学校代码 10530　　学　　号 201230111747

分 类 号 TP311　　密　　级

湘潭大学

硕 士 学 位 论 文

不确定图中的生成树算法研究

学 位 申 请 人 唐 杰

指 导 老 师 文 中 华 教授

学 院 名 称 信息工程学院

学 科 专 业 计算机科学与技术

研 究 方 向 不 确 定 图

二〇一四年十一月三十日

不确定图中的生成树算法研究

学 位 申 请 人 唐 杰

导师姓名及职称 文 中 华 教授

学 院 名 称 信息工程学院

学 科 专 业 计算机科学与技术

研 究 方 向 不 确 定 图

学位申请级别 工 学 硕 士

学位授予单位 湘 潭 大 学

论文提交日期 2015-04-20

Research of Spanning Tree on Uncertain Graph

Candidate Jie Tang

Supervisor Professor Zhonghua Wen

College College of Information Engineering

Program Computer Science and Technology

Specialization Uncertain Graph

Degree Master of Engineering

University Xiangtan University

Date April 20th, 2015

湘潭大学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湘潭大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

涉密论文按学校规定处理。

作者签名：　　日期：　　年　　月　　日

导师签名：　　日期：　　年　　月　　日

摘　　要

图论作为数学的一个分支，在计算机科学中也扮演着非常重要的角色。许多问题都可以通过抽象化之后转化为图论问题，然后通过计算机将对应的图论算法实现来得到问题的解。随着大数据时代的到来，图数据的规模也随之增大，因此图算法的效率对于解决图论问题至关重要。

由于现实问题中可能会遇到很多的意外情况，如机器故障，信号传输的不稳定性等，最终导致实际数据存在不确定性。因此不确定图已经成为一个非常热门的研究领域。本文对不确定图中的生成树算法进行了研究，主要内容分为以下四个部分：

(1) 由于不确定图的边存在不确定性，所以传统的最小生成树定义已经不能完全适用于不确定图。因此提出了最优生成树的概念，它主要是在最小生成树定义的基础上增加了对概率的最优化定义。然后我们借助贪心的思想设计了不确定图中最优生成树求解算法。

(2) 为了能够找到不确定图中前K个最优的最小生成树，我们设计了Top-K最小生成树算法，并提出了基于并查集的优化算法和基于启发式搜索的优化算法。当K值较小时，基于启发式搜索的优化算法具有明显的优势。之后，我们又对最小生成树在不确定图中的可靠性进行了分析。

(3) 针对不确定有向图，给出了最优树形图的定义，并以刘朱算法为基础，设计了最优树形图算法。最后将TOP-K最小生成树算法的思想和最优树形图的性质相结合设计了TOP-K最优树形图查询算法。

最后，我们对不确定图中最优生成树的研究进行了相关的实验验证，实验结果表明，设计的算法的运行效率与理论分析基本一致，且能够完全正确的得到问题的解。

关键词：不确定图，蕴含图，最小生成树，最优生成树，最优树形图。

Abstract

Key words: uncertain graph, implicated graph, minimum spanning tree, optimized spanning tree, optimized arborescence.

目 录

第一章　绪 论

§1.1　研究背景和意义

1736年，瑞士数学家欧拉针对哥尼斯堡七桥问题发表了图论的首篇论文，奠定了图论的基础。此后，图论经过漫长的发展，直到1936年哥尼格发表了第一本图论专著，从此图论成为一门独立的学科。时至今日，图论已经渗透到了许多领域，它在自然科学和社会科学中都有广泛的应用。

在生物信息学中，蛋白质交互网络[1, 2]用来记录蛋白质之间的交互作用。我们可以用无向图来表示蛋白质交互网络，其中顶点用来表示蛋白质，边用来表示蛋白质之间的交互; 在无线传感器网络中[3, 4]，传感器和他们之间的通信可以用图来表示; 在社交网络中(例如Facebook、Twitter), 图可以用来描述人与人之间的社会关系。这些由不同领域产生的图结构也可以成为“图数据”，由于他们之间具有很多的共性，因此研究图数据本身的性质具有重要的意义。

由于数据获取技术的随机错误与测量误差、数据传输的故障与延时、多源集成数据的不完整性和不一致性等多种原因，大量的图数据具有不确定性。例如，生物信息学中，由于蛋白质交互的高通量生物检测技术存在固有误差，因此实验测得的蛋白质交互是否真实存在是不确定的[5–8]；在道路网络中，由于全球定位系统的数据传输故障和延时，当前网络中的交通流量数据无法确切放映实际情况，因而具有不确定性。在无线传感器网络中，由于物理世界的干扰、无线传感器网络的不稳定性、传感器节点的易失效性等原因，两个传感器之间是否可以正常通信是不确定的[9, 10]。我们称这些具有不确定因素的图数据为“不确定图数据”或“不确定图”。

在无线传感器网络中，一般都需要在监测区域布置大量的传感器，而每个传感器的能量是有限的，使用最小的能耗保证整个传感器网络的畅通显然具有重要的意义[11]。在理想情况下，我们可以将传感器网络用一个带权无向图来表示，权值代表两个传感器之间单位时间内通信所需的能耗，这样，可以使用经典的最小生成树算法(Prim、Kruskal)来获得一颗总能耗最小的网络拓扑结构树。然而，由于两个传感器之间的通信可能会出现故障，导致某些传感器之间的通信链路断开，这时我们需要做到能够快速的更新网络拓扑结构来保证网络的畅通。因此对不确定图中最小生成树的研究具有重要意义。

由于信息技术的飞跃发展，图数据的规模也越来越大，不确定因素对图数据的影响也越来越得到重视。国内外许多学者已经开始对不确定图数据展开了一些列研

究，也取得了很多的成果。在国内，邹等人已经出版了不确定图数据挖掘一书，国际上也出版了Managing and Mining Graph Data [12]和Mining Graph Data [13]等书籍。在一些国际顶级会议上(如SIGKDD、ICDE等)，也专门设立了不确定图数据的相关议程。

综上所述，对不确定图数据的研究具有重要的科学意义和广泛的应用价值。无论是与其它学科相互结合还是将传统的图论问题应用到不确定图模型上，都有很多待解决的问题。

§1.2 研究现状

本节主要对不确定图在国内外已有的研究进行大致分类并进行简要的分析。

(1) 频繁子图模式挖掘

频繁子图模式挖掘也是确定图数据挖掘中的经典问题 [14–17]，目的是从一组确定图中发现频繁出现的子图模式。而在不确定图数据中，现有的研究主要集中在期望频繁子图模式挖掘 [18–20]、概率频繁子图模式挖掘 [21, 22]和稠密子图挖掘 [23, 24]。前两者的主要目的是求解支持度和置信度满足一个指定阈值的频繁子图模式，在不确定图中的稠密子图挖掘具有代表性的有Top-K极大团挖掘和紧密顶点子集挖掘。

(2)分类和聚类

我们知道，分类和聚类算法在数学、统计学、计算机科学和生物学等领域都有很多应用。在不确定图数据中，对分类算法的研究主要是K近邻查询算法 [25–27]。例如，文献 [25]提出了基于SimRank [28]度量的方法来求解给定节点的k个最相似的节点，文献 [27]也对不确定图中的KNN进行了研究，它提出了三种方法(median-distance, majority-distance, expected-reliable-distance)来度量两个节点之间的距离。然而对聚类算法的研究主要有层次聚类算法 [29, 30]。文献 [29]首先将求出的强连通子图作为聚类中心，然后对余下的顶点进行层次聚类，文献 [30]主要是基于LinLog能量模型及其Newman-Girvan的模块化聚类，提出了一种多层次模块化聚类算法框架。

(3)基于最短路、最大流等图论模型

在图论中，Dijkstra、Prim和Edmonds-Karp算法都是用来解决图论中经典问题(最短路、最小生成树和最大流)的方法，然而这些方法并不能直接应用在不确定图数据中。因而，一些学者对不确定图中的最短路径、最大流等问题进行了研究 [31–34]，文献 [31]首先定义了概率语义下的最短路径问题，然后设计了求大于给定阈值的所有最短路径的算法；文献 [33]给出了最大流可靠性的概率计算模型，提出了一种基于简单路径组合思想的算法和基于状态空间划分的最可靠最大流改进算法。

§1.3 本文的结构安排

第一章　简要介绍了不确定图的研究背景、国内外研究现状等相关知识。

第二章　定义了不确定图中的最优生成树和次优生成树等概念，并设计了相应的求解算法。

第三章　针对Top-K最小生成树算法进行了深入分析，且根据K值的不同提出了两种优化算法。

第四章　对不确定图中最小生成树的可靠性进行研究。

第五章　对全文进行了总结，并对未来的相关工作进行探讨。

第二章　不确定图中最优生成树和次优生成树算法

§2.1　问题定义

定义 2.1.1 (不确定图) 不确定图是一个四元组$\mathcal{G} = (V, E, W, P)$，其中$V$是顶点集，$E$是边集，$W = \{w(e) \,|\, e \in E, w(e) \in \mathbb{N}^+\}$是边的权重集，$P = \{p(e) \,|\, e \in E, p(e) \in (0, 1]\}$是边存在可能性的集合。

定义 2.1.2 (蕴含图) 令不确定图$\mathcal{G} = (V, E, W, P)$，若确定图$G = (V_G, E_G, W_G)$是$\mathcal{G}$的一个蕴含图，则必然满足$V_G = V$，$E_G \subseteq E$和$W_G = \{w(e)|e \in E_G\} \subseteq W$。

从定义2.1.1可知，不确定图$\mathcal{G} = (V, E, W, P)$每条边以$p(e)$的概率存在，在可能世界模型下，每条边有存在和不存在两种可能性，所以可以派生出$2^{|E|}$个蕴含图。本文沿用文献[18, 27, 35]对不确定图模型所做的假设，即不确定图中不同边的概率分布相互独立。将蕴含图G和不确定图$\mathcal{G}$之间的关系表示为$\mathcal{G} \Rightarrow G$。基于以上假设，蕴含图$G$存在的概率为

$$\Pr(\mathcal{G} \Rightarrow G) = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)). \tag{2.1}$$

记不确定图$\mathcal{G}$的所有蕴含图的集合为$Imp(\mathcal{G})$。由文献[20]可知$\mathcal{G}$中所有蕴含图出现的概率和为1，即

$$\sum_{G \in Imp(\mathcal{G})} Pr(\mathcal{G} \Rightarrow G) = 1. \tag{2.2}$$

例 1 图2.1(a)为一个不确定图$\mathcal{G}_1$，边上的两个数字分别代表权值和概率，由于该不确定图有五条可能出现的边，因此该不确定图有2^5个蕴含图。图2.1(b)为不确定图$\mathcal{G}_1$的一个蕴含图，显然该蕴含图存在的概率为$p(e_2) \times p(e_3) \times p(e_4) \times p(e_5) \times (1 - p(e_1))) = 0.03528$。

在传统图论中，图的最小生成树被定义为边的权值和最小的生成树，一个图可能存在多个最小生成树，这些最小生成树具有相同的权值和。然而在不确定图中，每条边还有一个存在的概率，这样会导致每一颗最小生成树只有一定的概率存在，我们可以通过一个稳定性来区分最小生成树的好坏，即

$$R_T = \prod_{e \in E_T} p(e) \tag{2.3}$$

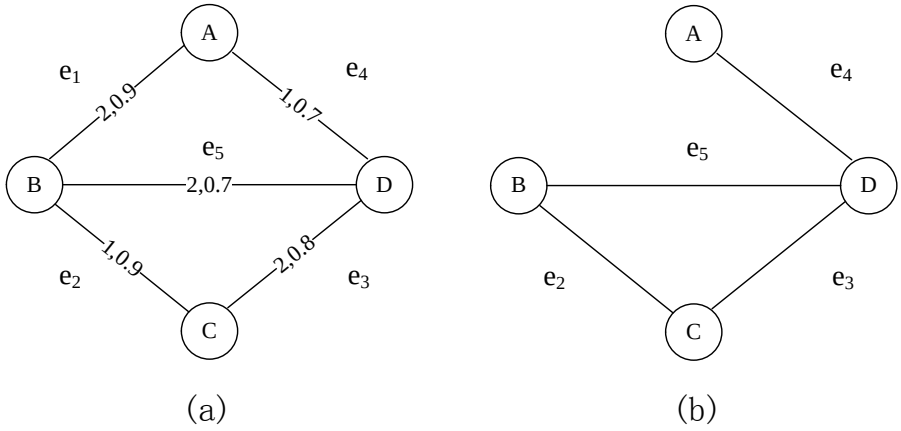


图 2.1 例 1：(a) 不确定图$\mathcal{G}$; (b) 图$\mathcal{G}$的一个蕴含图。

公式2.3的含义与公式2.1有所不同，公式2.3表示最小生成树的存在概率，也可以解释为所有包含最小生成树T的蕴含图存在的概率和。当这个值越大时，说明它的稳定性越高，即越不容易遭到破坏。

