A Tabu Search with a Double Neighborhood Strategy

Paula Amaral[1,2(✉)], Ana Mendes[1], and J. Miguel Espinosa[3]

[1] Nova SST–FCT Nova, Campus de Caparica, 2829-516 Caparica, Portugal
paca@fct.unl.pt
[2] NovaMaths CMA Nova, Campus de Caparica, 2829-516 Caparica, Portugal
[3] IT & Digital Department, GALP, Lisbon, Portugal

Abstract. Tabu Search (TS) is a well known and very successful method heuristic approach for hard optimization problems, linear or nonlinear. It is known to produce very good solutions, optimal or close to optimal for some hard combinatorial optimization problems. The drawback is that we have in general no optimality certificate, but this is the price to be paid for problems where the exact methods are too costly in terms of time and computational memory. One of the drawbacks of TS is that the strategies must be studied and refined for every instance. Many proposals exist to enhance the efficiency of this method heuristic. TS is particularly fragile in cases where there are many local optima of the problem. TS may be slow in the process of escaping a region of attraction of a local optima. If the strategy for evaluating the solutions in the neighborhood takes time to move away from the local optimum then it may compromise the search efficiency. In this paper we propose a double neighborhood strategy with opposite optimization directions (minimization and maximization). While one search for the best solution in the neighborhood the second search for the worse, and two parallel process develop switching from the minimization to maximization and vice-versa, when in consecutive iterations there is no improvement in solution. With this proposal, it is intended that the research can escape the attraction zone of a local optimum, more quickly allowing the research space to be better explored. We present an application to a Knapsack problem.

Keywords: Tabu Search · Neighborhood search · Double neighborhood

1 Introduction

Tabu Search [8–10,18] is a metaheuristic that has successfully been applied to find good feasible solutions for hard optimization problems [12,17]. There are many successful applications of TS, for instance in timetabling scheduling [3], in Neural Networks [22], Vehicle Routing [5,7,19], as a general solver for the constraint satisfaction problems [14]. In general it can be described as a neighborhood search method incorporating techniques for escaping local optima and avoid cycling [11]. A first level Tabu Search (TS) comprises the following concepts in each iteration:

This work is funded by national funds through the FCT - Fundação para a Ciëncia e a Tecnologia, I.P., under the scope of the project UIDB/00297/2020 (Center for Mathematics and Applications).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13378, pp. 219–230, 2022.
https://doi.org/10.1007/978-3-031-10562-3_16

- Current starting solution - Start search point.
- Search Neighborhood - Points that will be inspected from the current solution.
- Move - A basic operation in the definition of the neighborhood.
- Evaluation - A procedure to evaluate the points in the neighborhood.
- Tabu list - The tabu moves that are not allowed in the current iteration
- Aspiration Criteria - May revoke a tabu status.

A general, very basic, iteration of TS will consist in finding a set of points in the neighborhood of the current point. Evaluate these points and chose the one that has the best evaluation, as long as the move associated to this point is not tabu. If it is tabu we can apply an aspiration criteria or not. Next we add the move, or solution, or a related attribute that generated the best evaluated point to the tabu list. After we proceed to the next iteration from the current point. There are many interesting additional refinements that can greatly increase the performance of TS.

In this paper a new approach for the TS is proposed. We use two parallel searches, with two distinct neighborhood N_1 and N_2, and opposite strategies in every iteration. At every iteration if in N_1 it is applied the minimization direction in N_2 we apply a maximization direction. The search strategy switches if a local optima is recognized. In this way, when a local minimum is found, we escape from that local optima by following the path along the maximization direction. When a local maximum is encountered, the direction of search switches to minimization to explore new local optima.

2 Motivation

What led to the study of this new strategy was the fact that many times the tabu search takes time to move away from the local optimum. With this proposal, it is intended that the search can escape the attraction zone of a local optimum, more quickly allowing the search space to be more explored.

