# Hybrid Approaches based on Simulated Annealing, Tabu Search and Ant Colony Optimization for Solving the k-Minimum Spanning Tree Problem

El Houcine Addou[1], Abelhafid Serghini[2]
LANO Laboratory FSO-ESTO Mohammed I University
BV Mohamed VI BP 717, Oujda 60000, Morocco

El Bekkaye Mermri[3]
FSO Mohammed I University
BV Mohamed VI BP 717, Oujda 60000, Morocco

***Abstract*—In graph theory, the k-minimum spanning tree problem is considered to be one of the well-known NP hard problems to solve. This paper address this problem by proposing several hybrid approximate approaches based on the combination of simulated annealing, tabu search and ant colony optimization algorithms. The performances of the proposed methods are compared to other approaches from the literature using the same well-known library of benchmark instances.**

***Keywords*—k-Minimum spanning tree; metaheuristics; simulated annealing; ant colony optimization algorithms; tabu search; approximation algorithms**

## I. Introduction

In this work we attempt to provide some approximate methods to solve the well-known combinatorial optimization: The $k$-minimum spanning tree ($k$-MST) problem. We want to find a tree in an edge-weighted graph $G = (V, E)$ that have exactly $k$ edges and which minimize the sum of the weights of its edges. The mathematical formulation of this problem has been introduced by [1], It was demonstrated that the $k$-MST problem is known to be NP-hard [2], it is very difficult to find optimal solutions to large-scale problems that can be formulated as a $k$-MST within acceptable time.

In literature, several approaches using metaheuristics were proposed to tackle the $k$-MST problem [3], [4], [6], [5], [7]. In [8], three approaches to deal with the $k$-MST were presented: an evolutionary computation, ant colony optimization (ACO) and tabu search (TS), in order to show and compare the performance of these methods, the authors built a library named KCTLIB which contains some graph instances for $k$-MST problems, after that they executed their programs, numerical results showed that ACO algorithm is the best choice to deal with $k$-MST with small cardinalities, while TS is the best choice for k-MST with large cardinalities. these conclusions had inspired the authors in [14] to hybridize TS with ACO and the authors in [16] to hybridize TS with SA, they have applied their approaches to some graph instances from KCTLIB, and they have presented the results of their methods.

The objective of this paper is to attempt new hybrid approaches by coupling simulated annealing (SA), TS and ACO algorithms. We aim to find new best $k$-MST solutions, numerical experiments were performed using two graph instances from KCTLIB. Our experiments show that the suggested approaches are able to find new best values for the two graphs and for several cardinalities.

The paper is structured as follows. The problem formulation is presented in section II . In the section III we give the description of the main components of SA, TS, and ACO algorithms. the results of the computational experiments and the discussion are reported in section IV . Finally, in section V we give some conclusions.

## II. Problem Formulation

Given an undirected edge-weighted graph $G = (V, E)$ on a set $V$ of $n$ vertices , and a positive cost function $w(e)$ on the set of edges $E$, $|V|$ is the number of vertices of $G$, $|E|$ is the number of edges. A spanning tree (ST) of $G$ is a connected subgraph that contains all vertices and without any cycle. The $k$-spanning tree $(k \leq |V| - 1)$ that we note $T_k$ is a tree that contains $k$ edges, if $k = |V| - 1$ we get a spanning tree. we note by $X_k$ the set of possible $k$-spanning trees, The set of all possible $k$-spanning trees is denoted by $X_k$. the $k$-minimum spanning tree ($k$-MST) problem asks for a $k$-spanning tree with the minimum sum of weights. The $k$-MST problem can be formulated as follows:

$$\begin{cases} \text{Minimize} & \displaystyle\sum_{e \in E(T_k)} w(e) \\ \text{Subject to} & T_k \in X_k, \end{cases}$$

$E(T_k)$ denotes edges set of $T_k$ .

An optimal solution can be easily found in case of small problem size by enumerating all $k$-spanning trees in a given graph. However, it has been demonstrated that the $k$-MST problem is a NP-hard problem. Therefore, it is very important to develop new approximate methods using metaheuristics in order to provide solutions to real-world problems in reasonable time.

