第27卷第4期
2007年12月

数学理论与应用
MATHEMATICAL THEORY AND APPLICATIONS

Vol. 27 No. 4
Dec. 2007

一种基于Split－findmin和Set－maxima的最小支撑树灵敏度分析方法*

杨晓凌 谢 政 陈 挚
(国防科技大学理学院，长沙，410073)

摘 要 本文首先根据最小支撑树的截性质和圈性质给出了灵敏度分析的基本公式，然后基于现代图论算法中经典的Split－findmian数据结构介绍了树上边的灵敏度分析算法，最后将非树边的灵敏度分析转化为已有成熟的算法的Set－maxima问题进行处理.

关键词 最小支撑树灵敏度分析 Split－findmin数据结构 Set－maxima问题

Sensitivity Analysis of Minimum Spanning Trees Based on Split－Findmin and Set－Maxima

Yang Xiaoling Xie Zheng Chen Zhi
(School of Science, National University of Defense Technology, Changsha, 410073)

Abstract First of all, this paper presents the basic formula of sensitivity analysis according to the Cut property and Cycle property of minimum spanning trees (MST), then we introduce sensitivity analysis algorithm of tree edges based on classical Split－findmin data structure in modern graph theory algorithms, finally we solve analysis sensitivity problem of non－tree edges by turning it into Set－maxima problem, and at present muture algorithm exists for Set－maxima problem.

Keywords Sensitivity analysis of MST Split－findmin data structure Set－maxima Problem

1 引 言

最小支撑树的灵敏度分析是指，对给定一个网络$G=(V,E,w)$(其中w表示边权)及G的一个最小支撑树$T=\mathrm{MST}(G)$，G各边的权值在什么范围内变化时，最小支撑树T的权值仍为最小. 称上述各边变化的最大范围为最小支撑树的边灵敏度，记边e的灵敏度为$\mathrm{sens}(e)$.

本文第二节将介绍最小支撑树的某些重要性质，并且由这些性质出发我们给出最小支撑树灵敏度分析的基本公式. Split － findmin数据结构的概念以及由此产生的最小支撑树树上

* 吴 颉教授推荐
收稿日期：2007.7.18

边灵敏度分析的主要思想及算法，将在第三节中介绍. 最后我们将给出非树边的灵敏度分析向 Set − maxima 问题转化的具体方法.

2 最小支撑树的性质及灵敏度分析基本公式

定理 2.1[1] 若 T 为树，则 T 为无圈图，且在任意一对顶点之间加上一条边 e，$T+e$ 中恰有一个圈.

定理 2.2 给定一个无向连通网络 $G=(V,E,w)$ 及 G 的一个最小支撑树 $T=\mathrm{MST}(G)$，则对 G 中任何非空顶点子集 V'，截集 $[V',V\backslash V']$ 上权值最小的边必在 T 上.

定理 2.3 给定一个无向连通网络 $G=(V,E,w)$ 及 G 的一个最小支撑树 $T=\mathrm{MST}(G)$，则 G 中任何圈上权值最大的边必不在 T 上.

分别称定理 2.2 和定理 2.3 为最小支撑树的截性质和圈性质.

给定一个无向连通网络 $G=(V,E,w)$，以及 G 的一个最小支撑树 $T=\mathrm{MST}(G)$. 对非树边 e，由定理 2.1，可令 $C(e)\cup\{e\}$ 表示 $T+e$ 中唯一的圈，则 $C(e)$ 为 T 上连接边 e 两顶点间的一条路. 对树上的边 e，令 $C^{-1}(e)=\{f\in E(G)\backslash E(T)\mid e\in C(f)\}$，则 $C^{-1}(e)$ 表示在图 G 上由 $T\backslash e$ 所定义截集中不在树 T 上的边的集合. 由定理 2.2 和定理 2.3 可给出最小支撑树灵敏度分析的基本公式[3]：

$$\mathrm{sens(e)}=\begin{cases}\min\limits_{f\in c^{-1}(e)}\{w(f)\}, & e\in T\\ \max\limits_{f\in C(e)}\{w(f)\}, & e\notin T\end{cases}\tag{1}$$

这里，$\min\varnothing=\infty$.

