A Polynomial Algorithm for Interval Minimum Spanning Tree Problem

A. Hossain

Key Words: *Interval algorithm; interval minimum spanning tree problem; network; uncertainty.*

Abstract. *A new polynomial algorithm is presented for solving the minimum spanning tree problem. The algorithm is applicable to the case when the generalized distance associated with each arc is non-negative, interval or real. The interval algorithm is developed on the base of midpoint and half-width representation of intervals and is more efficient than the interval algorithm that could be proposed by using traditional interval description. The complexity of the new algorithm is evaluated.*

1. Introduction

There are many reasons why network models and algorithms are widely used, i.e., they exactly represent the real world systems, they facilitate extremely efficient solution to large real problems, they can solve problems with significantly more variables and constraints than can be solved by other optimization techniques, etc. [13].

The Minimum Spanning Tree (*MST*) problem is one of the traditional combinational optimization problems. The MST algorithm is one of the simplest algorithm in network algorithms, but at the same time this algorithm has many real applications, and it is very popular.

In 1926 the MST problem was first formulated and solved by Boruvka. He was involved in the rural electrification of Southern Moravia, where the author gave a solution to find the most economical layout of a power-line network [3,18]. The algorithm presented by Boruvka was based on a kind of cut property.

The classical MST algorithm is based on two sets of nodes: the set of connected nodes, and the set of unconnected nodes. The MST algorithm starts with any node from the unconnected set of the network and moving it in to the set of connected node. The next step is to identify another node from the unconnected set that is closest to any node in the set of connected nodes, and to connect these two nodes. This process will continue until all the nodes are connected [8,13,15].

In [17], the author proposed an algorithm to find MST for undirected and connected network. The running time of the algorithm is $O(m \log\log n)$. Three sets T, NS, AS are used in Yao's algorithm. T is used to collect edges of the final spanning tree. The set NS contains the nodes sets corresponding to the connected components of the spanning tree got so far. AS contains, for each node set W in NS, an edge set $A(W)$. Denote the starting node by Stn, $Stn \in n$. At the beginning, $NS = \{\{Stn\} \mid n \in N\}$, and ES = {{all edges incident upon Stn}| $Stn \in n$}.

A variant of the MST problem which models a location problem is discussed in [2]. An $O(n^2)$ algorithm is proposed for constrained MST problem. To reduce the problem complexity a *divide-and-conquer* technique is applied (divide-and-conquer technique – breaking down a problem into two or more sub-problems of the same type, until these become simple enough to be solved directly).

Two fast algorithms to find the minimum spanning tree in undirected and directed networks are given in [3]. The first algorithm is $O(m \log \beta(m,n))$ – time algorithm to find the undirected minimum spanning tree, where $\beta(m,n) = \min\{i \mid \log^{(i)} n \le m/n\}$ and $\log^0 n = n$, $\log^{(i+1)} x = \log\log^{(i)} x$. The second algorithm is $O(n \log n + m)$ – time algorithm to determine the directed minimum spanning trees. The F-heaps (*Fibonacci heap*) were used to obtain fast algorithms for MST problem. A *heap* (sometimes called a *priority queue*) is an abstract data structure consisting of a collection of items, each with a real-valued key, on which many operations are possible.

In [9], the authors introduced two types of indirect covering tree problems, and these are the minimum cost covering subtree (*MCCS*) problem and the maximal indirect covering subtree (*MICS*) problem. The objective of the MCCS is to find the minimum cost collection of arcs that form a subtree and satisfy covering constraints for nodes of the network. The reduction techniques used to solve the location set covering problem are extended to solve the MCCS problem. The MICS chooses that subtree which maximizes the demand within a distance standard of nodes of the subtree.

An algorithm for the max + sum spanning tree (*MSST*) problem is given in [14]. The MSST problem is as follows: $\underset{T \in T(G)}{Minimize}[\underset{a \in T}{Max}\{w_a\} + \sum_{a \in T} c_a]$, where $T(G)$ is the set of all spanning trees of G. Each edge $a \in A$ has two weights c_a and w_a respectively. The authors designated theses as *c*-weight and *w*-weight and their units are the same.