为了区分每颗生成树，我们需要给每颗生成树进行编号，假设所有生成树的边都按照编号从小到大进行排序，我们称字典序较小的生成树具有更小的编号。图2.2为图2.1(a)的所有最小生成树，它们的字典序分别为$e_1e_2e_4$, $e_2e_3e_4$和$e_2e_4e_5$，因此图2.2(a)所示的最小生成树的编号最小，图2.2(c)所示的最小生成树的编号最大，

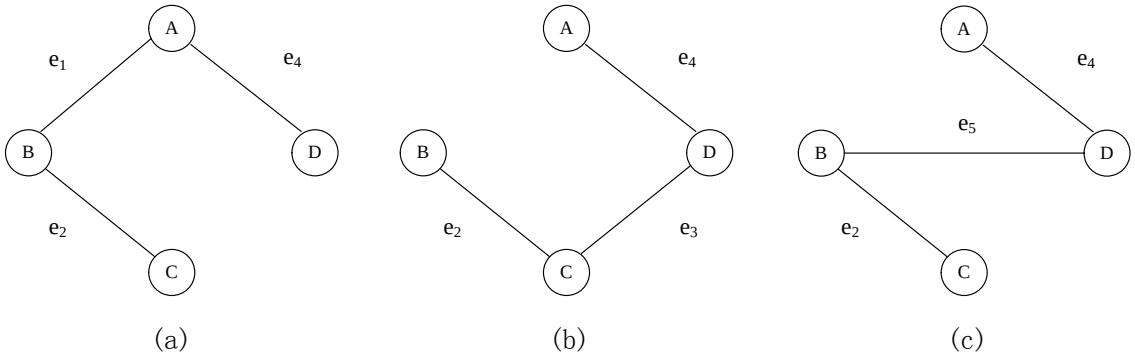


图 2.2 图2.1(a)的最小生成树。

定义 2.1.3 (最小生成树) 设不确定图$\mathcal{G} = (V, E, W, P)$，若生成树$T$满足$\forall T'$ $(\sum_{e\in E_T} W(e) \leq \sum_{e'\in E_{T'}} W(e'))$，则称$T$为不确定图$\mathcal{G}$的最小生成树，记编号最小的最小生成树为$T_M^{\mathcal{G}}$，其边集为$E_M^{\mathcal{G}}$，边的权值和为$W_M^{\mathcal{G}}$。

定义 2.1.4 (最小最大乘积生成树) 设不确定图$\mathcal{G} = (V, E, W, P)$，若生成树$T$满足$\forall T'(\prod_{e\in E_T} P(e) \leq \prod_{e'\in E_{T'}} P(e'))$，则称$T$为不确定图$\mathcal{G}$的最小乘积生成树，记编

号最小的最小乘积生成树为$T_{PL}^{\mathcal{G}}$，其边集为$E_{PL}^{\mathcal{G}}$，边的权值和为$W_{PL}^{\mathcal{G}}$。若生成树T满足$\forall T'(\prod_{e\in E_T} P(e) \geq \prod_{e'\in E_{T'}} P(e'))$，则称$T$为不确定图$\mathcal{G}$的最大乘积生成树，记编号最小的最大乘积生成树为$T_{PB}^{\mathcal{G}}$，其边集为$E_{PB}^{\mathcal{G}}$，边的权值和为$W_{PB}^{\mathcal{G}}$

定义 2.1.5 (最优生成树) 设$Imp(\mathcal{T})$为不确定图$\mathcal{G}$中所有最小生成树的集合，若最小生成树T满足$\forall T' \in Imp(\mathcal{T})(R_{T'} \leq R_T)$，则称$T$为不确定图$\mathcal{G}$的最优生成树。记编号最小的最优生成树为$T_O^{\mathcal{G}}$，其边集为$E_O^{\mathcal{G}}$，边的权值和为$W_O^{\mathcal{G}}$。

显然，最优生成树也不是唯一的，我们也可以将最优生成树看作不确定图中的最小生成树中的最大乘积生成树。

§2.2 最优生成树算法

定理 2.2.1 设不确定图$\mathcal{G} = (V, E, W, P)$和$\mathcal{G}' = (V, E, W, P')$，若两个不确定图满足$\forall e \in E(P'(e) = \log_2(P(e)))$，则$W_{PL}^{\mathcal{G}} = 2^{W_M^{\mathcal{G}'}}$。

证 明 设T为$\mathcal{G}$和$\mathcal{G}'$的任意一颗生成树，其边集为E_T，由定理的假设可知

$$\sum_{e\in E_T} P'(e) = \log_2 \prod_{e\in E_T} P(e)$$

显然生成树T在$\mathcal{G}'$中的边权和与在$\mathcal{G}$中边权的乘积成正比，当T为$\mathcal{G}'$的最小生成树时，则有$W_M^{\mathcal{G}'} = \log_2 W_{PL}^{\mathcal{G}}$，即$W_{PL}^{\mathcal{G}} = 2^{W_M^{\mathcal{G}'}}$。 ∎

由定理2.2.1可知，要想求得最小乘积生成树，可以将图中所有边的权值进行log变换，然后求新图的最小生成树。我们将在后面使用该思路去求解不确定图的最优生成树。

推论 2.2.1 若生成树的边权和是最小的，那么其边权的乘积也是最小的；若生成树的边权和是最大的，那么其边权的乘积也是最大的。

我们知道，使用$kruskal$算法求解最小生成树的第一步是需要对图中所有的边进行排序，因此可以假设不确定图$\mathcal{G} = (V, E, W, P)$中所有的边已经按照如下优先级进行排序：

- 将权值较小的边排在前面；
- 对于权值相同的边，则将概率较大的边排在前面；

- 若前两者的值都相同，则将编号较小的边排在前面。

我们称排在较前面的边具有更高的优先级。接下来，我们给出最优生成树算法的步骤：

(1) 新建一个图G，其顶点集和不确定图$\mathcal{G}$相同，边集为空；

(2) 依次处理排序好的边，若这条边连接的两个顶点不在同一个连通分量中，则将这条边加入图G;

(3) 重复第3步，直到图G中所有的顶点都连通。

这样，我们得到的图G既是不确定图$\mathcal{G}$的最优生成树, 显然该最优生成树也是编号最小的最优生成树$T_O^{\mathcal{G}}$。

定理 2.2.2 *最优生成树算法是正确的。*

证 明 显然最优生成树的算法步聚和$Kruskal$算法的步聚一致，因此我们得到的生成树一定是最小生成树。根据最小生成树的性质，所有的最小生成树应该具有相同的边权值序列（假设边权值按照从小到大排序），例如，图2.2中的三颗最小生成树的边权值序列都是$(1,1,2)$。假设最小生成树的边权值中有x个不同的值，我们用序列$(t_1,t_2,\cdots,t_x)$来表示最小生成树的边序列，其中集合t_i中的所有边具有相同边权值，例如图2.2(a)中的最小生成树可以表示为$(\{e_2,e_4\},\{e_1\})$，即$t_1=\{e_2,e_4\},t_2=\{e_1\}$。很容易知道，对于所有的最小生成树，其边序列$(t_1,t_2,\cdots,t_y)(y<=x)$构成的图具有相同数目的连通分量，且相对应的连通分量中的顶点集相同。根据最优生成树算法可知，由相同权值的边构成的连通分量，其边的概率和是最大的。再由推论2.2.1可知，这些边的概率乘积也是最大的，因此最优生成树算法是正确的。 ∎

例 2 *图2.3为一个不确定图$\mathcal{G}$，按照优先级排序后的边集为$\{e_1,e_2,\cdots,e_{10}\}$。图2.4为图2.3中不确定图的最优生成树求解过程，粗线标记的边为加入生成树中的边，在图(2)到图(3)时，由于顶点D和顶点E已经在同一个连通分量，所以边e_3没有加入生成树的边集中，图(6)中粗线构成的生成树则为最优生成树。从图中可知，$W_O^{\mathcal{G}}=13$且$R_{T_O^{\mathcal{G}}}=0.14112$。*

算法 1为最优生成树算法的伪代码，我们使用了并查集数据结构来快速合并两个连通分量，每个连通分量都会有一个唯一的根节点，通过函数GetRoot即能快速查询到每个顶点的根节点。合并连通分量只需要一条赋值语句即可，函数OST中的第7行则是合并顶点u和顶点v所在的连通分量。

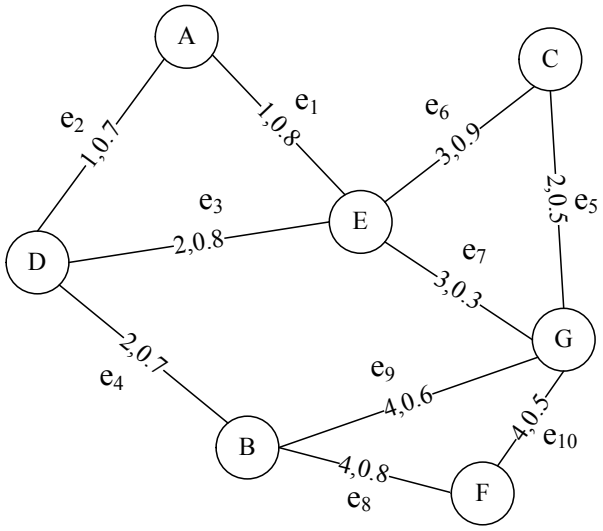


图 2.3 例 2：不确定图$\mathcal{G}$。

Algorithm 1 最优生成树算法

模块1： 并查集

initialization $RT_i \leftarrow i(i = 1, 2, \cdots, |V|)$

procedure GETROOT(u)
  if $u = RT_u$ **then return** u
  else return $(RT_u \leftarrow \text{GETROOT}(RT_u))$

模块2： 最优生成树算法

procedure OST($\mathcal{G}$) ▷ $\mathcal{G} = (V, E, W, P)$
  $E_O^{\mathcal{G}} = \emptyset$
  for each $e_i \leftarrow (u, v) \in E$ **do**
    $ru \leftarrow \text{GETROOT}(u)$
    $rv \leftarrow \text{GETROOT}(v)$
    if $ru \neq rv$ **then**
      $RT_{ru} = rv$
      $E_O^{\mathcal{G}} = E_O^{\mathcal{G}} \cup \{e_i\}$
  return $E_O^{\mathcal{G}}$

