ORSA Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Tabu Search—Part I

Fred Glover,

To cite this article:
Fred Glover, (1989) Tabu Search—Part I. ORSA Journal on Computing 1(3):190-206. http://dx.doi.org/10.1287/ijoc.1.3.190

ORSA Journal on Computing
Vol. 1, No. 3, Summer 1989

0899-1499/89/0103-0190 $01.25

Tabu Search—Part I

FRED GLOVER *U.S. West Chair in Systems Science, Center for Applied Artificial Intelligence, Graduate School of Business, Box 419, University of Colorado, Boulder, CO 80309-0419, BITNET: fglover@colorado*

(Received: June 1988; final revision received November 1988; accepted February 1989)

This paper presents the fundamental principles underlying tabu search as a strategy for combinatorial optimization problems. Tabu search has achieved impressive practical successes in applications ranging from scheduling and computer channel balancing to cluster analysis and space planning, and more recently has demonstrated its value in treating classical problems such as the traveling salesman and graph coloring problems. Nevertheless, the approach is still in its infancy, and a good deal remains to be discovered about its most effective forms of implementation and about the range of problems for which it is best suited. This paper undertakes to present the major ideas and findings to date, and to indicate challenges for future research. Part I of this study indicates the basic principles, ranging from the short-term memory process at the core of the search to the intermediate and long term memory processes for intensifying and diversifying the search. Included are illustrative data structures for implementing the tabu conditions (and associated aspiration criteria) that underlie these processes. Part I concludes with a discussion of probabilistic tabu search and a summary of computational experience for a variety of applications. Part II of this study (to appear in a subsequent issue) examines more advanced considerations, applying the basic ideas to special settings and outlining a dynamic move structure to insure finiteness. Part II also describes tabu search methods for solving mixed integer programming problems and gives a brief summary of additional practical experience, including the use of tabu search to guide other types of processes, such as those of neural networks.

Tabu search is a strategy for solving combinatorial optimization problems whose applications range from graph theory and matroid settings to general pure and mixed integer programming problems. It is an adaptive procedure with the ability to make use of many other methods, such as linear programming algorithms and specialized heuristics, which it directs to overcome the limitations of local optimality.

Tabu search has its origins in combinatorial procedures applied to nonlinear covering problems in the late 1970s,[2] and subsequently applied to a diverse collection of problems ranging from scheduling and computer channel balancing to cluster analysis and space planning.[3,4,6,7] Latest research and computational comparisons involving traveling salesman, graph coloring, job shop flow sequencing, integrated circuit design and time tabling problems have likewise disclosed the ability of tabu search to obtain high quality solutions with modest computational effort, generally dominating alternative methods tested.[1,12,13,18,22] A recent independent development of several of its ideas[10] also has been applied successfully to maximum satisfiability problems.[11] Such applications, for problems ranging in size from hundreds to millions of variables, are elaborated in Section 10.

1. Background and Notation

To describe the workings of tabu search, we represent a combinatorial optimization problem in the following form.

$$\text{(P)} \qquad \text{Minimize} \quad c(x)\colon x \in X \quad \text{in } R_n.$$

The objective function $c(x)$ may be linear or nonlinear, and the condition $x \in X$ is assumed to constrain specified components of x to discrete values. In some settings (P) may represent a modified form of some original problem, as where X is a superset of the vectors that normally qualify as feasible, and $c(x)$ is a penalty function, designed to assure that optimal solutions to (P) likewise are optimal for the problem from which it derived.

A wide range of procedures, heuristic and optimal, for solving various problems capable of being written in the form (P) can be characterized conveniently by reference to sequences of *moves* that lead from one *trial solution* (selected $x \in X$) to another. We will define a move s to consist of a mapping defined on a subset $X(s)$ of X:

$$s\colon X(s) \to X.$$

Subject classification: Programming: integer, heuristic.

Associated with $x \in X$ is the set $S(x)$ which consists of those moves $s \in S$ that can be applied to x; i.e., $S(x) = \{s \in S: x \in X(s)\}$ (and we may thus also write $X(s) = \{x \in X: s \in S(x)\}$). The set $S(x)$ can be viewed as a "neighborhood function" (see, e.g. [19, 20]).

Common Examples of Moves

As a prelude to the main concerns of this paper, it is useful to elaborate briefly the nature of moves that are relevant to the contexts we will be examining. In many applications of interest, if x' and x'' are distinct elements of the set $X(s)$, then $s(x') \neq s(x'')$; i.e., it is convenient to classify moves so that distinct trial solutions are transformed into new trial solutions that are also distinct. A simple example is the set of moves between adjacent vertices of the zero-one unit hypercube, as illustrated by the move $s(x) = x + e_j$, where e_j is the unit n-vector with a 1 in position j. (If X denotes the set of all such vertices, then the appropriate form of $X(s)$ in this case is given by $X(s) = \{x \in X: x_j = 0\}$.) Another example is the common move in graph theory and matroid settings that consists of an augmentation step defined relative to an alternating path. In a nonlinear optimization context, a standard move is $s(x) = x + ud$, where d is a specified direction vector, such as a generalized gradient, and u is a scalar step size. This latter type of move can map distinct trial solutions into the same vector unless its classification is made to depend on the identity of both u and d.

An important type of move from mixed integer programming is the familiar composite move that increments or decrements the value of one of the integer variables, and then determines the values of the continuous variables by solving the resulting linear program. Notationally, for this case we write $x = (x_I, x_c)$, where x_I and x_c respectively are the vectors of integer and continuous variables, and let $X = \{x: A_I x_I + A_c x_c = b, x \geq 0$ and x_I integer$\}$. The composite move for mixed integer programming may then be expressed as a mapping from x' to x'' given by $s(x_I', x_c') = (x_I'', x_c'')$, where $x_I'' = s_I(x_I')$ and x_c'' is an optimal solution to

$$\text{Minimize} \quad c(x_I'', x_c): (x_I'', x_c) \in X.$$

In particular, $x_I'' = s_I(x_I')$ represents a component mapping of the form $x_I'' = x_I' + e_j$ or $x_I'' = x_I' - e_j$; and the condition $(x_I'', x_c) \in X$ identifies x_c as an element of the set $\{x_c: A_c x_c = b - A_I x_I'', x_c \geq 0\}$. We will refer again to this type of move in Part II.

2. A Simple Form of Tabu Search

We first present tabu search in a simple form that discloses two of its key elements: that of constraining the search by classifying certain of its moves as forbidden (i.e., tabu), and that of freeing the search by a short term memory function that provides "strategic forgetting." Later sections introduce elaborations of the basic procedure, exposing the dual relationship between tabu restrictions and aspiration criteria as means for constraining and guiding the search process, and introducing the use of intermediate and long term memory functions that operate in counterpoint to the function of short term memory.

It is convenient to start by reference to the familiar class of approaches known as hill climbing heuristics, which undertake to progress unidirectionally from their starting point to a local optimum. (In the present minimization context, the hill is "inverted" so that the direction of climbing is downward.)

HILL CLIMBING HEURISTIC FOR (P)

1. Select an initial $x \in X$.
2. Select some $s \in S(x)$ such that
$$c(s(x)) < c(x).$$
If no such s exists, x is a local optimum and the method stops. Otherwise,
3. Let $x := s(x)$
and return to Step 2.

While conceptually simple in general outline, such hill climbing heuristics can have useful (and sometimes subtle) characteristics, and in fact encompass a variety of elegant mathematical algorithms. If X is the set of feasible extreme points for a linear program and $S(x)$ is the set of moves that lead from x to an adjacent extreme point, then the simplex method (employing implicit or explicit perturbation to avoid degeneracy) is such a hill climbing procedure. Similarly, if X is the dual feasible region for a linear program, and $S(x)$ is the set of moves that determines an improving gradient vector by mapping x into the point $(1, 1, \ldots, 1)$, and then chooses a step size that selects a new point about 0.9 (preferably) of the distance from x to the dual feasible boundary, then the Karmarkar dual affine linear programming method likewise is such a hill climbing procedure. Combinatorial search strategies are well advised, therefore, to heed the potentialities that reside within such methods.

The chief limitation of a hill climbing procedure in a combinatorial problem setting is that the local optimum obtained at its stopping point, when no improving moves are possible, may not be a global optimum. Tabu search guides such a heuristic to continue exploration without becoming confounded by an absence of improving moves, and without falling back into a local optimum from which it previously emerged. From its ability to incorporate and guide another

procedure, in amended form as a subroutine, tabu search may be viewed as a metastrategy for combinatorial problem solving.

The operation of the procedure in simplified form may be described as follows. A subset T of S is created whose elements are called *tabu moves*. The elements of T are determined by a non-Markovian function that utilizes historical information from the search process, extending up to t iterations in the past, where t can be fixed or variable depending on the application or stage of search. Membership in T is by means of an itemized list or by reference to a set of *tabu conditions* (e.g., linear inequalities or logical relationships) expressed indirectly in terms of a current trial solution x; for example, by letting T take the form $T(x) = \{s \in S: s(x)$ violates the tabu conditions$\}$. For an appropriately determined T and an evaluator function denoted by OPTIMUM, subsequently identified in greater detail, the procedure may be described as follows.