## III. Proposed Approaches

### A. Simulated Annealing

SA is an approximate algorithm inspired from thermodynamics, which was introduced in [9] and it is widely used to address many discrete and continuous optimization problems, such as the travelling salesman, and vehicle routing problem, etc. Researchers are also applying it to multi-criteria

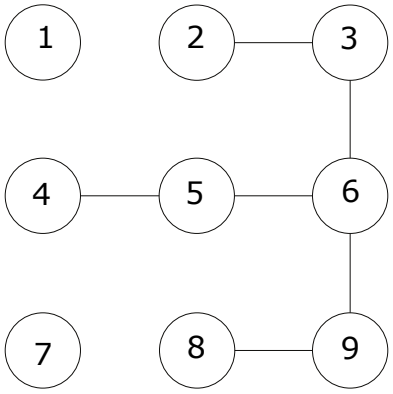


Fig. 1. A 6-ST of 7 Vertices.

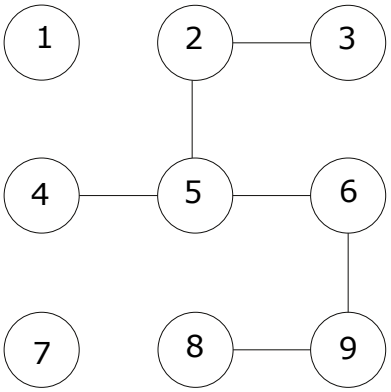


Fig. 2. A Neighborhood Element of the 6-ST.

optimization problems [10], [11]. SA is a neighborhood search method. The basic concept of this method is to move at each iteration from a solution $x$ to another one that belongs to its neighborhood $N(x)$. Before describing the main SA components, we present the neighborhood structure used in this method.

### B. The Neighborhood

To move around and allow the algorithm to discover the search space, the possible transitions or movements from one solution to another are subject to the chosen neighborhood structure. The neighborhood of a tree $T_k$ is the set of $k$-ST built by deleting one edge from $T_k$ and replacing it by an edge of the graph $G$ and which doesn't belong to $T_k$. We choose randomly an edge from the current $T_k$ and replace it by another one, which is also is selected randomly from $G \backslash T_k$. To clarify the neighborhood structure, we take the example of a tree composed by six edges in Fig. 1. The 6-ST contains the following edges: (2,3),(3,6),(6,5),(5,4),(6,9),(9,8).

Fig. 2 represents a neighborhood element of the 6-ST in Fig. 1, where the edge (3,6) is replaced by the edge (2,5).

### C. SA Algorithm

The SA settings are adjusted empirically, as follows, the main lines of the proposed SA algorithm are as follows:

1) Set the initial values of the following parameters:
   a) $T_0$: The initial temperature
   b) $\alpha$: Factor of cooling
   c) $TMP\_LEVELS$: The maximum number of temperature levels; the temperature is not decreased at each iteration but it is decreased by level
   d) $TMP\_RANGE$: Number of iterations per level
   e) $TIME\_TO\_DIVERSIFY$: the time limit of the algorithm to launch the diversification based on the ACO.
2) Initial solution: Use the Prim method to build a $k$-subtree
3) SA procedure
   - Until $TMP\_LEVELS$ repeat
   a) Repeat for $TMP\_RANGE$ iterations:
      - Select randomly a $k$-ST in the neighborhood of the current solution.
      - Calculate the objective function of the current solution $f_1$
      - Calculate the objective function of the neighborhood solution $f_2$
      - Calculate $F$ = $f_2$ - $f_1$
        If $F < 0$ then set the neighborhood element as the current solution
        Otherwise, the neighborhood element will be the current solution with a probability equal to $exp(-F/T)$, $T$ is the value of the temperature in the current iteration
   b) Use the geometric cooling schedule to decrease the current temperature

### D. Tabu Search

TS was introduced by F. Glover and Laguna in [12], [17], [13]. This metaheuristic method is used at large scale to find approximate solutions for real-world optimization problems, TS can be applied for example in logistics, resource planning, telecommunications, scheduling, etc.

