A Hybrid Algorithm Based on Tabu Search and Ant Colony Optimization for k-Minimum Spanning Tree Problems

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

Graduate School of Engineering, Hiroshima University,
1-4-1, Kagamiyma, Higashi-Hiroshima, 739-8527, Japan
katagiri-h@hiroshima-u.ac.jp
http://www.hil.hiroshima-u.ac.jp/

Abstract. This paper considers an efficient approximate algorithm for solving k-minimum spanning tree problems which is one of the combinatorial optimization in networks. A new hybrid algorithm based on tabu search and ant colony optimization is provided. Results of numerical experiments show that the proposed method updates some of the best known values and that the proposed method provides a relatively better performance with solution accuracy over existing algorithms.

Keywords: k-minimum spanning tree, tabu search, ant colony optimization, hybrid algorithm, approximate solution.

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 minimum spanning tree (MST) problems; when $k = |V| - 1$ where V is a cardinality of vertices in a graph, the 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, e.g. telecommunications [10], facility layout [8], open pit mining [16], oil-field leasing [12], matrix decomposition [2,3] and quorum-cast routing [4].

The k-MST problem was firstly introduced by Hamacher *et al.* [12] in 1991. Since the k-MST problem is NP-hard [7,15], it is difficult to solve large-scale problems within a practically acceptable time. Therefore, it is very important to construct approximate solution methods which quickly find a near optimal solution. Blum and Blesa [1] proposed several approximate solution methods including metaheuristics such as evolutionary computation, ant colony optimization and tabu search.

In this paper, we propose a new hybrid approximate solution algorithm based on tabu search and ant colony optimization. In order to demonstrate efficiency of the proposed solution method, we compare the performances of the proposed method with those of existing algorithms.

V. Torra, Y. Narukawa, and M. Inuiguchi (Eds.): MDAI 2009, LNAI 5861, pp. 315–326, 2009.

2 Problem Formulation and Existing Solution Methods

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-MST problem is formulated as

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

where $\mathcal{T}_k$ is the set of all possible k-subtrees 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 above problem is to seek a k-subtree with the minimum sum of weights, called k-MST. If the problem size is small, the problem can be easily solved by finding an optimal solution after enumerating all possible k-subtrees in a given graph. If the size of problem is not so large, it can be solved by some exact solution algorithm such as a branch and bound method [4] and a branch and cut algorithm [9].

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 [15]. Therefore, it is important to construct not only exact solution methods but also efficient approximate solution methods.

Blum and Blesa [1] proposed three approximate solution algorithms for k-minimum spanning tree problems which are based on evolutionary computation, tabu search and ant colony optimization. They compared their performances through benchmark instances [14] and showed that an ant colony optimization approach is the best for relatively small ks, whereas a tabu search approach has an advantage for large ks with respect to solution accuracy.

In this paper, we propose an efficient hybrid algorithm based on tabu search and ant colony optimization by combining the desirable features of both algorithms.

3 Proposed Algorithm

The outline of the proposed algorithm is as follows:

Step 1 (Generation of an initial solution). For a node selected at random, the application of Prim method is continued until a k-subtree is constructed. Let the obtained k-subtree be an initial solution and the current solution T_k^{cur}.

Step 2 (Initialization of parameters). Initialize the tabu lists and the values of parameters such as tabu tenure tl_{ten} and aspiration criterion levels.

Step 3 (Tabu search-based local search). Search the neighborhood based tabu search, and store a set of local minimum solutions. If the current tabu tenure tl_{ten} is greater than tt_{max}, go to Step 4. Otherwise, return to Step 2.

Step 4 (Ant colony optimization-based intensification procedure). Explore the promising region intensively based on ant colony optimization.

Step 5 (Terminal condition). If the current computational time is greater than $TimeLimit$, terminate the algorithm. Otherwise, return to Step 2.

Let T_k^{cur}, T_k^{gb} and T_k^{lb} be the current solution, the best found solution and local optimum solution, respectively. Then, we describe the details on the procedures in Steps 3 and 4.