SIMPLE TABU SEARCH

1. Select an initial $x \in X$ and let $x^* := x$. Set the iteration counter $k = 0$ and begin with T empty.
2. If $S(x) - T$ is empty, go to Step 4.
   Otherwise, set $k := k + 1$ and select $s_k \in S(x) - T$ such that $s_k(x) = \text{OPTIMUM}(s(x): s \in S(x) - T)$.
3. Let $x := s_k(x)$. If $c(x) < c(x^*)$, where x^* denotes the best solution currently found, let $x^* := x$.
4. If a chosen number of iterations has elapsed either in total or since x^* was last improved, or if $S(x) - T = \varnothing$ upon reaching this step directly from Step 2, stop.
   Otherwise, update T (as subsequently identified) and return to Step 2.

Three aspects of this version deserve emphasis: (1) the use of T provides the "constrained search" element of the approach, and hence the solutions generated depend critically on the composition of T and the way it is updated at Step 4; (2) the method makes no reference to the condition of local optimality, except implicitly where a local optimum improves on the best solution previously found; (3) a "best" move (rather than an improving move) is chosen at each step, employing the criteria embedded in the OPTIMUM function.

After a preliminary discussion of these issues, an example will be provided to make the discussion concrete and to indicate data structures convenient for implementing the procedure.

First we consider some of the forms of OPTIMUM and the tabu set T. A natural choice for the OPTIMUM function is given by selecting $s_k(x)$ so that

$$c(s_k(x)) = \text{Minimum}(c(s(x)): s \in S(x) - T).$$

In cases where exclusion from T can be expressed as a requirement to satisfy a set of inequality constraints (such as bounds on variables), and the set $S(x)$ can be similarly characterized, the solution $s(x)$ obtained from defining OPTIMUM in this fashion may represent the outcome of solving an auxiliary optimization problem. (Such a possibility is directly relevant to integer programming applications using a linear programming method as a heuristic subroutine, as elaborated in Part II.)

By the preceding form of OPTIMUM, each execution of Step 2 moves from the current x to an $s(x)$ that yields the greatest improvement—or, lacking the possibility of improvement, the least disimprovement—in the objective $c(x)$, subject to the restriction that only non-tabu moves are allowed. In cases where the set $S(x) - T$ may be large, and processed by itemization rather than auxiliary solution, it is appropriate for the OPTIMUM function to be based on a strategy for sampling this region, shrinking $S(x) - T$ for the purpose of identifying the minimum $c(s(x))$. Picking the first s such that $c(s(x)) < c(x)$, if one exists, provides one instance of such a sampling strategy, but tabu search generally proceeds more aggressively. This approach contrasts with methods that undertake to progress slowly to a local optimum, such as the simulated annealing methods based on Metropolis (or "cooling") strategies, which rely on the premise that a slow descent, appropriately regulated, will make the local optimum more nearly a global one. (There nevertheless may be advantages for integrating tabu search with such methods, as subsequently noted.)

The rationale underlying the aggressive orientation of tabu search derives from two considerations. First is that many optimization problems can be solved optimally by making a "best available move" at each step, a phenomenon that goes beyond the well known results for greedy algorithms. Minimum cost network flow problems and weighted matroid intersection problems, for example, can be solved by moves based on successively identified best (shortest) paths. Choosing moves other than best ones can lead to inferior solutions for these methods, while other methods for these problems, which do not require best moves at each step, are not injured (and are often accelerated) by choices with the highest evaluations.

The second consideration underlying the aggressive orientation of tabu search stems from the procedural organization by which local optimality does not present a barrier, and hence offers less compelling reasons for delaying the approach to it. Rather than spend proportionately more time in regions whose solutions

are proportionately less attractive, tabu search undertakes to devote the larger share of its effort to exploring regions where solutions are good. (In situations where slow descent strategies appear valuable,[14, 17] they may be used to provide an initial point of departure.)

The simple rule that selects the minimum $c(s(x))$, subject to the tabu restrictions, has in fact proved successful in a variety of applications. When the minimum is expensive to compute, an approximation to it may be chosen as an alternative to reducing the sample region.

A similarly straightforward but likewise effective form for the set T is given by

$$T = \{s^{-1}: s = s_h \text{ for } h > k - t\}$$

where k is the iteration index and s^{-1} is the inverse of the move s; i.e., $s^{-1}(s(x)) = x$. In words, T is the set of those moves that would "reverse" (or "undo") one of the moves made in the t most recent iterations of the search process. Thus, the operation of updating such a T at Step 3 of the tabu search procedure in effect consists of setting $T := T - s_{k-t}^{-1} + s_k^{-1}$. (The minus and plus signs indicate the operations of deleting and adding elements to a set.) By convention, when $k \leqslant t$ the reference to s_{k-t}^{-1} is disregarded.

The indicated form for the tabu set is based on the assumption that the likelihood of cycling, i.e., of following a sequence of moves that leads back to a solution visited in the past, is inversely related to the distance of the current trial solution x from that previous solution. If distance is measured in terms of the number of moves taken (hence the number of iterations made) since the previous solution was visited, subject to the stipulation that no intervening move is allowed to reverse one of its predecessors, then T is designed to counter cycling according to the stipulated assumption.

Within the structure of certain choice rules and tabu conditions, the goal more generally is to avoid returning to a previous *solution state*, e.g., to a previously visited solution where the best available (non-tabu) move for leaving that solution is the same as before. In this case, T more appropriately takes the form $T = T_1 \cup T_2$, where $T_1 = \{s_h^{-1}: h > k - t_1\}$ and $T_2 = \{s_h: h > k - t_2\}$.

Thus, conjecturally, by preventing the choice of a move that represents the reverse of any taken during a sequence of t iterations, the procedure moves progressively away from all solution states of the previous t iterations (in a sense determined by the nature of the moves in S), and for t appropriately large, the likelihood of return effectively vanishes. It does not follow, however, that the objective of tabu search is to choose t large. From a competing perspective, the smaller the value of t, the greater the latitude of choice the method has to drive toward solutions that the function OPTIMUM finds preferable. An empirical discovery from the application of tabu search methods is that t has a highly stable range of values that both prevent cycling and lead to remarkably good solutions. A challenge to research would be the development of theory to explain this phenomenon.

Indeed, in practice, T rarely takes the form previously indicated, for two reasons. First, in some settings, when the move s is chosen from $S(x)$, the tabu condition that prevents s^{-1} from being selected also must prevent a larger set of moves, dominated by s^{-1}, from being selected. (Issues concerning "dominated" and "deficient" moves are treated in Section 6.) Second, for considerations of memory conservation and ease of processing, it is often desirable to record less than the full range of attributes required to characterize a move, or the solution to which it is applied, and hence a partial range of attributes is recorded instead (which potentially may be shared by other moves or solutions). Under these circumstances, a tabu list T does not consist simply of the moves s_h^{-1} for $h > k - t$, but of collections C_h of moves, where each C_h defines its membership by certain attributes, embodied in its tabu conditions, which cause it to contain s_h^{-1} and other moves that likewise satisfy these conditions. Thus, more generally, T characteristically takes the form

$$T = \cup\, C_h: h > k - t \quad (\text{where } s_h^{-1} \in C_h).$$

The sets C_h also may be allowed to contain elements other than s_h^{-1} for strategic regions related to the prevention of cycling, as will be indicated in Part II of this study in the context of mixed integer programming.

One additional important aspect concerning the management of T deserves mention. In the case where $S(x) - T$ is empty in Step 2 of the Simple Tabu Search procedure, the update of T at Step 4 departs from the update of T implicit in the preceding discussion (which in effect drops the tabu restriction recorded t iterations ago) by creating a prioritization equivalent to basing the choice on dropping the smallest number of elements from T, in the sequence from oldest to youngest, that will enable some move(s) to regain a non-tabu status. Such a prioritization can be shown to lead to an optimum in a finite number of steps in simple settings if t is allowed to grow with the iteration k, although this priority scheme potentially allows superfluous moves that can be avoided by refinement. (An example where finiteness is assured occurs in the case where giving a tabu label to a move reversal corresponds to giving a tabu label to the solution reached by this reversal—provided the number of solutions is finite and there exists a sequence of moves leading from the starting solution to an optimal solution.) In practice, for applications examined thus far, neither such refinements in priorities nor a growing t value have proven necessary.

An Example

Consider a problem of creating an optimal partition of a set of elements E into sets E_i, $i \in N = \{1, \ldots, n\}$. Each element e in E has a weight $w(e)$, and each set E_i has a target weight W_i. Define $w(E_i)$ to be the weight of the elements of E assigned to E_i, that is, $w(E_i) = \sum (w(e): e \in E_i)$, where $w(E_i) = 0$ if E_i is empty. Then the goal is to minimize the sum of absolute deviations of the weights $w(E_i)$ from the target weights W_i:

$$\text{Minimize} \quad \sum (|\, w(E_i) - W_i \,|: i \in N)$$

A reasonable set of moves to embed within tabu search for this problem arises by starting with some arbitrary assignment of elements to the sets, and using exchanges that swap an element e_i currently in set E_i with an element e_j currently in set E_j. We allow the special case where one of e_i or e_j is a "null" element of zero weight, which provides a partial exchange that simply moves an element from one set to another.

The OPTIMUM function can take the form of prescribing the swap that yields the most favorable change in the minimization objective (whether an increase or decrease), restricting attention to moves that do not qualify as tabu. At a corresponding level of simplicity, the tabu set T is given the role of preventing the reversal of moves made during the most recent t iterations.