TS is based on simple ideas inspired from the human memory, it is a local search method which is known to have the tendency to be stuck in local extremums, TS address this problem by prohibiting already visited solutions and avoiding the problems of cycles, in this way, the whole solutions space can have the chance to be visited. For more details about the components description and the complete algorithm of TS algorithm integrated in our proposed hybrid approaches please refer to [14],

### E. Ant Colony Optimization

ACO algorithm is a probabilistic technique for tackling computational problems which can be reduced to seeking optimal paths in graphs, it was first introduced by Dorigo, Maniezzo and Colorni [15].

The principle of the first algorithm is inspired from the capability of ants to seek the best path from their colony to source food and vice versa. As they move, they deposit an organic compound on the ground called pheromone, paths

(solutions) with a higher pheromone level have a higher probability of being selected by ants (better solutions).

In this paper, the ACO is used in order to diversify the search process, and to explore new regions that may have not been visited in previous iterations by SA algorithm. For more details about the components description and the complete ACO algorithm used in this paper refer to [14].

### F. First Hybrid Approach: SA Combined with ACO

In [8], ACO algorithm showed a high diversification ability which allowed it to explore new areas of solutions. To take advantage of this feature, we propose a hybrid approach that combines SA and ACO. The ACO algorithm is used in order to diversify the search process when the SA algorithm can no longer improve the current solution.

In summary: an initial solution will be generated using the Prim method, then the SA will be launched until until that we don't observe any improvement is occurring on the current solution; at that time we call the ACO algorithm to move the search to other regions of solutions space. Next, the SA will resume. Below the outline of this hybrid approach that we note Hybrid SA-ACO :

Step 1. Initialization of SA parameters and generation of an initial solution (see Section III-C).
Step 2. SA and ACO
- a) Launch the SA algorithm as described in Section III-C until $TIME\_TO\_DIVERSIFY$ is reached.
- b) Run the ACO algorithm as described in [8]
- c) Repeat steps 2.a and 2.b until $TMP\_LEVEL$ is reached.

$TIME\_TO\_DIVERSIFY$: ACO diversification procedure is launched when this time is reached.

### G. Second Hybrid Approach: SA Combined with ACO and TS

In the literature, TS has shown a high intensification potential in finding good solutions in narrow search space for the k-MST problem, the experimentations carried out by [8], have proven this ability, the numerical results showed also that TS is very efficient in case of k-MST with large cardinalities. In [14] TS had had showed another ability which is that it can be a good partner in a hybrid algorithm, the combination with the ACO algorithm had allowed to find new best values for the objective function. In [16] another hybrid approach was proposed by combining TS and SA. These results were obtained even that the neighborhood structures chosen by each approach were different

In this hybrid approach, we combine the three meta-heuristics seen before, namely, SA, TS and ACO, we will exploit the advantages of each of them to find better solutions. This approach starts with an initial solution generated using the Prim method, then the SA will take the hand to improve this solution under the control of the parameter $TIME\_TO\_DIVERSIFY$; the ACO will be launched to take the search to another region of solutions not yet explored. When the stopping criterion ($TMP\_LEVEL$) is reached, the TS is launched on the best solution found to intensify the search in its neighborhood and get the best one.

The outline of this algorithm, that we note Hybrid SA-ACO-TS, is as follows:

Step 1. Initialization of SA parameters and generation of an initial solution (see Section III-C).
Step 2. SA and ACO
- a) Run the SA algorithm until $TIME\_TO\_DIVERSIFY$
- b) Launch the ACO algorithm
- c) Repeat the two previous actions until $TMP\_LEVEL$

Step 3. Tabu Search
- a) Run the TS algorithm as described in [14], the best solution found in Step 2 is the starting solution of this step.

## IV. Experimental Results

We have performed numerical experiments using two large regular graphs taking from the benchmark instance KCTLIB. The Table I shows the configuration of these graphs.

TABLE I. Characteristics of the Two Regular Graphs.

| Graph name | $\|V\|$ | $\|E\|$ | Average degree of vertices |
|---|---|---|---|
| Graph 1 : 1000_4_01.gg | 1000 | 2000 | 4 |
| Graph 2 : g400_4_05.g | 1000 | 2000 | 4 |

The results obtained by the proposed approaches are compared to those obtained with the following methods:

- Two solution algorithms proposed in [8], namely TS algorithm and ACO algorithm, denoted by TSB and ACOB, respectively.
- The hybrid solution algorithm proposed in [14], denoted by HybridK.
- The simulated annealing algorithm with restart strategy by [7], denoted by SA.
- The hybrid approach that combine simulated annealing and tabu search by [16].

