# Minimum Spanning Tree on Uncertain Graphs

Anzhen Zhang(✉), Zhaonian Zou, Jianzhong Li, and Hong Gao

Harbin Institute of Technology, Harbin 150001, China
azzhang@hit.edu.cn

**Abstract.** In recent years, lots of data in various domain can be represented and described by uncertain graph model, such as protein interaction networks, social networks, wireless sensor networks, etc. This paper investigates the most reliable minimum spanning tree problem, which aims to find the minimum spanning tree (MST) with largest probability among all possible MSTs on uncertain graphs. In fact, the most reliable MST is an optimal choice between stability and cost. Therefore it has wide applications in practice, for example, it can serve as the basic constructs in a telecommunication network, the link of which can be unreliable and may fail with certain probability. A brute-force method needs to enumerate all possible MSTs and the time consumption grows exponentially with edge size. Hence we put forward an approximate algorithm in $O(d^2|V|^2)$, where $d$ is the largest vertex degree and $|V|$ is vertex size. We point out that the algorithm can achieve exact solution with expected probability at least $(1-(\frac{1}{2})^{(d+1)/2})^{|V|-1}$ and the expected approximation ratio is at least $(\frac{1}{2})^{d|V|}$ when edge probability is uniformly distributed. Our extensive experimental results show that our proposed algorithm is both efficient and effective.

## 1 Introduction

Recently, lots of data in various domain can be represented by graph model, such as the web, social networks, and cellular systems. Such networks are often subject to uncertainties caused by noise, incompleteness and inaccuracy in practice [1]. Incorporating uncertainty to graphs leads to uncertain graphs, each edge of which is associated with an edge existence probability to quantify the likelihood that this edge exists in the graph. For example, in a telecommunication network, a link can be unreliable and may fail with certain probability [14]; in a social network, the probability of an edge may represent the uncertainty of a *link prediction* [6]; in a wireless sensor network, communication links between sensor nodes often suffer from inevitable physical interference [15]. The uncertain graph, also referred as probabilistic graph, addresses such scenarios conveniently in a unified way.

There has been extensive research on the minimum spanning tree on exact graphs which are precise and complete. Given a connected exact graph $G = (V, E)$, each edge has a non-negative weight, a spanning tree $T$ of $G$ is a tree whose edges connect all the nodes in $G$, the sum weight of all edges in $T$ is the cost

© Springer International Publishing AG 2016
W. Cellary et al. (Eds.): WISE 2016, Part II, LNCS 10042, pp. 259–274, 2016.
DOI: 10.1007/978-3-319-48743-4_21

of $T$. Among all spanning trees the one with minimum cost is minimum spanning tree, which is short for MST. A number of algorithms have been proposed to compute MST for exact graph, among which Prim and Kruskal gave two classical algorithms in polynomial time [2,3].

The problem of computing MST is fundamental for uncertain graphs, just as they are for exact graphs. Obviously we can still obtain the MST with minimum cost in uncertain graph by neglecting the possibility of edges and treat the uncertain graph as exact graph. We call the obtained MST *minimum cost MST*, however, it suffers from low reliability and may fail to exist since we leave the possibility of edges out of consideration. Actually we can obtain the MST with largest probability by setting the weight of edge $e$ to be $-log(p(e))$ and executing MST algorithm for exact graphs. The MST obtained this way has the largest existing probability but has no guarantee for the cost, we call it *maximum probability MST* for simplicity. To make a balance between those two types of MST, we propose *the most reliable MST* which has largest probability among all possible MSTs and it can be found in a wide range of network applications.

In a telecommunication network, the most reliable MST can serve as the basic constructs to help connect all nodes in the network at least cost while the stability of network can still be guaranteed [16]. In the scenario of advertisement promotion in social network with edge uncertainty measures the intimacy between two friends and edge weight quantifies the cost of spreading, the most reliable MST can help discover a most effective propagation path with minimum advertisement cost. Another practical application is data aggregation tree in wireless sensor network [15]. Most reliable MST can serve as an optimal initial data aggregation tree with sink node to be the root and source nodes to be leafs since it takes transmission energy cost and link failure probability into consideration, which contributes to the construction of data aggregation tree that maximizes the network lifetime.

In uncertain graph, possible instantiations of the graph are commonly referred to as worlds or implicated graph, the probability of a world is calculated based on the probability of its edges, as will shown in later section. There exists at least one MST in each connective implicated graph, all MSTs of the connective implicated graphs form the MST set of the uncertain graph and the most frequent MST in the set is the most reliable one, denoted by $MST_{max}$.

A brute-force method is to enumerate all connected implicated graphs and compute MST for each of them, however, enumerating all implicated graph needs $O(2^{|E|})$time, where $|E|$ is the edge size. The exact algorithm is not feasible since the scale of graph is large in practical applications. Hence we present an approximate algorithm in $O(d^2|V|^2)$, where $d$ is the maximum degree of vertexes and $|V|$ is vertex size. Our theoretical analysis shows that it has pretty good performance in aspects of accuracy and approximate ratio. Extensive experiment results on synthetic sets show that our approximate algorithm outperforms the other three algorithms in terms of stability and cost.

The rest of the paper is organized as follows. We define the most reliable MST problem in Sect. 2. An approximate algorithm and its performance analysis is presented in Sect. 3. We show experimental studies in Sect. 4 and present related work in Sect. 5. Finally, we conclude this paper in Sect. 6.

## 2 Problem Formulation

In this section, we formally present the uncertain graph model and introduce the problem of most reliable MST in an uncertain graph.

