Simulated annealing algorithm with restart strategy for optimizing k-minimum spanning tree problems

El Houcine Addou[*], Abelhafid Serghini[**]
El Bekkaye Mermri[***]

[*]LANO Laboratory FSO, University Mohammed Premier, Oujda, Morocco
e.addou@ump.ac.ma,
[**]LANO Laboratory FSO-ESTO, University Mohammed Premier, Oujda, Morocco
a.serghini@ump.ma,
[***]FSO, Department of Mathematics, University Mohammed Premier , Oujda, Morocco
e.mermri@ump.ac.ma

Abstract. In this paper we consider the k-minimum spanning tree problem that generalizes the famous minimum weight spanning tree problem which is one of the major classes of combinatorial optimization problems. We propose an approach for solving this problem based on the Simulated Annealing algorithm. The performance of the proposed method is compared with existing metaheuristics using the well-known benchmark instances KCTLIB.

1 Introduction

In this paper we are interested in solving one of the well known combinatorial optimization problems: The k-minimum spanning tree problem (k-MST). The objective is to find a subtree with exactly k edges in an edge-weighted graph $G = (V, E)$, such that the sum of the weights is minimal. The combinatorial optimization model was introduced at first by Hamacher et al. (1991) for the relinquishment of petroleum licenses. It was demonstrated that the k-MST problem is NP-hard and it is very difficult to solve problems that can be formulated as a k-MST within a reasonable time (Fischetti et al., 1994; Ravi et al., 1996). In the literature, there have been several local search methods based on metaheuristic algorithms proposed for solving the k-MST problem. In 2005 Blum and Blesa (2005) suggested three metaheuristics: evolutionary computation (EC), ant colony optimization (ACO) and tabu search (TS). They compared their performances through benchmark instances KCTLIB and showed that an ACO approach is the best for small cardinality, whereas TS is the best for large cardinality. In 2012, a new hybrid algorithm that combines TS and ACO is provided by Katagiri et al. (2012). The purpose of the present paper is to offer new method for solving the k-MST problem. We propose an approximate approach to solve the problem of k-MST based on the simulated annealing (SA) algorithm. We were motivated by the fact that the SA algorithm was proposed to solve the problem of the generalized minimum spanning tree problem (Pop et al., 2007).However, to our knoledge, there is no previous work using SA algorithm to tackle the k-MST. Our primary focus was on developing an algorithm capable of producing solutions of high quality. Results of numerical experiments through the same benchmark instances show that the proposed method updates some of the best known values with a very short time and that the proposed method provides a better performance with solution accuracy over existing algorithms. The paper is divided as follow: In the next section we present the problem statement. The components and outline of the SA algorithm is described in Section 3. In Section 4 we report results of the computational experiments on the benchmark instances KCTLIB, a library for the k-MST maintained by C. Blum and M. Blesa, which was obtained directly by contacting via e-mail with the author M.J Blesa (Blum and Blesa, 2005) and we compare the results of our approach with those of existing methods. Finally, some concluding remarks and future work are given in Section 5.

2 Problem definition

Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. A subgraph T of G is called a spanning tree (ST) if it covers all the vertices of the graph, is connected and has no cycle. A k-spanning tree $(k \leq |V| - 1)$, denoted by T_k,is a connected subgraph of G with k edge and has no cycle. When $k \leq |V| - 1$, we get a spanning tree. The set of all possible k-spanning trees is denoted by X_k. Then a k-MST problem can be expressed as:

$$\begin{gathered}\text{Minimize } \textstyle\sum_{e \in E(T_k)} w(e) \\ \text{subject to } T_k \in X_k\end{gathered}$$

where $E(T_k)$ denotes the set of edges of T_k and $w(e)$ is the weight of the edge e. The problem is to seek a k-spanning tree with the minimum sum of weights. In case of small problem size an optimal solution can be found after enumerating all possible k-spanning trees in a given graph. If the size of the problem is not so large, some exact solution algorithm such as a branch and bound method (Cheung and Kumar, 1994) and a branch and cut algorithm (Freitag, 1993) can solve the problem. However, it has been shown that the k-MST problem is NP-hard even if the edge weight is in 1, 2, 3 for all edges, or if a graph is fully connected. The problem is also NP-hard for planar graphs and for points in the plane (Ravi et al., 1996). Therefore, it is necessary to build efficient approximate solution methods based on metaheuristic approaches that are very useful to find an approximate optimal solution in a reasonable time.