An initial version of the interval algorithm is presented for solving the minimum spanning tree under parametric uncertainty in [5,6], and the final version of the interval minimum spanning tree algorithm is given in [7]. This algorithm is an interval extension of the classical minimum spanning tree algorithm. The exact values of the parameters of a given network are uncertain and included within some intervals. The author considered the interval length of a branch is non-negative, and represented by L_{ij}, $L_{ij} = [\underline{l}_{ij}, \bar{l}_{ij}]$. The algorithm is based

on midpoint and half-width notation of an interval. The algorithm is applicable when the parameters of a given network are real or interval.

Recently, in [11,16], the authors discussed a variant of the MST problem, called the Robust Spanning Tree problem (*RST* problem), where the edge costs are uncertain and lie within intervals.

In [16], the authors proposed a mixed integer programming formulation for the RST problem. The authors described the concepts of weak edge and strong edge used to preprocess the network efficiently prior to solution of the RST problem by mixed-integer programming. A mixed-integer programming formulation for the solution of the RST problem is proposed. They used preprocessing technique in the mixed integer program, based on removing edges from the network which are not weak, and by forcing strong edges into the solution. The proposed RST contains only edges which lie on some MST for some scenario (weak edges). They consequently did not consider non-weak edges when looking for a robust spanning tree.

In [11], a branch and bound algorithm for solving the RST problem is proposed. The complexity of the algorithm is $O(n^{n-1}m^2 \log m)$. The authors compared their algorithm with other RST algorithms. The result showed that the new algorithm is faster than the other RST algorithms.

In [1], the authors presented a survey paper on the MST problem. The authors described old algorithms, as well as, new ones for the MST problem, and classified these algorithms according to their similarities or the tools used and subproblems solved. The authors compared modern algorithms with the classical ones and evaluated their relative performance through extensive empirical tests, using a reasonably large-size problem.

The aim of this paper is to develop a simple and effective method and algorithm for solving the minimum spanning tree problem under parametric uncertainty. It is assumed that the uncertainty about the generalized length (distance, cost, time, etc.) of the branches is described by intervals, and the mean-value lemma [4] is applied to develop the interval algorithm.

The paper is organized as follows. The theoretical preliminaries are given in the second section. The interval minimum spanning tree method and algorithm are presented in the third section. The conclusion is made in section 4.

2. Theoretical Preliminaries

The interval analysis concepts are introduced [10,12]. Let R be an interval. We will denote its lower (left) endpoint by $\underline{r}$ and its upper (right) endpoint by $\overline{r}$, so that $R = [\underline{r}, \overline{r}]$.

The set of all intervals will be denoted by $I(R)$. Let $R, S \in I(R)$, and let * denote any of the interval arithmetic operations, * = +, –, ×, /. Then the set theory definition of the interval arithmetic operations is as follows:

(1) $R * S = \{r * s \mid r \in R, s \in S\}$.

It follows that the sum of $R = [\underline{r}, \overline{r}]$, $S = [\underline{s}, \overline{s}]$ denoted by $R + S$, is the interval $R + S = [\underline{r}, \overline{r}] + [\underline{s}, \overline{s}] = [\underline{r} + \underline{s}, \overline{r} + \overline{s}]$.

The product $R \times S$ is again an interval

$$R \times S = [\min\{\underline{r}\,\underline{s}, \underline{r}\,\overline{s}, \overline{r}\,\underline{s}, \overline{r}\,\overline{s}\}, \max\{\underline{r}\,\underline{s}, \underline{r}\,\overline{s}, \overline{r}\,\underline{s}, \overline{r}\,\overline{s}\}]$$

For $R, S > 0$ the definition reduces to

(2) $R \times S = [\underline{r}\,\underline{s}, \overline{r}\,\overline{s}]$.

The half-width of an interval $R = [\underline{r}, \overline{r}]$ is the real number, $w(R) = \frac{1}{2}(\overline{r} - \underline{r})$, and the midpoint of R is the real number, $m(R) = (\underline{r} + \overline{r})/2$.

Using the set inclusion relation $\subseteq$ and the relation $\leq$, we can define the supremum-like (*sup*) and infimum-like (*inf*) elements [10]:

(3) $\sup(R, S) = [\sup(\underline{r}, \underline{s}), \sup(\overline{r}, \overline{s})]$;

(4) $\inf(R, S) = [\inf(\underline{r}, \underline{s}), \inf(\overline{r}, \overline{s})]$.

