



ELSEVIER

European Journal of Operational Research 91 (1996) 565–572

EUROPEAN  
JOURNAL  
OF OPERATIONAL  
RESEARCH

## Theory and Methodology

# A heuristic algorithm for the mini-max spanning forest problem<sup>1</sup>

Takeo Yamada<sup>\*</sup>, Hideo Takahashi, Seiji Kataoka

*Department of Computer Science, The National Defense Academy, Yokosuka, Kanagawa 239, Japan*

Received June 1994; revised September 1994

### Abstract

In this paper we formulate the mini-max spanning forest problem for undirected graphs as a generalization of the minimum spanning tree problem. We prove that the former problem is  $\mathcal{NP}$ -hard. Then we develop a heuristic algorithm to solve this problem approximately, and give an upper bound for relative errors. Through a series of numerical tests, we examine the performance of the developed algorithm.

**Keywords:** Spanning forest; Mini-max problem; Heuristics; Graph theory

### 1. Introduction

Consider constructing an electric power supply network to serve a set of cities from more than one power plant. The network must be economical, and at the same time it is desired to be as 'fair' as possible, in the sense that the maximum of the total length of the wire connected to each plant is of minimum. Alternatively, we may consider an election of  $r$  seats in a country consisting of  $r$  cities and several villages. The problem is to divide the country into  $r$  electoral districts in such a way that each district consists of a city and some villages which are not separated by other districts. The total population of each district must be as balanced as possible.

We formulate these as the *mini-max spanning forest problem* (MMSFP for short) in the follow-

ing way. Let  $G = (V, E)$  be an *undirected graph* where  $V$  is the set of *nodes* and  $E \subseteq V \times V$  is the set of *edges*. Each edge  $e \in E$  has an associated integer *weight*  $w(e) > 0$ . We assume  $G$  is *connected* and *simple*, i.e., there exist neither self-loops nor multiple edges. A *tree* is a connected acyclic subgraph of  $G$ . For a tree  $T$ , its weight  $w(T)$  is defined as the sum of the weights of its constituent edges. More generally, an acyclic subgraph of  $G$  is said to be a *forest*. A forest consists of a set of mutually disjoint trees. A tree is a *spanning tree* of  $G$  if it touches all the nodes of  $G$ . A *spanning forest* is similarly defined. Given a set of *root nodes*  $U := \{u_1, u_2, \dots, u_r\} \subseteq V$ , an  *$U$ -rooted spanning forest*  $F$  is a spanning forest of  $G$  consisting of  $r$  disjoint trees  $T_1, T_2, \dots, T_r$  such that  $u_i$  is a node of  $T_i$  ( $i = 1, 2, \dots, r$ ). The *value* of such a forest is defined by

$$w(F) := \max_{1 \leq i \leq r} \{w(T_i)\}. \quad (1)$$

In MMSFP we wish to find the  $U$ -rooted spanning forest  $F^*$  that minimizes  $w(F)$  over all

<sup>\*</sup> Corresponding author.

<sup>1</sup> Paper presented at APORS'94, July 25–29, Fukuoka, Japan.

$U$ -rooted spanning forests of  $G$ .

Clearly, for  $r = 1$  the problem is the standard *minimum spanning tree problem* (MSTP), which can be solved by the polynomial time algorithms due to Kruskal [5] or Prim [6]. See also Graham [3] on MSTP. For  $r \geq 2$ , however, we prove MMSFP is  $\mathcal{NP}$ -hard. Then, we develop a heuristic algorithm to solve this problem for  $r = 2$ . The algorithm first constructs a spanning forest using either Prim's method or the greedy method, and then tries to improve the forest by successively exchanging parts of component trees until no further improvement is possible. Computational complexity of this algorithm is  $O(W |V|^2)$ , where  $W$  is the sum of weights over all edges. We also show that Prim's algorithm gives a good lower bound for MMSFP. This enables us to derive an error bound for the heuristic algorithm. In computational experiments for randomly generated sample problems, we find that the algorithm yields solutions of only a few percent relative error.

### 2. $\mathcal{NP}$ -hardness of MMSFP

In this section we prove MMSFP is  $\mathcal{NP}$ -hard for  $r \geq 2$ . Clearly, it suffices to show the case of  $r = 2$ . First, we note that the following decision problem is  $\mathcal{NP}$ -complete [2,4].

