贪婪算法在构建物流网络中的应用

任文轩[1,2]
(1.中国科学技术大学 计算机科学与技术学院,安徽 合肥 230027;
2.浙江海洋学院 公共实验中心,浙江 舟山 316004)

摘 要:分析了目前我国物流产业发展的现状,提出了如何利用 Dijkstra 算法在一个物流网络的各个节点之中,找出适合作为物流中心的节点。然后根据实际情况的需要,提出了一种经过改进的并行 Prim 算法。根据这一算法可以在整个物流网络当中找出两条以上的物流送货线路,从而提高物流运输的工作效率并在一定程度上减少物流产生的损耗。

关键词:物流网络;Dijkstra 算法;Prim 算法
中图分类号:TP312 文献标识码:B 文章编号:1674-7720(2011)23-0016-03

Greedy algorithm in constructing the logistics network application

Ren Wenxuan[1,2]
(1.Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China;
2.Public Experiment Center, Zhejiang Ocean University, Zhoushan 316004, China)

Abstract: This paper analyzes the current situation of logistics industry in China and puts forward how to use the Dijkstra algorithm in a logistics network to find out the suitable node as a logistics center node. Then according to the actual needs, improves parallel Prim algorithm. According to this algorithm, more than two transportation lines in the entire logistics network can be found, thereby improving the transportation efficiency and to a certain extent, reduce the loss of the logistics.

Key words: logistics network; the Dijkstra algorithm; the Prim algorithm

贪婪技术就是在每一步操作中,“贪婪”地选择最佳操作,并希望通过一系列局部的最优选择进而对全局问题产生一个最优解。以收银员找零问题为例,在中国广泛使用的人民币的纸币面额有:d_1=100(100 元)、d_2=50(50 元)、d_3=20(20 元)、d_4=10(10 元)、d_5=5(5 元)、d_6=1(1 元)。当用这些面额的纸币来给出 37 元的找零时,大多数人都会选择给出一张 20 元、一张 10 元、一张 5 元和两张 1 元的纸币。这就是“贪婪”的想法在影响着人们。首先给出一张 20 元的纸币就可以将余额降到最低,然后再用同样的思路进行下面的操作。同样人们也会想把这种技术应用到其他问题的求解上。

在现代社会中,物流产业已经成为国民经济发展的动脉,其发展程度可以说是衡量一国现代化程度和综合国力的重要标志之一,被喻为促进经济增长的“加速器”[1]。但是当前我国物流成本占 GDP 比例较高,而下降较为缓慢,反映出我国物流效益整体水平仍然较低。而要改善现状就要从需求的物流服务水平出发,以尽可能小的物流费来实现整个物流网络的合理化。整个物流系统可以用连节点(物流中心、需求点)和运输路线构成的物流网络来表示。而如何在这个物流网络中找到可以使位于物流中心的多个配送人员可以更快地将货物送至需求点的路径,即是本文所要讨论的问题。

1 问题描述

从图论的角度可以把某个物流区域归纳为一个图 $G=(V,E)$[2],其中 V_i 其为连节点(物流中心、需求点),$E_{ij}=[V_i,V_j]$ 为连接节点 V_i、V_j 的边,并且均有一个非负权值 $Q(E_{ij})=Q_{ij}$ 用来记录两个连节点之间的路径的损耗,则最后所求得的配送路线即可以看作是一个最小生成树,并且是图 G 的一个子图 $G^*=(V^*,E^*)$,且 $V=V^*$,E 包含 E^*。

如图 1 所示,假设无向图 G 表示一个物流网络,其中 V_0、V_1、V_2、V_3、V_4、V_5、V_6、V_7、V_8、V_9 分别表示 10 个连

节点，E_{01}、E_{02}、E_{23} 等为各个节节点之间的路径。下面需要做的工作就是在这 10 个连节点之中，找出一个最适合作为物流中心的节点，然后再从该节点出发继续寻找最优的物流路径。而在此之前需要将无向图 G 存储在计算机中，其存储方式有数组储存方式和多重双向链表存储方式两种。

