# Chapter 9
# Tabu Search

Michel Gendreau and Jean-Yves Potvin

## 9.1 Introduction

Over the last 25 years, hundreds of papers presenting applications of tabu search, a heuristic method originally proposed by Glover (1986), to various combinatorial problems have appeared in the operations research literature (see for example Glover and Laguna 1997; Glover et al. 1993; Jaziri 2008; Pardalos and Resende 2002; Rego and Alidaee 2005). In several cases, the methods described provide solutions very close to optimality and are among the most effective, if not the best, to tackle the difficult problems at hand. These successes have made tabu search extremely popular among those interested in finding good solutions to the large combinatorial problems encountered in many practical settings. Several papers, book chapters, special issues and books have surveyed the rich tabu search literature (a list of some of the most important references is provided at the end). In spite of this abundant literature, there still seem to be many researchers who, while they are eager to apply tabu search to new problem settings, find it difficult to properly grasp the fundamental concepts of the method, its strengths and its limitations, and to come up with effective implementations. The purpose of this chapter is thus to focus on the fundamental concepts of tabu search. Throughout the chapter, two relatively straightforward, yet challenging and relevant, problems will be used to illustrate these concepts: the job shop scheduling problem and the capacitated plant location problem.

M. Gendreau
Département de mathématiques et de génie industriel, École Polytechnique de Montréal and CIRRELT, Montréal, Canada

J.-Y. Potvin (✉)
Département d'informatique et de recherche opérationnelle, Université de Montréal and CIRRELT, Montréal, Canada
e-mail: potvin@iro.umontreal.ca

E.K. Burke and G. Kendall (eds.), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, DOI 10.1007/978-1-4614-6940-7_9,
© Springer Science+Business Media New York 2014

## 9.2 Illustrative Problems

### *9.2.1 The Job-Shop Scheduling Problem*

The job shop scheduling problem is one of the most studied problems in combinatorial optimization and a large number of papers and books deal with the numerous procedures that have been proposed to solve it, including several tabu search implementations. Although a large number of variants are found in the literature (and even more in the real world), the *classical* problem can be stated as follows. We first assume that $n$ jobs must be scheduled on $m$ machines. Each job corresponds to a fixed sequence of $m$ operations, one per machine, where each operation must be processed on a specific machine for a specified duration. Note that the processing order on the machines does not need to be the same from one job to another. Each machine can process at most one operation at a time and, once started, an operation must be completed without interruption. The goal is to assign operations to time slots on the machines in order to minimize the maximum completion time of the jobs, which is also known as the makespan. A solution to this problem can be seen as a set of $m$ permutations of the $n$ jobs, one for each machine, with the associated machine schedules (Anderson et al. 1997).

### *9.2.2 The Capacitated Plant Location Problem*

The capacitated plant location problem is one of the basic problems in location theory. It is encountered in many application settings that involve locating facilities with limited capacity to provide services. The problem can be formally described as follows. A set of customers $I$ with demands $d_i, i \in I$, for some product are to be served from plants located in a subset of sites from a given set $J$ of *potential sites*. For each site $j \in J$, the fixed cost of *opening* the plant at $j$ is $f_j$ and its capacity is $K_j$. The cost of transporting one unit of the product from site $j$ to customer $i$ is $c_{ij}$. The objective is to minimize the total cost, i.e. the sum of the fixed costs for open plants and the transportation costs.

Letting $x_{ij}$ $(i \in I, j \in J)$ denote the quantity shipped from site $j$ to customer $i$ (the $x_{ij}$ are the so-called *flow variables*) and $y_j$ $(j \in J)$ be a 0–1 variable indicating whether or not the plant at site $j$ is open (the $y_j$ are the location variables), the problem can be formulated as the following mathematical program:

$$
\begin{aligned}
\text{Minimize } z = & \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\
\text{subject to: } & \sum_{j \in J} x_{ij} = d_i, && i \in I \\
& \sum_{i \in I} x_{ij} \leq K_j y_j, && j \in J \\
& x_{ij} \geq 0, && i \in I, j \in J \\
& y_j \in \{0,1\}, && j \in J.
\end{aligned}
$$

*Remark 9.1.* For any vector $\tilde{y}$ of location variables, optimal (w.r.t. to this plant configuration) values for the flow variables $x(\tilde{y})$ can be retrieved by solving the associated transportation problem:

$$\begin{aligned} \text{Minimize } z(\tilde{y}) &= \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \\ \text{subject to: } &\sum_{j\in J} x_{ij} = d_i, && i \in I \\ &\sum_{i\in I} x_{ij} \le K_j\tilde{y}_j, && j \in J \\ &x_{ij} \ge 0, && i \in I,\ j \in J. \end{aligned}$$

If $\tilde{y} = y^*$, the optimal location variable vector, the optimal solution to the original problem is simply given by $(y^*, x(y^*))$.

