Reliability Evaluation of the Minimum Spanning Tree on Uncertain Graph

Jie Tang[1], Yuansheng Liu[1*], Zhonghua Wen[2]

[1] Ministry of Education Key Laboratory of Intelligent Computing and Information Processing, College of Information Engineering, Xiangtan University, Xiangtan 411105, Hunan, China.
[2] Department of Computer & Communication, Hunan Institute of Engineering, Xiangtan 411104, Hunan, China.

* Corresponding author. Tel.: 13636550479; email: yyuanshengliu@gmail.com
Manuscript submitted June 6, 2014; accepted October 29, 2014.
doi: 10.17706/jcp.10.1.45-56

Abstract: Minimum spanning tree is a minimum-cost spanning tree connecting the whole network, but it couldn't be directly obtained on uncertain graph. In this paper, we define the reliability as the *existence probability* of all minimum spanning trees and present an algorithm for evaluating reliability of the minimum spanning tree on uncertain graph. The time complexity of the algorithm is $O(Nmn)$, where n, m and N stand for the number of vertices, edges and minimum spanning trees, respectively. Because this algorithm spends more time finding minimum spanning tree, we propose an improved algorithm whose time complexity is $O(Nm)$. The improved algorithm uses disjoint set data structure so that the average time complexity on finding a new minimum spanning tree is $O(m/n)$. The two algorithms are analyzed in detail and the experiment results agree with theoretical analysis.

Key words: Implicated graph, minimum spanning tree, uncertain graph.

1. Introduction

IN graph theory, a spanning tree of an undirected graph is a tree that contains all the vertices. A minimum spanning tree (MST) of an edge-weighted graph is a spanning tree with the minimum sum of edge weights among all spanning trees. Many practical problem can be modeled by using MST, and it has been widely applied in many fields such as wireless sensor networks [1], [2], cluster analysis [3]-[5] and data storage [6].

There have been many studies of the MST problem dealing with *deterministic* graphs and have been designed some well known algorithm such as Kruskal [7] and Prim [8], in which the MST problem can be solved in polynomial time. In deterministic graph, all edges are assumed to be fixed, but such an assumption might not always be right. For example, the links in wireless sensor network (WSN) may be impassable caused by noise, collision and congestion, we can use *uncertain graph* to represent the WSN.

Some researchers assume that each edge of an uncertain graph has an *existence probability*. An uncertain graph is also refereed to as a *probabilistic graph* [9]. Each possible subgraph of the uncertain graph is called *implicated graph*. Their research mainly focus on graph mining [10]-[12], graph queries [13]-[15] and basic graph structure [16], [17].

However, many studies for the MST problem assume the edge weights are uncertain, these researchers regard the edge weights as random variables, fuzzy variable and interval data, then present probabilistic

MST (PMST) problem, fuzzy MST (FMST) problem and interval data MST (IDMST) problem. Many efficient algorithms have also been developed by some researchers [18]-[20].

In multicast routing protocols, MST is one of the most effective methods to multicast the massages from a source node to the destinations. When a connected link goes down, the network topology need to be updated. So evaluating the reliability of MST is significant. In this paper, we assume that each edge of a uncertain graph has a existence probability and a weight. The existence probabilities might represent the fault possibility in links and the edge weights represent time or cost. We try to evaluate the reliability of MST, which is the existence probability of all MSTs.

The number of implicated graph increases exponentially with the number of edges, it need take a lot of time if we enumerate each implicated graph. In this paper, we propose a fundamental algorithm, called FERM (Fundamental algorithm of Evaluating the Reliability of MST), and an improved algorithm (*IERM*) to evaluating the reliability of the MST. The two algorithms classify all implicated graphs by finding all MSTs. The improved algorithm uses disjoint structure and search algorithm to get the swap-edges of each edge, so that the time complexity required to find a new MST is $O(m/n)$. Thus, the improved algorithm is more than n times as fast as the fundamental algorithm in theory.

The remaining of this paper is organized as follows. Section 2 gives the problem formulation. In Section 3, two algorithms for reliability evaluation of MST on uncertain graph are presented. We report our experiment result in Section 4, and a conclusion reached in Section 5.

2. Problem Formulation

Definition 1 (Uncertain graph): An *uncertain graph* $\mathcal{G}=(V,E,W,P)$, is defined over a set of vertices V, a set of edges E, a set of edge weights $W=\{w(e)|e\in E, w(e)\in \mathbb{N}^+\}$, and a set of probabilities $P=\{p(e)|e\in E, p(e)\in(0,1]\}$ of edge existence.

An *implicated graph* $G=(V_G,E_G,W_G)$ of an uncertain graph $\mathcal{G}$ is an certain graph which is realized by sampling each edge in $\mathcal{G}$ according to the probability $p(e)$. We denote the relationship between G and $\mathcal{G}$ as $\mathcal{G}\Rightarrow G$. Clearly, we have $V_G\subseteq V$, $E_G\subseteq E$ and $W_G=\{w(e)|e\in E_G\}\subseteq W$. There are a total of $2^{|E|}$ implicated graphs, because each edge provides us with a binary sampling decision. Following the same assumption of the existing uncertain graph models [10], [13], [21], we assume that uncertain variables of different edges are *mutually independent*. Based on this assumption, the probability of sampling the implicated graph G from the uncertain graph $\mathcal{G}$ is

$$\Pr(\mathcal{G}\Rightarrow G)=\prod_{e\in E_G}p(e)\prod_{e\in E\setminus E_G}(1-p(e)).$$

Let $Imp(\mathcal{G})$ denotes the set of all implicated graphs of the uncertain graph $\mathcal{G}$. Moreover, it is easy to know that function $\Pr(\mathcal{G}\Rightarrow G)$ defines a probability distribution over $Imp(\mathcal{G})$.