3.1 Tabu Search-Based Local Search

For a set $V(T_k)$ of nodes included in k-subtree T_k, we define

$$V_{NH}(T_k) := \{v | \{v, v'\} \in E(G), v \notin V(T_k), v' \in V(T_k)\}.$$

Let T_k^{NH} be a local minimum solution of k-subtree obtained by adding $v_{in} \in V_{NH}(T_k)$ to T_k and deleting $v_{out} \in V(T_k)$. Then, the neighborhood of T_k denoted by $NH(T_k)$ is defined as a whole set of possible T_k^{NH} in G.

In the proposed local search algorithm, the next solution through transition is selected as the k-subtree that has the best objective function value of all solutions $T_k^{NH} \in N(T_k^{cur})$ as follows:

$$T_k^{NH_{best}} := \arg \min_{T_k^{NH} \in NH(T_k^{cur})} \{f(T_k^{NH})\}.$$

In order to avoid cycling among a set of some solutions, we use two tabu lists *InList* and *OutList*, which keep the induces of removed edges and added edges, respectively. A tabu tenure, denoted by θ, is a period for which it forbids edges in the tabu lists from deleting or adding. In details, at the beginning, we set an initial value of the tabu tenure tl_{ten} to tt_{min} which is the minimum tabu tenure defined as

$$tt_{min} := \min \left\{ \left\lfloor \frac{|V|}{20} \right\rfloor, \frac{|V| - k}{4}, \frac{k}{4} \right\}.$$

Let nic_{int} be the period of the best found solution T_k^{gb} not being updated. If $nic_{int} > nic_{max}$, then tabu tenure is updated as $tl_{ten} \leftarrow tl_{ten} + tt_{inc}$, where

$$nic_{max} := \max\{tt_{inc}, 100\}, \qquad tt_{inc} := \left\lfloor \frac{tt_{max} - tt_{min}}{10} \right\rfloor + 1.$$

If the current tabu tenure tl_{ten} is greater than tt_{max} defined as

$$tt_{max} := \left\lfloor \frac{|V|}{5} \right\rfloor,$$

the local search algorithm is terminated, and intensification strategy based on ant colony optimization is performed.

When checking whether the transition from the current solution to some solution in T_k^{NH} is acceptable, if an edge e in $InList$ or $OutList$, which is related to the transition as the added edge or deleted edge, satisfies the condition $\gamma_e > f(T_k^{NH})$, then the transition is permitted. The parameter γ_e called *aspiration criterion level* is given to all of edges and is initially set to

$$\gamma_e = \begin{cases} f(T_k^{cur}), & e \in E(T_k^{cur}) \\ \infty, & e \notin E(T_k^{cur}). \end{cases} \tag{1}$$

In each explored solution T_k, γ_e is updated as $\gamma_e \leftarrow f(T_k)$ for every $e \in E(T_k)$.

The following is the details on the proposed local search algorithm.

[Tabu search-based local search algorithm]

Step 1(Initialization of the list of a deleted node). Let $V_{in} \leftarrow V_{NH}(T_k^{cur})$.

Step 2(Decision of a deleted node). If $V_{in} = \emptyset$, terminate the algorithm. Otherwise, go to Step 2-1.

Step 2-1. Find

$$v_{in} := \arg\min_{v \in V_{in}} \left\{ \frac{\sum_{v' \in V(T_k^{cur})} w(e)}{d(v)} \middle| e = (v, v') \right\}$$

and set $V_{in} \leftarrow V_{in} \backslash v_{in}$, where $d(v)$ is the number of edges existing between $v \in V$ and T_k^{cur}. Go to Step 2-2.

Step 2-2. Find $E_{in_1} := \{(v, v_{in}) \mid v \in V(T_k^{cur})\}$ (see Fig. 1) and $e_{\min_1} := \arg\min_{e \in E_{in_1}} \{w(e)\}$. Set $T_{k+1}^{NH} \leftarrow (V(T_k^{cur}) \cup v_{in}, E(T_k^{cur}) \cup e_{min_1})$ and $E_{in_1} \leftarrow E_{in_1} \backslash e_{\min_1}$ (see Fig. 2), and go to Step 2-3.