To accomplish the prevention of move reversals, and implicitly to define T, we employ two arrays, TABULIST and TABUSTATE. (Analogous arrays are useful for a wide range of applications of tabu search.) TABULIST(p), $p = 1, \ldots, t$ records the attributes selected to provide a shorthand, or coding, to embody the tabu restrictions applicable to associated moves. Such attributes can be expressed at varying levels of detail. At an extreme, the entire composition of the sets $E_1, \ldots, E_n$ could be recorded before and after a move, and this record then used to check the corresponding composition for a proposed future move to see if it qualifies as tabu. A simpler set of attributes appropriate to our example consists of the two ordered pairs $(i, w(e_i))$ and $(j, w(e_j))$. These suffice to identify the fact that the associated swap consists of transferring an element of weight $w(e_i)$ from set E_i to set E_j, and an element of weight $w(e_j)$ from set E_j to E_i.

Note that the precise elements e_i and e_j are not identified as part of these attributes, only their weights. Such a reduced level of specificity can have strategic value. In particular, preventing the reverse swap of the weights is more appropriate in the present context than simply preventing the reverse swap of the elements. Moreover, a sensible implementation of tabu search will additionally forbid a move such that $w(e_i) = w(e_j)$. These observations represent instances of broader principles involving **equivalent** and **deficient** moves as discussed in a later section.

The attributes $(i, w(e_i))$ and $(j, w(e_j))$ are used in this example to code the tabu restriction that forbids a move if it adds either an element of weight $w(e_i)$ to set E_i or an element of weight $w(e_j)$ to set E_j. It is important to observe that the same set of attributes can be used to code other tabu restrictions. For instance, $(i, w(e_i))$ and $(j, w(e_j))$ could instead be used to forbid moves that drop an element of weight $w(e_j)$ from set E_i or an element of weight $w(e_i)$ from set E_j. (By contrast, forbidding a move that drops an element of weight $w(e_i)$ from set E_i or of weight $w(e_j)$ from set E_j would block a move repetition rather than a move reversal.) These same attributes could also be used to prevent moves that both add and drop the indicated elements to create a move reversal. This latter type of tabu condition is less restrictive (i.e., it forbids a smaller collection of moves) than those previously specified.

The TABUSTATE array has the role of making the tabu restrictions easy to implement. For our example, suppose that the elements to be partitioned have r distinct weights, w_q, $q = 1, \ldots, r$. Then we let TABUSTATE take the form of an $n \times r$ matrix TABUSTATE(i, q), where TABUSTATE(i, q) has a positive value if an element of weight w_q is forbidden to be added to set E_i, and TABUSTATE(i, q) has a zero value otherwise. For reasons subsequently explained, a positive TABUSTATE(i, q) value names the most recent position p on TABULIST where the attribute (i, q) is recorded (with the purpose of prohibiting an element of weight w_q from being added to set E_i).

The full implementation of tabu search for our example problem then becomes as follows. TABULIST(p), which in our example stores two ordered pairs for each value of $p = 1, \ldots, t$, is treated as a circular list. To do this, the value p is updated by $p := p + 1$ when the iteration counter is updated by $k := k + 1$, except that p is reset to 1 whenever its value would be increased from t to $t + 1$. Attributes recorded at position p of TABULIST automatically erase the attributes stored there previously. When this erasure occurs, the TABUSTATE array (which begins with all entries 0) is modified so that the entries (i, q) such that TABUSTATE(i, q) = p are reset to yield TABUSTATE(i, q) = 0.

The ordered pairs (i, q) that identify these entries of TABUSTATE which are slated to be reset to 0 are precisely the two ordered pairs recorded as attributes on TABULIST(p). There is one exception: it is possible that after setting TABUSTATE(i, q) = p (when (i, q) is recorded on TABULIST(p)), a later iteration may again involve dropping an element of weight w_q from set E_i, thus resetting TABUSTATE(i, q) to a new value. Thus, it suffices simply to check whether a pair (i, q) stored

on TABULIST(p) yields TABUSTATE(i, q) = p, and if not, the operation of assigning TABUSTATE(i, q) a 0 value is bypassed. A parallel step can be used to facilitate processing of aspiration levels, as subsequently discussed in Section 4.

By means of these updating rules, at each point where a swap of elements e_i and e_j is to be evaluated as a candidate for the best move of the current iteration, the array elements TABUSTATE(i, $w(e_i)$) and TABUSTATE(j, $w(e_j)$) disclose at once whether the move is tabu (according to whether at least one of the two elements is positive). This check of the TABUSTATE array to see if a move is tabu is postponed until after the first local optimum is attained, though such a conditional override of tabu status is achieved automatically with the use of aspiration criteria.

One item remains to be resolved: the updating that occurs when all available moves of the current iteration are classed as tabu (i.e., when Step 4 of the Simple Tabu Search Procedure is reached immediately after discovering $S(x) - T$ is empty in Step 2). The prioritization of tabu moves previously mentioned may be accomplished by reference to the values of the entries in the TABUSTATE array, where the prescribed move becomes the one that produces the best change in the minimization objective subject to being among those moves with highest priority. Each nonzero entry of TABUSTATE implicitly identifies an associated iteration value k, and hence by allowing entries associated with smaller k values to impart higher priorities, the effect can be achieved of choosing a best move subject to discarding only the oldest tabu restrictions.

3. Uses of Aspiration Levels

An important element of tabu search is the incorporation of an *aspiration level function* $A(s, x)$, alluded to earlier, whose value depends on a specified move s and/or vector x. We say that the aspiration level is *attained* if

$$c(s(x)) < A(s, x).$$

The role of $A(s, x)$ is to provide added flexibility to choose good moves by allowing the tabu status of a move to be overridden if the aspiration level is attained. The goal is to do this in a manner that retains the ability to avoid cycling.

To discuss a way to achieve this goal, we will refer to a move in a second (narrower) sense as a particular instance of a mapping, e.g., speaking of a move "from x to $s(x)$." Such a move, whose identity depends on x as well as s, will be called a *solution-specific move.*

There are three strategic levels for avoiding cycling, which involve preventing the solution-specific move from x to $s(x)$ if: (1) $s(x)$ has been visited before; (2) the move s has been applied to x before; (3) the move s^{-1} has been applied to $s(x)$ before.

Although (1) is the only criterion for preventing a solution-specific move that fully assures cycling will not occur, the process of checking whether the tabu status of a move can be overridden on the basis of (1) generally requires more memory and greater effort than is convenient to employ. If a tabu move is allowed to be made provided only that condition (2) fails, then it is possible to reverse a move as soon as it is made, going back to a solution just visited. Experiments have shown that tabu lists thus designed to prevent repetition rather than reversal of moves typically do not work well, for reasons that are not difficult to imagine. Condition (3), however, is compatible with the tabu list structure. By using the check for (3) to avoid cycling, and allowing a tabu move that leads from x to $s(x)$ to be made unless the solution-specific move from $s(x)$ to x had occurred earlier, the attempt to prevent a return to a previously generated solution clearly will be supported more effectively than by relying on condition (2). Adding condition (2) to condition (3) serves to strengthen the approximation of meeting condition (1).

The relevance of these observations for creating an effective aspiration level function can be clarified by a concrete example, as follows. Define $A(s, x)$ = the best (smallest) value of $c(x')$ that could be achieved by *reversing* a previous solution-specific move from some x' to some $s'(x')$ such that $c(s'(x')) = c(s(x))$.

Although the definition appears somewhat forbidding, the underlying concept is quite simple, and such an aspiration level function is easy to record and update. Suppose that possible values for $c(x)$, for different x vectors, are given by the integers $q = 1, 2, \ldots, U$, and let BEST(q) = the minimum (best) $c(x)$ value that could have been attained by reversing a previous move that produced $c(s(x)) = q$. Initially, set BEST(q) = $U + 1$. Then when a solution-specific move from x to $s(x)$ is made, update BEST(q), for $q = c(s(x))$, by the rule BEST(q) = Min(BEST(q), $c((x))$). Subsequently the aspiration level may be allowed to override a tabu move that leads from (a different) x to $s(x)$ if

$$c(s(x)) < \text{BEST}(c(x)).$$

This particular type of aspiration is an instance of checking condition (3).

The corresponding form of (2) in this setting is given by defining $A(s, x)$ = the best (smallest) value of $c(s'(x'))$ achieved for all previous solution-specific moves from some x' to some $s'(x')$, where $c(x') = c(x)$. This alternative aspiration level function can be handled by initializing BEST(q) as before, but when a solution-specific move from x to $s(x)$ is made BEST(q)

is updated for $q = c(x)$, setting

$$\text{BEST}(q) = \text{Min}(\text{BEST}(q), c(s(x)).$$

The condition for overriding a tabu move that leads from (a different) x to $s(x)$ is identical to that given before; i.e.,

$$c(s(x)) < \text{BEST}(c(x)).$$

Conditions (2) and (3) are therefore readily combined by making *both* updates of BEST(q) previously indicated, since the check to override tabu status is the same for these cases.

Other aspiration level functions that can be easily checked and updated are derived by maintaining similar records of other attributes of a move—for example, defining a value BEST(q) where q denotes the index of a variable that is incremented or decremented. Using multiple "BEST" lists increases the likelihood that a tabu move will be allowed to escape its tabu status (by the requirement that at least one of the associated inequalities holds), although entailing greater effort for checking and recording. An interesting alternative is to define $A(s, x)$ only in relation to moves that are currently tabu, which leads to the considerations of the next section.

4. Integrated Tabu Restrictions and Aspiration Level Criteria

The tabu restrictions and aspiration level criteria of tabu search play a dual role in constraining and guiding the search process. Tabu restrictions allow a move to be regarded admissible if they *do not* apply, while aspiration criteria allow a move to be regarded as admissible if they *do* apply (i.e., if they are satisfied). This complementarity of the notions underlying the tabu restrictions and aspiration criteria enables them to be integrated into a common framework.