*Remark 9.2.* An optimal solution of the original problem can always be found at an extreme point of the polyhedron of feasible flow vectors defined by the constraints

$$\sum_{j\in J} x_{ij} = d_i,\ i \in I$$

$$\sum_{i\in I} x_{ij} \le K_j,\ j \in J$$

$$x_{ij} \ge 0,\ i \in I,\ j \in J.$$

This property follows from the fact that the capacitated plant location problem can be interpreted as a fixed-charge problem defined in the space of the flow variables. This fixed-charge problem has a concave objective function that always admits an extreme point minimum. The optimal values for the location variables can easily be obtained from the optimal flow vector by setting $y_j$ equal to 1 when $\sum_{i\in I} x_{ij} > 0$, and to 0 otherwise.

## 9.3 Basic Concepts

### *9.3.1 Historical Background*

Before introducing the basic concepts of tabu search, we believe it is useful to go back in time to try to better understand the genesis of the method and how it relates to previous work.

Heuristics, i.e. approximate solution techniques, have been used since the beginnings of operations research to tackle difficult combinatorial problems. With the development of complexity theory in the early 1970s, it became clear that, since most of these problems were indeed NP-hard, there was little hope of ever finding efficient exact solution procedures for them. This realization emphasized the role of heuristics for solving the combinatorial problems that were encountered in real-life

applications and that needed to be tackled, whether or not they were NP-hard. While many different approaches were proposed and experimented with, the most popular ones were based on hill climbing. The latter can roughly be summarized as an iterative search procedure that, starting from an initial feasible solution, progressively improves it by applying a series of local modifications or moves (for this reason, hill climbing is in the family of local search methods). At each iteration, the search moves to an improving feasible solution that differs only slightly from the current one. In fact, the difference between the previous and the new solution amounts to one of the local modifications mentioned above. The search terminates when no more improvement is possible. At this point, we have a local optimum with regard to the local modifications considered by the hill climbing method. Clearly, this is an important limitation of the method: unless one is extremely lucky, this local optimum will often be a fairly mediocre solution. The quality of the solution obtained and computing times are usually highly dependent upon the *richness* of the set of transformations (moves) considered at each iteration.

In 1983, a new heuristic approach called simulated annealing (Kirkpatrick et al. 1983) was shown to converge to an optimal solution of a combinatorial problem, albeit in infinite computing time. Based on analogy with statistical mechanics, simulated annealing could be interpreted as a form of controlled random walk in the space of feasible solutions. The emergence of simulated annealing indicated that one could look for other ways to tackle combinatorial optimization problems and spurred the interest of the research community. In the following years, many other new approaches, mostly based on analogies with natural phenomena, were proposed such as tabu search, ant systems (Dorigo 1992) and threshold methods (Dueck and Scheuer 1990). Together with some older ones, in particular genetic algorithms (Holland 1975), they gained an increasing popularity. Now collectively known under the name of meta-heuristics, a term originally coined by Glover (1986), these methods have become, over the last 25 years, the leading edge of heuristic approaches for solving combinatorial optimization problems.

### *9.3.2 Tabu Search*

In 1986 Fred Glover proposed a new approach, which he called tabu search, to allow hill climbing to overcome local optima. In fact, many elements of this first tabu search proposal, and some elements of later elaborations, had already been introduced (Glover 1977), including short-term memory to prevent the reversal of recent moves, and longer-term frequency memory to reinforce attractive components. The basic principle of tabu search is to pursue the search whenever a local optimum is encountered by allowing non-improving moves; cycling back to previously visited solutions is prevented by the use of memories, called tabu lists, that record the recent history of the search. The key idea to exploit information to guide the search can be linked to the informed search methods proposed in the late 1970s in the field of artificial intelligence (Nilsson 1980). It is important to note that Glover

did not see tabu search as a proper heuristic, but rather as a metaheuristic, i.e. a general strategy for guiding and controlling *inner* heuristics specifically tailored to the problems at hand.

### *9.3.3 Search Space and Neighborhood Structure*

As just mentioned, tabu search extends hill climbing methods. In fact, the basic tabu search can be seen as simply the combination of hill climbing with short-term memories. It follows that the two first basic elements of any tabu search heuristic are the definition of its search space and its neighborhood structure.