Our algorithms are coded in C programming language and runned on a computer with a CPU Intel(R) Core(TM) i5, 2.5x2.5 GHz, 4GB RAM. It should be stressed that for TS and ACO algorithms, we have used the same parameter settings as in [14],

Our algorithms have been executed ten times on each graph instance, after that, we record the best, worst, mean and objective function values, also we record the mean time.

Table II presents the SA parameter settings adopted to tackle the graphs 1 and 2. In Tables III- IV results of the proposed approaches are shown. The objective function written in bold style face means that they are best among all obtained values. BNV column gives the best new value of our approaches.

TABLE II. SA PARAMETER SETTINGS ADOPTED TO GRAPHS IN TABLE I.

| $k$ | $T_0$ | $T_f$ | TMP_RANGE | TIME_TO_DIVERSIFY | TMP_LEVELS |
|---|---|---|---|---|---|
| 200 | 15 | 0.01 | 10000 | 20 s | 40 |
| 400 | 15 | | 2000 | | 30 |
| 600 | 15 | | 5000 | | 30 |
| 800 | 10 | | 2000 | | 30 |
| 900 | 10 | | 4000 | | 30 |

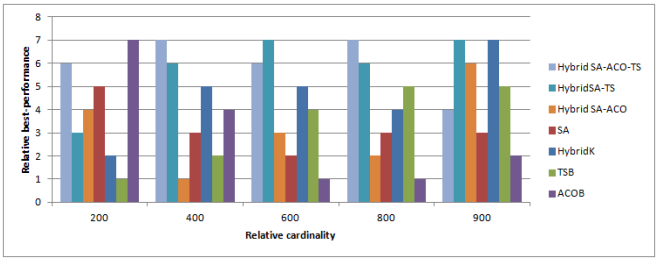


Fig. 3. Performance of Different Approaches for the First Regular Graph.

To easily read and well interpret the Tables III- IV, we transform them into charts, Fig. 3 and 4 represent the order of the performance of each method and for different cardinalities, a higher order value corresponds to better performance.

Fig. 3 show that: for $k = 200$, ACOB is the best; however for other cardinalities we notice that our proposed hybrid approach SA_ACO_TS is very competitive, because we have improved the best known solutions for $k = 400$ and $k = 800$. Fig. 4 show that for $k = 400$, Hybrid SA_ACO_TS is the best; however for other cardinalities we notice that Hybrid SA_TS approach is very efficient in terms of best and mean values. It should be stressed that we have improved the best known solutions for $k = 400$ and $k = 800$. In summary, results reveal that:

- SA is not efficient in case of large graphs.
- SA is very efficient when coupled with TS in finding optimal k-MST solutions.
- TS is known for its great ability to intensify the search, the obtained results had confirmed that; it can be coupled with SA and/or ACO algorithm to obtain good performances.
- ACO is very efficient when its coupled with TS.
- We have achieved good results by hybrid approaches only when TS is part of the hybrid approach, this is due to its high intensification ability.
- Our proposed approaches have consumed more time to provide optimal solutions.

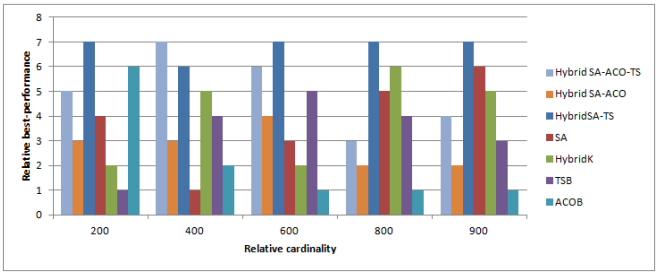


Fig. 4. Performance of Different Approaches for the First Regular Graph.