To describe this framework, it is appropriate to introduce notation for the attributes of moves that are used to define tabu status. Specifically, let $a_p(s, x)$, $p = 1, \ldots, g$, be a set of functions that identify selected attributes of move s applied to a solution x. To illustrate, we will refer to an approach used to integrate aspiration criteria and tabu restrictions for the traveling salesman problem,[5] based on using the familiar 2-OPT moves[16] that delete two nonadjacent edges of a tour and add back the unique two edges that create a new tour. Attribute functions for this application are defined in a straightforward way by letting $a_1(s, x)$ and $a_2(s, x)$ identify the two added edges and letting $a_3(s, x)$ and $a_4(s, x)$ identify the two dropped edges (taking $g = 4$).

The degree of specificity in identifying such attributes of moves can vary. For example, $a_1(s, x)$ and $a_2(s, x)$ can refer to the added edges without concern for any particular ordering or can differentiate the edges more precisely—e.g., in the approach of [5], $a_1(s, x)$ identifies the longer of the two added edges and $a_2(s, x)$ identifies the shorter (with a corresponding differentiation of the dropped edges).

To implement these attribute functions, let E denote the set of all elements that may be identified as attributes of moves. In the traveling salesman example, E consists of the set of all edges of the graph. It is of course possible also to identify other relevant attributes of moves in this example that make the nature of E more complex; e.g., we may introduce $a_5(s, x)$ and $a_6(s, x)$ to refer to the values $c(x)$ and $c(s(x))$, in which case E would consist not only of edges but also of tour lengths. More generally, in such cases E should be treated as a collection of sets that contain elements for different classes of attributes. (It may be noted that the identification of $a_5(s, x)$ and $a_6(s, x)$ indicated here corresponds more closely to the development of the type of aspiration level criteria discussed in Section 3.) Aspiration levels are then defined relative to the elements of E, and governed by the same tenure structure previously indicated for creating and maintaining tabu lists.

In particular, a different tabu list T_p, $p = 1, \ldots, g$, is created for each attribute $a_p(s, x)$, defining

$$T_p = \{a_p(s_h, x^h): h > k - t_p\},$$

where s_h and x^h refer to the move s and vector x at iteration h. In the traveling salesman setting this corresponds to creating a separate tabu list for each edge of the 2-OPT swap, and giving each list T_p its own length t_p. (The shorter of the two edges dropped from a tour may reasonably be given a shorter tenure on a tabu list than the longer, for example, to allow it to be added back more quickly.)

Tabu status, or equivalently "aspiration status," is now defined by linking these tabu lists with an aspiration function $A(e)$ defined on the elements e of E. Thus, when an element $e = a_p(s, x)$ is an attribute of the move from x to $s(x)$ (as for example, where e is one of the edges added to the tour represented by x), the value $A(e)$ is updated according to criteria analogous to those indicated in Section 3, e.g., $A(e) = \text{Min}(A(e), c(x))$ or $A(e) = \text{Min}(A(e), c(x), c(s(x)))$, etc. $A(e)$ initially is assigned a large ("infinite") value for each e in E, and is subsequently again assigned this value whenever e does not belong to any of the tabu lists.

The tabu status, or aspiration status, of a move overall then can be made a function of the status of its attributes. Given that a candidate move s applied to x will produce an objective value $c(s(x))$, it is natural to assign an attribute of this move a passing status (indicating that it passes the aspiration test) if $c(s(x))$ is

smaller than the aspiration value for the indicated attribute. Specifically, an attribute $a_p(s, x)$ receives a passing status if $c(s(x)) < A(e)$ for $e = a_p(s, x)$. All elements e not on a tabu list therefore automatically receive a passing status. Others receive a passing status if the move to which they contribute produces an objective value superior to the aspiration value recorded when such elements became tabu. The move s, or more precisely the transition from x to $s(x)$, therefore may be assigned a passing status if all, or a selected number of its attributes receive a passing status. (Related alternatives might include, for example, assigning a move a passing status if an attribute of sufficient importance receives a passing status.)

By this development, it is apparent that the representation of the tabu set T as a collection of moves (i.e., as a subset of S) in the Simple Tabu Search Procedure of Section 2 is not adequate for broader concerns. In general, T may be viewed as a collection of pairs (s, x), hence of solution-specific moves, characterized by a selected set of attributes. The specification $s \in S(x) - T$ thus becomes replaced by the specification $s \in S(x)$ and $(s, x) \notin T$. (In the case where the pairs (s, x) of T are capable of identifying solutions, without reference to the moves applied to them, the exclusion of (s, x) from T may be interpreted more simply as excluding $s(x)$ from T.)

In sum, aspiration criteria and tabu restrictions can be made subject to a common organizational framework, and viewed as different aspects of the same conceptual principle. At an extreme, as aspiration $A(e)$ that is set smaller than any possible $c(x)$ value corresponds to a preemptive tabu status. The motive underlying this form of integration of aspiration criteria and tabu restrictions is the hypothesis that different attributes of moves indeed can have relatively different influences on the quality of solutions generated, and thus should be subject to different tenures of tabu status (i.e., should be recorded on tabu lists of different lengths) and be governed by different levels of aspiration—both of which can be achieved by the same general mechanism.

5. Representation of the Search by a Directed Graph

The tabu search process as described this far can be viewed conveniently from a graph theory perspective. Let $G = (N, A)$ be a diagraph whose node set N is the set X of trial solutions, and whose arc set A is the set of all ordered pairs $(x, s(x))$; that is, the arcs of G consist of all solution-specific moves obtained by applying the moves of S to the elements of X. Thus, a particular move s corresponds to the collection of arcs $\{(x, s(x)): x \in X(s)\}$. The set of arcs in the forward star of a given node x, i.e., that take x as their initial endpoint, is $\{(x, s(x)): s \in S(x)\}$. We note that G is symmetric, since for each arc $(x, s(x))\}$ there is an arc $(s(x), x)$, obtained by applying the move s^{-1} to node $s(x)$.

A sequence of trial solutions obtained by applying a succession of moves to a given starting trial solution, hence a sequence of the form $x^1, x^2, \ldots, x^r$, where $x^{t+1} = s(x^t)$ for some $s \in S(x^t)$, $i = 1, \ldots, r$, identifies a path in G whose arcs are the ordered pairs $(x^t, s(x^t))$. Thus, a series of iterations of tabu search traces a path in G. By reference to the iteration counter k, the successive arcs of this path have the form $(x, s_k(x))$ for consecutive values of k. By these conventions, a tabu list T of the form $T = \{s_h^{-1}: h > k - t\}$, or of the form $T = \cup C_h$: $h > k - t$ (where $s_h^{-1} \in C_h$), consists of collections of arcs that include each arc $(s_h(x), x)$ which is the symmetric counterpart of the arc $(x, s_h(x))$ generated during one of the last t iterations of the method. Thus, in particular, T includes as a subset the arcs of the path that visits the nodes $s_h(x)$, for $h \geq k - t$, in reverse order.

From a theoretical standpoint, analysis of the effect of T on the sequence of solutions generated depends both on the nature of the moves in S and on the attributes of these moves which are used to identify the collections C_h. A potentially appealing starting point for such an analysis is the simple digraph that results for pure zero-one integer programming problems when X is the set of vertices of the zero-one unit hypercube, $c(x)$ is a penalty function defined over this set (e.g., to assure that constraints such as $Ax \leq b$ are satisfied at optimality), and S is the set of moves that transition among adjacent vertices (i.e., each move has the form $s(x) = x + e_j$ or $s(x) = x - e_j$ for some $j = 1, \ldots, n$). Then S and X respectively have $2n$ and 2^n elements, every x has exactly n arcs leaving it, and every s is a collection of 2^{n-1} arcs. The implications of applying a simple version of tabu search to such a graph G, under various assumptions about $c(x)$ and the relation of t and n, may provide a useful foundation for understanding the behavior of more advanced versions of the procedure in more complex settings.

6. The Role of Dominant and Deficient Moves

Dominance and equivalence operate somewhat differently in tabu search than in usual optimization contexts by requiring reference to the nature of the move employed. To illustrate, consider a zero-one knapsack problem of the following form

Maximize

$$7x_1 + 6x_2 + \cdots + 4x_7$$

$$4x_1 + 5x_2 + \cdots + 3x_7 \leq 20$$

If a tabu condition prevents a move that replaces $x_1 = 0$ by $x_1 = 1$, then by apparent dominance considerations it should also prevent the move that replaces $x_2 = 0$ by $x_2 = 1$. In the reverse direction, a tabu condition that prevents a move replacing $x_2 = 1$ by $x_2 = 0$ should also prevent the move from $x_1 = 1$ to $x_1 = 0$. Note that such dominance, characterized relative to moves, is not the same as the type of dominance that implies a variable will be zero in an optimal solution. (Here, possibly both x_1 and x_2 may equal 1.)

In the case where x_2 has coefficients identical to those of x_1, so that their corresponding moves are viewed as equivalent, similar extensions of tabu restrictions apply. Equivalence operates differently from strict dominance, however, in that a move is not allowed if a strictly dominating alternative is available. (E.g., the move from $x_1 = 1$ to $x_1 = 0$ will not be allowed if $x_2 = 1$ in the current trial solution.) Equivalent moves carry no corresponding limitation, though they may be ranked arbitrarily in order to be treated in the same manner as strictly dominating moves.

