TABU SEARCH

Fred Glover
Manuel Laguna

Abstract

This article explores the meta-heuristic approach called tabu search, which is dramatically changing our ability to solve a host of problems in applied science, business and engineering. Tabu search has important links to evolutionary and "genetic" methods, often overlooked, through its intimate connection with scatter search and path relinking — evolutionary procedures that have recently attracted attention for their ability to facilitate the solution of complex problems. The adaptive memory designs of tabu search have also provided useful alternatives and supplements to the types of memory embodied in neural networks, allowing enhancements of neural network processes in practical settings.

1. Tabu Search Background and Relevance

Faced with the challenge of solving hard optimization problems that abound in the real world, classical methods often encounter great difficulty. Vitally important applications in business, engineering, economics and science cannot be tackled with any reasonable hope of success, within practical time horizons, by solution methods that have been the predominant focus of academic research throughout the past three decades (and which are still the focus of many textbooks).

The meta-heuristic approach called tabu search (TS) is dramatically changing our ability to solve problems of practical significance. Current applications of TS span the realms of resource planning, telecommunications, VLSI design, financial analysis, scheduling, space planning, energy distribution, molecular engineering, logistics, pattern classification, flexible manufacturing, waste management, mineral exploration, biomedical analysis, environmental conservation and scores of others. In recent years, journals in a wide variety of fields have published tutorial articles and computational studies documenting successes by tabu search in extending the frontier of problems that can be handled effectively — yielding solutions whose quality often significantly surpasses that obtained by methods previously applied. Table 1.1 gives a partial catalog of example applications. A more comprehensive list, including summary descriptions of gains achieved from practical implementations, can be found in Glover and Laguna, 1997. Reports of recent TS implementations can also be found on the web page http://www.upt.pt/tabusearch.

A distinguishing feature of tabu search is embodied in its exploitation of adaptive forms of memory, which equips it to penetrate complexities that often confound alternative approaches. Yet we are only beginning to tap the rich potential of adaptive memory strategies, and the discoveries that lie ahead promise to be as important and exciting as those made to date. The knowledge and principles that have emerged from the TS framework give a foundation to create practical systems whose capabilities markedly exceed those available earlier. At the same time, there are many untried variations that may lead to further advances. A conspicuous feature of tabu search is that it is dynamically growing and evolving, drawing on important contributions

by many researchers. (The web site previously cited provides links to the work of these researchers.)

Table 1.1. Illustrative tabu search applications.

Scheduling
- Flow-Time Cell Manufacturing
- Heterogeneous Processor Scheduling
- Workforce Planning
- Classroom Scheduling
- Machine Scheduling
- Flow Shop Scheduling
- Job Shop Scheduling
- Sequencing and Batching

Design
- Computer-Aided Design
- Fault Tolerant Networks
- Transport Network Design
- Architectural Space Planning
- Diagram Coherency
- Fixed Charge Network Design
- Irregular Cutting Problems

Location and Allocation
- Multicommodity Location/Allocation
- Quadratic Assignment
- Quadratic Semi-Assignment
- Multilevel Generalized Assignment
- Lay-Out Planning
- Off-Shore Oil Exploration

Logic and Artificial Intelligence
- Maximum Satisfiability
- Probabilistic Logic
- Clustering
- Pattern Recognition/Classification
- Data Integrity
- Neural Network |Training and Design

Technology
- Seismic Inversion
- Electrical Power Distribution
- Engineering Structural Design
- Minimum Volume Ellipsoids
- Space Station Construction
- Circuit Cell Placement

Telecommunications
- Call Routing
- Bandwidth Packing
- Hub Facility Location
- Path Assignment
- Network Design for Services
- Customer Discount Planning
- Failure Immune Architecture
- Synchronous Optical Networks

Production, Inventory and Investment
- Flexible Manufacturing
- Just-in-Time Production
- Capacitated MRP
- Part Selection
- Multi-item Inventory Planning
- Volume Discount Acquisition
- Fixed Mix Investment

Routing
- Vehicle Routing
- Capacitated Routing
- Time Window Routing
- Multi-Mode Routing
- Mixed Fleet Routing
- Traveling Salesman
- Traveling Purchaser

Graph Optimization
- Graph Partitioning
- Graph Coloring
- Clique Partitioning
- Maximum Clique Problems
- Maximum Planner Graphs
- P-Median Problems

General Combinational Optimization
- Zero-One Programming
- Fixed Charge Optimization
- Nonconvex Nonlinear Programming
- All-or-None Networks
- Bilevel Programming
- General Mixed Integer Optimization

1.1 General Tenets

The word *tabu* (or *taboo)* comes from Tongan, a language of Polynesia, where it was used by the aborigines of Tonga island to indicate things that cannot be touched because they are sacred. According to Webster's Dictionary, the word now also means “a prohibition imposed by social custom as a protective measure" or of something "banned as constituting a risk.” These current more pragmatic senses of the word accord well with the theme of tabu search. The risk to be avoided in this case is that of following a counter-productive course, including one which may lead to entrapment without hope of escape. On the other hand, as in the broader social context where “protective prohibitions” are capable of being superseded when the occasion demands, the “tabus” of tabu search are to be overruled when evidence of a preferred alternative becomes compelling.

The most important association with traditional usage, however, stems from the fact that tabus as normally conceived are transmitted by means of a social memory which is subject to modification over time. This creates the fundamental link to the meaning of "tabu" in tabu search. The forbidden elements of tabu search receive their status by reliance on an evolving memory, which allows this status to shift according to time and circumstance.

More particularly, tabu search is based on the premise that problem solving, in order to qualify as intelligent, must incorporate *adaptive memory* and *responsive exploration*. The adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. Since local choices are guided by information collected during the search, TS contrasts with memoryless designs that heavily rely on semirandom processes that implement a form of sampling. Examples of memoryless methods include semigreedy heuristics and the prominent “genetic” and “annealing” approaches inspired by metaphors of physics and biology. Adaptive memory also contrasts with rigid memory designs typical of branch and bound strategies. (It can be argued that some types of evolutionary procedures that operate by combining solutions, such as genetic algorithms, embody a form of implicit memory. Special links with evolutionary methods, and implications for establishing more effective variants of them, are discussed in Glover and Laguna (1997).)