Step 2-3. Find $e_{\min_1} := \arg\min_{e \in E_{in_1}} \{w(e)\}$, and set $T_{k+1}^{NH} \leftarrow T_{k+1}^{NH} \cup e_{\min_1}$ and $E_{in_1} \leftarrow E_{in_1} \backslash e_{\min_1}$ (see Fig. 3). Go to Step 2-4.

Step 2-4. For a set E_{loop} of edges which compose a loop in Step 2-3, find $e_{\max} := \arg\max_{e \in E_{loop}} \{w(e)\}$ and set $T_{k+1}^{NH} \leftarrow T_{k+1}^{NH} \backslash e_{\max}$ (see Fig. 3)D

Step 2-5. If $E_{in_1} = \{\emptyset\}$, then set $V_{out} \leftarrow V(T_k^{cur})$ and go to Step 3. Otherwise, return to Step 2-4.

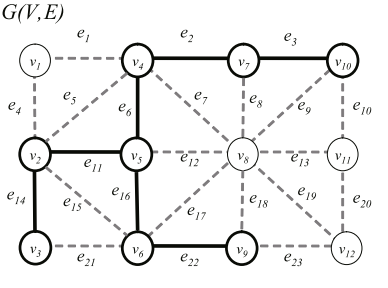

Fig. 1. Current solution (when $v_{in} = v_8$ and $E_{in_1} = \{e_7, e_8, e_9, e_{12}, e_{17}, e_{18}\}$)

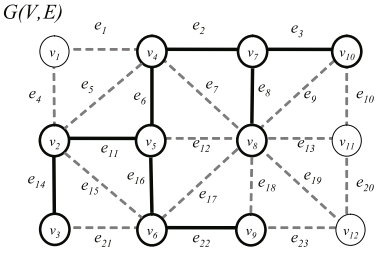

Fig. 2. $k+1$-subtree T_{k+1}^{NH} (e_8 is added to the current solution)

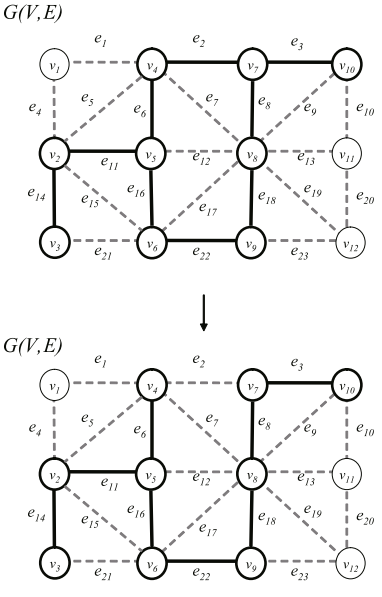

Fig. 3. Improvement of $k+1$-subtree T_{k+1}^{NH} (e_{18} is added, and then e_2 is deleted so that a new $k+1$-subtree T_{k+1}^{NH} is constructed)

Step 3(Decision of an added node for constructing T_k^{NH}). If $V_{out} = \{\emptyset\}$, then return to Step 2. Otherwise, go to Step 3-1.

Step 3-1. Find

$$v_{out} := \arg \max_{v \in V_{out}} \left\{ \frac{\sum_{v' \in V(T_k^{cur})} w(e)}{d(v)} \middle| e = (v, v') \right\}$$

and set $V_{out} \leftarrow V_{out} \backslash v_{out}$. Go to Step 3-2.

Step 3-2. Find $e_{\min}^{out} := \arg\min_{e \in \{(v_{out}, v')\}} \{w(e) \mid v' \in T_k^{cur}\}$. If $f(T_k^{NH_{best}}) < \left(\sum_{e \in E(T_{k+1}^{NH})} w(e)\right) - w(e_{\min}^{out})$ for $e_{\min}^{out}$, then return to Step 3. Otherwise, go to Step 3-3.