定理 2.2.3 最优生成树算法的时间复杂度为$O(mlogm)$，其中m为不确定图中边的数目。

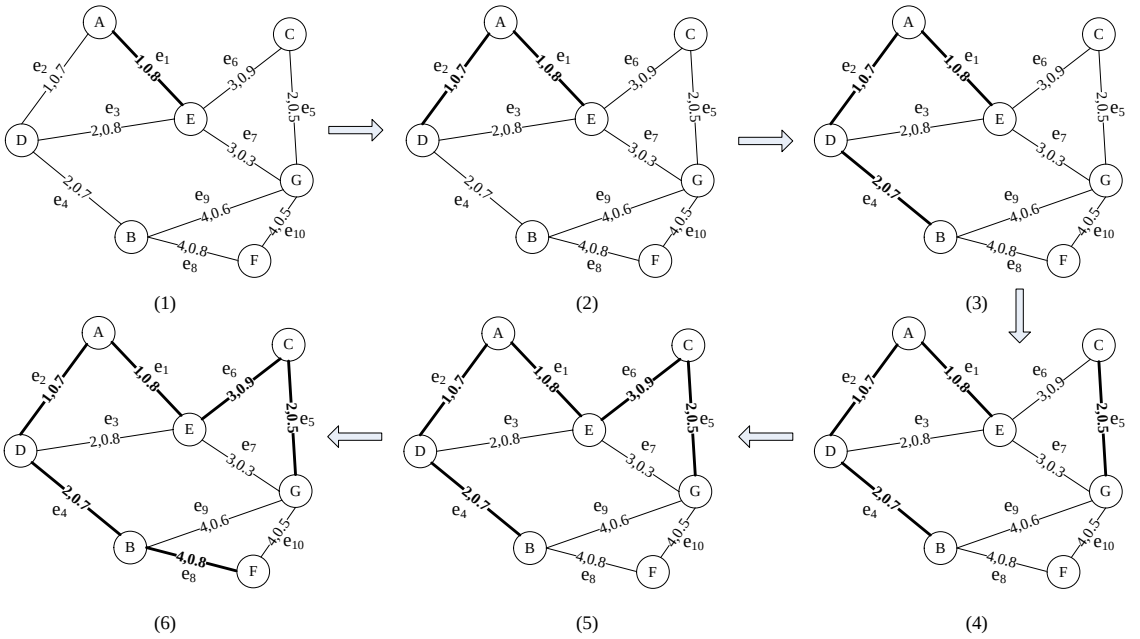


图 2.4 例 2：最优生成树计算过程。

证 明 首先我们可以使用推排序或者归并排序对不确定图中所有的边进行排序，时间复杂度为$O(mlogm)$，由于并查集均摊的时间复杂度为$O(1)$，所以算法1中第4-8行处理一条边的时间复杂度为$O(1)$，那么函数OST的时间复杂度则为$O(m)$，因此算法的总时间复杂度为$O(mlogm)$。 ∎

§2.3 次优生成树算法

定义 2.3.1 (次优生成树) 设$Imp(\mathcal{T})$为不确定图$\mathcal{G}$中所有最小生成树的集合，若最小生成树T满足$\forall T' \in (Imp(\mathcal{T}) \setminus \{T_O^{\mathcal{G}}\})(R_{T'} \leq R_T)$，则称$T$为不确定图$\mathcal{G}$的次优生成树。记编号最小的次优生成树为$T_{SO}^{\mathcal{G}}$，其边集为$E_{SO}^{\mathcal{G}}$，边的权值和为$W_{SO}^{\mathcal{G}}$。

由定义2.3.1可知，当最优生成树不唯一时，次优生成树也是最优生成树。显然，我们可以借用求次小生成树的思想求解次优生成树，步聚如下：

(1) 选择一条不在生成树$T_O^{\mathcal{G}}$中的边加入生成树$T_O^{\mathcal{G}}$；

(2) 假设新加入的边为$e = (u, v)$，寻找生成树$T_O^{\mathcal{G}}$中从顶点u到顶点v的路径中优先级最低的边并删除。这样可以得到一颗新的生成树；

(3) 重复前面2步，直到所有的边都已经处理，这样我们能够得到$|E| - |V| + 1$颗新的生成树，从这些生成树中求出最优的生成树则是不确定图$\mathcal{G}$的次优生成树。

在第二步中，我们需要知道最优生成树中任意两个顶点之间优先级最低的边，我们可以在求解最优生成树后进行预处理得到。假设$F(i,j)$为在最优生成树中从顶点i到顶点j的路径中优先级最低的边，则求$F(i,j)$的步骤如下：

1. 对最优生成树中的每个顶点进行深度优先搜索；
2. 在搜索的过程中记录这条路径中优先级最低的边，假设搜索树的根节点为i，当前节点为j，则我们用记录的优先级最低的边替换$F(i,j)$。

算法2为次优生成树算法，函数DFSGetF中的参数分别代表生成树的根节点、当前遍历节点的父节点、当前节点和最优生成树中的边集。我们以最优生成树中的每个节点为根节点进行深度优先搜索求得最优生成树中任意两个顶点之间的路径中优先级最低的边。

Algorithm 2 次优生成树算法

模块1： 求根节点到其它节点的路径中优先级最低的边

procedure DFSGetF($root, u, v, e, E_O^{\mathcal{G}}$)
  for each $v' \in V \wedge (v',u) \in E_O^{\mathcal{G}} \wedge v' \neq u$ **do** ▷ $e' = (v',u)$
    if $w(e') > w(e) \vee (w(e') = w(e) \wedge p(e') < p(e'))$ **then**
      $F(root, v') = e'$
      DFSGetF($root, v, v', e', E_O^{\mathcal{G}}$)
    else
      $F(root, v') = e$
      DFSGetF($root, v, v', e, E_O^{\mathcal{G}}$)

模块2： 次优生成树算法

procedure SOST($\mathcal{G}, E_O^{\mathcal{G}}$) ▷ $\mathcal{G} = (V, E, W, P)$
  for each $v \in V$ **do**
    DFSGetF($v, -1, v, null, E_O^{\mathcal{G}}$)
  $SOSet = \emptyset$
  for each $e' \in E$ **do** ▷ $e' = (u', v')$
    $SOSet = SOSet \cup (T_O^{\mathcal{G}} + F(u', v') \setminus e)$
  return 集合$SOSet$中最优的生成树

定理 2.3.1 次优生成树算法的时间复杂度为$O(n^2)$，其中n为不确定图中顶点的数目。

证 明 函数DFSGetF遍历整个最优生成树的时间复杂度为$O(n)$，而函数DFSGetF总共会被调用n次，所以函数SOST中第2-3的时间复杂度为$O(n^2)$。而第5-6行的时间复杂度为$O(m)$，其中m为不确定图中的边数，因此总的时间复杂度为$O(n^2+m) \approx O(n^2)$。 ∎

§2.4 本章小结

首先给出了最小生成树和最大乘积生成树的定义，然后通过前两者定义的结合给出了不确定图中的最优生成树和次优生成树的定义，即最优生成树可以看作是满足最小生成树条件的最大乘积生成树。然后证明了最大乘积生成树算法是可以和最小生成树算法进行相互转换的，最后结合最优生成树的性质提出了最优生成树算法和次优生成树算法。

第三章　不确定图中Top-K最小生成树算法

§3.1　问题描述

在不确定图中，可能存在许多不同的最小生成树，由于边集的不同，他们的边概率乘积也会不同。在这一章中，我们设计算法目的是求得前K个边概率乘积最大的最小生成树，显然该算法也能获得所有的最优生成树。

§3.2　基本算法

定理 3.2.1 设$E_T = \{e_1, e_2, \cdots, e_{n-1}\}$为不确定图$\mathcal{G} = (V, E, W, P)$的任意一颗最小生成树的边集，其中$n = |V|$，图的最小生成树可以通过$Prim$或$Kruskal$算法求得，令$A_i = \{e_1, e_2, \cdots, e_i\}$，其中$i \in \{1, 2, \cdots, n-1\}$。我们将蕴含图作以下划分：

$$D_i = \begin{cases} \{G | G \in Imp(\mathcal{G}) \wedge e_1 \notin E\}, & i = 0 \\ \{G | G \in Imp(\mathcal{G}) \wedge A_i \subseteq E \wedge e_{i+1} \notin E\}, & 0 < i < n-1 \\ \{G | G \in Imp(\mathcal{G}) \wedge A_i \subseteq E\}, & i = n-1 \end{cases} \tag{3.1}$$

则等式$\bigcup_{i=0}^{n-1} D_i = Imp(\mathcal{G})$成立，且$D_i \cap D_j = \varnothing\ (i < j)$。设$A_0 = \varnothing$且不确定图不存在边$e_n$，则公式3.1可以统一表示为第二种情况。

证 明 可以分两部分进行证明：

(1) $\bigcup_{i=0}^{n-1} D_i = Imp(\mathcal{G})$。设不确定图的所有蕴含图的集合$Imp(\mathcal{G}) = \{G_1, G_2, \cdots, G_r\}$，其中$r = |Imp(\mathcal{G})|$。若$E_{G_i} \cap E_T = \{e_{x_1}, e_{x_2}, \cdots, e_{x_y}\}$，其中$x_j < x_{j+1} (j = 1, 2, \cdots, y-1)$，则必然存在一个$k$满足：

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

可以将蕴含图G_i进行如下归类：

$$\begin{cases} G_i \in D_0, & x_1 \neq 1 \\ G_i \in D_k, & x_1 = 1 \end{cases}$$

因此可以得到$Imp(\mathcal{G}) \subseteq \bigcup_{i=0}^{n-1} D_i$，由$D_i$的定义可知$D_i \subseteq Imp(\mathcal{G}) (i = 0, 1, \cdots, n-1)$，由此可知$\bigcup_{i=0}^{n-1} D_i \subseteq Imp(\mathcal{G})$，因此$\bigcup_{i=0}^{n-1} D_i = Imp(\mathcal{G})$。

(2) $D_i \cap D_j = \emptyset (i < j)$。由公式 3.1可知，$D_i \cap D_j = \{G \,|\, G \in Imp(\mathcal{G}) \wedge A_j \subseteq E \wedge e_{i+1} \notin E\}$，由于$e_{i+1} \in A_j$，所以$A_j \subseteq E$和$e_{i+1} \notin E$不可能同时成立，因此$D_i \cap D_j = \emptyset$。

因此定理 3.2.1成立。 ∎

引理 3.2.1 设$E_T = \{e_1, e_2, \cdots, e_{n-1}\}$为不确定图$\mathcal{G} = (V, E, W, P)$的任意一颗最小生成树的边集，令边集$E_I = \{e_1, e_2, \cdots, e_k\}$为$E_T$的一个子集；边集$E_O \subseteq E$且$E_O \cap E_T = \emptyset$；若集合$S = \{G \,|\, G \in Imp(\mathcal{G}) \wedge E_I \subseteq E_G \wedge E_O \cap E_G = \emptyset\}$，即集合$S$为包含边集$E_I$中的所有边且不包含边集$E_O$中的所有边的蕴含图集合，则我们可以将集合$S$进行如下划分：

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

其中$E_{I_i} = E_I \cup \{e_{k+1}, e_{k+2}, \cdots, e_i\}$，$E_{O_i} = E_O \cup \{e_{i+1}\}$，显然$\bigcup_{i=k}^{n-1} D_i = S$成立。

由引理 3.2.1可知，我们可以通过边集划分的方法来获取所有的最小生成树，然后将最小生成树的边乘积按照从大到小的顺序排列，选取前K个最小生成树，即可得到Top-K最小生成树。算法 3为TOP-K最小生成树算法，函数GetMST$(E_I, E_O, \mathcal{G})$将返回一颗不确定图$\mathcal{G}$中的最小生成树，且该最小生成树的边集一定包含E_I且不包含E_O。函数FindAllMST中的第6行如果成立则表示E_I已经是一颗最小生成树的边集，所以不用继续划分，直接加入集合$MSTSet$即可。

例 3 图 3.1(a)为一个不确定图，图 3.1(b)为不确定图中主蕴含图的其中一颗最小生成树，其中边的权值和为4，图 3.2为函数调用FindAllMST$(\emptyset, \emptyset, \mathcal{G})$的全过程，我们使用了树形结构来表示，节点中左右两边表示传递给被调用函数的参数，左边表示可以包含的边的集合，右边表示不可以包含的边的集合，带有粗边框的节点表示到此为止已经找到了一颗最小生成树，这颗生成树的边集为节点左侧的边集，粗边框右上角的数字表示这个生成树中所有边的概率乘积。显然，最优生成树和次优生成树的边集分别为$\{e_2, e_3, e_5\}$和$\{e_2, e_3, e_6\}$。