To compare intervals the concept of metric ρ is introduced. For each R and S in $I(R)$ the distance ρ is defined by

(5) $\rho(R, S) = \frac{1}{2}\{|\underline{r} - \underline{s}| + |\overline{r} - \overline{s}|\}$.

Now the intervals R and S can be compared. The following important results hold in [4].

$R \leq S$ iff (if and only if);

(6) $\rho(R, \inf(R, S)) \leq \rho(S, \inf(R, S))$.

In a similar way,

$R \geq S$ iff

(7) $\rho(R, \sup(R, S)) \leq \rho(S, \sup(R, S))$.

Two intervals R and S are said to be equivalent $R \sim S$ if the following condition holds:

(8) ρ (R, sup (R, S)) = r (S, sup (R, S));

(9) ρ (R, inf (R, S)) = r (S, inf (R, S)).

It means that $|\underline{r} - \underline{S}| = |\overline{s} - \overline{r}|$, i.e., the midpoints of R and S coincide.

In practical cases when $R \sim S$ and one have to make a choice in the sense of $\leq$, the condition (6) should be modified. We say that $R \leq S$ if

(10) ρ (R, inf (R, S)) = r (S, inf (R, S)) and $\underline{r} \leq \underline{S}$

or

(11) ρ (R, inf (R, S)) = ρ (S, inf (R, S)) and $\overline{r} \leq \overline{s}$.

We use, further, the notation $R \leq S$ in the usual sense, when $\underline{r} \leq \underline{S}$ and $\overline{r} \leq \overline{s}$, and in the case of inclusion, $R \subseteq S$, when ρ (R, inf (R, S)) $\leq \rho$ (S, inf (R, S)).

The conditions (6) and (7) lead to the following result, as proven in [4].

Let $m(P)$ denote the midpoint of P, $m(P) = (\underline{p} + \overline{p})/2$. Then

(12) $R \le S$ iff $m(R) \le m(S)$.

Let $[m(R), \Delta(R)]$ denote the interval R, $R = [\underline{r}, \overline{r}]$, where $m(R) = (\underline{r} + \overline{r})/2$ is the midpoint of R, and $\Delta(R) = (\underline{r} - \overline{r})/2$ is the half-width of R, so that

(13a) $R = [m(R) - \Delta(R), m(R) + \Delta(R)]$;

(13b) $R = [m(R), \Delta(R)]$.

The following result is easily shown:

Let R, S, and $T \in I(R)$. Then $T = R + S$ iff

(14) $m(T) = m(R) + m(S)$;

(15) $\Delta(T) = \Delta(R) + \Delta(S)$.

3. Interval Minimum Spanning Tree Method and Algorithm

The minimum spanning tree algorithm starts with any node and joining it to the closest node in the network. The resulting two nodes form a connected set, C, with the remaining nodes comprising in the unconnected set, $\overline{C}$. Next, the node from the unconnected set that is closest to any node in the connected set is determined and the two nodes are connected. The process is repeated until the unconnected set becomes empty [8,13,15].

The interval minimum spanning tree problem can be formulated as follows [7]: an undirected network G, $G = (N, A)$, is given, where $N = \{1, \ldots, n\}$ is the set of the nodes and $A = \{(i, j), (k, l), \ldots, (y, z)\}$ is the finite set of branches of interval lengths L_{ij}, $L_{ij} = [\underline{l}_{ij}, \overline{l}_{ij}]$, joining nodes in N.

A set T of connecting branches is to be found, such that all nodes of the network are connected, and the total interval length L of the connected branches is *minimum* in comparison with the lengths of all possible trees, $L = \sum_{(i,j)\in T} [\underline{l}_{ij}, \overline{l}_{ij}] \rightarrow \min$. The set T is termed *Interval Minimum Spanning Tree*. We will consider nonnegative interval lengths of branches, $\underline{l}_{ij} \ge 0$.

Let D_{ij} be the interval distance between node i and node j, and $D_{ij} = [\underline{d}_{ij}, \overline{d}_{ij}]$.

Denote the set of all connected nodes by C, and the set of unconnected nodes by $\overline{C}$, and $\overline{C} = N \mid C$, where N is the set of all nodes.

