基于权重约束的最大密度路径改进算法

刘坤良 张大坤 武继刚
(天津工业大学计算机科学与软件学院 天津 300387)

摘 要 给定一棵树,树上的每个节点被赋予一对数值,它们分别表示节点的值和权重。基于权重约束的最大密度路径算法用于搜索树上的最大密度路径,即最大密度路径上所有节点的值之和与节点的权重之和的比值是所有路径中最大的。通过研究发现,现有的基于权重约束的最大密度路径算法有一定的局限性。文中提出了突破该局限性的可行性方案,进而设计并改进了基于权重约束的最大密度路径算法。

关键词 最大密度路径,最大密度子树,动态规划
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.08.027

Improved Algorithm for Finding Weight-constrained Maximum-density Path

LIU Kun-liang ZHANG Da-kun WU Ji-gang
(School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China)

Abstract A tree was given in which each node is associated with a pair of numbers that represent the value and the weight of the node. The weight constrained maximum-density path algorithm is to find the maximum-density path in the tree, that is the ratio of the summation of the nodes' values to the summation of the nodes' weights is maximum. After studing the current algorithm, we found that the maximum density path algorithm has limitations. We gave the way of bypassing the limitations. Finally we designed and improved the algorithm.

Keywords Maximum-density path, Maximum-density subtree, Dynamic programming

1 引言

给定一棵具有 n 个节点的树 $T=(V,E)$,树上的每个节点被赋予一对数值(val_v, w_v),分别表示节点的值和权重,其中 val_v 为一实数,表示节点 v 的数值大小;w_v 为一正整数,表示节点 v 的权重。假设用 $D(T)$表示树 T 的密度,则 $D(T)=\frac{\sum_{v\in V} val_v}{\sum_{v\in V} w_v}$。假设用 $w(T)$表示树 T 的权重之和,则 $w(T)=\sum_{v\in V} w_v$。树 T 的子树 $T'=(V',E')$为树 T 的一个连通子图,其中 $V'\subseteq V, E'\in E$。给定一权重范围(w_{min}, w_{max}),基于权重约束的最大密度子树算法(MDT)用于搜索满足条件 $w_{min}\leqslant w(T')\leqslant w_{max}$的最大密度子树 $T'=(V',E')$。如果子树被限定为节点之间的路径,MDT 问题就变成了基于权重约束的最大路径(MDP)问题。MDP 算法在计算生物学领域有着广泛的应用,比如基因序列比对。MDT 算法在计算机网络、交通网络及物流网络设计中应用广泛。比如在交通网络设计中,怎样在有限的预算之下改造交通流量最高的路段。Hoong Chuin Laiu[2]在树上的每条边赋予确定的权重和长度的情况下,给出了基于长度约束的最大密度路径算法。该算法的时间复杂度为 $O(n\log^2 n)$,当所有边的长度相同的情况下,时间复杂度可以降为 $O(n\log n)$。以此为基础,又实现了基于长度约束的最大密度子树算法,算法的时间复杂度为 $O(nl^2)$,其中 n 为树的节点个数,l 为路径或子树的最短约束长度。Hsin-Hao Su[3]提出了改进的基于长度约束的最大子树算法,该算法在树上所有边的长度都是正整数的情况下,时间复杂度为 $O(nu\log n)$,边的长度一致的情况下,该算法的时间复杂度可以降为 $O(nl\log\frac{n}{l})$,其中,u 为子树的最大约束长度。Sun-Yuan Hsieh[1]在树上的每个节点给定权重和节点值的情况下,设计了基于权重约束的最大密度路径(MDP)和基于权重约束的最大密度子树(MDT)算法,算法的时间复杂度分别为 $O(w_{max}n)$和 $O(w_{max}^2 n)$。其中 w_{max}为最大约束权重值,n 为树上节点的个数。在把节点对应的值和权重改为赋值给边的情况下,同样可以解决边约束的问题。我们通过研究发现该算法有实现上的局限性,如果按照算法本身的设计去实现并不能找到有效的最大密度路径或最大密度子树。本文对此问题进行研究,并作了改进。改进后的算法能有效地找到最大密度路径或最大密度子树,平均情况下的时间复杂度优于原算法,最坏情况的时间复杂度与原算法相同。