**PARTITION.** *Given  $n$  positive integers  $w_1, w_2, \dots, w_n$ , is there a set  $S \subseteq \{1, 2, \dots, n\}$  such that*

$$\sum_{i \in S} w_i = \sum_{i \in S^c} w_i \tag{2}$$

*holds? (Here  $S^c$  denotes the complement of  $S$ .)*

Let  $H_n$  be the undirected *bipartite graph* with the set of nodes  $\{a, b\} \cup \{1, 2, \dots, n\}$  and the set of arcs  $\{(a, i), (b, i) | i = 1, 2, \dots, n\}$ . The weights of arcs  $(a, i)$  and  $(b, i)$  are both  $w_i$ . Let  $\mathcal{F}_n$  be the set of  $\{a, b\}$ -rooted spanning forests of  $H_n$ , and define

$$w_n^* := \min_{F \in \mathcal{F}_n} \{w(F)\}. \tag{3}$$

We note that  $w_n^*$  is readily obtained from the solution to MMSFP for  $H_n$  with root nodes  $\{a, b\}$ .

Further, let  $W := \sum_{i=1}^n w_i$ . Then, the following is obvious.

**Lemma 1.** (i)  $w_n^* \geq \frac{1}{2}W$ .

(ii) *Equality holds in (i) if and only if the answer to PARTITION is 'yes'.*

This implies the following.

**Theorem 1.** *MMSFP is  $\mathcal{NP}$ -hard.*

### 3. A heuristic algorithm

In the sequel, we discuss MMSFP for a graph  $G = (V, E)$  with two root nodes, i.e.,  $r = 2$ . Let  $F$  be a spanning forest of  $G$ . For an arbitrary node  $u \in V$ , there exists a unique path along  $F$  from this node to a root node. The corresponding root node is denoted as  $r(u)$ . Any nodes on the path from  $u$  to  $r(u)$  are *ancestor* of  $u$ , and  $u$  is a *descendant* of such a node. By  *$u$ -rooted subtree* of  $F$  we mean the restriction of  $F$  to the set of nodes  $\{u\} \cup \{v | u \text{ ancestor of } v\}$ . This is denoted as  $F_u$ . The *parent node*  $u^+$  of  $u$  is an ancestor of  $u$  which is adjacent to  $u$ . In this case,  $u$  is a *child* of  $u^+$ . An edge  $(u, v) \in E$  is to be a *cut edge* of  $F$  if  $r(u) \neq r(v)$ .

Given a spanning forest  $F = (T_1, T_2)$  and a cut edge  $e = (u, v)$ , we can obtain another spanning forest  $F(u, v)$  by disconnecting  $F_u$  from the tree to which it originally belonged and attaching it to the other side through  $e$ . This operation, illustrated in Fig. 1, is referred to as *edge switching* at  $(u, v)$ . If  $u$  and  $v$  belong to  $T_i$  and  $T_j$  ( $i \neq j$ ) respectively, by edge switching we obtain  $F(u, v)$  consisting of trees

$$T_i^\perp := T_i \setminus F_u \setminus \{(u, u^+)\}$$

and

$$T_j^\perp := T_j \cup F_u \cup \{e\}.$$

Correspondingly the value of the forest changes from

$$w(F) = \max\{w(T_i), w(T_j)\}$$

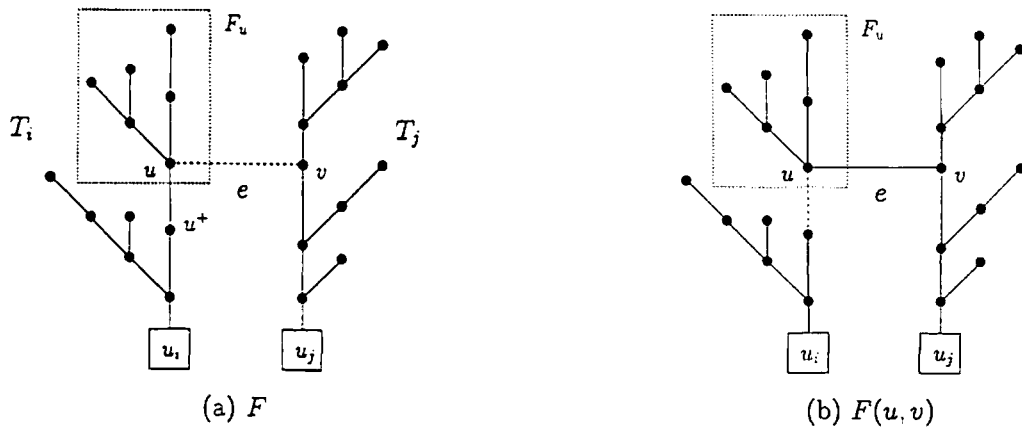


Fig. 1. (a) Before and (b) after edge switching.

