低代价最短路径树的快速算法*

王 涛+, 李伟生

(北京交通大学 计算机与信息技术学院,北京 100044)

A Fast Low-Cost Shortest Path Tree Algorithm

WANG Tao+, LI Wei-Sheng

(School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China)

+ Corresponding author: E-mail: wtzyb4446@163.com, http://www.njtu.edu.cn

Received 2003-06-09; Accepted 2003-09-05

Wang T, Li WS. A fast low-cost shortest path tree algorithm. *Journal of Software*, 2004,15(5):660~665.
http://www.jos.org.cn/1000-9825/15/660.htm

Abstract: Low-Cost Shortest Path Tree is a commonly-used multicast tree type, which can minimize the end-to-end delay and at the same time reduce the bandwidth requirement if possible. Based on the low-cost shortest path tree algorithm DDSP (destination-driven shortest path) and through improving on the search procedure, a Fast Low-cost Shortest Path Tree (FLSPT) algorithm is presented in this paper. The Shortest Path Tree constructed by the FLSPT algorithm is the same as that constructed by the DDSP algorithm, but its computation complexity is lower than that of the DDSP algorithm. The simulation results with random network models show that FLSPT algorithm is more effective.

Key words: multicast; SPT (shortest path tree); steiner tree; MST (minimum spanning tree)

摘 要: 低代价最短路径树是一种广泛使用的多播树.它能够在保证传送时延最小的同时尽量降低带宽消耗.在 DDSP(destination-driven shortest path)算法的基础上,通过改进节点的搜索过程,提出了快速低代价最短路径树算法 FLSPT(fast low-cost shortest path tree).该算法构造的最短路径树与 DDSP 算法构造的树具有相同的性能,但其时间复杂度低于 DDSP 算法.随机网络模型的仿真结果表明,FLSPT 算法效率更高.

关键词: 多播;最短路径树;Steiner 树;最小生成树

中图法分类号: TP301 **文献标识码**: A

多播是一种群组通信的手段,要求将信息从一个数据源同时传送到多个目的地.为了有效地支持实时的大数据量数据传送的应用,一个好的多播路由算法应该能够控制数据传送的延时和带宽消耗.构造多播树是解决多播路由问题的常用方法.有 3 种不同类型的多播树[1]:基于数据源的树、Steiner 树和基于中心的树.这些不同类型的树中具有代表性的是基于源的最短路径树(shortest path tree,简称 SPT)和 Steiner 树.SPT 最主要的优点就是它使得源节点到每个目的节点的时延最小;而 Steiner 树则使得所有连接的消耗总和最小,最大限度地共享带宽,如果网络中除源节点外的每个节点都是目的节点,这棵树就是最小生成树(minimum spanning tree,简称

* 作者简介: 王涛(1980－),男,广西桂林人,硕士生,主要研究领域为计算机网络技术,图论算法设计于分析;李伟生(1945－),男,教授,主要研究领域为算法的设计与分析(并行算法、图论算法、最优化算法),网络数据库技术.

MST).

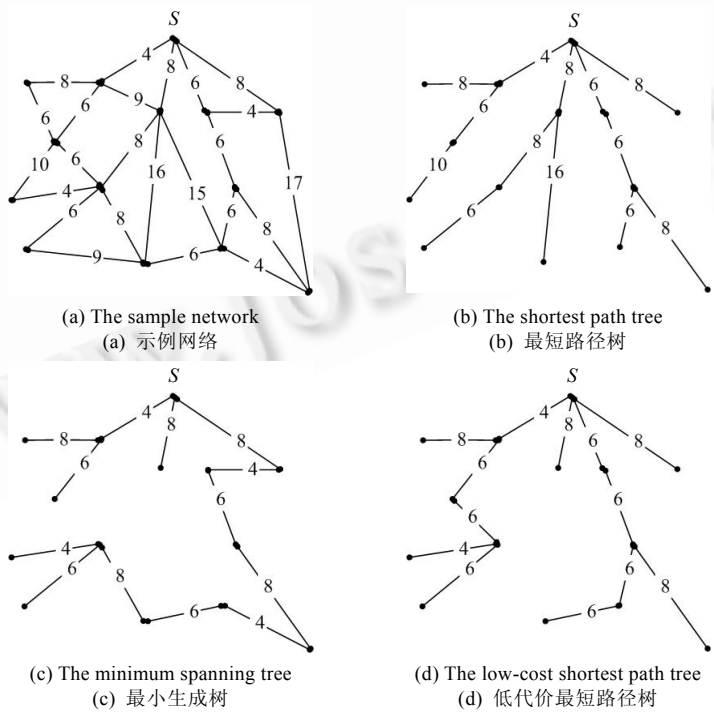
在给定的网络中,构造从源节点到其他目的节点的多播树,几乎不可能使它同时达到最小延时和最小总消耗,因此,一些学者研究构造满足指定时延限制的 Steiner 树[2~4].当网络变得足够大时,从源节点到一些目的节点的最短路径常常不只一条,而由此构成的 SPT 也就不是惟一的.在众多的 SPT 中,每棵树的总消耗也不尽相同.因而,有必要研究如何降低 SPT 的总消耗问题.