The emphasis on responsive exploration in tabu search, whether in a deterministic or probabilistic implementation, derives from the supposition that a bad strategic choice can yield more information than a good random choice. In a system that uses memory, a bad choice based on strategy can provide useful clues about how the strategy may profitably be changed. (Even in a space with significant randomness a purposeful design can be more adept at uncovering the imprint of structure.)

Responsive exploration integrates the basic principles of intelligent search, i.e., exploiting good solution features while exploring new promising regions. Tabu search is concerned with finding new and more effective ways of taking advantage of the mechanisms associated with both adaptive memory and responsive exploration. The development of new designs and strategic mixes makes TS a fertile area for research and empirical study.

1.2 Use of Memory

The memory structures in tabu search operate by reference to four principal dimensions, consisting of recency, frequency, quality, and influence (Figure 1.1). *Recency-based* and *frequency-based* based memory complement each other, and have important characteristics we amplify in later sections. The *quality* dimension refers to the ability to differentiate the merit of

solutions visited during the search. In this context, memory can be used to identify elements that are common to good solutions or to paths that lead to such solutions. Operationally, quality becomes a foundation for incentive-based learning, where inducements are provided to reinforce actions that lead to good solutions and penalties are provided to discourage actions that lead to poor solutions. The flexibility of these memory structures allows the search to be guided in a multi-objective environment, where the goodness of a particular search direction may be determined by more than one function. The tabu search concept of quality is broader than the one implicitly used by standard optimization methods.

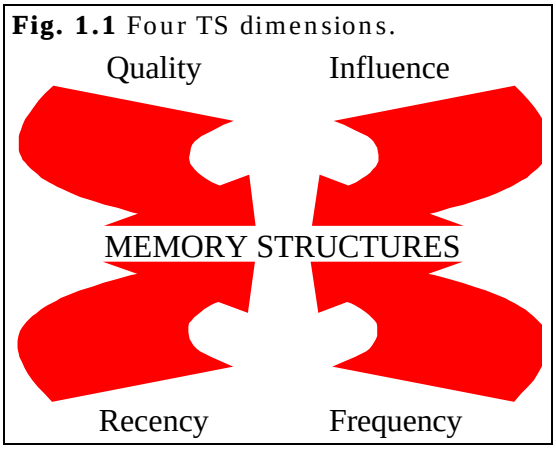


Fig. 1.1 Four TS dimensions.

The fourth dimension, *influence*, considers the impact of the choices made during the search, not only on quality but also on structure. (In a sense, quality may be regarded as a special form of influence.) Recording information about the influence of choices on particular solution elements incorporates an additional level of learning. By contrast, in branch and bound, for example, the separation rules are prespecified and the branching directions remain fixed, once selected, at a given node of a decision tree. It is clear however that certain decisions have more influence than others as a function of the neighborhood of moves employed and the way that this neighborhood is negotiated (e.g., choices near the root of a branch and bound tree are quite influential when using a depth-first strategy). The assessment and exploitation of influence by a memory more flexible than embodied in such tree searches is an important feature of the TS framework.

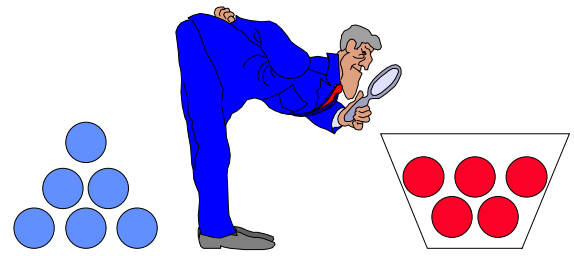
The memory used in tabu search is both *explicit* and *attributive*. Explicit memory records complete solutions, typically consisting of elite solutions visited during the search. An extension of this memory records highly attractive but unexplored neighbors of elite solutions. The memorized elite solutions (or their attractive neighbors) are used to expand the local search.

Alternatively, TS uses attributive memory for guiding purposes. This type of memory records information about solution attributes that change in moving from one solution to another. For example, in a graph or network setting, attributes can consist of nodes or arcs that are added, dropped or repositioned by the moving mechanism. In production scheduling, the index of jobs may be used as attributes to inhibit or encourage the method to follow certain search directions.

1.3 Intensification and Diversification

Two highly important components of tabu search are intensification and diversification strategies. Intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found good. They may also initiate a return to attractive regions to search them more thoroughly. Since elite solutions must be recorded in order to examine their immediate neighborhoods, explicit memory is closely related to the implementation of intensification strategies. As Figure 1.2 illustrates, the main difference between intensification and diversification is that during an intensification stage the search focuses on examining neighbors of elite solutions.

Fig. 1.2 Intensification and diversification.



Here the term "neighbors" has a broader meaning than in the usual context of "neighborhood search." That is, in addition to considering solutions that are adjacent or close to elite solutions by means of standard move mechanisms, intensification strategies generate "neighbors" by either grafting together components of good solution or by using modified evaluation strategies that favor the introduction of such components into a current (evolving) solution. The diversification stage on the other hand encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before. Again, such an approach can be based on generating subassemblies of solution components that are then "fleshed out" to produce full solutions, or can rely on modified evaluations as embodied, for example, in the use of penalty / incentive functions.

Intensification strategies require a means for identifying a set of elite solutions as basis for incorporating good attributes into newly created solutions. Membership in the elite set is often determined by setting a threshold which is connected to the objective function value of the best solution found during the search. However, considerations of clustering and "anti-clustering" are also relevant for generating such a set, and more particularly for generating subsets of solutions that may be used for specific phases of intensification and diversification. In the following sections, we show how the treatment of such concerns can be enhanced by making use of special memory structures. The TS notions of intensification and diversification are beginning to find their way into other meta-heuristics. It is important to keep in mind that these ideas are somewhat different than the old control theory concepts of "exploitation" and "exploration," especially in their implications for developing effective problem solving strategies.