The search space is simply the space of all possible solutions that can be considered (visited) during the search. For instance, in the job shop scheduling problem of Sect. 9.2.1, the search space could simply be the set of feasible solutions to the problem, where each point in the search space corresponds to a set of $m$ machine schedules that satisfies all the specified constraints. While in that case the definition of the search space seems quite natural, it is not always so. Consider now the capacitated plant location problem of Sect. 9.2.2: the feasible space involves both integer location and continuous flow variables that are linked by strict conditions; moreover, as already indicated, for any feasible set of values for the location variables, one can fairly easily retrieve optimal values for the flow variables by solving the associated transportation problem. In this context, one could obviously use as a search space the full feasible space; this would involve manipulating both location and flow variables, which is not an easy task. A more attractive search space is the set of feasible vectors of location variables, i.e. feasible vectors in $\{0,1\}^{|J|}$ (where $|J|$ is the cardinality of set $J$), any solution in that space being *completed* to yield a feasible solution to the original problem by computing the associated optimal flow variables. It is interesting to note that these two possible definitions are not the only ones. Indeed, on the basis of Remark 9.2, one could also decide to search instead the set of extreme points of the set of feasible flow vectors, retrieving the associated location variables by simply noting that a plant must be open whenever some flow is allocated to it. In fact, this type of approach was used successfully by Crainic et al. (2000) to solve the fixed-charge multi-commodity network design problem, which is a more general problem that includes the capacitated plant location problem as a special case. It is also important to note that it is not always a good idea to restrict the search space to feasible solutions. In many cases, allowing the search to move to infeasible solutions is desirable, and sometimes necessary (see Sect. 9.4.3 for further details).

Closely linked to the definition of the search space is that of the neighborhood structure. At each iteration of tabu search, the local transformations that can be applied to the current solution, denoted $S$, define a set of neighboring solutions in the search space, denoted $N(S)$ (the neighborhood of $S$). Formally, $N(S)$ is a subset of the search space defined by

$$N(S) = \{\text{solutions obtained by applying a single local transformation to } S\}.$$

In general, for any specific problem at hand, there are many more possible (and even, attractive) neighborhood structures than search space definitions. This follows from the fact that there may be several plausible neighborhood structures for a given definition of the search space. This is easily illustrated on our job shop scheduling problem. In order to simplify the discussion, we assume in the following that the search space is the feasible space.

Simple neighborhood structures for the job shop scheduling problem are obtained by considering the sequence of jobs associated with a machine schedule, where the position of a job in the sequence corresponds to its processing order on the machine. For example, one can move a job at another position in the sequence or interchange the position of two jobs. While these neighborhood structures involve only one or two jobs, the neighborhoods they define contain all the feasible schedules that can be obtained from the current one either by moving any single job at any other position or by interchanging any two jobs. Examining these neighborhoods can thus be fairly demanding. In practice, it is often possible to reduce the computational burden, by identifying a restricted subset of moves that are feasible and can lead to improvements. We refer the interested reader to Vaessens et al. (1996) and Anderson et al. (1997) for a more detailed discussion of these issues.

When different definitions of the search space are considered for a given problem, neighborhood structures will inevitably differ to a considerable degree. This can be illustrated on our capacitated plant location problem. If the search space is defined with respect to the location variables, neighborhood structures will usually involve the so-called Add/Drop and Swap moves that respectively change the status of one site (i.e. either opening a closed facility or closing an open one) and move an open facility from one site to another (this move amounts to performing simultaneously an Add move and a Drop move). If, however, the search space is the set of extreme points associated with feasible flow vectors, these moves become meaningless. One should instead consider moves defined by the application of pivots to the linear programming formulation of the transportation problem, where each pivot operation modifies the flow structure to move the current solution to an adjacent extreme point.

The preceding discussion should have clarified a major point: choosing a search space and a neighborhood structure is by far the most critical step in the design of any tabu search heuristic. It is at this step that one must make the best use of the understanding and knowledge he/she has of the problem at hand.

### *9.3.4 Tabus*

Tabus are one of the distinctive elements of tabu search when compared to hill climbing. As already mentioned, tabus are used to prevent cycling when moving away from local optima through non-improving moves. The key realization here is that when this situation occurs, something needs to be done to prevent the search from tracing back its steps to where it came from. This is achieved by making certain

actions tabu. This might mean not allowing the search to return to a recently visited point in the search space or not allowing a recent move to be reversed. For example, in the job shop scheduling problem, if a job $j$ has been moved to a new position in a machine schedule, one could declare tabu moving that job back to its previous position for some number of iterations (this number is called the tabu tenure of the move).