3 Proposed approach

3.1 Simulated annealing

SA is a metaheuristic algorithm inspired by thermodynamics which was described for the first time by Scott Kirkpatrick et al. (Kirkpatrick et al., 1983) in 1983, to solve a huge number of combinatorial optimization problems. SA algorithm is a local search algorithm which has been widely used in discrete and continuous optimization problems. At first, SA algorithm was mainly used in combinatorial optimization fields, such as the problem of the travelling salesman (Lam et al., 2013), quadratic assignment problem (Ghandeshtani et al., 2010) and vehicle routing problem (Alrefaei et al., 2013) etc. Recently, researchers apply it in constrained and multi-criteria optimization problems (Robini and Reissman, 2013; Liu et al., 2014). The basic concept of SA, which is a neighborhood search method, is to explore the space of solutions by moving, at each iteration, from a solution x to the best solution in its neighborhood $N(x)$. First, we begin by describing the structure of the neighborhood of a fusible solution used in our approach, and subsequently we will describe the main component of this approach.

3.2 The neighbourhood structure

In order to move systematically through the search space, the possible move from one solution to another is restricted by the neighborhood structure chosen. The neighborhood of a solution T_k is the set of k-ST formed by removing one edge from T_k and changing it by another edge of G which does not belong to T_k. To illustration the neighborhood structure, we consider an example of a tree with 7 edges in figure 1. The 7-ST is represented by the set of edges: (1,2),(2,3),(3,6),(6,5),(5,4),(6,9),(9,8). Figure 2 shows an element of the neighborhood of the 7-ST in figure 1, where the edge (3,6) is changed by the edge (2,5). In our move we

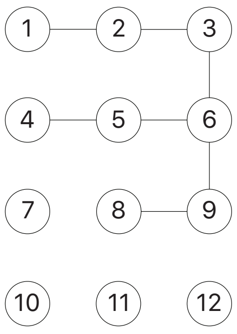

FIG. 1: A 7-ST in a graph of 12 vertices.

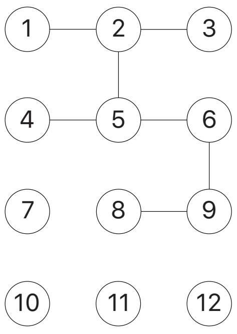

FIG. 2: A neighborhood element of the original 7-ST.

randomly choose an element from the neighborhood of the k-ST as follows: first, we randomly select an edge from the current k-ST to be deleted from T_k, then we randomly select an edge from the graph G which is part from the current k-ST, to be added to T_k.

3.3 The SA component

The algorithm is a single-solution algorithm based on the hill climbing method, that to avoid the problem of stagnating in local optima. SA accepts with a certain probability worse solutions. The algorithm begins with an initial solution generated using a constructive method such as the Prim algorithm. In each iteration, a solution from the neighborhood is chosen depending on a predefined neighborhood structure and evaluated using a fitness function. If the move improves the current solution, that is the selected solution is better than the current one, so it is accepted; otherwise it is accepted with a certain probability determined by the Boltzmann probability $P = exp(\theta/T)$, where θ is the difference between the fitness of the current solution and the selected one, and T is a controller parameter called the temperature, which periodically decreases during the search process according to the cooling schedule chosen. We have chosen to use the geometric cooling schedule method because it is the most common(Dreo et al., 2003). The temperature is decreased as follow: $T = \alpha * T$ (with $0.8 < \alpha < 1$), where α is the cooling factor. The cooling process should be carried out cautiously(A. Dowsland, 1995). The temperature is decreased during the search process not at each move but after a certain number of moves, we denote $TEMP\_RANGE$ the number of iterations that the search performs at a particular temperature. The algorithm stops only if the best found solution does not improve during a certain number of levels of temperature that we note MAX_TEMP_LEVEL. In the absence of general theoretical results that can be really exploited, the user cannot escape from an empirical adjustment of SA parameters, so there are no choices of SA parameters that will be good for all problems, and there is no general way to find the best choices for a given problem.