Fig. 1(a) shows an uncertain graph $\mathcal{G}_1$. The two numbers on each edge represent weight and existence probability, respectively. The uncertain graph $\mathcal{G}_1$ has 2^5 implicated graphs because $\mathcal{G}_1$ has five edges. Fig. 1(b) shows an implicated graph of the uncertain graph $\mathcal{G}_1$, which shows in Fig. 1(a), and the sampling probability of the implicated graph is $p(e_2)\times p(e_3)\times p(e_4)\times p(e_5)\times(1-p(e_1)))=0.03528$.

Definition 2 (Main implicated graph): Given an uncertain graph $\mathcal{G}=(V,E,W,P)$, we define the implicated graph $\hat{G}=(V,E,W)$ as the *main implicated graph* of $\mathcal{G}$.

Fig. 1(c) shows the main implicated graph of $\mathcal{G}_1$. It is clear that the main implicated graph has three MSTs $\{e_1e_2e_3\}$, $\{e_2e_3e_4\}$ and $\{e_2e_4e_5\}$ and the total weight of any MST is equal to 4.

Definition 3 (Reliable implicated graph): Given an uncertain graph $\mathcal{G}=(V,E,W,P)$. Let G be any implicated graph of $\mathcal{G}$, T be an MST of implicated graph G and $W^G=\sum_{e\in T} w(e)$ be the weight of the MST T. If $W^G=W^{\hat{G}}$, then G is a *reliable implicated graph* of the uncertain graph $\mathcal{G}$.

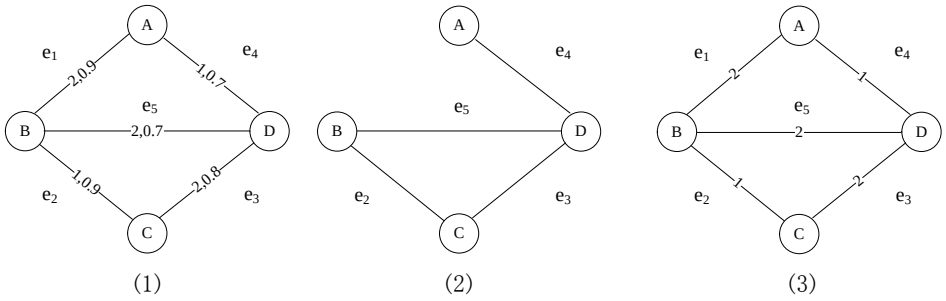


Fig. 1. Example: (a) uncertain graph $\mathcal{G}$; (b) an implicated graph of $\mathcal{G}_i$; (c) Main implicated graph of $\mathcal{G}_1$.

Definition 4 (Reliability of MST): Given an uncertain graph $\mathcal{G}=(V,E,W,P)$. Let $R=\{G|G\in Imp(\mathcal{G})\wedge W^G=W^{\hat{G}}\}$ denotes the set of all reliable implicated graph of the uncertain graph $\mathcal{G}$. The reliability of MST on uncertain graph $\mathcal{G}$ can be defined as follows:

$$r=\sum_{G\in R}\Pr(\mathcal{G}\Rightarrow G).$$

Table 1. Reliability Evaluation of the MST on Uncertain Graph Fig. 1(a).

implicated graph	containing MSTs	probability
$\{e_1,e_2,e_3,e_4,e_5\}$	$\{e_1,e_2,e_4\}$, $\{e_2,e_3,e_4\}$, $\{e_2,e_4,e_5\}$	0.31752
$\{e_1,e_2,e_3,e_4\}$	$\{e_1,e_2,e_4\}$, $\{e_2,e_3,e_4\}$	0.13608
$\{e_1,e_2,e_4,e_5\}$	$\{e_1,e_2,e_4\}$, $\{e_2,e_4,e_5\}$	0.07938
$\{e_2,e_3,e_4,e_5\}$	$\{e_2,e_3,e_4\}$, $\{e_2,e_4,e_5\}$	0.03528
$\{e_1,e_2,e_4\}$	$\{e_1,e_2,e_4\}$	0.03402
$\{e_2,e_3,e_4\}$	$\{e_2,e_3,e_4\}$	0.01512
$\{e_2,e_4,e_5\}$	$\{e_2,e_4,e_5\}$	0.00882
others	none	0.37378

Table 1 shows the existence probability of the implicated graph, which contains at least one MST in $\{e_2e_4e_5\}$, $\{e_1e_2e_4\}$ or $\{e_2e_4e_5\}$. There are seven implicated graphs containing MST, so the reliability of the MST is 0.62622.