Set $C = \{Stn\}$ and $= \overline{C} = N \mid C$, where Stn is the starting node.

Find the unconnected node j^* that is nearest to node Stn

(16) $D_{j^*} = \min_{j\in\overline{C}} \{D_{(Stn)j}\}$

where $D_{j^*} = [\underline{d}_{j^*}, \overline{d}_{j^*}]$; $D_{(Stn)j} = [\underline{d}_{(Stn)j}, \overline{d}_{(Stn)j}]$.

The computational complexity of a straightforward interval generalization of the algorithm described above is relatively high, because the comparison of intervals would be based on using infimum-like intervals and distances, and the interval arithmetic operations are more complex than the traditional ones.

A simple interval algorithm can be developed for the interval representation (13b), $R = [m(R), \Delta(R)]$, where $m(R)$ and $\Delta(R)$ are midpoint and half-width of the interval R and the conditions (12), (14), and (15).

The interval algorithm consists of the following generalized steps:

Step 1. Describe the network using midpoint and half-width representation of intervals (13b), $D_{ij} = [d_{ij}, \Delta_{ij}]$.

Denote the set of all connected nodes by C, and the set of unconnected nodes by $\overline{C}$, $\overline{C} = N \mid C$, where N is the set of all nodes.

Denote the set of connected branches by T.

Denote the starting node by Stn.

Set $C = \{Stn\}$, $\overline{C} = N \mid C$, and $T = \varnothing$.

Set $d_{kr} = M$, $M >> 0$, when it is impossible to connect directly nodes k and r, k, $r \in N$.

Step 2. Find the unconnected node j^* that is nearest to node Stn.

$$d_{(Stn)j^*} = \min_{j\in\overline{C}} \{d_{(Stn)j}\}.$$

Connect nodes Stn and j^*.

Set $C = \{Stn, j^*\}$, $\overline{C} = N \mid C$, $T = \{T, (Stn, j^*)\}$.

Step 3. Identify the unconnected node that is closest to any of the connected node, and then connect these two nodes.

Correct the sets C, $\overline{C}$, and T.

If there is a tie, arbitrarily choose between them.

Obtain

$$d_{r^*k^*} = \min_{r\in C, k\in\overline{C}} \{d_{rk}\}$$

Connect nodes r^* and k^*.

Set $C_{new} = \{C, k^*\}$; $\overline{C}_{new} = N \mid C_{new}$; $T = \{T, (r^*, k^*)\}$;

$C = C_{new}$ and $\overline{C} = \overline{C}_{new}$.

Step 4. If $\overline{C} \ne \varnothing$ go to *step* 3.

Step 5. Compute the midpoint $m(L)$ of the interval length L of the minimum spanning tree by adding the midpoints of all connecting branches, $m(L) = \sum_{(i,j)\in T} d_{ij}$.

Compute the half-width $\Delta(L)$ of the interval length of the tree by adding the half-widths Δ_{ij} of all connecting branches, $\Delta(L) = \sum_{(i,j)\in T} \Delta_{ij}$.

Obtain the interval length of the minimum spanning tree in the form (13a).

3.1. Analysis of the Complexity of the Interval Minimum Spanning Tree Algorithm

Consider the undirected network G, $G = (N, A)$, in *figure 1*, where $N = \{1, 2, 3, 4, 5, 6\}$, $n = 6$, and $A = \{(1, 2), (1, 3), \ldots, (5, 6)\}$. The generalized lengths of the branches are as follows: $d_{12} = d_{21} = 5$, $d_{13} = d_{31} = 15$, etc. C, $\overline{C}$, T are the set of connected node(s), the set of unconnected node(s), and the set of connected branche(s), respectively.

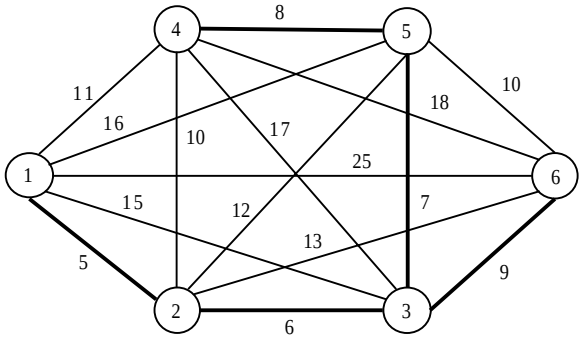

Figure 1. An undirected network with six nodes

Iteration 1. $C = \{1\}$, $\overline{C} = \{2, 3, 4, 5, 6\}$, $T = \varnothing$.