3.4 Restart strategy

In the literature authors suggested many implementations for the restarting strategy, see for instanceYu et al. (2017) . The restart mechanism used in our approach is very simple and effective. The algorithm restarts only if the current best solution has not been improved for a fixed limit time given as a parameter that we note $MAX\_TIME\_RESTART$. Once the algorithm restarts, the current temperature is reset to the same temperature recorded when the best solution was found, and the current solution is replaced by the best one.

3.5 Proposed SA algorithm

The outline of the proposed algorithm is as follows:

— Step 1: Initialization of parameters

1. Set the cooling factor α,
2. Set the initial temperature T_0,
3. Set the number of iterations per temperature level $TEMP\_RANGE$,
4. Set MAX_TEMP_LEVEL,
5. Set $MAX\_TIME\_RESTART$.

— Step 2: Generate the initial solution Select a vertex in a random way, and then continuously apply the well-known Prim method until a k-subtree is built.

— Step 3: Local search procedure
- Repeat until MAX_TEMP_LEVEL,

1. Repeat $TEMP\_RANGE$ times the following instructions:
   - Randomly generate a neighbor solution using the neighbor structure as previously described.
   - Calculate the fitness of the current solution θ_1.
   - Calculate the fitness of the random selected solution θ_2.
   - Calculate the acceptance probability (Boltzmann probability): $\theta = \theta_2 - \theta_1$
   If $\theta<0$ then the selected solution becomes the current solution;
   Otherwise, the selected solution becomes the current solution with a probability equal to $exp(-\theta/T)$, where T is the current temperature.
2. If MAX_TIME_RESTART is reached then launch the restart strategy as described above, otherwise decrease the temperature using the geometric cooling schedule.

3.6 Computational results

In this section, experiments were carried out by using two kinds of graphs from the well-known benchmark instances KCTLIB, in order to compare our results with those obtained from different methods available in the literature. To show the competitiveness of the proposed approach, we compare the results obtained with:

— Two solution algorithms proposed by Blum and Blesa (2005), namely TS algorithm and ACO algorithm that we denote TSB and ACOB, respectively;

— The hybrid solution algorithm proposed by Katagiri et al. (2012) , that we denote HybridK.

The proposed algorithm are coded in C programming language and tested on a computer with a processor $Intel(R)Core(TM)i5 - 2450M, 2.5 * 2.5gigahertz$ and 4 gigabyte of RAM. The parameter settings of TS and ACO used in our approach are the same as ones provided by Blum-Blesa and Katagiri et al. for all the experiments. We have run our algorithm ten times on four graphs taken from the well-known benchark instances KCTLIB used by the authors in Blum and Blesa (2005). Table 1 shows the characteristics of these four graphs. We compute

Graph	Name	Type	Vertices number	Edges number	Average vertex degree
1	bb45x5_1.gg	grid	225	400	3.55
2	bb45x5_2.gg	grid	225	400	3.55
3	1000_4_01.gg	regular	1000	2000	4
4	g400_4_05.g	regular	1000	2000	4

TAB. 1: Characteristics of the four graphs.

the best, mean and worst objective function values for each proposed approach. It should be stressed that the parameter settings of SA has been adjusted empirically, and we give below the list of these parameters:

- T_0: initial temperature.
- T_f: minimal temperature, when it is reached the algorithm should be stopped.
- α: cooling factor set to 0.9.
- $TEMP\_RANGE$: number of iterations per temperature level.
- $MAX\_TIME\_RESTART$: This parameter represents the maximum time after which the algorithm should be restarted.
- MAX_TEMP_LEVEL: This parameter plays the role of a stopping criteria, if the best found solution does not improve during a certain number of levels of temperature then the process should be stopped.

