DOI：10.7652/xjtuxb201308007

# 改进的时延约束 Steiner 树算法

徐剑[1,2]，倪宏[2]，邓浩江[2]，刘磊[2]
（1. 中国科学院大学，100049，北京；2. 中国科学院声学研究所国家网络新媒体工程技术研究中心，100190，北京）

**摘要**：针对现有时延约束 Steiner 树算法时间复杂度较高以及生成的组播树代价较高的问题，提出了一种改进的时延约束 Steiner 树算法。该算法采用 Dijkstra 算法路径递增的基本思想和链路共享的方法，在快速搜索阶段，依次搜索到当前树有最小可行代价的节点，将目的节点通过最小可行代价路径加入组播树；在异常处理阶段，将遗漏的目的节点通过最小时延路径加入组播树，进而生成满足时延约束的 Steiner 树。理论分析和实验结果表明，与同类算法相比，该算法能够以较低的时间复杂度，取得较好的组播树代价。

**关键词**：Steiner 树；代价；时延约束；路径递增；链路共享
**中图分类号**：TP393 **文献标志码**：A **文章编号**：0253-987X(2013)08-0038-06

## An Advanced Algorithm for Delay-Constrained Steiner Tree

XU Jian[1,2], NI Hong[2], DENG Haojiang[2], LIU Lei[2]
(1. University of Chinese Academy of Sciences, 100049, Beijing; 2. National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, 100190, Beijing)

**Abstract**: An advanced algorithm for delay-constrained Steiner tree is proposed to focus on the problem that the existing delay-constrained Steiner tree algorithms usually lead to high multicast cost and high time complexity. The proposed algorithm adopts the idea of path increasing in Dijkstra algorithm and the method of link sharing. It iteratively searches the node with the least feasible cost to the current tree and adds the destination node to the current tree through its feasible least cost path in the fast search stage. The missing destination nodes are added to the current tree through its least delay path in the exception handling stage of the algorithm. The delay-constrained Steiner tree is constructed through these two stages of the algorithm. Theoretical analysis, experimental results and comparisons with similar algorithms show that the proposed algorithm can construct multicast tree in lower cost and time complexity.

**Keywords**: Steiner tree; cost; delay-constrained; path increasing; link sharing

以实时多媒体应用（如网络会议、网络电视和网络教室等）为代表的新型网络应用，对组播（multicast）通信服务提出了迫切的要求。组播利用树结构来分发数据，借助中间节点进行分发处理，数据分组只在组播树的分叉节点才会被复制，而在每条链路上只需传递一次，从而实现了高效的内容并发传输。组播既可以在网络层实现，也可以在应用层实现，如 IP 组播[1]、覆盖多播网络（overlay multicast network，OMN）[2-3]等。

根据不同应用需求生成覆盖源节点和目的节点

收稿日期：2012-09-28。 作者简介：徐剑（1985—），男，博士生；倪宏（通信作者），男，研究员，博士生导师。 基金项目：国家高技术研究发展计划资助项目（2011AA01A102）；国家科技支撑计划资助项目（2011BAH11B04）；中国科学院战略性先导科技专项子课题（XDA06010302）。
网络出版时间：2013-05-13 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20130513.1746.003.html

集的组播树是组播路由研究的核心问题。对于实时多媒体应用，从用户的角度出发，希望能得到一定的服务质量保证，特别是端到端传输时延需限制在可接受范围内；从运营者的角度出发，希望能生成代价尽可能小的组播树，以优化带宽等网络资源的使用；同时从用户和运营者的角度出发，则希望能在保证服务质量的同时最小化组播树代价。为此，研究者提出了时延约束的最小代价树问题，在数学上可归结为时延约束 Steiner 树（delay-constrained Steiner tree，DCST）问题[4-5]。

本文首先给出了问题描述，接着提出了一种改进的解决 DCST 问题的启发式算法，与典型的同类算法相比，该算法在组播树代价、时间复杂度等方面均有良好的性能表现，能够以较低的时间复杂度取得较好的组播树代价。