Iteration 2. $\min(d_{12}, d_{13}, d_{14}, d_{15}, d_{16}) = \min(5, 15, 11, 16, 25) \Rightarrow 5$

$C = \{1, 2\}$, $\overline{C} = \{3, 4, 5, 6\}$, $T = \{(1, 2)\}$;

In the general case, when the number of nodes is n, $(\boldsymbol{n-1})$ branches are compared, and $\boldsymbol{(n-2)}$ comparisons are needed.

Iteration 3. $\min(\min(d_{13}, d_{14}, d_{15}, d_{16}), \min(d_{23}, d_{24}, d_{25}, d_{26})) = \min(11, 6) \Rightarrow 6$;

$C = \{1, 2, 3\}$, $\overline{C} = \{4, 5, 6\}$, $T = \{(1, 2), (2, 3)\}$;

In the general case, $\boldsymbol{2(n-2)}$ branches are compared, and $\boldsymbol{2(n-2)-1}$ comparisons are needed.

Iteration 4. $\min(\min(d_{14}, d_{15}, d_{16}), \min(d_{24}, d_{25}, d_{26}), \min(d_{34}, d_{35}, d_{36})) = \min(11, 10, 7) \Rightarrow 7$;

$C = \{1, 2, 3, 5\}$, $\overline{C} = \{4, 6\}$, $T = \{(1, 2), (2, 3), (3, 5)\}$.

In the general case, $\boldsymbol{3(n-3)}$ branches are compared, and $\boldsymbol{3(n-3)-1}$ comparisons are needed.

Iteration 5. $\min(\min(d_{14}, d_{16}), \min(d_{24}, d_{26}), \min(d_{34}, d_{36}), \min(d_{54}, d_{56})) = \min(11, 10, 9, 8) \Rightarrow 8$;

$C = \{1, 2, 3, 5, 4\}$, $\overline{C} = \{6\}$, $T = \{(1, 2), (2, 3), (3, 5), (5, 4)\}$.

In the general case, $\boldsymbol{4(n-4)}$ branches are compared, and $\boldsymbol{4(n-4)-1}$ comparisons are needed.

Iteration 6. $\min(d_{16}, d_{26}, d_{36}, d_{46}, d_{56}) = \min(25, 13, 9, 18, 10) \Rightarrow 9$;

$C = \{1, 2, 3, 4, 5, 6\}$, $\overline{C} = \{0\}$, $T = \{(1, 2), (2, 3), (3, 5), (5, 4), (3, 6)\}$.

In the general case, $\boldsymbol{(n-1)(n-(n-1)}$ branches are compared, and $\boldsymbol{((n-1)(n-(n-1)-1)}$ comparisons are needed.

The number of comparisons needed for n nodes is: $\{1\ (n-1)-1\} + \{2\ (n-2)-1\} + \{3\ (n-3)-1\} + \ldots + \{(n-2)\ (n-(n-2))-1\} + \{(n-1)\ (n-(n-1))-1\} = (n-1)\left(\frac{n^2}{2}-1\right) - \sum_{i=1}^{n-1} i^2$.

The total number of comparisons (*comp*) and additions (*addi*) that are needed to compute interval minimum spanning tree is:

$$comp = (n-1)\left(\frac{n^2}{2}-1\right) - \sum_{i=1}^{n-1} i^2;$$

$addi = (n-1) + (n-1) + 2$.

So, the running time of the algorithm is bounded by

$$O\left(comp = \frac{n^3}{2},\ addi = 2n\right).$$

Comment

Maximum $(n-1)$ addition(s) are needed to compute the total midpoint and $(n-1)$ addition(s) are needed to compute the total half-width. Another 2 additions are needed to obtain the traditional interval representation.

Note that if the interval formula (16) were used, then each comparison of two intervals $V = [\underline{v}, \overline{v}]$ and $W = [\underline{w}, \overline{w}]$ includes the following comparisons and additions:

Step 1. if $\overline{v} \le \underline{w} \rightarrow$ set Lab = V, go to the next interval comparison $\Rightarrow$ 2 comparisons.

