Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol II
IMECS 2009, March 18 - 20, 2009, Hong Kong

A Tabu Search Based Approximate Solution Algorithm for k-minimum Spanning Tree Problems

Hideki Katagiri, Ichiro Nishizaki, Tomohiro Hayashida and Jun Ishimatsu *

Abstract—**This paper considers k-minimum spanning tree problems. An existing solution algorithm based on tabu search includes an iterative procedure of applying an exact solution method to obtain a local optimal solution. This article provides a new tabu search based approximate solution method that does not iteratively solve minimum spanning tree problems. Results of numerical experiments show that the proposed method provides a good performance in terms of accuracy over those of existing methods.**

Keywords: *k-minimum spanning tree, tabu search, k-cardinality, approximation solution method*

1 Introduction

A k-minimum spanning tree (k-MST) problem is one of combinatorial optimization problems formulated in networks, and the objective of the problem is to find a subtree with exactly k edges, called k-subtree, such that the sum of the weights attached to edges is minimal. The k-MST problem is a generalized version of MST problems; when $k = |V| - 1$ where V is a cardinality of vertices in a graph, k-MST problem corresponds to the MST problem. The wide varieties of decision making problems in the real world can be formulated as k-MST problems, and many articles discuss the applications of k-MST problems to various decision making problems: telecommunications [8], facility layout [6], open pit mining [17], oil-field leasing [10], matrix decomposition [2, 3] and quorum-cast routing [4].

The k-MST problem was firstly introduced by Hamacher *et al.* [10] in 1991. Since the k-MST problem is NP-hard [5, 14], it is difficult to solve large-scale problems within a practically acceptable time. Therefore, it is very important to construct solving methods which quickly obtain a near optimal solution.

As for existing approximate solution methods for k-MST problems, Blum *et al.* [1] proposed several metaheuristic approaches which includes evolutionary computation, ant colony optimization and tabu search. Urošević *et al.* [18] provided Variable Neighborhood Search (VNS). Recently, Katagiri *et al.* [12] developed a solution method which uses a combination of tabu search and exact solution method. They showed that their method provides a better performance than existing methods for dense graphs with high cardinality k through some numerical experiments.

In this paper, we propose a new tabu search based approximate solution method which is combined use of tabu search and exact solution algorithm for local search. Our local search algorithm obtains local optimal solutions of k-MST problem without solving MST problems iteratively unlike the existing method by Katagiri *et al.*. In order to demonstrate efficiency of the proposed solution method, we compare the performances of the proposed method with those of several existing methods.

2 Problem formulation

Given that a graph $G = (V, E)$ where V is a set of vertices and E is a set of edges, k-subtree T_k is defined as

$$T_k \in G,\ k \leq |V| - 1.$$

Then a k-minimum spanning tree problem is formulated as

$$\begin{array}{ll} \text{minimize} & \sum_{e \in E(T_k)} w(e) \\ \text{subject to} & T_k \in \mathcal{T}_k, \end{array}$$

where $\mathcal{T}_k$ is the set of all k-subtree T_k in G, $E(T_k)$ denotes the edges of T_k and $w(e)$ is a weight attached to an edge e. The aim of this problem is to seek a k-subtree with minimum sum of weights. If the problem size is small, the problem can be easily solved by finding the k-subtree with the minimum sum of weights after enumerating all possible k-subtrees in a given graph.

Even if the size of problem is not so large, it can be solved by some exact solution algorithm. As for exact solution algorithms for k-MST problems, a branch and bound method [4] and a branch and cut algorithm [7] have been developed and implemented.

*H. Katagiri, I. Nishizaki, T. Hayashida and J. Ishimatsu are with Graduate School of Engineering, Hiroshima University, 1-4-1 Kagamiyama, Higashi-Hiroshima, 739-8527 Japan (phone: +81-82-424-7709; fax: +81-82-422-7195; email: {katagiri-h, nisizaki, hayashida, ishimatsu}@hiroshima-u.ac.jp)

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 also for points in the plane [14]. Therefore, it is impossible to solve large-scale problems within a practically acceptable time even if efficient exact solution methods are applied.

