Fourth International Workshop on Computational Intelligence & Applications
IEEE SMC Hiroshima Chapter, Hiroshima University, Japan, December 10 & 11, 2008

# An approximate solution method based on tabu search for $k$-minimum spanning tree problems

Jun Ishimatsu, Hideki Katagiri, Ichiro Nishizaki and Tomohiro Hayashida
Graduate School of Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima City, Hiroshima, 739-8527 Japan
email:{ishimatsujun,katagiri-h,nisizaki,hayashida}@hiroshima-u.ac.jp

***Abstract*—This paper considers $k$-minimum spanning tree problems. An existing solution algorithm based on tabu search, which was proposed by Katagiri et al., includes an iterative solving procedure of minimum spanning tree (MST) problems for subgraphs to obtain a local optimal solution of $k$-minimum spanning tree problems. 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 for relatively high cardinality $k$.**

## I. Introduction

A $k$-minimum spanning tree ($k$-MST) problem is a combinatorial optimization problem to find a subtree with exactly $k$ edges, i.e., $k$-subtree, such that the sum of the weights is minimal. The $k$-MST problem was firstly introduced by Hamacher et al. [6] in 1993, and it can be applied to many real-world problems in wide variety of decision making, e.g. in telecommunications [13], facility layout [10], open pit mining [1], oil-field leasing [6], matrix decomposition [16], [17] and quoram-cast routing [18].

Since the $k$-MST problem is NP-hard [11], [12], 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, Urosevic et al. [3] provided approximate solution methods based on Variable Neighborhood Search (VNS). Blum et al. [2] also proposed several metaheuristic approaches. Recently, Katagiri et al. [5] developed a solution method which uses a combination of tabu search and an iterative solving procedure for minimum spanning tree (MST) problems. 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 with an efficient local search algorithm. Our local search algorithm obtains local optimal solutions of $k$-MST not solving MST problems iteratively. In order to demonstrate efficiency of the proposed solution method, we compare the performances of the proposed method with those of existing methods by Blum et al. and Katagiri et al.

This paper is organized as follows: Section 2 provides problem formulation. In Section 3, we introduce a summary of tabu search.

## II. 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

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

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 above 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 [18] and a branch and cut algorithm [7] have been developed and implemented.

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 [12]. Therefore, it is impossible to solve large-scale problems within a practically acceptable time even if the problems is solved by efficient exact solution methods.

Therefore, it is important to construct not only exact solution methods but also efficients approximate solution methods. Metaheuristic approaches such as genetic algorithms are useful for getting an approximate optimal solution. Blum et al. [2] proposed three metaheuristic approaches to $k$-minimum spanning tree problems, namely, evolutionary computation, tabu search and ant colony optimization. They compared their performances through benchmark instances [8] 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 $k$s, whereas a tabu search approach has an advantage for large $k$s in terms of accuracy.

Recently, Katagiri et al. [5] proposed a tabu-search-based approximate solution method which includes a procedure of

iteratively solving minimum spanning tree (MST) problems. 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 $k$s.

## III. Summary of tabu search and variable neighborhood search

### A. tabu search

Tabu search [4] is one of metaheuristics and is the extension of local search. Let $\boldsymbol{x}^c$ be a current solution. Local search generally improves the current solution because it moves from the current solution $\boldsymbol{x}^c$ to a solution $\boldsymbol{x}' \in N(\boldsymbol{x}^c)$ which is better than the current solution, where $N(\cdot)$ is a given neighborhood structure. 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 often 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 algorithm from returning to recently visited solutions.

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 leads to a new solution better than the current best solution.

The outline of TS is as follows:

**Step 1** Generate an initial solution $\boldsymbol{x}$ and initialize a tabu list $TL$.

**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 if a termination condition is satisfied. If not, then update $TL$ and return to Step 2.

In Step 2, a tabu list memorizes solution attributes. A tabu tenure, i.e., the length of the tabu list determines the behavior of the algorithm. A larger tabu tenure forces the search process to explore larger regions, because it forbids revisiting a higher number of solutions.

In step 3, it is checked whether the algorithm satisfies a termination condition. 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.

