文章编号:1671 - 6833(2012)05 - 0125 - 05

基于点割集的并行最短路径算法

张清华[1,2], 李 鸿[1], 沈 文[1]

(1. 重庆邮电大学 计算机科学与技术研究所, 重庆 400065; 2. 重庆邮电大学 数理学院, 重庆 400065)

摘 要: 在研究和分析了 Dijkstra 算法的基础上,在 Dijkstra 算法中通过引入点割集和割点的思想来改进 Dijkstra 算法,该方法首先利用点割集或割点把原问题分解成多个子图,然后对每个子图并行求最短路径,最后通过点割集或割点求出整个原问题的最短路径,从而降低算法的时间复杂度,提高算法的效率.

关键词: 割点;最短路径算法;Dijkstra 算法;并行计算;粒计算

中图分类号: TP301.6 **文献标志码**: A doi:10.3969/j.issn.1671 - 6833.2012.05.028

0 引 言

在 18 世纪 30 年代,一个非常有趣的问题引起了欧洲数学家的浓厚兴趣,这个问题要求遍历普鲁士的哥尼斯堡七桥中的每一个座桥恰好一次后回到出发点. 瑞士著名的数学家欧拉证明了这是不可能完成的. 此后,欧拉发表了著名的论文《依据几何位置的解题方法》,这是图论领域的第一篇论文,标志着图论的诞生[1]. 图论与其它的数学分支不同. 它不像群论、拓扑等学科那样有一套完整的理论体系和解决问题的系统方法. 最短路问题是重要的最优化问题之一,也是图论研究中一个经典算法问题,它不仅直接应用于解决生产实践中的众多问题,而且也经常被作为一种基本工具,用于解决其他的最优化问题以及预测和决策问题.

作者主要讨论了最短路径算法(Dijkstra 算法),并将割点和点割集引入 Dijkstra 算法,然后把两者有机地结合起来形成一种改进算法,即运用粒计算[2]的思想,把原问题分解成许多小规模的子问题,并且子问题之间互不相关,则可并行计算子问题,从而降低算法的计算复杂度.

1 经典 Dijkstra 算法的主要思想

1.1 相关定义

定义 1[3] 边上有权重的图称为加权图. 若边 e 标记数 k,称边 e 的权为 k. . 在加权图中,链(迹、路)的长度为链(迹、路)上的所有边的权值的和.

在加权图中,我们经常需要找出两个指定点之间的最短路(如有最小长度的路),通常称其为最短路问题. 其中,公认解决这类问题比较典型的算法则是 Dijkstra 算法.

单源点最短路径的一个著名算法就是 Dijkstra 算法,它用于计算一个节点到其他所有节点的最短路径. 主要特点是以起始点为中心向外层层扩展,直到扩展到终点为止. Dijkstra 算法是求出一个连通加权简单图中从结点 a 到 z 的最短路. 边 $\{i,j\}$ 的权 $w(i,j)>0$,且结点 x 的标号为 $L(x)$. 结束时,$L(z)$ 是从 a 到 z 的最短路的长度. 该算法能得出最短路径的最优解,但由于它遍历计算的节点很多,所以效率低.

1.2 算法描述

Dijkstra 算法解决了有向图 $G=(V,E)$ 上带权的单源最短路径问题,但要求所有边的权值非负. 这个算法是通过为每个顶点 v 保留目前为止所找到的从 s 到 v 的最短路径来工作的. 初始时,原点 s 的路径长度值被赋为 0($d[s]=0$),同时把所有其他顶点的路径长度设为无穷大,即表示我们不知道任何通向这些顶点的路径(对于 V 中所有顶点 v 除 s 外 $d[v]=\infty$). 当算法结束时,$d[v]$ 中储存的便是从 s 到 v 的最短路径,或者如果路径不存在的话

收稿日期:2012 - 05 - 20;**修订日期**:2012 - 07 - 01

基金项目:重庆市教委科学技术研究项目(KJ110512)和重庆市教委教改项目(No. 103161)资助;重庆邮电大学研究生教育创新计划资助项目(Y201110).

作者简介:张清华(1974 -),男,重庆邮电大学副教授,博士,硕士生导师,主要从事图论教学及其教学研究.