## 1　问题描述及相关算法介绍

网络可用图 $G=(V,E)$ 来描述，其中 $V$ 为节点（路由器或端系统）集合，$E$ 为边（节点之间的链路）集合，网络规模 $n=|V|$。组播组可用 $g=s\cup D$ 来描述，其中 $s$ 为源节点，$D(D\subseteq V)$ 为目的节点集合，目的节点数为 $m=|D|$。

如果节点 $u$ 与 $v$ 有边直接相连，则 $u$ 是 $v$ 的邻居节点，记为 $u\in\text{adjacency}(v)$；边 $\forall e\in E$ 的代价函数为 $\text{cost}(e)$，时延函数为 $\text{delay}(e)$；节点 $u$ 到 $v$ 的路径 $P(u,v)$ 的代价为 $\text{cost}(P(u,v))=\sum_{e\in P(u,v)}\text{cost}(e)$，时延为 $\text{delay}(P(u,v))=\sum_{e\in P(u,v)}\text{delay}(e)$。给定时延上限 $\Delta$，如果 $\text{delay}(P(u,v))\leqslant\Delta$，则称 $P(u,v)$ 是节点 $u$ 到 $v$ 的可行路径。

时延约束 Steiner 树（DCST）问题定义如下：给定网络 $G=(V,E)$，给定时延上限 $\Delta$，为组播组 $g=s\cup D$ 求解一棵以 $s$ 为根、覆盖 $D$ 中所有节点的组播树，记为 $T(V_T,E_T)$，$D\subseteq V_T\subseteq V$，$E_T\subseteq E$，使得 $s$ 到每个目的节点的路径均是可行的，且树的链路代价之和最小，即满足

$$\left.\begin{aligned}&\min\sum_{e\in T}\text{cost}(e)\\&\text{s.t.}\quad\text{delay}(P(s,u))\leqslant\Delta,\forall u\in D,P(s,u)\in T\end{aligned}\right\}\tag{1}$$

组播树 $T$ 中源节点和目的节点以外的节点称之为 Steiner 节点。

DCST 问题是一个 NP 完全问题[6-9]，长期以来，人们已提出了多种启发式解决方案，如由 Kompella、Pasquale and Polyzos 提出的 KPP 算法[6]、Zhu 等人提出的边界最短组播算法（bounded shortest multicast algorithm，BSMA）[7]以及周灵等人提出的延迟约束最小路径启发式算法（delay-constrained minimum path heuristic，DCMPH）[9]等。

KPP 算法首先求出任意 2 个节点之间满足时延约束的最小代价路径，并以此构建完全封闭图，然后通过最小生成树算法产生组播树，最后将组播树中的边用原始网络中的链路代替，算法的时间复杂度为 $O(\Delta n^3)$。

BSMA 算法首先使用迪杰斯特拉最短路径树算法（Dijkstra shortest path tree，Dijkstra SPT）求出从源节点到各目的节点的最小时延路径生成最小时延树，在不违反时延约束的情况下，通过 kth-最短路算法替换树中代价较高的超边，使树的总体代价不断降低。该算法求解的组播树代价较小，但其时间复杂度是 $O(kn^3\log n)$，不适应求解大型网络[8]。

DCMPH 算法同样首先使用 Dijkstra SPT 算法生成最小时延树，并计算目的节点到各节点的最小代价，如果目的节点到当前树的最小代价路径时延满足要求，将目的节点通过该路径加入组播树；否则将目的节点通过最小时延路径加入组播树，去除形成的环路。DCMPH 算法的时间复杂度为 $O(mn^2)$。

## 2　改进的时延约束 Steiner 树算法

本文借鉴 Dijkstra 算法路径递增的基本思想和链路共享的方法，提出了一种改进的时延约束 Steiner 树算法（advanced delay-constrained Steiner tree，ADCS），在保证目的节点与源节点路径时延满足约束的前提下，通过目的节点之间共享链路来优化组播树代价。本文假设链路时延和时延约束均为正整数[6]，因为从实时多媒体应用的体验要求和链路时延测量的精度考虑，时延精确到毫秒时可认为是整数，另外对于时延要求极其苛刻的情况，可以先将非整数时延离散化[10]。