3. The MST Reliability Evaluation

3.1. Fundamental Algorithm

Theorem 1: Given an uncertain graph $\mathcal{G}=(V,E,W,P)$. Let n be the number of vertices of uncertain graph $\mathcal{G}$, respectively, $T=\{e_1,e_2,\cdots,e_{n-1}\}$ be an MST of the main implicated graph $\hat{G}$ and $A_i=\{e_1,e_2,\cdots,e_i\}$, where $i\in\{1,\cdots,n-1\}$. We define

$$D_i=\begin{cases}\{G|G\in Imp(\mathcal{G})\wedge e_1\notin E_G\}, & i=0,\\ \{G|G\in Imp(\mathcal{G})\wedge A_i\subseteq E_G\wedge e_{i+1}\notin E_G\}, & 0<i<n-1,\\ \{G|G\in Imp(\mathcal{G})\wedge A_i\subseteq E_G\}, & i=n-1,\end{cases}$$

That is, D_i represents a set of graph whose edge set contains set A_i and excludes edge e_{i+1} (if exists) and these graphs in set D_i are a subgraph of the main implicated graph $\hat{G}$. Then, we have the following two results:

1) $\bigcup_{i=0}^{n-1} D_i = Imp(\mathcal{G})$

2) $D_i \cup D_j = \varnothing$, where $i < j$

Proof: We prove the two parts separately:

1) For any implicated graph $G_i \in Imp(\mathcal{G})$, if $E_{G_i} \cap T = \{e_{x_j}, e_{x_{j+1}}, \cdots, e_{x_y}\}$, where $x_j < x_{j+1} (j = 1, 2, ..., y-1)$, then there must exist a *k* meet:

$$\begin{cases} x_j = x_{j-1} + 1, & j \le k, \\ x_j > x_{j-1} + 1, & j = k+1. \end{cases}$$

Then, we have the following:

$$\begin{cases} G_i = D_0, & x_1 \ne 1, \\ G_i = D_k, & x_1 = 1. \end{cases}$$

So we have $Imp(\mathcal{G}) \subseteq \bigcup_{i=0}^{n-1} D_i$. According to Equation (1), we have $\bigcup_{i=0}^{n-1} D_i \subseteq Imp(\mathcal{G})$.

2) According to Equation (1), we have $D_i \cup D_j = \{G | G \in Imp(\mathcal{G}) \wedge A_j \subseteq E_G \wedge e_{i+1} \notin E_G\}$. However, $A_j \subseteq E_G$ and $e_{i+1} \notin E_G$ can't be achieved at the same time since $e_{i+1} \in A_j$. Thus, $D_i \cup D_j = \varnothing$.

Corollary 1: Let $I = \{e_1, e_2, \cdots, e_k\}$ and $O \subseteq E$. Now assume that there exist a minimum spanning tree $T = \{e_1, e_2, \cdots, e_{n-1}\}$ *s.t.* $I \subseteq T$ and $T \cap O = \varnothing$. Let $S = \{G | G \in Imp(\mathcal{G}) \wedge I \subseteq E_G \wedge O \cap E_G = \varnothing\}$, we define

$$D_i = \{G | G \in S \wedge I_i \subseteq E_G \wedge O_i \cap E_G = \varnothing\}$$

where $I_i = I \cup \{e_{k+1}, e_{k+2}, \cdots, e_i\}$, $O_i = O \cup \{e_{i+1}\}$. Then, we have $\bigcup_{i=k}^{n-1} D_i = S$.

According to Corollary 1, $Imp(\mathcal{G})$ can be divided recursively until each implicated graph in set S contains the same MST, namely, $I = T$. It is clear that the existence probability of set S is $\prod_{e \in I} p(e) \times \prod_{e \in O} (1 - p(e))$. We'll call the function *GetMST* to get a minimum spanning tree whose edge set contains set *I* and exclude set *O*, which is implemented by Kruskal' or Prim's algorithm. The fundamental algorithm, called *FERM* (Fundamental algorithm of Evaluating the Reliability of MST), is outlined as following:

Example 1: Fig. 2(a) is an uncertain graph $\mathcal{G}_2$ and Fig. 2(b) is a minimum spanning tree of its main implicated graph. It is obvious that when all edge weights have different values, the MST must be unique, so the edge weights in Fig. 2(a) are set as 1 or 2 to better understand function call $RMST(\varnothing, \varnothing, \mathcal{G}_2)$. Fig. 3 builds a multiway-tree to simulate function call $RMST(\varnothing, \varnothing, \mathcal{G}_2)$. The nodes with bold bolder show that the algorithm has already found a set of reliable implicated graph. As we can see from Fig. 3, all reliable implicated graphs are divided into nine nodes. However, the uncertain graph $\mathcal{G}_2$ has $2^6 = 64$ implicated graph. Thus, the Algorithm 1 can avoid large scale computing than enumerating all implicated graphs.