Therefore, it is important to construct not only exact solution methods but also efficient approximate solution methods. Metaheuristic approaches such as genetic algorithms are useful for getting an approximate optimal solution. Blum *et al.* [1] proposed three metaheuristic approaches to k-MST problems, namely, evolutionary computation, tabu search and ant colony optimization. They compared their performances through benchmark instances [13] and showed that the performance of their metaheuristics depends not only on the instances but also on the cardinality k. For example, an ant colony optimization approach is the best for relatively small ks, whereas a tabu search approach has an advantage for large ks in terms of accuracy.

Recently, Katagiri *et al.* [12] proposed a tabu search based approximate solution method which includes a procedure of iteratively solving MST problems for variable subgraphs. They showed that their algorithm has a better performance in terms of accuracy in comparison with those of existing methods for dense graph with large ks.

3 Summary of tabu search

Tabu search (TS) [9] is one of metaheuristics which is the extended exploration method based on local search. In order to make this paper self-contained, this section devotes describing the outline of tabu search.

Let $\boldsymbol{x}^c$ be a current solution on search space. Local search generally finds a better solution $\boldsymbol{x}' \in N(\boldsymbol{x}^c)$ where $N(\boldsymbol{x}^c)$ is a neighborhood of $\boldsymbol{x}^c$. For simplicity, suppose that $\boldsymbol{x}^c$ is a local minimum solution and that the next solution $\boldsymbol{x}'$ is selected as the best solution among $N(\boldsymbol{x}^c)$. If the local search is applied for $\boldsymbol{x}'$, then $\boldsymbol{x}'$ is moved back to $\boldsymbol{x}^c$ because $\boldsymbol{x}^c$ is the best solution among a neighborhood $N(\boldsymbol{x}')$. In this way, cycling among solutions easily occurs around local minima. In order to avoid such cycling, TS algorithms use a short-term memory. The short-term memory is implemented as a set of tabu lists that store solution attributes. Attributes usually refer to components of solutions, moves, or differences between two solutions. Tabu lists prevent the exploration from returning to lately visited solutions or search domain.

Aspiration criteria permit a part of moves in the tabu list to cancel any tabu status. The typical aspiration criterion is to accept a tabu move if it updates the current best objective function value. The outline of TS is as follows:

Step 1 Generate an initial solution $\boldsymbol{x}$ and initialize a set of tabu lists TL.

Step 2 Find the best solution $\boldsymbol{x}' \in N(\boldsymbol{x})$ such that $\boldsymbol{x}' \notin TL$, and set $\boldsymbol{x} := \boldsymbol{x}'$.

Step 3 Stop the algorithm if termination conditions are satisfied. If not, then update TL and return to Step 2.

In Step 2, tabu lists generally memorize solution attributes. A tabu tenure, which is the period of time when each of tabu moves is in the tabu lists, affects the behaviors and performances of the algorithm.

In Step 3, it is checked whether the algorithm satisfies termination conditions. The termination condition is usually related to the iteration number of the algorithm and/or the iteration number of not updating the current best solution. If the termination conditions are not satisfied, then tabu lists TL are updated; new solution attributes are added in the lists, and some solution attributes are deleted if the iteration number that the attributes are tabu move is beyond their tabu tenure.

4 Proposed algorithm

In this section, we briefly explain our tabu search based algorithm. The most important feature of the proposed algorithm is that it does not include a MST algorithm iteratively for a subgraph with exactly $k+1$ vertices unlike the solution method by Katagiri *et al.* [12]. When minimum spanning tree algorithms is applied for a fixed subgraph in the original graph, the obtained solution is regarded as a local minimum of k-MST problem. In this sense, MST algorithms is worth using for local search. However, there are many cases where it does not need to use MST algorithms in order to find a local optimal solution. For example, when the number of candidates for deleting and adding edges is only one, applying MST algorithms is too much cost for obtaining a local minimum. Therefore, in this paper, we consider a new local search algorithm which moves the current solution to a local minimum solution without using MST algorithms.