### 2.1　算法的基本思想

ADCS 算法的基本思想是，从空集扩展生成时延约束 Steiner 树，采用链路共享的方法，依次选择到当前树有最小可行代价的目的节点，将该节点通过最小可行代价路径加入组播树，使每次扩展时树代价增加最小。但是，查找遍历所有可行路径的时间复杂度过高，为了降低时间复杂度，借鉴 Dijkstra 算法路径递增的基本思想来定义节点搜索顺序，并允许错过部分代价较高的可行路径，以实现快速搜

索。ADCS 算法主要包括以下 3 个阶段:准备阶段、快速搜索阶段和异常处理阶段。

2.1.1 准备阶段 准备阶段运用 Dijkstra SPT 算法,计算从源节点到各目的节点的最小时延路径生成最小时延树 $T_{\text{delay}}$,判断是否存在满足时延约束 $\Delta$ 的组播树,如果不存在则算法结束,否则进入快速搜索阶段。$T_{\text{delay}}$保存源节点到各目的节点的最小时延路径,这些路径可用于在异常处理阶段遗添加漏的目的节点。

2.1.2 快速搜索阶段 快速搜索阶段为网络中每个节点引入 2 个变量 $\delta$ 和 $\pi$。$\delta(u,i)$记录节点 $u$ 到当前树时延约束等于 $i$ 的最小代价,$\pi(u,i)$记录 $u$ 在时延约束等于 $i$ 的最小代价路径上的父节点。

节点到树时延约束等于 $i$ 的最小代价定义如下:对于给定树 $T$(以 $s$ 为根)、节点 $u$,$u$ 经过路径 $P(s,u)$到达 $T$ 的代价是 $P(s,u)$,去掉共享链路的路径代价,记为 $\text{cost}(u,P)=\sum_{e\in P,e\notin T}\text{cost}(e)$(表示节点通过该路径加入树后树代价的增加值),$u$ 到 $s$ 的时延约束等于 $i$ 的路径集合记为 $S_P(u,i)=\{P\mid \text{delay}(P)=i\}$,则 $u$ 到 $T$ 时延约束等于 $i$ 的最小代价$\delta(u,i)=\min\{\text{cost}(u,P)\mid P\in S_P(u,i)\}$,进而可得 $u$ 到 $T$ 的最小可行代价 $\delta(u)=\min\{\delta(u,i)\mid i\in[0,\Delta]\}$,$\delta(u)$相应的路径为 $u$ 到 $T$ 的最小可行代价路径。$\delta(u,i)$满足递归关系式

$$\delta(u,i)=\begin{cases}0, & u=s,i=0\\ \min\{\delta(w,i-\text{delay}(w,u))+\text{cost}(w,u)\mid \\ \quad w\in\text{adjacency}(u)\}, & u\neq s,0\leqslant i\leqslant\Delta\\ \infty, & \text{其他}\end{cases}\tag{2}$$

快速搜索阶段依次搜索具有最小 $\delta$ 值(到当前树有最小可行代价)的节点,如果该节点是目的节点,则将该节点通过最小可行代价路径加入组播树。快速搜索阶段具体包括以下步骤。

(1)初始化。当前时延约束 Steiner 树 $T=\varnothing$,待搜索节点集合 $Q=V$,令 $\delta(s,0)=0$,令 $\delta(u,i)=\infty$,$u\neq s$,$0\leqslant i\leqslant\Delta$,令 $\pi(u,i)=-1$。

(2)从 $Q$ 中选择到 $T$ 有最小可行代价(小于$\infty$)的节点,如果有多个节点可选,则选择路径时延最小的节点,记为 $u$,其最小可行代价路径时延为 $i$,执行步骤 3。

(3)如果 $u$ 是源节点 $s$,将 $u$ 加入 $T$,执行步骤 5;如果 $u$ 是目的节点,令 $w=u$,$k=i$,执行步骤 4,将 $u$ 到 $T$ 的最小可行代价路径加入 $T$,否则直接执行步骤 5。