是无穷大. 算法维护两个顶点集 S 和 Q. 集合 S 保留了我们已知的所有 $d[v]$ 的值已经是最短路径的值顶点,而集合 Q 则保留其他所有顶点. 集合 S 初始状态为空,而后每一步都有一个顶点从 Q 移动到 S. 这个被选择的顶点是 Q 中拥有最小的 $d[u]$ 值的顶点. 当一个顶点 u 从 Q 中转移到了 S 中,算法对每条外接边 (u, v) 进行拓展.

作者用大 O 符号将该算法的运行时间表示为边数 m 和顶点数 n 的函数. Dijkstra 算法最简单的实现方法是用一个链表或者数组来存储所有顶点的集合 Q,所以搜索 Q 中最小元素的运算只需要线性搜索 Q 中的所有元素. 这样的话算法的运行时间是 $O(n^2)$. 对于边数少于 n^2 的稀疏图来说,我们可以用邻接表来更有效的实现该算法.

2 割点与点割集

定义 2(割点)[3] 如果在图 G 中删去一个结点 u 后,图 G 的连通分枝数增加,即 $W(G-u) > W(G)$,则称结点 u 为 G 的割点,又称关节点. 没有割点的非凡连通图称为块. G 中不含割点的极大连通子图称为图 G 的块.

定义 3(点割集)[3-4] 如果图 G 的顶点集的一个真子集 T 满足 $W(G-T) > W(G)$ 或 $G-T$ 平凡图,且对于任意的 S,如果满足 S 是 T 的真子集,则 $W(G-S) = W(G)$,则称 T 为 G 的一个点割集.

定义 4(连通度)[3,5] 设 G 是连通图,称 $K(G) = \min\{|T| \mid T$ 是 G 的点割$\}$ 为 G 的点连通度或连通度.

由深度优先生成树可得出两类关节点的特性[6-7]:

(1)若生成树的根有两棵或两棵以上的子树,则此根顶点必为关节点.

(2)若生成树中某个非叶子节点 V,其某棵子树与 V 的祖先节点无连接,则 V 为关节点.

用深度优先搜索算法求出图的全部割点,再进行分块,即可达到简化问题的效果.

定理 1[8] 在深度优先搜索过程中,若 $e = uv$ 是父子边,且 $k(u) > 1$,$l(v) \geq k(u)$,则 u 是割点.

定理 2[8] 若 r 是深度优先搜索过程中生成的树的根,则 r 是图 G 割点的充分必要条件是至少有两条以 r 为尾的父子边.

3 基于割点的 Dijkstra 改进算法

3.1 相关定理证明

为了引出基于割点改进的最短路径算法,首先得解决下面这几个问题,才能证明算法的正确性和可行性,下面的性质是基于图 1 所示的图从 a 到 z 的最短路径问题.

性质 1 图 1 中从 a 到 z 的最短的路径一定经过割点.

证明 我们不妨使用反证法,首先假设最短路径不经过割点,从 a 到 z 有一条最短路径,我们不妨把其它没有经过的点全部删除点,这时把割点删除后,图就不连通了,则最短路径肯定无法找到,因此,这与假设矛盾,则最短的路径一定经过割点.

性质 2 图 1 中从 a(起始点)到 z(终点)的最短路径同从 a(起始点)到 f(割点)的最短路径和 f(割点)到 z(终点)的最短路径求和结果是等价的.

证明 为了方便讲解,我们不妨先构造一个简单的示意图(如图 1 所示),首先找到所求加权图 G 的所有割点(如点 f),由定理 1 可知,最短路径一定经过割点(点 f),我们不妨以割点为基准线,将图分成几个部分,显然,这里每个部分通过 Dijkstra 能求出最短路径,再以割点为衔接点,组成图 G 的最短路径,因此,这是等价的,证毕.

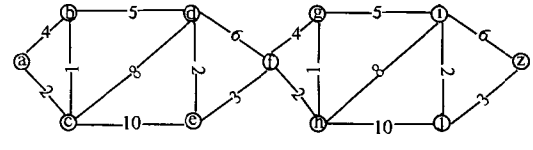

图 1 示意图 G

Fig. 1 Diagram G

性质 5 (最短路径的子路径是最短路径[6])对于一给定的带权有向图 $G = (V, E)$,所定义的权函数为 $w: E \to R$. 设 $p = <v_1, v_2, \cdots, v_k>$ 是从 v_1 到 v_k 的最短路径. 对于任意 i, j,其中 $1 \leq i \leq j \leq k$,设 $p_{ij} = <v_i, v_{i+1}, \cdots, v_j>$ 为 p 中从顶点 v_i 到顶点 v_j 的子路径. 那么,p_{ij} 是从 v_i 到 v_j 的最短路径.