to

$$w(F(u, v)) = \max\{w(T_i^\pm), w(T_j^\pm)\},$$

where

$$w(T_i^\pm) = w(T_i) - w(F_u) - w(\{u, u^+\})$$

and

$$w(T_j^\pm) = w(T_j) + w(F_u) + w(e).$$

Let

$$\delta_F(u, v) := w(F) - w(F(u, v)).$$

This represents the degree of improvement of the forest by edge switching at  $(u, v)$ .

The heuristic algorithm starts with an arbitrary spanning forest  $F$  and tries to improve it successively by repeating edge switchings until no further improvement is possible. When improvement has ceased, let the spanning forest be  $F = (T_1, T_2)$  at this stage and define  $V_i$  as the set of nodes of  $T_i$  ( $i = 1, 2$ ). Let  $G_i$  denote the subgraph of  $G$  restricted to  $V_i$ . Note that  $G_1$  and  $G_2$  are mutually disjoint. Let  $\bar{T}_i$  be the minimum spanning tree within  $G_i$ . By definition  $w(\bar{T}_i) \leq w(T_i)$ , and thus we obtain  $w(\bar{F}) \leq w(F)$  for  $\bar{F} = (\bar{T}_1, \bar{T}_2)$ . Obtaining  $\bar{F}$  from  $F$  is referred to as *re-optimization* of the spanning forest  $F$ . The whole algorithm is as follows:

**Heuristic algorithm**

*Step 1.* Find an initial spanning forest  $F^0$ , and let  $F \leftarrow F^0$ .

*Step 2.* If there exists a cut edge  $(u, v)$  such that  $\delta_F(u, v) > 0$ , go to Step 3; else go to Step 4.

*Step 3.* Improve the spanning forest by edge switching at  $(u, v)$ ; i.e.,  $F \leftarrow F(u, v)$ , and go to Step 2.

*Step 4.* Re-optimize  $F$  to obtain  $\bar{F}$ .

If  $F \neq \bar{F}$ , let  $F \leftarrow \bar{F}$  and go to Step 2; else stop.

The starting spanning forest  $F^0$  may be obtained by applying Prim's method to the *modified graph*  $G_0 = (V_0, E_0)$ , which is obtained from  $G$  by identifying two root nodes. The following 'greedy' method tries to construct a spanning forest by successively adopting the feasible edge that causes the *least* increase in the value of resulting forest. This usually gives a better initial spanning forest than Prim's method. To describe the greedy method, let  $F'$  be a (not necessarily spanning) forest of  $G$ . For  $e \in E$  the subgraph consisting of the edges of  $F'$  and  $e$  is denoted by  $F' \cup \{e\}$ . Let  $E_{F'} := \{e \in E \mid e \text{ is incident to } F', \text{ and } F' \cup \{e\} \text{ is a forest}\}$ .

**Greedy method for initial spanning forest**

*Step 1.* Let the forest  $F' = (T'_1, T'_2)$  be initially  $T'_i = (V_i, E_i)$  with  $V_i = \{u_i\}$  and  $E_i = \phi$  ( $i = 1, 2$ ). Its value is  $w(F') = 0$ .

*Step 2.* Take  $e^* \in E_{F'}$  that minimizes  $w(F' \cup \{e\})$  over all  $e \in E_{F'}$ , and let  $F' \leftarrow F' \cup \{e^*\}$ .

Step 3. Stop if  $F'$  is a spanning forest. Otherwise, go to Step 2.

In implementing the heuristic algorithms, with respect to a spanning forest  $F$  we require to keep the following data for each node  $u \in V$ :

- the parent  $u^+$ ,
- the root node  $r(u)$ , and
- the weight of  $u$ -rooted subtree  $w(F_u)$ .

