Chapter 2
Tabu Search

Michel Gendreau and Jean-Yves Potvin

Abstract This chapter presents the fundamental concepts of Tabu Search (TS) in a tutorial fashion. Special emphasis is put on showing the relationships with classical local search methods and on the basic elements of any TS heuristic, namely, the definition of the search space, the neighborhood structure, and the search memory. Other sections cover other important concepts such as search intensification and diversification and provide references to significant work on TS. Recent advances in TS are also briefly discussed.

2.1 Introduction

Over the last 30 years, hundreds of papers presenting applications of Tabu Search (TS), a heuristic method originally proposed by Glover in 1986 [30], to various combinatorial problems have appeared in the operations research literature. In several cases, the methods described provide solutions very close to optimality and are

M. Gendreau
Département de mathématiques et de génie industriel, Polytechnique Montréal, Montreal, QC, Canada

Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Montreal, QC, Canada
e-mail: michel.gendreau@cirrelt.net

J.-Y. Potvin (✉)
Département d'informatique et de recherche opérationnelle, Université de Montréal, Montreal, QC, Canada

Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Montreal, QC, Canada
e-mail: potvin@iro.umontreal.ca

M. Gendreau, J.-Y. Potvin (eds.), *Handbook of Metaheuristics*,
International Series in Operations Research & Management Science 272,
https://doi.org/10.1007/978-3-319-91086-4_2

among the most effective, if not the best, to tackle the difficult problems at hand. These successes have made TS 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 TS literature (a list of some of the most important references is provided in a later section). In spite of this abundant literature, there still seem to be many researchers who, while they are eager to apply TS 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 to address this situation by providing an introduction in the form of a tutorial focusing on the fundamental concepts of TS. Throughout the chapter, a relatively straightforward, yet challenging and relevant, problem will be used to illustrate these concepts: the Classical Vehicle Routing Problem (CVRP). This problem will be introduced in the following section. The remainder of the chapter is organized as follows. The basic concepts of TS, like the search space, neighborhood structure, and short-term tabu lists, are described and illustrated in Sect. 2.3. Intermediate, yet critical, concepts, such as intensification and diversification, are described in Sect. 2.4. This is followed in Sect. 2.5 by a brief discussion of advanced topics in TS, and in Sect. 2.6 by a short list of key references on TS and its applications. Section 2.7 provides practical tips for newcomers struggling with unforeseen problems as they first try to apply TS to their favorite problem. Section 2.8 concludes the chapter with some general advice on the application of TS to combinatorial problems.

2.2 The Classical Vehicle Routing Problem

Vehicle Routing Problems have very important applications in the area of distribution management. As a consequence, they have become some of the most studied problems in the combinatorial optimization literature and a large number of papers and books (see [65], for example) deal with the numerous procedures that have been proposed to solve them. These include several TS implementations that currently rank among the most effective. The Classical Vehicle Routing Problem (CVRP) is the basic variant in that class of problems. It can formally be defined as follows. Let $G = (V, A)$ be a graph where V is the vertex set and A is the arc set. One of the vertices represents the depot at which a fleet of m identical vehicles of capacity Q is based, and the other vertices represent customers that need to be serviced. With each customer vertex v_i are associated a demand q_i and a service time t_i. With each arc (v_i, v_j) of A are associated a cost c_{ij} and a travel time t_{ij}. The CVRP consists in finding a set of routes such that:

- Each route begins and ends at the depot;
- Each customer is visited exactly once by exactly one route;
- The total demand of the customers assigned to each route does not exceed Q;

- The total duration of each route (including travel and service times) does not exceed a specified value L;
- The total cost of the routes is minimized.

A feasible solution for the problem thus consists in a partition of the customers into m groups, each of total demand no larger than Q, that are sequenced to yield routes (starting and ending at the depot) of duration no larger than L. This problem will be used in the following to illustrate how various TS concepts can be applied in practice.

2.3 Basic Concepts

Before introducing the basic concepts of TS, the next subsection first goes back in time to try to better understand the genesis of the method and how it relates to previous work.