2. Tabu Search Foundations and Short Term Memory

Tabu search can be applied directly to verbal or symbolic statements of many kinds of decision problems, without the need to transform them into mathematical formulations. Nevertheless, it is useful to introduce mathematical notation to express a broad class of these problems, as a basis for describing certain features of tabu search. We characterize this class of problems as that of optimizing (minimizing or maximizing) a function $f(x)$ subject to $x \in \mathbf{X}$, where $f(x)$ may be linear or nonlinear, and the set $\mathbf{X}$ summarizes constraints on the vector of decision variables x. The constraints may include linear or nonlinear inequalities, and may compel all or some components of x to receive discrete values. While this representation is useful for discussing a number of problem solving considerations, we emphasize again that in many applications of combinatorial optimization, the problem of interest may not be easily formulated as an objective function subject to a set of constraints. The requirement $x \in \mathbf{X}$, for example, may specify logical conditions or interconnections that would be cumbersome to formulate mathematically, but may be better be left as verbal stipulations that can be then coded as rules.

Tabu search begins in the same way as ordinary local or neighborhood search, proceeding iteratively from one point (solution) to another until a chosen termination criterion is satisfied. Each $x \in \mathbf{X}$ has an associated neighborhood $\mathbf{N}(x) \subset \mathbf{X}$, and each solution $x' \in \mathbf{N}(x)$ is reached from x by an operation called a *move*.

As an initial point of departure, we may contrast TS with a simple descent method where the goal is to minimize $f(x)$ (or a corresponding ascent method where the goal is to maximize $f(x)$). Such a method only permits moves to neighbor solutions that improve the current objective function value and ends when no improving solutions can be found. A pseudo-code of a generic descent method is presented in Figure 2.1. The final x obtained by a descent method is called a local optimum, since it is at least as good or better than all solutions in its neighborhood. The evident shortcoming of a descent method is that such a local optimum in most cases will not be a global optimum, i.e., it usually will not minimize $f(x)$ over all $x \in \mathbf{X}$.

Fig. 2.1 Descent method.

1) Choose $x \in \mathbf{X}$ to start the process.
2) Find $x' \in \mathbf{N}(x)$ such that $f(x') < f(x)$.
3) If no such x' can be found, x is the local optimum and the method stops.
4) Otherwise, designate x' to be the new x and go to 2).

The version of a descent method called *steepest descent* scans the entire neighborhood of x in search of a neighbor solution x' that gives a smallest $f(x')$ value over $x' \in \mathbf{N}(x)$. Steepest descent implementations of some types of solution approaches (such as certain path augmentation algorithms in networks and matroids) are guaranteed to yield globally optimal solutions for the problems they are designed to handle, while other forms of descent may terminate with local optima that are not global optima. In spite of this attractive feature, in certain settings steepest descent is sometimes impractical because it is computationally too expensive, as where $\mathbf{N}(x)$ contains many elements or each element is costly to retrieve or

evaluate. Still, it is often valuable to choose an x' at each iteration that yields a "good" if not smallest $f(x')$ value.

2.1 Candidate List Strategies

The relevance of choosing good solutions from current neighborhoods is magnified when the guidance mechanisms of tabu search are introduced to go beyond the locally optimal termination point of a descent method. Thus, an important first level consideration for tabu search is to determine an appropriate *candidate list strategy* for narrowing the examination of elements of **N**(x), in order to achieve an effective tradeoff between the quality of x¢and the effort expended to find it. Here quality may involve considerations beyond those narrowly reflected by the value of $f(x')$. If a neighborhood space is totally random, then of course nothing will work better than a totally random choice. (In such a case there is no merit in trying to devise an effective solution procedure.) Assuming that neighborhoods can be identified that are reasonably meaningful for a given class of problems, the challenge is to define solution quality appropriately so that evaluations likewise will have meaning. By the TS orientation, the ability to use history in creating such evaluations then becomes important for devising effective methods. This is true as well in the use of candidate list strategies. Several key forms of such strategies are highlighted in Glover and Laguna (1997).

3. Memory and Tabu Classifications

An important distinction in TS arises by differentiating between short term memory and longer term memory. Each type of memory is accompanied by its own special strategies. However, the effect of both types of memory may be viewed as modifying the neighborhood **N**(x) of the current solution x. The modified neighborhood, which we denote by **N***(x), is the result of maintaining a selective history of the states encountered during the search.

In the TS strategies based on short term considerations, **N***(x) characteristically is a subset of **N**(x), and the tabu classification serves to identify elements of **N**(x) excluded from **N***(x). In TS strategies that include longer term considerations, **N***(x) may also be expanded to include solutions not ordinarily found in **N**(x). Characterized in this way, TS may be viewed as a dynamic neighborhood method. This means that the neighborhood of x is not a static set, but rather a set that can change according to the history of the search. This feature of a dynamically changing neighborhood also applies to the consideration of selecting different component neighborhoods from a *compound* neighborhood that encompasses multiple types or levels of moves, and provides an important basis for parallel processing. Characteristically, a TS process based strictly on short term strategies may allow a solution x to be visited more than once, but it is likely that the corresponding reduced neighborhood **N***(x) will be different each time. With the inclusion of longer term considerations, the likelihood of duplicating a previous neighborhood upon revisiting a solution, and more generally of making choices that repeatedly visit only a limited subset of **X**, is all but nonexistent. From a practical standpoint, the method will characteristically identify an optimal or near optimal solution long before a substantial portion of **X** is examined.

A crucial aspect of TS involves the choice of an appropriate definition of **N***(x). Due to the exploitation of memory, **N***(x) depends upon the trajectory followed in moving from one solution to the next (or upon a collection of such trajectories in a parallel processing environment).