To illustrate this idea more clearly, in Fig. 1 is represented a function whose objective is to determine the global minimum. Considering two neighborhoods, the first uses the classical strategy. In this case it will be selected the solution with the minimum value of the neighborhood and in the second it will alternate between the selection of the maximum and of the minimum value.

In the example, for the first neighborhood the tabu search starts at the initial solution, x_0 and considering the assigned interval as a neighborhood, the best point in the neighborhood is x_1. In the second iteration the best point in the neighborhood of x_1, $N(x_1)$ is the point x_2. This spot is a local minimizer. With this classic strategy, many iterations will be needed to escape this local optimum because, the minimum is chose, so the left point in interval representing the neighborhood is picked. That's why we chose to use two neighborhoods. In addition to the usual strategy of choosing the best

solution in If we want to escape this region of attraction of x_2 we must climb the hill as fast as possible. A second strategy is implemented, which pursues a sequence of worst solutions until a local maximum is found. In addition to the usual strategy of choosing the best solution in Fig. 1, if we switch from the minimum in the neighborhood to the maximum, the point picked in every iteration will be the right-hand-side of the interval. After a few iterations a local maximum is attained. Once a local maximum is reached, the strategy is reversed (minimum) and the best solutions in the neighborhood are chosen. When a local optimum is obtained, in this case a local minimum, the strategy is reversed again, changing the search direction again to the maximum of the objective function. Thus, with two neighborhoods, the second neighborhood, starting from the initial solution x_0, selects the point x_3 as the next center of the next neighborhood. The search continues and in the second iteration it moves to the solution x_4, as a local maximum has been reached, the second neighborhood strategy is inverted and starts to select the best solution in the neighborhood. In the next iteration as you are minimizing the $N(x_4)$ is the point x_5 and so on until reaching the global optimum, x^*. In this example it can be seen that with the use of this strategy the search can converge more quickly to an optimal point. In addition the neighborhood space will be better explored. In the next section we make a brief description of a classic Tabu Search, before presenting the Double Neighborhood Tabu Search approach.

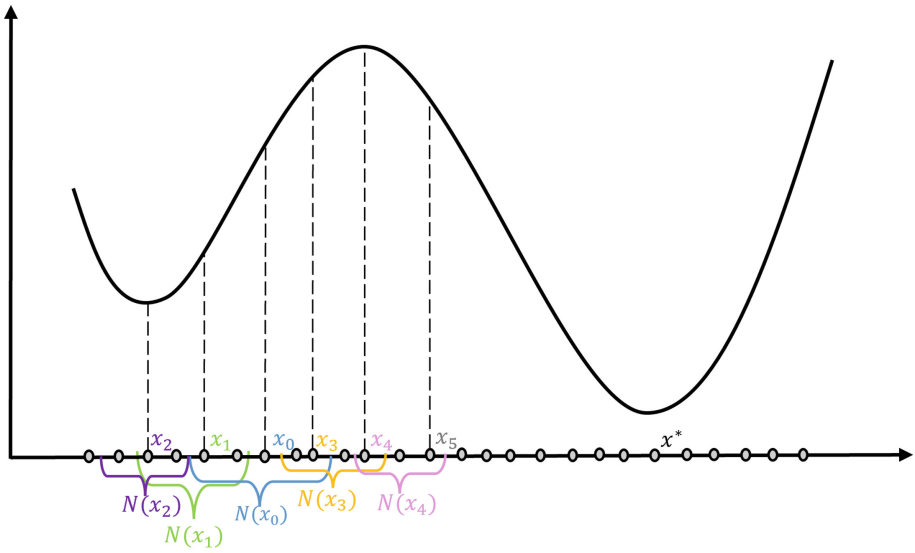

Fig. 1. Example of double neighborhood

3 Tabu Search

The Tabu Search is a metaheuristic that explores the space of solutions, moving at each iteration to the best solution in the neighborhood, not accepting movements that could lead to the solutions already visited. In the Algorithm 1, the main steps of the Tabu Search [17] are presented.