2 问题描述

在一棵具有根节点 r 的树 T 上,任一节点 v 的父节点用 $par(v)$表示,其子节点用 $child_T(v)$表示,节点的度用 $\deg(v)$表示,本文中节点的度是指该树上节点 v 的子节点的个数,节点的层用 $level(v)$表示。节点 v 的所有子孙节点组成的以 v 为根节点的子树用 T_v 表示。一个节点不能作为子树处理。对任一节点 v,从其出发的路径有两个方向:一个是向其孩子

到稿日期:2013-06-12 返修日期:2013-07-24
刘坤良(1976-),男,硕士,讲师,主要研究方向为算法设计与分析、三维可视化,E-mail:2003_asecliu@126.com。

节点延伸的路径，另一个是向父节点方向延伸的路径。

定义 1（基于权重约束的最大密度路径算法） 给定一个具有 n 个节点的树 T，每个节点被赋予一数值对 (val_v, w_v)，并且给定一权值范围 (w_{min}, w_{max})，搜索满足条件 $w_{min} \leqslant w(T') \leqslant w_{max}$ 的最大密度路径。

定义 2（基于权重约束的最大密度子树算法） 给定一个具有 n 个节点的树 T，每个节点被赋予一数值对 (val_v, w_v)，并且给定一权值范围 (w_{min}, w_{max})，搜索满足条件 $w_{min} \leqslant w(T') \leqslant w_{max}$ 的最大密度子树。

基于权重约束的最大密度路径算法[1]

带根节点的树 T 中的任一节点 v，我们用 $D_v^1[i]$ 和 $D_v^2[i]$ 分别表示从节点 v 出发向子节点延伸的所有权重值之和为 i 的路径中的数值之和的最大值和次大值。用 $U_v[i]$ 表示从节点 v 出发向父节点方向延伸的所有权重之和为 i 的路径中的最大值。

利用自底向上的动态规划技术，我们可以计算出树上每个节点的 $D_v^1[i]$ 和 $D_v^2[i]$，其中 $i = w_v, w_v+1, \cdots, w_{max}$。假设内部节点 v 有 m 个子节点 $v_1, v_2, \cdots, v_m$，则 $D_v^1[i]$ 和 $D_v^2[i]$ 的计算公式为：

$$D_v^1[i] = \max\{D_{v_j}^1[i-w_v] + val_v \mid 1 \leqslant j \leqslant m\}$$

$$D_v^2[i] = 2nd\_\max\{D_{v_j}^1[i-w_v] + val_v \mid 1 \leqslant j \leqslant m\}$$

利用自上而下的动态规划技术，我们可以计算出树上每个节点的 $U_v[i]$，其中 $i = w_v, w_v+1, \cdots, w_{max}$。$U_v[i]$ 的计算公式为：

$$U_v[i] = \max\begin{cases} U_{par(v)}[i-w_v] + val_v \\ D_{par(v)}^1[i-w_v] + val_v \\ D_{par(v)}^2[i-w_v] + val_v \end{cases}$$

如果 v 的值不包含在 $D_{par(u)}^1[i-w_v]$ 中，则 $D_{par(u)}^1[i-w_v]$ 的值有效，否则无效；如果 v 的值包含在 $D_{par(u)}^1[i-w_v]$ 中并且 $D_{par(v)}^2[i-w_v]$ 存在，则 $D_{par(v)}^2[i-w_v]$ 的值有效。