(4)如果 $w\notin T$,则将 $w$ 及路径上相应的边 $e(w,\pi(w,k))$加入 $T$,令 $\delta(w,k)=0$,如果 $w$ 已搜索过,则重新加入 $Q$,沿路径前进,$p=\pi(w,k)$,$k=k-\text{delay}(p,w)$,$w=p$。重复此步,直至 $w\in T$,执行步骤 5。

(5)对 $u$ 的每个邻居节点 $v$,令 $l=i+\text{delay}(u,v)$,如果 $v\notin T$、$l\leqslant\Delta$ 且 $\delta(v,l)>\delta(u,i)+\text{cost}(u,v)$,则更新 $v$ 到 $T$ 的代价信息,令 $\delta(v,l)=\delta(u,i)+\text{cost}(u,v)$,$\pi(v,l)=u$;如果 $v$ 已搜索过,$\delta(v,l)$是 $v$ 的最小可行代价且 $l$ 是最小时延,则 $v$ 重新加入 $Q$。

(6)将 $u$ 从待搜索集合 $Q$ 中删除。

(7)重复执行步骤 2~6,当 $T$ 覆盖所有目的节点,则 ADCS 算法正常结束;当步骤 2 中无法找到满足要求的节点,此时 $T$ 未能覆盖所有目的节点,则进入异常处理阶段。

2.1.3 异常处理阶段 快速搜索阶段虽然可以实现快速遍历,并为大部分目的节点找到最小可行代价路径,但是因为其快速性,该过程会跳过一些可行但是代价较高的路径,导致某些目的节点最终无法在快速搜索阶段加入组播树,进而无法生成覆盖所有目的节点的组播树。

对于遗漏的未能加入组播树的目的节点,将在异常处理阶段通过最小时延路径加入组播树,此时可能会出现环路,需要消除环路。消除环路过程如下:新路径加入组播树时,如果发现新路径上节点已在组播树中,将形成环路,则沿原路径删除节点直至遇到目的节点或源节点。

### 2.2 算法正确性分析

**定理 1** 在快速搜索阶段,目的节点均通过其到当前树的最小可行代价路径加入组播树。

**证明** 使用反证法证明。假设在快速搜索阶段,目的节点 $u$ 以路径 $P_1$ 加入组播树 $T$,$u$ 到树最小可行代价路径为 $P_2$。$u$ 沿 2 条路径到达源节点 $s$ 的时延分别为 $\text{delay}(P_1)$、$\text{delay}(P_2)$。由于 $P_2$ 是 $u$ 到 $T$ 的最小可行代价路径,如果 $u$ 以 $P_1$ 加入 $T$,则此时有 $\delta(u)=\delta(u,\text{delay}(P_1))$,发生此情况是因为搜索 $u$ 时未搜索过 $P_2$,但是 $u$ 在 $P_2$ 上的父节点 $w$ 的最小代价满足条件 $\delta(w)\leqslant\delta(w,\text{delay}(P_2)-\text{delay}(w,v))<\delta(u,\text{delay}(P_2))<\delta(u)$,按照搜索顺序 $w$ 必先于 $u$ 被搜索,因此搜索 $u$ 前 $P_2$ 必搜索过,假设不成立,定理 1 得证。