Algorithm 1 The *FERA* algorithm

Comment Calls from $RMST(\varnothing,\varnothing,\mathcal{G})$

1. Procedure $RMST(I,O,\mathcal{G})$ > $\mathcal{G}=(V,E,W,P)$
2. $T \leftarrow GetMST(I,O,\mathcal{G})$
3. $k \leftarrow$ the number of elements of set I
4. $r \leftarrow 0$
5. IF $\sum_{e\in T} W^{\hat{G}}$ THEN
6. IF $T = I$ THEN
7. $r \leftarrow \prod_{e\in I} p(e)\times\prod_{e\in O}(1-p(e))$
8. ELSE
9. FOR $i \leftarrow k$ TO $|V|-1$ DO
10. $I_i \leftarrow I \cup \{e_{k+1}, e_{k+2}, \cdots, e_i\}$
11. $O_i \leftarrow O \cup \{e_{i+1}\}$
12. $r \leftarrow r + RMST(I_i, O_i, \mathcal{G})$
13. RETURN r

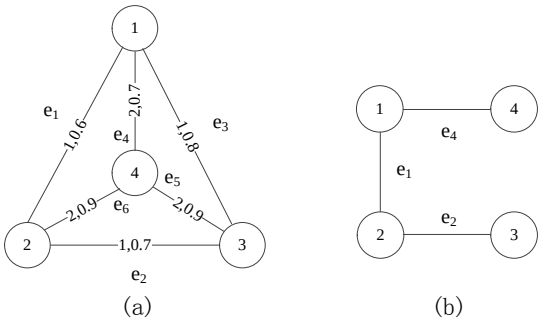


Fig. 2. Example: (a) uncertain graph $\mathcal{G}_2$; (b) an MST of main implicated graph of Fig. 2(a).

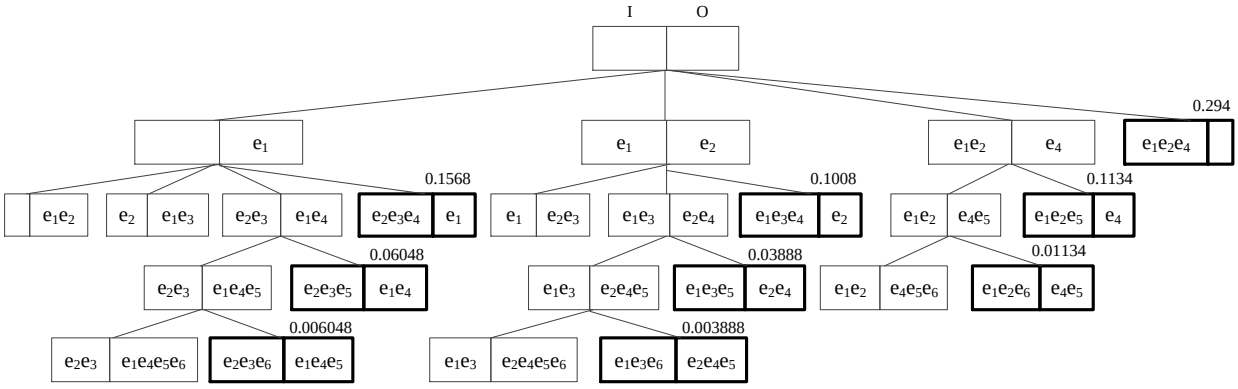


Fig. 3. The recursive procedure of the function call $RMST(\varnothing,\varnothing,\mathcal{G}_2)$.

Theorem 2: The time complexity of the Algorithm 1 is $O(Nnm)$, where N denotes the number of MSTs included in main implicated graph $\hat{G}$, n and m denote the number of vertices and edges, respectively.

Proof: Note that each time the function $RMST$ is called, it will call the function $GetMST$ once and produces at most n-1 recursive calls. So the total number of function call $RMST$ is at most $Nn+1\approx Nn$. However, it requires $O(m)$ time to solve an MST problem. Thus, the total time complexity is $O(Nnm)$.

3.2. Improved Algorithm

The fundamental algorithm should call the function $GetMST$ to obtain a new MST. This subsection

designs an improved algorithm, called $IERM$ (Improved algorithm of Evaluating the Reliability of MST), to obtain a new MST by performing edge replacement operation on previous MST. Let $T_1 \xrightarrow{e} T_2$ denotes the edge replacement operation, where T_2 is the result of T_1 after the edge replacement operation, $T_2 = T_1 \cup \{e'\} \setminus \{e\}$, and e' is the *swap-edge* of e.

Theorem 3: Assume that T_a is a spanning tree and T_b is an MST of implicated graph *G*. Let $I = \{e_1, e_2, \cdots, e_k\}$ denotes the set of different edges, where $e_i \in T_a$ and $e_i \notin T_b$. We can perform the edge replacement operation *k* times, and then obtain

$$T_a(T_0) \xrightarrow{e_1} T_1 \xrightarrow{e_2} T_2, \cdots, T_{k-1} \xrightarrow{e_k} T_b(T_k),$$

and

$$\sum_{e \in T_{i-1}} w(e) \geq \sum_{e \in T_i} w(e).$$

Proof: For $i = 1 \sim k$, repeat the following two steps:

1) Delete e_i from T_{i-1}, and then the tree T_{i-1} divided into two connected-components V_1 and V_2.
2) Assume that the path from vertex *u* to *v* in tree T_b is $u(u_0) \to u_1, \cdots, u_{x-1}, \to v(u_x)$, denotes as P_v^u. There must exist $i(i < x)$ such that $u_i \in V_1$ and $u_{i+1} \in V_2$. Let $e_i' = (u_i, u_{i+1})$. Because of T_b is a MST, it is easy to know that $w(e_i') \leq w(e_i)$, $T_{i-1} \xrightarrow{e_i} T_i$, and $\sum_{e \in T_i} w(e) = \sum_{e \in T_{i-1}} w(e) + w(e_i') - w(e_i) \leq \sum_{e \in T_{i-1}} w(e)$.