Since tabu search prevents certain moves from occurring, it may well be that employing the evaluation criteria of a standard heuristic within OPTIMUM may lead to selecting as "best" a move that should not be made. For example, in the knapsack problem one heuristic is to swap variables in the solution as long as feasibility is maintained, replacing $x_i = 1$ by $x_j = 1$ for a specified pair i, j, and evaluating the swap based on its contribution to the objective function. The heuristic operates without complication when employed as a hill climbing procedure, but new circumstances arise when the heuristic is embedded within the OPTIMUM function. The evaluation of the heuristic must then be modified as a result of the ability of tabu search to encounter conditions not met during hill climbing. Specifically, in the preceding knapsack problem example, the swap that replaces $x_1 = 1$ by $x_2 = 1$ can be seen to have a higher evaluation, i.e., yield a better objective function change, than the swap that replaces $x_1 = 1$ by $x_7 = 1$. (Note that both represent nonimproving moves.) Tabu search, however, classifies the former move as a *deficient move*, because it allows no possibility of an improved solution. Such moves are excluded from consideration by OPTIMUM regardless of the fact that they may receive higher evaluations than other moves by a standard hill climbing heuristic.

7. Tabu Lists and Strategic Oscillation

A major aspect of tabu search derives from inducing search behavior that compounds the types of patterns produced by a single tabu list. One of the effects of such a tabu list is to create a succession of solutions in which the objective function value oscillates. It is advantageous in certain applications, as where constraints may confine feasible solutions to a fairly narrow region, to use additional tabu lists to induce strategic oscillation of other parameters. To illustrate, one problem successfully approached in this matter was a lock box application modeled as a p-median problem with p fixed. A second tabu list was created for this problem that compelled successive moves to climb or descend to alternating depths on each side of the fixed p value, keeping track of the best candidate solution each time the value p was reached at the point of crossing. A similar approach clearly could be applied to problems where p is variable.

The use of strategic oscillation of this type has several advantages. First, it permits the execution of moves that are less complex than might otherwise be required. In the lock box application, for example, moves lead that directly from one solution to another while enforcing no deviation from the fixed p value involve exchanges that are combinatorially more numerous (and that require more effort to evaluate) than the moves that allow variations around p. A second advantage is that moving outside of a boundary and returning from different directions uncovers opportunities for improvement that are not readily attainable when the search is more narrowly confined. Such procedures also encourage the use of a function for evaluating the attractiveness of moves that varies depending on the location and direction of search (see, e.g., [3]).

From a more general standpoint, many types of heuristic search procedures have "mirror opposites," as exemplified by the dichotomous classifications of methods as "constructive or destructive," "interior or exterior," "feasible or infeasible," "primal or dual," and so forth. Rather than selecting only one type of search from such a classification, a wider range of opportunity is opened by alternating between them. Opposing strategies typically can be organized so that each point of crossing from one to the other represents a point of local optimality (relative to a given search orientation). Tabu lists that do not simply prohibit certain move reversals, but compel such crossings and returns, offer an effective way to avoid the suboptimal entrapment of standard searches.

An example of this latter type occurs for a multidimensional knapsack problem from a capital budgeting application, where the boundaries between feasibility and infeasibility can be crossed in either direction by successively incrementing or successively decrementing variables. (Heuristics based on locally best increments and decrements are easily specified for this problem because the coefficients of its inequality constraints are all nonnegative.) Thus these heuristics can be readily embedded in an oscillating tabu search

procedure that compels a specified number, say k, of moves beyond the feasible boundary in a given direction before permitting a return. In one variant, k starts at 1 and is successively incremented to a maximum, m, then successively decremented back to 1. Another variant allows $m + 1 - k$ iterations for each value of k, thus focusing more heavily on moves closer to the boundary. In addition to the conditions compelling this oscillation, two tabu lists are employed to prevent cycling, one list governing reversal of increments and one governing reversal of decrements. (Such an approach can of course be used to rebound to alternate depths from a boundary rather than to cross it.)

Another advantage of an oscillating strategy is to permit the discovery of good solutions for perturbed conditions, where constraints may be slightly relaxed or tightened. Knowledge of solutions that are inside and outside feasibility boundaries can be useful to a variety of analyses. As in the case of the type of tabu conditions that prevents move reversals, the conditions that compel oscillation can be based on graduated penalties rather than preemptive stipulations. Oscillations then are driven by a *pendulum function*, which redistributes weights associated with satisfying specified sets of constraints (treated hierarchically or in parallel) and with improving the objective function value. The redistribution causes the search to reverse direction at selected, possibly variable, distances (or numbers of moves) from a boundary. The pattern of behavior relative to a given set of constraints may be viewed as roughly analogous to that associated with a sine wave. An inviting avenue of research, not yet explored, is to impose constraints on alternative "harmonics" that are coordinated to bring their points of boundary crossing progressively closer until they at last converge (before separating again at the next swing of the pendulum). Such a strategy may be employed to provide good solutions to problems whose constraints may otherwise be difficult to handle.

8. Intermediate and Long-Term Memory Functions

Intermediate and long-term memory functions are employed within tabu search to achieve regional intensification and global diversification of the search. Combined with the short-term memory functions fulfilled by the tabu lists, the intermediate and long-term functions provide an interplay between "learning" and "unlearning." We comment only briefly on their form here, providing an additional specific illustration of how they can be employed in the context of the mixed integer programming problem in Part II of this study.

Intermediate term memory operates by recording and comparing features of a selected number of best trial solutions generated during a particular period of search. Features that are common to all or a compelling majority of these solutions (such as values received by particular variables) are taken to be a regional attribute of good solutions. The method then seeks new solutions that exhibit these features, by correspondingly restricting or penalizing available moves during a subsequent period of regional search intensification. Instances of this strategy have been applied in early heuristic approaches to the traveling salesman problem (see, e.g., [8, 16]).

To illustrate, the structure of tabu search leads at once to one type of intensification strategy that is useful for solving large problems. Because the search strongly tends to focus on generating solutions that are good, it also tends generally to incorporate only a subset of decision elements (variables, edges, etc.) into these solutions. For example, in the traveling salesman context, for moderately dense graphs the number of different edges incorporated into the tours on any given solution pass is generally only a fraction of the total edges. Thus, after some initial number of iterations, the method can discard all edges not yet incorporated into any tour and then devote itself to the resulting smaller problem. Because iterations are now much faster to execute, the search can examine many more alternatives in a given span of time, as well as focus on possibilities that are likely to be attractive.

More advanced uses of intermediate-term memory involve creating a network of good solutions as a matrix for generating other solutions with good properties. In particular, a collection of good solutions is used to define a subregion of the search space that contains these points (and others nearby or reachable by convex combinations, etc.) to provide a focus for intensified search, and for launching explorations into neighboring regions. Such an approach can be augmented by the use of advanced forms of discrimination analysis, which seek to generate a parsimonious collection of inequalities or logical conditions of two types: (1) to encompass good solutions; (2) to separate good solutions from bad. In combinatorial settings, such conditions will commonly be based on discrimination functions involving both hierarchical (conditional) structures and separating hyperplane structures.

An interesting avenue for research is to seek to characterize patterns for good solutions that can be expressed as trajectories; e.g., that specify transitions among good solutions by reference to inferred difference equations. A simple example is the following. Tabu search itself provides a trajectory that links good solutions, but this trajectory can be streamlined by creating a smallest set of moves to go from each improved solution to the next, and reordered (e.g., fanning out from one or more central points, or inward from

peripheral points) to give a basis for extending the trajectory into additional regions. Moves then can be chosen that most nearly match directional vectors established by such trajectories or that extrapolate systematic changes in values of variables in going from one solution to another.

The long-term memory function, whose goal is to diversify the search, employs principles that are roughly the reverse of those for intermediate-term memory. Instead of inducing the search to focus more intensively on regions that contain (or that can be extrapolated from) good solutions previously found, the long-term memory function guides the process to regions that markedly contrast with those examined thus far. The approach differs from those methods that seek diversity by generating a series of random starting points, and hence which afford no opportunity to learn from the past. The objective is to create evaluation criteria that can be used by a heuristic search process which is specifically designed to produce a new starting point, thus generating such a point by purposeful instead of random means. (Uses of restricted randomization by means of probability assignments are discussed in Section 9.) These evaluation criteria penalize the features that long-term memory finds to be prevalent in previous executions of the search process.

A traveling salesman application again provides a convenient illustration. A simple form of long-term memory in this setting is a count of the number of times each edge appears in the tours generated. Penalizing each edge on the basis of this count favors the generation of "good" starting tours (according to the heuristic chosen for producing such tours) that tend to avoid those edges most commonly used in the past. (This sort of approach can be viewed as using **frequency based** tabu criteria in contrast to **recency based** tabu criteria.) One way to implement such a long-term strategy is by means of a long-term tabu list, periodically activated, which may employ tabu conditions of increased stringency to drive the solution process into unexplored territory.

The diversity achieved by this criterion can be increased by retaining the penalties for a period after transferring to the (possibly different) heuristic incorporated within tabu search, and using the tabu search procedure to seek improved solutions. Afterward, the penalties are dropped and tabu search proceeds according to its normal evaluation criteria. This same type of procedure can be used to continue directly from the present point of search to a new region without going back to generate a new solution from scratch.

Over a wider time frame, the intermediate and long-term functions can be layered in additional, potentially overlapping levels. An inviting avenue of research is to join such a layered memory strategy with a pattern recognition component, thereby refining and nesting the concept of "regions" to which intensification and diversification of search is applied. Ample opportunity exists for parallel processing that pursues paths from divergent starting points, then pulls back and weeds out the best outcomes, as a means of expanding the base of possibilities.

