq B^*$ since $N(x^k) \cap A^* = \emptyset$, using the property (2) of this lemma. This completes the proof of iii).

iv) We shall use proof by contradiction to show that $A^* = \emptyset$. Let us consider $x^{i^*} \in A^*$ such that $i^* = \max\{i : x^i \in A^*\}$, which implies that $i^* < h^* = h_k$. By construction, there exists a path $P(x^{i^*}, x^{h^*})$ that connects the solution x^{i^*} and the solution x^{h^*}. The immediate successor of x^{h^*} along the path $P(x^{i^*}, x^{h^*})$ is a neighborhood solution of x^{i^*}, which belongs to B^*. This leads to the contradiction $N(x^{i^*}) \cap B^* \neq \emptyset$ with the result ii), so the proof is completed. □

The principal result of convergence is

Theorem. *Suppose that assumptions A1 and A2 hold and E is finite. Then the CTS algorithm terminates after exploring all solutions in E.*

Proof: From Lemma 3, (4) and (5), it follows that(1) is satisfied, by Lemma 1, the sequence S^k has a limit S^*, this means that there exists an iteration k^* such that $S^* = S^k$ for all $k > k^*$. Now using Lemma 2, we obtain the result claimed. □

5. Conclusion

We have shown that the conjecture of Glover on Tabu Search is true, by proving the finite termination of the CTS algorithm proposed in Section 3. This demonstrates that simple Tabu Search (using neither intensification nor diversification) can be considered as an exact method for combinatorial problems where the domain is discrete and finite. It supposes that the structure of the neighborhood allows the whole search space to be explored. This result holds even if the domain is constrained. However, in this case, the structure of the neighborhood must permit a complete exploration of the set of feasible solutions even if it means passing through the infeasible domain.

In applying TS, it is possible that all available moves are tabu and in the case of the CTS algorithm this condition is an indication of cycling. Therefore, an aspiration-by-default criterion is used that allows the "least tabu" move to be chosen (i.e. the one whose tabu

status is oldest, and hence closest to the point of being free to be selected). Such a choice is often employed in various applications of TS. The theoretical result provided in this work justifies that this is a suitable choice to avoid cycling that repeats indefinitely.

In our approach, neither the objective function nor the evaluations of the moves have been taken into account. However, the results hold even with sophisticated versions of TS provided that the trajectory followed stays connected at each iteration.

The current paper does not address issues of implementation, but methods do exist such as REM of C-Sequence (see, for example, Glover, 1990; Dammeyer et al., 1991) which can be adapted to handle the memory and labeling functions of CTS algorithm more efficiently. Such issues will be considered in future work.

References

Aboudi, R. and K. Jörnsten. (1994). "Tabu Search for General Zero-One Integer Programs Using the Pivot and Complement Heuristic." *ORSA Journal of Computing* 6, 82–93.

Balas, E. and C.H. Martin. (1980). "Pivot and Complement—A Heuristic for 0-1 Programming." *Management Science Research* 26, 86–96.

Dammeyer, F., P. Forst, and S. Voß. (1991). "On the Cancellation Sequence Method of Tabu Search." *ORSA Journal on Computing* 3, 262–265.

Dammeyer, F. and S. Voß. (1993). "Dynamic Tabu List Management using the Reverse Elimination Method." *Annals of Operations Research* 41, 31–46.

Feo, T.A., M.G.C. Resende, and S.H. Smith. (1994). "A Greedy Randomized Adaptive Search for Maximum Independent Set." *Operations Research* 42, 860.

Garey, M.R. and D.S. Johnson. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Fransisco: W.H. Freeman and Company.

Glover, F. (1989). "Tabu Search, Part 1." *ORSA Journal on Computing* 1, 190–206.

Glover, F. (1990). "Tabu Search, Part 2." *ORSA Journal on Computing* 2, 4–32.

Glover, F. and M. Laguna. (1997). *Tabu Search*. Dordrecht: Kluwer Academic Publishers.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Hanafi, S., and A. Fréville. (1996). "An Efficient Tabu Search Approach for the 0-1 Multidimensional Knapsack Problem." In Fred Glover (ed.), Special Issue on Tabu Search, to appear in *European Journal of Operational Research*.

Hanafi, S., A. Fréville, and A. El Abdellaoui. (1995). "Comparison of Heuristics for the 0-1 Multidimensional Kanapsack Problem." In I.H. Osman and J.P. Kelly (eds.), *Metaheuristics: Theory and Applications*. Dordrecht: Kluwer, pp. 449–466.

Hübscher, R. and F. Glover. (1994). "Applying Tabu Search with Influential Diversification to Multiprocessor Scheduling." *Computer and Operation Research* 13, 877–884.

Kirkpatrick, S., C.D. Gelatt, and M.P. Vechi. (1983). "Optimization by Simulated Annealing." *Science* 220, 671–680.

Lin, S. and B.W. Kernighan. (1973). "An Effective Heuristic Algorithm for the Traveling Salesman Problem." *Operations Research* 21, 443–452.

Lokketangen, A. and F. Glover. (1996). "Solving Zero-One Mixed Integer Programming Problems Using Tabu Search." Working paper, University of Colorado, USA.

Mladenovic, N. and P. Hansen. (1996). "Variable Neighborhood Search." *Les Cahiers du GERAD*, G-96-49.

Osman, I.H. and G. Laporte. (1996). "Meta-Heuristics: A Biography." *Annals Opnl. Res.* 63, 513–628.

Reeves, C. (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Oxford: Blackwell.