于是根据定理 2.2、定理 2.3，对非 T 上的边 e，其权值增加量可以任意，但减少量必须小于 $w(e)-\mathrm{sens}(e)$. 类似的，对 T 上的边 e，其权值增加量必须小于 $\mathrm{sens}(e)-w(e)$，减少量则可以任意.

3 基于 split − findmin 数据结构的最小支撑树树上边的灵敏度分析

定义 1 Split − findmin

数据对象：$S=\{s_i\mid s_i=\{a_{1i},a_{2i},\cdots,a_{ik}\},a_{ji}$ 有权值 $k(a_{ji}),i=1,2,\cdots,N;k$ 与 i 相关$)\}$；

数据关系：$\boldsymbol{R}=\{\langle s_{i-1},s_i\rangle\mid s_{i-1},s_i\in\boldsymbol{S},i=1,2,\cdots,N-1\}$；

基本操作

init$(p_1,p_2,\cdots,p_n)$：初始化 $S=\{p_1,p_2,\cdots,p_n\}$，且 $k(p_i)=\infty,i=1,2,\cdots,n$；

spilt(p_j)：令 $S(p_i)=(p_j,\cdots,p_{i-1},p_i,\cdots,p_k),\boldsymbol{S}=\boldsymbol{S}\backslash S(p_i)\cup\{(p_j,\cdots,_{i-1})\},(p_i,\cdots,p_k)\}$；

findmin(p)：反回 $\min\limits_{f\in S(p)}\{k(p)\}$；

descreasekey(p,w)：令 $k(p)=\min\{k(p),w\}$

首先构造一个 Split − findmin 数据结构，具体要求为：

(1) 将最小支撑树 T 的顶点按某种顺序重新排列后得到序列 $(u_1,u_2,\cdots,u_n)$，初始化 $S=\{(u_1,u_2,\cdots,u_n)\}$.

一般地，S 中的序列对应于整个最小支撑树 T 的顶点集或者某个子树，并且对任意子树，与其对应的序列中各项点的标号构成了一个连续的正整数集.

(2)$k(v)$ 对应的是截集$[S(v),V\backslash S(v)]$ 中与 v 关联的边中仅值最小者.

定义 2[4]　设树 T 的根为 v_0，u，v 为树 T 上的一对顶点，满足如下两条件的顶点称之为 u，v 的最小公共祖先(LCA(u,v)：

(1)LCA(u,v) 是 u，v 的公共祖先；

(2) 与 u，v 的其它公共祖先相比，LCA(u,v) 与根 v_0 的距离最远.

基于 Split － findmin 数据结构的最小支撑树树上边的灵敏度分析算法如下：

Step 1　以任意顶点为根使最小支撑树 T 成为有根树.

对任意非树边(u,v)，除非 u 是 v 的祖先或 v 是 u 的祖先，否则用$(u,$LCA$(u,v))$ 和$(v,$LCA$(u,v))$ 代替边(u,v)，新边将延用旧边的权值.

若两顶点间引入多条边，则保留权值最小的一条边.

Step 2　以任意顶点为根，从根出发对 T 进行深度优先搜索，在探索过程中对各顶点进行标号，对每个点顶点将首次搜索到该顶点时它得到的标号作为该顶点的下标. 使得对任意子树，与其对应的序列中各顶点的标号构成了一个连续的正整数集.

进行操作 init$(u_1,u_2,0\cdots,u_n)$，即根为 u_1.