The approach of storing complete solutions (*explicit* memory) generally is used in a highly selective manner, because it can consume an enormous amount of space and time when applied to each solution generated. A scheme that emulates this approach with limited memory requirements is given by the use of hash functions. (Also, explicit memory has a valuable role when selectively applied in strategies that record and analyze certain "special" solutions.) Regardless of the implementation details, short term and long term memory functions provide important cornerstones of the TS methodology. These functions characteristically make use of recency-based memory and frequency-based memory. For purposes of illustration, due to the limited space available in this article, we will restrict our discussion to short term recency-based memory.

3.1 Recency-Based Memory

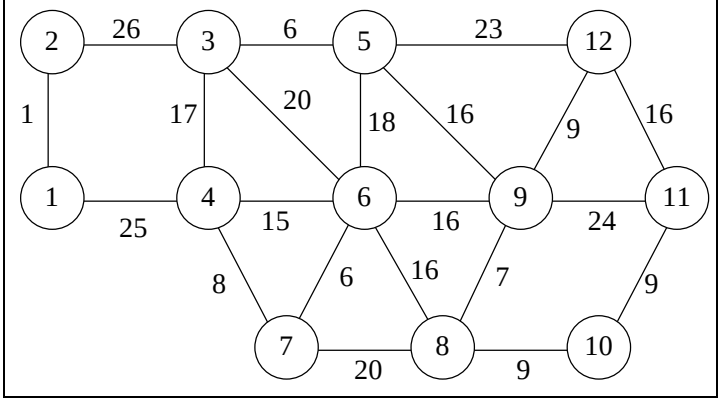
Recency-based memory, as its name suggests, keeps track of solutions attributes that have changed during the recent past. (A liberal definition of "recent" also allows such memory to have a role in longer term strategies.)

To exploit this memory, selected attributes that occur in solutions recently visited are labeled *tabu-active*, and solutions that contain tabu-active elements, or particular combinations of these attributes, are those that become tabu. This prevents certain solutions from the recent past from belonging to $\mathbf{N}^*(x)$ and hence from being revisited. Other solutions that share such tabu-active attributes are also similarly prevented from being visited. Note that while the tabu classification strictly refers to solutions that are forbidden to be visited, by virtue of containing tabu-active attributes (or more generally by violating certain restriction based on these attributes), we also often refer to moves that lead to such solutions as being tabu. We illustrate these points with the following example.

Minimum k-Tree Problem Example

The *Minimum k-Tree* problem seeks a tree consisting of k edges in a graph so that the sum of the weights of these edges is minimum (Lokketangen, et al. 1994). An instance of this problem is given in Figure 3.1, where nodes are shown as numbered circles, and edges are shown as lines that join pairs of nodes (the two "endpoint" nodes that determine the edge). Edge weights are shown as the numbers attached to these lines. A tree is a set of edges that contains no cycles, i.e., that contains no paths that start and end at the same node (without retracing any edges).

Fig. 3.1 Weighted undirected graph.



Assume that the move mechanism is defined by edge-swapping, as subsequently described, and that a greedy procedure is used to find an initial solution. The greedy construction starts by choosing the edge (*i*, *j*) with the smallest weight in the graph, where *i* and *j* are the indexes of the nodes that are the endpoints of the edge. The remaining *k*-1 edges are chosen successively to minimize the increase in total weight at each step, where the edges considered meet exactly one node from those that are endpoints of edges previously chosen. For $k = 4$, the greedy construction performs the steps in Table 3.1.

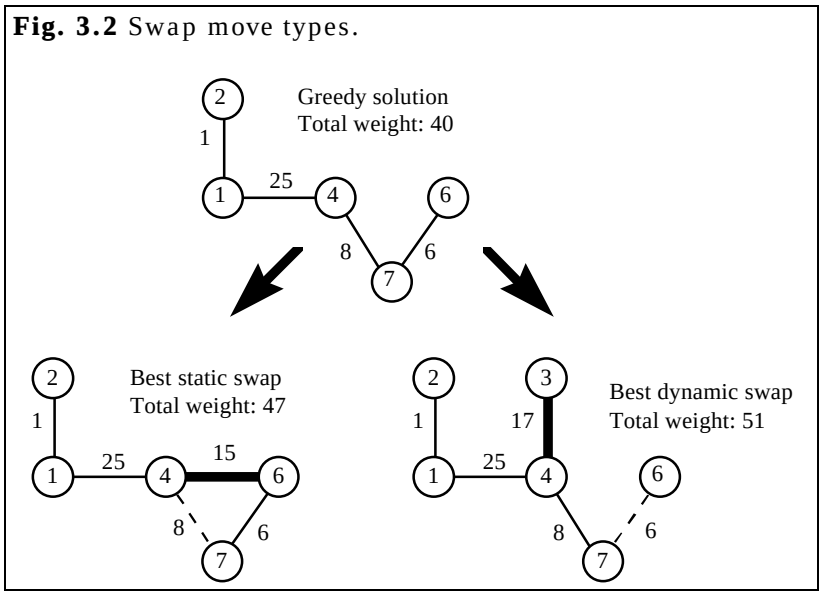
Table 3.1 Greedy construction.

Step	Candidates	Selection	Total Weight
1	(1,2)	(1,2)	1
2	(1,4), (2,3)	(1,4)	26
3	(2,3), (3,4), (4,6), (4,7)	(4,7)	34
4	(2,3), (3,4), (4,6), (6,7), (7,8)	(6,7)	40

The construction starts by choosing edge (1,2) with a weight of 1 (the smallest weight of any edge in the graph). After this selection, the candidate edges are those that connect the nodes in the current partial tree with those nodes not in the tree (i.e., edges (1,4) and (2,3)). Since edge (1,4) minimizes the weight increase, it is chosen to be part of the partial solution. The rest of the selections follow the same logic, and the construction ends when the tree consists of 4 edges (i.e., the value of *k*). The initial solution in this particular case has a total weight of 40.