Step 2. if $\overline{w} \le \underline{v} \rightarrow$ set Lab = W, go to the next interval comparison $\Rightarrow$ 2 comparisons.

Step 3. if $V = W$ ($\underline{v} = \underline{w}$ and $\overline{v} = \overline{w}$) $\rightarrow$ set Lab = V, W, go to the next interval comparison $\Rightarrow$ 2 comparisons.

Step 4. if $\underline{v} \le \underline{w}$ and $\overline{v} \le \overline{w} \rightarrow$ set Lab = V, go to the next interval comparison $\Rightarrow$ **4** comparisons.

Step 5. if $\underline{w} \le \underline{v}$ and $\overline{w} \le \overline{v}$ set Lab = W, go to the next interval comparison $\Rightarrow$ **4** comparisons.

Step 6. if $\underline{v} \le \underline{w} \rightarrow$ set $\rho_1 = \rho\ (V, \text{inf}) = \overline{v} - \overline{w}$; set $\rho_2 = \rho\ (W, \text{inf}) = \underline{w} - \underline{v} \Rightarrow$ **2** comparisons, **2** additions;
if $\rho_1 < \rho_2 \rightarrow$ set Lab = V
else set Lab = W go to the next interval
comparison $\Rightarrow$ **1** comparison.

Step 7. if $\underline{w} \le \underline{v} \rightarrow$ set $\rho_1 = \rho\ (V, \text{inf}) = \underline{v} - \underline{w}$; set $\rho_2 = \rho\ (W, \text{inf}) = \overline{w} - \overline{v} \Rightarrow$ **2** comparisons, **2** additions;
if $\rho_1 \le \rho_2 \rightarrow$ set Lab = V
else set Lab = $W \Rightarrow$ **2** comparisons.

To compare two intervals **1** to **21** comparisons and **0** to **4** additions are needed. So, if we develop the interval minimum spanning tree algorithm based on a traditional interval representation, the complexity of the algorithm will be very high.

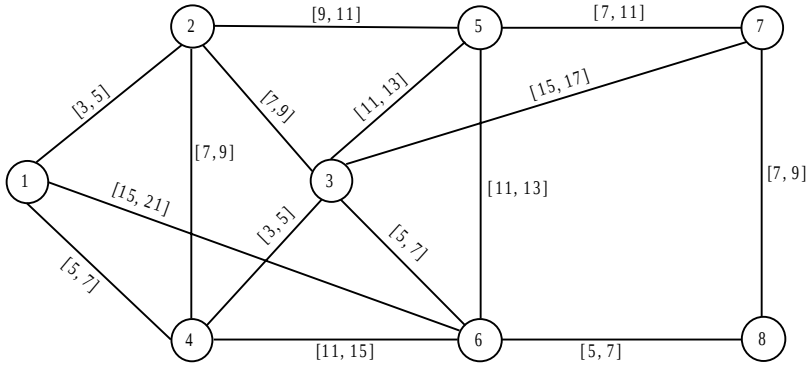

Figure 2. Network with traditional interval representation

3.2. Numerical Example

Let consider the network in *figure 2*. The parameters (values) along the branches give the costs (or generalized lengths) D_{ij} of establishing links between nodes i and j. The cost is uncertain and represented by upper and lower limits.

The graphical network *of figure 2* is presented in the *table 1*, using midpoint and half-width notation (*step* 1).

minimum interval cost L:

$m(L) = d_{12} + d_{14} + d_{43} + d_{36} + d_{68} + d_{87} + d_{75} = 43;$

$\Delta(L) = \Delta_{1\text{-}2} + \Delta_{1\text{-}4} + \Delta_{4\text{-}3} + \Delta_{3\text{-}6} + \Delta_{6\text{-}8} + \Delta_{8\text{-}7} + \Delta_{7\text{-}5} = 8.$

Hence, $L = [m(L), \Delta(L)] = [43, 8]$, and the minimum interval cost is obtained in the usual interval notation:

$L = [\{m(L) - \Delta(L)\}, \{m(L) + \Delta(L)\}] = [(43 -8), (43 +8)] = [35, 51].$

The minimum interval cost $L = [\mathbf{35, 51}]$.