2.3.1 Historical Background

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 70s, it became clear that, since most of these problems were 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 one was based on Local Search (LS) improvement techniques. LS can be roughly 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). 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 solutions amounts to one of the local modifications mentioned above). The search terminates when it encounters a local optimum with respect to the transformations that it considers, an important limitation of the method: unless one is extremely lucky, this local optimum is often a fairly mediocre solution. In LS, 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 of the heuristic.

In 1983, the world of combinatorial optimization was shattered by the appearance of a paper [82] where it was shown that a new heuristic approach called Simulated Annealing (SA) could converge to an optimal solution of a combinatorial problem, albeit in infinite computing time. Based on an analogy with statistical mechanics, SA can be interpreted as a form of controlled random walk in the space of feasible solutions. The emergence of SA 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 were proposed, mostly based on analogies with natural phenomena (like TS, Ant Colony Optimization, Particle Swarm Optimization, Artificial Immune Systems) which, together with some older ones, such as Genetic Algorithms [38], gained an increasing popularity. Now collectively known under the name of metaheuristics (a term originally coined by Glover in [30]), these methods have become over the last 20 years the leading edge of heuristic approaches for solving combinatorial optimization problems.

2.3.2 Tabu Search

Building upon some of his previous work, Fred Glover proposed a new approach, which he called Tabu Search, to allow local search methods to overcome local optima [30]. In fact, many elements of this first TS proposal, and some elements of later TS elaborations, were introduced in [29], including short term memory to prevent the reversal of recent moves, and longer term frequency memory to reinforce attractive components. The basic principle of TS is to pursue LS whenever it encounters a local optimum 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, a key idea that can be linked to artificial intelligence concepts. It is also important to remark that Glover did not see TS 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.

2.3.3 Search Space and Neighborhood Structure

As we just mentioned, TS is an extension of classical LS methods. In fact, a basic TS can be seen as simply the combination of LS with short-term memories. It follows that the two first basic elements of any TS heuristic are the definition of its search space and its neighborhood structure.

The search space of an LS or TS heuristic is simply the space of all possible solutions that can be considered (visited) during the search. For instance, in the CVRP example described in Sect. 2.2, 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 vehicles routes satisfying all the specified constraints. While in that case the definition of the search space seems quite natural, it is not always so. In the Capacitated Plant Location Problem (CPLP), for instance, customers must be served from plants located in a subset of potential sites. In this context, one could use the full feasible search space made of binary location variables (a site is open or closed) and continuous flow variables. A more attractive search space, though, is obtained by restricting the search space to the binary location variables, from which the complete solution can be obtained by solving the associated transportation problem to get the optimal flow variables. One could also decide to search for the extreme points of the set of feasible flow variable vectors, retrieving the associated location variables by noting that a plant must be open whenever some flow is allocated to it [17]. 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. 2.4.3 for further details).

Closely linked to the definition of the search space is that of the neighborhood structure. At each iteration of LS or TS, 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 made of all 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 CVRP example that has been the object of several TS implementations. To simplify the discussion, we suppose in the following that the search space is the feasible space. Simple neighborhood structures for the CVRP involve moving at each iteration a single customer from its current route; the selected customer is inserted in the same route or in another route with sufficient residual capacity. An important feature of these neighborhood structures is the way in which insertions are performed: one could use random insertion or insertion at the best position in the target route; alternately, one could use more complex insertion schemes that involve a partial reoptimization of the target route, such as GENI insertions [25]. Before proceeding any further it is important to stress that while we say that these neighborhood structures involve moving a single customer, the neighborhoods they define contain all the feasible route configurations that can be obtained from the current solution by moving any customer and inserting it in the stated fashion. Examining the neighborhood can thus be fairly demanding.