性质 6 图 1 中从 a 到 z 最短路径一定经过点割集中的一个点.

证明 我们不妨使用反证法,首先假设最短路径不经过某个点割集,从 a 到 z 有一条最短路径,我们不妨把其它没有经过的点全部删除点,这时把点割集中所有的点删除后,图就不连通了,则最短路径肯定无法找到,因此,这与假设矛盾,则最短的路径一定经过点割集中某一个点.

3.2 基于割点的 Dijkstra 改进算法描述

Step1:使用深度优先算法求出所有的割点 C_i;

Step2：以割点为基准，分解成多个加权图 G_i；

Step3：对每个加权图 G_i 用 Dijkstra 算法求出最短路径 P_i；

Step4：将每个最短路径 P_i，以割点 C_i 为衔接点，合并成最短路径 P，P 即为所求.

图 2 为程序流程图：

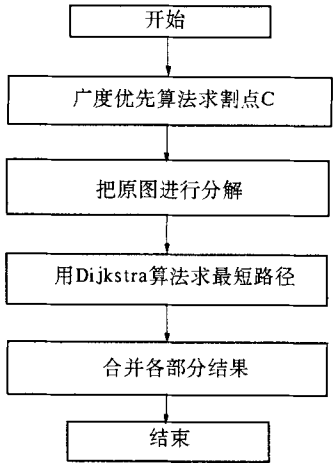

图 2　算法流程图

Fig. 2　Algorithm flowchart

3.3　算法伪代码

Procedure Cut_vertex Dijkstra

Select 割点 C_l

把原图以割点 C_l 分解成多个 G_k

for k: = 1 to ($|C_l|+1$)

for i: = 1 to n_k

$L(v_i)$: = ∞

$L(a_i)$: = 0

S: = ∅

{初始化标记，a 的标记为 0，其余结点标记为∞，S 是空集}

while $z_i \notin S$

begin

u: = 不属于 S 的 L(u)最小的一个顶点

S: = $S \cup \{u\}$

for 所有不属于 S 的顶点 v

f $L(u)+w(u,v)<L(v)$

then $L(v)$: = $L(u)+w(u,v)$

{这样就给 S 中添加带最小标记的顶点并且更新不在 S 中的顶点的标记}

end{$L(z_k)$ = 从 a 到 z_k 的最短路的长度}

再将所求全并

end{$L(z)$ = 从 a 到 z 的最短路的长度}

3.4　算法复杂度分析

现在来分析一下该算法的计算复杂度，首先通过前文知道 Dijkstra 算法的复杂度为 $O(n^2)$，因为通过割点把原图分解成了多个相同的小问题，我们不妨设有 k 个割点，结点最多子图有 n_{max} 个，因此它的计算复杂度为 $O(n_{max}^2)$. 从这个式子可以看出，计算复杂度与割点的位置有一定的关系，总之，不管怎么样，只要 $n_{max}<n$，则计算复杂度就降低，显然，只要有割点存在，上式就成立.

因为将图分成了小图进行同类问题求解也就是它们求解过程是可以同步操作，互不影响，则可以进行并行计算，从这一方面来说，也大大提高了计算的效率.

4　实验分析与结果讨论

为了更好地说明问题，下面将举一个例子来说明，通过 matlab 编写求割点[9-10]与最短路径的程序，其程序代码均可在文献[8]中找到，将它们组合起来，加入求解运行时间的代码.

我们不妨作如图 3 所示的加权图，此图根据文献[3]上的例子改变而来，一共有 11 个点，其中有一个割点（点 v_6）.