Fig. 1 shows the flowchart of our algorithm.

Our algorithm begins at selecting a node at random. Then, continue to apply the Prim method, which is one of the exact solution algorithms for MST problems, until a k-subtree is constructed.

4.1 Local search

Let $V(T_k)$ be a set of vertices which compose a k-subtree T_k.

One of the most important key to success is that we efficiently construct a neighborhood of T_k, denoted by

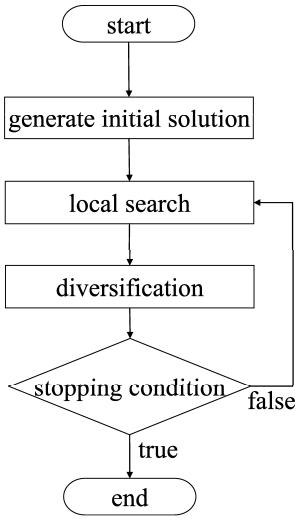

Figure 1: Flowchart of our algorithm

$N(T_k)$. Since the algorithm is a little complex, we explain the algorithm together with a simple example (see Fig. 2).

The algorithm for constructing elements in neighborhood

Step 1 For a current solution (see Fig. 1), construct $E^1_{add}(v) =: \{(v, v') \in E(G) \mid v \in V(T_k), v' \notin V(T_k)\}$ and find $e^1_{\min} = \mathrm{argmin}\{w(e) \mid e \in E^1_{add}(v)\}$. Construct $T^{ex}_{k+1} := (V(T_k) \cup v_{add}, E(T_k) \cup e^1_{\min})$ such that $e^1_{\min} = (v, v_{add})$, $v \in V(T_k), v_{add} \notin V(T_k)$. Go to Step 2.

Step 2 Update $F^1_{add} \leftarrow F^1_{add} \backslash e^1_{\min}$. If $F^1_{add} = \{\phi\}$, then go to Step 4. Otherwise, find $e^1_{\min} := \mathrm{argmin}\{w(e) \mid e \in E^1_{add}(v)\}$ and update $T^{ex}_{k+1} \leftarrow (V(T_k) \cup v_{add}, E(T_k) \cup e^1_{\min})$ such that $e^1_{\min} = (v, v_{add})$, $v \in V(T_k), v_{add} \notin V(T_k)$. Go to Step 3.

Step 3 Let E_{loop} be a set of edges which compose a loop. Find $e_{\max} := \mathrm{argmax}\{w(e) \mid e \in E_{loop}\}$ and update $E(T^{ex}_{k+1}) \leftarrow E(T^{ex}_{k+1}) \backslash e_{\max}$ (see Fig. 3). Go to Step 4.

Step 4 Delete a node $v_{del} \in V(T^{ex}_{k+1}) \backslash v_{add}$ (see Fig. 4). Update $V(T^{new}_k) \leftarrow V(T^{ex}_{k+1}) \backslash v_{del}$ and $E(T^{new}_k) \leftarrow E(T^{ex}_{k+1}) \backslash \{(v, v_{del}) \mid v \in V(T^{ex}_{k+1})\}$. Go to Step 5.

Step 5 Construct $E^2_{add} := \{(v_i, v_j) \in E(G) \mid v_i, v_j \in V(T^{new}_k), (v_i, v_j) \notin E(T^{cur}_k), i \neq j\}$, and go to Step 6.

Step 6 Find $e^2_{\min} := \mathrm{argmin}\{w(e) \mid e \in E^2_{add}\}$ to the current solution. Update $E(T^{new}_k) \leftarrow E(T^{new}_k) \cup e^2_{\min}$ and $E^2_{add} \leftarrow E^2_{add} \backslash e^2_{\min}$. Go to Step 7.