Table 1. Representation of the network in *figure 2* using midpoint and half-width notation

From Node	To Node							
	1	2	3	4	5	6	7	8
1	[0, 0]	[4, 1]	M	[6, 1]	M	[18, 3]	M	M
2	[4, 1]	[0, 0]	[8, 1]	[8, 1]	[10, 1]	M	M	M
3	M	[8, 1]	[0, 0]	[4, 1]	[12, 1]	[6, 1]	[16, 1]	M
4	[6, 1]	[8, 1]	[4, 1]	[0, 0]	M	[13, 2]	M	M
5	M	[10, 1]	[12, 1]	M	[0, 0]	[12, 1]	[9, 2]	M
6	[18, 3]	M	[6, 1]	[13, 2]	[12, 1]	[0, 0]	M	[6, 1]
7	M	M	[16, 1]	M	[9, 2]	M	[0, 0]	[8, 1]
8	M	M	M	M	M	[6, 1]	[8, 1]	[0, 0]

M represents the case when there is no possible direct connection between nodes i and j, $M >> 0$.

Using the algorithm as described in section 3, the following computational results for the minimum spanning tree problem are obtained and summarized in the *tables*.

Table 2 and *table 3* represent the results of iterations 1 and 2, respectively. In a similar way the results of iterations 3 to 7 can be obtained. After iteration 7, it is finally found that all the nodes have been connected. Our problem is now essentially solved. We need only look in *tables* and find which nodes are connected to give the solution of the minimum spanning tree problem. These are set out in *table 4*, from which it can be seen that the total cost (midpoint) is 43 units, and the total half-width is 8 units. The interval minimum spanning tree network is graphed in *figure 3*.

We obtain the midpoint $m(L)$ and half-width $\Delta(L)$ of the

4. Conclusions

In this paper, we proposed an interval algorithm for solving minimum spanning tree problem, based on midpoint and half-width representation of intervals (13b), and the conditions (12), (14), and (15). The new interval algorithm is applicable when the parameters are real or interval valued.

The new approach yields simple and computationally effective algorithm, when the exact values of the parameters are unknown, but the upper and lower limits within which the values are expected to fall are given. Instead of comparing intervals using the distance (5) and infimum-like intervals (4), the conditions (10) and (11), the new algorithm compares real values, i.e., the midpoints of the intervals.

The complexity of the interval MST algorithm is evaluated,

Table 2. The result of step 2 iteration 1

From conne. node	To Nodes								Minimum distance	Total connected nodes *C*
	1	2	3	4	5	6	7	8		
1	[0, 0]	[4,1]	M	[6, 1]	M	[18, 3]	M	M	[4, 1]	{1, 2}
The new connected node is 2, the minimum distance is 4, and the half-width is 1.										

Table 3. The result of step 2 iteration 2

From conne. node	To Nodes								Min. dis.	Min. distance from *i* nodes	Total connected nodes *C*
	1	2	3	4	5	6	7	8			
1			M	[6, 1]	M	[18, 3]			[6, 1]	min= {[6, 1], [8, 1]}= [6, 1]	{1, 2, 4}
2			[8, 1]	[8, 1]	[10, 1]	M			[8, 1]		
The next connected node is 4, the minimum distance is 6, and the half-width is 1.											

Table 4. The optimal solution of the minimum spanning tree problem

Iter.	Branch	Distance (midpoint)	Half-width
1.	1-2	4	1
2.	1-4	6	1
3.	4-3	4	1
4.	3-6	6	1
5.	6-8	6	1
6.	8-7	8	1
7	7-5	9	2
		Total distance = 43	Total half-width = 8

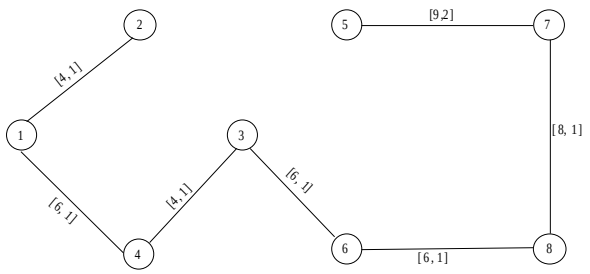

Figure 3. Optimal solution of the minimum spanning tree problem

it is apolynomial algorithm. The running time of the algorithm is limited by $O(comp = \frac{n^3}{2},\ addi = 2n)$.