Thus, this theorem is proven.

Theorem 4: Let *T* be an MST included in graph *G*. Assume that edge $e \in T$ connects two connected-components, called V_1 and V_2, then for any MST T' whose edge set doesn't contain edge *e*, there exists an edge $e' = (u', v')$ such that $u' \in V_1$, $v' \in V_2$ and $w(e') = w(e)$.

Let $e = (u, v)$. Assume that the path P_v^u in tree T' is $u(u_0) \to u_1, \cdots, u_{x-1}, \to v(u_x)$. Then, there must exist $i(i < x)$ such that $u_i \in V_1$ and $u_{i+1} \in V_2$. Assume $e' = (u_i, u_{i+1})$, we have

- If $w(e') > w(e)$, let $T^* = T' \cup \{e\} \setminus \{e'\}$, then we has $\sum_{e \in T^*} w(e) < \sum_{e \in T'} w(e)$.
- If $w(e') < w(e)$, let $T^* = T \cup \{e'\} \setminus \{e\}$, then we has $\sum_{e \in T^*} w(e) < \sum_{e \in T} w(e)$.

The above two situations obviously contradict the fact that *T* and T' are an MST. Therefore, we have $w(e_i) = w(e)$.

Assume that $E(V_1, V_2, G) = \{e \in E_G | e \in V_1 \times V_2\}$. According theorem theorem 4, if there exists e', such that $e' \in E(V_1, V_2, G)$ and $w(e') = w(e)$, then we can obtain a new MST by edge replacement operation, that is $newT = T \cup \{e'\} \setminus \{e\}$. Otherwise, the graph *G* doesn't exist any other MST whose edge set doesn't contain the edge *e*.

To find swap-edges of each edge in tree *T*, we could traverse all edges in graph *G* from small to large according to its weight. For each edge $e^* \in G$, we will perform the following two steps:

1) Let $e^* = (u^*, v^*)$, then traverse the path $P_{v^*}^{u^*}$, it is easy to know that e^* is swap-edge of these

edges whose weights are equal to $w(e^*)$.

2) Use disjoint structure to compress the path $P_{v^*}^{u^*}$.

To find the path between two vertices of tree *T*, we should know the relationship (parent-child, brotherhood or other) of two vertices firstly. Assume that vertex 1 is root-node, the function $GetChild$ is defined as follow:

$$GetChild(T, w) = \{v | v \text{ is a child-node of } w \text{ in } T\}$$

The following algorithm numbers each subtree of the tree *T*:

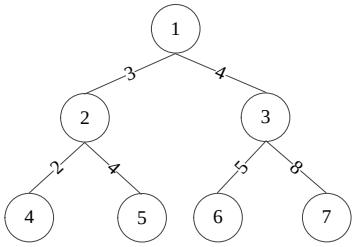
Algorithm 2 Number subtrees.

Comment We assume that the entry function is $SetNumber(T, 1, 1)$

1. Procedure $SetNumber(T, w, num)$
2. $L_w \leftarrow num, R_w \leftarrow num$
3. $C \leftarrow GetChild(T, w)$
4. FOR EACH $v \in C$ DO
5. $R_w \leftarrow SetNumber(T, v, R_w + 1)$
6. RETURN R_w

Definition 5 (Least common ancestor): Let *T* be a rooted tree. The *least common ancestor* between two vertices *v* and *w* is defined as the lowest node in *T* that has both *v* and *w* as descendant-nodes (where we allow a vertex to be a descendant-node of itself)

Example 2: Fig. 4 shows an MST *T* and execution result from function call $SetNumber(T, 1, 1)$. According the Algorithm 2, we know that L_i is generated by pre-order traversal and $R_i = \max\{L_j |$ vertex j is descendant-node of vertex $i\}$.



L_1: 1 R_1: 7
L_2: 2 R_2: 4
L_3: 5 R_3: 7
L_4: 3 R_4: 3
L_5: 4 R_5: 4
L_6: 6 R_6: 6
L_7: 7 R_7: 7

Fig. 4. An MST T & Number ranges of each subtree.

Example 2 shows that if $L_j \in [L_i, R_i]$, then vertex *j* is descendant-node of vertex *i*. Assume that $u_1, u_2 \in T$, we take the following steps to find the path between vertex u_1 and u_2:

1) Traverse these vertices from u_1 to 1 until we find a vertex *v* such that $L_{u_2} \in [L_v, R_v]$.

2) Traverse these vertices from vertex u_2 to v.

Therefore, $P_{u_2}^{u_1}$ can be represented as $u_1 \to \cdots \to v \to \cdots \to u_2$. It's clear that vertex *v* is the least common ancestor(LCA) between two vertices u_1 and u_2.

Next, we use disjoint structure to compress the path $P_{u_2}^{u_1}$. Let RT_i be current LCA of vertex *i*, then for any vertex *w* in the path $P_{u_2}^{u_1}$, we have $RT_w = v$. Assume that the function $GetFather(T, x)$ returns the

father-node of vertex *x* in tree *T*. Thus, the algorithm of finding swap-edges shows as Algorithm 3.

Algorithm 3 Find swap-edges of each edge in tree *T*.