Algorithm 1. Tabu Search

Data: S_0 Initial solution;
Stoping criteria
Result: S^* Best solution found
$T \leftarrow []$;
$S \leftarrow S_0$;
$S^* \leftarrow S$;
repeat
  Generate $N(S)$ neighborhood solutions of S;
  Find best solution $\hat{S}$ in $N(S)$ that is not tabu ;
  if $\hat{S}$ *better than* S^* **then**
    $S^* \leftarrow \hat{S}$
  end
  $S \leftarrow \hat{S}$;
  Updates the tabu list T;
until *Stop Criteria*;

3.1 Initial Solution

To start the Tabu Search, it is necessary to define an initial solution, from which the first search neighborhood is defined. Different strategies can be used to determine this solution, for example, the solution can be generated randomly or determined through a heuristic. In the second case, one of the most used heuristics for their simplicity are those based on *greedy* methods.

Generating a random initial solution becomes faster, however the algorithm may take longer to converge. On the other hand, the use of a heuristic usually leads to a local optimum with a smaller number of iterations. However, it should be noted that this does not mean that in all cases the use of better initial solutions will lead the algorithm to converge to better local optima [21].

3.2 Search Space

The search space is the space that contains all feasible solutions to the problem. In Tabu Search, for nonconvex problems, the idea is to explore the solutions space as comprehensively and as completely as possible.

3.3 Structure of the Neighborhood

In each iteration, a current solution is considered, denoted by S, and the set of neighborhood solutions is defined by $N(S)$. Formally $N(S)$ is a subset of the search space that contains all solutions obtained by a transformation of S according to some rule. For each problem there is more than one possible neighborhood structure that can be applied.

There are also several strategies to select the solution in the neighborhood, some of them being [15,21]:

- Select the best solution in the neighborhood, that is, all solutions in the neighborhood are examined and the solution with the best value for the objective function is chosen. For large neighborhoods, this type of exploration turns out to be quite time-consuming, which implies a great computational effort.
- Select the first admissible solution, this strategy consists of choosing the first solution in the neighborhood that satisfies all the constraints of the problem and is not tabu, so only a partial exploration of the neighborhood is performed.
- Another strategy that can be applied is to randomly select a solution from the neighborhood.

The partial evaluation of the neighborhood may imply a deterioration in the quality of the solution, since not all solutions are analyzed in each iteration. However, at a global level this limitation may end up not having a negative influence on the quality of the solution produced, because a good final solution can still be achieved by exploring regions different than those obtained if the full neighborhood was inspected [20].

The construction of efficient tabu search algorithms implies a good balance between the computational burden of each iteration (more complex neighborhoods with a greater number of solutions to explore) and the number of iterations (in an attempt to better explore all solution spaces). More complex neighborhoods may enrich the research, since more solutions are considered at each iteration, but at the same time there is a risk of the computational effort becoming excessive.

Simple Neighborhood Structure. Suppose that a solution is indexed by a vector with natural numbers. A simple neighborhood is created by simple movements, where at each iteration a single variable switches its position. For instance, in a Vehicle Routing Problem (VPR) a vector represents a sequence of costumers visited in one route. Switching the position of a component of the vector, in this case means to change the order by which the costumers in that route is visited, or even changed to another route. In the example of the Fig. 2, customer 5 was moved, which will be placed in the position before customer 2. This action can be presented as a move, that can be recorder in different ways, for instance by (5), the number of the costumer that changed the position, by the pair (5,6) meaning that costumer 5 moved from position 6, or (5, 6, 2) meaning that costumer 5 moved from position 5 to 2.