## V. CONCLUSION

This article suggests new approaches to address the $k$-MST problem. Hybrid approaches combining SA, TS and ACO were presented. In order to show the performance of the these methods, we compared them with other works from the literature using the same benchmark data KCTLIB. The numerical experiments showed that TS is effective to tackle the $k$-MST problems when it is combined with SA or ACO or both. In our future works, we will focus on how to improve the computational time of our approaches and then address the same problem in case of multi-objective optimization.

## REFERENCES

[1] H. W Hamacher, K. Jörnsten, and F Maffioli. Weighted k-cardinality trees, technical report. *Technical Report 91.023*, Politecnico di Milano, Dipartimento di Elettronica, Italy, 1991.

[2] Matteo Fischetti, Horst W. Hamacher, Kurt Jörnsten, and Francesco Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24(1):11–21, 1994.

[3] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.

[4] Matthias Ehrgott, Horst. W. Hamacher, J. Freitag, and F. Maffioli. Heuristics for the k-cardinality tree and subgraph problems. *Fachbereich Mathematik*, 14(8):24, 1996.

[5] Shun Yan Cheung and A. Kumar. Efficient quorumcast routing algorithms. *Proceedings of INFOCOM '94 Conference on Computer Communications*, pages 840–847 vol.2, 1994.

[6] Freitag J. K Ehrgott M. Tree/k subgraph: a program package for minimal weighted k-cardinality-trees and -subgraphs. *European Journal of Operational Research*, 1(93), 214, 1996.

[7] El Houcine Addou, Abelhafid Serghini, and El Bekkaye Mermri. Simulated annealing algorithm with restart strategy for optimizing k-minimum spanning tree problems. *In EDA 2018*, Business Intelligence & Big Data, RNTI-B-14:321–330, 2018.

[8] Christian Blum and Maria J. Blesa. New metaheuristic approaches for the edge-weighted k-cardinality tree problem. *Computers & Operations Research*, 32(6):1355–1377, 2005.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220(4598):671–680, 1983.

[10] Marc C. Robini and Pierre-Jean Reissman. From simulated annealing to stochastic continuation: a new trend in combinatorial optimization. *Journal of Global Optimization*, 56(1):185–215, 2013.

[11] Linzhong Liu, Haibo Mu, Juhua Yang, Xiaojing Li, and Fang Wu. A simulated annealing for multi-criteria optimization problem: DBMOSA. *Swarm and Evolutionary Computation*, 14:48–65, 2014.

[12] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[13] F. Glover. Tabu search part ii. *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[14] Hideki Katagiri, Tomohiro Hayashida, Ichiro Nishizaki, and Qingqiang Guo. A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems. *Expert Systems with Applications*, 39(5):5681–5686, 2012.

[15] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[16] El Houcine Addou, Abelhafid Serghini, and El Bekkaye Mermri. A hybrid algorithm based on simulated annealing and tabu search for k-minimum spanning tree problems. *In ICOA 2019*, pages 1–6, April 2019.

TABLE III. RESULTS OBTAINED BY EACH APPROACH FOR THE FIRST REGULAR GRAPH.