### 2.1 Model of Uncertain Graphs

Let $\mathcal{G} = (V, E, P, W)$ be an uncertain graph, where $V$ and $E$ denote the set of vertexes and edges respectively. $P : E \rightarrow (0, 1]$ is a function assigning existence possibility values to edges. $W$ is weight function, and $w(e)$ is the weight of edge $e \in E$.

An uncertain graph has many existence forms due to the uncertainty of edges, each deterministic form $g = (V, E_g)$ is called *implicated graph* and is denoted by $\mathcal{G} \Rightarrow g$. Each edge $e \in E_{\mathcal{G}}$ is selected to be an edge of $g$ with probability $P(e)$. The total number of implicated graphs is $2^{|E|}$ since each edge has two cases as to whether or not that edge is present in the graph. We assume that all existence possibilities of edges are independent, the probability of an uncertain graph $\mathcal{G}$ implicating an exact graph $g$ is

$$P(\mathcal{G} \Rightarrow g) = \prod_{e \in E(g)} P(e) \prod_{e' \in E(G) \setminus E(g)} (1 - P(e')) \tag{1}$$

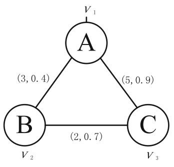
where $P(e)$ is the existence possibility of edge $e$. To gain more intuition on the uncertain graph model and the implicated graph, We present a simple example in the following.

*Example 1.* Consider the uncertain graph $\mathcal{G}$ in Fig. 1(a). There are $2^3 = 8$ implicated graphs, and the probability of each graph is calculated based on Eq. (1), as shown in Fig. 1(b). Take $g2$ for example, $P(\mathcal{G} \Rightarrow g2) = 0.4 \times (1 - 0.9) \times (1 - 0.7) = 0.012$. The probability of the other implicated graphs are calculated in the same way.

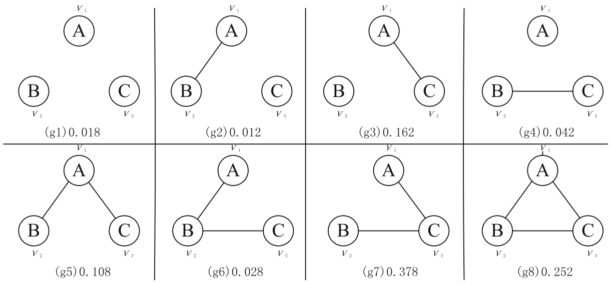
### 2.2 Most Reliable MST

There exists at least one MST in each connective implicated graph of uncertain graph $\mathcal{G}$ with probability same as the implicated graph containing it, all possible MSTs form MST set $\{MST_1, MST_2, \cdots\}$, and $MST_i$ denotes the edge set of the $i$th MST.

Obviously a MST for an implicated graph may be MST for another implicated graph, hence each MST in MST set is accompanied with certain frequency and the one with maximum frequency is most reliable, that is $MST_{max}$. The existing probability $P(MST_i)$ of $MST_i$ is determined by the implicated graph it belongs



(a) Uncertain graph G



(b) Imp(G)

**Fig. 1.** A simple example

to. As we mentioned above, $MST_i$ could be MST of several implicated graphs, so $P(MST_i)$ is the sum probability of all implicated graphs whose MST is $MST_i$ and can be mathematically quantified as follows.

$$P(MST_i) = \sum_{g \in Imp(G)} P(G \Rightarrow g) \cdot I_1(g) \cdot I_2(g) \tag{2}$$

where $I_1(g)$ and $I_2(g)$ are indicator functions.

$$I_1(g) = \begin{cases} 1 & \text{if } g \text{ is connected} \\ 0 & \text{otherwise} \end{cases}$$

$$I_2(g) = \begin{cases} 1 & \text{if } MST_i \text{ is MST of } g \\ 0 & \text{otherwise} \end{cases}$$

Now we give the formal definition of $MST_{max}$, that is the MST with maximum probability.

$$MST_{max} = argmax\{P(MST_i)\} \tag{3}$$

In the following example, we show how to exactly compute $MST_{max}$ of uncertain graph $\mathcal{G}$ in Fig. 1(a).

*Example 2.* There are 8 implicated graphs of $\mathcal{G}$ as shown in Fig. 1(b), only 4 of them are connected, namely, $g5$, $g6$, $g7$ and $g8$. After computing the MST of

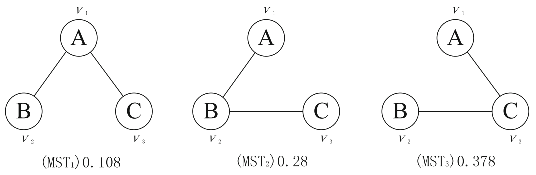


**Fig. 2.** MST set of graph G

the four subgraphs respectively, we get three different MSTs, $MST_1$, $MST_2$ and $MST_3$ as shown in Fig. 2. The MSTs of $g6$ and $g8$ are actually same to each other, that is $MST_2$. We can compute the existing probability of $MST_i$ using Eq. (3), $i \in 1, 2, 3$.

$$P(MST_1) = P(\mathcal{G} \Rightarrow g5) = 0.108$$
$$P(MST_2) = P(\mathcal{G} \Rightarrow g6) + P(\mathcal{G} \Rightarrow g8) = 0.028 + 0.252 = 0.28$$
$$P(MST_3) = P(\mathcal{G} \Rightarrow g7) = 0.378.$$

According to Eq. (3), $MST_{max}$ is the MST with largest probability in MST set, which refers to $MST_3$ in our example, thus we obtain the most reliable MST $MST_{max} = \{AC, BC\}$, and $P(MST_{max}) = P(MST_3) = 0.378$. We denote the cost of MST as $|MST|$, then $|MST_{max}| = |MST_3| = 5 + 2 = 7$.