The swap move mechanism, which is used from this point onward, replaces a selected edge in the tree by another selected edge outside the tree, subject to requiring that the resulting subgraph is also a tree. There are actually two types of such edge swaps, one that maintains the current nodes of the tree unchanged (static) and one that results in replacing a node of the tree by a new node (dynamic). Figure 3.2 illustrates the best swap of each type that can be made starting from the greedy solution. The added edge in each case is shown by a heavy line and the dropped edge is shown by a dotted line.

The best move of both types is the static swap of Figure 3.3, where for our present illustration we are defining *best* solely in terms of the change on the objective function value. Since this best move results in an increase of the total weight of the current solution, the execution of such move abandons the rules of a descent approach and sets the stage for a tabu search process. (The feasibility restriction that requires a tree to be produced at each step is particular to this illustration, since in general the TS methodology may include search trajectories that violate various types of feasibility conditions.)



Given a move mechanism, such as the swap mechanism we have selected for our example, the next step is to choose the key attributes that will be used for the tabu classification. Tabu search is very flexible at this stage of the design. Problem-specific knowledge can be used as guidance to settle on a particular design. In problems where the moves are defined by adding and deleting elements, the labels of these elements can be used as the attributes for enforcing tabu status. Here, in the present example, we can simply refer to the edges as attributes of the move, since the condition of being *in* or *out of the tree* (which is a distinguishing property of the current solution) may be assumed to always be automatically known by a reasonable solution representation.

Choosing Tabu Classifications

Tabu classifications do not have to be symmetric, that is, the tabu structure can be designed to treat added and dropped elements differently. Suppose for example that after choosing the static swap of Figure 3.2, which adds edge (4,6) and drops edge (4,7), a tabu status is assigned to both of these edges. Then one possibility is to classify both of these edges tabu-active for the same number of iterations. The tabu-active status has different meanings depending on whether the edge is added or dropped. For an added edge, tabu-active means that this edge is not allowed to be dropped from the current tree for the number of iterations that defines its tabu tenure. For a dropped edge, on the other hand, tabu-active means the edge is not allowed to be included in the current solution during its tabu tenure. Since there are many more edges outside the tree than in the tree, it seems reasonable to implement a tabu structure that keeps a recently dropped edge tabu-active for a longer period of time than a recently added edge. Notice also that for this problem the tabu-active period for added edges is bounded by k, since if no added edge is allowed to be dropped for k iterations, then within k steps all available moves will be classified tabu.

The concept of creating asymmetric tabu classifications can be readily applied to settings where add/drop moves are not used.

Illustrative Tabu Classifications for the Min k-Tree Problem

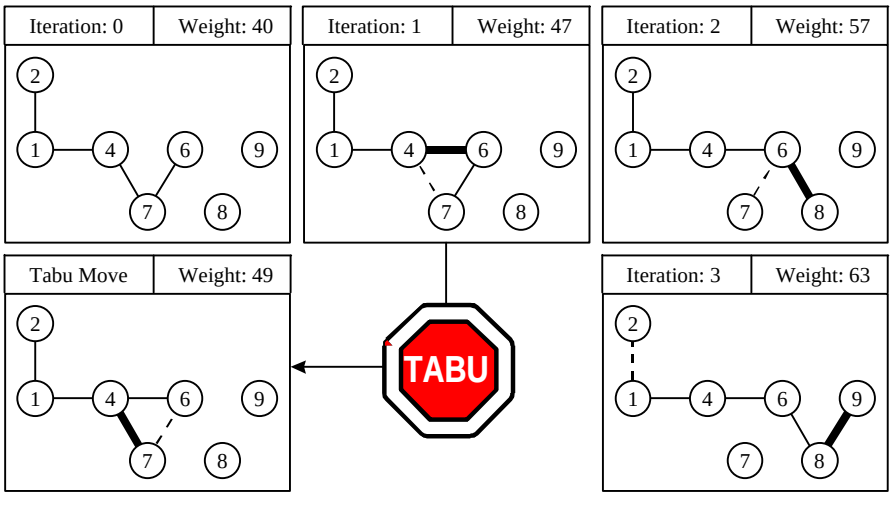
As previously remarked, the tabu-active classification may in fact prevent the search from visiting solutions that have not been examined yet. We illustrate this phenomenon as follows. Suppose that in the Min *k*-Tree problem instance of Figure 3.1, dropped edges are kept tabu-active for 2 iterations, while added edges are kept tabu-active for only one iteration. (The number of iterations an edge is kept tabu-active is called the *tabu tenure* of the edge.) Also assume that we define a swap move to be tabu if either its added or dropped edge is tabu-active. If we examine the full neighborhood of available edge swaps at each iteration, and always choose the best that is not tabu, then the first three moves are as shown in Table 3.2 below (starting from the initial solution found by the greedy construction heuristic). The move of iteration 1 is the static swap move previously identified in Figure 2.3. Diagrams showing the successive trees generated by these moves, starting with the initial greedy solution, are given in Figure 3.3.

Table 3.2 TS iterations.

Iteration	Tabu-active net tenure		Add	Drop	Weight
	1	2			
1			(4,6)	(4,7)	47
2	(4,6)	(4,7)	(6,8)	(6,7)	57
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	63

The net tenure values of 1 and 2 in Table 3.2 for the currently tabu-active edges indicate the number of iterations that these edges will remain tabu-active (including the current iteration).

Fig. 3.3 Effects of attributive short term memory.



At iteration 2, the reversal of the move of iteration 1 (that is, the move that now adds (4,7) and drops (4,6)) is clearly tabu, since both of its edges are tabu-active at iteration 2. In addition, the move that adds (4,7) and drops (6,7) is also classified tabu, because it contains the tabu-

active edge (4,7) (with a net tenure of 2). This move leads to a solution with a total weight of 49, a solution that clearly has not been visited before (see Figure 3.3). The tabu-active classification of (4,7) has modified the original neighborhood of the solution at iteration 2, and has forced the search to choose a move with an inferior objective function value (i.e., the one with a total weight of 57). In this case, excluding the solution with a total weight of 49 has little effect on the quality of the best solution found (since we have already obtained one with a weight of 40).