Step 7 If T^{new}_k is a tree, then terminate the algorithm. Otherwise, return to Step 6.

The local search finds a k-subtree whose objective function value is the best, namely,

$$T^{NH_{best}}_k = \mathrm{argmin}\{f(T^{NH}_k) | T^{NH}_k \in N(T_k)\}.$$

If the pair of two nodes which are candidates for being exchanged are not in tabu lists, we regard such transitions as feasible transitions. Otherwise, we search $N(T_k)/T^{NH_{best}}_k$ so as to find a feasible transition. This procedure is continued until a feasible transition is obtained.

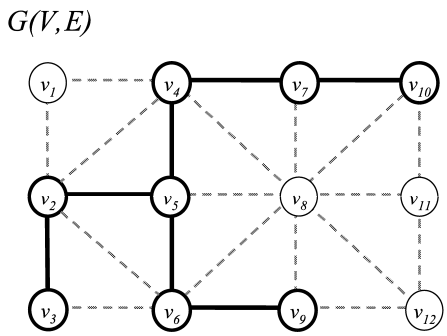

Figure 2: Example of $|V| = 12$, $|E| = 23$, $k = 7$ (Bold lines are edges which form the current solution)

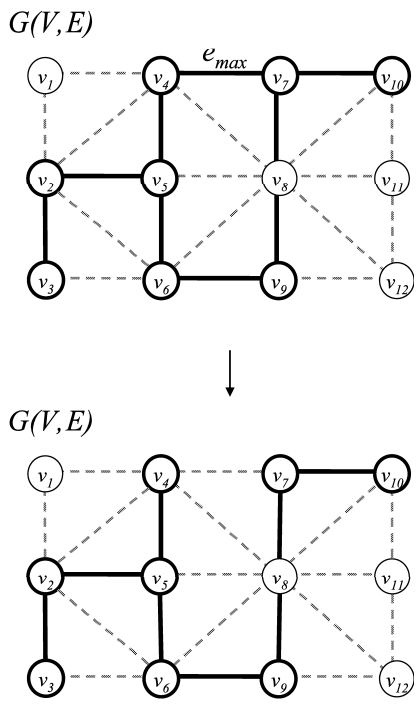

Figure 3: Example of deleting the maximum weighted edge e_{max} in a loop ($v_{add} = v_8$)

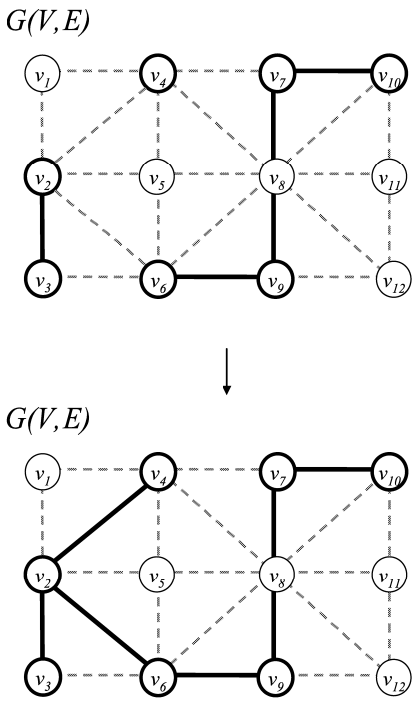

Figure 4: Example of deleting the edge $v_{del} = v_5$

In this paper, we use two tabu lists *InList* and *OutList*, which keep the induces of removed edges and to store that of added edges, respectively. Tabu tenure, denoted by θ, is a period for which it forbids edges in the tabu lists from deleting or adding. At the beginning, $InList$ and $OutList$ are set empty. A parameter θ is set the default value. Let nic_{max} and θ_{inc} be given parameters. If the current best solution is not updated nic_{max} times, then we regards this situation as cycling and increase tabu tenure using (1).

$$\theta \leftarrow \theta + \theta_{inc} \tag{1}$$

If the period of being tabu move is beyond θ_{max}, terminate the local search algorithm.