**定理 2** 消除环路过程不会破坏组播树的时延上界。

**证明**　在快速搜索阶段，任意节点到组播树 $T$ 的最小可行代价路径仅有 1 条（如果有多条路径代价相等，ADCS 算法只会选择其中 1 条加入 $T$），因此，2 条最小可行代价路径之间不会形成环路。同理，2 条最小时延路径也不会形成环路。因此，环路只可能在最小可行代价路径和最小时延路径之间形成。如图 1 所示。目的节点 $v$、$w$ 分别通过路径 $s\rightarrow P_1\rightarrow u\rightarrow P_3\rightarrow v$ 和 $s\rightarrow P_2\rightarrow u\rightarrow P_4\rightarrow w$ 加入 $T$，形成环路。不妨假设 $s\rightarrow P_1\rightarrow u\rightarrow P_3\rightarrow v$ 为最小可行代价路径，$s\rightarrow P_2\rightarrow u\rightarrow P_4\rightarrow w$ 是最小时延路径，显然 $P_2$ 是 $u$ 的最小时延路径，于是有 $\mathrm{delay}(P_1)\geqslant\mathrm{delay}(P_2)$。消除环路过程中，沿原路径 $P_1$ 删除节点直至遇到目的节点或源节点，不会将已覆盖的目的节点删除，同时路径 $s\rightarrow P_2\rightarrow u\rightarrow P_3\rightarrow v$ 时延也满足约束，所以消除环路过程不会破坏当前组播树的时延上限。

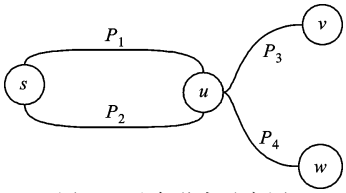


图 1　环路形成示意图

根据定理 1、2 可知，ADCS 算法在快速搜索阶段将目的节点通过当前树的最小可行代价路径加入组播树，通过所有目的节点共享链路来降低组播树代价，从而优化组播树代价；在异常处理阶段将遗漏的目的节点通过最小时延路径加入组播树，消除环路过程也不会破坏组播树时延上界。因此，如果存在满足时延约束的组播树，ADCS 算法就能找到一棵满足时延约束的较低代价组播树。

### 2.3　算法时间复杂度分析

（1）准备阶段：Dijkstra SPT 算法的时间复杂度为 $T_1=O(n^2)$。

（2）快速搜索阶段：步骤 1 初始化的时间复杂度为 $O(\Delta n)$；步骤 2 从待搜索集合中选择最合适的节点进行下一步扩展，即从向量 $\delta$ 中找出最小值，根据采用数据结构的不同算法复杂度不同，本文采用数组来实现，时间复杂度为 $O(\Delta n)$；步骤 3、4 将目的节点通过最小可行代价路径加入当前树，时间复杂度为 $O(l_u)$（$l_u$ 是路径节点数）；步骤 5 更新邻居节点的最小代价信息，时间复杂度为 $O(\Delta d_u)$（$d_u$ 是未加入组播树的邻居节点数），显然有 $d_u\leqslant|\mathrm{adjacency}(u)|$；步骤 7 循环执行步骤 2～6，循环次数即节点总搜索次数，虽然存在部分节点重新搜索情况，但历次实验表明节点平均搜索次数少于 2 次，故步骤 7 的时间复杂度为 $O(n)$。快速搜索阶段的时间复杂度 $T_2=O\left(\Delta n+\Delta n^2+\sum_{u=1}^{m}l_u+\Delta\sum_{u=1}^{n}d_u\right)$，因 $\sum_{u=1}^{m}l_u\leqslant|V_{\mathrm{T}}|\leqslant n$，$\sum_{u=1}^{n}d_u\leqslant\sum_{u=1}^{n}|\mathrm{adjacency}(u)|<n^2$，故 $T_2=O(\Delta n^2)$。

（3）异常处理阶段：对于未加入组播树的目的节点，将最小时延路径加入组播树，判断是否存在环路，消除环路，最坏时间复杂度为 $O(\Delta l_u|V_{\mathrm{T}}|)$，因此异常处理阶段的最坏时间复杂度 $T_3=O\left(\Delta\sum_{u=1}^{m}l_u|V_{\mathrm{T}}|\right)=O(\Delta n^2)$。