The space complexity for this data structure is  $O(|V|)$ .

We now evaluate the computational complexity of the algorithm. We find an initial spanning forest  $F^0$  either by Prim's method or by the greedy method in  $O(|V|^2)$  steps. Next, given a spanning forest  $F$  we need  $O(|E|)$  steps to find a cut edge with  $\delta_F(u, v) > 0$ . In edge switching at  $(u, v)$ , we change the parent node of  $u$  from  $u^+$  to  $v$ , which can be done in  $O(1)$  steps. Now we update  $r(u)$  and  $w(F_u) (\forall u \in V)$  so as to correspond to the improved spanning forest. For this  $O(|V|^2)$  steps suffice. Thus, in total one iteration of edge switching can be performed in  $O(|E| + |V|^2) = O(|V|^2)$  steps. Clearly, re-optimization requires  $O(|V|^2)$  steps. In the heuristic algorithm either edge switching or re-optimization is repeated, and the value of the forest decreases at least by 1 in every two iterations. Let  $W := \sum_{e \in E} w(e)$ . Then the process comes to an end in at most  $O(W)$  iterations. Thus the total number of steps is at most  $O(W|V|^2)$ . Therefore, it is a *pseudo-polynomial* algorithm [1].

#### 4. Evaluation of errors

Consider the minimum spanning tree on the modified graph  $G_0$ . Clearly the spanning tree corresponds uniquely to a spanning forest in  $G$ . Let  $F^0 = (T_1^0, T_2^0)$  denote the resulting spanning forest. Further, let  $F^* = (T_1^*, T_2^*)$  be the optimal solution to MMSFP, and  $F^\dagger = (T_1^\dagger, T_2^\dagger)$  the approximate solution obtained by the heuristic algorithm of the previous section. For an arbitrary spanning forest  $F = (T_1, T_2)$ , define

$$\underline{w}(F) := \frac{1}{2} \{w(T_1) + w(T_2)\}.$$

Let  $\underline{w}^0 := \underline{w}(F^0)$ . Then, we note the following relation:

$$w(F^\dagger) \geq w(F^*) \geq \underline{w}^0. \tag{4}$$

Note that  $F^0$  is first obtained in calculating  $F^\dagger$  if we use Prim's method for the initial spanning forest. From (4) the relative error  $\{w(F^\dagger) - w(F^*)\} / w(F^*)$  of the heuristic solution  $F^\dagger$  is guaranteed to be at most

$$\{w(F^\dagger) - \underline{w}^0\} / \underline{w}^0. \tag{5}$$

#### 5. An illustrative example

Consider the planar graph  $G$  of Fig. 2, consisting of 100 nodes and 260 edges with root nodes  $a$  and  $b$  shown by black squares. The weight of each arc is its length rounded to integer. Fig. 3a is the initial spanning forest  $F_P^0$  obtained by applying Prim's method to the modified graph  $G_0$ . At this stage, all nodes are connected to root node  $a$ . The weight of each tree is  $w(T_a) = 4495$  and  $w(T_b) = 0$ ; thus the value of the forest is  $w(F_P^0) = 4495$  as shown in the figure with a star superscribed to the weight of the tree. By edge switching with respect to the edge shown with dashed line in Fig. 3a, we obtain the spanning forest  $F_P^1$  of Fig. 3b with  $w(F_P^1) = 4380 < w(F_P^0)$ .

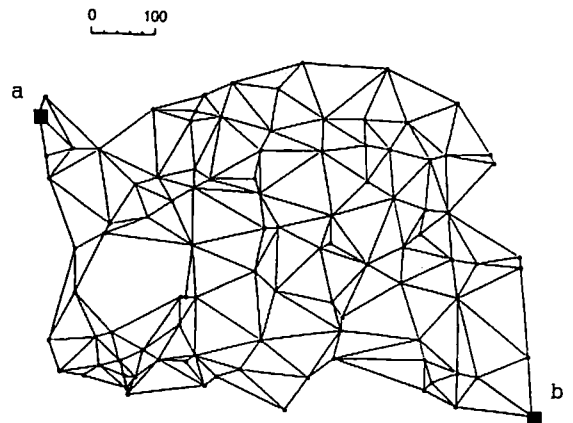


Fig. 2. A Planer graph  $P_{100,260}$ .