Module 1: Disjoint structure

Initialization $RT_i \leftarrow i(i=1,2,\cdots,n)$

1. **Procedure** $GetRoot(u)$
2. IF $u = RT_u$ THEN RETURN u
3. ELSE RETURN $(RT_u \leftarrow GetRoot(RT_u))$

Module 2: Find swap edges

1. Procedure $FindSwapEdge(G,T)$
2. $S = \varnothing$
3. FOR EACH $e \leftarrow (u,v) \in E_G$ DO
4. $lca \leftarrow ru \leftarrow GetRoot(u)$
5. $rv \leftarrow GetRoot(v)$
6. WHILE $rv \notin [L_{lca}, R_{lca}]$ DO
7. $f \leftarrow GetFather(T, lca)$
8. $lca \leftarrow GetRoot(f)$
9. Let P_{rv}^{ru} be $ru \rightarrow \cdots \rightarrow lca \rightarrow \cdots \rightarrow rv$
10. FOR EACH edge e' in path P_{rv}^{ru} DO
11. IF $w(e') = w(e)$ THEN
12. $S \leftarrow S \cup \{(e', e)\}$
13. FOR EACH vertex w in path P_{rv}^{ru} DO
14. $RT_w \leftarrow lca$
15. RETURN S

Example 3: Assume that $e_1 = (2,5), e_2 = (3,6), e_3 = (4,5), e_4 = (5,6), w(e_3) = 4, w(e_4) = 5$ and $e_3, e_4 \in G$, as shown in Fig. 5. We first find the path between vertex 4 and 5 because of $w(e_3) < w(e_4)$, it's clear that the edge e_3 is a swap-edge of edge e_1 as $w(e_1) = w(e_3)$, then we replace the two vertices 4 and 5 with their LCA, that is $RT_4 = 2$ and $RT_5 = 2$. Next, we deal with the edge e_4, we can assume that $e_4 = (5,6) = (2,6)$ because of $RT_5 = 2$. So it is easy to know that $S = \{(e_1, e_3), (e_2, e_4)\}$.

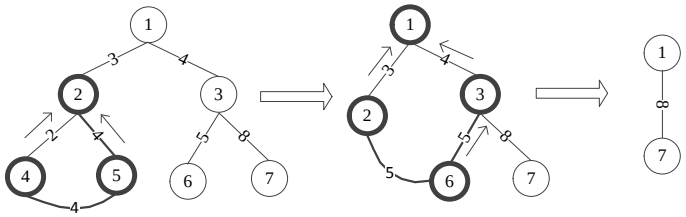


Fig. 5. The path-compressed procedure.

From what mentioned above, the improved algorithm is outlined as Algorithm 4.

Theorem 5: The time complexity of the improved algorithm is $O(Nm)$.

Proof: Note that $SetNumber$ can be done in $O(n)$, $FindSwapEdge$ traverses all edges of a graph in $O(m)$, each edge in tree *T* is visited just once because of path-compressed, thus, $FindSwapEdge$ can be done in $O(m)$ (assume that $m > n$). Then, the average time complexity required to find a new minimum

spanning tree is $O(m/n)$, so the time complexity of the improved algorithm is $O(Nn^*(m/n)=O(Nm)$.

Algorithm 4 The *IERM* algorithm

Comment Calls from $IMST(\varnothing,\varnothing,T,\mathcal{G})$
1. Procedure $IMST(I,O,T,\mathcal{G})$ > $\mathcal{G}=(V,E,W,P)$
2. $r \leftarrow 0$
3. IF $\sum_{e\in T} = W^{\hat{G}}$ THEN
4. IF $T = I$ THEN
5. $r \leftarrow \prod_{e\in I} p(e) \times \prod_{e\in O}(1-p(e))$
6. ELSE
7. FOR $i \leftarrow k$ TO $|V|-1$ DO
8. $I_i \leftarrow I \cup \{e_{k+1}, e_{k+2}, \cdots, e_i\}$
9. $O_i \leftarrow O \cup \{e_{i+1}\}$
10. $G \leftarrow (V, E \setminus O, W)$
11. $SetNumber(T,1,1)$
12. $S \leftarrow FindSwapEdge(G,T)$
13. IF $\exists e'((e_{i+1},e') \in S)$ THEN
14. $T' \leftarrow T \cup \{e'\} \setminus \{e_{i+1}\}$
15. $r \leftarrow r + IMST(I_i, O_i, T', \mathcal{G})$
16. RETURN r

4. Experiment Results

For the experiments, we utilize the block-random graph mode [22], which can generate both the Erdos-Renyi random graph and Scale-free random graph. The existence probabilities of each edge are uniformly generated between 0 and 1. These algorithms were implemented in C++, all experiments were performed on an HASEE K470P-i5 notebook with 2.4GHz CPU and 4GB RAM, running Windows 7.

Experiment 1: We consider that the main implicated graph is a complete graph with edge weights fixed at the same value. In this case, all the spanning trees are MSTs, and the total number of spanning tree is n^{n-2} according to theorem by Cayley [23].

Table 2 shows the results of two algorithms, *N* denotes the number of MSTs in the main implicated graph, K_n denotes that the main implicated graph is a complete graph with *n* vertices; the values *Ratios* denote the ratio of *FERM* to *IERM*. According to the theoretical analysis, the time complexity of algorithm *FERM* should be *n* times faster than algorithm *IERM*, but the code implementation to algorithm *IERM* is more complex, so the actual ratios *Ratios* are lower than the expected ratios.