Tables 2 and 3 describe the values of each SA parameter for tackling the grid graphs 1 and 2 and the regular graphs 3 and 4, respectively, for several cardinalities. This choice of cardinality values was done in order to have a reasonable comparison between our algorithm and the existing ones, and because we already have the values of the objective function of these cardinalities in Katagiri et al. (2012). Tables 4- 7 show the results of the proposed approach for grid graphs 1 and 2 and regular graphs 3 and 4, respectively. In this tables, SA denotes our proposed algorithm; HybridK denotes the hybrid TS and SA algorithm of Katagiri et al. (2012) ; TBS and ACOB denote, respectively, the AC and TS methods implemented by Blum and Blesa (2005). Values of the objective function written in bold-faced means that are best among all values obtained, so far, by the four algorithms. BNV represents the best new values

k	T_0	T_f	TEMP_RANGE	MAX_TIME_RESTART(s)	MAX_TEMP_LEVEL
40	10		40000	15	30
80	25		30000	25	25
120	20	2	15000	25	25
160	20		8000	25	25
200	20		2000	20	30

TAB. 2: SA parameter settings adopted to graphs 1 and 2 in Table 1.

k	T_0	T_f	TEMP_RANGE	MAX_TIME_RESTART(s)	MAX_TEMP_LEVEL
200	15		10000		40
400	15		2000		30
600	15	0.01	5000	20	30
800	10		2000		30
900	10		4000		30

TAB. 3: SA parameter settings adopted to graphs 3 and 4 in Table 1.

which have been obtained by our approach. In case of grid graphs, our experiments have been performed under the condition $TimeLimit = 200s$ whereas $TimeLimit = 300s$ in Katagiri et al. (2012). In case of regular graph the parameter MAX_TEMP_LEVEL is used as a stopping criteria, the average time needed by our approaches is given in the result tables.

k	BNV		SA	HybridK	TSB	ACOB
		Best	**695**	**695**	696	**695**
40		Mean	**695**	**695**	696	695.4
		Worst	**695**	**695**	696	696
	1551	Best	**1551**	1552	1579	1572
80	**1560.4**	Mean	**1560.4**	1565.1	1592.7	1581.2
		Worst	**1572**	**1572**	1615	1593
		Best	**2444**	**2444**	2546	2457
120	**2455.3**	Mean	**2455.3**	2457.9	2558.5	2520.3
		Worst	2472	**2465**	2575	2601
		Best	**3688**	**3688**	3724	3700
160		Mean	3698.8	**3688**	3724.9	3704.7
		Worst	3705	**3688**	3729	3720
		Best	**5461**	**5461**	5462	**5461**
200		Mean	5461.4	**5461**	5462.4	5469
		Worst	5462	**5461**	5463	5485

TAB. 4: Comparison results for the grid graph 1.

Figures 3- 4 represents the order of the performance of the four methods for different cardinalities. The higher order means the better performance. As may be observed in the Tables 4- 5 and clearly in the Figure 3, the results for grid graphs 1-2 show that for 2 out of 10 cardinalities we improve the best known solutions. In the remaining 8 cardinalities our proposed algorithm finds the same best solutions as was found by the HybridK. In term of mean and worst values, the performance of the proposed SA is better than TSB and ACOB. But more importantly, the results show that our algorithm finds for each cardinality the same best solution in a very short amount of computation time. The results in Tables 6 (Figure 4) for regular graph 1 show that the performance of our approach is better than ACOB for cardinalities higher than 400 and is also better than HybridK and TSB for cardinalities equal to 200. For other cases of cardinality, the performance of our approach is not high. The results in Table 7 for regular graph 2 show

k	BNV		SA	HybridK	TSB	ACOB
40		Best	**654**	**654**	**654**	**654**
		Mean	**654**	**654**	**654**	**654**
		Worst	**654**	**654**	**654**	**654**
80		Best	**1617**	**1617**	**1617**	**1617**
		Mean	1624.1	1619.1	**1617.1**	1626.9
		Worst	1638	1620	**1619**	1659
120	**2631**	Best	**2631**	2632	2651	2637
	2637	Mean	**2637**	2641.3	2677.9	2664.6
		Worst	2652	**2648**	2719	2706
160		Best	**3757**	**3757**	3815	3757
		Mean	3776.7	**3764.3**	3815.0	3797.6
		Worst	3808	**3779**	3815	3846
200		Best	**5262**	**5262**	**5262**	**5262**
		Mean	**5262**	**5262**	5268.6	5272
		Worst	**5262**	**5262**	5296	5288