### B. variable neighborhood search

Variable neighborhood search proposed by Mladenovic and Hansen [15] is summarized as follows:

**Variable Neighborhood Search (VNS)**

*Initialization.* Select the set of neighborhood structures $\mathcal{N}_p, p = 1, \ldots, p_{max}$,that will be used in the search; find an initial solution $T$; choose a stopping condition;

*Repeat* the following sequence until the stopping condition is met:

1) Set $p \leftarrow 1$;
2) Until $p = p_{max}$, repeat the following steps:
   a) (Shaking) Generate a tree $T'$ at random from the $p^{th}$ neighborhood of $T(T' \in \mathcal{N}_p(T))$;
   b) (Local search) Apply some local search method with $T'$ as initial solution; denote with $T''$ the so obtained local optimum;
   c) (Move or not) If this local optimum is better than the incumbent, move there $(T \leftarrow T'')$, and continue the search with $\mathcal{N}_1(p \leftarrow 1)$; otherwise,set $p \leftarrow p + 1$;

## IV. Proposed algorithm

### A. Initial solution

At first, an edge $e = \{v, v'\}$ is chosen uniformly at random. With this edge, a 1-subtree $T_1$ with is generated. Then $k-1$ edges are added to the subtree so as to construct a $k$-subtree. Finally, a solution algorithm for MST problems is applied for the subgraph of which vertices are selected as the $k$-subtree. In this way, an initial solution of $k$-minimum spanning tree is obtained.

### B. Neighborhood

Let us introduce a distance $\eta(T_1, T_2)$ between any two solutions (trees with cardinality $k$) $T_1$ and $T_2$ as a cardinality of difference between their edge sets, i.e.,

$$\eta(T_1, T_2) = |V_{T_1} \setminus V_{T_2}| = |V_{T_2} \setminus V_{T_1}|$$

Note that the distance functions above may be viewed as Hamming distances if each solution is represented by $0-1$ vectors having 1 if an edge belongs to the solution $T$ and 0 otherwise. The neighborhood $\mathcal{N}_p(T_1)$ consists of all solutions (subtrees) with distance $p$ from $T_1 : T_2 \in \mathcal{N}_p(T_1) \Leftrightarrow \eta(T_1, T_2) = p$. It is clear that this function is metric.

### C. Shake

We use the procedure of Shake proposed by Mladenovic and Urosevic [14].

The distance function $\eta$ is used in our shaking step. In order to obtain $T' \in \mathcal{N}_p(T)$, the following procedure is done with $p$ cycles.

**Step 1** Choose at random a set of a deleted vertex $v_{del} \in V(T)$ and an added vertex $v_{add} \notin V(T)$.

**Step 2** Apply the **Transition Algorithm** (see Section V). If the transition is infeasible, then return to Step 1. Otherwise, terminate.

### D. Local search

We use tabu search as a local search for an initial solution which is obtained by Shaking step. The flowchart of tabu search is as follows (see Fig.1):

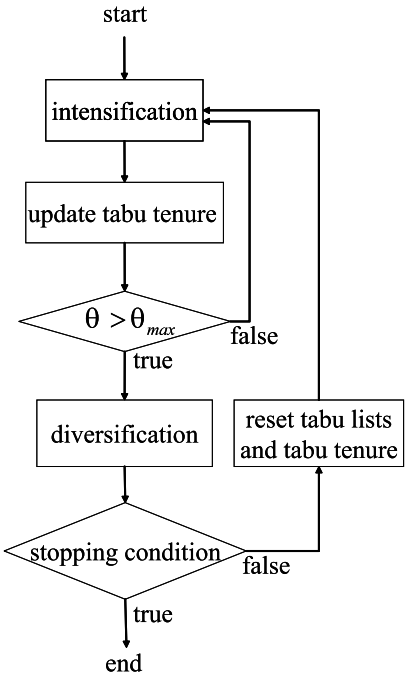


Fig. 1. Flowchart of local search

*1) Intensification:* The procedure of *Intensification* repeats transitions based on hill climbing with tabu lists and aspiration criterion:

*a) Transition strategy:* There are two major transition strategies; one is *best improvement strategy* and the other is *first improvement strategy*. The best improvement strategy is a search strategy which selects the best solution of all solutions in neighborhood as the next solution. On the other hand, the first improvement strategy selects the firstly found solution of which objective function value is better than that of the current solution. In this paper, we use the first improvement strategy because it takes much computational time for search a neighborhood if we use the best improvement strategy. These strategies are also applied when **Transition Algorithm** is used in Intensification procedure.