In other situations, however, additional precautions must be taken to avoid missing good solutions. These strategies are known as aspiration criteria. Alternative forms of aspiration criteria are very important in tabu search. For the moment we observe simply that if the tabu solution encountered at the current step instead had a weight of 39, which is better than the best weight of 40 so far seen, then we would allow the tabu classification of this solution to be overridden and consider the solution admissible to be visited. The aspiration criterion that applies in this case is called the *improved-best* aspiration criterion. (It is important to keep in mind that aspiration criteria do not compel particular moves to be selected, but simply make them available, or alternately rescind evaluation penalties attached to certain tabu classifications.)

One other comment about tabu classification deserves to be made at this point. In our preceding discussion of the Min *k*-Tree problem we consider a swap move tabu if either its added edge or its dropped edge is tabu-active. However, we could instead stipulate that a swap move is tabu only if both its added and dropped edges are tabu-active. In general, the tabu status of a move is a function of the tabu-active attributes of the move (i.e., of the new solution produced by the move).

3.2 A First Level Tabu Search Approach

We now have on hand enough ingredients for a first level tabu search procedure. Such a procedure is sometimes implemented in an initial phase of a TS development to obtain a preliminary idea of performance and calibration features, or simply to provide a convenient staged approach for the purpose of debugging solution software. While this naive form of a TS method omits a number of important short term memory considerations, and does not yet incorporate longer term concerns, it nevertheless gives a useful starting point for demonstrating several basic aspects of tabu search.

We start from the solution with a weight of 63 as shown previously in Figure 3.3, which was obtained at iteration 3. At each step we select the least weight non-tabu move from those available, and use the improved-best aspiration criterion to allow a move to be considered admissible in spite of leading to a tabu solution. The reader may verify that the outcome leads to the series of solutions shown in Table 3.3, which continues from iteration 3, just executed. For simplicity, we select an arbitrary stopping rule that ends the search at iteration 10.

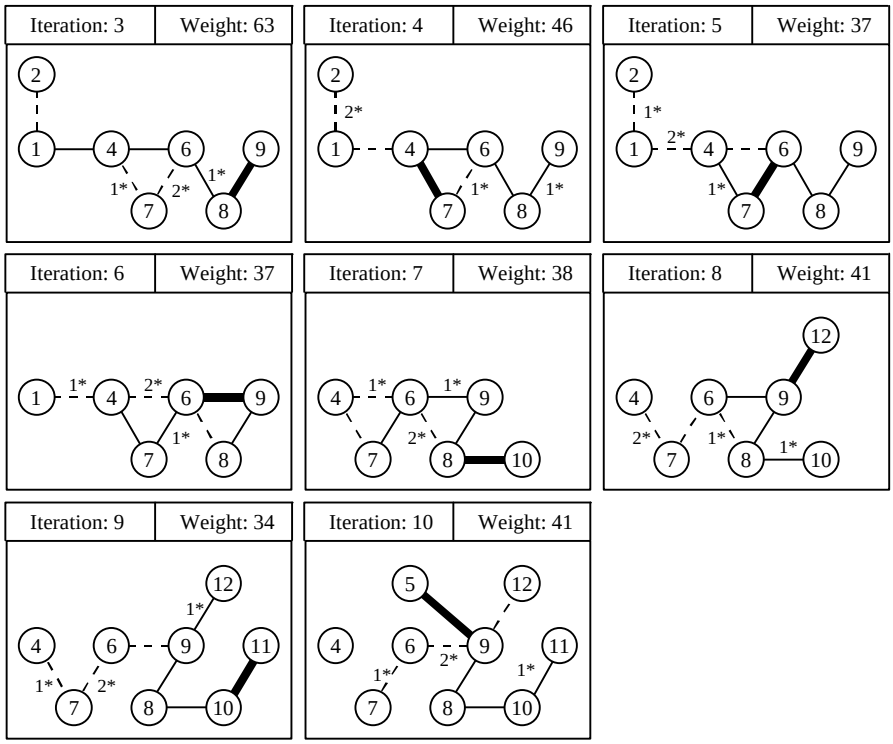
Table 3.3 Iterations of a first level TS procedure.

Iteration	Tabu-active net tenure		Add	Drop	Move Value	Weight
	1	2				
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	6	63
4	(6,7), (8,9)	(1,2)	(4,7)	(1,4)	-17	46
5	(1,2), (4,7)	(1,4)	(6,7)	(4,6)	-9	37*
6	(1,4), (6,7)	(4,6)	(6,9)	(6,8)	0	37
7	(4,6), (6,9)	(6,8)	(8,10)	(4,7)	1	38
8	(6,8), (8,10)	(4,7)	(9,12)	(6,7)	3	41
9	(4,7), (9,12)	(6,7)	(10,11)	(6,9)	-7	34*
10	(6,7), (10,11)	(6,9)	(5,9)	(9,12)	7	41

The successive solutions identified in Table 3.3 are shown graphically in Figure 3.4 below. In addition to identifying the dropped edge at each step as a dotted line, we also identify the dropped edge from the immediately preceding step as a dotted line which is labeled 2^*, to indicate its current net tabu tenure of 2. Similarly, we identify the dropped edge from one further step back by a dotted line which is labeled 1^*, to indicate its current net tabu tenure of 1. Finally, the edge that was added on the immediately preceding step is also labeled 1^* to indicate that it likewise has a current net tabu tenure of 1. Thus the edges that are labeled with tabu tenures are those which are currently tabu-active, and which are excluded from being chosen by a move of the current iteration (unless permitted to be chosen by the aspiration criterion).

As illustrated in Table 3.3 and Figure 3.4 the method continues to generate different solutions, and over time the best known solution (denoted by an asterisk) progressively improves. In fact, it can be verified for this simple example that the solution obtained at iteration 9 is optimal. (In general, of course, there is no known way to verify optimality in polynomial time for difficult discrete optimization problems, i.e., those that fall in the class called NP-hard. The Min *k*-Tree problem is one of these.)