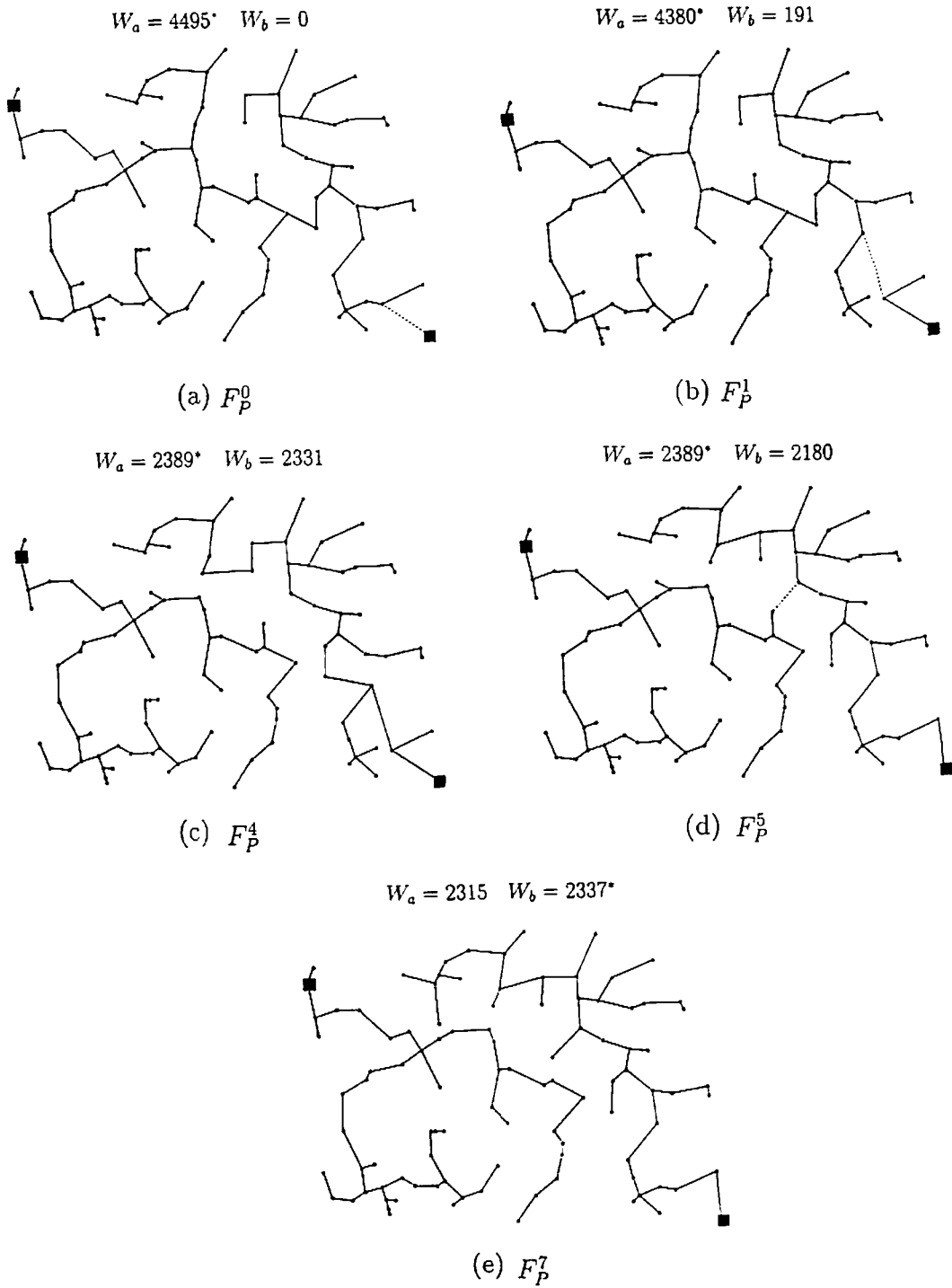


Fig. 3. A heuristic process starting from  $F_P^0$  obtained by Prim's method.

Repeating edge switching three more times, we arrive at the spanning forest  $F_P^4$  as shown in Fig. 3c.