基于权重约束的最大密度路径算法框架如下所示：

```
Algorithm MDP(T, r, w_min, w_max)
1. computer the levels of all v∈V(T)
2. for all v∈V(T) in non-increasing order of the levels do
3.   if v is a leaf then
4.     D_v^1[w_v]←val_v
5.   else
6.     for i←w_v to w_max do
7.       D_v^1[i]←max{D_{v_j}^1[i−w_v]+val_v | v_j is a child of v}
8.       D_v^2[i]←2nd−max{D_{v_j}^1[i−w_v]+val_v | v_j is a child of v}
9.       computer d_v^1[i] and d_v^2[i]
10. for all v∈V(T) in increasing order of the levels do
11.    if v is the root then
12.      U_v[w_v]←val_v
13.    else
14.      for i←w_v to w_max do
15.        if v is not the contributor of D_{par(v)}^1[i−w_v] then
16.          U_v[i]←max{U_par(v)[i−w_v]+val_v, D_par(v)^1[i−w_v]+
                 val_v}
17.          computer u_v[i]
18.        else
19.          U_v[i]←max{U_par(v)[i−w_v]+val_v, D_par(v)^2[i−w_v]+
                 val_v}
20.          computer u_v[i]
21. return max_{v∈V(T)} { max_{w_min≤i≤w_max, i≠0} { D_v^1[i]/i, U_v[i]/i } }
```

其中，如果节点 v 是叶子节点并且 $i = w_v$，那么 $D_v^1[i] = val_v$，其他情况下取 $-\infty$。所有节点的 $D_v^2[i] = -\infty$，$0 < i \leqslant w_{max}$。该算法的时间复杂度为 $O(w_{max}n)$。

2.1 MDP 算法分析

通过研究上述算法的运行过程，我们发现非叶子节点的 $D_v^1[i]$、$D_v^2[i]$、$U_v[i]$ 在没有赋值的情况下取默认值 $-\infty$ 是不恰当的，算法的运行结果并不能得到最大密度路径，如图 1 所示。非叶子节点应该是和叶子节点一样 $D_v^1[w_v] = val_v$，$U_v[w_v] = val_v$，其他情况下取默认值 $-\infty$。而且从算法第 6－8 行和 14－19 行我们可以看出，对树上的内部节点来说，从节点的权重值 w_v 一直到 w_{max} 之间的 $D_v^1[i]$、$D_v^2[i]$ 及 $U_v[i]$ 都要进行比较、计算取最大值，根本没有考虑推导公式中权重值 $i - w_v$ 是否存在，从而造成计算时间上的浪费。这就需要优化原算法的实现过程，避免无效计算。

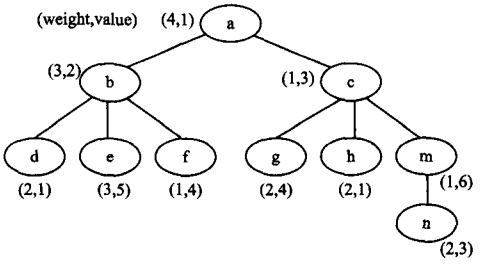

图 1 MDP 算法分析结果图

例 1 如图 1 所示，显而易见如果给定权重值范围为 (1, 7)，最大密度路径应该为节点 $c-m$ 所形成的路径，最大密度路径的权重之和为路径上节点 c 和 m 的权重之和 2，值为 c 和 m 的值之和 9，其路径密度值为 $(3+6)/(1+1) = 4.5$。但是根据上述算法的计算过程，可以得出图 2 所示的结果，可以看出其并没有找到路径 $c-m$。对于节点 c，在给定权重值范围为 (1, 7) 的情况下，$D_c^1[i]$ 中 i 的所有可能有效取值只能为 1、2、3、4，而当 i 的值为 5、6、7 时，$D_c^1[i]$ 根本没有有效数值，所以这时 MDP 算法中第 7、8 行就存在冗余计算。$U_v[i]$ 的计算也存在同样问题。

$$D_m^1[1] = \max\{D_n^1[1-1]+6\} = -\infty$$

$$D^1 c_[1] = \max\{D_{v_j}^1[1-1]+3 \mid v_j \in (g,h,m)\} = -\infty$$

$$D_c^1[2] = \max\{D_{v_j}^1[2-1]+3 \mid v_j \in (g,h,m)\} = -\infty$$

$$U_c[2] = \max\{U_a[1-1]+3, D_a^1[1-1]+2\} = -\infty$$

$$U_m[1] = \max\{U_c[2-1]+6, D_c[2-1]\} = -\infty$$

图 2 改进前 MDP 算法部分计算过程

2.2 MDP 算法的改进算法 MDP-I 算法

对树上的每个节点 v，我们附加一个线性表 $wList_v$ 来存储该节点上所有 $D_v^1[i]$ 为有效数值时所对应的权重 i。如果节点 v 有父节点，则对 $wList_v$ 中的每个权重值 e：