An intuitive way to compute $MST_{max}$ is to enumerate all implicated subgraphs and compute corresponding MSTs of the connected subgraphs, each MST is with a corresponding existing probability and the one with largest probability is the $MST_{max}$. Detailed algorithm is shown in Algorithm 1.

**Algorithm 1.** Naive Algorithm

Input: Uncertain graph $\mathcal{G} = (V, E, P, W)$
Output: The edge set $A$ of $MST_{max}$

```
1: $MST\_Edge = \varnothing; MST\_P = 0; MST\_W = 0; Imp = \varnothing;$
2: $map < Edge, (P, W) > MST\_SET$
3: **for** $i \leftarrow 2^{|V|-1} - 1$ to $2^{|E|} - 1$ **do**
4:    Transform $i$ to its binary form, denoted by $i\_binary$ // i.e.100111
5:    **if** The number of 1's in $i\_binary$ is not smaller than $|V| - 1$ **then**
6:       Add all edges whose index corresponds to 1 in $i\_binary$ in $Imp$
7:       $MST\_P = \prod_{e \in Imp} P(e) \prod_{e' \in E \setminus Imp} (1 - P(e'))$
8:       $MST\_Edge, Imp\_W$ = Prim($Imp$)
9:       **if** $MST\_Edge$ is not in $MST\_SET$ **then**
10:         Add ($MST\_Edge, (MST\_P, MST\_W)$) in $MST\_SET$
11:      **else if** $MST\_Edge$ is in $MST\_SET$ **then**
12:         Modify $P = MST\_P \cdot P$ in original $< Edge, (P, W) >$ pair whose $Edge =$
            $MST\_Edge$
13: return $Edge$ with maximum probability in $MST\_SET$
```

In Naive Algorithm, we enumerate all implicated algorithms and compute MST using Prim algorithm. To find the MST with largest frequency, we use a map whose key is edge set of MST, and value is a pair of probability and weight, the map can hash all MSTs with same edge set together. After computing all MSTs of connected implicated graphs, we find the one with largest probability, that is $MST_{max}$. There are total $2^{|E|}$ implicated graph of $\mathcal{G}$, the efficiency of the brute-force algorithm is unsatisfactory when applied to large-scale graphs due to its exponential growing rate.

## 2.3 Edge Induced Combinational Method

According to the definition of most reliable MST in Sect. 2.2, it is not easy to find the most reliable MST in polynomial time. However, we find out another way to compute the probability of MST in a combinational way. Without lose of generality, we suppose that the edge weight is different from the others. For $MST_i$ in MST set, the set of all edges not in $MST_i$ is denoted by $R_i$, $R_i = E - MST_i$ by definition. For an edge $e_i$ in $R_i$, adding $e_i$ in $MST_i$ will form a circle due to the connectivity of $MST_i$. Next we give the definition of safe edge and dangerous edge in $R_i$.

**Definition 1.** *Safe edge. For an edge $e_i$ in $R_i$, it is a safe edge to $MST_i$ if it has largest weight in the circle when adding $e_i$ in $MST_i$.*

**Definition 2.** *Dangerous edge. For an edge $e_i$ in $R_i$, it is a dangerous edge to $MST_i$ if it does not has largest weight in the circle when adding $e_i$ in $MST_i$, in other words, there exists an edge whose weight is larger than $e_i's$.*

Based on the above definition, we divide the remaining edge set $R_i$ into two separate sets, namely, $RS_i$ and $RD_i$. All safe edges in $R_i$ are placed in $RS_i$, and all dangerous edges are in $RD_i$. Now we move on to give a combinational way to compute the probability of $MST_i$ and we prove that the probability computed in this way is same as that given by Eq. 2.

$$P(MST_i) = \prod_{e_i \in MST_i} p(e_i) \cdot \prod_{e_j \in RD_i} (1 - P(e_j)) \tag{4}$$

**Theorem 1.** *The probability calculated by Eq. 4 is equal to that obtained by Eq. 2.*

*Proof.* We only give a sketch of the proof, the detailed proof is omitted due to page limit. Apparently, the $MST_i$ itself is a implicated graph whose MST is $MST_i$, based on this we can further add any safe edge to $MST_i$ and they will not affect the MST of newly constructed implicated graphs, that is $MST_i$. This is because safe edge has largest weight in the circle and will be discarded. Further more, we can prove that any dangerous edge added to $MST_i$ will affect the original structure, therefore all edges in $RD_i$ can not exist, which is shown in Eq. 4.

*Example 3.* $BC$ is a dangerous edge to $MST_1$, according to Eq. 4, we have $P(MST_1) = P(AB) \cdot P(AC) \cdot (1 - P(BC)) = 0.4 \times 0.9 \times (1 - 0.7) = 0.108$, which is same as the result in Example 2. $P(MST_2)$ and $P(MST_3)$ can be computed in a similar way.

## 3 Greedy Algorithm

In this section, we present an approximate algorithm of the most reliable MST. The detailed algorithm description is shown in Sect. 3.1 and we analyse the performance of our approximate algorithm in Sect. 3.2. The complexity analysis is in Sect. 3.3.