Another application of these ideas occurs in settings where local optima are strong attractors, or "mini black holes" that can be left behind, once visited, only by an especially strong effort. For example, preliminary experimentation may disclose that under certain conditions—as when a local optimum is reached by a fairly long or steep descent—the path away from the local optimum consists of a slow and irregular upward climb. In such cases, a faster and more direct withdrawal from a local optimum may be desirable.

Three strategies appear relevant to an accelerated retreat, drawing on a mix of intensification and diversification ideas. First is to impose more restrictive tabu conditions, which may include the use of attributes to characterize tabu status that exclude larger sets of moves than usual from consideration. Second is to use the history of the first part of the upward climb to project where the process is heading, and then to favor moves that lead in the direction thus established. Third is to invoke a strategy that penalizes moves based on the number of iterations their identifying attributes have been included in solutions of the past, for a chosen horizon. After the retreat from the local optimum has progressed sufficiently (by a measure experimentally established for the application), the strategy is relaxed and the method resumes its normal operation.

The process of invoking and relaxing such a strategy of retreat, based on the contour of the solution path, can also be used to invoke and relax ordinary tabu conditions in response to the solution trajectory. The use of a dynamic aspiration level, drawing on the ideas sketched earlier, may in fact be viewed as one means for pursuing such an objective. Conceptually, the goal in all of these processes, as in the simplest types of tabu search schemes, is to obtain a useful estimate of an **escape distance** from a local optimum, as a way to determine which moves should be considered legitimate.

9. Probabilistic Tabu Search

The incorporation of a probabilistic element within tabu search offers an auxiliary means of pursuing its basic strategic goals. Basing choices of moves on the assignment of probabilities yields several interesting trade-offs between different forms of control and different sources of efficiency. On one level, control is gained by the inclusion of new search parameters (i.e., the weights which determine the probabilities assigned),

while on another level control is lost by the recourse to randomization that a more systematic approach would avoid. We propose the view that randomization is a means for achieving diversity without reliance on memory. (Diversity in this context ranges from the local type provided by the operations of a tabu list, to the global type provided by the long-term memory approaches of the preceding section.) By this view, the use of randomization, via assigned probabilities, allows a gain in efficiency by obviating extensive record keeping and evaluation operations that a more systematic pursuit of diversity may require. At the same time, it entails a loss in efficiency by allowing duplications and potentially unproductive wandering that a more systematic approach would seek to eliminate.

Three strategic principles of tabu search stand out among those that lend themselves to a probabilistic treatment. In summary, they may be expressed as follows: (1) more attractive moves, yielding smaller $c(x)$ values under minimization, have a higher status (and thus should receive higher probability of acceptance); (2) the status of a move is diminished (entailing a lower probability of acceptance) if it belongs to a class that includes the reversal or, alternatively, repetition of a move recently made; (3) thresholds of aspiration, based on previous performance, can override an otherwise diminished status, yielding a reinforced or even a preemptive basis for selection (acceptance with probability 1). For problems with large numbers of alternatives, these principles operate within the context of restricting attention to candidate sets of moves.

As a basis for comparing a probabilistic version of tabu search with another probabilistic strategy, it may be noted that these principles are markedly different from those underlying simulated annealing. In simulated annealing, for example, all improving moves have the same status, as reflected in the stipulation that any improving move encountered in a random sequence of examination will be accepted automatically. Nonimproving moves in simulated annealing have a status which starts relatively high when $c(x)$ is high (though the change induced in $c(x)$ also plays a role) and gradually diminishes in successive stages until the probability of acceptance is extinguished. In accordance with this, a starting solution is selected which is far from optimality, to allow the process to operate for an extended time at stages where $c(x)$ takes on larger values, and (presumably) to assure that the local optimum to which the process ultimately descends will be a good one.

The elements that underlie the assignment of probabilities for tabu search, by contrast, maintain the distinction between relative attractiveness of alternative moves at all stages, without giving unimproving moves higher status at the beginning and without progressively diminishing their chance for consideration at later stages. Likewise, tabu search does not specify that the process should start with a solution far from optimality or that the best solution is to be identified by the local optimum reached at the conclusion of an eventual undeviating descent. Such strongly contrasting strategic frameworks are likely to have similarly contrasting effects. (Effects that lead to different types of solution paths do not necessarily imply differences in solution quality. However, it is predicted in [3] that efforts to improve simulated annealing will lead to replacing its rules with those more closely resembling the prescriptions of tabu search—a prediction that appears to gain support from recent studies.[1,9])

The embodiment of the preceding first-order strategy considerations of tabu search in a probabilistic approach is entirely straightforward. The probabilities governing the acceptance of moves from a specified candidate set derive from three sources, extracted directly from the three strategy considerations indicated. These sources are: (S1) move attractiveness, related to changes induced in $c(x)$; (S2) tabu status, related to tenure on a tabu list; (S3) aspiration level, related to the value of $c(x)$ achieved in relation to a historical standard (possibly maintained as an inverse form of tabu status).

Consider first the source (S1). By our orientation, the introduction of probabilities has the role of lessening dependence on memory (i.e., the element of randomization makes it possible to be different from the past without remembering the past). Accordingly, we can pose an extreme, or "degenerate," form of probabilistic tabu search based on (S1) alone. For this extreme (and also for versions incorporating considerations from other sources) the determination of probabilities can be made in both a nonsequential and a sequential manner.

In the nonsequential approach, consider a positive-valued weight function $w(s)$ which reflects the relative attractiveness of applying the move s to the current solution x; e.g., $w(s') > w(s'')$ if $c(s'(x)) < c(s''(x))$. (The function w may depend on x in ways other than determined by the objective function value.) Then, if $S^*(x)$ is a candidate set of moves taken from $S(x)$, we may choose a move s' from $S^*(x)$ by constructing a sample of size 1 from the uniform distribution with the probability $P(s')$ given by

$$P(s') = \frac{w(s')}{\sum w(s):\ s \in S^*(x)}.$$

A sequential approach can be based on an approximation of the foregoing probability assignment (e.g., by starting with a small sample of moves from $S^*(x)$ as a first guess for the denominator of $P(s')$), or can take a somewhat different form. One set of alternatives

arises by viewing the choice of a move as the outcome of duels between successive contestants. Each duel produces a winner (which begins with an arbitrary move from the set). As each move in succession competes with the previous winner, the better of the two contestants (the one that yields a smaller value of $c(x)$) is selected with a prescribed probability. For example, in a very simple type of strategy, the better of two moves may be chosen as the new winner with probability 0.8, 0.7, or 0.6 according to whether: (a) only one of the two contestants is an improving move, (b) both are nonimproving or (c) both are improving. Such a sequential strategy creates a bias against earlier moves in the sequence, and hence it is appropriate to compensate for this by adjusting the indicated probabilities to start higher and decay through the sequence.

Next consider the source (S2). The use of probabilities based on this source yields a strategy that more closely resembles the form of constrained search that occurs in the nonprobabilistic case. Whether used in conjunction with (S1) or in isolation, (S2) has the effect of altering the choice process by introducing a probability of acceptance based on tabu status, where tabu moves with shorter residence on a tabu list (i.e., which have become tabu more recently) receive a lower probability of acceptance than those of longer residence. We illustrate how this operates in the sequential mode of examination. On a tabu list of length 10, for example, we may define "tabu probabilities" assigned to moves from shortest to longest residence as the elements from a sequence such as

$$1/10 \quad 1/9 \quad 1/8 \quad 1/7 \quad \cdots \quad 1/3 \quad 1/2 \quad 1$$

or

$$0.1 \quad 0.2 \quad 0.3 \quad 0.4 \quad \cdots \quad 0.8 \quad 0.9 \quad 1$$

or some combination of the two. (Any monotonic sequence that begins with a fraction and ends with the value 1 is appropriate.) In reality, the length of the tabu list for this example is effectively 9, since the tenth element is accepted for consideration with probability 1. All non-tabu moves, similarly, are regarded to have tabu probabilities equal to 1.

In comparing two moves, whether incorporating probabilities based on (S1) or not, the move normally selected as the winner will be accepted as the new incumbent with a probability which is the ratio of its tabu probability to that of its competitor, bounding this ratio above by 1. For example, a winner which is not a tabu move will be accepted automatically. If the winner is a tabu move with tabu probability 1/6 and its competitor is a tabu move with tabu probability 1/3, then the winner will be accepted as the "true winner" only with probability 1/2 (= 1/6 ÷ 1/3). By analogy to the case for sequential examination using (S1), adjustments can be made to offset the bias against a tabu move selected as a winner earlier in the sequence. Such tabu probabilities can also be used in a nonsequential choice process by multiplying them by the weights $w(s)$, $s \in S^*(x)$, to produce amended weights for an assignment of probabilities under (S1).

Finally, the use of (S3) in producing probabilities is similarly straightforward. Aspiration levels can offset the diminished chance of acceptance produced by tabu probabilities, and can also amend probabilities that are assigned without reference to tabu status. Based on the stringency and tenure of an aspiration level, a move whose probability of acceptance is determined by criteria based on either (S1) or (S2) may have this probability elevated part or all of the way to 1.

Clearly it is possible to mix the probabilistic and nonprobabilistic elements of tabu search in a variety of ways. For example, a tabu list can be divided into probabilistic and nonprobabilistic components, where a specified number of elements most recently added receive a tabu status in the ordinary way. Equivalently, these elements may be given a tabu probability equal to 0 in the type of implementation illustrated for (S2). Similarly, moves that exceed other thresholds can be given acceptance probabilities of 0 to 1 as a means of blending the probabilistic and nonprobabilistic elements.