所以，ADCS 算法时间复杂度 $T_1+T_2+T_3=O(\Delta n^2)$，远小于 KPP 算法和 BSMA 算法，与 DCMPH 算法相当。

## 3　仿真实验

为了验证本文提出算法的正确性和有效性，本节在 Waxman 随机网络模型[11]上进行仿真实验，对比 KPP、BSMA、DCMPH、ADCS 算法的性能。仿真实验采用 NS-2 平台，通过 GT-ITM 拓扑生成器生成 Waxman 随机网络模型，模型参数取 $\alpha=0.3$，$\beta=0.3$，$l=20$ ms，其中 $\alpha$ 表示网络中 2 点邻接的概率，$\beta$ 表示网络中长边数和短边数的比值，而邻接点间的直线距离可作为链路延迟，即链路延迟最大值为 $\lceil 2^{1/2}l\rceil$ ms，链路代价是在 1～10 之间均匀分布的随机数。实验时对每种仿真环境使用 10 个不同的网络，每个网络对随机生成且存在满足时延约束组播树的 100 个组播组测试，共 1 000 次，求其平均值作为结果。

### 3.1　树代价

**实验 1**　比较时延变化时各种算法生成的组播树代价，网络规模固定为 100，组播组目的节点数固定为 40，时延约束在 20～150 ms 之间变化，实验结果如图 2 所示。可以看出，随着时延约束的放松，可行路径增加，各算法生成的组播树代价不断降低直至稳定。ADCS、BSMA 算法生成的组播树代价明显小于 KPP、DCMPH 算法。在时延约束较小的情况下，ADCS 算法生成的组播树代价高于 BSMA 算法，随着时延约束的逐渐放松，ADCS 算法生成的组播树代价略低于 BSMA 算法。这得益于 ADCS 算法在快速搜索阶段依次添加使树代价增加最小的目的节点，随着时延约束的增加，ADCS 算法能在快速处理阶段将所有目的节点加入组播树，不需要进行

异常处理。ADCS 算法的总体平均组播树代价比 BSMA 算法低 1.3%。

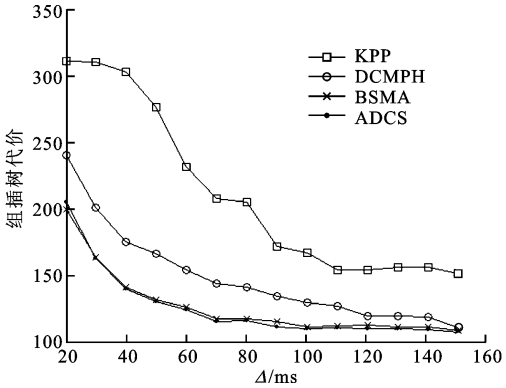


图 2 时延约束变化时的组播树代价

**实验 2** 比较组规模变化时各种算法生成的组播树代价，网络规模固定为 100，组播组目的节点数在 20～80 之间变化，时延约束为 70 ms，仿真结果如图 3 所示。可以看出，随着组播组目的节点数的增加，组播树进行扩展，各算法生成的组播树代价不断增加。组规模较小时，ADCS 算法与 BSMA 算法相差无几，随着目的节点数目的增加，可共享的链路增加，ADCS 算法生成的组播数代价略低于 BSMA 算法。ADCS 算法的总体平均组播树代价比 BSMA 算法低 1.5%。

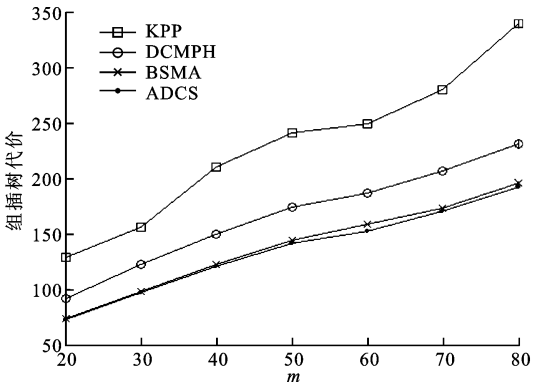


图 3 组规模变化时的组播树代价

**实验 3** 比较网络规模变化时各种算法生成的组播树代价，组播组目的节点数保持 20 不变，网络规模由 50 开始，每次增加 10 个，时延约束为 70 ms，仿真结果如图 4 所示。可以看出，随着网络规模的增加，目的节点到源节点的可行路径增加，各算法生成的组播树代价逐渐降低，最后由于时延约束限制而趋于稳定。ADCS 算法的总体平均组播树代价比 BSMA 算法低 1.3%。