### 3.1 Algorithm Description

The greedy algorithm on uncertain graph is similar to Prim algorithm on exact graph. Specifically, given connective uncertain graph $\mathcal{G} = (V, E, P, W)$, we maintain a tree $A$, which starts from a random root $r$ and spans an edge at each step until $A$ covers all nodes in $V$. At each step, a *light edge with largest probability* connecting $A$ and an isolated node in $\mathcal{G}_A = (V, A)$ will be added in $A$, $\mathcal{G}_A = (V, A)$ is a forest whose node set is same to $\mathcal{G}$, but edge set is $A$. Initially $\mathcal{G}_A = (V, A)$ is a forest with $|V|$ isolated nodes, with the spanning of $A$, $\mathcal{G}_A$ adds edges in $A$ increasingly. Intuitively $A$ spans an edge whose one endpoint is in $A$ but the other one is not. *light edge* refers to edge with minimum weight in $E$, the *light edge with largest probability* is defined as follows.

**Definition 3.** *Light edge with largest probability(LELP). Add all edges connecting $A$ and isolated node in forest $\mathcal{G}_A = (V, A)$ into queue $S$, sort $S$ by edge weight in ascending order, for the ith edge in $S$, say $e_i$, the probability of adding $e_i$ to $A$ is calculated by Eq. 5, denoted by $\widehat{P}(e_i)$, which is called* join probability*, the edge with largest join probability in $S$ is LELP.*

$$\widehat{P}(e_i) = (\prod_{j=1}^{i-n_i-1} (1 - P(e_j))) \cdot P(e_i) \tag{5}$$

where $1 \leq i \leq |S|$. $n_i$ is the number of edges whose weight is same to the $i$th edge but position is ahead of it.

The complete algorithm is outlined in Algorithm 2. The input is an uncertain graph $\mathcal{G}$ and the edge set $A$ of $MST_{max}$ is the output. The probability of $A$ is given in Eq. 6, which is the product of the join probability of all edges in $A$. To gain a better understanding of Algorithm 2, we compute $MST_{max}$ of a simple uncertain graph step by step in the following example.

$$\hat{P}(A) = \prod_{e_i \in A} \hat{P}(e_i) \tag{6}$$

**Algorithm 2.** Greedy Algorithm

```
Input: Uncertain graph $\mathcal{G} = (V, E, P, W)$
Output: The edge set $A$ of approximate $MST_{max}$

 1: $MST\_V = \varnothing; A = \varnothing; S = \varnothing; \hat{P}(A) = 1$
 2: Randomly select a root node , say $r$, add it in $MST\_V$
 3: Add all edges connected with $r$ in queue $S$
 4: while $|MST\_V| < |V|$ do
 5:     Sort $S$ by weight in ascending order
 6:     Calculate the probabality of each edge joinning in $A$ by Eq. 5;
 7:     Get the edge with maximum probabiity by max heap, say $(u, v)$
 8:     Add $(u, v)$ in $A$, suppose $u$ is already in $MST\_V$, $v$ is not,then add $v$ in $MST\_V$
 9:     Update $\widehat{P}(A) = \widehat{P}(A) \cdot \hat{P}(u, v)$
10:     Delete all edges connecting $v$ in $S$
11:     Add edges whose one endpoint is $v$ but the other endpoint is not in $MST\_V$ in $S$
12: return A
```

*Example 4.* The input uncertain graph $\mathcal{G}'$ is in Fig. 3 with four vertexes and four edges, the weight and existence probability are labeled as binary group on edges.

Initially, we select a node randomly, say $a$, add $a$ in $MST\_V$ and add edges connecting $a$ in $S$, that is $(a, b)$ and $(a, h)$. sort $S$ in ascending order of weight. We calculate the probability for each edge adding in $A$ using Eq. 4. $\hat{P}(a, b) = P(a, b) = 0.8$, $\hat{P}(a, h) = (1 - P(a, b)) \cdot P(a, h) = 0.12$. We maintain a max heap $H$ to obtain the edge with maximum probability, $(a, b)$ in this case. Add $(a, b)$ in $A$ and the new node $b$ in $MST\_V$.

Next adjust $S$, delete edges which contains vertex $b$ in $S$, that is $(a, b)$, then add all edges whose one endpoint is $b$ but the other one is not in $MST\_V$, that is $(b, h)$ and $(b, c)$. Sort $S$ again according to edge weight. Compute the probability of edges in $S$, $\hat{P}(a, h) = P(a, h) = 0.6$, $\hat{P}(b, c) = P(b, c) = 0.2$, $\hat{P}(b, h) = (1 - P(a, h)) \cdot (1 - P(b, c)) \cdot P(a, b) = 0.224$. The edge with largest probability is $(a, h)$, add it to $A$ and vertex $h$ in $MST\_V$ and delete $(a, h)$ and $(b, h)$ in $S$. Only $(b, c)$ is in $S$, so we add it in $A$ directly and insert vertex $c$ into $MST\_V$, the $MST_{max}$ edge set $A = (a, b)(a, h)(b, c)$, $\hat{P}(MST_{max}) = \widehat{P}(a, b) \cdot \hat{P}(a, h) \cdot \hat{P}(b, c) = 0.096$.

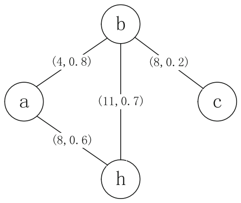


**Fig. 3.** Uncertain graph $G'$

## 3.2 Greedy Selectivity

In this section, we will evaluate the performance of the greedy algorithm we proposed from two aspects, namely, accuracy rate and approximate ratio. To begin with, we analyse the greedy selectivity of our problem.