Table 2. The Performance Evaluation in the Same Weight for K_n

		Time(s)			
G	*N*	*FERM*	*IERM*	*Ratios*	*Reliability*
K_5	125	0.00100	0.00100	1.0	0.729287
K_6	1296	0.00500	0.00500	1.0	0.949277
K_7	16807	0.08100	0.07000	1.1	0.812293
K_8	262144	1.35800	1.06700	1.2	0.981199
K_9	4782969	29.2060	22.1580	1.3	0.990147
K_{10}	100000000	671.767	521.091	1.3	0.990147

Experiment 2: We set the edge weight to a integer number between 1 and 100 since the costs or

distances are not all the same in the real world. In order to analyze the effect of the number of vertices and edges on the run-time, the experiment changes the number of vertices and edges, respectively. Fig. 6 shows run-time comparison chart for algorithms *FERM* and *IERM*. In the four subgraphs, the number of vertices is set to 100, 200, 300 and 400, respectively.

As the figure shows, the time cost of algorithm *IERM* is unrelated to the number of vertices, the run-time ratios of *FERM* to *IERM* increases with the number of vertices and edges. In Fig. 6(a), when the number of edges is equal to 3000, the value $N*m$ of the algorithm *IERM* is approximately equal to 2.1 million and the run-time is approximately equal to 28 seconds. Due to the complexity of the code, the run-time of the algorithm *IERM* agree with the experiment data. when the number of vertices is 400 and the number of edges is 3000, algorithm *IERM* is 14 times faster than algorithm *FERM*.

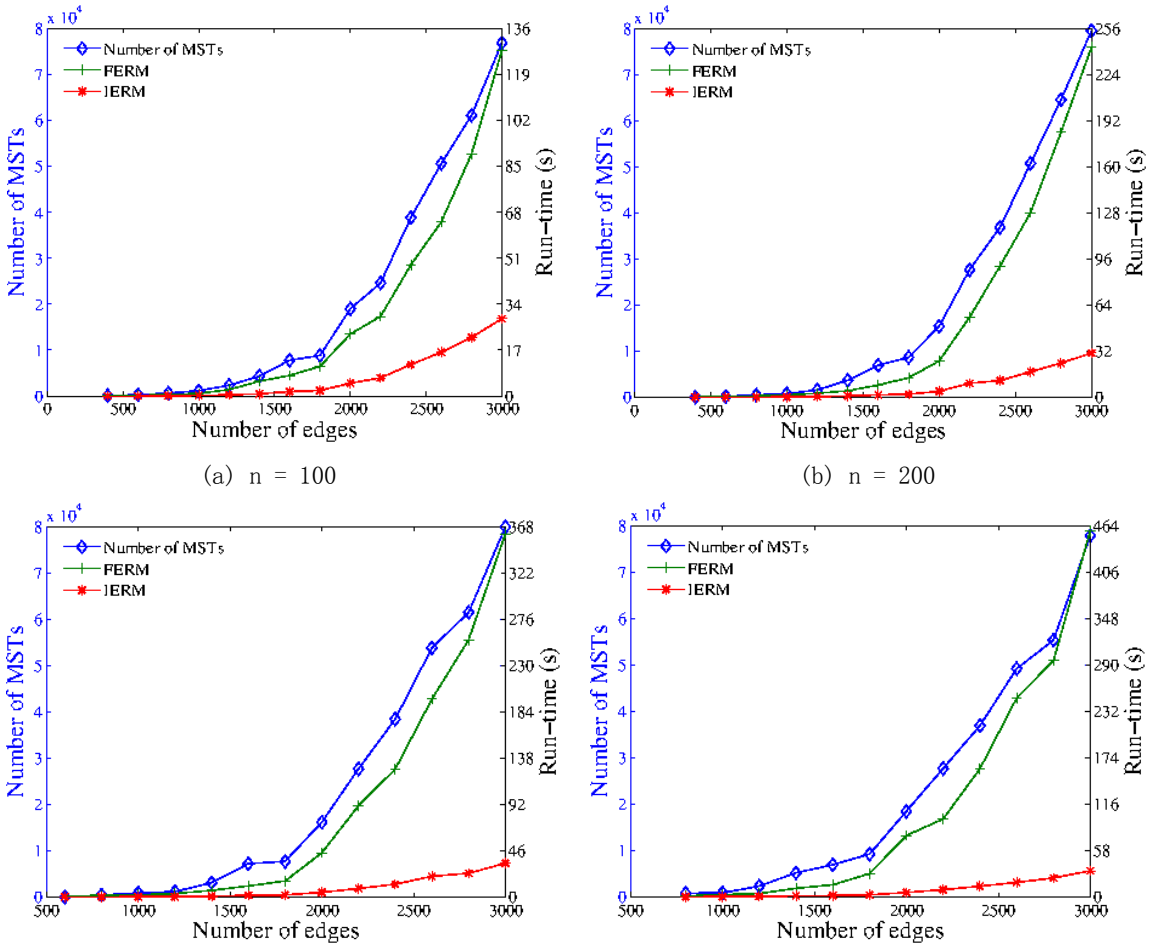


Fig. 6. The performance evaluation with different number of vertices.