定理 3.2.2 算法 3的时间复杂度为$O(Nnm)$，其中N为不确定图$\mathcal{G}$中最小生成树的个数，n和m分别代表不确定图$\mathcal{G}$中的顶点数和边数。

证 明 显然，每次调用函数FindAllMST时，需要调用一次函数GetMST并且产生$n-1$次的递归调用。这样，函数FindAllMST最多被调用$Nn + 1 \approx Nn$次。然而，函数GetMST求解最小生成树的时间复杂度为$O(m)$。因此，算法 3的时间复杂度为$O(Nnm)$。 ∎

Algorithm 3 TOP-K最小生成树算法

模块1： TOP-K最小生成树算法

procedure TopKMST($K, \mathcal{G}$) ▷ $\mathcal{G} = (V, E, W, P)$
    $MSTSet =$ FindAllMST($\emptyset, \emptyset, \mathcal{G}$)
    将集合$MSTSet$中所有最小生成树按照边乘积的值从大到小排序
    return 集合$MSTSet$中前K个最小生成树

模块2： 查找所有最小生成树

procedure FindAllMST($E_I, E_O, \mathcal{G}$)
    $E_T \leftarrow$ GetMST($E_I, E_O, \mathcal{G}$)
    $k \leftarrow |E_I|$
    $MSTSet \leftarrow \emptyset$
    if $E_T = E_I$ **then**
        $MSTSet = MSTSet \cup \{T\}$
    else
        for $i \leftarrow k$ **to** $|V| - 1$ **do**
            $E_{I_i} \leftarrow E_I \cup \{e_{k+1}, e_{k+2}, \cdots, e_i\}$
            $E_{O_i} \leftarrow E_O \cup \{e_{i+1}\}$
            $MSTSet \leftarrow MSTSet \cup$ FindAllMST($E_{I_i}, E_{O_i}, \mathcal{G}$)
    return $MSTSet$

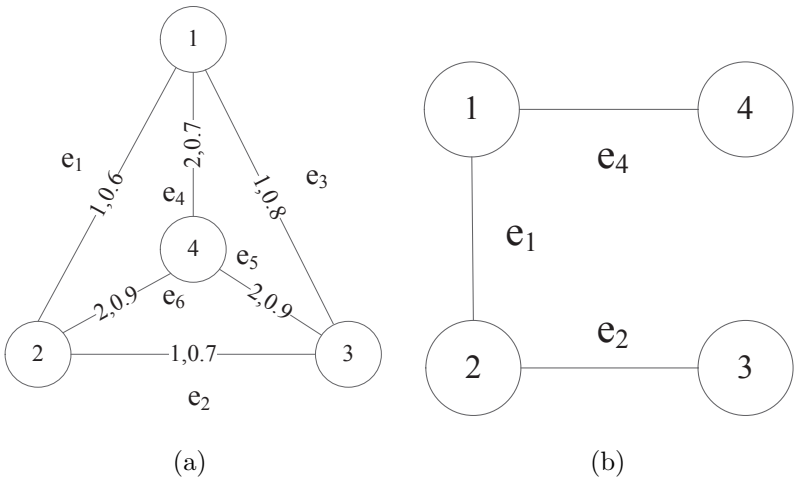


图 3.1 例 3：(a) 不确定图$\mathcal{G}$; (b) 图$\mathcal{G}$中的一颗最小生成树

§3.3 并查集优化

在算法3中，我们每次都需要调用函数GetMST来获取一颗新的最小生成树，

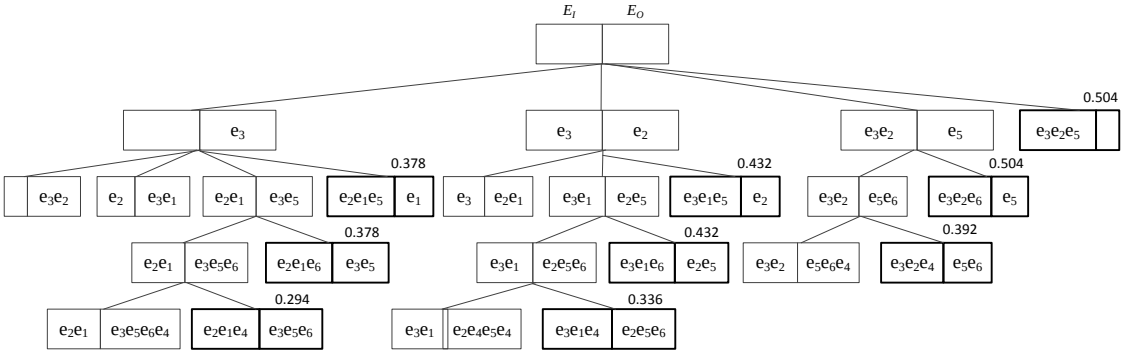


图 3.2 例 3：函数FindAllMST$(\emptyset, \emptyset, \mathcal{G})$的递归调用过程。

由于新的生成树和上层调用的生成树有很多相似之处，因此在这一小节中，我们将对该方法进行改进，我们希望通过一个边的替换操作，能够很快的获得一颗新的最小生成树，假设我们用$T_1 \xrightarrow[e']{e} T_2$表示一个边替换操作，该操作表示在生成树$T_1$中添加边$e$并删除$e'$后可以得到生成树$T_2$，即$T_2 = T_1 \cup \{e'\} \setminus \{e\}$，我们称边$e'$为为边$e$的替换边。

定理 3.3.1 *在不确定图$\mathcal{G}$中，假设T_a为一颗生成树，T_b为一颗最小生成树，若集合$E_{Diff} = \{e_1, e_2, \cdots, e_k\}$中的元素表示这两颗生成树中不同的边，即满足$e_i \in E_{T_a}$并且$e_i \notin E_{T_b}$。则我们可以通过重复执行下面的边替换操作$k$次，使得$T_a$变为$T_b$，即*

$$T_a(T_0) \xrightarrow[e'_1]{e_1} T_1 \xrightarrow[e'_2]{e_2} T_2, \cdots, \rightarrow T_{k-1} \xrightarrow[e'_k]{e_k} T_b(T_k),$$

且

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

证 明 我们可以重复执行下面的操作k次：

(1) 假设边e_i连接的两个顶点分别为u和v，若从生成树T_{i-1}中移除边e_i，则可以得到两个连通分量，我们将其设为V_1和V_2；

(2) 若在生成树T_b中从顶点u到顶点v的路径为$u(u_0) \rightarrow u_1, \cdots, u_{x-1}, \rightarrow v(u_x)$(我们也可以用$P_v^u$来表示这个路径)，那么必然存在$i(i < x)$使得$u_i \in V_1$且$u_{i+1} \in V_2$。令$e'_i = (u_i, u_{i+1})$，由于$T_b$是最小生成树，显然$w(e'_i) \leq w(e_i)$，因此我们可以通过边替换$T_{i-1} \xrightarrow[e'_i]{e'} T_i$得到生成树$T_i$且$\sum_{e \in T_i} w(e) = \sum_{e \in T_{i-1}} w(e) + w(e'_i) - w(e_i) \leq \sum_{e \in T_{i-1}} w(e)$。

∎

定理 3.3.2 设T为不确定图$\mathcal{G}$中的最小生成树，假设边$e \in E_T$连接的两个连通分量为V_1和V_2。那么对于任意不包含边e的最小生成树T'，必然存在一条边$e' = (u', v')$满足$w(e') = w(e)$，其中$u' \in V_1$，$v' \in V_2$。

证 明 假设边e连接的两个顶点分别为u和v，在生成树T'中的路径P_v^u为$u(u_0) \to u_1, \cdots, u_{x-1}, \to v(u_x)$。那么必然存在一个$i(i < x)$满足$u_i \in V_1$且$u_{i+1} \in V_2$，设$e' = (u_i, u_{i+1})$，则有下列两种情况：

(1) 如果$w(e') > w(e)$, 可以设$E_{T^*} = E_{T'} \cup \{e\} \backslash \{e'\}$，那么$\sum_{e \in E_{T^*}} w(e) < \sum_{e \in E_{T'}} w(e)$；

(2) 如果$w(e') < w(e)$，可以设$E_{T^*} = E_T \cup \{e'\} \backslash \{e\}$，那么$\sum_{e \in E_{T^*}} w(e) < \sum_{e \in E_T} w(e)$。

显然，以上两种情况与T和T'是最小生成树冲突，因此，$w(e_i) = w(e)$。 ∎

设$E(V_1, V_2, \mathcal{G}) = \{e \in E \,|\, e \in V_1 \times V_2\}$。根据定理 3.3.2可知，如果存在边$e'$满足$e' \in E(V_1, V_2, \mathcal{G})$且$w(e') = w(e)$，那么我们可以通过边替换操作$T \xrightarrow[e']{e} newT$得到一颗新的最小生成树，即$E_{newT} = T \cup \{e'\} \setminus \{e\}$，否则该不确定图不存在其它包含边$e$的最小生成树。

为了发现在生成树T中每条边的交换边，我们可以按照权值从小到大遍历不确定图$\mathcal{G} = (V, E, W, P)$中的所有边，对于每条边$e^* \in E$，我们将执行下面两个步骤：

(1) 假设边e^*连接的两个顶点分别为u^*和v^*，那么我们可以遍历路径$P_{v^*}^{u^*}$。显然这条路径上所有权值和边e^*权值相等的边都可以被边e^*替换，即边e^*为他们的替换边；

(2) 处理完路径$P_{v^*}^{u^*}$上所有边后，则可以使用并查集来压缩路径$P_{v^*}^{u^*}$。

为了查找最小生成树T中任意两个顶点的路径，我们需要知道任意两个顶点之间的关系。首先，我们选择一个顶点作为树的根节点，然后通过深度优先搜索对每个顶点进行编号，这样就可以通过编号区间来判定两个顶点之间的关系，我们定义函数GetChild为：

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

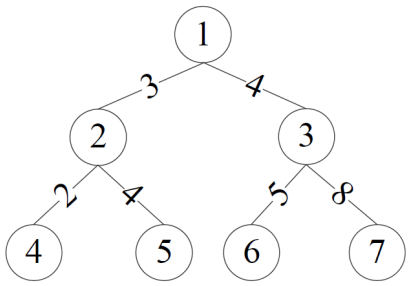
算法 4是对子树中的节点进行编号，显然，从根节点开始进行前序遍历的顺序即为每个节点编号的顺序。我们可以调用函数SetNumber$(T, 1, 1)$为生成树T中的所有顶点进行编号。

Algorithm 4 子树编号算法

```
1: procedure SetNumber(T, w, num)
2:     L_w ← num, R_w ← num
3:     C ← GetChild(T, w)
4:     for each v ∈ C do
5:         R_w ← SetNumber(T, v, R_w + 1)
6:     return R_w
```

定义 3.3.1 (最近公共祖先) 假设T为不确定图$\mathcal{G}$中的最小生成树，编号为1的顶点为根节点，则顶点u和顶点v的最近公共祖先为生成树T中的顶点，该顶点即是顶点u的祖先也是顶点v的祖先，且该顶点的编号最小。

例 4 图 3.3是通过函数调用SetNumber$(T, 1, 1)$对顶点编号后的最小生成树T。根据算法 4可知，L_i是通过前序遍历得到的结果，而$R_i = \max\{L_j \mid$ 顶点j 为顶点i的子孙节点$\}$，即R_i为以顶点i为根节点的子树中L_i的最大值。



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

图 3.3 编号后的最小生成树。

由图3.3可知，如果$L_j \in [L_i, R_i]$，则顶点j是顶点i的子孙节点，设u_1, u_2为生成树T的两个顶点，则我们可以通过以下步骤发现他们之间的路径：