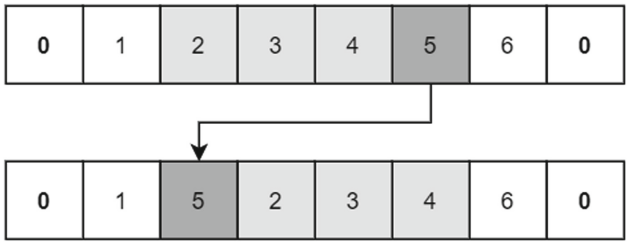

Fig. 2. Example simple neighborhood structure

3.4 Tabu List

To avoid revisiting a previous solution and in order to escape from a local optima, a list representing the moves that are tabu, meaning moves that could not be performed, is defined throughout the algorithm.

For instance, in the previous example, if the tabu $\{5\}$ is defined it means that in a number of sequential iterations, the position of costumer 5 is not allowed to change. If it is defined as tabu the pair (5, 6) means that costumer 5 cannot be moved back to position 6. If (5, 6, 2) is defined as tabu, it implies that costumer 5 cannot move from position 2 to 6. More complex tabu list elements are more interesting but may lead to a more time consuming method.

3.5 Tabu Tenure

The tabu tenure defines the number of iterations during which a certain tabu movement is kept in the tabu list. The value of the tabu tenure may be fixed or may be dynamically defined. One of the drawback of Tabu Search is related to the need of tuning the tabu tenure, whose choice affects the performance of the algorithm. In [16] the authors presented a method to automatically manage the tabu tenure in the TS using a Decision Expert System based on a Fuzzy Inference Rule Based System (FIRBS). Two concepts "frequency" and "inactivity" related to the number of times a move was attempt and the last time it was called were used in the construction of the FIRBS.

3.6 Aspiration Criteria

The aspiration criteria allows to revoke a tabu status under some predefined condition [13].

3.7 Stopping Criteria

In tabu search there are several ways to end the search process, such as:

- After a fixed number of iterations;
- After a certain number of iterations with no improvement in the value of the best solution;

– When the objective function reaches a predefined value. This criterion prevents the execution of the algorithm when a solution is considered sufficiently good.

Several additional features can be incorporated in a TS, such as Aspiration Criteria, Intensification, Diversification, Multistart. In this section we just mentioned the basic features of TS.

4 Double Neighborhood Tabu Search (DNTS)

As already mentioned as an alternative to having just one type of search, with one strategy, we propose considering two neighborhoods, with to independently processes and with opposite strategies (minimization and maximization).

Algorithm 2. Change of Neighborhood Strategy

```
The search starts with the neighborhood maximizing the value of the objective function;
if search is maximizing then
    if a local maximum is reached then
        The neighborhood strategy switches to minimization;
    end
end
if search is minimizing then
    if a local minimum is reached then
        The neighborhood strategy switches to maximization;
    end
end
```

4.1 The Double Neighborhood Algorithm

In the classic approach, the common search, consists in selecting the solution of the neighborhood that presents the better value for the objective function. The DNTS, on the other hand, considers simultaneously two neighborhoods (Neighborhood 1 and Neighborhood 2) and varies between the selection of the best solution from the neighborhood and the worst one, that is, it alternates between a direction of minimization and maximization in order to escape as quickly as possible from a region of the search space corresponding to a local optimum.

At the beginning of the Tabu Search, we start by choosing the worst value of the objective function in neighborhood 2. When a local optimum is reached, that is, when there are only better values in the neighborhood than the current solution, the exchange is made and the search begins to choose the values in the neighborhood. The search changes again when after t iterations there was no improvement in the value obtained

by the neighborhood. In the Algorithm 2 there is a description of how the exchanges in neighborhood search strategy are carried out, for a minimization problem.

The addition of a new neighborhood, implies the definition of new parameters, as for instance the *tabu tenure*, and the number of iterations during which the strategy (minimization or maximization) is maintained. As the neighborhoods are independent, then *tabu tenure* will also be, i.e. the neighborhoods can be defined with different *tabu tenure* values. The way in which the parameter values are determined may vary according to the problem under study, and are defined at the beginning of the algorithm. We should run the program for different values of *tabu tenure*, to fine-tune the parameters to determine the best values of *tabu tenure* for the problem.