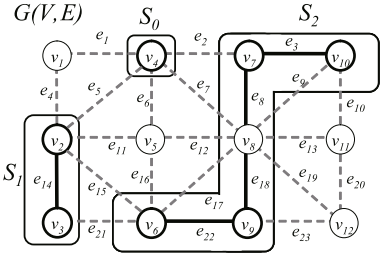

Fig. 4. Set of super-nodes

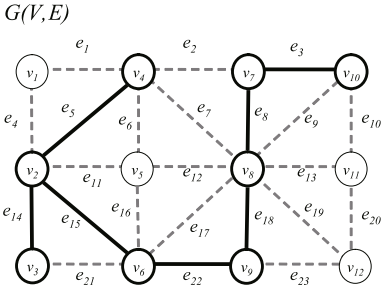

Fig. 5. Solution T_k^{NH} in neighborhood $NH(T_k)$

Step 3-3. For a set of super-node S_r, $r = 0, 1, 2 \cdots$, each of which is a connected component obtained by deleting v_{in} from T_k^{NH}, find $E_{in_2} := \{(v_i, v_j) \mid v_i \in S_k, v_j \in S_l, k \neq l\}$ (see Fig. 4). Go to Step 3-4.

Step 3-4. Find $e_{\min_2} := \arg\min_{e \in E_{in_2}} \{w(e)\}$. If $f(T_k^{NH_{best}}) < w(e_{\min_2}) + \sum_{e \in E(T_k^{NH})} w(e)$ for $e_{\min_2}$, then return to Step 3. Otherwise, go to Step 3-5.

Step 3-5. If there is no loop in $e_{\min_2} \cup T_k^{NH}$, then set $E(T_k^{NH}) \leftarrow E(T_k^{NH}) \cup e_{\min_2}$ and $E_{in_2} \leftarrow E_{in_2} \backslash e_{\min_2}$. Otherwise, set $E_{in_2} \leftarrow E_{in_2} \backslash e_{\min_2}$. Go to Step 3-6.

Step 3-6. If T_k^{NH} is a tree (see Fig. 5)Cthen set $f(T_k^{NH_{best}}) \leftarrow f(T_k^{cur})$ and return to Step 3. Otherwise, return to Step 3-4.

3.2 Ant Colony Optimization-Based Intensification Procedure

In this section, we describe the details on the ant colony optimization-based intensification procedure performed in Step 4.

Ant Colony Optimization (ACO) [5,6] is a metaheuristic approach for solving hard combinatorial optimization problems. This basic behavior is the basis for a cooperative interaction which leads to the emergence of shortest paths by depositing a substance called *pheromone* on the ground so as to minimize the length of the path between nest and food source.

In this paper, we propose an intensification algorithm based on ant colony optimization by extending the Blum-Blesa algorithm [1]. One of the characteristics in the proposed algorithm is that our algorithm deposits pheromone on the edges selected in the local optimal solutions which were obtained by the tabu search-based local algorithm. This procedure allows the proposed hybrid algorithm to intensively explore the promising region.

[**Intensification algorithm based on ant colony optimization**]

Step 1 (Setting of learning rate). Set the learning rate of each solution in E_{lb} to the value defined by

$$\rho = \begin{cases} 0.15, & cf < 0.7 \\ 0.1, & 0.7 \leq cf \leq 0.95 \\ 0.05, & cf > 0.95, \end{cases}$$

where cf is a convergence factor defined by

$$cf \leftarrow \frac{\sum_{e \in E_{lb}} \tau_e}{|E_{lb}| \cdot \tau_{max}}.$$

Step 2 (Update of pheromone). Update the amount of pheromone assigned to each edge e as follows:

$$\tau_e = f_{mmas}(\tau_e + \rho(\delta_e - \tau_e))$$

where

$$f_{mmas}(x) = \begin{cases} \tau_{min}, & x < \tau_{min} \\ x, & \tau_{min} \leq x \leq \tau_{max} \\ \tau_{max}, & x > \tau_{max} \end{cases}, \quad \delta_e = \begin{cases} 1, & e \in E_{lb} \\ 0, & e \notin E_{lb}. \end{cases}$$