(1) 从顶点u_1开始向父节点进行遍历，直到找到一个顶点v满足$L_{u_2} \in [L_v, R_v]$；

(2) 从顶点u_2开始向父节点进行遍历，直到顶点v停止。

这样，路径$P_{u_2}^{u_1}$可以表示为$u_1 \to \cdots \to v \to \cdots \to u_2$，且顶点$v$为顶点$u_1$和$u_2$的最近公共祖先。

接下来，我们使用并查集来压缩路径$P_{u_2}^{u_1}$。设RT_i为顶点i被缩点后所在的连通分量中编号最小的顶点，显然对于每个在路径$P_{u_2}^{u_1}$中的顶点w，满足$RT_w = v$。我们

设函数GetFather(T,x)的功能为返回顶点x的父节点。算法5为替换边求解算法，在函数FindSwapEdge中，第4-8行为求顶点u和v的最近公共祖先，伪代码中使用了lca来表示。第10-12行中，我们扫描从顶点u到顶点v的路径中的每一条边，如果他的权值和替换边的权值相同，则说明找到了替换边。最后在第13-14行更新这个路径中每个顶点的RT值。

Algorithm 5 查找生成树T中每条边的替换边。

Module 1: 并查集

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** $(RT_u \leftarrow \text{GetRoot}(RT_u))$

Module 2: 查找替换边

1: **procedure** FindSwapEdge(G, T)
2: $S = \emptyset$
3: **for each** $e \leftarrow (u, v) \in E_G$ **do**
4: $lca \leftarrow ru \leftarrow \text{GetRoot}(u)$
5: $rv \leftarrow \text{GetRoot}(v)$
6: **while** $rv \notin [L_{lca}, R_{lca}]$ **do**
7: $f \leftarrow \text{GetFather}(T, lca)$
8: $lca \leftarrow \text{GetRoot}(f)$
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

例 5 如图3.4所示，设$e_1 = (2,5), e_2 = (3,6), e_3 = (4,5), e_4 = (5,6), w(e_3) = 4, w(e_4) = 5$ 且$e_3, e_4 \in V$。由于$w(e_3) < w(e_4)$，所以我们首先查找顶点4和顶点5之间的路径，由于$w(e_1) = w(e_3)$，所以边e_3为边e_1的替换边。显然顶点2为顶点4和顶

点5的最近公共祖先，我们用顶点2来替换路径P_5^4，则$RT_4 = 2$且$RT_5 = 2$。之后我们再处理边e_4，我们知道顶点5已经被其最近公共祖先替换，这样$e_4 = (5,6) = (2,6)$，又因为$w(e_2) = w(e_4)$，所以边e_4为边e_2的替换边，因此集合$S = \{(e_1, e_3), (e_2, e_4)\}$。

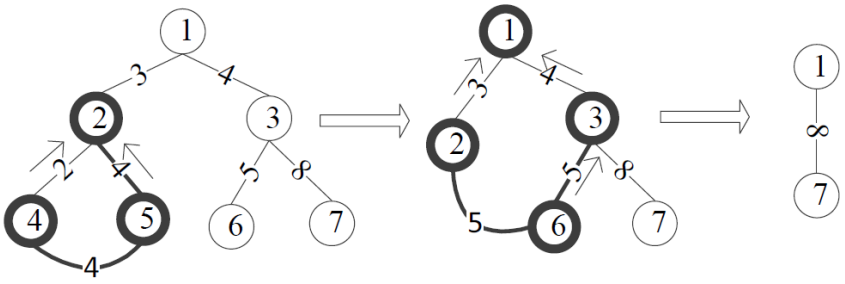


图 3.4 路径压缩过程。

算法6为优化后的算法，首先调用函数SetNumber对当前的最小生成树进行编号，然后调用函数FindSwapEdge求当前的最小生成树中所有边的替换边，最后，通过边集划分的方法，将边集E_I和E_O进一步细化，通过得到的替换边获得新的最小生成树。

定理 3.3.3 算法6的时间复杂度为$O(Nm)$，其中N为不确定图$\mathcal{G}$中最小生成树的个数，m为不确定图$\mathcal{G}$中的边数。

证 明 首先，函数SetNumber只需要对生成树搜索一遍，所以函数SetNumber的时间复杂度为$O(n)$；其次，函数FindSwapEdge需要遍历不确定图$\mathcal{G}$中的每一条边，由于使用了路径压缩，生成树中的每条边则最多只会被访问一次。所以可以得到函数FindSwapEdge的时间复杂度为$O(m)$(假设$m > n$)；这样，使用边替换操作查找一颗新的最小生成树的平均时间复杂度为$O(m/n)$，因此算法6的时间复杂度为$O(Nn * (m/n)) = O(Nm)$。 ∎

§3.4 启发式搜索A*优化

由算法3和算法6可知，当参数K很小时，该算法也需要获取所有的最小生成树，时间复杂度显然很高，因此我们将使用启发式搜索来进行优化。

定理 3.4.1 设T为不确定图$\mathcal{G} = (V, E, W, P)$中的一颗最小生成树且$e \in E_T$，边$e$划分的两个连通分量分别为$V_1$和$V_2$，设$e' \neq e$为集合$E(V_1, V_2, \mathcal{G})$中边权值第二大的边，则边集$E_{T'} = E_T \cup \{e'\} \setminus \{e\}$对应的生成树$T'$为不确定图$\mathcal{G}' = (V, E \setminus \{e\}, W, P)$中的最小生成树。

Algorithm 6 TOP-K最小生成树算法-并查集优化。

模块1： TOP-K最小生成树算法-并查集优化

procedure TopKMST($K, \mathcal{G}$) ▷ $\mathcal{G} = (V, E, W, P)$
  $E_T \leftarrow \text{OST}(\mathcal{G})$
  $MSTSet = \text{FindAllMST}(\emptyset, \emptyset, E_T, \mathcal{G})$
  将集合$MSTSet$中所有最小生成树按照边乘积的值从大到小排序
  return 集合$MSTSet$中前K个最小生成树

模块2： 查找所有最小生成树

procedure FindAllMST($E_I, E_O, E_T, \mathcal{G}$)
  $MSTSet \leftarrow \{T\}$
  $G \leftarrow (V, E \setminus E_O, W)$
  SetNumber($T, 1, 1$)
  $S \leftarrow \text{FindSwapEdge}(G, T)$
  for $i \leftarrow k$ **to** $|V| - 2$ **do**
    $E_{I_i} \leftarrow E_I \cup \{e_{k+1}, e_{k+2}, \cdots, e_i\}$
    $E_{O_i} \leftarrow E_O \cup \{e_{i+1}\}$
    if $\exists e'((e_{i+1}, e') \in S)$ **then**
      $E_{T'} \leftarrow E_T \cup \{e'\} \setminus \{e_{i+1}\}$
      $MSTSet \leftarrow MSTSet \cup \text{FindAllMST}(E_{I_i}, E_{O_i}, E_{T'}, \mathcal{G})$
  return $MSTSet$

证 明 我们使用反证法来证明，假设T'不是不确定图$\mathcal{G}'$中的最小生成树。那么，必然存在两条边$c \notin T'$和$d \in T'$满足：(1)存在生成树T''，其边集$E_{T''} = E_{T'} \cup \{c\} \setminus \{d\}$；(2)$w(c) < w(d)$。显然$c \in E(V_1, V_2, \mathcal{G})$，否则边集$E_{\widetilde{T}} = E_T \cup \{c\} \setminus \{d\}$对应的生成树$\widetilde{T}$的边权值和比最小生成树$T$更小，这样与$T$是最小生成树矛盾，因此我们可以得到

$$w(e) \leq w(e') \leq w(c) < w(d)$$

设边集$E_T \cup \{e'\}, E_T \cup \{c\}, E_{T'} \cup \{c\}$组成的环路分别为$C_0, C_1, C_2$。显然$C_2 = C_0 \otimes C_1$且$d \in C_2$，这样，$d \in C_0$和$d \in C_1$至少有一个成立，我们定义：

$$E_{\widehat{T}} = \begin{cases} E_T \cup \{e'\} \setminus \{d\}, & d \in C_0 \\ E_T \cup \{c\} \setminus \{d\}, & d \in C_1 \end{cases} \tag{3.2}$$

那么，生成树$\widehat{T}$的权值和比最小生成树T更小，产生矛盾，因此原定理成立。 ∎

显然，定理3.4.1也同样适用于最小(最大)乘积生成树。由于启发式搜索一般都需要使用一个估价函数来评估当前状态被处理的优先级，由算法6可以知道，函数FindAllMST每次最多产生$n-2$次递归调用，而每次的边替换操作都会选择一个权值相等且概率最大的边进行替换，由定理3.4.1可知，递归调用后求得的最小生成树的边概率乘积必然比当前层的最小生成树T的边概率乘积小，因此，我们只要使用边替换操作后求的的最小生成树的边概率乘积作为估价值即可。算法7为使用启发式搜索的TOP-K最小生成树算法，其中$elem.T$是以$elem.E_T$为边集的最小生成树。$PriorQue$按照传入的第一个参数进行从大到小排序，可以使用堆排序来实现$PriorQue$的插入和删除操作，每次我们都从$PriorQue$中选择边概率乘积最大的最小生成树进行扩展，这样，当从$PriorQue$中出队的个数达到K个时算法终止。

Algorithm 7 TOP-K最小生成树算法-启发式搜索优化。

1: **procedure** TopKMST($K, \mathcal{G}$)
2: $TopKMST \leftarrow \emptyset$
3: $E_T \leftarrow \mathrm{OST}(\mathcal{G})$
4: $PriorQue \leftarrow \emptyset$
5: $PriorQue.push([\prod_{e \in E_T} p(e), \emptyset, \emptyset, E_T])$ ▷ $elem : [priority, E_I, E_O, E_T]$
6: **while** $PriorQue \neq \emptyset$ **do**
7: $elem = PriorQue.pop()$
8: $TopKMST = TopKMST \cup \{elem.E_I\}$
9: **if** $|TopKMST| \geq K$ **then**
10: **return** $TopKMST$
11: $G \leftarrow (V, E \setminus elem.E_O, W)$
12: SetNumber($elem.T, 1, 1$)
13: $S \leftarrow$ FindSwapEdge($G, elem.T$)
14: **for** $i \leftarrow k$ **to** $|V| - 2$ **do**
15: $E_{I_i} \leftarrow elem.E_I \cup \{e_{k+1}, e_{k+2}, \cdots, e_i\}$
16: $E_{O_i} \leftarrow elem.E_O \cup \{e_{i+1}\}$
17: **if** $\exists e'((e_{i+1}, e') \in S)$ **then**
18: $E_{T'} \leftarrow elem.E_T \cup \{e'\} \setminus \{e_{i+1}\}$
19: $PriorQue.push([\prod_{e \in E_{T'}} P(e), E_{I_i}, E_{O_i}, E_{T'}])$
20: **return** $TopKMST$

定理 3.4.2 算法7的时间复杂度为$O(KmlogN)$，其中K为需要查找的$Top\text{-}K$最小生成树，N为不确定图$\mathcal{G}$中最小生成树的个数，m为不确定图$\mathcal{G}$中的边数。

证 明 显然$PriorQue$中元素最多为N个，如果$PriorQue$使用最大堆实现，则插入和删除的平均时间复杂度为$O(logN)$，再由定理3.3.3可知，获得K个最小生成树的时间为$O(Km)$, 因此算法7的时间复杂度为$O(KmlogN)$。 ∎

由定理3.4.2可知，当K较小时，$PriorQue$中的元素也会较少，这样，算法7的时间复杂度将接近$O(m)$，所以算法7非常适合K值较小的情况。

例 6 图3.5为函数TopKMST$(9,\mathcal{G})$的调用过程，其中参数$\mathcal{G}$为图3.1(a)中的不确定图。在图3.5中，边上的数字代表优先队列的扩展顺序，每次都会选择一个边概率乘积最大的最小生成树进行扩展。从图中可以看出，任何一个节点对应的最小生成树的边概率乘积都要比其子节点或子孙节点对应的最小生成树的边概率乘积要大，这与我们之前的分析完全一致。