More complex neighborhood structures for the CVRP, such as the λ-interchange [50], are obtained by allowing simultaneously the movement of customers to different routes and the swapping of customers between routes. In [54], moves are defined by ejection chains that are sequences of coordinated movements of customers from one route to another; for instance, an ejection chain of length 3 would involve moving a customer v_1 from route R_1 to route R_2, a customer v_2 from R_2 to route R_3 and a customer v_3 from R_3 to route R_4. Other neighborhood structures involve the swapping of sequences of several customers between routes, as in the Cross-exchange [63]. These types of neighborhoods have seldom been used for the CVRP, but are common in TS heuristics for its time-windows extension, where customers must be visited within a pre-specified time interval. We refer the interested reader to [9, 27] for a more detailed discussion of TS implementations for the CVRP and the Vehicle Routing Problem with Time Windows.

When different definitions of the search space are considered for a given problem, neighborhood structures will inevitably differ to a considerable degree. In the case of the CPLP, alluded to above, if the search space corresponds to the location variables only, one could use operators to change the status of these variables (from open to closed and conversely). If, however, the search space is made of the extreme points of the set of feasible flow variable vectors, one could instead consider moves defined by the application of pivots to the linear programming formulation of the transportation problem to move the current solution to an adjacent extreme point. Thus, choosing a search space and a neighborhood structure is by far the most critical step in the design of any TS 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.

2.3.4 Tabus

Tabus are one of the distinctive elements of TS when compared to LS. As we 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 declaring tabu (disallowing) moves that reverse the effect of recent moves. For instance, in the CVRP example, if customer v_1 has just been moved from route R_1 to route R_2, one could declare tabu moving back v_1 from R_2 to R_1 for some number of iterations (this number is called the tabu tenure of the move). Tabus are also useful to help the search move away from previously visited portions of the search space and thus perform more extensive exploration.

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 specific information that is recorded. 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 problem. In the CVRP, one could define tabus in several ways. To continue our example where customer v_1 has just been moved from route R_1 to route R_2, one could declare tabu specifically moving back v_1 from R_2 to R_1 and record this in the short-term memory as the triplet (v_1, R_2, R_1). Note that this type of tabu will not constrain the search much and that cycling may occur if v_1 is then moved to another route R_3 and then from R_3 to R_1. A stronger tabu would involve prohibiting moving back v_1 to R_1, without consideration for its current route, and be recorded as (v_1, R_1). An even stronger tabu would be to disallow moving v_1 to any other route and would simply be noted as (v_1).

Multiple tabu lists can be used simultaneously and are sometimes advisable. For example, when different types of moves are used to generate the neighborhood, it might be a good idea to keep a separate tabu list for each type. 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 [31, 32, 58, 60, 61]. 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 that are then usually stored as tags in an array (the entries in this array will usually record the iteration number until which a move is tabu; see [25], for more details).

2.3.5 Aspiration Criteria

While central to TS, 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 searching 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, which is found in almost all TS implementations, consists in allowing a move, even if it is tabu, if 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 instance [19, 37]), but they are rarely used. The key rule in this respect is that if cycling cannot occur, tabus can be disregarded.

2.3.6 A Template for Simple Tabu Search

We are now in the position to give a general template for TS, integrating the elements we have seen so far. We suppose that we are trying to minimize a function $f(S)$ over some domain and we apply the so-called best improvement version of TS, i.e., the version in which one chooses at each iteration the best available move (this is the most commonly used version of TS).