TAB. 5: Comparison results for the grid graph 2.

k	BNV		SA	HybridK	TSB	ACOB
200		Best	3372	3393	3438	**3312**
		Mean	3450	3453.1	3461.4	**3344.1**
		Worst	3514	3517	3517	**3379**
		Mean time (s)	300	300	300	
400		Best	7713	**7659**	7712	7661
		Mean	7772.8	7764	7780.2	**7703**
		Worst	7851	7819	7825	**7751**
		Mean time (s)	974	300	300	
600		Best	12858	**12785**	12801	12989
		Mean	12908.1	12836.6	**12821.8**	13115.6
		Worst	12971	13048	**12869**	13199
		Mean time (s)	1948	300	300	
800		Best	19114	19099	**19093**	19581
		Mean	19213.7	**19101.1**	19112.6	19718.7
		Worst	19275	**19128**	19135	19846
		Mean time (s)	1948	300	300	
900		Best	22865	**22827**	22843	23487
		Mean	23052	**22827**	22859.2	23643
		Worst	23165	**22827**	22886	23739
		Mean time (s)	1029	300	300	

TAB. 6: Comparison results for the regular graph 3.

that the performance of our approach is almost better than ACO and TSB, and also is better than HybridK for cardinalities lower than 600 and bigger than 800. It should be stressed also that the computational time of SA is fairly large in case of large graphs.
It is clear that the performance of SA approach is very high in case of small graph than other already existing approaches. Our approach is competitive in case of large graph compared to TSB and ACOB, but it is not so effective than the hybrid algorithm HybridK. In case of large graphs, these results can be justified by the fact that SA algorithm is a local search method which does not incorporate any strategies for expanding the search area and diversifying the search in order to explore other regions from the space of the solutions. In addition to that, the intensification ability of SA is not so high, so it can be also reviewed and improved given that all the solutions found by the SA method are not the best ones.

k	BNV		SA	HybridK	TSB	ACOB
		Best	3639	3667	3692	**3632**
200		Mean	3699.9	3697.5	3722.0	**3670.1**
		Worst	3784	3738	3751	**3710**
		Mean time (s)	2735	300	300	
		Best	8378	**8323**	8358	8376
400		Mean	8430	**8357.1**	8385.6	8408.3
		Worst	8510	8424	**8415**	8442
		Mean time (s)	1301	300	300	
		Best	13761	13807	**13735**	14085
600		Mean	13788.2	13824.3	**13759.4**	14164.5
		Worst	13841	13900	**13820**	14235
		Mean time (s)	2082	300	300	
		Best	20127	**20110**	20130	20661
800		Mean	20169	**20129.9**	20142.9	20811.3
		Worst	20218	**20143**	20155	20940
		Mean time (s)	3802	300	300	
		Best	24032	**24035**	24044	24782
900		Mean	24045.3	**24035**	24052.6	24916
		Worst	24052	**24035**	24064	25037
		Mean time	4339	300	300	

TAB. 7: Comparison results for the regular graph 4.

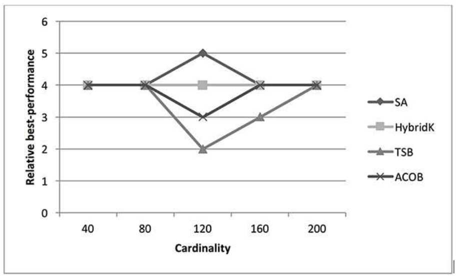

FIG. 3: Performance comparison for the grid graph 2.

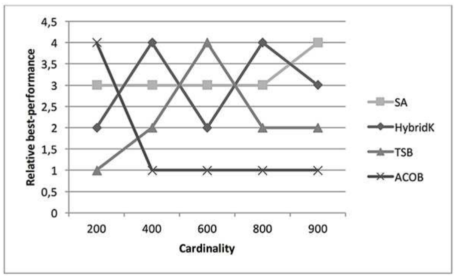

FIG. 4: Performance comparison for the regular graph 4.