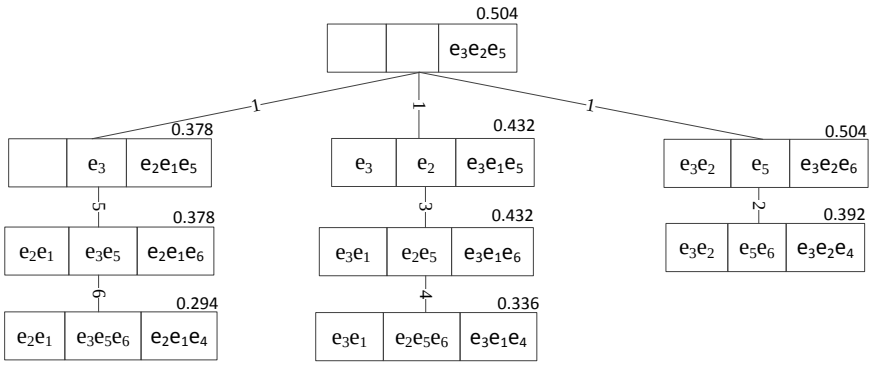


图 3.5 例 6：函数TopKMST$(9,\mathcal{G})$的调用过程。

§3.5 实验结果

图3.6为算法6和算法7的运行时间对比，该不确定图的顶点数为34，边数为196，最小生成树的个数为575624。由图3.6可知，当K值小于270000时，算法7优于算法6。由于最小生成树的个数最多为575624，所以当K大于600000时，算法7的运行时间不再增长。

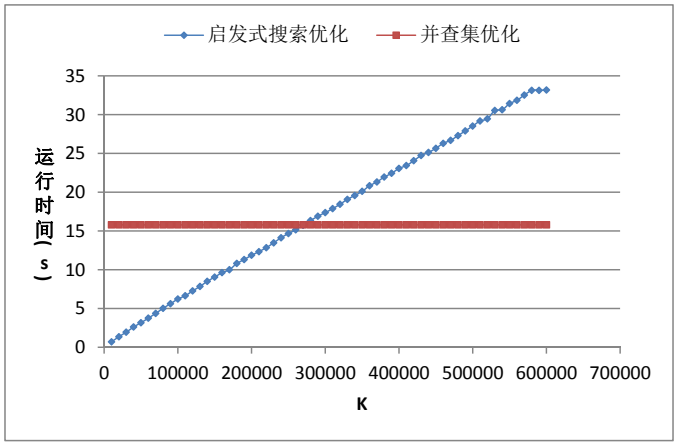


图 3.6 并查集优化和启发式搜索优化的实验对比。

§3.6 本章小结

将不确定图中的子图按照最小生成树的边集进行划分，这样保证每一个子图集合都唯一对应一颗最小生成树，有效的避免了同一个最小生成树被重复计算的情况。通过挖掘最小生成树的性质，使用边替换的方式可以快速的获得每个子图集合对应的最小生成树，并且提出了基于并查集优化和启发式搜索优化的算法，这两种算法分别适用于K值较大和较小的情况。最后的实验结果也与理论分析相符。

第四章　不确定图中最小生成树的可靠性研究

§4.1　问题定义

定义 4.1.1 (可靠蕴含图) 设G为不确定图$\mathcal{G}=(V,E,W,P)$的任意蕴含图，T_M^G为图G的最小生成树，若满足$\sum_{e\in E_{T_M^G}} w(e)=W_M^{\mathcal{G}}$，则称$G$为不确定图$\mathcal{G}$的可靠蕴含图。

定义 4.1.2 (不确定图中最小生成树的可靠性) 设不确定图$\mathcal{G}=(V,E,W,P)$，令$S=\{G|G\in Imp(\mathcal{G})\wedge W_M^G=W_M^{\mathcal{G}}\}$为不确定图$\mathcal{G}$中所有可靠蕴含图的集合，则我们定义不确定图$\mathcal{G}$中最小生成树的可靠性为：

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

例 7 表 *4.1*的第一列显示了图 **??**中的不确定图的蕴含图边集，该不确定图一共有三颗最小生成树(其边集分别为$\{e_1,e_2,e_4\},\{e_2,e_3,e_4\}$和$\{e_2,e_4,e_5\}$)，因此一共有7个可靠蕴含图(表 *4.1*的前7行对应的蕴含图)，由于非可靠蕴含图出现的概率和为0.37378，因此该不确定图的最小生成树的可靠性为$1-0.37378=0.62622$。

§4.2　最小生成树的可靠性求解算法

定理 4.2.1 设不确定图$\mathcal{G}=(V,E,W,P)$且不确定图$\mathcal{G}'=(V,E',W,P)$，其中$E'=E\setminus\{e\}$，则$\sum_{G\in Imp(\mathcal{G})} Pr(\mathcal{G}\Rightarrow G)=\sum_{G\in Imp(\mathcal{G}')} Pr(\mathcal{G}'\Rightarrow G)$.

证 明 由公式 2.1可知

$$\begin{aligned}\sum_{G\in Imp(\mathcal{G})\wedge e\in E} Pr(\mathcal{G}\Rightarrow G)&=\sum_{G\in Imp(\mathcal{G})\wedge e\in E} p(e)Pr(\mathcal{G}'\Rightarrow G\setminus e)\\&=p(e)\sum_{G\in Imp(\mathcal{G}')} Pr(\mathcal{G}'\Rightarrow G)\end{aligned}$$

同理

$$\sum_{G\in Imp(\mathcal{G})\wedge e\notin E} Pr(\mathcal{G}\Rightarrow G)=(1-p(e))\sum_{G\in Imp(\mathcal{G}')} Pr(\mathcal{G}'\Rightarrow G)$$

表 4.1 图 ??的可靠性评估表.

蕴含图边集	包含的最小生成树	出现概率
$\{e_1, e_2, e_3, e_4, e_5\}$	$\{e_1, e_2, e_4\}$, $\{e_2, e_3, e_4\}$, $\{e_2, e_4, e_5\}$	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

因此

$$\sum_{G\in Imp(\mathcal{G})} Pr(\mathcal{G}\Rightarrow G) = \sum_{G\in Imp(\mathcal{G})\wedge e\in E} Pr(\mathcal{G}\Rightarrow G) + \sum_{G\in Imp(\mathcal{G})\wedge e\notin E} Pr(\mathcal{G}\Rightarrow G)$$
$$= \sum_{G\in Imp(\mathcal{G}')} Pr(\mathcal{G}'\Rightarrow G)$$

∎

引理 4.2.1 设不确定图$\mathcal{G}=(V,E,W,P)$，则 $\sum_{G\in Imp(\mathcal{G})} Pr(\mathcal{G}\Rightarrow G)=1.$

证 明 设$E=\{e_1,e_2,\cdots,e_m\}$，且

$$\mathcal{G}_0=\mathcal{G}$$
$$\mathcal{G}_1=\mathcal{G}_0\setminus\{e_1\}$$
$$\mathcal{G}_2=\mathcal{G}_1\setminus\{e_2\}$$
$$\cdots\cdots$$
$$\mathcal{G}_m=\mathcal{G}_{m-1}\setminus\{e_m\}$$

则由定理 4.2.1可知

$$\sum_{G\in Imp(\mathcal{G})} Pr(\mathcal{G}\Rightarrow G) = \sum_{G\in Imp(\mathcal{G}_1)} Pr(\mathcal{G}_1\Rightarrow G) = \cdots = \sum_{G\in Imp(\mathcal{G}_m)} Pr(\mathcal{G}_m\Rightarrow G)$$

由于不确定图$\mathcal{G}_m$的边集为空，所以 $\sum_{G\in Imp(\mathcal{G}_m)} Pr(\mathcal{G}_m\Rightarrow G)=1$。 ∎

引理 4.2.2 设不确定图$\mathcal{G}=(V,E,W,P)$，且$E_I,E_O\in E$，则

$$\sum_{G\in Imp(\mathcal{G})\wedge E_I\subseteq E_G\wedge E_O\cap E_G=\emptyset} Pr(\mathcal{G}\Rightarrow G)=\prod_{e\in E_I}p(e)\prod_{e\in E_O}(1-p(e))$$

在引理 3.2.1中，我们知道可以通过边集划分的方法来获取最小生成树。同样的，这种划分也适用于不确定图中所有的蕴含图，也就是说，我们可以将图 3.2看成是不确定图中所有蕴含图集合的划分过程，节点的左边表示蕴含图必然包含的边的集合，即E_I，右边为蕴含图一定不包含的边的集合，即E_O。当这个节点不能继续划分时，如果边集E_I为不确定图中某颗最小生成树的边集，那么划分到这个节点中所有的蕴含图即为可靠蕴含图。因此，由引理 4.2.2可以得到改节点中所有可靠蕴含图的存在概率之和。算法 8为不确定图中最小生成树的可靠性求解算法。

Algorithm 8 不确定图中最小生成树的可靠性求解算法

Comment Calls from IMST$(\emptyset,\emptyset,OST(\mathcal{G}),\mathcal{G})$

procedure IMST$(E_I,E_O,E_T,\mathcal{G})$ ▷ $\mathcal{G}=(V,E,W,P)$
  $r\leftarrow\prod_{e\in E_I}p(e)\times\prod_{e\in E_O}(1-p(e))$
  $G\leftarrow(V,E\setminus E_O,W)$
  SetNumber$(T,1,1)$
  $S\leftarrow$ FindSwapEdge(G,T)
  for $i\leftarrow k$ **to** $|V|-2$ **do**
    $E_{I_i}\leftarrow E_I\cup\{e_{k+1},e_{k+2},\cdots,e_i\}$
    $E_{O_i}\leftarrow E_O\cup\{e_{i+1}\}$
    if $\exists e'((e_{i+1},e')\in S)$ **then**
      $E_{T'}\leftarrow E_T\cup\{e'\}\setminus\{e_{i+1}\}$
      $r\leftarrow r+\text{IMST}(E_{I_i},E_{O_i},E_{T'},\mathcal{G})$
  return r

§4.3 实验结果

实验 1 首先我们针对不确定图为完全图且所有边的权值都相等时的数据进行实验，其所有的生成树都为最小生成树。根据*Cayley*定理 [36]，最小生成树的个数为n^{n-1}，其中n为不确定图中顶点的个数。

表 4.2 不确定图为完全图时算法的执行效率对比

		运行时间(s)			
G	N	$FERM$	$IERM$	时间比	可靠性
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

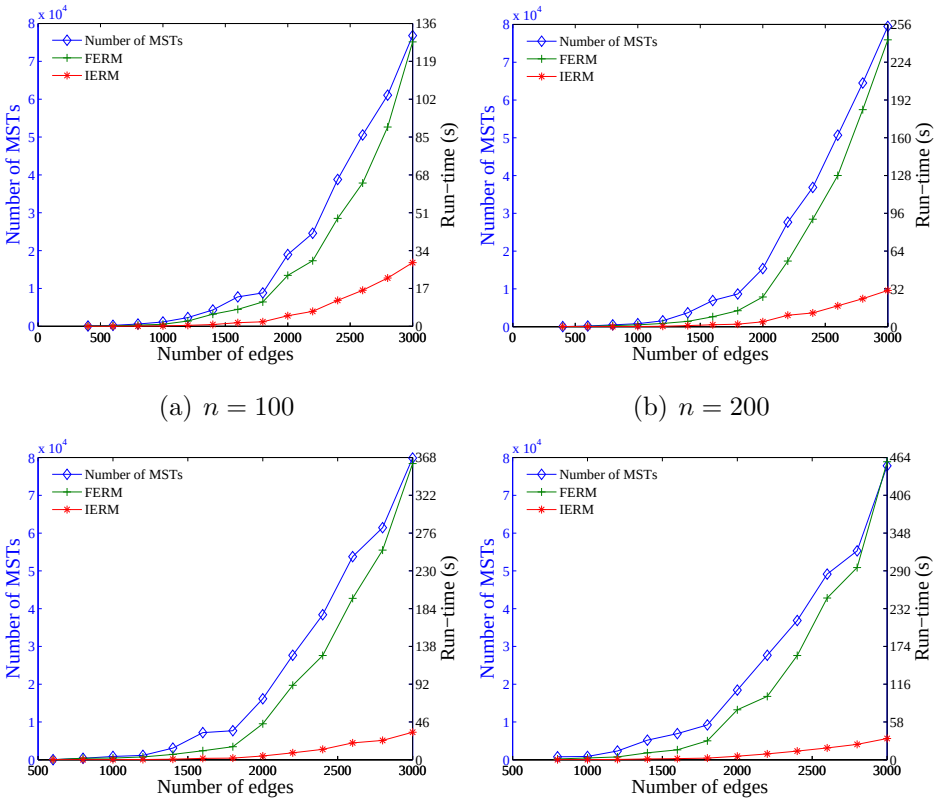
表 4.2为算法$FERM$和$IERM$的运行结果，其中算法$FERM$没有采用并查集优化。表中的N表示不确定图中最小生成树的数量，K_n表示该不确定图是一个顶点数为n的完全图，时间比即为算法$FERM$与算法$IERM$的执行时间之比，从表中可以看出，随着图规模的增大，时间比也越来越大，这是因为算法$IERM$的程序时间相对复杂，导致常数级处理较多，所以当数据规模较少时效率不容易体现出来。

