网络最短路径算法的改进及实现

李 峰, 张建中*
(厦门大学通信工程系, 福建 厦门 361005)

摘要: 从节约存储空间和提高运算速度方面对 Dijkstra 最短路径算法进行了改进. 定义新的节点类来高效存储网络的拓扑信息, 节省了计算机存储空间; 采用满二叉堆数据结构对节点进行排序并选取最短路径节点, 大大提高算法效率. 仿真例子表明, 对于某些网络结构, 改进算法能把传统 Dijkstra 算法的时间复杂度由原来的 $O(N^2)$ 近似降至 $O(N)$.

关键词: Dijkstra 算法; 邻接节点数组; 堆排序
中图分类号: TP 301.6 **文献标识码**: A **文章编号**: 0438-0479(2005)Sup-0236-03

不同领域的许多科技问题都可以归结为网络图论中的最短路径问题[1~4]. 传统的最短路径算法, 即 Dijkstra 算法针对弧权值非负的图结构, 求解图上从任一节点(源点)到其余各节点的最短路径. 其时间复杂度为 $O(N^2)$ (N 为节点总数); 采用邻接矩阵法存储网络图, 需要开辟 $N \times N$ 的存储空间, 当 N 较大时, 计算效率和存储效率都很低. 针对上述问题, 许多学者提出了改进措施. 例如, 王杰臣等提出了图的节点-弧段联合结构表示法, 以减少存储空间[5]; 王涛等通过改进节点的搜索过程, 提出了快速低代价最路径树算法[6]; 等等. 本文分析了 Dijkstra 算法存在的问题, 采用不同的数据存储和最短路径节点的搜寻方法, 提高了最短路径算法的性能.

1 Dijkstra 算法及问题分析

在求从网络中的某一节点(源点)到其余各节点的最短路径时, 经典 Dijkstra 算法将网络中的节点分成三部分: 未标记节点、临时标记节点和最短路径节点(永久标记节点). 算法开始时源点初始化为最短路径节点, 其余为未标记节点, 算法执行过程中, 每次从最短路径节点往相邻节点扩展, 非最短路径节点的相邻节点修改为临时标记节点, 判断权值是否更新后, 在所有临时标记节点中提取权值最小的节点, 修改为最短路径节点后作为下一次的扩展源, 再重复前面的步骤, 当所有节点都做过扩展源后算法结束. 具体算法描述如下:

设在一非负权简单连通无向图 $G = < V, E, W >$ (V: 顶点集, E: 边集, W: 边权值) 中, d 为图 G 的邻接矩阵, 求源点 p_0 到其余所有节点 p_i 的最短路径长度. (i) 将 V 分为未标记节点子集 N、临时最短路径节点子集 T 和最短路径节点子集 S, 每个节点上的路径权值为 $D(i)$. 初始化: $S = \{p_0\}$, $T =$, $N = V - S$, $D(0) = 0$, $D(i) = \infty$; (ii) 更新: 将新加入 S 集合的节点 p_s 作为扩展源, 计算从扩展源到相邻节点的路径值. 若该值比节点上的原值小, 则用该值替换原值, 否则保持原值不变, 即 $D(i) = \min\{D(s) + d[s][i], D(i)\}$, 并将这些相邻节点之中的未标记节点归为临时标记节点, 即 $T = T \cup p_i$, $N = N - p_i$; (iii) 选择: 在 T 中选择具有最小路径值 $D(s)$ 的节点 p_s, 归入集合 S 中, 即 $S = S \cup p_s$, $T = T - p_s$; (iv) 迭代判断: 若 $T =$, 算法结束, 否则转(ii).

在上述算法中, 图 G 的邻接矩阵 d 的存储量为 $N \times N$, 一般都是大型稀疏矩阵, 这将耗费大量资源, 存储那些无意义的矩阵元素. 另一方面, 在算法的每次迭代中, 由于 T 集合中的节点无序排列, 每次选择最短路径节点都必须访问 T 集合中所有的节点, 当节点数目很大时, 非常耗时, 成为应用 Dijkstra 算法的瓶颈.

2 Dijkstra 算法改进

针对 Dijkstra 算法存在的问题, 我们在网络数据的存储和最短路径节点的选取上, 进行了改进.

2.1 网络数据的存储

根据网络图的拓扑信息特征, 定义了新的节点类, 在节点类中用邻接节点数组存储相邻节点的所有信息, 数据文件保存为{节点编号; 相邻节点数目; 相邻节点编号集合; 相邻弧段权值集合}的格式. 该结构对不直接连通的节点之间的信息不予保存, 这样整个图只要 N 个节点类对象就可以完全存储, 从而节省了存储

收稿日期: 2005-03-09
基金项目: 福建省自然科学基金(D0310001)资助
作者简介: 李峰(1981-), 男, 硕士研究生.
* 通讯作者: zjz98@sina.com

空间. 同时, 在执行算法时, 可以随时直接获得某个节点的所有相邻节点的全部信息以及对应的弧信息, 并利用相邻节点数目来控制循环, 提高效率.