Tabus are stored in a short-term memory of the search (the tabu list) and usually only a fixed and fairly limited quantity of information is recorded. In any given context, there are several possibilities regarding the recorded information. One could record complete solutions, but this requires a lot of storage and makes it expensive to check whether a potential move is tabu or not; it is therefore seldom used. The most commonly used tabus involve recording the last few transformations performed on the current solution and prohibiting reverse transformations (as in the example above); others are based on key characteristics of the solutions themselves or of the moves.

To better understand how tabus work, let us go back to our reference problems. In the job shop scheduling problem, one could define tabus in several ways. To continue our example where a job $j$ has just been moved from position $p_1$ to position $p_2$, one could declare tabu specifically moving back $j$ to position $p_1$ from position $p_2$ and record this in the short-term memory as the triplet $(j, p_2, p_1)$. Note that this type of tabu will not constrain the search much, but that cycling may occur if $j$ is then moved to another position $p_3$ and then from $p_3$ to $p_1$. A stronger tabu would involve prohibiting moving back $j$ to $p_1$ (without consideration for its current position) and be recorded as $(j, p_1)$. An even stronger tabu would be to disallow moving $j$ at all, and would simply be noted as $(j)$.

In the capacitated plant location problem, tabus on Add/Drop moves should prohibit changing the status of the affected location variable and can be recorded by noting its index. Tabus for Swap moves are more complex. They could be declared with respect to the site where the facility was closed, to the site where the facility was opened, to both locations (i.e. changing the status of both location variables is tabu), or to the specific swapping operation.

Multiple tabu lists can be used simultaneously and are sometimes advisable. For example, in the capacitated plant location problem, if one uses a neighborhood structure that contains both Add/Drop and Swap moves, it might be a good idea to keep a separate tabu list for each type of move.

Standard tabu lists are usually implemented as circular lists of fixed length. It has been shown, however, that fixed-length tabus cannot always prevent cycling, and some authors have proposed varying the tabu list length during the search (Glover 1989, 1990; Skorin-Kapov 1990; Taillard 1990, 1991). Another solution is to randomly generate the tabu tenure of each move within some specified interval. Using this approach requires a somewhat different scheme for recording tabus, which are usually stored as tags in an array. The entries in this array typically record the iteration number until which a move is tabu. More details are provided in Gendreau et al. (1994).

### 9.3.5 Aspiration Criteria

While central to the tabu search method, tabus are sometimes too powerful. They may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the search process. It is thus necessary to use algorithmic devices that will allow one to revoke (cancel) tabus. These are called aspiration criteria. The simplest and most commonly used aspiration criterion, found in almost all tabu search implementations, allows a tabu move when it results in a solution with an objective value better than that of the current best-known solution (since the new solution has obviously not been previously visited). Much more complicated aspiration criteria have been proposed and successfully implemented (see, for example, de Werra and Hertz 1989, and Hertz and de Werra 1991), but they are rarely used. The key rule is that if cycling cannot occur, tabus can be disregarded.

### 9.3.6 A Template for Simple Tabu Search

We are now in the position to give a general template for tabu search, integrating the elements we have seen so far. We suppose that we are trying to minimize a function $f(S)$ (sometimes known as an objective or evaluation function) over some domain and we apply the so-called *best improvement* version of tabu search, i.e. the version in which one chooses at each iteration the best available move (even if this results in an increase in the function $f(S)$). This is the most commonly used version of tabu search.

#### Notation

- $S$, the current solution
- $S^*$, the best-known solution
- $f^*$, value of $S^*$
- $N(S)$, the neighborhood of $S$
- $\tilde{N}(S)$, the *admissible* subset of $N(S)$ (i.e. non-tabu or allowed by aspiration)
- $T$, tabu list.

#### Initialization

Choose (construct) an initial solution $S_0$.
Set $S \leftarrow S_0$, $f^* \leftarrow f(S_0)$, $S^* \leftarrow S_0$, $T \leftarrow \emptyset$.

**Search**

While *termination criterion not satisfied* do

- Select $S$ in $\arg\min_{S' \in \tilde{N}(S)}[f(S')]$;
- If $f(S) < f^*$, then set $f^* \leftarrow f(S)$, $S^* \leftarrow S$
- Record tabu for the current move in $T$ (delete oldest entry if necessary)

Endwhile.

In this algorithm, argmin returns the subset of solutions in $\tilde{N}(S)$ that minimizes $f$.

### *9.3.7 Termination Criteria*

Note that we have not specified in our template a termination criterion. In theory, the search could go on forever, unless the optimal value of the problem at hand is known beforehand. In practice, obviously, the search has to be stopped at some point. The most commonly used stopping criteria in tabu search are

- After a fixed number of iterations (or a fixed amount of CPU time)
- After some number of consecutive iterations without an improvement in the objective function value (the criterion used in most implementations)
- When the objective function reaches a pre-specified threshold value.