$$D_{par(v)}^1[e+w_v] = \max\{D_{par(v)}^1[e+w_v], D_v^1[e]+val_v\}$$

其中，$e+w_v \leqslant w_{max}$，$e \in wList_v$，$wList_{par(v)} = wList_{par(v)} \cup (e+w_v)$，如果 $D_{par(v)}^1[e+w_v] > D_v^1[e]+val_v$，则有计算公式：

$$D_{par(v)}^2[e+w_v] = \max\{D_{par(v)}^2[e+w_v], D_v^1[e]+val_v\}$$

其中，$e+w_v \leqslant w_{max}$，$e \in wList_v$。

另外再附加一个线性表 $wList2_v$ 来存储该节点上所有 $U_v^1[i]$ 为有效数值时所对应的权重 i。如果节点 v 有子节点，

则对 $wList2_v$ 中的每个权重值 e 及对应的每个子节点 v_j：

$$U_{v_j}[e+w_{v_j}]=\max\begin{cases}U_v[e]+val_{v_j}\\ D_v^1[e]+val_{v_j}\\ D_v^2[e]+val_{v_j}\end{cases}$$

如果 $e+w_{v_j}\notin wList_v$，则 $D_v^1[e]+val_{v_j}$ 有效，否则无效。如果 $e+w_{v_j}\in wList_v$ 并且 $e+w_{v_j}\in wList2_v$，$D_v^2[e]+val_{v_j}$ 有效，否则无效。

基于权重约束的最大密度路径改进算法如下所示：

```
Algorithm MDP-I(T, r, w_min, w_max)
1. compute the level of all v∈V(T)
2. for all v∈V(T) in non-increasing order of the level do
3.     D_v^1[w_v]=val_v
4.     wList_v=wList_v ∪ w_v
5.   for i←0 to wList_v.length()−1 do
6.     if D^1_par(v)[w_par(v)]+val_v>D^1_par(v)[w_par(v)+wList_v[i]]
7.       D^1_par(v)[w_par(v)+wList_v[i]]←max{D^1_par(v)[w_par(v)+wList_v
         [i]], D^1_par(v)[w_par(v)]+val_v}
8.       wList_par(v)=wList_par(v) ∪ w_par(v)+wList_v[i]
9.     else
10.      D^2_par(v)[w_par(v)+wList_v[i]]←max{D^2_par(v)[w_par(v)+wList_v
         [i]], D^1_par(v)[w_par(v)]+val_v}
11.    compute d^1_par(v)[w_par(v)+wList_v[i]] and d^2_par(v)[w_par(v)+wL-
       ist_v[i]]
12. for all v∈ V(T) in increasing order of the levels do
13.    U_v[w_v]←val_v
14.    wList 2_v=wList 2_v ∪ w_v
15.  for i←0 to wList 2_v.length()−1 do
16.      for all v_j∈child_T(v) do
17.        if wList 2[i]+w_vj ∉ wList_vj
18.          U_vj[wList 2[i]+w_vj]=max{U_vj[wList 2[i]+w_vj],
             D_v^1[wList 2[i]+w_vj]}
19.        else
20.          U_vj[wList 2[i]+w_vj]=max{U_vj[wList 2[i]+w_vj],
             D_v^2[wList 2[i]+w_vj]}
21.          compute u_vj[wList2[i]+w_vj]
22.  return max_{v∈V(T)} { max_{i∈wList_v ∪ i∈wList 2_v} { D_v^1[i]/i , U_v[i]/i } }
```

由于 MDP-I 通过对每个节点附加两个线性列表，分别记录节点向下路径和向上路径中值为有效值的所有权重，避免了为无效值时的冗余计算，故减少了计算量，提高了计算速度。比如，同样对于图 1 中的节点 m，$D_m^1[i]$（其中 $i\leqslant w_{max}$）的值是由其子节点的有效值计算得到，由于节点 m 只有一个子节点，子节点只有一个有效的 $D_n^1[i]$ 值，即 $i=2$ 时，故 m 的所有有效值通过一次计算就可以得到。而 MDP 算法中 $D_m^1[i]$ 的计算需要 i 从 1 到 7 每个值都要判断一次，其中有 6 次是无效计算。