4 CONCLUSION AND FUTURE WORK

This paper studies the well-known k-MST problem. A new approach using SA with a restart strategy is developed. In order to check the performance of the proposed method we compare the results obtained with those of other metaheuristics, the computational experiments show that the proposed algorithm is highly efficient in case of small graphs, and should be improved in case of large graphs. In our future work we will incorporate the SA method by intensification and diversification strategies and also we will couple it with other metaheuristics in order to build a hybrid approach that can tackle large and other classes of graphs.

References

A. Dowsland, K. (1995). Simulated annealing. In *Modern Heuristic Techniques for Combinatorial Problems* (Colin R. Reeves ed.). McGraw-Hill.

Alrefaei, M., A. Diabat, A. Alawneh, R. Al-aomar, and M. N. Faisal (2013). *Simulated Annealing for Multi Objective Stochastic Optimization*, Volume 2.

Blum, C. and M. J. Blesa (2005). New metaheuristic approaches for the edge-weighted k-cardinality tree problem. *32*(6), 1355–1377.

Cheung, S. Y. and A. Kumar (1994). Efficient quorumcast routing algorithms. In *13th Proceedings IEEE INFOCOM '94. Networking for Global Communications*, Volume 2, pp. 840–847.

Dreo, J., A. Petrowski, P. Siarry, and E. Taillard (2003). *Métaheuristiques pour l'optimisation difficile*. Algorithmes. EYROLLES.

Fischetti, M., H. W. Hamacher, K. Jörnsten, and F. Maffioli (1994). Weighted k-cardinality trees: Complexity and polyhedral structure. *24*(1), 11–21.

Freitag, J. (1993). *Minimal k-cardinality trees*. Master thesis.

Ghandeshtani, K. S., N. Mollai, S. M. H. Seyedkashi, and M. M. Neshati (2010). New simulated annealing algorithm for quadratic assignment problem. pp. 7.

Hamacher, H. W., K. Jörnsten, and F. Maffioli (1991). Weighted k-cardinality trees, technical report. *Politecnico di Milano, Dipartimento di Elettronica, Italy.*

Katagiri, H., T. Hayashida, I. Nishizaki, and Q. Guo (2012). A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems. *39*(5), 5681–5686.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. *220*(4598), 671–680.

Lam, Y. M., K. H. Tsoi, and W. Luk (2013). Parallel neighbourhood search on many-core platforms. *8*(3), 281–293.

Liu, L., H. Mu, J. Yang, X. Li, and F. Wu (2014). A simulated annealing for multi-criteria optimization problem: DBMOSA. *14*, 48–65.

Pop, P., C. Sabo, C. Sitar, M. V Cr, and Aciun (2007). Solving the generalized minimum spanning tree problem with simulated annealing. *16*, 42–53.

Ravi, R., R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi (1996). Spanning trees short or small. *9*(2), 178–200.

Robini, M. C. and P.-J. Reissman (2013). From simulated annealing to stochastic continuation: a new trend in combinatorial optimization. *56*(1), 185–215.

Yu, V. F., A. P. Redi, Y. A. Hidayat, and O. J. Wibowo (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *53*, 119–132.

Résumé

Dans ce papier nous considérons le problème d'Arbre Couvrant Minimal de cardinalité k (k-MST) qui généralise le fameux problème de l'arbre couvrant minimal qui est l'un des classes majeurs d'optimisation combinatoire. La modélisation de ce problème a été présentée au début par Hamacher et al. (1991), et il a été démontré par la suite que le problème de k-MST est un problème NP-complet, et c'est pourquoi il est nécessaire de développer des approches approximatives basées sur des métaheuristiques pour résoudre ce problème dans un temps polynomial. Plusieurs approches ont été proposées dans la littérature pour aborder ce sujet. Blum and Blesa (2005) ont proposés trois métaheuristiques: les algorithmes évolutionnaires, les algorithmes de colonies de fourmis et la recherche tabou. Un algorithme hybride combinant TS et ACO est fourni par Katagiri et al. (2012). Dans ce papier nous avons proposé une nouvelle approche basée sur l'algorithme du recuit simulé. Les expérimentations numériques réalisées sur des instances de graphes de la bibliothèque KCTLIB proposés par Blum and Blesa (2005), comparés avec les résultats présentés dans Blum et Blesa (2005) et Katagiri et al. (2012), ont montré l'efficacité de notre approche.