4.2 The Knapsack Problem

For this new approach for the Tabu Search, results obtained for a class of optimization problem are now presented. The tests were performed for a classic optimization problem, the Knapsack Problem, which has a simple to formulation and has many interesting practical applications [4], such as optimizing investments, cargo loading [6], capital budgeting [2].

This is a well-known optimization problem and consists of filling a backpack with objects with a certain weight and value. The purpose of the problem is to fill the backpack with objects with the highest possible value, but the weight of the objects cannot exceed the maximum weight supported by the backpack. The mathematical formulation of the problem is the following:

N = Number of objects;
P = Capacity of the Knapsack;
p_i = weight of object i, for $i = 1, \ldots N$
v_i = value of of object i, for $i = 1, \ldots N$

$$x_i = \begin{cases} 1 & \text{if object is placed in the knapsack} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, \ldots N$$

$$\max \sum_{i=1}^{N} v_i x_i$$

$$\text{s.t .}$$

$$\sum_{i=1}^{N} p_i x_i \leq P$$

$$x_i \in \{0, 1\} \text{ for } i = 1, \ldots N$$

Algorithm 3. Tabu Search with Double Neighborhood for a minimization problem

Data: S_0 Initial solution;
Stopping criteria
Result: S^*, fo^* Best solution and best value found
$T_1 \leftarrow []$;
$T_2 \leftarrow []$;
$S_1 \leftarrow S_0$;
$S_2 \leftarrow S_0$;
$Mode_1 = \min$;
$Mode_2 = \max$;
// If $Mode_k = \min$ then search for the solution in the neighborhhod with minimum objective function value
// If $Mode_k = \max$ then search for the solution in the neighborhhod with maximum objective function value
$fo^* = v(S_0)$; // $v(S)$ is the objective function value of S
$S^* \leftarrow S_0$; // S^* is the incumbent solution
;
repeat
  Generate $N_1(S_1)$ neighborhood solutions of S_1;
  Find solution corresponding to $Mode_1$, $\hat{S_1}$ in $N(S_1)$ that is not tabu ;
  Generate $N_2(S_2)$ neighborhood solutions of S_2;
  Find solution corresponding to $Mode_2$, $\hat{S_2}$ in $N(S_2)$ that is not tabu ;
  if $v(\hat{S_1}) < fo^*$ **then**
    $S^* \leftarrow \hat{S_1}$;
    $fo^* = v(\hat{S_1})$; // Update the incumbent solution
  end
  if $v(\hat{S_2}) < fo^*$ **then**
    $S^* \leftarrow \hat{S_2}$;
    $fo^* = v(\hat{S_2})$; // Update the incumbent solution
  end
  if $Mode_2 = \max \wedge v(\hat{S_2}) < v(S_2)$ **then**
    $Mode_2 = \min$; // S_2 may be a local maximizer
  end
  if $Mode_2 = \min \wedge v(\hat{S_2}) > v(S_2)$ **then**
    $Mode_2 = \max$; // S_2 may be a local minimizer
  end
  if $Mode_1 = \max \wedge v(\hat{S_1}) < v(S_1)$ **then**
    $Mode_1 = \min$
  end
  if $Mode_1 = \min \wedge v(\hat{S_1}) > v(S_1)$ **then**
    $Mode_1 = \max$
  end
  $S_1 \leftarrow \hat{S_1}$;
  $S_2 \leftarrow \hat{S_2}$;
  Updates the tabu list T_1 and T_2;
until *Stopping Criteria*;