目前对低代价最短路径树构造问题的研究不多[5,6],而且相关算法都是在国外文献中提出来的,国内并没有针对此问题的研究.文献[5]提出最优最短路径树算法 SBPT(shortest best path tree),通过在构造 SPT 的过程中采用贪心策略选择最短路径,使所构造的生成树的总消耗得以降低.该算法每添加一个目的节点到生成树上的时间复杂度为 $O(n^2)$(n 为网络中节点总数),构造有 m 个目的节点的 SPT 总时间复杂度为 $O(mn^2)$,算法的时间复杂度较高而不适合大型网络求解.文献[7]提出了目的驱动多播路由算法 DDMC(destination-driven multicasting).该算法通过使目的节点之间尽可能共享路径来降低多播树的总消耗.文献[6]提出了目的驱动最短路径树算法 DDSP(destination-driven shortest path).它采用 Dijkstra 算法路径递增的基本思想,并结合 DDMC 算法目的节点共享路径的方法,当源节点到某节点的最短路径不惟一时,总是选择一条与其他目的节点的共享路径最长的最短路径,从而降低所构造 SPT 的总消耗.该算法时间复杂度为 $O(e\log n)$(e 和 n 分别为网络中边和节点的数目),与目的节点数目无关,因而适合计算目的节点数较多的最短路径树.

DDSP 算法在添加一个节点到生成树上时,要重新从已计算的节点中搜索其最优的父节点,存在重复计算.为了进一步提高计算效率,本文在 DDSP 算法的基础上进行改进,提出一种计算低代价最短路径树的快速算法 FLSPT(fast low-cost shortest path tree).它构造的多播树与 DDSP 算法构造的多播树具有相同性能,但时间复杂度比 DDSP 算法要低.

1 多播树

下面给出一个示例,说明各种不同类型多播树之间的差别,如图 1 所示.



(a) The sample network (a) 示例网络

(b) The shortest path tree (b) 最短路径树

(c) The minimum spanning tree (c) 最小生成树

(d) The low-cost shortest path tree (d) 低代价最短路径树

Fig.1 Different type multicast trees

图 1 不同类型的多播树

有给定网络如图 1(a)所示,节点 S 为多播源节点,其余为目的节点,图中各连线上的数值表示网络上该连接的消耗.设 CT 表示多播生成树上所有连接的消耗总和,PT 表示根节点 S 到各目的节点的路径长度总和.

如图 1(b)所示是一棵通过 Dijkstra 最短路径算法构造的 SPT,其 CT=100,PT=180.如图 1(c)所示是一棵通过 Prim 算法构造的 MST,其 CT=80,PT=206.该多播树上所有连接的消耗总和在所有多播树中最小,但其路径长度总和大于如图 1(b)所示的 SPT.图 1(d)也是一棵 SPT.该多播树的 CT=82,PT=180,其消耗总和比图 1(b)的 SPT 消耗总和要小,正是低代价最短路径树算法所要达到的目标.

2 FLSPT 算法

2.1 相关工作及算法基本思路

DDSP 算法[6]在保证与源节点路径最短的前提下,选择与已计算的目的节点最近的路径,通过目的节点之间共享尽可能长的路径来降低生成树的总消耗.DDSP 算法基本思路如下:

1) 引入向量 CST,$rCST$ 和 S.$CST(u)$表示节点 u 到源节点 s 的最短路径长度,$rCST(u)$表示节点 u 到生成树 T 上最近的目的节点的距离.$S(u)$为 1 则表示节点 u 已经计算过,否则表示 u 尚未计算.

2) 初始化向量 CST,$rCST$ 和 S,对于图 G 中任意节点 u,$S(u)$赋值为 0,如果存在源节点 s 到节点 u 的边,则将 $CST(u)$和 $rCST(u)$赋值为边(s,u)的权值 $cost[s,u]$,否则赋值为∞.初始化生成树 T 以源节点 s 为其根节点,$CST(s)$和 $rCST(s)$赋值为 0,$S(s)$赋值为 1.