2.2　节点排序和最短路径节点选取

在 Dijkstra 算法中, 网络图中的每个节点均需实现从未标记节点到临时标记节点, 再从临时标记节点到最短路径节点的转变. 后种转变需要从大量的无序存储的临时标记节点中选取最短路径节点. 我们采用 Willioms 提出的堆排序方法, 将临时标记节点整理为一个最小二叉堆, 这样, 从未标记节点到临时标记节点的转变就被化为堆元素的插入和向上调整过程, 最短路径节点始终处于二叉堆的顶部; 从临时标记节点到最短路径节点的转变就成为堆中最顶部节点的简单移出. 由于每次操作均针对节点地址进行, 节省了空间; 寻找最短路径节点只需要访问二叉堆中的某一个分支, 大大提高了效率.

3　改进 Dijkstra 算法的实现

首先给出节点类和堆类中主要数据定义的 C++ 代码, 然后根据代码阐述改进 Dijkstra 算法的具体实现步骤.

3.1　类结构中主要的成员定义如下

```
class GraphStruct  {
public:
  int PointNo;  //节点编号
  int NearNum;  //相邻节点数目
  double Dist;  //节点上的最短路径值
  int HeapPosition;  //用于在二叉堆中定位
  GraphStruct** NearGraph;  //邻接节点数组
  double* NearArcValue;  //相邻弧段权值
};
```

3.2　堆结构中主要的成员定义如下

```
class MinHeap  {
public:
  GraphStruct** heapArr;  //建堆
  void FilterDown(GraphStruct* Point);  //向下调整
  void FilterUp(GraphStruct* Point);  //向上调整
  void Insert(GraphStruct* d);  //插入元素
  GraphStruct* DeleteTop();  //获取最短路径节点
  bool IsEmpty() const;  //判断堆是否为空
};
```

3.3　算法具体步骤

(i) 生成节点对象, 从数据文件读入节点编号、相邻节点、相邻弧段权值, 并初始化节点对象的其它成员信息;

(ii) 创建所需要的二叉堆, 为其分配足够空间;

(iii) 源节点相关数据初始化, 由当前源节点向相邻节点扩展.

1) 若为未标记节点则更新此节点的 Dist, 节点修改为临时标记节点, 插入到堆中, 调用 FilterUp 调整二叉堆, 修改相关节点上的 HeapPosition;

2) 若节点已经在堆中, 即为临时标记节点, 则判断是否需要更新 Dist, 若需要更新, 则修改节点的 Dist, 接着调用 FilterUp 调整二叉堆, 修改相关节点的 HeapPosition;

(iv) 移出堆中顶部的元素, 将弹出元素修改为最短路径节点, 作为下一次扩展的源点. 同时将堆尾元素移至顶部, 调用 FilterDown 调整二叉堆, 修改相关节点的 HeapPosition. 判断堆是否为空, 若不为空, 转步骤(iii); 若堆已空, 程序结束.