Suppose we have already known the structure of $MST_{max}$, that is edges in $MST_{max}$ are given, we can redefine the join probability $\widehat{P}(e_i)$ as Eq. 7. For the $i$th edge $e_i$ in queue $S$, we put all edges whose weight is lighter than $ei$ in a set $SA_i$, $SA_i = \{e_1, e_2 \ldots e_{i-n_i-1}\}$.

$$\widehat{P}^{'}(e_i) = P(e_i) \cdot ( \prod_{j=1}^{i-n_i-1} (1 - P(e_j)) + \sum_{k=1}^{|SA_i|} \sum_{e_{z1} \ldots e_{zk} \in SA_i} \{\prod_{x=1}^{x=k} P(e_{zx}) \cdot \prod_{\substack{e_m \neq e_{zj} \\ j \in [1,k] \\ e_m \in SA_i}} (1 - P(e_m)) \cdot I_3(e_{z1} \ldots e_{zk})\}) \tag{7}$$

where $I_3(e_{z1} \ldots e_{zk})$ is a indicator, which indicates whether there exists a path from $e_{zx}$ to $e_i$ in $MST_{max}$ so that $w(e_i)$ is smaller than some edge on that path, $x \in [1, k]$. If no such edge exists, another case in which $I_3(e_{z1} \ldots e_{zk}) = 1$ is that there is no path from $e_{zx}$ to $e_i$ for all $x \in [1, k]$.

$$I_3(e_{z1} \ldots e_{zk}) = \begin{cases} 1 & \exists x \in [1,k], w(e_i) \text{is smaller than} \\ & \text{some edge on the path from } e_{zx} \\ & \text{to } e_i \text{ or } \forall x \in [1,k] \text{ there has no} \\ & \text{path from } e_{zx} \text{ to } e_i \text{ in } MST_{max} \\ 0 & \text{otherwise} \end{cases}$$

We modify line 6 of Algorithm 2 by using Eq. 7 instead of Eq. 5, the other lines remain unchanged. The MST obtained this way is denoted by $MST_{new}$ and the MST obtained by original algorithm is named $MST_{old}$. The probability of $MST_{new}$ is $P(MST_{new}) = \prod_{e_i \in MST_{new}} \hat{P}^{'}(e_i)$ and we can prove the following theorem is true. We omit the proof due to the page limit.

**Theorem 2.** *For 2-connected uncertain graph $\mathcal{G} = (V, E, P, W)$, $MST_{new}$ obtained from modified Algorithm 2 is same as $MST_{max}$ in Algorithm 1, that is they have the same edge set and their probability and weight are equal. Formally, $MST_{new} = MST_{max}$, $P(MST_{new}) = P(MST_{max})$ and $W(MST_{new}) = W(MST_{max})$*

Next we will analyse the performance of the approximate algorithm we proposed in Sect. 3.1. Suppose the queue $S$ contains $\{e_1, e_2, e_3 \cdots\}$ in ascending order of their weight currently, then we have the following lemma.

**Lemma 1.** *If the existence probability of edge obeys uniform distribution in (0,1), then the probability of $\widehat{P}^{'}(e_1) > \widehat{P}^{'}(e_k)$ is at least $\frac{1}{2}$ for $k > 1$.*

*Proof.* $E[P(e_i)] = \frac{1}{2}$, $P(P(e_i) \geq \frac{1}{2}) = \frac{1}{2}$, $P(P(e_i) < \frac{1}{2}) = \frac{1}{2}$, $\widehat{P}'(e_1) = P(e_1)$ and $\widehat{P}'(e_2) = P(e_2)[(1 - P(e_1)) + P(e_1)I_3(e_1)]$, suppose $P(I_3(e_1) = 0) = p_0$.

The first case is that $P(e_1) \geq \frac{1}{2}$ and $P(e_2) \geq \frac{1}{2}$. If $I_3(e_1) = 0$, $\widehat{P}'(e_2) = P(e_2) \cdot (1 - P(e_1)) < \frac{1}{2}P(e_2) < \frac{1}{2} < P(e_1) = \widehat{P}'(e_1)$. However, if $I_3(e_1) = 1$, $\widehat{P}'(e_2) = P(e_2) > \frac{1}{2}$, then $\widehat{P}'(e_1)$ has $\frac{1}{2}$ probability larger than $\widehat{P}'(e_2)$. Thus the probability in this case is $\frac{1}{2} \cdot \frac{1}{2} \cdot [p_0 + (1 - p_0) \cdot \frac{1}{2}]$. The probability in the other three cases can be computed in a similar way. The total probability of $\widehat{P}'(e_1) > \widehat{P}'(e_2)$ is $\frac{1}{2} + \frac{7p_0}{48}$. Besides, the probability of $\widehat{P}'(e_1) > \widehat{P}'(e_k)$ is obviously larger than $\frac{1}{2}$ for $k > 2$.

**Theorem 3.** *For 2-connected uncertain graphs* $\mathcal{G} = (V, E, P, W)$*, if the edge probability is independent and identically distributed in* $(0, 1)$ *uniformly, the greedy algorithm can obtain the accurate* $MST_{max}$ *with expected probability at least* $(1 - (\frac{1}{2})^{d/2})^{|V|-1}$.

*Proof.* In our former analysis, we should select an edge $e_i$ with largest $\widehat{P}'(e_i)$ at each step, so that we can obtain the accurate $MST_{max}$. However, we apply $\hat{P}(e_i)$ in our approximate algorithm, there are two cases that $e_i$ can still be selected in $MST_{max}$. The first case is that $e_i$ with largest $\widehat{P}'(e_i)$ also has largest $\hat{P}(e_i)$ among all candidate edges in queue $S$. The second case is that all indicate function $I_3(e_k) = 0$ for $k \in [1, i-1]$. The expected correct probability for $e_i$ is at least $(\frac{1}{2} + \frac{1}{2} \cdot (1 - (\frac{1}{2})^{(d-1)/2}))$. The detailed proof is omitted due to the page limit.