Notation

- S, the current solution,
- S^*, the best-known solution,

- f^*, the 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, the 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 $argmin_{S' \in \tilde{N}(S)}[f(S')]$;
    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).
```

2.3.7 Termination Criteria

One may have noticed that we have not specified in our template above 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 TS are:

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

In complex tabu schemes, the search is usually stopped after completing a sequence of phases, the duration of each phase being determined by one of the above criteria.

2.3.8 Probabilistic TS and Candidate Lists

In regular TS, one must evaluate the objective for every element of the neighborhood $N(S)$ of the current solution. This can prove extremely expensive from the computational standpoint. An alternative is to instead consider only a random sample $N'(S)$ of $N(S)$, thus reducing significantly the computational burden. Another attractive feature of this alternative 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 must be noted that, in that case, one may miss excellent solutions (more on this topic in Sect. 2.7.3). Probabilities may also be applied to activating 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 $N(S)$ (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 TS implementation from one that is more solidly grounded. Relevant designs for candidate list strategies are discussed in [35]. We also discuss a useful type of candidate generation approach in Sect. 2.4.4. Another interesting approach for the CVRP is the granular TS [66], where only arcs that are likely to be found in good solutions (i.e., short ones) are considered, thus reducing the size of the underlying graph.

2.4 Intermediate Concepts

Simple TS 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.

2.4.1 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 to make sure that the best solutions in these areas are indeed found. From time to time, one would thus stop the normal searching process to perform an intensification phase. 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 a CVRP application, one could record how long an arc has been used. A typical approach to intensification is to restart the search from the best currently known solution and to fix the components that seem more attractive. To continue the CVRP example, one could fix the arcs that have been used for the largest number of iterations and perform a restricted search on the remaining arcs. Another technique that is often used consists in changing the neighborhood structure to one allowing more powerful or more diverse moves. In the CVRP example, one could therefore allow more complex insertion moves or switch to an ejection chain neighborhood structure [33]. In probabilistic TS, one could increase the sample size or switch to searching without sampling.

Intensification is used in many TS implementations, but it is not always necessary. This is because there are many situations where the search performed by the normal process is thorough enough. There is thus no need to spend time exploring more carefully the portions of the search space that have already been visited, and this time can be used more effectively as we shall see right now.

2.4.2 Diversification

One of the main problems of all methods based on local search approaches, and this includes TS 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 pretty 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 CVRP application, one could note how many times each customer has been moved from its current route. 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 forcing a few rarely used components in the current solution (or the best known solution) and restarting the search from this point. In a CVRP heuristic, customers that have not yet been moved frequently could be forced into new routes. The second diversification method, 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 (see [59] for an extensive discussion on these two techniques). A third way of achieving diversification is strategic oscillation as we will see in the next subsection.

Before closing this subsection, we would like to stress that ensuring proper search diversification is possibly the most critical issue in the design of TS 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.

2.4.3 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. This occurs, for example, in CVRP instances where the route capacity or duration constraints are too tight to allow moving customers effectively between routes. 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. 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, for instance, [25] 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 as early as 1977 in [29] and used since in several successful TS procedures (an important early variant oscillates among different types of moves, hence neighborhood structures, while another oscillates around a selected value for a critical function).