Step 3 (Generation of a k-subtree). Replace the weight attached to each edge e in G by $w_d(e)$ defined as

$$w_d(e) \leftarrow \frac{w(e)}{\tau_e}.$$

Stating from a randomly selected node, a k-subtree T_k is constructed by applying the Prim method. After that, replace $w_d(e)$ by the original weight $w(e)$ and construct a k-subtree T_k^{cur} by applying the Prim method again to the subgraph whose nodes and edges are $V(T_k)$ and $E(T_k)$, respectively.

In this paper, we set the initial values of τ_e, the values of τ_{min} and τ_{max} to 0.5, 0.001 and 0.999, respectively.

4 Numerical Experiments

In order to compare the performances of our method with two of existing solution algorithms proposed by Blum and Blesa [1]. 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.

Tables 1-4 and 5 show the results for several existing instances [14] and our new instances, respectively. Bold face means that it is the best obtained value among the three algorithms to be compared. In Tables 1-4, BNV denotes the best known values which have been obtained by Blum and Blesa through their

Table 1. Grid graph [14]

Graph	k	BNV		Objective function values		
				HybridK	TSB	ACOB
$\|N\| = 225$	40	695	best	**695**	696	**695**
$\|E\| = 400$			mean	**695**	696	695.4
$\bar{d}(v) = 3.55$			worst	**695**	696	696
(bb45x5_1.gg)	80	*1552	best	**1552**	1579	1572
		(1568)	mean	**1565.1**	1592.7	1581.2
			worst	**1572**	1615	1593
	120	*2444	best	**2444**	2546	2457
		(2450)	mean	**2457.9**	2558.5	2520.3
			worst	**2465**	2575	2601
	160	*3688	best	**3688**	3724	3700
		(3702)	mean	**3688**	3724.9	3704.7
			worst	**3688**	3729	3720
	200	5461	best	**5461**	5462	**5461**
			mean	**5461**	5462.4	5469
			worst	**5461**	5463	5485
$\|N\| = 225$	40	654	best	**654**	**654**	**654**
$\|E\| = 400$			mean	**654**	**654**	**654**
$\bar{d}(v) = 3.55$			worst	**654**	**654**	**654**
(bb45x_5_2.gg)	80	1617	best	**1617**	**1617**	**1617**
			mean	1619.1	**1617.1**	1626.9
			worst	1620	**1619**	1659
	120	*2632	best	**2632**	2651	2637
		(2633)	mean	**2641.3**	2677.9	2664.6
			worst	**2648**	2719	2706
	160	3757	best	**3757**	3815	**3757**
			mean	**3764.3**	3815.0	3797.6
			worst	**3779**	3815	3846
	200	5262	best	**5262**	**5262**	**5262**
			mean	**5262**	5268.6	5272
			worst	**5262**	5296	5288

Table 2. Regular graph [14]

graph	k	BNV		Objective function values		
				HybridK	TSB	ACOB
$\|N\| = 1000$	200	3308	best	3393	3438	**3312**
$\|E\| = 2000$			mean	3453.1	3461.4	**3344.1**
$d(v) = 4$			worst	3517	3517	**3379**
(1000-4-01.g)	400	7581	best	**7659**	7712	7661
			mean	7764	7780.2	**7703**
			worst	7819	7825	**7751**
	600	12708	best	**12785**	12801	12989
			mean	12836.6	**12821.8**	13115.6
			worst	13048	**12869**	13199
	800	19023	best	19099	**19093**	19581
			mean	**19101.1**	19112.6	19718.7
			worst	**19128**	19135	19846
	900	22827	best	**22827**	22843	23487
			mean	**22827**	22859.2	23643
			worst	**22827**	22886	23739
$\|N\| = 1000$	200	3620	best	3667	3692	**3632**
$\|E\| = 2000$			mean	3697.5	3722.0	**3670.1**
$d(v) = 4$			worst	3738	3751	**3710**
(g400-4-05.g)	400	8206	best	**8323**	8358	8376
			mean	**8357.1**	8385.6	8408.3
			worst	**8424**	8415	8442
	600	13584	best	13807	**13735**	14085
			mean	13824.3	**13759.4**	14164.5
			worst	13900	**13820**	14235
	800	20076	best	**20110**	20130	20661
			mean	**20129.9**	20142.9	20811.3
			worst	**20143**	20155	20940
	900	24029	best	**24035**	24044	24782
			mean	**24035**	24052.6	24916
			worst	**24035**	24064	25037