**Theorem 4.** *The expected approximate ratio is at least* $(\frac{1}{2})^{d|V|}$.

*Proof.* We consider the worst case in which $\widehat{P}'(e_i) = P(e_i)$ but $\widehat{P}(e_i) = P(e_i) \cdot \prod_{k=1}^{k=i-1}(1 - P(e_k))$, the approximation ratio is $r = \prod_{k=1}^{k=i-1}(1 - P(e_k))$. Due to $P(e_i)$ is independent random variable, we have $E[\prod P(e_i)] = \prod E[P(e_i)]$, hence $E[r] = E[\prod_{k=1}^{k=i-1}(1 - P(e_k))] = \prod_{k=1}^{k=i-1} E[(1 - P(e_k))] = (\frac{1}{2})^{i-1}$, for $|V| - 1$ edges, the ratio is $r^{|V|-1} > (\frac{1}{2})^{d|V|}$

## 3.3 Complexity Analysis

In this section, we analyse running time in the worst case. We denote the maximum vertex degree as $d$, it is obvious to see $1 \leq d \leq |V - 1|$, the length of $S$ in $i$th iteration is denoted as $|S_i|$, then we have the following relations:

$$\begin{cases} |S_1| \leq d \\ |S_{i+1}| \leq (|S_i - 1|) + (d - 1) \end{cases} \tag{8}$$

The general term formula of arithmetic progression is $|S_i| \leq (d - 2) \cdot i + 2 = O(di)$. The total run time is $T(n) = \sum_{i=1}^{|V|}(O(d^2 i) + O(lgdi) + O(1) + O(di) + O(d)) = O(d^2|V|^2)$.

## 4 Experiments

In this section, we present experimental results studying the effectiveness and efficiency of greedy algorithm.

### 4.1 Environment and Datasets

Our algorithms were implemented using C++ and the Standard Template Library(STL), and were conducted on a 2.4 GHz Dual Core Intel(R) core(TM) CPU with 2.0 GB RAM running Ubuntu 12.04.

We conduct our experiments on two kinds of synthetic datasets, one of which is generated from real datasets and the other is generated randomly. The first dataset is obtained by assigning a random weight to each edge of real uncertain graphs, the weight is a integer among $[0, 100]$, The real datasets in our experiments are Nature and Flickr, Nature is a protein-protein interaction(PPI) uncertain graph and Flickr is a social network, the scale and connectivity of these two graphs is shown in Table 1, where $N(MST)$ is the number of connected components in graph.

**Table 1.** Synthetic datasets

| Uncertain graph | Graph scale(V,E) | Connectivity | N(MST) |
|---|---|---|---|
| Nature | (2708,7123) | No | 63 |
| Flickr | (21594,1008258) | No | 1732 |

The second datasets is generated randomly, to be specific, the edges between vertexes and the edge weight and probability are generated randomly after fixing vertex size, edge weight is a integer among $[0, 100]$ and the probability is among $(0, 0.99]$. We generated three random datasets, the first one is characterized by its average vertex degree, which is 1.23, but the size of vertexes grows from 1k to 10k, we denote this dataset as *man-made1*. The second dataset contains 10 graphs whose vertex sizes are fixed to be 1k, but average vertex degree grows from 3 to 7.5. The third dataset is a set of small uncertain graphs whose vertex size is between 4 to 50 and average degree is 3, we denote it as *man-made3*.

### 4.2 Analysis of Greedy Algorithm

If the uncertain graph is not connective, we calculate the most reliable forest $MSF_{max}$. Specifically, when greedy algorithm terminates with a spanning tree $A$, the vertex size of $A$ is smaller than that of $G$, we add $A$ in $MSF_{max}$ and randomly select a new root node not in $A$ at the same time, repeat this process until all vertexes in $G$ are covered by $MSF_{max}$. Next, we analyse the effect of vertex size $|V|$ and average vertex degree $\overline{d}$, as they are two main factor affect the runtime of greedy algorithm.

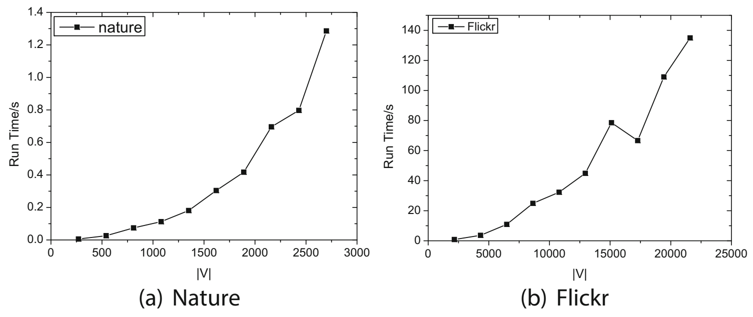


**Fig. 4.** Execution time with different $|V|$

**Effect of $|V|$.** In this experiment, we tested the runtime under different graph scales. We extracted subgraphs from 10 % to 100 % of Nature and Flickr and executed greedy algorithm on subgraphs separately. the results are shown in Fig. 4.

Through the curve in Fig. 4, we can see runtime exhibited parabolically trend increase with the $|V|$, which agrees with our theoretical analysis in Sect. 3.3. However, when we look into Fig. 4(b) carefully we find that the runtime at graph scale (17273,102356) is less than that at (15113,97743) , this is because the average vertex degree $\overline{d}$ of graph (17273,102356) is smaller than that of graph (15113,97743). Therefore we tested the effect of $\overline{d}$ on running time in following experiments.