2.4.4 Surrogate and Auxiliary Objectives

There are many problems for which the true objective function is quite costly to evaluate. 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 [16].

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 the variant of the CVRP in which the fleet size is not fixed, but is rather the primary objective (i.e., one is looking for the minimal fleet size allowing a feasible solution). In this problem, except for solutions where a route has only one or a few customers assigned to it, most neighborhood structures will lead to the situation where all elements in the neighborhood score equally with respect to the primary objective (i.e., all allowable moves produce solutions with the same number of vehicles). In such a case, it is absolutely necessary to define an auxiliary objective function to orient the search.

Such a function must measure in some way the desirable attributes of solutions. In our example, one could, for instance, use a function that would favor solutions with routes having just a few customers, thus increasing the likelihood that a route can be totally emptied in a subsequent iteration. It should be noted that coming up with 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 (see [24], for an illustration).

2.5 Advanced Concepts

The concepts and techniques described in the previous sections are sufficient to design effective TS heuristics for many combinatorial problems. Early TS implementations, several of which were extremely successful, relied indeed almost exclusively on these algorithmic components. Modern TS implementations, however, exploit more advanced concepts and techniques. While it is clearly beyond the scope of an introductory tutorial, such as this one, to review this type of advanced material, we would like to give readers some insight into it (readers who wish to learn more about this topic should consider the key references provided in the next section).

Various techniques have been devised for making the search more effective. These include methods for exploiting better the information that becomes available during search and creating better starting points, as well as more powerful neighborhood operators and parallel search strategies (on this last topic, see the advances reported in [3] and the chapter on parallel metaheuristics in this Handbook; for specific implementation examples of TS on CPU-based parallel platforms, see [13, 42], and for GPU-based platforms, see [46, 67]). The numerous techniques for making better use of 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 or elements of excellent solutions are often identified quite early in the searching process, but that the challenge is to complete these fragments or to recombine them [33, 35, 39, 53, 55, 64]. Other methods, such as the Reactive TS [6, 48], attempt to find ways of making the search move away from local optima that have already been visited. An important issue is the general approach for exploiting the search framework provided by TS. Some favor simplicity, that is, a search strategy with only a few parameters and based on simple neighborhood operators, as illustrated by the Unified TS [14, 15, 22]. Others propose complex neighborhood operators, thus leading to large or very large neighborhood searches [1, 2].

Another important research area in TS (this is, in fact, pervasive in the whole metaheuristics field) is hybridization, i.e., using TS in conjunction with other solution approaches such as adaptive large neighborhood search [69], genetic algorithms [41, 45, 47, 49], constraint programming [8, 10, 18, 52] or integer programming techniques (there is a whole chapter on this topic in [35]).

TS has also been successful in domains outside its traditional ones (graph theory problems, scheduling, vehicle routing), for example: continuous optimization [7, 11, 12, 21, 40, 68], multi-criteria optimization [36, 40], stochastic programming [5], mixed integer programming [51, 57], dynamic decision problems [26, 28, 56], etc. These domains confront researchers with challenges that ask for innovative extensions of the method.

2.6 Key References

Readers who wish to read other introductory papers on TS can choose among several ones [23, 31, 34, 37, 62]. The book by Glover and Laguna [35] is the ultimate reference on TS: 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 TS that yet remain to be fully exploited. Also valuable are the books and special issues made up from selected papers presented at the recent Metaheuristics International Conferences (MIC) in 2011 [20], 2013 [44] and 2015 [4]. The last MIC conference was held in Barcelona in 2017 and the conference web site can be accessed at mic2017.upf.edu.