The foregoing proposed uses of probabilities can easily be controlled to assure an optimal solution will be found with probability 1 as the number of iterations is allowed to grow indefinitely. Such a result occurs even if (S1) is the only source of the probabilities, provided $S^*(x)$ is taken to be $S(x)$ (or a nonempty random sample of $S(x)$) and all assigned probabilities are bounded from below by a positive constant, given that elements are examined in a random order for the sequential case. Then it is only necessary to assume that the digraph G of Section 5 is finite, and a directed path exists from each node of G to each other node. We sketch an informal proof as follows. Under the indicated assumptions, each time a node of G is visited by the method, there is a positive probability of choosing each arc that leaves the node. (This probability is bounded from below by a constant which is some fraction of the positive constant of the assumptions.) Suppose that a particular node (e.g., corresponding to an optimal solution) is never visited. Then after an initial finite number of iterations, there are two subsets of nodes, N' and N'', where each node in N' will be visited an infinite number of times and each node in N'' will never again be visited. At least one node of N' must have an arc that leads to a node of N''. The probability of not choosing this arc is bounded above

by a constant less than 1. Thus, the probability that N' does not contain the associated node of N'' goes to 0 as the number of iterations goes to infinity, which establishes the assertion that the initial excluded node will be visited with probability 1.

Probability assignments based on incorporating (S2) and (S3) similarly can easily be controlled to assure the same outcome will hold. Aspiration levels that elevate probabilities of selecting moves that produce new solutions can locally override the stipulation that no move in $S^*(x)$ receives a probability smaller than a positive constant. It is also permissible to periodically allow the probabilities to be more nearly uniform for a selected number of steps—a strategy analogous to purging the tabu list in the nonprobabilistic version. It is noteworthy that the assurance of finding an optimal solution does not depend on the heuristic power of skewing the probabilities to conform with the principles of tabu search. This fact encourages the orientation that probabilities operate as an escape hatch to hedge against persistent wrong moves in the absence of reliable knowledge (or a more perceptive strategy) to guide the search.

Analogous incorporation of probabilistic elements in intermediate and long-term memory functions of tabu search are likewise possible. In each case, incorporating a randomized element introduces trade-offs in memory capacity and efficiency of processing. (For example, increased reliance on a random factor in (S1) should decrease the effective length of a tabu list, while increased reliance on a random factor in (S2) should have the opposite effect.) Experimentation to characterize the relative effects of these trade-offs remains to be undertaken.

10. Practical Applications and Future Directions

Tabu search originated as a method for solving real world combinatorial problems in scheduling and covering. Its adaptability to solving other types of combinatorial problems by changing the definition of a move, and by modifying the definition of OPTIMUM and the composition of the tabu lists, brought about its application to additional problem settings. Some of the practical problems to which it has been applied are as follows.

1. A scheduling problem for distributing workloads among machines to assure all processing demands are adequately met over the time horizon. This problem was one of the first tabu search applications,[2] formulated as a nonlinear (quadratic) generalized covering problem involving 300 to 500 integer variables and approximately 100 constraints with positive right hand sides. Using the strategic oscillation approach of Section 7, tabu search succeeded in obtaining solutions with objective function values less than half as large as other methods tested, and required 10 to 20 seconds CPU time per problem on the CDC 6600 computer.

2. A computer channel balancing problem for assigning loads to channels to minimize deviations from targets, constituting a generalized multidimensional bin packing problem.[3,4] Problems corresponding to zero-one MIPs involving 106 to 1000 variables were solved in less than 1 minute on a Honeywell DPS-8 computer. The resulting solutions were dramatically superior to those obtained by the commercial MPS system, which was allowed to run up to 15 minutes and consumed from 9 to 150 times as much CPU time as tabu search per problem. For the goal of minimizing deviations, the smallest, median and largest of the "minimized" deviations for 20 problems were as follows[4]:

	Deviations		
Method	Smallest	Median	Largest
Commercial MPS program	60	7327	Infinity[a]
Tabu search	0	10	18

[a] No feasible solution found.

3. A subset criterion and clustering problem used in space planning and architectural design. Problems corresponding to zero-one MIPs with over 25,000 variables and 50,000 constraints were solved in less than one minute on a V77 minicomputer. (The software is incorporated in the SPDS system used by commercial space planning firms, as reported in [6].)

4. A large scale employee scheduling problem to assign employees to work stations and duty rosters, while satisfying restrictions imposed by time period demands, employee availabilities, union rules and company policy.[7] The integration of tabu search with pattern recognition and decomposition strategies succeeded in solving problems corresponding to zero-one MIPs with 1,000,000 to 4,000,000 variables and 3,400 to 9,000 constraints to within 98% of optimality in 22–24 minutes on an IBM PC microcomputer.

In addition to such practical applications, experiments are currently in progress applying tabu search to classical combinatorial problems. A preliminary exploration of the traveling salesman problem[15] has tested seven problems, the first three ranging from 25 to 57 cities with known optimal solutions, and the last four ranging from 50 to 110 cities, representing "small but hard" problems from the study by Stewart[21] whose optimal solutions are unknown.

Using the standard moves of the 2-OPT heuristic,[16] and employing the simple tabu list and aspiration level structure suggested for the traveling salesman

problem in [3], tabu search easily found the known optimal solutions for each of the first three problems, requiring an average of 5 minutes (depending on the randomly generated starting tour) for the largest problem (57 cities) on an IBM AT microcomputer. The four "hard" problems each were subjected to 16 solution attempts, employing four different starting solutions and four different tabu list sizes (ranging from $n/4$ to n). The best solutions generated by tabu search for all but the 110 city problem typically were found in less than one minute on a VAX 11/780 and in all but 5 of the 64 runs these solutions were superior to the previous best solutions reported for these problems (matching the previous best for all but 2 of the remaining runs). The 110 city problem lived up to its reputation in difficulty, requiring from 4 minutes to nearly 20 minutes to find the best solutions for a given run, though in each case this solution was superior to the best previously found.

Figures 1 and 2 show the results of two typical runs for a 75 node problem from the "hard" group. The horizontal line in these figures identifies the tour length of 553 previously conjectured to be optimal. The graphed points in the figures identify the subset of solutions generated by tabu search that consisted of those "local optima" (generated subject to the tabu conditions) which were strictly better than all their predecessors. (Thus, other local optima were generated but not shown. The method does not keep track of these or note their "local optimality" character.) The first point in each figure therefore represents the local optimum found by making 2-OPT moves before the tabu search procedure is activated. In each case tabu search goes well beyond this first point, and also beyond the best previously known tour length.

An interesting feature of this comparison is that the previous best solutions from the study of [21] were

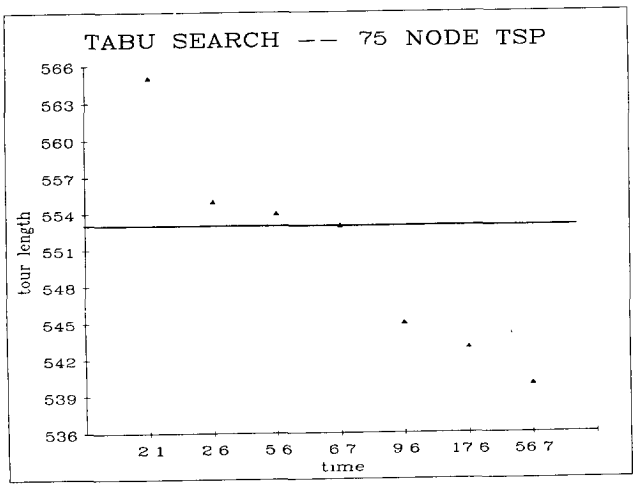

Figure 1. Typical run for 75 node problem.

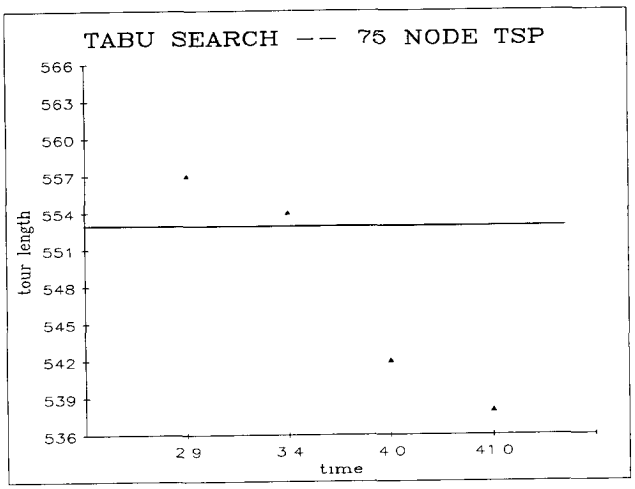

Figure 2. Typical run for 75 node problem.

obtained by multiple solution attempts using a refinement of the 3-OPT heuristic, which is somewhat more powerful than the 2-OPT heuristic incorporated into the tabu search implementation. The outcome illustrates the ability of tabu search to guide a relatively simple heuristic in a manner that allows it to outperform procedures that are normally superior. Such an ability can be advantageous in applications where, unlike the traveling salesman setting, the problem does not have a simple structure susceptible to highly specialized procedures, and the state-of-the-art is unable to benefit from years of study on sophisticated ways to exploit such structure.