| $k$ | BNV | | Hybrid SA_ACO_TS | Hybrid SA_ACO | Hybrid SA_TS | SA | HybridK | TSB | ACOB |
|---|---|---|---|---|---|---|---|---|---|
| 200 | | Best | 3336 | 3374 | 3375 | 3372 | 3393 | 3438 | **3312** |
| | | Mean | 3354.5 | 3420.2 | 3419.9 | 3450 | 3453.1 | 3461.4 | **3344.1** |
| | | Worst | **3373** | 3463 | 3514 | 3514 | 3517 | 3517 | 3379 |
| | | Mean time (s) | 742 | 300 | 600 | 300 | 300 | 300 | 300 |
| 400 | 7621 | Best | **7621** | 7717 | 7646 | 7713 | 7659 | 7712 | 7661 |
| | | Mean | **7669.6** | 7822.4 | 7689.8 | 7772.8 | 7764 | 7780.2 | 7703 |
| | | Worst | **7739** | 7899 | 7778 | 7851 | 7819 | 7825 | 7751 |
| | | Mean time (s) | 6328 | 300 | 3468 | 974 | 300 | 300 | 300 |
| 600 | | Best | 12783 | 12805 | **12756** | 12858 | 12785 | 12801 | 12989 |
| | | Mean | 12821.1 | 12887.4 | **12795.4** | 12908.1 | 12836.6 | 12821.8 | 13115.6 |
| | | Worst | **12864** | 12999 | 12845 | 12971 | 13048 | 12869 | 13199 |
| | | Mean time (s) | 21815 | 1644 | 11904 | 1948 | 300 | 300 | 300 |
| 800 | 19065 | Best | **19065** | 19144 | 19073 | 19114 | 19099 | 19093 | 19581 |
| | | Mean | 19110 | 19162.3 | **19081.6** | 19213.7 | 19101.1 | 19112.6 | 19718.7 |
| | | Worst | 19155 | 19177 | **19095** | 19275 | 19128 | 19135 | 19846 |
| | | Mean time (s) | 7200 | 2509 | 3180 | 1948 | 300 | 300 | 300 |
| 900 | | Best | 22845 | 22942 | **22827** | 22865 | **22827** | 22843 | 23487 |
| | | Mean | 22851 | 22962.7 | 22834.5 | 23052 | **22827** | 22859.2 | 23643 |
| | | Worst | 22866 | 22995 | 22851 | 23165 | **22827** | 22886 | 23739 |
| | | Mean time (s) | 6827 | 1440 | 9263 | 1029 | 300 | 300 | 300 |

TABLE IV. RESULTS OBTAINED BY EACH APPROACH FOR THE SECOND REGULAR GRAPH.

| $k$ | BNV | | Hybrid SA_ACO_TS | Hybrid SA_ACO | Hybrid SA_TS | SA | HybridK | TSB | ACOB |
|---|---|---|---|---|---|---|---|---|---|
| 200 | | Best | 3636 | 3661 | **3630** | 3639 | 3667 | 3692 | 3632 |
| | | Mean | 3671 | 3686.8 | **3653** | 3699.9 | 3697.5 | 3722.0 | 3670.1 |
| | | Worst | 3716 | 3722 | **3682** | 3784 | 3738 | 3751 | 3710 |
| | | Mean time (s) | 995 | 1028 | 2777 | 2735 | 300 | 300 | 300 |
| 400 | 8240 | Best | **8240** | 8359 | 8292 | 8378 | 8323 | 8358 | 8376 |
| | | Mean | **8293.7** | 8393.2 | 8343.1 | 8430 | 8357.1 | 8385.6 | 8408.3 |
| | | Worst | **8367** | 8436 | 8472 | 8510 | 8424 | 8415 | 8442 |
| | | Mean time (s) | 4272 | 1369.1 | 4200 | 1301 | 300 | 300 | 300 |
| 600 | | Best | 13681 | 13743 | **13617** | 13761 | 13807 | 13735 | 14085 |
| | | Mean | 13708 | 13816.4 | **13665.6** | 13788.2 | 13824.3 | 13759.4 | 14164.5 |
| | | Worst | 13744 | 13876 | **13690** | 13841 | 13900 | 13820 | 14235 |
| | | Mean time (s) | 28727 | 2032 | 16042 | 2082 | 300 | 300 | 300 |
| 800 | | Best | 20149 | 20208 | **20108** | 20127 | 20110 | 20130 | 20661 |
| | | Mean | 20170 | 20251 | 20143.6 | 20169 | **20129.9** | 20142.9 | 20811.3 |
| | | Worst | 20189 | 20279 | 20167 | 20218 | **20143** | 20155 | 20940 |
| | | Mean time (s) | 21355 | 2739 | 7980.6 | 3802 | 300 | 300 | 300 |
| 900 | | Best | 24039 | 24103 | **24030** | 24032 | 24035 | 24044 | 24782 |
| | | Mean | 24070 | 24136 | **24034.6** | 24045.3 | 24035 | 24052.6 | 24916 |
| | | Worst | 24090 | 24172 | 24040 | 24052 | **24035** | 24064 | 25037 |
| | | Mean time | 8756 | 1646 | 15671.6 | 4339 | 300 | 300 | 300 |

[17] F. Glover and M. Laguna. Tabu search. *Handbook of Combinatorial Optimization,Springer,Boston*, pp. 2093–2229, 1998.