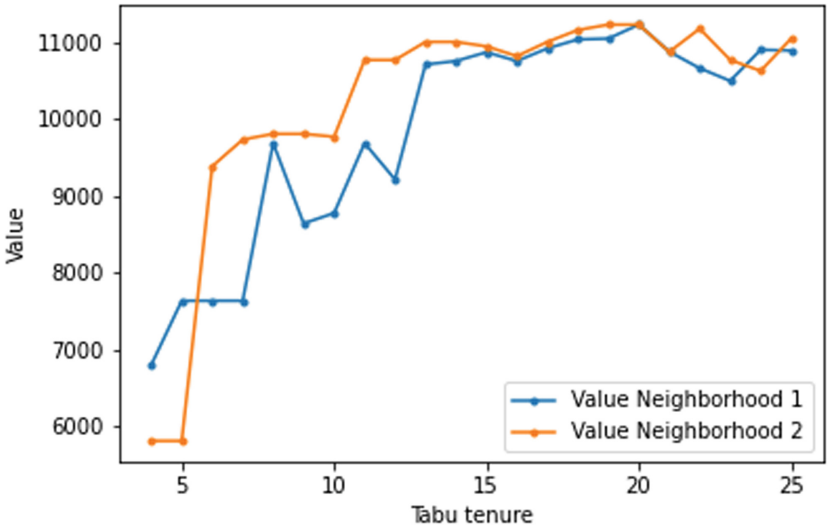

Fig. 3. Variation of the result obtained according to the *tabu tenure*

4.3 Computational Experience

The data sets used to test the backpack problem were taken from [1], where small and large sets were used. The solution was encoded by an N dimensional array of $0\backslash 1$. A value in component i represented variable x_i. The neighborhood used for the tests was simple, in this problem the items are or are not included in the backpack, so the neighborhood consisted of changing the state of an item, from 0 to 1 or vice-versa. To fine-tune the *tabu tenure* several values were tested for the *tabu tenure* in each of the data sets and it was chosen the best. It should be noted that the second neighborhood has one more parameter that needs to be adjusted, which is the number of iterations that neighborhood 2 must remains with the same strategy (min or max) before switching it.

During the tests to determine the best value of *tabu tenure*, it was found that the value of this parameter is generally lower in neighborhood 2 than in neighborhood 1. In the Fig. 3 it can be seen that when the value of *tabu tenure* is lower the results of neighborhood 2 are better than those of neighborhood 1. *tabu tenure* has this behavior because with the structure of neighborhood 2 it is less likely that the search will enter a loop, so it is not necessary to limit as many elements as in neighborhood 1, as this is more likely to create loops.

The results obtained for each of the neighborhoods as the iteration in which the best solution was obtained are found in Table 1 and the graphic representation presented above corresponds to the data set with 200 elements in the table. Through its analysis we can conclude that both neighborhoods obtained the optimal value for the example presented, however neighborhood 1 needed a greater number of iterations to obtain the optimal solution. Analyzing the results obtained, it can be concluded that the use of the double neighborhood can present better results with a smaller number of iterations for large data sets. On smaller datasets using a common tabu search seems to be sufficient.

Table 1. Results obtained for the double neighborhood in the knapsack problem

Number of elements	Optimal value	Best value Neigh. 1	Best iteration Neigh. 1	Best value Neigh. 2	Best Iteration Neigh. 2
10	295	295	19	295	31
20	1024	1024	0	1024	0
4	35	35	4	35	6
4	23	23	0	23	0
15	481.07	481.07	0	481.07	0
10	52	52	11	52	22
7	107	107	0	107	0
23	9767	9767	32	9767	26
5	130	130	0	130	0
20	1025	1025	0	1025	0
100	9147	9147	127	9147	110
200	11238	11238	3725	11238	700
500	28857	25806	4968	24500	1998
1000	54503	42499	1219	43701	331
2000	110625	67743	295	67000	209

5 Conclusions

In this paper a new approach to the tabu search was proposed and tested, which consists of the use of two independent neighborhoods with the switching of the search strategy in the inspection of the solutions in the neighborhood. In the instances tested, for a Knapsack problem, it was shown that it was a good strategy and that it can be applied to different types of problems. However, although the results obtained for this approach are better than those obtained by the common tabu search, this strategy requires more computational effort. However, we believe in the merits of this new proposal specially in problems with many local optima.