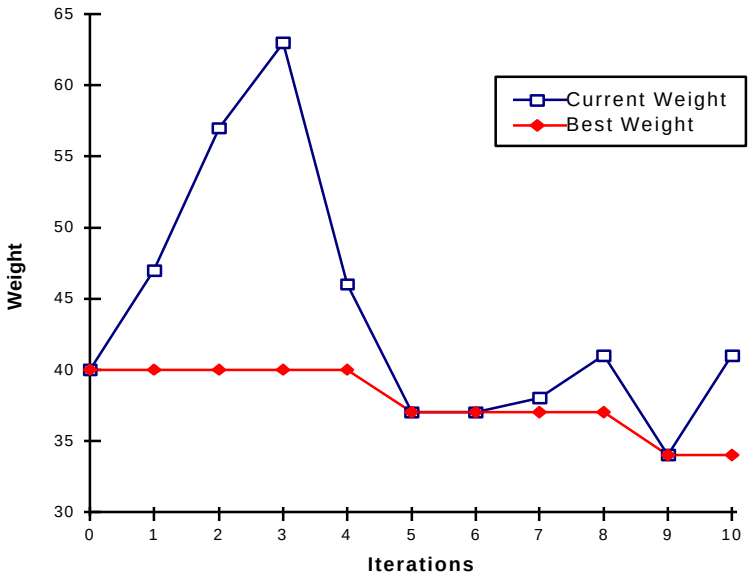
Fig. 3.4 Graphical representation of TS iterations.



It may be noted that at iteration 6 the method selected a move with a move value of zero. Nevertheless, the configuration of the current solution changes after the execution of this move, as illustrated in Figure 3.4.

The selection of moves with certain move values, such as zero move values, may be strategically controlled, to limit their selection as added insurance against cycling in special settings. We will soon see how considerations beyond this first level implementation can lead to an improved search trajectory, but the non-monotonic, gradually improving, behavior is characteristic of TS in general. Figure 3.5 provides a graphic illustration of this behavior for the current example.

Fig. 3.5 TS search trajectory.



We have purposely chosen the stopping iteration to be small to illustrate an additional relevant feature, and to give a foundation for considering certain types of longer term considerations. One natural way to apply TS is to periodically discontinue its progress, particularly if its rate of finding new best solutions falls below a preferred level, and to restart the method by a process designated to generate a new sequence of solutions.

Classical restarting procedures based on randomization evidently can be used for this purpose, but TS often derives an advantage by employing more strategic forms of restarting. We illustrate a simple instance of such a restarting procedure, which also serves to introduce a useful memory concept.

Critical Event Memory

Critical Event memory in tabu search, as its name implies, monitors the occurrence of certain *critical events* during the search, and establishes a memory that constitutes an aggregate summary of these events. For our current example, where we seek to generate a new starting solution, a critical event that is clearly relevant is the generation of the previous starting solution. Correspondingly, if we apply a restarting procedure multiple times, the steps of generating all preceding starting solutions naturally qualify as critical events. That is, we would prefer to depart from these solutions in some significant manner as we generate other starting solutions.

Different degrees of departure, representing different levels of *diversification*, can be achieved by defining solutions that correspond to critical events in different ways (and by activating critical event memory by different rules). In the present setting we consider it important that new starting solutions not only differ from preceding starting solutions, but that they also differ from other solutions generated during previous passes. One possibility is to use a blanket approach that considers each complete solution previously generated to represent a critical event. The aggregation of such events by means of critical event memory makes this entirely

practicable, but often it is quite sufficient (and, sometimes preferable) to isolate a smaller set of solutions.

For the current example, therefore, we will specify that the critical events of interest consist of generating not only the starting solution of the previous pass(es), but also each subsequent solution that represents a "local TS optimum," i.e. whose objective function value is better (or no worse) than that of the solution immediately before and after it. Using this simple definition we see that four solutions qualify as critical (i.e., are generated by the indicated critical events) in the first solution pass of our example: the initial solution and the solutions found at iterations 5, 6 and 9 (with weights of 40, 37, 37 and 34, respectively).

Since the solution at iteration 9 happens to be optimal, we are interested in the effect of restarting before this solution is found. Assume we had chosen to restart after iteration 7, without yet reaching an optimal solution. Then the solutions that correspond to critical events are the initial solution and the solutions of iterations 5 and 6. We treat these three solutions in aggregate by combining their edges, to create a subgraph that consists of the edges (1,2), (1,4), (4,7), (6,7), (6,8), (8,9) and (6,9). (Frequency-based memory refines this representation by accounting for the number of times each edge appears in the critical solutions, and allows the inclusion of additional weighting factors.)

To execute a restarting procedure, we penalize the inclusion of the edges of this subgraph at various steps of constructing the new solution. It is usually preferable to apply this penalty process at early steps, implicitly allowing the penalty function to decay rapidly as the number of steps increases. It is also sometimes useful to allow one or more intervening steps after applying such penalties before applying them again.

For our illustration, we will use the memory embodied in the subgraph of penalized edges by introducing a large penalty that effectively excludes all these edges from consideration on the first two steps of constructing the new solution. Then, because the construction involves four steps in total, we will not activate the critical event memory on subsequent construction steps, but will allow the method to proceed in its initial form.

Applying this approach, we restart the method by first choosing edge (3,5), which is the minimum weight edge not in the penalized subgraph. This choice and the remaining choices that generate the new starting solution are shown in Table 3.4.

Table 3.4 Restarting procedure.