Step 3　设 $u_1^{(1)},u_2^{(1)},\cdots,u_{l_1}^{(1)}$ 是 u_1 的所有孩子，对 $u_1^{(1)},u_2^{(1)},\cdots,u_{l_1}^{(1)}$ 执行操作 spilt$(u_j^{(1)})$，$j=1,2,,\cdots,l_1$.

对所有的非树边$(u_k,u_1)(k>1)$，执行操作 descreasekey$(u_k,w(u_k,u_1))$.

Step 4　令 $i=i+1$. 若 $i=n$，算法终止，否则执行以下步骤：

$\forall\ i>1$，sens$(u_i,$parent$(u_i))=$ findmin(u_i).

设 $u_1^{(1)},u_2^{(1)},\cdots,u_{l_1}^{(1)}$ 是 u_i 的所以孩子，对 $u_1^{(1)},u_2^{(1)},\cdots,u_{l_1}^{(1)}$ 执行操作 spilt$(u_j^{(i)},j=1,2,\cdots,l_i$.

对所有的非树边$(u_k,u_i)(k>i)$，执行操作 descreasekey$(u_k,w(u_k,u_1))$.

转 Step 4.

该算法的复杂度为 $O(m\log\alpha(m,n)$. 此处，m，n 分别为图 G 的边数和顶点数，α 为反 Ackermann 函数.

4　最小支撑树非树边的灵敏度分析

定义 3　Set － maxima 问题：设 X 是含 n 个元素的有序空间，S 为 X 的若干子集构成的集族，即 $S=\{S_i|S_i\subset X,i=1,2,\cdots,m\}$，如何计算集合$\{\max S_i|i=1,2,\cdots,m\}$.

目前用于解决 Set － maxima 问题的算法主要分两种：

随机算法和矩阵分解法.

1990 年，Goddard，Schulman 和 King 提出了复杂度为 $O(nlog(m/n)+n)$ 随机算法[5].

1998 年 Liberatore 利用矩阵分解法对 KOMLOS 算法进行了拓展，且只需要进行 $\min\{O(m+n)\log^*(m+n)),O(n\log n)\}$ 次比较[6].

记 $E\backslash E(T)=\{e_1,e_2,\cdots,e_l\}$，$l\leq(m-n+1)$，最小支撑树非树边灵敏度分析算法如下：

Step 1 若 $i > l$,转 Step 3. 否则,对非树边 $e_i = (u_i, v_i)$,寻找 $LCA(u_i, v_i)$;

Step 2 由 $LCA(u_i, v_i)$ 出发在最小支撑树 T 上寻找最短路 $P_T(u_i, kLCA(u_i, v_i))$ 和 $P_T(v_i, LCA(u_i, v_i))$(采用 Dijkstra 算法),进而得到 $C(e_i)$. $i = i + 1$,转 Step 1;

Step 3 令 $X = E(T), S = \{S_i | S_i = E(C(e_i))\}$.

利用 Set − maxima 算法求出

$$\{\max S_i | i = 1, 2, \cdots, m\},$$

则 $\text{sens}(e_i) = \max S_i, i = 1, 2, \cdots, l$. 算法结束.

该算法的复杂度为 $O(mn^2)$.

参考文献

[1] 谢 政. 网络算法与复杂性(第二版). 长沙:国防科技大学出版社,2003.

[2] 严蔚敏,吴伟民. 数据结构(C 语言版). 北京:清华大学出版社,1996.

[3] S. Pettie. Sensitivity Analysis of Minimum Spananing Trees in Sub−inverse−Ackermann Time. ISSAC 2005, LNCS 2005, 3827, 964−973.

[4] M. A. Bender, M. Farach−Colton. The LCA Problem Revisited. LATIN2000, LNCS 1776, 88−94.

[5] W. Goddard, V. King, L Schulman. Optimal Randomized Algorithm for Local Sorting and Set − Maximan. ACM 1990.

[6] V. Liberatore. Matroids Decomposition Methods for the Set Maxima Problem. Society for Industrial and Applied Mathematics, 1998, 400−409.