**Effects of $\overline{d}$.** We ran greedy algorithm on man-made1 dataset, since the average degree on man-made1 is set to be 1.23, we can examine the single effect of $|V|$ on runtime, as shown in Fig. 5(a). The result shows that runtime grows more smoothly when $\overline{d}$ is fixed and no outliers occur. Next we fix $|V|$ and test the effect of $\overline{d}$ on runtime, we apply greedy algorithm on man-made2 dataset whose vertex size is fixed to be 1k and the edge size grows from 3k to7.5k, which means

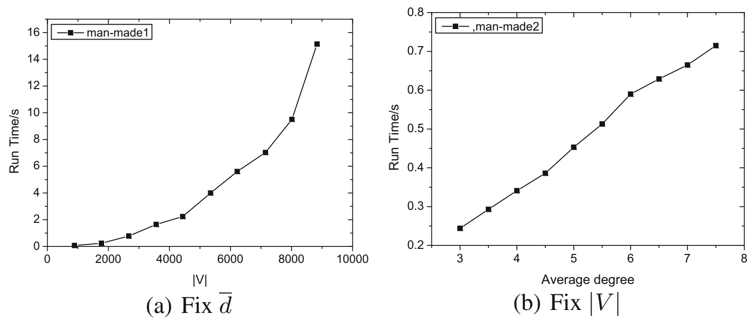


**Fig. 5.** Execution time with different $\overline{d}$

$\overline{d}$ is in $[3, 7.5]$, the result of this experiment is shown in Fig. 5(b). We can see that runtime grows linearly as $\overline{d}$ increases when $|V|$ is fixed.

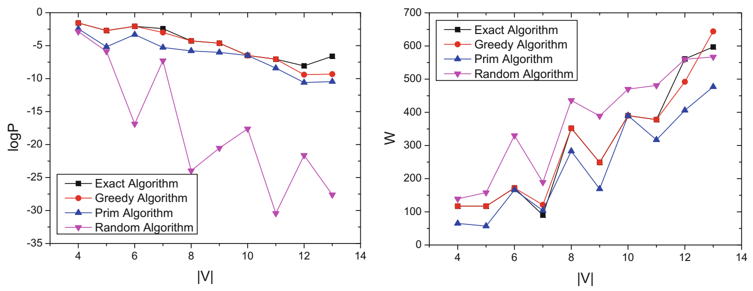
**Accuracy.** To quantify the accuracy of greedy algorithm, we compare probability and weight of resulting MST with the other three algorithms. We conduct our experiment on man-made3 dataset since the time complexity of exact algorithm is in $O(2^{|E|})$. Furthermore, the probability of MST is very small for graphs whose edge existing probability is relatively small, with the increasing of edges in MST, the probability decreases sharply and it is not convenient for us to record and analyse. Hence we amplify the probability of each edge by computing its log value since log function increases monotonically. Here we have the equation $log \prod_i P_i = \sum_i logP_i$.

We design two contrast experiments to see the effectiveness of our proposed greedy algorithm. The first one is adapted Prim algorithm, which obtains the MST with minimum cost by neglecting the probability on each edge. The other one is a random algorithm, we randomly select one edge and add it in MST.

We compare the probability and weight of MSTs obtained from four algorithms, namely, exact algorithm, greedy algorithm, Prim algorithm and Random algorithm, as shown in Fig. 6(a) and (b). From the figures we can see that the probability and weight of MST obtained by greedy algorithm is same as exact algorithm in most cases. Furthermore, greedy algorithm provides a better approximation to exact solution on probability compared with the other two algorithms.

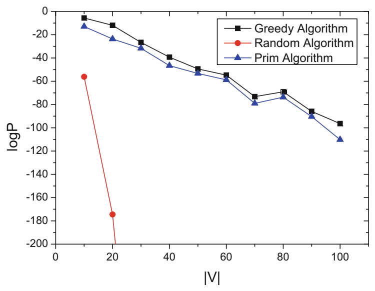
Next we extend our experiment on larger scale graphs without exact algorithm as shown in Fig. 7(a) and (b). We can see greedy algorithm can achieve better probability all the time with a little loss of weight.

Through the experiments conducted above, we come to the following conclusions. First, there are mainly two factors that effect runtime of greedy algorithm, vertex size and vertex degree, furthermore, the grow trend of runtime roughly agrees with our theoretical analysis. Second, our greedy algorithm provides a good approximation to exact algorithm not only on probability but also on weight.

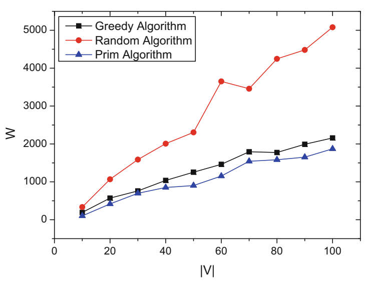


(a) compare logP (b) compare weight

**Fig. 6.** Compare exact algorithm and greedy algorithm



(a) compare logP



(b) compare weight

**Fig. 7.** Compare greedy algorithm and the other two algorithm

## 5 Related Work

The minimum spanning tree problem have been studied extensively in the literature under the term stochastic geometry. The main work focus on computing the expected lengths of the MST in stochastic graphs, to the best of our knowledge, there has no previous work on most reliable MST in uncertain graph. To begin with, we survey the work about MST in stochastic graph model, specifically, it mainly includes existential uncertainty model, locational uncertain model and randomly weighted graph model.