The tabu status of an attribute can be revoked if it leads to a solution with smaller cost than that of the best solution identified having that attribute. The aspiration level γ_e of an attribute is initially set equal to the cost of the initial solution T^{int} if edge e belongs to this solution, and to ∞ otherwise. At every iteration, the aspiration level of each attribute $e \in E(T_k)$ of the current solution is updated to $min\{\gamma_e, f(T_k)\}$, where $f(T_k)$ stands for the cost value of solution T.

4.2 Diversification strategy and stopping condition

A diversification procedure, using the residence frequency memory function, will lead to the exploration of region of the solution space not previously visited. The residence frequency memory records the number of times a specific element has been a part of the solution.

Frequency-based memory is one of the long-term memories and consists of gathering pertinent information about the search process so far. In our algorithm, we use residence frequency memory, which keeps each track of the number of iterations where vertices have been explored.

The diversification procedure begins at selecting a node at random. Then, we continue to apply the Prim method, which is one of the solution algorithm for minimum spanning tree problems, until k-subtree is obtained.

$$w_d(e) \leftarrow \frac{Freq(e)}{t} \times w(e) \tag{2}$$

where t is the current iteration number, and $Freq(e)$ is the number of searching e.

In this paper, we give the stopping condition as a very simple one; if the iteration number is beyond a given value, then terminate the algorithm.

5 Numerical experiments

In order to compare the performances of our method with those of representative existing solution algorithms, we solve the benchmark instances provided by Blum [13] and our new instances. Tables 1 and 2 show the results for instances by Blum [13] and our new instances, respectively.

We use C as the programming language and compiled all software with C-Compiler: Microsoft Visual C++ 7.1. All the metaheuristic approaches were tested on a PC with Celeron 3.06GHz CPU and RAM 1GB under Microsoft Windows XP. In the tables shown, RTSK, TSK and TSB represent tabu search approaches by this paper, Katagiri *et al.* and Blum *et al.*, respectively. We executed each method in 30 runs and computed the *best*, *mean* and *worst* objective function values for each method.

Table 1 shows that the performance of the proposed method is clearly better than that of the existing method by Katagiri *et al.*. Also, our algorithm provides better performance than the method by Blum *et al.*, for high cardinality k, whereas the performance of the method by Blum *et al.* is the best for low cardinality k.

Table 2 shows the results for new instances which are more dense than the existing benchmark instances. It is observed from Table II that our algorithm provides better performance than the methods by Katagiri *et al.* and Blum *et al.* except for the best objective value of one instance.