实验 2 为了验证顶点数和边数对算法的影响，我们将边的权值控制在1到100之间。图 4.1为算法$FERM$和$IERM$在不同顶点数下的时间对比图。

从图 4.1可以得出算法$IERM$的时间复杂度与顶点数无关，这也验证了我们在第三章对并查集优化算法的理论分析。从图中还可以看出，这两个算法的运行时间都会随着边数的增多而增大。

§4.4 本章小结

本章提出了在不确定图中最小生成树的可靠性问题，并设计了基于边集划分的可靠性求解算法，为了避免该算法在每次搜索产生分支后都要重新求最小生成树，使用并查集的思想在原有的最小生成树上进行边的删除和替换来更快地获得新的最小生成树。实验结果验证了算法的高效性和理论分析的正确性。



(a) $n = 100$ (b) $n = 200$

(c) $n = 300$ (d) $n = 400$

图 4.1 顶点数变化时算法的执行效率对比

第五章　不确定图中最优树形图的扩展研究

§5.1　问题定义

定义 5.1.1 (不确定有向图) 不确定有向图是一个四元组$\vec{\mathcal{G}} = (V, \vec{E}, W, P)$，其中$V$是顶点集，$\vec{E}$是有向边集，$W = \{w(e) \,|\, e \in \vec{E}, w(e) \in \mathbb{N}^+\}$是边的权重集，$P = \{p(e) \,|\, e \in \vec{E}, p(e) \in (0, 1]\}$是边存在可能性的集合。

定义 5.1.2 (最优树形图) 令不确定有向图$\vec{\mathcal{G}} = (V, \vec{E}, W, P)$，设$root$为不确定有向图的根节点，若有向生成树$\vec{T}$满足以下条件：

(1) 顶点$root$能够通过有向生成树的边到达图中任意顶点；

(2) 对于任意满足条件*(1)*的有向生成树$\vec{T}'$，要么其边权和大于生成树$\vec{T}$的边权和，要么其边权和等于生成树$\vec{T}$的边权和且其边的概率乘积不大于生成树$\vec{T}$的边的概率乘积；

则称有向生成树$\vec{T}$为不确定有向图$\vec{\mathcal{G}}$的最优树形图。

在确定有向图中，最小树形图则是以$root$为根节点的最小生成树，且节点$root$能够通过树中的有向边到达图中任意节点。在1965年，朱永津和刘振宏提出的朱刘算法 [37]能够在$O(VE)$的时间复杂度内获得最小树形图；而在1986年，Gabow等人提出了更好的算法 [38]，能够在$O(E + VlogV)$的时间复杂度内求得该问题的解。

在不确定有向图中，由于受边概率的影响，不能完全套用已有的算法去求最优树形图，接下来我们借用朱刘算法的思想来求解最优树形图并提供理论证明。

§5.2　最优树形图算法

首先，我们给出最优树形图算法的流程：

(1) 找到除了$root$以外其它节点的权值最小的入边，如果有多条边符合条件，则选择边概率最大的入边。用inW_i记录这条边的权值, inP_i记录这条边的概率；

(2) 如果除了$root$节点之外，还存在其它的孤立节点，则不存在最优树形图；

(3) 找到图中所有的环，并对环进行缩点和重新编号；

(4) 更新其它点到环上的点的权值和概率。

显然，除了第(4)步需要对边的概率进行更新以外，其余的步骤与朱刘算法无异，假设节点u在该环上且该环被压缩后的新编号为$newV$，那么对于每一条进入节点u的边(v, u, w, p)，在新图中则建立边$(v, newV, w - inW_u, p/inP_i)$。

例 8 *在图 5.1中，最左边为一个不确定有向图，假定v_4为图中的$root$节点，根据最优树形图的求解步骤，首先需要获得每个顶点的入边，由图可知顶点$v_1, v_2, v_3, v_5, v_6, v_7, v_8, v_9$的入边分别为$e_2, e_1, e_5, e_3, e_{10}, e_8, e_9, e_{11}$，我们将环上的点重新编号为$U_1$，然而，$e_4, e_6, e_7$的权值和概率分别调整为$(4, 1), (4, 1), (4, 2)$，当权值相等时我们则选择概率最大的边，所以我们选择边e_7连接顶点v_4和U_1，由于新图中不存在环，算法结束，最后只要将选择的边进行展开则可得到图的最优树形图。*

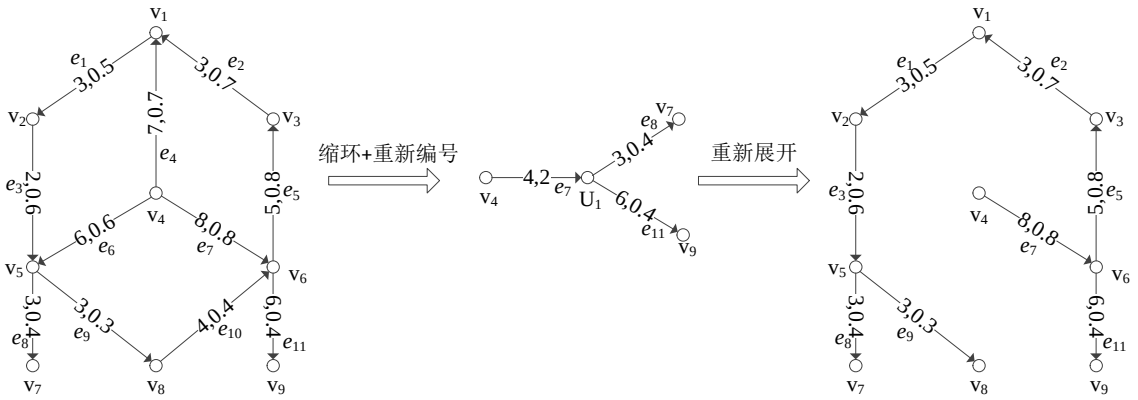


图 5.1 最优树形图求解过程。

定理 5.2.1 *最优树形图算法是正确的。*

证 明 首先，若对每个顶点选择入边后没有形成环路，那么所选的入边的集合显然就是最优树形图。若形成环路C，可以通过反证法证明只需要删除环中的一条边，然后用另外一条指向环的边替换即可。假设存在另一个更优的解S_1，环中至少有两条边不在S_1中，若$e_1 \in C$且$e_1 \notin S_1$，我们只需将e_1加入S_1且删除S_1中与e_1有相同终点的边则可以得到一个新的解S_2，显然S_2不会比S_1差，因此我们只需要替换掉环中的一条边即可，而具体由哪一条边进行替换，则需要对所有指向环的边的权值和概率进行更改，然后重复最优树形图算法直至不存在环路即可。 ∎

算法 9为最优树形图算法的伪代码，4-7行对应算法流程的第一步，即求所有点的入边；8-15行则是根据入边的集合构造新的不确定有向图，最后将原图中选择的边和新图中的边进行合并得到最优树形图的边集，第9行的函数BuildNewGraph()构造新图的流程如下：

(1) 对于边集inE构成的每个环路，环路上的所有顶点将由一个新的顶点替换，其余顶点的编号不变；

(2) 对于任意属于图中且不在环中的边$e=(v_1,v_2)$，若顶点v_2为环中的一个顶点，则边e的权值更新为$w(e)-w(inE_{v_2})$，概率更新为$p(e)/p(inE_{v_2})$。否则该边保持不变。

Algorithm 9 不确定图中最优树形图算法

```
1:  procedure OTG(G⃗, root)                                  ▷ G⃗ = (V, E⃗, W, P)
2:      inE ← ∅
3:      retE ← ∅
4:      for each v ∈ V⃗ ∧ v ≠ root do
5:          for each e = (v1, v2) ∈ E⃗ ∧ v2 = v do
6:              if w(e) < w(inE_v) ∨ (w(e) = w(inE_v) ∧ p(e) > p(inE_v)) then
7:                  inE_v ← e
8:      if inE构成的图存在环路 then
9:          newG⃗ = BuildNewGraph(G⃗, inE)
10:         retE = OTG(newG⃗, root)
11:         for each e = (v1, v2) ∈ inE ∧ e ∉ retE do
12:             if ∀e' = (v1', v2') ∈ retE(v2' ≠ v2) then
13:                 retE ← retE ∪ {e}
14:     else
15:         retE ← inE
16:     return retE
```

定理 5.2.2 算法9的时间复杂度为$O(nm)$，n和m分别为不确定有向图的顶点数和边数。

证 明 在获取每个顶点的入边时，需要遍历不确定有向图的每一条边，所以获取入边集合的时间复杂度为$O(m)$。而只有当入边集合构成的图存在环路时才会继续递

归，每一次缩环后，新图的顶点数至少减少一个，因此最坏情况下需要构造n次新图，因此总的时间复杂度为$O(mn)$。 ∎

我们之前都是假设最优树形图的根节点$root$已经被告知，若需要求根节点任意的最优树形图，则只要在图中新增加一个顶点和n条边，这n条边分别为新增的顶点指向其它n个顶点，且边的权值为原图中最大的权值加一，最后在得到的最优树形图中删除新增的顶点即可。

§5.3 TOP-K最优树形图算法

在第三章中，介绍了TOP-K最优生成树算法，我们主要使用了边集划分的思想来求一个新的最优生成树，显然，该方法对于不确定有向图也同样适用，最容易想到的TOP-K最优树形图算法则是在边集划分之后，每次都使用上一节介绍的最优树形图算法获得一颗最优树形图, 这样，算法的时间复杂度则为$O(Nn^2m)$，其中N为最优树形图的个数，显然这个时间复杂度有点高，我们并没有完全利用好最优树形图的性质。

在第三章的并查集优化中，我们主要利用了定理3.3.2中的性质，定理3.3.2说明，我们只需要对最优生成树进行一次边替换操作则可以得到一颗新的最优生成树，然而这个思想在不确定有向图中同样适用，而且应用在最优树形图中会更加简单。

定理 5.3.1 *设$\vec{T}$为不确定有向图$\vec{\mathcal{G}}$中的最小树形图，假设边$e=(u,v)\in\vec{E}_{\vec{T}}$连接的两个连通分量为$V_1$和$V_2$。那么对于任意不包含边$e$的最小树形图$\vec{T}'$，必然存在一条边$e'=(u',v')$满足$w(e')=w(e)$，其中$u'\in V_1$，$v'\in V_2$。*

证 明 我们可以通过反证法来证明该定理的正确性，假设存在一个最小树形图$\vec{T}*$，该最小树形图不存在边$e*=(u*,v*)$满足$w(e*)=w(e)$，其中$u*\in V_1$，$v*\in V_2$。那么该假设显然也适用于不确定图，这与定理3.3.2的正确性相悖，因此定理5.3.1是正确的。 ∎