### *9.3.8 Probabilistic Tabu Search and Candidate Lists*

Normally, one must evaluate the objective function for every element of the neighborhood $N(S)$ of the current solution. This can be extremely expensive from a computational standpoint. In probabilistic tabu search, only a random sample $N'(S)$ of $N(S)$ is considered, thus significantly reducing the computational overhead. Another attractive feature is that the added randomness can act as an anti-cycling mechanism. This allows one to use shorter tabu lists than would be necessary if a full exploration of the neighborhood was performed. On the negative side, it is possible to miss excellent solutions (see Sect. 9.6.3 for more detail). It is also possible to probabilistically select when to apply tabu criteria.

Another way to control the number of moves examined is by means of candidate list strategies, which provide more strategic ways of generating a useful subset $N'(S)$ of $N(S)$. In fact, the probabilistic approach can be considered to be one instance of a candidate list strategy, and may also be used to modify such a strategy. Failure to adequately address the issues involved in creating effective candidate lists is one of the more conspicuous shortcomings that differentiates a naive tabu

search implementation from one that is more solidly grounded. Relevant designs for candidate list strategies are discussed in Glover and Laguna (1997). We also discuss a useful type of candidate generation approach in Sect. 9.4.4.

## 9.4 Extensions to the Basic Concepts

Simple tabu search as described above can sometimes successfully solve difficult problems, but in most cases, additional elements have to be included in the search strategy to make it fully effective. We now briefly review the most important of these.

### *9.4.1 Intensification*

The idea behind the concept of search intensification is that, as an intelligent human being would probably do, one should explore more thoroughly the portions of the search space that seem *promising* in order to make sure that the best solutions in these areas are found. In general, intensification is based on some intermediate-term memory, such as a recency memory, in which one records the number of consecutive iterations that various *solution components* have been present in the current solution without interruption. For instance, in the capacitated plant location problem, one could record how long each site has had an open facility. A typical approach to intensification is to restart the search from the best currently known solution and to *freeze* (fix) in it the components that seem more attractive. To continue with our capacitated plant location problem, one could freeze a number of facilities in sites that have been often selected in previous iterations and perform a restricted search on the other sites. Another technique that is often used consists of changing the neighborhood structure to one allowing more powerful or more diverse moves. In the capacitated plant location problem, if Add/Drop moves were used, Swap moves could be added to the neighborhood structure. In probabilistic tabu search, one could increase the sample size or switch to searching without sampling. Intensification is used in many tabu search implementations, although it is not always necessary. This is because there are many situations where the normal search process is thorough enough.

### *9.4.2 Diversification*

One of the main problems of all methods based on local search, and this includes tabu search in spite of the beneficial impact of tabus, is that they tend to be too *local* (as their name implies), i.e. they tend to spend most, if not all, of their time in a

restricted portion of the search space. The negative consequence of this fact is that, although good solutions may be obtained, one may fail to explore the most interesting parts of the search space and thus end up with solutions that are still far from the optimal ones. Diversification is an algorithmic mechanism that tries to alleviate this problem by forcing the search into previously unexplored areas of the search space. It is usually based on some form of long-term memory of the search, such as a frequency memory, in which one records the total number of iterations (since the beginning of the search) that various *solution components* have been present in the current solution or have been involved in the selected moves. For instance, in the capacitated plant location problem, one could record the number of iterations during which each site has had an open facility. In the job shop scheduling problem, one could note how many times each job has been moved. In cases where it is possible to identify useful *regions* of the search space, the frequency memory can be refined to track the number of iterations spent in these different regions.

There are two major diversification techniques. The first, called restart diversification, involves introducing a few rarely used components in the current solution (or the best known solution) and restarting the search from this point. In the capacitated plant location problem, one could thus open one or more facilities at locations that have seldom been used up to that point and resume searching from that plant configuration (one could also close facilities at locations that have been used the most frequently). In the job shop scheduling problem, a job that has not occupied a particular position in a machine schedule can be forced to that position. The second diversification method, called continuous diversification, integrates diversification considerations directly into the regular searching process. This is achieved by biasing the evaluation of possible moves by adding to the objective a small term related to component frequencies. An extensive discussion on these two techniques is provided by Soriano and Gendreau (1996). A third way of achieving diversification is strategic oscillation, which is discussed in the next section.

We would like to stress that ensuring proper search diversification is possibly the most critical issue in the design of tabu search heuristics. It should be addressed with extreme care fairly early in the design phase and revisited if the results obtained are not up to expectations.