At this stage, no further edge switching improves the value of forest. Thus we re-optimize the forest, which produces  $F_P^5$  of Fig. 3d. Although the value remains the same, i.e.,  $w(F_P^5) = w(F_P^4)$ ,  $F_P^5$  is not identical to  $F_P^4$ . We go on the whole process all over again, and through two more edge switchings obtain  $F_P^7$  of Fig. 3e. Edge switching terminates at this point. This time re-optimization yields the same spanning forest  $F_P^7$ ; thus the heuristic process stops here. We obtain an approximate solution  $F_P^7$  with  $w(F_P^7) = 2337$ .

Similarly, Fig. 4a shows the initial spanning forest  $F_G^0$  obtained by the greedy method of Section 3. This time we have  $w(F_G^0) = 2362$ . By edge switching we obtain  $F_G^1$  with  $w(F_G^1) = 2305$ , and the heuristic process stops at this stage.

Note that for  $F_P^0$  we have

$$\underline{w}^0 = \frac{1}{2}(4495 + 0) = 2247.5.$$

Then, from (5) the relative error of  $F_P^7$  is guaranteed to be at most

$$\{w(F_P^7) - \underline{w}^0\} / \underline{w}^0 = (2237 - 2247.5) / 2247.5 = 0.0398.$$

Similarly, the relative error of  $F_G^1$  is at most  $(2305 - 2247.5) / 2247.5 = 0.0256$ .

### 6. Numerical experiments

To evaluate the performance of the heuristic algorithm of this paper, a series of experiments were carried out for the sample problems of the following type. Problem  $P_{n,m}$  is a planar graph with  $n$  nodes and  $m$  edges. For example, Fig. 2 depicts  $P_{100,260}$ . Problems  $K_n$  and  $K_{n,n}$  are the complete graph and the complete bipartite graph respectively. In these problems, the nodes are randomly distributed in  $[0, 800] \times [0, 600]$ , and the weights of edges are rounded euclidean lengths.

Table 1 summarizes the result of experiments on DECstation 3000. Several cases were tested for each problem class with the number of nodes and edges varied as shown in the table. The column *Prim* and *Greedy* denote the results obtained by starting the heuristic algorithm with respective initialization methods. *Steps* stands for the number of steps to convergence. *Time* is the CPU time in seconds, and *Error* represents the relative error of (5) shown in percentage. These are averages of ten runs for each problem with different pair of root nodes.

From this table we see the following:

- The greedy and Prim's methods lead to approximate solutions of comparable quality. Also,

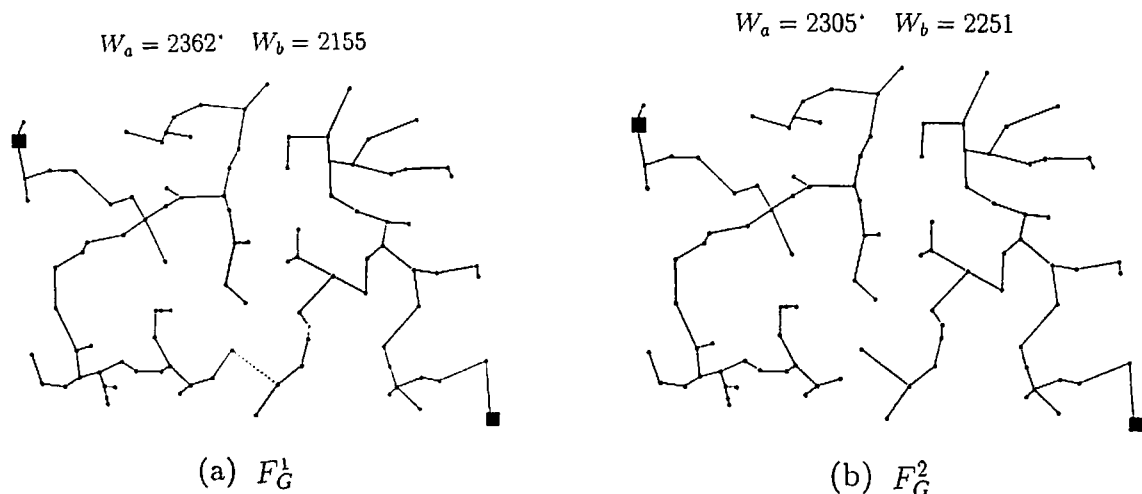


Fig. 4. A heuristic process starting from  $F_G^0$  obtained by the greedy method.