3) 选择节点 u,使得

$$CST(u)=\min\{CST(w)|1\le w\le n,S(w)=0\}.$$

令 $S(u)$=1.选择节点 p 作为节点 u 的父节点,使得节点 p 满足

$$CST(u)=CST(p)+cost[p,u] \text{ 且 } rCST(u)=rCST(p)+cost[p,u].$$

如果节点 u 为一个目的节点,则修改 $rCST(u)$=0,并将节点 u 添加到生成树 T.

4) 对于任意节点 v,如果 $S(v)$=0 且 $cost[u,v]<\infty$,修改此节点的 $CST(v)$和 $rCST(v)$值.如果

$$CST(u)+cost[u,v]<CST(v),$$

则令

$$CST(v)=CST(u)+cost[u,v],\quad rCST(v)=rCST(u)+cost[u,v];$$

否则,如果

$$CST(u)+cost[u,v]=CST(v) \text{ 且 } rCST(v)>rCST(u)+cost[u,v],$$

则令

$$rCST(v)=rCST(u)+cost[u,v].$$

5) 重复步骤 3)和步骤 4),直到所有目的节点都已添加到生成树 T 上为止.

该算法采用 Dijkstra 算法路径递增的基本思想,并结合 DDMC 算法[7]目的节点共享路径的方法,通过用 $rCST$ 向量保存节点到已计算目的节点的最近距离,在有多条最短路径时总是选择能够使 $rCST$ 分量值最小的路径,最大限度地与其他目的节点共享路径,因而能够降低最短路径树的总消耗.与 SBPT 算法[5]相比,DDSP 算法时间复杂度较低,而且与目的节点数目无关,适合计算目的节点数较多的多播树.此外,DDSP 算法中每个节点只需知道其邻接节点的信息,无须保存全网络状态.

仔细研究可以发现,DDSP 算法中的步骤 4)在访问当前计算节点的邻接节点时,只是修改邻接节点的 CST 和 $rCST$ 分量值,并不修改邻接节点的父节点标识,而是当某一节点被选中作为当前计算节点时(步骤 3)),才从其他节点中选择满足最小 $CST(u)$和 $rCST(u)$的节点作为其父节点,在最坏情况下需要访问所有节点才能找出其父节点,导致计算的重复.

为了进一步提高计算效率,本文在 DDSP 算法的基础上提出 FLSPT 算法.在根据当前计算节点访问其邻接节点时,如果需要修改邻接节点的最短路径长度和最近目的节点距离,则同时修改此邻接节点的父节点标识为当前计算节点,因此,当某一节点被选中作为当前计算节点时,无须重新计算其父节点,可以减少计算量.因为邻

接节点的最短路径长度值和最近目的节点距离值是根据当前计算节点进行修改的,所以在此时修改父节点标识与在后面根据向量值重新计算父节点是等效的.因此,FLSPT 算法计算得出的最短路径树性能与 DDSP 算法一样,但是计算效率比 DDSP 算法要高.

2.2 算法描述

给定图 $G=\{V,E\}$,节点 s 为多播源节点,目的节点集合为 D,构造以 s 为根节点且包含所有目的节点的最短路径树 T.假定对于任意存在的边(u,w),其权值为 $C(u,w)>0$.

引进 3 个向量 *Dist*,*Mist* 和 *Parent*.*Dist*[*u*]表示当前搜索到的从源节点 *s* 到节点 *u* 的最短路径长度;*Parent*[*u*]表示在 FLSPT 算法所选择的最短路径上节点 *u* 的前一个节点,也就是最短路径生成树 *T* 上节点 *u* 的父节点;*Mist*[*u*]表示生成树 *T* 上已计算的目的节点到节点 *u* 的最短距离.设 *ADJ*(*u*)表示节点 *u* 的邻接节点集,*Q* 为一个待发展节点的序列,存储已被访问但尚未被添加到生成树 *T* 上的节点.

```
FLSPT Algorithm(G,s,D)
1.  Initialize tree T with source node s and clear Q;
2.  For all w∈V Do                                          //算法初始化
3.      If w∈ADJ(s) Then
4.          Dist[w]←C(s,w); Mist[w]←C(s,w); Parent[w]←s;
5.          Q←Q∪{w};
6.      Else
7.          Dist[w]←∞; Mist[w]←∞; Parent[w]←nil;
8.      End If
9.  End For
10. Dist[s]←0; Mist[s]←0; Parent[s]←nil;                    //源节点为第 1 个已计算节点
11. While there exists a node in D that has not been added to tree T Do
12.     Select node u from Q which statisfies Dist[u]=min{Dist[m]|m∈Q};
13.     If u is a destination node Then                     //如果是目的节点则将最短路径添加到 T
14.         Establish shortest path from source node s to node u;
15.         Mist[u]←0;                                      //目的节点则需将 Mist 向量清零
16.     End If
17.     For all w∈ADJ(u) Do
18.         If Dist[w]=∞ Then Q←Q∪{w} End If
19.         If Dist[u]+C(u,w)<Dist[w] Then                  //调整节点的可达最短路径长度
20.             Dist[w]←Dist[u]+C(u,w); Mist[w]←Mist[u]+C(u,w); Parent[w]←u;
21.         Else If Dist[u]+C(u,w)=Dist[w] Then             //最短路径不惟一
22.             If Mist[u]+C(u,w)<Mist[w] Then              //发现更优的父节点
23.                 Mist[w]←Mist[u]+C(u,w); Parent[w]←u;    //记录最优父节点
24.             End If
25.         End If
26.     End For
27. End While
```