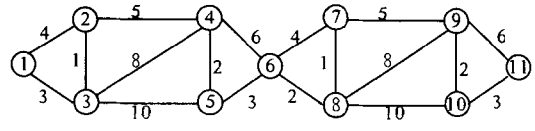

图 3　含有割点的图

Fig. 3　Contains a cut vertex of Figure

计算步骤：

Step1：找出加权图的割点，这里为 v_6；

Step2：以割点为基准，将图分成左右两部份，其中一部分 G_1 包含 v_1 至 $v_6$6 个点，另一部分 G_2 包含 v_6 至 v_{11}6 个点；

Step3：对 G_1、G_2 分别用 Dijkstra 算法求出最短路径；

Step4：将所求的最短路径 P_1 和 P_2，以割点 v_6 为衔接点，合并成一条最短路径 P.

运算结果如表 1 所示.

在实际情况中，数据往往较为复杂，因此割点可能不存在，如图 4 所示. 那么，我们可以采用点割来进行分解原图，根据定理 6，子图最后一定能用点割集中的某个点连接起来. 从图 4 不难观察出，点割集不是惟一，那么怎样寻找点割才是最好的呢？因此，在寻找点割集时，按照以下三大原则执行：

（1）尽量保证点割集合的大小也连通度相等；

表1 含有割点的加权图运行结果

Tab.1 Operating results of the weighted graph contains a cut vertex

最短距离	标号顺序	标号索引	运行时间 t_1	运行时间 t_2
0	1	1	0.015600s (Dijkstra算法)	0.007800s (基于割点的Dijkstra改进算法)
3	3	3		
2	2	1		
8	4	2		
10	5	4		
13	6	5		
16	8	8		
15	7	6		
21	9	7		
23	10	9		
26	11	10		

(注:标号索引是最短路径中该点的前一点的标号,如:v_1到v_2最短距离是3,标号索引是v_3,v_3的标号索引是v_1,则v_1至v_2的最短路径需经过v_3,因此$P_{12}=v_1\rightarrow v_3\rightarrow v_2$.)

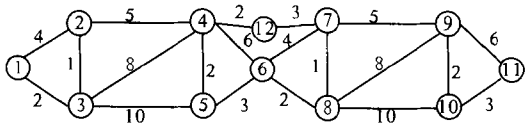

图4 不含割点的图

Fig.4 Not Contains a cut vertex of Figure

(2)通过点割集将图进行分块,尽量保证每个块点的个数相差最小;

(3)运用并行的思想,整体运行的时间就是最大块运行的时间.

计算步骤:Step1:因为找不到割点,则按以上三大原则寻找点割,此处为v_6和v_{12};

Step2:以点割为基准,将图分成左右两部份,其中一部分G_1包含v_1至v_6和v_{12}7个点,另一部分G_2包含v_6至v_{12}7个点;

Step3:对G_1、G_2分别用Dijkstra算法求出最短路径;

Step4:将所求的最短路径P_1和P_2,以点割集运算结果如表2所示.

5 结束语

笔者采用逐步细分的思想,把大问题通过一定的方法分解成多个小问题,从而降低计算的时间复杂度.但是,同时也引入了求割点带来的开销,因此,本方法只能在一些特定的条件下,发挥它的优势.总之,作者从一个粒计算求解复杂问题[2,11]的视角来看待问题,对Dijkstra算法进行了小小的改进.在复杂的实际问题中,可能不存在割点,我们就退而求其点割集,然而怎么寻找点割集,且让其满足本文所述的三大原则,这将是我们要继续研究的内容.

表2 不含有割点的加权图运行结果

Tab.2 Operating results of the weighted graph not contains a cut vertex

最短距离	标号顺序	标号索引	运行时间 t_1	运行时间 t_2
0	1	1	0.031200s (Dijkstra算法)	0.015600s (基于点割的Dijkstra改进算法)
3	3	3		
2	2	1		
8	4	2		
10	5	4		
13	12	5		
13	6	12		
14	7	7		
18	8	7		
20	9	9		
23	10	10		
10	11	4		

在实际分解子问题时,割点与点割集可结合起来使用,先用割点将原图分解成子图,若子图结点数仍然较大,可用点割来进一步分解,而且每个子问题属于同一类问题.因此,可以在互不影响的情况进行计算,即并行实现,则整体的运行时间就是最大块运行的时间.

显然,本文的方法应用于某一类型的加权图时,效率优于不改进的最短路径算法.那么,如何来制定一个判断的标准呢?

(1)当一个加权图有多个割点或者连通度较小时,则采用本文的算法能够很好地提高效率.

(2)点连通度大于多少时,进行分解计算效果不会明显优于不分解的情况.