Table 1: Experimental results for existing benchmark instances

graph	k		Objective value		
			RTSK	TSK	TSB
$\|N\| = 1000$	200	best	3683	4098	**3609**
$\|E\| = 1250$		mean	3762.4	4373.2	**3685.9**
$d = 2.5$		worst	3933	4587	**3771**
$\sigma(d) = 1.57$	400	best	9291	9936	**8976**
		mean	9512.7	10088.0	**9091.0**
		worst	9610	10202	**9301**
	600	best	**16233**	17243	16282
		mean	**16259.0**	17320.1	16323.7
		worst	**16267**	17330	16454
	800	best	**26516**	27170	26552
		mean	**26525.1**	27172.9	26687.3
		worst	**26573**	27173	26755
	900	best	**32981**	**32981**	33147
		mean	**32981.0**	33284.0	33187.6
		worst	**32981**	33459	33233
$\|N\| = 400$	80	best	1473	1478	**1466**
$\|E\| = 800$		mean	1488.7	1562.4	**1477.5**
$d = 4.00$		worst	1512	1627	**1500**
$\sigma(d) = 0.00$	160	best	3233	3361	**3217**
		mean	3241.7	34522.5	**3240.0**
		worst	3261	3449	**3259**
	240	best	**5212**	5270	5215
		mean	**5213.8**	5432.5	5224.6
		worst	**5224**	5531	5234
	320	best	**7682**	7684	**7682**
		mean	**7682.3**	7697.9	7682.9
		worst	7691	7719	**7684**
	360	best	**9249**	9256	9250
		mean	**9249**	9257.8	9257
		worst	**9249**	9259	9260
$\|N\| = 1000$	200	best	1060	1130	**1047**
$\|E\| = 5000$		mean	1080.1	1166.5	**1063.7**
$d = 10.0$		worst	1115	1225	**1078**
$\sigma(d) = 3.22$	400	best	2540	2682	**2499**
		mean	2569.8	2698.9	**2535.9**
		worst	**2604**	2725	**2604**
	600	best	4529	4681	**4516**
		mean	**4539.4**	4705.6	4548.6
		worst	**4551**	4718	4603
	800	best	7286	7405	**7281**
		mean	**7286**	7418.9	7324.7
		worst	**7286**	7434	7405
	900	best	**9256**	9375	9291
		mean	**9256**	9375.0	9323.9
		worst	**9256**	9376	9372
$\|N\| = 450$	90	best	**135**	138	**135**
$\|E\| = 8168$		mean	138.3	145.1	**136.7**
$d = 36.30$		worst	141	155	**140**
$\sigma(d) = 16.83$	180	best	**337**	349	**337**
		mean	**337.6**	352.2	346.5
		worst	340	**356**	434
	270	best	**630**	643	**630**
		mean	**631.5**	649.2	653.4
		worst	**638**	654	728
	360	best	**1060**	1062	**1060**
		mean	**1060.1**	1062	1098.7
		worst	**1062**	1070	1158
	405	best	**1388**	1389	1391
		mean	**1388.0**	1389.4	1410.8
		worst	**1389**	1390	1467

Table 2: Experimental results for new benchmark instances

graph	k		Objective value		
			RTSK	TSK	TSB
$\|N\| = 300$	60	best	**541**	546	554
$\|E\| = 20000$		mean	**548.0**	572.0	628.8
$d = 133.3$		worst	**553**	606	2180
$\sigma(d) = 36.57$	120	best	**1218**	1267	1229
		mean	**1228.3**	1304.4	1613.4
		worst	**1240**	1384	3189
	180	best	**2164**	2241	2169
		mean	**2166.1**	2256.5	2838.9
		worst	**2195**	2264	4517
	240	best	**3564**	**3564**	3566
		mean	**3564.0**	3568.8	4313.7
		worst	**3564**	3581	5906
	270	best	**4690**	**4690**	**4690**
		mean	**4690**	**4690**	5326.3
		worst	**4690**	**4690**	6635
$\|N\| = 300$	60	best	**351**	354	357
$\|E\| = 30000$		mean	**354.6**	359.8	503
$d = 200.00$		worst	**355**	364	2552
$\sigma(d) = 38.99$	120	best	**869**	897	877
		mean	**875.6**	912.4	1228.2
		worst	**898**	922	3047
	180	best	**1645**	1661	1653
		mean	**1646.1**	1746.6	2194.9
		worst	**1652**	1784	3947
	240	best	**2737**	**2737**	2740
		mean	**2737.0**	2753.0	3290.3
		worst	**2737**	2765	5019
	270	best	**3491**	**3491**	**3491**
		mean	**3491**	**3491**	4322.2
		worst	**3491**	**3491**	5730
$\|N\| = 300$	60	best	232	**224**	**224**
$\|E\| = 40000$		mean	**234.4**	238.1	405.4
$d = 266.67$		worst	**238**	257	2430
$\sigma(d) = 24.61$	120	best	**541**	547	554
		mean	**550.3**	567.4	1075.3
		worst	**557**	589	2787
	180	best	**984**	1031	986
		mean	**985.1**	1053.5	1653.1
		worst	**988**	1066	3271
	240	best	**1631**	1656	1647
		mean	**1633.9**	1659.3	2493.3
		worst	**1634**	1676	3939
	270	best	**2106**	2107	2107
		mean	**2106**	2108.5	2845.5
		worst	**2106**	2109	4440