### *9.4.3 Allowing Infeasible Solutions*

Accounting for all problem constraints in the definition of the search space often restricts the searching process too much and can lead to mediocre solutions. In such cases, constraint relaxation is an attractive strategy, since it creates a larger search space that can be explored with *simpler* neighborhood structures. Constraint relaxation is easily implemented by dropping selected constraints from the search space definition and adding to the objective, weighted penalties for constraint violations. In the capacitated plant location problem, this can be done by allowing solutions with flows that exceed the capacity of one or more plants. This, however, raises

the issue of finding correct weights for constraint violations. An interesting way of circumventing this problem is to use self-adjusting penalties, i.e. weights are adjusted dynamically on the basis of the recent history of the search. Weights are increased if only infeasible solutions were encountered in the last few iterations, and decreased if all recent solutions were feasible; see Gendreau et al. (1994) for further details. Penalty weights can also be modified systematically to drive the search to cross the feasibility boundary of the search space and thus induce diversification. This technique, known as strategic oscillation, was introduced in Glover (1977) and used since in several successful tabu search procedures. An important early variant oscillates among alternative types of moves, hence neighborhood structures, while another oscillates around a selected value for a critical function.

### *9.4.4 Surrogate and Auxiliary Objectives*

There are many problems for which the true objective function is quite costly to evaluate, a typical example being the capacitated plant location problem when one searches the space of location variables. Remember that, in this case, computing the objective value for any potential solution entails solving the associated transportation problem. When this occurs, the evaluation of moves may become prohibitive, even if sampling is used. An effective approach to handle this issue is to evaluate neighbors using a surrogate objective, i.e. a function that is correlated to the true objective, but is less computationally demanding, in order to identify a small set of promising candidates (potential solutions achieving the best values for the surrogate). The true objective is then computed for this small set of candidate moves and the best one selected to become the new current solution. An example of this approach is found in Crainic et al. (1993).

Another frequently encountered difficulty is that the objective function may not provide enough information to effectively drive the search to more interesting areas of the search space. A typical illustration of this situation is observed when the fixed costs for open plants in the capacitated plant location problem are much larger than the transportation costs. In this case, it is indicated to open as few plants as possible. It is thus important to define an auxiliary objective function to orient the search. Such a function must measure in some way the desirable attributes of the solutions. In our example, one could use a function that would favor, for the same number of open plants, solutions with plants having just a small amount of flow, thus increasing the likelihood of closing them in subsequent iterations. It should be noted that developing an effective auxiliary objective is not always easy and may require a lengthy trial and error process. In some other cases, fortunately, the auxiliary objective is obvious for anyone familiar with the problem at hand (for an illustration, see the work of Gendreau et al. 1993).

## 9.5 Promising Areas for Future Applications

The concepts and techniques described in the previous sections are sufficient to design effective tabu search heuristics for many combinatorial problems. Most early tabu search implementations, several of which were extremely successful, relied indeed almost exclusively on these algorithmic components (Friden et al. 1989; Hertz and de Werra 1987; Skorin-Kapov 1990; Taillard 1991). Nowadays, however, most leading-edge research in tabu search makes use of more advanced concepts and techniques. While it is clearly beyond the scope of an introductory tutorial such as this to review this type of advanced material, we would like to give readers some insight into it by briefly describing some current trends. Readers who wish to learn more about this topic should read our survey paper (Gendreau 2002) and some of the other references provided here.

A large part of the recent research in tabu search deals with various techniques for making the search more effective. These include methods for better exploitation of the information that becomes available during search and creating better starting points, as well as more powerful neighborhood operators and parallel search strategies. For more details, see the taxonomy of Crainic et al. (1997), the survey of Cung et al. (2002) and the book of Alba (2005). The numerous techniques for utilizing the information are of particular significance since they can lead to dramatic performance improvements. Many of these rely on elite solutions (the best solutions previously encountered) or on parts of these to create new solutions, the rationale being that *fragments* of excellent solutions are often identified quite early in the search process. However, the challenge is to complete these fragments or to recombine them (Glover 1992; Glover and Laguna 1993, 1997; Rochat and Taillard 1995). Other methods, such as the reactive tabu search of Battiti and Tecchiolli (1994), are aimed at finding ways to move the search away from local optima that have already been visited.

Another important trend is hybridization which is, in fact, a pervasive trend in the whole metaheuristics field (for example, a series of workshops on hybrid metaheuristics has been held each year since 2004, see Blum et al. 2008). Tabu search has been used in conjunction with many different solution approaches such as genetic algorithms (Crainic and Gendreau 1999; Fleurent and Ferland 1996), Lagrangean relaxation (Grünert 2002), constraint programming (Pesant and Gendreau 1999), column generation (Crainic et al. 2000) and neural networks (Ateme-Nguema and Dao 2009; Wang et al. 2009). A whole chapter on this topic is found in Glover and Laguna (1997). Problem-specific information and simple heuristics can also be used in conjunction with different components of tabu search. For example, in Burke et al. (1998), problem-specific heuristics are used to realize diversification.