*b) Tabu list:* Tabu lists store the induces of the edges that were recently added or removed. As described before, every move consists of two steps; the first step is to remove one edge $e \in T^c$ from the current spanning tree $T^c$, and the second step is to add an edge in $N_p(T^c - e) \setminus \{e\}$ to $T^c - e$. The status of the forbidden moves are explained as: If a vertex $v_j$ is in the tabu list, then our algorithm forbids the addition or deletion of the vertex $v_j$. In this paper, we use two tabu lists *InList* and *OutList*, which keep the induces of removed edges and to sore that of added edges, respectively. Tabu tenure, denoted by $\theta$, is a period for which it forbids vertices in the tabu lists from deleting or adding.

*c) Aspiration criterion:* When an attribute is declared tabu, all solutions possessing this attribute are implicitly declared tabu. However, some of these solutions may have never been considered by the search. To remedy this, an aspiration criterion is defined to override the tabu status of a solution. One common aspiration criterion is to allow tabu solutions yielding better solution values than that of the best known solution.

In our TS implementation, we apply an extension of the aspiration level concept by associating an attribute to each vertex of the graph. 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 $\gamma_v$ of an attribute is initially set equal to the cost of the initial solution $T^{int}$ if vertex $v$ belongs to this solution, and to $\infty$ otherwise. At every iteration, the aspiration level of each attribute $v \in V(T)$ of the current solution is updated to $min\{\gamma_v, f(T)\}$, where $f(T)$ stands for the cost value of solution $T$.

*2) Update tabu tenure:* Let $nic_{max}$ and $\theta_{inc}$ 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).

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

*3) Diversification strategy:* 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 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 the situation that some spanning tree is formed. Let $d(T_k) = \sum_{v \in V(T_k)} Freq(v)$ denote a criterion for diversification. In a manner similar to intensification strategy as described above, **Transition Algorithm** is repeated with $k$ cycles, where $Freq(v)$ is the frequency of vertex $v$ to be searched.

*4) Reset tabu:* $InList$ and $OutList$ are set empty. A parameter $\theta$ is reset the default value.

*5) Stopping condition):* If the iteration is beyond a given value, then terminate.

## V. New transition algorithm

The most important feature of the proposed algorithm is that it does not apply a minimum spanning tree algorithm iteratively for a subgraph with exactly $k + 1$ vertices unlike the solution method by Katagiri et al. [5]. Since minimum spanning tree algorithms find an optimal spanning tree for a fixed subgraph, the obtained solution is considered as a local minimum of $k$-minimum spanning tree problem. In this sense, MST algorithms is worth using for local search. However, there are many cases where it dose not need to use MST algorithms in order to find a local optimal solution. Therefore,

in this paper, we consider a new transition algorithm which move the current solution to a local minimum solution, not using MST algorithms. As described in the previous section, this algorithm is applied for vertex transition in Shaking and any local search of Intensification and Diversification procedures.

Our transition algorithm which obtains a local optimal solution of $k$-minimum spanning tree, not using the algorithm for solving MST problems, consists of two stages. Since the algorithm is a little complex, we explain the outline of the algorithm using a simple example (see Fig.2). Let $v_{del} \in V(T_k)$ and $v_{add} \notin V(T_k)$ be the vertices selected as the deleted and added vertices, respectively.

**Transition algorithm for the first stage**

**Step 1** Let $S_t, t = 1, 2, 3, \cdots$ be a set of sub-trees which is constructed by deleting $v_{del}$. Merge each $S_t$ into a vertex, called *super-vertex* and let $S_0$ be $v_{add}$. Then construct $G'(V', E')$ according to following equation (see Fig.3):

$$\begin{aligned} V' &\leftarrow \{S_t | t = 0, 1, 2 \cdots\} \\ E' &\leftarrow \{(S_i, S_j) | (S_i, S_j) \in E, i \neq j\} \end{aligned}$$

**Step 2** Obtain a minimum spanning tree problem using some algorithm such as Prim method or Kruskul method (see Fig. 4).

**Step 3** Go to the second stage and apply the transition algorithm for the second stage.

**Transition algorithm for the second stage**

Let the dotted lines denote the edges between super-vertices (see Fig.5).

**Step 1** Find edge $e_{max}$ whose weight is the maximum from among all the edges included in super-vertices. Find edge $e_{min}$ whose weight is the minimum from among all the dotted edges.

**Step 2** If $w(e_{max}) > w(e_{min})$, then go to Step 3. Otherwise, go to Step 4.

**Step 3** Delete the edge $e_{max}$ and add the edge $e_{min}$. Go to Step 5.

**Step 4** Attach label "explored" to a set of dotted lines that connects subgraphs which are derived by deleting the edge $e_{max}$.