tremendous experiment for several months. The values with $*$ denotes new best known values that are updated by the proposed algorithm. HybridK, TSB and ACOB represent the proposed algorithm, tabu search algorithm [1] and ant colony optimization algorithm [1] by Blum and Blesa. We executed each method for 30 runs under the condition that $TimeLimit = 300(s)$ and computed the *best*, *mean* and *worst* objective function values for each method. We describe '$-$' in the tables when the algorithms do not derive solutions within the given time limit.

Tables 1-4 show that the performance of the proposed method is better than those of the existing algorithms by Blum and Blesa, especially in the case of high cardinality k and high degree graphs, whereas the performance of the ant colony optimization algorithm by Blum and Blesa is the best for low cardinality

Table 3. Instances constructed from Steiner tree problems [14]

graph	k	BNV		Objective function values HybridK	TSB	ACOB
$\|N\| = 1000$	200	1018	best	**1034**	1036	1036
$\|E\| = 5000$			mean	1048.6	1047.3	**1045.9**
$\bar{d}(v) = 10.0$			worst	1063	**1056**	**1056**
(steind15.g)	400	2446	best	**2469**	2493	2665
			mean	**2480.7**	2502.5	2806.6
			worst	**2492**	2524	2928
	600	4420	best	**4426**	4442	5028
			mean	**4433**	4454.6	5398.4
			worst	**4451**	4490	5602
	800	7236	best	**7236**	7252	8457
			mean	**7236.9**	7272.8	8839.6
			worst	**7237**	7308	9006
	900	9248	best	**9256**	9283	10873
			mean	**9256**	9294.2	11166.3
			worst	**9256**	9304	11423

Table 4. Instances constructed from graph coloring problems [14]

graph	k	BNV		Objective function values HybridK	TSB	ACOB
$\|N\| = 450$	90	135	best	**135**	**135**	**135**
$\|E\| = 8168$			mean	**135.1**	135.3	135.7
$\bar{d}(v) = 36.30$			worst	137	**136**	137
(le450_15a.g)	180	336	best	**336**	337	352
			mean	**337**	337.1	374.4
			worst	**337**	338	419
	270	630	best	**630**	630	696
			mean	**630.1**	630.3	839
			worst	**631**	633	913
	360	1060	best	**1060**	**1060**	1267
			mean	**1060**	1064.1	1461.2
			worst	**1060**	1118	1566
	405	1388	best	**1388**	**1388**	1767
			mean	**1388**	1391.1	1888.7
			worst	**1388**	1392	2015

k. Table 5 shows that the proposed method is the best for instances of graphs with higher degrees than existing ones. It should be stressed here that as shown in Table 1, the proposed method updates some of best known values despite very short computational time limit (300s), while the time limits in the experiments by Blum and Blesa are fairly large, at most several hours. From these results, we can occlude that the proposed algorithm is considerably promising for solving k-minimum spanning tree problems.