### 3.2 运行时间

由于 KPP、BSMA 算法运行时间相对过高，图 5、图 6、图 7 集中比较了实验 1、实验 2、实验 3 中 ADCS 算法和 DCMPH 算法的运行时间。实验结

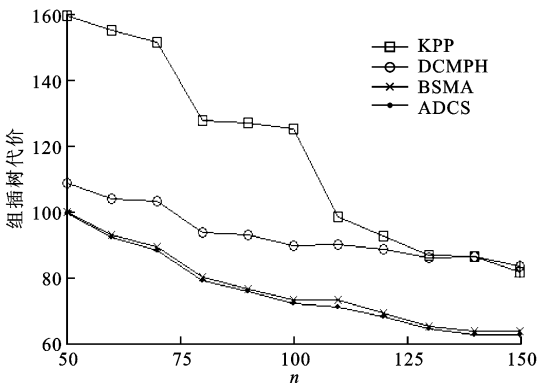


图 4 网络规模变化时的组播树代价

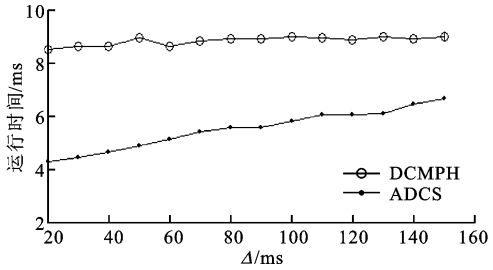


图 5 时延约束变化时的运行时间

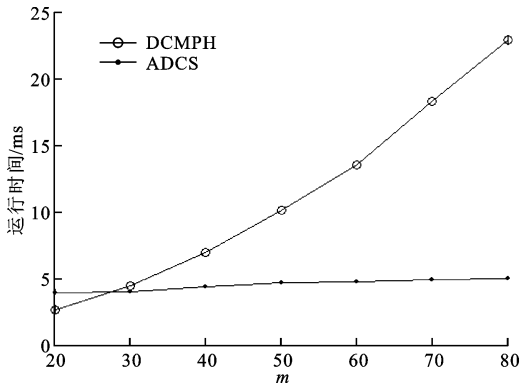


图 6 组规模变化时的组播树代价

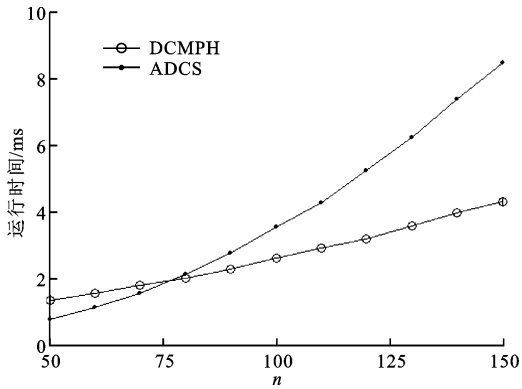


图 7 网络规模变化时的运行时间

果表明 ADCS 算法运行时间与组播组目的节点数无关，并随着时延约束、组播组规模的增加而增加，而且与 DCMPH 算法运行时间相当。虽然计算时间与计算环境等许多因素相关，但仿真结果初步验证了理论上对 ADCS 算法时间复杂度的分析。

## 4　总　结

本文提出了一种改进的时延约束 Steiner 树算法——ADCS。该算法在快速搜索阶段借鉴 Dijkstra 算法路径递增的基本思想改进节点搜索过程，降低了时间复杂度，采用链路共享的方法依次将到当前树有最小可行代价的目的节点通过最小可行代价路径加入组播树；在异常处理阶段将遗漏的目的节点通过最小时延路径加入组播树，进而生成满足时延约束的 Steiner 树。本文从理论上分析了算法的正确性和时间复杂度，通过随机网络模型的仿真结果表明，与典型的同类算法相比，ADCS 算法计算速度相对较快，且树代价较小。在下一步工作中，我们将致力于研究生成适用于大规模实时媒体流分发的组播树链路带宽容量的影响。

**参考文献：**


[1] DEERING S E, CHERITON D R. Multicast routing in a datagram internetworks and extended LANs [J]. ACM Transactions on Computer Systems, 1990,8(2): 85-110.