5. Conclusions

This paper investigates the problem of evaluating reliability of the minimum spanning tree on uncertain graph for the first time and designs two algorithms to solve this problem base on classifying implicated graphs. The improved algorithm *IERM* proposes a novel approach to obtain a new MST, which time complexity is $O(m/n)$. The experiment results verify the efficiency of this algorithm and accuracy of theoretical analysis.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (GrantNo.61272295, GrantNo.61105039, and GrantNo.61202398), Hunan Provincial Innovation Foundation For Post-graduate (No. CX2014B276).

References

[1] Khan, M., Pandurangan, G., & Kumar, V. A. (2009). Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, *20(1)*, 124-139.

[2] Saravanan, M., Ravi, R., & Prabaharan, S. (2013). A survey on distributed algorithms for constructing minimum spanning trees in wireless sensor networks. *American-Eurasian Journal of Scientific Research*, *8(4)*, 192-197.

[3] Zhong, C., Miao, D., & Wang, R. (2010). A graph-theoretical clustering method based on two rounds of minimum spanning trees. *Pattern Recognition, 43(3)*, 752-766.

[4] Barzily, Z., Volkovich, Z., Akteke-Öztürk, B., & Weber, G. W. (2009). On a minimal spanning tree approach in the cluster validation problem. *Informatica, 20(2)*, 187-202.

[5] Päivinen, N. (2005). Clustering with a minimum spanning tree of scale-free-like structure. *Pattern Recognition Letters*, *26(7)*, 921-930.

[6] Li, J., Yang, S., Wang, X., Xue, X., & Li, B. (2009, July). Tree-structured data regeneration with network coding in distributed storage systems. *Proceedings of IWQoS. 17th International Workshop on Quality of Service* (pp. 1-9).

[7] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society: Vol. 7. No. 1* (pp. 48-50).

[8] Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal, 36(6)*, 1389-1401.

[9] Hintsanen, P., & Toivonen, H. (2008). Finding reliable subgraphs from large probabilistic graphs. *Data Mining and Knowledge Discovery, 17(1)*, 3-23.

[10] Zou, Z., Li, J., Gao, H., & Zhang, S. (2010). Mining frequent subgraph patterns from uncertain graph data. *IEEE Transactions on Knowledge and Data Engineering, 22(9)*, 1203-1218.

[11] Zou, Z., Gao, H., & Li, J. (2010, July). Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 633-642).

[12] Papapetrou, O., Ioannou, E., & Skoutas, D. (2011, March). Efficient discovery of frequent subgraph patterns in uncertain graph databases. *Proceedings of the 14th International Conference on Extending Database Technology* (pp. 355-366).

[13] Potamias, M., Bonchi, F., Gionis, A., & Kollios, G. (2010). K-nearest neighbors in uncertain graphs. *Proceedings of the VLDB Endowment*: *Vol. 3, No. 1-2* (pp. 997-1008).

[14] Dimitrov, D., Singh, L., & Mann, J. (2013, January). Comparison queries for uncertain graphs. *Database and Expert Systems Applications*, 124-140.

[15] Ruan, W., Wang, C., Han, L., Peng, Z., & Bai, Y. (2013). Uncertain subgraph query processing over uncertain graphs. *Web Technologies and Applications*, 132-139.

[16] Zou, Z., Li, J., Gao, H., & Zhang, S. (2010, March). Finding top-k maximal cliques in an uncertain graph. *Proceedings of IEEE 26th International Conference on Data Engineering (ICDE)* (pp. 649-652).

[17] Yuan, Y., Chen, L., & Wang, G. (2010, January). Efficiently answering probability threshold-based shortest path queries over uncertain graphs. *Database Systems for Advanced Applications*, 155-170.

[18] Bertsimas, D. J. (1990). The probabilistic minimum spanning tree problem. *Networks, 20(3)*, 245-275.

[19] Janiak, A., & Kasperski, A. (2008). The minimum spanning tree problem with fuzzy costs. *Fuzzy Optimization and Decision Making, 7(2)*, 105-118.
[20] Chen, X., Hu, J., & Hu, X. (2009). A polynomial solvable minimum risk spanning tree problem with interval data. *European Journal of Operational Research, 198(1)*, 43-46.
[21] Zou, Z., Li, J., Gao, H., & Zhang, S. (2009, November). Frequent subgraph pattern mining on uncertain graph data. *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (pp. 583-592).
[22] Karrer, B., & Newman, M. E. (2011). Stochastic blockmodels and community structure in networks. *Physical Review E, 83(1)*, 016107.
[23] Cayley, A. (1889). A theorem on trees. *Quart. J. Math, 23(376-378)*, 69.

Jie Tang received his B.S. degree in computer science and technology from Xiangtan University, Hunan, China in June 2012. He is currently working towards his M.S. degree in computer science at Xiangtan University, China. His current research interests includes uncertain graph and graph theory.

Yuansheng Liu received the B.S. degree in software engineering from Xiangtan University, Hunan, China, in 2012. He is currently working towards his M.S. degree in computer science and technology at Xiangtan University, China. His current research interests include date mining and machine learning.

Zhonghua Wen received his Ph.D. degree in computer science from Sun Yat-sen University, Guangdong, China. He is currently a professor and PhD supervisor with Xiangtan University. He is also the department head of the Department of Computer and Communication, Hunan Institute of Engineering. His research interests include nondeterministic planning, graph theory and algorithm.