Further insights into the performance of tabu search are provided by Figures 3 and 4, which show the pattern of solutions generated over a subsequence of iterations for the "easy" 42 city problem. Figure 3 shows the first 500 iterations, starting from an evidently very poor initial tour, and Figure 4 shows the sequence from iteration 30 to iteration 130 in greater detail. This latter figure gives a clearer indication of the variability of tour lengths produced. (The optimum tour was found on iteration 83.) At the same time, it is clear that many of the tours generated are "good tours." This additional characteristic of tabu search—the tendency to produce a variety of solutions that fall in an attractive range—can be useful in situations where a mathematical model is used to generate candidate solutions that must then be evaluated further on the basis of external criteria.

An incidental finding of interest from this preliminary study concerns the appropriate magnitude of tabu list sizes. Previous applications had found effective tabu list sizes to lie in the range from 5 to 12, clustered around 7, a finding that appeared to be independent of problem size and structure. The much larger tabu list sizes for the traveling salesman problem, and their dependency on problem size, show that the choice of a

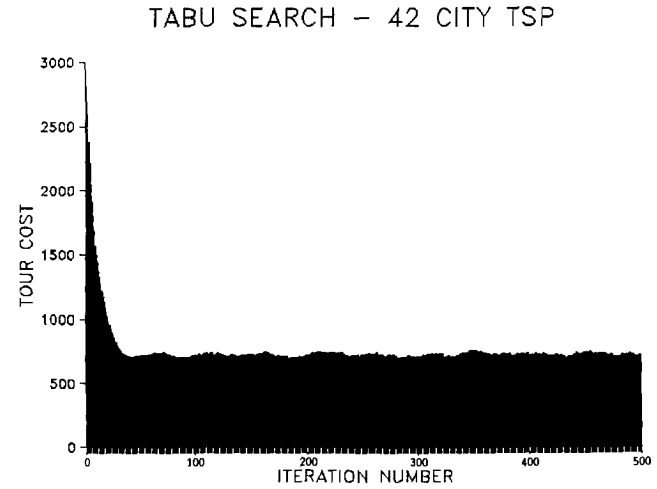

Figure 3. Tabu search, first 500 iterations.

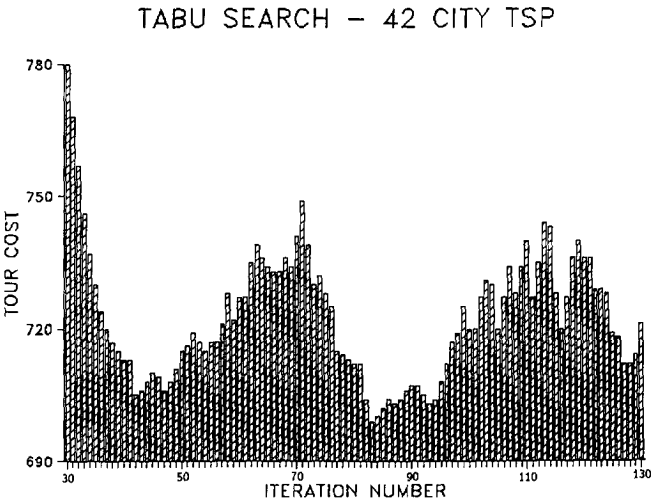

Figure 4. Tabu search, detail of iteration 30 to iteration 130.

good tabu list size is more subtle than previous empirical outcomes had suggested.

Studies have recently appeared that compare tabu search to simulated annealing. The study by Hertz and de Werra[12] conducts experimental comparisons for graph coloring problems ranging from 100 to 1000 nodes, demonstrating that tabu search obtains solutions of significantly higher quality than simulated annealing in this setting, while expending less computational effort. An investigation by Bhasin, Carreras and Taraporevola[1] examines methods for cell layout problems in integrated circuit design. Adopting the objective of minimizing channel density, the authors similarly find that tabu search gives better results than simulated annealing and while requiring a significantly lower run time. A study by Malek[18] involves computational comparisons for the traveling salesman problem, applying tabu search with a long-term memory diversification strategy, patterned after that discussed in Section 8. Malek reports the ability to obtain optimal solutions to the test problems with substantially greater frequency using tabu search than simulated annealing, while consuming only 1/3 to 1/25 the computational effort. The related work of Hansen and Jaumard,[11] based on the independent development of Hansen,[10] likewise deserves mention in this regard, obtaining results superior to the simulated annealing for maximum satisfiability problems. On the other hand, the study by Guruswany, Owens and Pandya[9] suggests there may be value to combining simulated annealing and tabu search in a parallel processing environment.

Studies comparing tabu search to more specialized methods also have recently appeared. A study by Widmer and Hertz[22] compares tabu search to six alternative approaches to the flow shop sequencing problem. Using an iteration cutoff rule that keeps tabu search from running longer than 12 minutes on an IBM PC Computer, for problems involving 20 jobs and 20 machines, the authors find that tabu search obtains solutions superior to all other methods in 80% of the cases. Doubling the cutoff limit produces superior solutions in 90% of the cases. A sequel to this study by Hertz, de Werra and Widmer[13] reports comparisons for time tabling and layout problems, disclosing that tabu search methods gives better results than any other method tested. Interestingly, these latter two implementations of tabu search, and the one cited previously for the graph coloring problem, make no use of aspiration level criteria or of the strategic elements described in the sections following Section 2.

These outcomes offer encouraging evidence that tabu search may have a useful place among the tools for solving combinatorial optimization problems. At the same time, it must be acknowledged that the potential and limitations of this class of methods remain less than fully charted, and many aspects of tabu search—both theoretical and applied—remain to be discovered. Possibilities for merging with other procedures (including, for example, alternative search strategies derived from neural networks, genetic algorithms and simulated annealing) likewise offer intriguing avenues for exploration.

REFERENCES

[1] B. Bhasin, C. Carreras and G. Taraporevala, 1988. *Global Router for Standard Cell Layout Designs*, Department of Electrical and Computer Engineering, The University of Texas, Austin.

[2] F. Glover, 1977. *Heuristics for Integer Programming Using Surrogate Constraints*, **Decision Sciences 8:1,** 156–166.

[3] F. Glover, 1986. *Future Paths for Integer Programming and Links to Artificial Intelligence*, **Computers and Operations Research 13:5,** 533–549.

[4] F. GLOVER, 1987. *Tabu Search Methods in Artificial Intelligence and Operations Research*, **ORSA Artificial Intelligence Newsletter 1:2,** 6.

[5] F. GLOVER, 1988. *TABU-SEARCH, Version II.0, Annotated Experimental Computer Code for Traveling Salesman Problems* (April) (available from the author on request).

[6] F. GLOVER, C. MCMILLAN and B. NOVICK, 1985. *Interactive Decision Software and Computer Graphics for Architectural and Space Planning*, **Annals of Operations Research 5:** 557–573.

[7] F. GLOVER and C. MCMILLAN, 1986. *The General Employee Scheduling Problem: An Integration of Management Science and Artificial Intelligence*, **Computers and Operations Research 13:5,** 563–593.

[8] B.L. GOLDEN and W.R. STEWART, 1985. *The Empirical Analysis of TSP Heuristics*, in E.L. Lawler, J. K. Lenstra, and A.H.G. Rinnooy Kan (eds.), **The Traveling Salesman Problem**, North-Holland, Amsterdam.

[9] M. GURUSWAMY, H. OWENS and M. PANDYA, 1988. *TSP MIXER*, Department of Electrical and Computer Engineering, The University of Texas, Austin.

[10] P. HANSEN, 1986. *The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming*, presented at the Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

[11] P. HANSEN and B. JAUMARD, 1987. *Algorithms for the Maximum Satisfiability Problem*, RUTCOR Research Report RR#43-87, Rutgers, New Brunswick, NJ.

[12] A. HERTZ and D. DE WERRA, 1987. *Using Tabu Search Techniques for Graph Coloring*, **Computing 29,** 345–351.

[13] A. HERTZ, D. DE WERRA and M. WIDMER, 1988. *Some New Applications of Tabu Search*, presented at the 13th International Symposium on Mathematical Programming, Tokyo, Japan.

[14] S. KIRKPATRICK, GELATT, JR. and M.P. VECCHI, 1983. *Optimization by Simulated Annealing*, **Science 220:4598,** 671–680.

[15] J. KNOX, 1988. *An Application of Tabu Search to the Symmetric Traveling Salesman Problem*, Ph.D. thesis (in progress), Center for Applied Artificial Intelligence, University of Colorado, Boulder.

[16] E.L. LAWLER, J.K. LENSTRA and A.H.G. RINNOOY KAN, (eds.), 1985. *The Traveling Salesman Problem*, North-Holland, Amsterdam.

[17] M. LUNDY and A. MEES, 1986. *Convergence of an Annealing Algorithm*, **Mathematical Programming 34,** 111–124.

[18] M. MALEK, 1988. *Search Methods for Traveling Salesman Problems*, Department of Electrical and Computer Engineering, The University of Texas, Austin.

[19] S. SAVAGE, 1976. *Some Theoretical Implications of Local Optimization*, **Mathematical Programming 10,** 354–366.

[20] S. SAVAGE, P. WEINER and A. BACCHI, 1976. *Neighborhood Search Algorithms for Guaranteeing Optimal Traveling Salesman Tours Must Be Inefficient*, **Journal of Computer and System Sciences 12:1,** 25–35.

[21] W.R. STEWART, JR., 1987. *Accelerated Branch Exchange Heuristics for Symmetric Traveling Salesman Problems*, **Networks 17,** 423–437 (to appear in the **European Journal of Operations Research**).

[22] M. WIDMER and A. HERTZ, 1987. *A New Approach for Solving the Flow Sequencing Problem*, ORWP 87/15, Department of Mathematics, University of Lausanne (August).