2.7 Tricks of the Trade

Newcomers to TS 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. This section is aimed at providing some help in this regard.

2.7.1 Getting Started

The following step-by-step procedure should provide 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.
2. Read several papers describing in detail applications in various areas to see how the concepts have been actually implemented by other researchers.
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 TS procedures fail.

2.7.2 More Tips

It is not unusual that, in spite of following carefully the preceding procedure, one ends up with a heuristic that nonetheless produces mediocre results. If this occurs, the following tips may prove useful:

1. If there are constraints, consider penalizing 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. 2.4).
2. Reconsider the neighborhood structure and change it if necessary. Many TS implementations fail because the neighborhood structure is too simple. In particular, one should make sure that the chosen neighborhood structure allows for a purposeful 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 to achieve this (see Sect. 2.4).
3. Collect more statistics.
4. Follow the execution of the algorithm step-by-step on some reasonably sized instances.
5. Reconsider diversification. As mentioned earlier, this is a critical feature in most TS implementations.
6. Experiment with parameter settings. Many TS 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).

2.7.3 Additional Tips for Probabilistic TS

While it is an effective way of tackling many problems, probabilistic TS creates problems of its own that need to be carefully addressed. The most important of these is the fact that, more often than not, the best solutions returned by probabilistic TS will not be local optima with respect to the neighborhood structure being used. This is particularly annoying since, in that case, better solutions can be easily obtained, sometimes even manually. An easy way to come around this is to simply perform a local improvement phase (using the same neighborhood operator) from the best found solution at the end of the TS itself. One could alternately switch to TS 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; in this fashion, the best solutions available in the various regions of the search space explored by the method will be found and recorded (similar special aspiration criteria for allowing the search to reach local optima at useful junctures are proposed in [34]).

2.7.4 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 TS, since the number of parameters required by most implementations is fairly large and since 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 run 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 parameters that appear to be robust, i.e., which do not seem to have a significant impact on the performance of the procedure.
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. 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 work in [16] provides a detailed description of the calibration process for a fairly complex TS procedure and can be used as a guideline for this purpose.

2.8 Conclusion

Tabu Search is a powerful algorithmic approach that has been applied with great success to many difficult combinatorial problems. A particularly nice feature of TS is that, like all approaches based on local search, it can quite easily handle 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 TS heuristics. These failures stem from two major causes: an insufficient understanding of fundamental concepts of the method (and we hope that this tutorial will 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 TS heuristic for a problem that he/she does not know well! This is because significant problem knowledge is absolutely required to perform the most basic steps of the development of any TS procedure, namely the choice of a search space and of an effective neighborhood structure. If the search space and/or the neighborhood structure are inadequate, no amount of TS 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 TS, which is quite aggressive in this respect (TS heuristics generally find 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.

References

1. S. Abdullah, S. Ahmadi, E.K. Burke, B. Dror, A. McCollum, Tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem. J. Oper. Res. Soc. **58**, 1494–1502 (2007)
2. R.K. Ahuja, O. Ergun, J.B. Orlin, A.P. Punnen, A survey of very large-scale neighborhood search techniques. Discrete Appl. Math. **123**, 75–102 (2002)
3. E. Alba, G. Luque, S. Nesmachnow, Parallel metaheuristics: recent advances and new trends. Int. Trans. Oper. Res. **20**, 1–48 (2013)