6 Conclusion

In this paper, we have proposed a new solution method based on tabu search for k-minimum spanning tree problems and compared the performance of the proposed method with those of existing methods through numerical experiments for several benchmark instances. It has been shown that the proposed method has provided better performances than the existing methods. In the near future, we will provide additional benchmark instances such as random graphs, geometric graphs or small- world graphs, and execute more numerical experiments to clarify the advantage of our method.

References

[1] Blum, C., Blesa, M.J., "New metaheuristic approaches for the edge-weighted k-cardinality tree

problem," *Computers & Operations Research*, Vol. 32, pp. 1355–1377, 2005.

[2] Borndorfer, R., Ferreira, C., Martin, A., "Matrix decomposition by branch- and-cut," *Technical Report*, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1997.

[3] Borndorfer, R., Ferreira, C., Martin, A., "Decomposing matrices into blocks," *SIAM Journal on Optimization*, Vol. 9, No. 1, pp. 236–269, 1998.

[4] Cheung, S.Y., Kumar, A., "Efficient quorum-cast routing algorithms," *Proceedings of INFOCOM*, Los Alamitos, USA, Silver Spring, MD: IEEE Society Press, 1994.

[5] Fischetti, M., Hamacher, H.W., Jörnsten, K., Maffioli, F., "Weighted k-cardinality trees: complexity and polyhedral structure, *Networks*, Vol. 24, pp. 11–21, 1994.

[6] Foulds, L.R., Hamacher, H.W., Wilson, J., "Integer programming approaches to facilities layout models with forbidden areas," *Annals of Operations Research*, Vol. 81, pp. 405–417, 1998.

[7] Freitag, J., *Minimal k-cardinality trees*, Master thesis, Department of Mathematics, University of Kaiserslautern, Germany, 1993.

[8] Garg, N., Hochbaum, D., "An $O(\log k)$ approximation algorithm for the k minimum spanning tree problem in the plane," *Algorithmica*, Vol. 18, No.1, pp. 111–121, 1997.

[9] Glover, F., Laguna, M., *Tabu Search*, Dordrecht: Kluwer Academic Publishers, 1997.

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

[11] Jörnsten, J., Løkketangen, A., Tabu search for weighted k-cardinality trees, *Asia-Pacific Journal of Operational Research*, Vol. 14, No.2, 9–26, 1997.

[12] Katagiri, H., Sakawa, M., Nishizaki, I., Kato K., Hayashida, T., "A tabu search algorithm for k-minimum spanning tree problems," *Proceedings of SCIS & ISIS 2006*, pp. 1524–1529, 2006.

[13] KCTLIB. http://iridia.ulb.ac.be/~cblum/kctlib/, 2003.

[14] Marathe, M.V., Ravi, R., Ravi, S.S., Rosenkrantz, D.J., Sundaram, R., "Spanning trees short or small," *SIAM Journal on Discrete Mathematics*, Vol. 9, No.2, pp. 178–200, 1996.

[15] Mladenović, N., Hansen, P., "Variable Neighborhood Search," *Computers & Operations Research*, Vol. 24 , pp. 1097–1110, 1997.

[16] Mladenović, N., Urošević, D., "Variable neighborhood search for the k-cardinality tree," *Proceedings of Fourth Metaheuristics International Conference*, pp. 743–747, 2001.

[17] Philpott, A.B., Wormald, N.C., *On the optimal extraction of ore from an open-cast mine*, New Zealand: University of Auckland, 1997.

[18] Urošević, D., Brimberg, J., Mladenović, N., "Variable neighborhood decomposition search for edge weighted k-cardinality tree problem," *Computers Operations Research*, Vol. 31, pp. 1205–1213, 2004.