Table 5. New instances

graph	k		Objective function values		
			HybridK	TSB	ACOB
$\|N\| = 500$	100	best	**1943**	1954	**1943**
$\|E\| = 15000$		mean	**1950.5**	1990.9	2022.3
$\bar{d}(v) = 60$		worst	**1966**	2023	2241
	200	best	**5037**	5063	5517
		mean	**5047.3**	5080.4	7444.4
		worst	**5066**	5221	9859
	300	best	**9758**	9821	-
		mean	**9769.6**	9922.6	-
		worst	**9795**	11696	-
	400	best	**16351**	16373	-
		mean	**16363.8**	16488	-
		worst	**16368**	17953	-
	450	best	**20929**	20934	-
		mean	**20929**	20945.2	-
		worst	**20929**	20992	-
$\|N\| = 500$	100	best	**1294**	1319	1398
$\|E\| = 30000$		mean	**1303.7**	1352.8	1743.5
$\bar{d}(v) = 120$		worst	**1321**	1385	2479
	200	best	**3064**	3150	4013
		mean	**3097.1**	3934.4	6861.4
		worst	**3127**	6032	9623
	300	best	**5312**	5380	-
		mean	**5312.5**	6471.9	-
		worst	**5318**	8308	-
	400	best	**8582**	8586	-
		mean	**8582**	9540	-
		worst	**8582**	11485	-
	450	best	**10881**	10882	-
		mean	**10881**	11300.4	-
		worst	**10881**	13570	-

5 Conclusion

In this paper, we have proposed a new hybrid approximate solution algorithm for k-MST 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 updates some of the best known values and that has provided better performances than the existing methods. We will execute more experiments to clarify the efficiency of the proposed algorithm as well as its advantages and disadvantages over the existing algorithms.

References

1. Blum, C., Blesa, M.J.: New metaheuristic approaches for the edge-weighted k-cardinality tree problem. Computers & Operations Research 32, 1355–1377 (2005)
2. Börndorfer, R., Ferreira, C., Martin, A.: Matrix decomposition by branch- and-cut. Technical Report, Konrad-Zuse-Zentrum für Informationstechnik, Berlin (1997)
3. Börndorfer, R., Ferreira, C., Martin, A.: Decomposing matrices into blocks. SIAM Journal on Optimization 9(1), 236–269 (1998)
4. Cheung, S.Y., Kumar, A.: Efficient quorum-cast routing algorithms. In: Proceedings of INFOCOM. IEEE Society Press, Los Alamitos (1994)
5. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactionson Systems, Man and Cybernetics: Part B 26(1), 29–41 (1996)
6. Dorigo, M., Gambardella, L.M.: Ant Colony System: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
7. Fischetti, M., Hamacher, H.W., Jörnsten, K., Maffioli, F.: Weighted k-cardinality trees: complexity and polyhedral structure. Networks 24, 11–21 (1994)
8. Foulds, L.R., Hamacher, H.W., Wilson, J.: Integer programming approaches to facilities layout models with forbidden areas. Annals of Operations Research 81, 405–417 (1998)
9. Freitag, J.: Minimal k-cardinality trees, Master thesis, Department of Mathematics, University of Kaiserslautern, Germany (1993)
10. Garg, N., Hochbaum, D.: An $O(\log k)$ approximation algorithm for the k minimum spanning tree problem in the plane. Algorithmica 18(1), 111–121 (1997)
11. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1997)
12. Hamacher, H.W., Jörnsten, K., Maffioli, F.: Weighted k-cardinality trees. Technical Report 91.023, Politecnico di Milano, Dipartimento di Elettronica, Italy (1991)
13. Jörnsten, J., Løkketangen, A.: Tabu search for weighted k-cardinality trees. Asia-Pacific Journal of Operational Research 14(2), 9–26 (1997)
14. KCTLIB (2003), `http://iridia.ulb.ac.be/~cblum/kctlib/`
15. Marathe, M.V., Ravi, R., Ravi, S.S., Rosenkrantz, D.J., Sundaram, R.: Spanning trees short or small. SIAM Journal on Discrete Mathematics 9(2), 178–200 (1996)
16. Philpott, A.B., Wormald, N.C.: On the optimal extraction of ore from an open-cast mine. University of Auckland, New Zealand (1997)