*Existential Uncertain Model.* Given a complete, weighted undirected graph $G = (V, E)$, on $n$ node and $m$ edges, called the master graph, where each node $v_i$ is active(or present) with independent probability $p_i$. When a node is inactive, all of its incident edges are also absent. We compute the expected minimum spanning tree cost for $G$, namely, $\sum p(H)MST(H)$, where the sum is over all node-induced subgraphs $H$ of $G$, $p(H)$ is the probability with which $H$ appears, and $MST(H)$ is the cost of its minimum spanning tree. This problem has been proven to be #P-hard by Kamousi and Suri in [7].

*Locational Uncertainty Model.* Given a metric space $P$. The location of each node $v \in V$ in the stochastic graph $G$ is a random point in the metric space and the probability distribution is given as the input. We assume the distributions are discrete and independent of each other. We use $p_{vs}$ to denote the probability that the location of node $v$ is point $s \in P$. The expected length of MST is $E[MST] = \sum_{\mathbf{r} \in R} Pr[\mathbf{r}] \cdot MST(\mathbf{r})$, where $\mathbf{r}$ is a realization of $G$ and can be represented by an $n$-dimensional vector $(r_1, \ldots, r_n) \in P^n$, where point $r_i$ is the location of node $v_i$ for $1 \leq i \leq n$, $\mathbf{r}$ occurs with probability $Pr[\mathbf{r}] = \prod_{i \in [n]} p_{v_i r_i}$, $MST(\mathbf{r})$ is the length of the minimum spanning tree spanning all points in $\mathbf{r}$. Huang and Li in [8] showed that computing $E[MST]$ in this model is also #P-hard.

*Randomly Weighted Graph Model* . In this model edge weights are independent nonnegative variables, Frieze and Steele in [9,10] showed that the expected value of the minimum spanning tree on such a graph with identically and independently distributed edges is $\varsigma(3)/D$ where $\varsigma(3) = \sum_{j=1}^{\infty} 1/j^3$ and $D$ is the derivative of the distribution at 0.

Another line is network reliability problem, which computes a measure of network reliability given failure probabilities for the arcs in a stochastic network where each arc can be in either of two states: operative or failed. The state of an arc is a random event that is statistically independent of the state of any other arc. J.Scott has proven that the functional reliability analysis of all-terminal problem is #P-complete in [5].

So far we have quickly reviewed minimum spanning tree problem on stochastic graphs, next we briefly survey problems under the semantic of uncertain. Researchers have studied many kinds of queries on uncertain database, such as Top-$k$ query [12], $k$-nearest neighbors querey [1], Probabilistic skylines [13]. In addition, lots of work have been done on uncertain graph, including discovering highly reliable subgraphs [14], discovering frequent subgraphs [4] and so on. However, to the best of our knowledge, there is no literature to date on discovering most reliable minimum spanning tree on uncertain graphs. This paper is the first one to investigate this problem.

## 6 Conclusion

This paper investigates the problem of the most reliable minimum spanning tree on uncertain graph data. The most reliable MST is an optimal choice between stability and cost, which has wide applications in practical. Since accurate algorithms take exponential time to enumerate all possible worlds, an approximate algorithm in polynomial time was proposed to discover an approximate MST and we analysis the accuracy and approximation rate of the approximate algorithm theoretically. The experimental results show that our greedy algorithm has high efficiency and approximation quality.

## References

1. Potamias, M., Bonchi, F., Gionis, A., et al.: K-nearest neighbors in uncertain graphs. In: VLDB (2010)
2. Prim, R.C.: Shortest connection networks and some generalizations. Bell Syst. Tech. J. **36**(6), 1389–1401 (1957)
3. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Am. Math. Soc. **7**(1), 48–50 (1956)
4. Zou, Z., Gao, H., Li, J.: Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In: SIGKDD (2010)
5. Provan, J.S., Ball, M.O.: The complexity of counting cuts and of computing the probability that a graph is connected. SIAM J. Comput. **12**(4), 777–788 (1983)
6. Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K., Toivonen, H.: Link discovery in graphs derived from biological databases. In: Leser, U., Naumann, F., Eckman, B. (eds.) DILS 2006. LNCS, vol. 4075, pp. 35–49. Springer, Heidelberg (2006). doi:10.1007/11799511_5

7. Kamousi, P., Suri, S.: Stochastic minimum spanning trees and related problems. In: ANALCO (2011)
8. Huang, L., Li, J.: Minimum spanning trees, perfect matchings and cycle covers over stochastic points in metric spaces. In: arXiv preprint arXiv (2012)
9. Frieze, A.M.: On the value of a random minimum spanning tree problem. Discret. Appl. Math. **10**(1), 47–56 (1985)
10. Steele, J.M.: On Frieze's (3) limit for lengths of minimal spanning trees. Discret. Appl. Math. **18**(1), 99–103 (1987)
11. Ball, M.O.: Computational complexity of network reliability analysis: an overview. IEEE Trans. Reliab. **35**(3), 230–239 (1986)
12. Soliman, M.A., Ilyas, I.F., Chang, K.C.-C.: Top-k query processing in uncertain databases. In: ICDE (2007)
13. Pei, J., Jiang, B., Lin, X., et al.: Probabilistic skylines on uncertain data. In: VLDB (2007)
14. Jin, R., Liu, L., Aggarwal, C.C.: Discovering highly reliable subgraphs in uncertain graphs. In: SIGKDD (2011)
15. Wu, Y., Fahmy, S., Shroff, N.B.: On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm. In: INFOCOM (2008)
16. Manfredi, V., Hancock, R., Kurose, J.: Robust routing in dynamic manets. In: Annual Conference of the International Technology Alliance (2008)