(3)在求解点连通度以用最小点割集时,同时满足前面提到的三大原则也是一个难点.

致谢:本文是图论教学中对Dijkstra算法改进的一种尝试,感谢重庆邮电大学研究生创新计划(Y201110)的资助.

参考文献:

[1] 燕子宗,张宝琪.图论及其应用[J].重庆科技学院学报,2007,9(2):121-123.

[2] 张清华,周玉兰,滕海涛.基于粒计算的认知模型.重庆邮电大学学报,2009,21(4):494-501.

[3] 殷剑宏,吴开来.图论及其算法[M].合肥:中国科学技术大学出版社,2003:63-70.

[4] 方文其,胡明晓.无向关系图视觉清晰化显示算法[J].计算机工程与科学,2011,33(6):51-56.

[5] 康晓军,王茂才.最短路径问题的一种高效实现

[J].微计算机信息,2009,25(3):218 - 219.
[6] THOMAS H,CORMEN CHARLES E,LEISERSON,et al.算法导论[M].潘金贵译.北京:机械工业出版社,2006.9:366.
[7] 许光汉.用广度优先搜索求割点和块的算法研究[J].北京航空航天大学学报,1991(2):87 - 94.
[8] 王海英,黄强.图论算法及其 MATLAB 实现[M].北京:北京航空航天大学出版社,2012.
[9] LI Wing-ning. An efficient algorithm for computing a minimum node cutest from a vertex-disjoint path set for timing optimization[C]//ACM symposium on Applied Computing, Nashville, Tennessee, USA, Feb, 1995:56 - 60.
[10] 李克清,黄瑜岳.基于右手法则的网格割点判定算法[J].计算机与数字工程,2010,38(7):6 - 8.
[11] TAN Zhi-bin. Minimal cut sets of s-t networks with k-out-of-nodes[J]. Reliability Engineering and System Safety,2003,82(1):49 - 54.

Parallel Shortest Path Algorithm Based on Vertex Cut-set

ZHANG Qing-hua[1,2], LI Hong[1], SHEN Wen[1]

(1. Institute of Computer Science & Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China;2. College of Mathematics & Physics, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

Abstract: In this paper, research and analysis on the basis of the Dijkstra algorithm, the Dijkstra algorithm by introducing the vertex of cut sets and cut the vertex of idea to improve the Dijkstra algorithm, this method first vertex cut-set or cut vertex to the original problem is decomposed into multiple sub-graph, then the shortest path on each sub-graph parallel to the shortest path of the original problem, and finally obtained by the vertex cut-set or cut vertex, which reduces the time complexity of the algorithm and improves the efficiency of the algorithm.

Key words: cut vertex; shortest path algorithm; Dijkstra algorithm; parallel computing; granular computing

(上接第 99 页)

The Optimization Design Study Based the Lower Yellow River Prestressed Pipe Pile Spur Dikes Structure

ZHANG Bao-sen[1,2], WANG Zhong -Fu[3], TIAN Zhi-zong[1,2], XIE Zhi-gang[1,2]

(1. Yellow River Institute of Hydraulic Research Zhengzhou 450000, China;2. Research Center on Levee Safety & Disaster Prevention Ministry of Water Resources Zhengzhou 450000, China;3. North China University of Water Conservancy and Hydroelectric Power, Institute of Geotechnical and Hydraulic Engineering, Zhengzhou 450011, China)

Abstract: New prestressed pipe pile is presented on the basis of analysis of the characteristics of traditional rockfill groins. Take Huayuankou for example, the combination of the orthogonal experiment principle of the spur dikes structure design, the overall displacement law with the parameters change of spur dikes is analyzed by FLAC3D finite difference code. The numerical results show that spur dikes of the horizontal displacement of the contribution of primary and secondary order as follows: the scour depth of pile length > flow rate > pile spacing and pile diameter > contacts beam size; under the operation conditions of flood flow 4 000 m^3, water velocity of 3 m/s, the scour depth is 16m operating conditions, the spur dikes structure program of the pile length 25m 0.6m pile spacing, pile diameter is 600 mm, contact beam size is 0.6 m × 0.6 m is optimal, to the safety factor of strength reduction method is 1.32.

Key words: spur dike; orthogonal design; FLAC3D; prestressed pipe pile; scour depth