[2] SHI S Y, TURNER J S. Multicast routing and bandwidth dimensioning in overlay networks [J]. IEEE Journal on Selected Areas in Communications, 2002, 20(8):1444-1455.

[3] BANERJEE S, KOMMAREDDY C, KAR K, et al. Construction of an efficient overlay multicast infrastructure for real-time applications [C]// IEEE International Conference on Computer Communications. Piscataway, NJ, USA: IEEE, 2003:1521-1531.

[4] WINTER P. Steiner problem in networks: a survey [J]. Networks, 1987,17(2):129-167.

[5] HWANG F K, RICHARDS D S. Steiner tree problems [J]. Networks, 1992,22(1):55-89.

[6] KOMPELLA C S, PASQUALE J C, POLYZOS G C. Multicasting for multimedia applications [C]// IEEE International Conference on Computer Communications. Piscataway, NJ, USA: IEEE, 1992:2078-2085.

[7] ZHU Q, PARSA M, GARCIA-LUNA-ACEVES J J. A source-based algorithm for delay-constrained minimum-cost multicasting [C]// IEEE International Conference on Computer Communications. Piscataway, NJ, USA: IEEE, 1995:377-385.

[8] SALAMA H F, REEVES D S, VINIOTIS Y. Evaluation of multicast routing algorithms for real-time communication on high-speed networks [J]. IEEE Journal on Selected Areas in Communications, 1997,15(3): 332-345.

[9] 周灵，孙亚民. 基于 MPH 的时延约束 Steiner 树算法[J]. 计算机研究与发展, 2008,45(5):810-816.
ZHOU Ling, SUN Yamin. A delay-constrained Steiner tree algorithm using MPH [J]. Journal of Computer Research and Development, 2008, 45(5): 810-816.

[10] CHEN Shigang, SONG M, SAHNI S. Two techniques for fast computation of constrained shortest paths [J]. IEEE/ACM Transactions on Networking, 2008, 16(1):105-115.

[11] WAXMAN B M. Routing of multipoint connections [J]. IEEE Journal on Selected Areas in Communications, 1988, 6(9): 1617 -1622.


**[本刊相关文献链接]**


汪志伟，曹建福，郑辑光. 一种面向分簇无线传感器网络的多信道跨层协议. 2013, 47(6): 61-67.[doi:10.7652/xjtuxb201306011]

李琳琪，杨新宇，刘蔚. 高动态网络的自决策式地理机会主义路由. 2013, 47(2):7-13. [doi:10.7652/xjtuxb201302 002]

夏秦，王志文，卢柯. 入侵检测系统利用信息熵检测网络攻击的方法, 2013, 47(2): 14-19. [doi: 10.7652/xjtuxb201302003]

温彦，刘晨，韩燕波. 一种用户主导的跨组织数据按需集成方法, 2013, 47(2): 116-123. [doi: 10.7652/xjtuxb201302020]

张赛，徐恪，李海涛. 微博类社交网络中信息传播的测量与分析, 2013, 47(2): 124-130. [doi: 10.7652/xjtuxb201302021]

陈国强，王宇平. 采用离散粒子群算法的复杂网络重叠社团检测, 2013, 47(1): 107-113. [doi: 10.7652/xjtuxb201301021]

盖炳帅，王劲林，刘学. 采用 Petri 网的业务性能分析方. 2012, 46(12):12-17. [doi:10.7652/xjtuxb201212003]

伍文，孟相如，马志强，等. 模块化动态博弈的网络可生存性态势跟踪方法. 2012, 46(12): 18-23. [doi:10.7652/xjtuxb201212004]


（编辑　刘杨　武红江）