3 算法实现结果

本文以 Visual C++6.0 为开发工具，随机生成了一棵节点具有权重和值的树，如图 3 所示，其中节点内的数字为节点的权重和值，节点左上角的字母为节点的名称，如树的根节点的名字为 a，权重为 3，值为 20。算法改进前如果给定权重值范围为(10,20)，对该树计算最大密度路径的结果是：最大密度路径的权重和为 16，密度为 4.8，路径为 e-b-a-d-f-i。算法改进后的运行结果是：最大密度路径的权重为 10，密度为 6.7，路径为 e-b-a-d。显然在给定权重范围为(10,20)的情况下最大密度路径应该为 e-b-a-d。

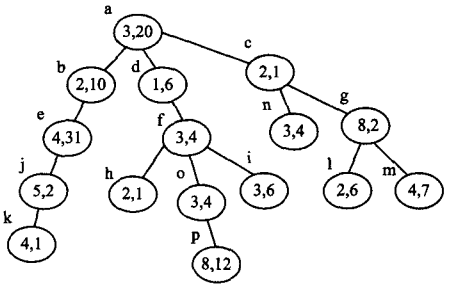

图 3 节点带权重和值的随机树

结束语 本文中，根据权重约束的最大密度路径算法的计算特点，发现现有的基于权重约束的最大密度路径算法有一定的局限性，其非叶子节点的 $D_v^1[i]$、$D_v^2[i]$、$U_v[i]$ 在没有赋值的情况下取默认值 $-\infty$ 是不恰当的，算法的运行结果并不能得到最大密度路径，非叶子节点应该是和叶子节点一样 $D_v^1[w_v]=val_v$，$U_v[w_v]=val_v$，其他情况下可以取默认值 $-\infty$。在此基础上又提出了改进的最大密度路径算法，改进后的算法相对原算法弥补了可能出现的找不到最优结果的缺陷，减少了冗余计算，提高了计算速度。接下来的工作，将在降低该算法时间复杂度的基础上，进一步研究如何降低其空间复杂度。

参 考 文 献

[1] Hsieh Sun-yuan, Chou Ting-yu. The weight-constrained maximum-density subtree problem and related problems in trees[J]. The Journal of Supercomputing, 2010, 54(3): 366-380

[2] Lau H C, Ngo T H, Nguyen B N. Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics[J]. Discrete Optimization, 2006(3): 385-391

[3] Su H-H, Lu C L, Tang Chuan-yi. An improved algorithm for finding a length-constrained maximum-density subtree in a tree [J]. Information Processing Letters, 2008(109): 161-164

[4] Lin Rung-ren, Kuo W-H, Chao Kun-mao. Finding a Length-Constrained Maximum-Density Path in a Tree[J]. Journal of Combinatorial Optimization, 2005, 9: 147-156

[5] Hsieh S-Y, Cheng C-S. Find a maximum-density path in a tree under the weight and length constrains[J]. Information Processing Letters, 2008(105): 202-205

[6] Lin R R, Kuo W H, Chao K M. Finding a length-constrained maximum-density path in a tree[J]. Journal of Combinatorial Optimization, 2005, 9(2): 147-156

[7] Chao K M. On computing all suboptimal alignments[J]. Information Sciences, 1998, 105: 189-207

[8] Chao K M, Hardison R C, Miller W. Recent developments in linear-space alignment methods: A survey[J]. Journal of Computational Biology, 1994, 1: 271-291

[9] Lin Y L, Jiang T, Chao K M. Algorithms for locating the length-constrained heaviest segments, with application to biomolecular sequences analysis[J]. Journal of Computer and System Sciences, 2002, 65(3): 570-586

[10] Chung K M, Lu H I. An optimal algorithm for the maximum-density segment problem[J]. SIAM Journal on computing, 2004 (34): 373-387