The literature on tabu search has also started moving away from its traditional application areas (graph theory problems, scheduling, vehicle routing) to new ones: continuous optimization (Rolland 1996), multi-objective optimization (Gandibleux et al. 2000), stochastic programming (Lokketangen and Woodruff 1996), mixed integer programming (Crainic et al. 2000; Lokketangen and Woodruff 1996), real-time

decision problems (Gendreau et al. 2006), etc. These new areas confront researchers with new challenges that, in turn, call for novel and original extensions of the method.

## 9.6 Tricks of the Trade

### *9.6.1 Getting Started*

Newcomers to tabu search, trying to apply the method to a problem that they wish to solve, are often confused about what they need to do to come up with a successful implementation. Basically, they do not know where to start. We believe that the following step-by-step procedure will help and provides a useful framework for getting started.

**A Step-by-Step Procedure**

1. *Read one or two good introductory papers* to gain some knowledge of the concepts and of the vocabulary (see the references provided in *Sources of Additional Information*).
2. *Read several papers describing in detail applications in various areas* to see how the concepts have been actually implemented by other researchers (see the references provided in *Sources of Additional Information*).
3. *Think a lot* about the problem at hand, focusing on the definition of the *search space* and the *neighborhood structure*.
4. *Implement a simple version* based on this search space definition and this neighborhood structure.
5. *Collect statistics* on the performance of this simple heuristic. It is usually useful at this point to introduce a variety of *memories*, such as frequency and recency memories, to really track down what the heuristic does.
6. *Analyze results* and *adjust* the procedure accordingly. It is at this point that one should eventually introduce mechanisms for search intensification and diversification or other intermediate features. Special attention should be paid to *diversification*, since this is often where simple tabu search procedures fail.

### *9.6.2 More Tips*

In spite of carefully following the procedure outlined above, it is possible to end up with a heuristic that produces mediocre results. If this occurs, the following tips may prove useful:

1. If there are *constraints*, consider *penalizing the violation of* them. Letting the search move to infeasible solutions is often necessary in highly constrained problems to allow for a meaningful exploration of the search space (see Sect. 9.4.3).
2. Reconsider the *neighborhood structure* and change it if necessary. Many tabu search implementations fail because the neighborhood structure is too simple. In particular, one should make sure that the chosen neighborhood structure allows for a sensible evaluation of possible moves (i.e. the moves that seem intuitively to move the search in the *right* direction should be the ones that are likely to be selected); it might also be a good idea to introduce a *surrogate objective* (see Sect. 9.4.4) to achieve this.
3. *Collect more statistics*. For example, recording the number and quality of previously visited local optima can be useful to find a good trade-off between intensification and diversification
4. *Follow the execution of the algorithm step by step* on some reasonably sized instances. (For example: Is the algorithm behaving as expected on particular solution configurations? Is the algorithm converging prematurely?)
5. Reconsider *diversification*. As mentioned earlier, this is a critical feature in most tabu search implementations.
6. *Experiment* with *parameter* settings. Many tabu search procedures are extremely sensitive to parameter settings; it is not unusual to see the performance of a procedure dramatically improve after changing the value of one or two key parameters (unfortunately, it is not always obvious to determine which parameters are the key ones in a given procedure).

### *9.6.3 Additional Tips for Probabilistic Tabu Search*

While probabilistic tabu search is an effective way of tackling many problems, it creates difficulties of its own that need to be carefully addressed. The most important of these occurs because, more often than not, the best solutions returned by probabilistic tabu search will not be local optima with respect to the neighborhood structure being used. This is particularly annoying since, when it happens, better solutions can be easily obtained, sometimes even manually. An easy way to address this is to simply perform a local improvement phase (using the same neighborhood operator) from the best found solution at the end of the tabu search itself. One could alternately switch to tabu search without sampling (again from the best found solution) for a short duration before completing the algorithm. A possibly more effective technique is to add, throughout the search, an intensification step without sampling. This will mean that the best solutions available in the various regions of the space explored by the method will be found and recorded. This is similar to the method proposed by Glover and Laguna (1993). They employed special aspiration criteria for allowing the search to reach local optima at useful junctures.