A numerical example is given to illustrate the efficient assessment of the solution and workability of the developed interval algorithm. It is possible to obtain an interval algorithm by using traditional interval description, but the new proposed algorithm based on midpoint and half-width representation is more efficient. The developed algorithm may be employed directly to many practical problems.

Reference

1. Bazlamaçci, C. F., K. S. Hindi. Minimum-weight Spanning Tree Algorithms. A Survey and Empirical Study. – *Computers & Operations Research*, 28, 2001, 767-785.

2. Chen, G., G. Zhang. A Constrained Minimum Spanning Tree Problem. – *Computers & Operations Research*, 2000, 867-875.

3. Gabow, H. N., Z. Galil, T. Spencer, R. E. Tarjan. Efficient Algorithms for Finding Minimum Spanning Trees in Undirected and Directed Graphs. – *Combinatorica*, 6, No. 2, 1986, 109-122.

4. Gatev, G. Interval Analysis Approach and Algorithms for Solving Network and Dynamic Programming Problems under Parametric Uncertainty. Proceedings of the Technical University of Sofia, 48, 1995, 345-350.

5. Gatev, G., A. Hossain. Algorithm for Solving Minimal Spanning Tree Problem under Parametric Uncertainty. Preprints, International IFAC Workshop, DECOM-TT 2004, Automatic Systems for Building the Infrastructure in Developing Countries, Regional and Global Aspects, Bansko, 2004, 145-149.

6. Hossain, A. Interval Algorithms for Solving Network and Dynamic Programming under Parametric Uncertainty, Master's Thesis, Technical University of Sofia, 1999.

7. Hossain, A. Network Models under Parametric Uncertainty, PhD Thesis, Bulgaria, 2009.

8. Hillier, S. F., G. J. Lieberman. Introduction to Operations Research. Fifth Edition, McGraw-Hill Publishing Company, New York, 1990.

9. Huston, V. A., C. ReVelle. Indirect Covering Tree Problems on Spanning Tree Networks. – *European Journal of Operational Research*, 65, 1993, 20-32.

10. Kalmikov, S. A., Yu. I. Shokyne, Z. H. Yuldashev. Interval Analysis Methods, Nauka, Novosibirsk, 1986 (in Russain).

11. Montemanni, R., L. M. Gambardella. A Branch and Bound Algorithm for the Robust Spanning Tree Problem with Interval Data. – *European Journal of Operational Research*, 161, 2005, 771-779.

12. Moore, R. E. Interval Analysis, N. J., Prentice Hall, 1966.

13. Phillips, Don T., and A. Garcia-Diaz. Fundamentals of Network Analysis. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1981.

14. Punnen, A. P., K. P. K. Nair. An $O(m \log n)$ Algorithm for the max + sum Spanning Tree Problem. – *European Journal of Operational Research*. 89, 1996, 423-426.

15. Taha, H. A. Operations Research. An Introduction, Fifth Edition, MacMillan Publishing Company, New York, 1992.

16. Yaman, H., O. E. Karaşan, M. Ç. Pinar. The Robust Spanning Tree Problem with Interval Data. – *Operations Research Letters*, 29, 2001, 31-40.

17. Yao, A. C. An $O(|E|\log\log|V|)$ Algorithm for Finding Minimum Spanning Trees. – *Information Processing Letters*, 4, 1975, No. 1, 21-23.

18. Zhou, G., M. Gen. An Effective Genetic Algorithm to the Quadratic Minimum Spanning Tree Problem. – *Computers Ops Res.*, 25, 1988, No. 3, 229-237.,

Manuscript received on 3.02.2010

Akter Hossain *was born in Dhaka, Bangladesh. He received Master's Engineering in Control Systems at Technical University of Sofia, Sofia (1999). He got Post-Graduation in Dynamic Modeling, Management and Engineering at Technical University of Sofia, Sofia (2002). At the present time he is waiting for PhD degree. His research interests are in the fields of decision-making, operations research, management.*

Contacts:

e-mail: akter_h@yahoo.com; makter@gmail.com