4. L. Amodeo, E.-G., Talbi, F. Yalaoui (eds.), *Recent Developments in Metaheuristics* (Springer International Publishing, Cham, 2018)
5. R. Aringhieri, Solving chance-constrained programs combining tabu search and simulation. Lect. Notes Comput. Sci. **3059**, 30–41 (2004)
6. R. Battiti, G. Tecchiolli, The reactive tabu search. ORSA J. Comput. **6**, 126–140 (1994)
7. R. Battiti, G. Tecchiolli, The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. Ann. Oper. Res. **63**, 151–188 (1996)
8. G. Berbeglia, J.-F. Cordeau, G. Laporte, A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. INFORMS J. Comput. **24**, 343–355 (2012)
9. O. Bräysy, M. Gendreau, Tabu search heuristics for the vehicle routing problem with time windows. TOP **10**, 211–237 (2002)
10. Y. Caseau, F. Laburthe, C. Le Pape, B. Rottembourg, Combining local and global search in a constraint programming environment. Knowl. Eng. Rev. **16**, 41–68 (2001)
11. R. Chelouah, P. Siarry, Tabu Search applied to global optimization. Eur. J. Oper. Res. **123**, 256–270 (2000)

12. R. Chelouah, P. Siarry, A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global optimization of multiminima functions. Eur. J. Oper. Res. **161**, 636–654 (2005)
13. J.-F. Cordeau, M. Maischberger, A parallel iterated tabu search heuristic for vehicle routing problems. Comput. Oper. Res. **39**, 2033–2050 (2012)
14. J.-F. Cordeau, M. Gendreau, G. Laporte, A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks **30**, 105–119 (1997)
15. J.-F. Cordeau, G. Laporte, A. Mercier, A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. **52**, 928–936 (2001)
16. T.G. Crainic, M. Gendreau, P. Soriano, M. Toulouse, A tabu search procedure for multicommodity location/allocation with balancing requirements. Ann. Oper. Res. **41**, 359–383 (1993)
17. T.G. Crainic, M. Gendreau, J.M. Farvolden, Simplex-based tabu search for the multicommodity capacitated fixed charge network design problem. INFORMS J. Comput. **12**, 223–236 (2000)
18. B. de Backer, V. Furnon, P. Shaw, P. Kilby, P. Prosser, Solving vehicle routing problems using constraint programming and metaheuristics. J. Heuristics **6**, 501–523 (2000)
19. D. de Werra, A. Hertz, Tabu search techniques: a tutorial and an application to neural networks. OR Spektrum **11**, 131–141 (1989)
20. L. Di Gaspero, A. Schaerf, T. Stützle (eds.), *Advances in Metaheuristics* (Springer, New York, 2013)
21. A. Duarte, R. Martí, F. Glover, F. Gortazar, Hybrid scatter tabu search for unconstrained global optimization. Ann. Oper. Res. **183**, 95–123 (2011)
22. Z. Fu, R. Eglese, L.Y.O. Li, A unified tabu search algorithm for vehicle routing problems with soft time windows. J. Oper. Res. Soc. **59**, 663–673 (2008)
23. M. Gendreau, J.-Y. Potvin, Tabu search, in *Search Methodologies - Introductory Tutorials in Optimization and Decision Support Techniques*, ed. by E.K. Burke, G. Kendall (Springer, New York, 2014), pp. 243–263
24. M. Gendreau, P. Soriano, L. Salvail, Solving the maximum clique problem using a tabu search approach. Ann. Oper. Res. **41**, 385–403 (1993)
25. M. Gendreau, A. Hertz, G. Laporte, A tabu search heuristic for the vehicle routing problem. Manag. Sci. **40**, 1276–1290 (1994)
26. M. Gendreau, F. Guertin, J.-Y. Potvin, É.D. Taillard, Parallel tabu search for real-time vehicle routing and dispatching. Transp. Sci. **33**, 381–390 (1999)
27. M. Gendreau, G. Laporte, J.-Y. Potvin, Metaheuristics for the capacitated VRP, in *The Vehicle Routing Problem*, ed. by P. Toth, D. Vigo. SIAM Monographs on Discrete Mathematics and Applications (SIAM, Philadelphia, 2002), pp. 129–154
28. M. Gendreau, F. Guertin, J.-Y. Potvin, R. Séguin, Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Transp. Res. C Emerg. Technol. **14**, 157–174 (2006)
29. F. Glover, Heuristics for integer programming using surrogate constraints. Decis. Sci. **8**, 156–166 (1977)
30. F. Glover, Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**, 533–549 (1986)
31. F. Glover, Tabu search - Part I. ORSA J. Comput. **1**, 190–206 (1989)
32. F. Glover, Tabu search - Part II. ORSA J. Comput. **2**, 4–32 (1990)
33. F. Glover, Ejection chains, reference structures and alternating path methods for traveling salesman problems. Discrete Appl. Math. **65**, 223–253 (1996)
34. F. Glover, M. Laguna, Tabu search, in *Modern Heuristic Techniques for Combinatorial Problems*, ed. by C.R. Reeves (Blackwell Scientific, Oxford, 1993), pp. 70–150
35. F. Glover, M. Laguna, *Tabu Search* (Kluwer Academic, Boston, 1997)
36. M.P. Hansen, Tabu search in multiobjective optimisation: MOTS, in *Proceedings of the 13th International Conference on Multiple Criteria Decision Making*, Cape Town (1997), pp. 574–586
37. A. Hertz, D. de Werra, The tabu search metaheuristic: how we used it. Ann. Math. Artif. Intell. **1**, 111–121 (1991)