Fig.2 FLSPT algorithm

图 2 FLSPT 算法

如图 2 所示的 FLSPT 算法与 DDSP 算法一样,当源节点到某节点最短路径不惟一时,总是选择一条与其他目的节点的共享路径最长的最短路径,从而降低所构造的 SPT 的总消耗.它们所构造的 SPT 具有相同的性

能.DDSP 算法在添加节点到生成树上时都需要重新从已计算的节点中搜索其父节点,而 FLSPT 算法在计算节点的可达最短路径长度时保存其最优的父节点,无须重新搜索,从而可以减少计算量.

2.3 算法时间复杂度分析

用 n 表示图 G 中节点数目,e 表示所有边的数目,$d(u)$表示节点 u 的邻接节点数目.

算法的初始化时间复杂度都是 $T_1=O(n)$.

算法的关键是从待发展节点序列 Q 中选择最合适的节点进行下一步扩展.解决这一问题的关键在于使用什么样的数据结构.较常用的数据结构有数组、单双链表、堆栈、哈希散列和队列等.将待发展节点序列 Q 构造成 Fibonacci 堆,则从 Q 中输出最小节点等操作可以在 $O(\log n)$时间内完成.每一个节点最多往序列 Q 添加一次,在将该节点添加到生成树上时从 Q 中取出,并且每添加一个新节点 u 到生成树上后需要访问 $d(u)$个 u 的邻接节点.在所构造的 SPT 中最多包含 n 个节点,所以算法构造 SPT 时间复杂度为

$$T_2 = n\log n + \sum_{i=1}^{n} d(V_i) = O(n\log n + e).$$

算法的总时间复杂度为

$$T = T_1 + T_2 = O(n) + O(n\log n + e) = O(n + n\log n + e) = O(n\log n + e).$$

它比 DDSP 算法的时间复杂度 $O(e\log n)$ 要低.

2.4 算法空间复杂度分析

FLSPT 算法除了进行图的存储以外,还需要 3 个包含 n 个分量的一维向量和一个待发展节点集来存储计算过程的中间信息,而待发展节点集中最多包含 n 个节点.因此,采用图的邻接表存储方式,FLSPT 算法的空间复杂度为

$$S = O(e+n) + 4O(n) = O(e+5n) = O(e+n).$$

FLSPT 算法与 DDSP 算法相比,少了一个标识节点是否已经计算过的向量 S,增加了一个存储代发展节点集的序列 Q,向量 S 和序列 Q 的存储空间都为 $O(n)$.因此,若采用相同的存储方式存储图,FLSPT 算法和 DDSP 算法的空间复杂度是相等的.

3 仿真实验及分析

为了验证本文提出的算法的正确性和有效性,采用多组随机网络模型进行实验,比较 DDSP 算法和 FLSPT 算法产生的最短路径树以及计算时间.我们发现,在所有的随机网络模型中,DDSP 算法和 FLSPT 算法产生的最短路径树性能是一样的,但 FLSPT 算法的计算时间远小于 DDSP 算法.