Step	Candidates	Selection	Total Weight
1	(3,5)	(3, 5)	6
2	(2,3), (3,4), (3,6), (5,6), (5,9), (5,12)	(5, 9)	22
3	(2,3), (3,4), (3,6), (5,6), (5,12), (6,9), (8,9), (9,12)	(8, 9)	29
4	(2,3), (3,4), (3,6), (5,6), (5,12), (6,8), (6,9), (7,8), (8,10), (9,12)	(8, 10)	38

Beginning from the solution constructed in Table 3.4, and applying the first level TS procedure exactly as it was applied on the first pass, generates the sequence of solutions shown in Table

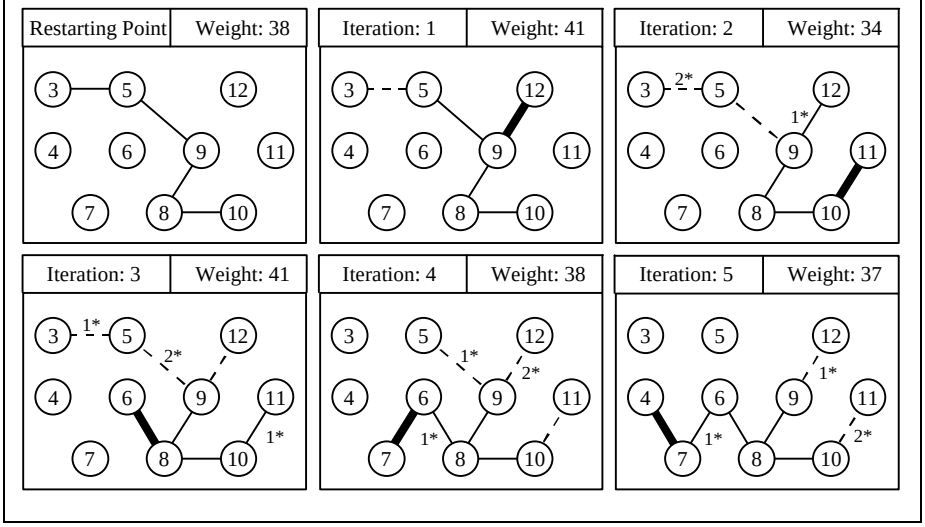
3.5 and depicted in Figure 3.6. (Again, we have arbitrarily limited the total number of iterations, in this case to 5.)

Table 3.5 TS iterations following restarting.

Iteration	Tabu-active net tenure		Add	Drop	Move Value	Weight
	1	2				
1			(9,12)	(3,5)	3	41
2	(9,12)	(3,5)	(10,11)	(5,9)	-7	34*
3	(3,5), (10,11)	(5,9)	(6,8)	(9,12)	7	41
4	(5,9), (6,8)	(9,12)	(6,7)	(10,11)	-3	38
5	(9,12), (6,7)	(10,11)	(4,7)	(8,10)	-1	37

It is interesting to note that the restarting procedure generates a better solution (with a total weight of 38) than the initial solution generated during the first construction (with a total weight of 40). Also, the restarting solution contains 2 "optimal edges" (i.e., edges that appear in the optimal tree). This starting solution allows the search trajectory to find the optimal solution in only two iterations, illustrating the benefits of applying a critical event memory within a restarting strategy. Related memory structures can also be valuable for strategies that drive the search into new regions by "partial restarting" or by directly continuing a current trajectory (with modified decision rules).

Fig. 3.6 Graphical representation of TS iterations after restarting.



In its complete form, tabu search contains a variety of elements that go beyond these first level concerns, and that open up possibilities for creating more powerful solution approaches. An associated collection of problem-solving principles is emerging that invites fuller exploration as a basis for exploiting structure in diverse contexts. The reader may consult Glover and Laguna (1997) for further information.

Bibliography

Glover, F. and M. Laguna (1997) *Tabu Search*, Kluwer Academic Publishers, Boston. This is the first comprehensive book on tabu search. The material presented in this article was principally adapted from this book.

Lokketangen, A. K. Jornsten and S. Storoy (1994) "Tabu Search within a Pivot and Complement Framework," *International Transactions in Operations Research*, Vol. 1, No. 3, pp. 305-316.

Additional References

Battiti, R. and G. Tecchiolli (1994a) "The Reactive Tabu Search," *ORSA Journal on Computing*, Vol. 6, No. 2, pp. 126-140. A description of a specialized tabu search implementation that seeks to dynamically control search parameters with feedback mechanisms.

Crainic, T. G., M. Toulouse and M. Gendreau (1997) "Toward a Taxonomy of Parallel Tabu Search Heuristics," *INFORMS Journal on Computing*, Vol. 9, No. 1, pp. 61-72. A broad view of alternative designs for parallel tabu search implementations.

Dell'Amico, M. and M. Trubian (1993) "Applying Tabu Search to the Job-Shop Scheduling Problem," *Annals of Operations Research*, Vol. 41, 231-252. A good example of a very effective implementation of tabu search in the context of production scheduling.

Faigle, U. and W. Kern (1992) "Some Convergence Results for Probabilistic Tabu Search," *ORSA Journal on Computing*, Vol. 4, No. 1, pp. 32-38. With some simplifying assumptions, the authors of this article present convergence results for the probabilistic variant of tabu search.

Glover, F. (1977) "Heuristics for Integer Programming Using Surrogate Constraints," *Decision Sciences*, Vol 8, No 1, 156-166. Seminal work on tabu search and scatter search.

Glover, F. (1989) "Tabu Search — Part I," *ORSA Journal on Computing*, Vol. 1, No. 3, pp. 190-206. First comprehensive description of tabu search.

Glover, F. (1990) "Tabu Search — Part II," *ORSA Journal on Computing*, Vol. 2, No. 1, pp. 4-32. The second part of this comprehensive description of tabu search introduces additional mechanisms such as the reverse elimination method.

Glover, F. (1995) "Tabu Thresholding: Improved Search by Nonmonotonic Trajectories," *ORSA Journal on Computing*, Vol. 7, No. 4, pp. 426-442. A description of a specialized form of tabu search known as tabu thresholding.

Rego, C. (1998) "A Subpath Ejection Method for the Vehicle Routing Problem," *Management Science*, Vol. 44, No. 10, pp. 1447-1459. An example of a tabu search implementation that employs ejection chain mechanisms to define effective solution neighborhoods.