Table 1  
The result of numerical experiments

Problem	Prim			Greedy			ARW
	Steps	Time	Error	Steps	Time	Error	
$P_{50,127}$	4.2	0.008	3.328	1.4	0.005	2.001	3.39
$P_{100,260}$	7.2	0.028	3.371	2.8	0.023	2.916	1.57
$P_{200,560}$	8.2	0.100	2.469	4.5	0.092	2.189	0.83
$P_{400,1120}$	11.3	0.425	1.901	3.0	0.348	0.846	0.42
$P_{600,1680}$	16.5	1.170	1.538	7.2	0.875	0.882	0.27
$P_{800,2240}$	13.8	2.082	0.895	6.0	1.468	0.414	0.20
$P_{1000,2800}$	19.0	2.600	0.907	10.1	2.207	0.832	0.16
$K_{20}$	2.3	0.003	15.305	2.1	0.003	16.511	18.21
$K_{40}$	3.2	0.025	7.250	3.4	0.027	7.803	11.78
$K_{60}$	2.9	0.068	6.558	2.2	0.070	3.134	9.91
$K_{80}$	2.8	0.170	4.895	2.5	0.177	3.394	8.83
$K_{100}$	3.7	0.345	4.238	3.3	0.377	3.244	7.75
$K_{120}$	4.2	0.602	5.851	3.3	0.648	3.726	7.13
$K_{140}$	3.0	0.792	5.356	2.8	0.872	2.924	6.58
$K_{20,20}$	3.0	0.012	2.084	2.8	0.012	1.985	3.77
$K_{30,30}$	3.5	0.035	1.308	3.3	0.038	1.560	2.59
$K_{40,40}$	5.0	0.097	1.819	4.4	0.093	2.231	1.93
$K_{50,50}$	4.0	0.173	0.810	2.5	0.153	0.748	1.57
$K_{60,60}$	5.2	0.313	1.394	3.7	0.318	1.507	1.35
$K_{70,70}$	5.6	0.498	1.172	3.6	0.450	1.367	1.15
$K_{80,80}$	3.8	0.660	0.876	3.7	0.665	0.879	1.01

with respect to the CPU time we observe no significant difference between these two methods.

- For the problems tested, the larger the size of problems the more accurate solutions we usually obtain.

- For planar graphs and bipartite graphs, the heuristic algorithm yields solutions within a few percent from optimality. However, this percentage is considerably higher for complete graphs.

To explain why approximation is poor for complete graphs, we introduce the *average relative weight* (ARW in Table 1) of an edge in a graph. This is defined as the percentage of the average weight of edges to the total weight of the minimum spanning tree in that graph. This is remarkably high for complete graphs. The squared correlation coefficient ( $R^2$ ) between *Error* and *ARW* was 0.898 for *Prim* and 0.782 for *Greedy*. Thus, the approximation appears to be poor for the problems with higher relative weights.

## 7. Concluding remarks

We have formulated the mini-max spanning forest problem, proved it to be  $\mathcal{NP}$ -hard and developed a heuristic algorithm to solve this problem. An upper bound was given for the relative error of this approximation. A series of numerical tests were carried out to evaluate the performance of the proposed algorithm. Also, an investigation was conducted to explain why the approximation is superior in some type of problems, while for others it is not that satisfactory. As a future research issue we mention an exact algorithm to solve MMSFP, which is currently under development.

## Acknowledgements

The authors are grateful to anonymous referees for their helpful comments.

## References

- [1] Ahuja, R.K., Magnanti, T.L., and Orlin, J.B., *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NY, 1993.
- [2] Garey, M.R., and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1978.
- [3] Graham, R.L., and Hell, P., "On the history of the minimum spanning tree problem", *Annals of the History of Computing* 7 (1985) 43–57.
- [4] Karp, R.M., "Reducibility among combinatorial problems", in: R.E. Miller and J.W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum, New York, 1972, 85–103.
- [5] Kruskal, J.B., "A shortest spanning subtree of a graph and the travelling salesman problem", *Proceedings of the American Mathematical Society* 7 (1956) 48–50.
- [6] Prim, R.C., "Shortest connection networks and some generalizations", *Bell Systems Technical Journal* 36 (1967) 1389–1401.