**Step 5** If all the dotted line is labeled "explored", then terminate. Otherwise, return to Step 1.

The above algorithm is applied for all vertex transition procedures in the subroutine of shaking and local search.

## VI. Numerical experiments

In order to compare the performances of our method with those of representative existing solution algorithms, we solve some benchmark instances which includes the instances provided by Blum [8] and our new instances. Tables I and II show the results for instances by Blum [8] and our new instances, respectively.

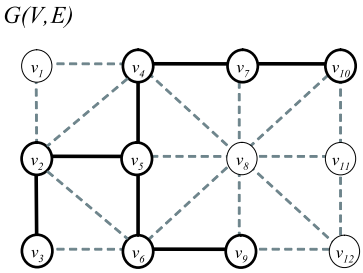


Fig. 2. Example of $|V| = 12$, $|E| = 23$, $k$ = 7 (Bold lines are edges which form the current solution)

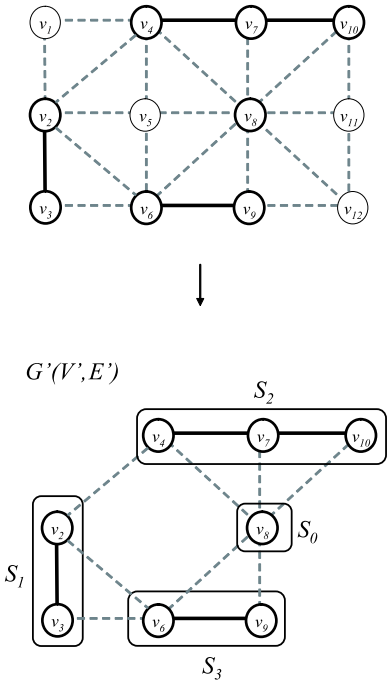


Fig. 3. Example of generating $G'$ from $G$ ($v_{del} = v_5$ and $v_{add} = v_8$)

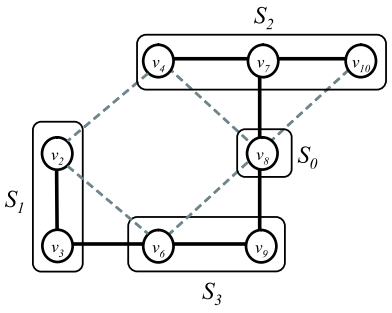


Fig. 4. Graph of super-vertices and its minimum spanning tree

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, TSI, TSK and TSB represent tabu search approaches by this paper, Katagiri et al. and Blum et

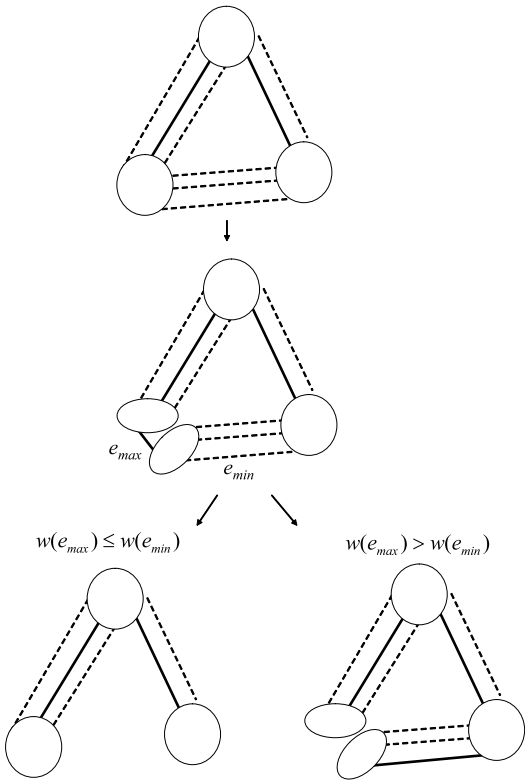


Fig. 5. Transition algorithm in the second stage

al., respectively. We executed each method in 30 runs and computed the *best*, *mean* and *worst* objective function values for each method. represents a mean of computational time.

Table I 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 II 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 as good a performance as the method by Katagiri et al.. Although the best values are often obtained by the method by Blum et al., the performances of our algorithm are fairly better than the method by Blum et al. in respect to the mean and worse objective function values. Therefore, we conclude that our algorithm provides a better robustness performance than the method by Blum et al.