38. J.H. Holland, *Adaptation in Natural and Artificial Systems* (The University of Michigan Press, Ann Arbor, 1975)
39. L.M. Hvattum, A. Lokketangen, F. Glover, Comparisons of commercial MIP solvers and an adaptive memory (tabu search) procedure for a class of 0-1 integer programming problems. Algorithm. Oper. Res. **7**, 13–20 (2012)
40. D.M. Jaeggi, G.T. Parks, T. Kipouros, P.J. Clarkson, The development of a multi-objective tabu search algorithm for continuous optimisation problems. Eur. J. Oper. Res. **185**, 1192–1212 (2008)
41. S.N. Jat, S. Yang, A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. J. Sched. **14**, 617–637 (2011)
42. J. Jin, T.G. Crainic, A. Lokketangen, A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. Eur. J. Oper. Res. **222**, 441–451 (2012)
43. S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing. Science **220**, 671–680 (1983)
44. H.C. Lau, G.R. Raidl, P. Van Hentenryck (eds.), New developments in metaheuristics and their applications. Special issue. J. Heuristics **22**(4), 359–664 (2016)
45. X. Li, L. Gao, An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int. J. Prod. Econ. **174**, 93–110 (2016)
46. T.V. Luong, L. Loukil, N. Melab, E.-G. Talbi, A GPU-based iterated tabu search for solving the quadratic 3-dimensional assignment problem, in *ACS/IEEE International Conference on Computer Systems and Applications*, Hammamet (2010). https://doi.org/10.1109/AICCSA.2010.5587019
47. T. Lust, J. Teghem, MEMOTS: a memetic algorithm integrating tabu search for combinatorial multiobjective optimization. RAIRO—Oper. Res. **42**, 3–33 (2008)
48. F. Mascia, P. Pellegrini, M. Birattari, T. Stützle, An analysis of parameter adaptation in reactive tabu search. Int. Trans. Oper. Res. **21**, 127–152 (2014)
49. S. Meeran, M.S. Morshed, A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study. J. Intell. Manuf. **23**, 1063–1078 (2012)
50. I.H. Osman, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann. Oper. Res. **41**, 421–451 (1993)
51. J.P. Pedroso, Tabu search for mixed integer programming, in *Metaheuristic Optimization via Memory and Evolution*, ed. by C. Rego, B. Alidaee (Kluwer Academic, Boston, 2005), pp. 247–261
52. G. Pesant, M. Gendreau, A constraint programming framework for local search methods. J. Heuristics **5**, 255–280 (1999)
53. C. Rego, B. Alidaee (eds.), *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search* (Kluwer Academic, Boston, 2005)
54. C. Rego, C. Roucairol, A parallel tabu search algorithm using ejection chains for the vehicle routing problem, in *Meta-Heuristics: Theory and Applications*, ed. by I.H. Osman, J.P. Kelly (Kluwer Academic, Boston, 1996), pp. 661–675
55. Y. Rochat, É.D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing. J. Heuristics **1**, 147–167 (1995)
56. A.G. Roesener, J.W. Barnes, An advanced tabu search approach to the dynamic airlift loading problem. Log. Res. **9**, 12:1–12:18 (2016)
57. L.H. Sacchi, V.A. Armentano, A computational study of parametric tabu search for 0-1 mixed integer program. Comput. Oper. Res. **38**, 464–473 (2011)
58. J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem. ORSA J. Comput. **2**, 33–45 (1990)
59. P. Soriano, M. Gendreau, Diversification strategies in tabu search algorithms for the maximum clique problem. Ann. Oper. Res. **63**, 189–207 (1996)
60. É.D. Taillard, Some efficient heuristic methods for the flow shop sequencing problem. Eur. J. Oper. Res. **47**, 65–74 (1990)
61. É.D. Taillard, Robust taboo search for the quadratic assignment problem. Parallel Comput. **17**, 443–455 (1991)

62. E. Taillard, Tabu search, in *Metaheuristics*, ed. by P. Siarry (Springer International Publishing, Cham, 2016), pp. 51–76
63. É.D. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, A tabu search heuristic for the vehicle routing problem with soft time windows. Transp. Sci. **31**, 170–186 (1997)
64. C.D. Tarantilis, C.T. Kiranoudis, BoneRoute - an adaptive memory-based method for effective fleet management. Ann. Oper. Res. **115**, 227–241 (2002)
65. P. Toth, D. Vigo (eds.), *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications (SIAM, Philadelphia, 2002)
66. P. Toth, D. Vigo, The granular tabu search and its application to the vehicle routing problem. INFORMS J. Comput. **15**, 333–346 (2003)
67. C. Tsotskas, T. Kipouros, A.M. Savill, The design and implementation of a GPU-enabled multi-objective tabu-search intended for real world and high-dimensional applications. Procedia Comput. Sci. **29**, 2152–2161 (2014)
68. G. Waligóra, Simulated annealing and tabu search for discrete-continuous project scheduling with discounted cash flows. RAIRO—Oper. Res. **48**, 1–24 (2014)
69. I. Žulj, S. Kramer, M. Schneider, A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. Eur. J. Oper. Res. **264**, 653–664 (2018)