当删除边$e=(u,v)$时，连通分量V_2则形成了一颗以顶点v为根节点的有向子树，且顶点v能够到达任意其它顶点，由于边的有向性，其它顶点则不能够到达顶点v，因此我们选择的替换边$e'=(u',v')$必然满足$v'=v$，即边的终点始终指向顶点v。

算法10为TOP-K最优树形图算法，该算法在求解替换边时，也只需要遍历不确定有向图，然后找到具有相同终点和权值的边即可，每次预处理的时间复杂度为$O(m)$，所以获得一个新的最小树形图的平均时间复杂度为$O(m/n)$，因此算法10的时间复杂度为$O(Nm)$，若使用优先队列优化，则时间复杂度为$O(KmlogN)$。

Algorithm 10 TOP-K最优树形图算法

模块1： TOP-K最优树形图算法

procedure TopKOTG($K, \vec{\mathcal{G}}$) ▷ $\vec{\mathcal{G}} = (V, \vec{E}, W, P)$
  $\vec{E}_{\vec{T}} \leftarrow \mathrm{OTG}(\vec{\mathcal{G}})$
  $MTGSet = \textsc{FindAllMTG}(\emptyset, \emptyset, \vec{E}_{\vec{T}}, \vec{\mathcal{G}})$
  将集合$MTGSet$中所有最小树形图按照边乘积的值从大到小排序
  return 集合$MTGSet$中前K个最小树形图

模块2： 查找所有最小树形图

procedure FindAllMTG($E_I, E_O, \vec{E}_{\vec{T}}, \vec{\mathcal{G}}$)
  $MTGSet \leftarrow \{\vec{T}\}$
  for $i \leftarrow k$ **to** $|V| - 2$ **do**
    $E_{I_i} \leftarrow E_I \cup \{e_{k+1}, e_{k+2}, \cdots, e_i\}$
    $E_{O_i} \leftarrow E_O \cup \{e_{i+1}\}$ ▷ $e_{i+1} = (u_{i+1}, v_{i+1})$
    边e_{i+1}划分的连通分量为V_1和V_2，其中$u_{i+1} \in V_1, v_{i+1} \in V_2$
    if $\exists e' = (u', v')(e' \neq e_{i+1} \wedge v' = v_{i+1} \wedge u' \in V_1)$ **then**
      $\vec{E}_{\vec{T}'} \leftarrow \vec{E}_{\vec{T}} \cup \{e'\} \setminus \{e_{i+1}\}$
      $MTGSet \leftarrow MTGSet \cup \textsc{FindAllMTG}(E_{I_i}, E_{O_i}, \vec{E}_{\vec{T}'}, \vec{\mathcal{G}})$
  return $MTGSet$

§5.4 本章小结

第六章　总结与展望

参考文献

[1] JENSEN L J, KUHN M, STARK M, et al. (2009) String 8-a global view on proteins and their functional interactions in 630 organisms. *Nucleic acids research*, **37**, 412–416.

[2] CEOL A, ARYAMONTRI A C, LICATA L, et al. (2009) Mint, the molecular interaction database: 2009 update. *Nucleic acids research*, **38**, 532–541.

[3] 李建中，李金宝，石胜飞. (2003) 传感器网络及其数据管理的概念、问题与进展. 软件学报, **14**, 1717–1727.

[4] SOHRABY K, MINOLI D, ZNATI T. (2007) *Wireless sensor networks: technology, protocols, and applications*. John Wiley & Sons.

[5] BADER J S, CHAUDHURI A, ROTHBERG J M, et al. (2003) Gaining confidence in high-throughput protein interaction networks. *Nature biotechnology*, **22**, 78–85.

[6] ASTHANA S, KING O D, GIBBONS F D, et al. (2004) Predicting protein complex membership using probabilistic network reliability. *Genome research*, **14**, 1170–1175.

[7] SUTHRAM S, SHLOMI T, RUPPIN E, et al. (2006) A direct comparison of protein interaction confidence assignment schemes. *BMC bioinformatics*, **7**, 360.

[8] JIANG R, TU Z, CHEN T, et al. (2006) Network motif identification in stochastic networks. *Proceedings of the National Academy of Sciences*, **103**, 9404–9409.

[9] GHOSH J, NGO H Q, YOON S, et al. (2007) On a routing problem within probabilistic graphs and its application to intermittently connected networks. pp. 1721–1729. IEEE.

[10] LIU Y, HE Y, LI M, et al. (2013) Does wireless sensor network scale? a measurement study on greenorbs. *Parallel and Distributed Systems, IEEE Transactions on*, **24**, 1983–1993.

[11] MURUGANATHAN S D, MA D C F, BHASIN R I, et al. (2005) A centralized energy-efficient routing protocol for wireless sensor networks. *Communications Magazine, IEEE*, **43**, 8–13.

[12] COOK D J, HOLDER L B. (2006) *Mining graph data*. John Wiley & Sons.

[13] AGGARWAL C C, WANG H. (2010) *Managing and mining graph data*. Springer.

[14] BORGELT C, BERTHOLD M R. (2002) Mining molecular fragments: Finding relevant substructures of molecules. pp. 51–58. IEEE.

[15] YAN X, HAN J. (2002) gspan: Graph-based substructure pattern mining. pp. 721–724. IEEE.

[16] KURAMOCHI M, KARYPIS G. (2001) Frequent subgraph discovery. pp. 313–320. IEEE.

[17] HUAN J, WANG W, PRINS J, et al. (2004) Spin: mining maximal frequent subgraphs from graph databases. pp. 581–586. ACM.

[18] ZOU Z, LI J, GAO H, et al. (2009) Frequent subgraph pattern mining on uncertain graph data. pp. 583–592. ACM.

[19] 邹兆年，李建中，高宏等. (2009) 从不确定图中挖掘频繁子图模式. 软件学报, **20**, 2965–2976.

[20] ZOU Z, LI J, GAO H, et al. (2010) Mining frequent subgraph patterns from uncertain graph data. *Knowledge and Data Engineering, IEEE Transactions on*, **22**, 1203–1218.

[21] ZOU Z, GAO H, LI J. (2010) Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. pp. 633–642. ACM.

[22] LI J, ZOU Z, GAO H. (2012) Mining frequent subgraphs over uncertain graph databases under probabilistic semantics. *The VLDB Journal*, **21**, 753–777.

[23] ZOU Z, LI J, GAO H, et al. (2010) Finding top-k maximal cliques in an uncertain graph. pp. 649–652. IEEE.

[24] 邹兆年，朱镕. (2013) 大规模不确定图上的top-k极大团挖掘算法. 计算机学报, **36**, 2146–2155.

[25] 张应龙，李翠平，陈红等. (2011) 不确定图上的knn 查询处理. 计算机研究与发展, **48**, 1850–1858.

[26] 张旭，何向南，金澈清等. (2011) 面向不确定图的k最近邻查询. 计算机研究与发展, **48**, 1871–1878.

[27] POTAMIAS M, BONCHI F, GIONIS A, et al. (2010) K-nearest neighbors in uncertain graphs. *Proceedings of the VLDB Endowment*, **3**, 997–1008.

[28] JEH G, WIDOM J. (2002) Simrank: a measure of structural-context similarity. pp. 538–543. ACM.

[29] 李俊辉. (2013) 基于不确定图的层次聚类算法研究. 中国管理信息化, **15**, 79–80.

[30] 丁悦. (2012) 不确定图聚类分析研究. Master's thesis. 西北农林科技大学.

[31] YUAN Y, CHEN L, WANG G. (2010) Efficiently answering probability threshold-based shortest path queries over uncertain graphs. pp. 155–170. Springer.

[32] 李鸣鹏，邹兆年，高宏等. (2012) 不确定图上期望最短距离的计算. 计算机研究与发展, **49**, 2208–2220.

[33] 蔡伟，张柏礼，吕建华等. (2012) 不确定图最可靠最大流算法研究. 计算机学报, **35**, 2371–2380.

[34] 张柏礼，吕建华，生衍等. (2014) 一种不确定图中最可靠最大流问题的解决方案. 计算机学报, **10**, 2084–2095.

[35] 张海杰, 姜守旭, 邹兆年. (2011) 不确定图上的高效top-k近邻查询处理算法. 计算机学报, **34**, 1885–1896.

[36] KELLY P J. (1957) A congruence theorem for trees. *Pacific J. Math*, **7**, 961–968.

[37] Chu, Yoeng-Jin and Liu, Tseng-Hong. (1965) On shortest arborescence of a directed graph. *Scientia Sinica*, **14**, 1396.

[38] Gabow, Harold N and Galil, Zvi and Spencer, Thomas and Tarjan, et al. (1986) Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, **6**, 109–122.

致 谢

本研究及学位论文是在我的导师文中华教授的亲切关怀和悉心指导下完成的。他严肃的科学态度，严谨的治学精神，精益求精的工作作风，深深地感染和激励着我。从课题的选择到论文的最终完成，文老师都始终给予我细心的指导和不懈的支持。三年来，文老师不仅在学业上给我以精心指导，同时还在思想、生活上给我以无微不至的关怀，在此谨向文老师致以诚挚的谢意和崇高的敬意。

感谢信息工程学院的各位老师，从本科入学到研究生毕业的七年里，无私地把知识传授给我们，并教我们求知、做人的真理。

感谢我的母校，给了我良好的学习环境。在这个美丽的校园里，我结识了许多的良师益友。

感谢在一起愉快的度过研究生生活的工科楼506各位同门，正是由于你们的热情和帮助，才使得我很快的融入这个研究环境。你们也是我在三年的科研道路上最坚实的伙伴!

感谢生我养我，含辛茹苦的父母，是你们为我的学习创造了条件，并在我的身后默默的支持着我。谢谢你们!

在论文即将完成之际，我的心情无法平静，从开始进入课题到论文的顺利完成，有多少可敬的师长、同学给了我无言的帮助，在这里请接受我诚挚的谢意!

附录A (攻读硕士学位期间发表的论文)

1. 唐杰，文中华，汪泉等. 不确定可逆规划的强循环规划解. 计算机研究与发展，2013，50(9): 1970-1980.

2. 唐杰，文中华，黄海平等. 不确定规划中一种观察信息高效约简算法. 计算机工程，2013，39(12): 162-166.

3. Jie Tang, Yuansheng Liu, Zhonghua Wen. Reliability evaluation of the minimum spanning tree of uncertain graphs. Journal of Computers, 2015, 10(1): 24-35.

4. Yuansheng Liu, Jie Tang, Tao Xie. Cryptanalyzing a RGB image encryption algorithm based on DNA encoding and chaos map. Optics & Laser Technology, 2014, 60: 111-115.

5. Xie Tao, Yuansheng Liu, Jie Tang. Breaking a novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system. Optik-International Journal for Light and Electron Optics, 2014, 125(24): 7166–7169.

6. 汪泉，文中华，唐杰等. 分层法求强循环规划解. 计算机科学，2013，40(11): 291-294.

7. 王进宗，文中华，唐杰等. 不确定规划领域中带权值的观察信息约简. 计算机工程与应用，2014(优先出版).

8. 龙凤，文中华，唐杰等. 不确定规划中可达关系的快速求解算法. 计算机工程，2015，41(1): 196-200.

附录B (攻读硕士学位期间参与的科研项目)

1. 快速更新不确定规划解的算法研究，湖南省研究生创新项目(项目编号: CX2014B276)，2014 - 2015年。主持人。

2. 基于模型检测的不确定规划的状态可达性及其应用研究，国家自然科学基金项目(项目编号: 61070232)，2011 - 2013年。参与人。

3. 不确定规划中的观察信息约简方法及其应用研究，国家自然科学基金项目(项目编号: 61272295)，2013 - 2016年。参与人。

附录C (攻读硕士学位期间获奖情况)

1. 2012 - 2013年湘潭大学三好学生。

2. 2012 - 2013年国家奖学金。

3. 2014年纪念毛泽东同志诞辰120周年环校长跑二等奖。