TABLE I

| graph | k | | Objective value TSI | TSK | TSB |
|---|---|---|---|---|---|
| $\|N\| = 1000$ | 200 | best | 4131 | 4098 | **3609** |
| $\|E\| = 1250$ | | mean | 4147.6 | 4373.2 | **3685.9** |
| $d = 2.5$ | | worst | 4174 | 4587 | **3771** |
| $\sigma(d) = 1.57$ | 400 | best | 9624 | 9936 | **8976** |
| | | mean | 9841.0 | 10088.0 | **9091.0** |
| | | worst | 9918 | 10202 | **9301** |
| | 600 | best | 16299 | 17243 | **16282** |
| | | mean | **16319.0** | 17320.1 | 16323.7 |
| | | worst | **16349** | 17330 | 16454 |
| | 800 | best | **26429** | 27170 | 26552 |
| | | mean | **26429** | 27172.9 | 26687.3 |
| | | worst | **26429** | 27173 | 26755 |
| | 900 | best | **32981** | **32981** | 33147 |
| | | mean | **32984.5** | 33284.0 | 33187.6 |
| | | worst | **32985** | 33459 | 33233 |
| $\|N\| = 400$ | 80 | best | 1627 | 1478 | **1466** |
| $\|E\| = 800$ | | mean | 1627 | 1562.4 | **1477.5** |
| $d = 4.00$ | | worst | 1627 | 1627 | **1500** |
| $\sigma(d) = 0.00$ | 160 | best | 3330 | 3361 | **3217** |
| | | mean | 3346.1 | 34522.5 | **3240.0** |
| | | worst | 3369 | 3449 | **3259** |
| | 240 | best | 5264 | 5270 | **5215** |
| | | mean | 5281.6 | 5432.5 | **5224.6** |
| | | worst | **5325** | 5531 | 5234 |
| | 320 | best | **7682** | 7684 | **7682** |
| | | mean | 7687.8 | 7697.9 | **7682.9** |
| | | worst | 7689 | 7719 | **7684** |
| | 360 | best | **9249** | 9256 | 9250 |
| | | mean | **9249** | 9257.8 | 9257 |
| | | worst | **9249** | 9259 | 9260 |
| $\|N\| = 1000$ | 200 | best | 1100 | 1130 | **1047** |
| $\|E\| = 5000$ | | mean | 1141.1 | 1166.5 | **1063.7** |
| $d = 10.0$ | | worst | 1175 | 1225 | **1078** |
| $\sigma(d) = 3.22$ | 400 | best | 2577 | 2682 | **2499** |
| | | mean | 2602.6 | 2698.9 | **2535.9** |
| | | worst | 2639 | 2725 | **2604** |
| | 600 | best | 4570 | 4681 | **4516** |
| | | mean | 4590.4 | 4705.6 | **4548.6** |
| | | worst | 4608 | 4718 | **4603** |
| | 800 | best | 7324 | 7405 | **7281** |
| | | mean | 7325.8 | 7418.9 | **7324.7** |
| | | worst | **7359** | 7434 | 7405 |
| | 900 | best | **9248** | 9375 | 9291 |
| | | mean | **9248** | 9375.0 | 9323.9 |
| | | worst | **9248** | 9376 | 9372 |
| $\|N\| = 450$ | 90 | best | 139 | 138 | **135** |
| $\|E\| = 8168$ | | mean | 141.3 | 145.1 | **136.7** |
| $d = 36.30$ | | worst | 145 | 155 | **140** |
| $\sigma(d) = 16.83$ | 180 | best | 346 | 349 | **337** |
| | | mean | 353.8 | 352.2 | **346.5** |
| | | worst | 357 | **356** | 434 |
| | 270 | best | 631 | 643 | **630** |
| | | mean | **632.1** | 649.2 | 653.4 |
| | | worst | **637** | 654 | 728 |
| | 360 | best | **1060** | 1062 | **1060** |
| | | mean | **1060.1** | 1062 | 1098.7 |
| | | worst | **1064** | 1070 | 1158 |
| | 405 | best | **1388** | 1389 | 1391 |
| | | mean | **1388.0** | 1389.4 | 1410.8 |
| | | worst | **1389** | 1390 | 1467 |

## VII. 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 though numerical experiments with several benchmark instances. It has been shown that the proposed method has an advantage of robustness over the existing methods. However, numerical experiments executed are not enough to conclude such advantage is still valid for other types of benchmark instances. 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.