为了产生具有实际网络特征的随机网络图,我们采用文献[8]中的方法.节点随机分布在直角坐标系内,边(u,v)的分布概率为

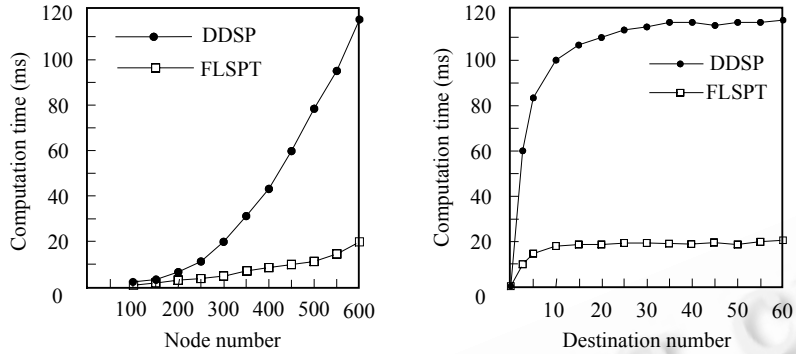
$$P(u,v) = \beta e^{-d(u,v)/L\alpha}.$$

这里,$d(u,v)$是节点 u 和 v 之间的 Euclid 距离,L 是任意两节点间的最大距离,α和β是调节网络特征的参数.如果α增加,长边相对短边的比例就增加;如果β增加,节点的度就随之增加.调整α和β可以产生不同类型的随机网络图,使之更接近实际网络.为了讨论方便起见,采用与文献[8]相同的参数:α=0.4,β=0.4.本文中所有的仿真实验均在主频 800MHz、内存 256M 的计算机上完成.

图 3(a)是目的节点数固定为 100 时,DDSP 算法和 FLSPT 算法的计算时间随网络节点数目变化的曲线,横坐标为网络节点数目,纵坐标为计算时间(单位:ms).从图中可见,当目的节点数目固定时,DDSP 算法和 FLSPT 算法的计算时间随着网络节点数目的增加而增加,但 FLSPT 算法的增长幅度较小.图 3(b)是网络节点数固定为 600 时,DDSP 算法和 FLSPT 算法的计算时间随目的节点数目变化的曲线.两种算法的计算时间都随目的节点数目的增加而增加,但 FLSPT 算法的增长幅度较小.

从图 3 可以看出,FLSPT 算法与 DDSP 算法一样,计算时间主要由网络节点数目决定.当目的节点数目达到

一定数量时,目的节点数目的增加对计算时间影响较小.



(a) Vary with nodes numbers
(a) 随网络节点数目变化

(b) Vary with destination numbers
(b) 随目的节点数目变化

Fig.3 Curves of computation time variety
图 3 计算时间变化曲线

4 结 语

本文对计算低代价最短路径树的相关算法进行了介绍,并提出一种快速的低代价最短路径树算法 FLSPT. FLSPT 算法与 DDSP 算法相比,所构造的最短路径树性能相同,而计算速度有很大提高,是一种值得推广使用的高效算法.

References:

[1] Diot C, Dabbous W, Crowcroft J. Multipoint communication: A survey of protocols, functions, and mechanisms. IEEE Journal on Selected Areas in Communications, 1997,15(3):277~290.

[2] Zhang BX, Kruntz MM, Chen CJ. A fast delay-constrained multicast routing algorithm. In: IEEE Communications Society, ed. Proc. of the 2001 IEEE Int'l Conf. on Communications, Vol 9. Los Alamitos: IEEE Press, 200. 2676~2680.

[3] Yang M, Xie XR. A fast routing algorithm for delay-constrained low-cost multicast. Journal of Computer Research and Development, 2000,37(6):726~730 (in Chinese with English abstract).

[4] Wang MZ, Xie JY, Zhang JY. Strategy of constructing minimum cost multicast routing tree with delay and delay variation bounds. Chinese Journal of Computers, 2002,25(5):534~541 (in Chinese with English abstract).

[5] Fujinoki H, Christensen K. The new shortest best path tree (SBPT) algorithm for dynamic multicast tree. In: IEEE Computer Society, ed. Proc. of the 24th IEEE Int'l Conf. on Local Computer Networks. Los Alamitos: IEEE Press, 1999. 204~211.

[6] Zhang BX, Mouftah HT. A destination-driven shortest path tree algorithm. In: IEEE Communications Society, ed. Proc. of the 2002 IEEE Int'l Conf. on Communications, Vol 4. Los Alamitos: IEEE Press, 2002. 2258~2262.

[7] Shaikh A, Shin KG. Destination-driven routing for low-cost multicast. IEEE Journal on Selected Areas in Communications, 1997, 15(3):373~381.

[8] Waxman BM. Routing of multipoint connections. IEEE Journal on Selected Areas in Communications, 1988,6(9):1617~1622.

附中文参考文献:

[3] 杨明,谢希仁.一个快速的时延有界低代价多播路由算法.计算机研究与发展,2000,37(6):726~730.

[4] 王明中,谢剑英,张敬辕.时延及时延抖动限制的最小代价多播路由策略.计算机学报,2002,25(5):534~541.