### 9.6.4 Parameter Calibration and Computational Testing

Parameter calibration and computational experiments are key steps in the development of any algorithm. This is particularly true in the case of tabu search, since the number of parameters required by most implementations is fairly large and the performance of a given procedure can vary quite significantly when parameter values are modified. The first step in any serious computational experimentation is to select a good set of benchmark instances (either by obtaining them from other researchers or by constructing them), preferably with some reasonable measure of their difficulty and with a wide range of size and difficulty. This set should be split into two subsets, the first one being used at the algorithmic design and parameter calibration steps, and the second reserved for performing the final computational tests that will be reported in the paper(s) describing the heuristic under development. The reason for doing so is quite simple: when calibrating parameters, one always runs the risk of overfitting, i.e. finding parameter values that are excellent for the instances at hand, but poor in general, because these values provide too good a *fit* (from the algorithmic standpoint) to these instances. Methods with several parameters should thus be calibrated on much larger sets of instances than ones with few parameters to ensure a reasonable degree of robustness. The calibration process itself should proceed in several stages:

1. Perform exploratory testing to find good ranges of parameters. This can be done by running the heuristic with a variety of parameter settings.
2. Fix the value of the parameters that appear to be *robust*, i.e. which do not seem to have a significant impact on the performance of the algorithm.
3. Perform systematic testing for the other parameters. It is usually more efficient to test values for only a single parameter at a time, the others being fixed at what appear to be reasonable values. One must be careful, however, for cross-effects between parameters. For example, assume that value $x_1$ for parameter $p_1$ leads to good results when the other parameters are fixed at their default values, and that value $x_2$ for parameter $p_2$ leads to good results when the other parameters are fixed at their default values. Then, it might happen that value $x_1$ for parameter $p_1$ *and* value $x_2$ for parameter $p_2$ lead to poor results. Where such effects exist, it can be important to jointly test pairs or triplets of parameters, which can be an extremely time-consuming task.

The paper by Crainic et al. (1993) provides a detailed description of the calibration process for a fairly complex tabu search procedure and can be used as a guideline for this purpose.

## 9.7 Conclusions

Tabu search is a powerful algorithmic approach that has been applied with great success to many difficult combinatorial problems. A particularly nice feature of tabu search is that it can quite easily handle the *dirty* complicating constraints that

are typically found in real-life applications. It is thus a really practical approach. It is not, however, a panacea: every reviewer or editor of a scientific journal has seen more than his/her share of failed tabu search heuristics. These failures stem from two major causes: an insufficient understanding of fundamental concepts of the method (and we hope that this tutorial may help in alleviating this shortcoming), but also, more often than not, a crippling lack of understanding of the problem at hand. One cannot develop a good tabu search heuristic for a problem that one does not know well! This is because significant problem knowledge is absolutely vital to perform the most basic steps of the development of any tabu search procedure, namely the choice of a search space and the choice of an effective neighborhood structure. If the search space and/or the neighborhood structure are inadequate, no amount of tabu search expertise will be sufficient to save the day. A last word of caution: to be successful, all metaheuristics need to achieve both depth and breadth in their searching process; depth is usually not a problem for tabu search, which is quite aggressive in this respect (it generally finds pretty good solutions very early in the search), but breadth can be a critical issue. To handle this, it is extremely important to develop an effective diversification scheme.

## Sources of Additional Information

- Good introductory papers on tabu search may be found in Glover and Laguna (1993), Glover et al. (1993), Hertz and de Werra (1991), Hindsberger and Vidal (2000), de Werra and Hertz (1989) and, in French, in Soriano and Gendreau (1997).
- The book by Glover and Laguna (1997) is the ultimate reference on tabu search. Apart from the fundamental concepts of the method, it presents a considerable amount of advanced material, as well as a variety of applications. It is interesting to note that this book contains several ideas applicable to tabu search that yet remain to be fully exploited.
- Two issues of *Annals of Operations Research* devoted respectively to *Tabu Search* (Glover et al. 1993) and *Metaheuristics in Combinatorial Optimization* (Laporte and Osman 1996) provide a good sample of applications of tabu search.
- The books made up from selected papers presented at the Metaheuristics International Conferences (MIC) are also extremely valuable. At this time, the books for the following conferences are available: Breckenridge 1995 (Osman and Kelly 1996), Sophia-Antipolis, 1997 (Voss et al. 1999), Angra dos Reis, 1999 (Ribeiro and Hansen 2002), Porto 2001 (Resende and de Sousa 2004), Kyoto, 2003 (Ibaraki et al. 2005) and Vienna, 2005 (Doerner et al. 2007). A book for the 2009 conference in Hamburg is also planned. Finally, a special issue of *Journal of Heuristics* was devoted to the 2007 conference in Montreal (Crainic et al. 2010).
- Three books of interest have also been published. The first one, edited by Rego and Alidaee (2005), deals with tabu search and scatter search. The one edited by