TABLE II

| graph | k | | Objective value TSI | TSK | TSB |
|---|---|---|---|---|---|
| $\|N\| = 300$ | 60 | best | 554 | **546** | 554 |
| $\|E\| = 20000$ | | mean | 579.9 | **572.0** | 628.8 |
| $d = 133.3$ | | worst | **603** | 606 | 2180 |
| $\sigma(d) = 36.57$ | 120 | best | 1236 | 1267 | **1229** |
| | | mean | **1294.2** | 1304.4 | 1613.4 |
| | | wrost | **1346** | 1384 | 3189 |
| | 180 | best | 2179 | 2241 | **2169** |
| | | mean | **2184.5** | 2256.5 | 2838.9 |
| | | worst | **2208** | 2264 | 4517 |
| | 240 | best | **3564** | **3564** | 3566 |
| | | mean | 3571.7 | **3568.8** | 4313.7 |
| | | worst | **3572** | 3581 | 5906 |
| | 270 | best | **4690** | **4690** | **4690** |
| | | mean | **4690** | **4690** | 5326.3 |
| | | worst | **4690** | **4690** | 6635 |
| $\|N\| = 300$ | 60 | best | 355 | **354** | 357 |
| $\|E\| = 30000$ | | mean | 361.6 | **359.8** | 503 |
| $d = 200.00$ | | worst | 367 | **364** | 2552 |
| $\sigma(d) = 38.99$ | 120 | best | 891 | 897 | **877** |
| | | mean | **898.5** | 912.4 | 1228.2 |
| | | worst | **920** | 922 | 3047 |
| | 180 | best | 1737 | 1661 | **1653** |
| | | mean | **1738.2** | 1746.6 | 2194.9 |
| | | worst | **1764** | 1784 | 3947 |
| | 240 | best | 2760 | **2737** | 2740 |
| | | mean | 2760.3 | **2753.0** | 3290.3 |
| | | worst | **2765** | **2765** | 5019 |
| | 270 | best | **3491** | **3491** | **3491** |
| | | mean | **3491** | **3491** | 4322.2 |
| | | worst | **3491** | **3491** | 5730 |
| $\|N\| = 300$ | 60 | best | 237 | **224** | **224** |
| $\|E\| = 40000$ | | mean | 243.3 | **238.1** | 405.4 |
| $d = 266.67$ | | worst | **255** | 257 | 2430 |
| $\sigma(d) = 24.61$ | 120 | best | 566 | **547** | 554 |
| | | mean | 572.3 | **567.4** | 1075.3 |
| | | worst | 605 | **589** | 2787 |
| | 180 | best | 1016 | 1031 | **986** |
| | | mean | **1034.9** | 1053.5 | 1653.1 |
| | | worst | **1055** | 1066 | 3271 |
| | 240 | best | 1671 | 1656 | **1647** |
| | | mean | 1678.2 | **1659.3** | 2493.3 |
| | | worst | 1696 | **1676** | 3939 |
| | 270 | best | **2107** | **2107** | **2107** |
| | | mean | **2107** | 2108.5 | 2845.5 |
| | | worst | **2107** | 2109 | 4440 |

## REFERENCES


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

[2] C. Blum, M.J. Blesa, New metaheurisitic approaches for the edge-weighted $k$-cardinality tree problem, *Computers & Operations Research*, Vol. 32, pp. 1355–1377, 2005.

[3] D. Urosevic, J. Brimberg, N. Mladenovic, Variable neighborhood decomposition search for edge weighted $k$-cardinality tree problem, *Computers Operations Research*, Vol. 31 , pp. 1205–1213, 2004.

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

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

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

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

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

[9] K. Jornsten, A. Lokkentangen, Tabu search for weighted $k$-cardinality trees, *Asia-Pacific Journal of Operational Research*, Vol. 14, No.2, 9–26, 1997.

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

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

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

[13] N. Garg, D. Hochbaum, 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.

[14] N. Mladenovic, D. Urosevic, Variable Neighborhood Search for $k$-Cardinality Tree, *Proceedings of Fourth Metaheuristics International Conference*, pp. 743–747, 2001.

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

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

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

[18] S.Y. Cheung, A. Kumar, Efficient quorumcast routing algorithms, *Proceedings of INFOCOM*, Los Alamitos, USA, Silver Spring, MD: IEEE Society Press, 1994.