3.4　仿真例子

在研究地震波在介质中的传播时, 常常要确定波从源点至介质各点传播路径, 即最小旅行时路径. 若把彼此相邻的源点与节点和节点与节点两两相连, 就形

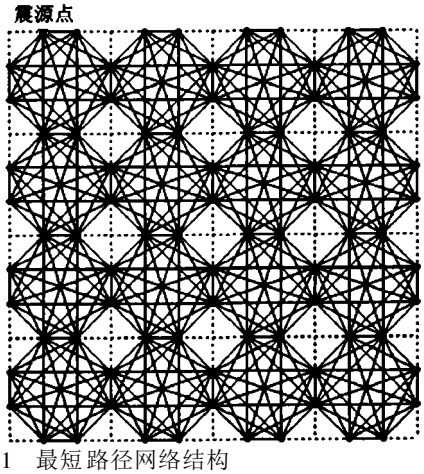

图 1　最短路径网络结构

Fig. 1　Shortest-path network structure

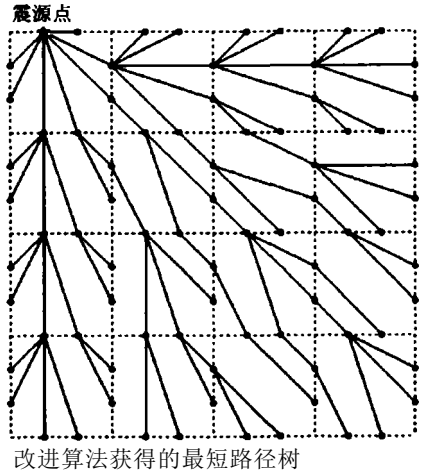

图 2　改进算法获得的最短路径树

Fig. 2　Shortest-path tree by improved algorithm

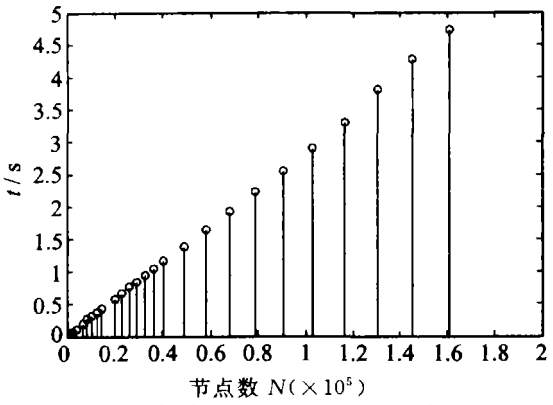

图 3 改进算法运行时间随节点数的变化

Fig. 3 Operation time versus node number using improved algorithm

成一个网络,当波在介质中的传播速度已知时,就可用网络最短路径算法求波从源点传至各节点的最小旅行时路径[4]. 如图 1 所示是一均匀介质中,震源点和规则分布的节点所形成的网络图;如图 2 所示是本文提出的改进 Dijkstra 算法求出的图 1 模型的震源点至每个节点上的最短路径树,可以看出,改进 Dijkstra 算法是正确的.

4 改进 Dijkstra 算法的时间复杂性分析

Dijkstra 算法采用广度优先的搜索策略,在访问了网络中所有的节点后,最终生成从源点到其余各点的最短路径树. 该算法在提取最短路径节点时需要访问所有的临时标记节点,效率低下,整个算法的运行时间为 $O(N^2)$(N 为节点总数);而改进的 Dijkstra 算法采用满二叉堆数据结构,仅存储临时标记节点,且在节点插入或权值更新时都只需直接向上调整位置,提取顶部节点后,向下调整过程中的循环次数不会超过满二叉堆的高度$[\log_2 n]$(n 为堆中节点个数),整个算法理论上最长运行时间仅为 $O(N\log_2 n)$. 但是,由于每次当前源节点向外扩展时向堆中加入少量的临时标记节点,同时移出堆顶节点,所以实际的 n 值远小于节点总数 N,即 $O(N\log_2 n)$ 远小于 $O(N^2)$. 在上述的仿真算例中,取不同 N 值,分别记录了我们的改进算法的运行时间(包括读入数据文件时间和算法运行时间),如图 3 所示. 从图中可以看出,在这种情况下,改进 Dijkstra 算法的实际运行时间与节点数 N 近似成线性关系,即其时间复杂度近似为 $O(N)$. 因此,在时间复杂性上,本文提出的改进算法远优于原 Dijkstra 算法.

5 结 论

在经典 Dijkstra 算法的基础上,为了节省计算机存储空间和提高运行效率,定义了新的节点类高效存储网络数据信息,使存储空间从 $N\times N$ 量级减少至 N 量级;利用二叉堆数据结构管理网络节点,从中搜寻最短路径节点,使算法的时间复杂度由原来的 $O(N^2)$ 降至 $O(N\log_2 n)$,对于上述仿真实例中的情形,可近似降为 $O(N)$,大大提高了该算法效率和性能.

参考文献:

[1] 严寒冰,刘迎春. 基于 GIS 的城市道路网最短路径算法探讨[J]. 计算机学报,2000,23(2):210-215.

[2] 纪晓东,王德隽,周继成. 通信网最短路径神经网络选择控制器[J]. 北京邮电大学学报,1996,19(2):46-52.

[3] 张建中,陈世军,余大祥. 最短路径射线追踪方法及其改进[J]. 地球物理学进展,2003,18(1):146-150.

[4] 王小忠,孟正大. 机器人运动规划方法的研究[J]. 控制工程,2004,11(3):280-284.

[5] 王臣杰,毛海城,杨得志. 图的节点-弧段联合结构表示法及其在 GIS 最优路径选取中的应用[J]. 测绘学报,2000,29(1):47-51.

[6] 王涛,李伟生. 低代价最短路径树的快速算法[J]. 软件学报,2004,15(5):660-665.

The Improvement and Implementation of the Network Shortest Path Algorithm

LI Feng, ZHANG Jian-zhong*

(Dept. of Communication Engineering, Xiamen University, Xiamen 361005, China)

Abstract: From economizing memory space and increasing operation speed, The Dijkstra Algorithm analysis efficiency was improved by using a new node class to store the topological information of the network. The data structure of full binary heap are used to get the shortest path value node and in which the new nodes are inserted in a queue. The emulational result proves that in some network structures, the complexity of the calculation time can be reduced from $o(N^2)$ to nearly $o(N)$ by improved algorithm.

Key words: Dijkstra algorithm; adjacency-node array; heapsort