Acknowledgement. We would like to thank the reviewers for their valuable comments and suggestions that helped to improve our manuscript.

References

1. Instances of 0/1 knapsack problem. http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP/
2. Abdul-Rahman, S., Yung, V.C.: Comparison of optimization and tabu search approach for solving capital budgeting based on knapsack problem (2019)
3. Amaral, P., Pais, T.C.: Compromise ratio with weighting functions in a tabu search multi-criteria approach to examination timetabling. Comput. Oper. Res. **72**, 160–174 (2016)

4. Assi, M., Haraty, R.A.: A survey of the knapsack problem. In: 2018 International Arab Conference on Information Technology (ACIT), pp. 1–6. IEEE (2018)
5. Cheeneebash, J., Nadal, C.: Using tabu search heuristics in solving the vehicle routing problem with time windows: application to a Mauritian firm. Univ. Maurit. Res. J. **16**, 448–471 (2010)
6. Cho, M.: The knapsack problem and its applications to the cargo loading problem. Anal. Appl. Math **48**, 48–63 (2019)
7. Cordeau, J.F., Laporte, G.: Tabu search heuristics for the vehicle routing problem, pp. 145–163. Springer, Boston (2005). https://doi.org/10.1007/0-387-23667-8_6
8. Gendreau, M.: An introduction to tabu search. In: Handbook of Metaheuristics, pp. 37–54. Springer, Cham (2003). https://doi.org/10.1007/0-306-48056-5_2
9. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publisher, Dordrecht (1997)
11. Hertz, A., Taillard, É., Werra, D.: A tutorial on tabu search (1992)
12. Hertz, A., de Werra, D.: The tabu search metaheuristic: how we used it. Ann. Math. Artif. Intell. **1**, 111–121 (1990). https://doi.org/10.1007/BF01531073
13. Hvattum, L.M.: On the value of aspiration criteria in tabu search. Int. J. Appl. Metaheuristic Comput. (IJAMC) **7**(4), 39–49 (2016)
14. Nonobe, K., Ibaraki, T.: A tabu search approach to the constraint satisfaction problem as a general problem solver. Eur. J. Oper. Res. **106**(2), 599–623 (1998). https://doi.org/10.1016/S0377-2217(97)00294-4
15. Osman, I.: Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann. Oper. Res. **41**, 421–451 (1993)
16. Pais, T.C., Amaral, P.: Managing the tabu list length using a fuzzy inference system: an application to examination timetabling. Ann. Oper. Res. **194**, 341–363 (2012). https://doi.org/10.1007/s10479-011-0867-6
17. Pirim, H., Bayraktar, E., Eksioglu, B.: Tabu search: a comparative study. In: Jaziri, W. (ed.) Tabu Search, chap. 1. IntechOpen, Rijeka (2008). https://doi.org/10.5772/5637
18. Prajapati, V.K., Jain, M., Chouhan, L.: Tabu search algorithm (TSA): a comprehensive survey. In: 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE), pp. 1–8. IEEE (2020)
19. Taillard, É., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.: A tabu search heuristic for the vehicle routing problem with soft time windows. Transp. Sci. **31**, 170–186 (1997)
20. Taillard, E.: Tabu search. In: Siarry, P. (ed.) Metaheuristics, pp. 51–76. Springer, Cham (2016)
21. Talbi, E.G.: Metaheuristics: From Design to Implementation. Wiley, Hoboken (2009)
22. Werra, D., Hertz, A.: Tabu search: a tutorial and an application to neural networks. OR Spektrum **11**, 131–141 (1989)