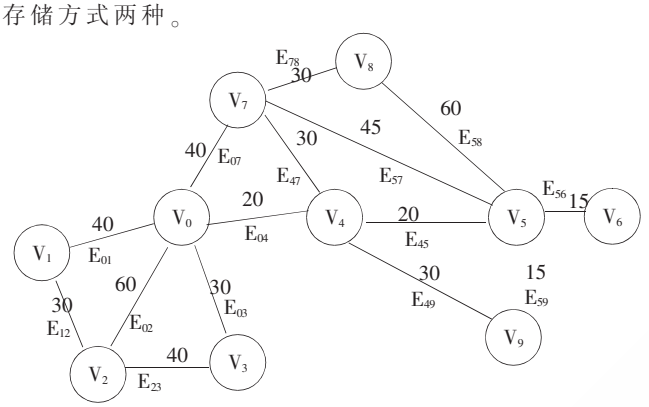

图 1 无向图 G

无向图 G 的数组储存方式如图 2 所示。

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9
V0	∞	40	60	30	20	∞	∞	40	∞	∞
V1	40	∞	30	∞	∞	∞	∞	∞	∞	∞
V2	60	30	∞	40	∞	∞	∞	∞	∞	∞
V3	30	∞	40	∞	∞	∞	∞	∞	∞	∞
V4	20	∞	∞	∞	∞	20	∞	30	∞	30
V5	∞	∞	∞	∞	20	∞	15	45	60	15
V6	∞	∞	∞	∞	∞	15	∞	∞	∞	∞
V7	40	∞	∞	∞	30	45	∞	∞	30	∞
V8	∞	∞	∞	∞	∞	60	∞	30	∞	∞
V9	∞	∞	∞	∞	30	15	∞	∞	∞	∞

图 2 邻接矩阵

为了在实际运算中更方便地解决问题，还需要采用多重双向链表的方式作为边的存储结构。其中每一个顶点用一个节点表示，它包含两个域：vex 域表示该顶点相关的信息，firstEdge 域表示第一条与该顶点相关联的边，如图 3 所示。

vex	firstEdge

图 3 每个顶点的一个节点的域

而存储边的多重双向链表内应当包含如下域：边的起始节点(Pvex)、边的终点节点(Nvex)、边上的权值(Weight)、指向起始节点和终点节点的下一条边的指针域(Pnext)和(Nnext)、指向起始节点和终点节点的上一条边的指针域(Pprior)和(Nprior)，如图 4 所示。该无向图 G 的链表存储结构如图 5 所示。

Pvex	Nvex	Weight	Nprior	Nnext	Pprior	Pnext

图 4 存储边的多重双向链表内的域

2 用 Dijkstra 算法来确定物流中心

Dijkstra 算法的思想是：首先求出从起点到最接近起点的顶点之间的最短路径，然后求出第二近的，以此类推。推而广之，在第 i 次迭代开始以前，该算法已经确定了 $i-1$ 条连接起点和离起点最近顶点之间的最短路径[3]。

该算法的部分伪代码如下：

```
Dijkstra(G,s)          //首先需要输入连通图 G=(V,E)和
                       //它的顶点 s，算法输出为对 V 中的每个
                       //顶点 v 来说，从 s~v 的最短路径长度 dv
Initialize(Q)          //将顶点队列初始化
for V 中的每一顶点 v do
dv←∞;
Insert(Q,v,dv)         //初始化优先队列中顶点的优先级
ds←0;Decrease(Q,s,ds)  //将 s 的优先级更新为 ds
Vt←Φ
for i←0 to |V|-1 do
u*←DeleteMin(Q)        //删除优先级最小的元素
Vt←  Vt ∪ {u*}
for V-Vt 中每一个和 u* 相邻的顶点 u  do
if  du* +w(u*,u)<du
    du←du* +w(u*,u);
Decrease(Q,u,du)
```

Dijkstra 算法的效率取决于图本身的数据结构，以及表示集合 $V-V_t$ 的优先队列的数据结构。本文的图采用邻接链表来表示，优先队列用最小堆来实现，所以其时间复杂度属于 $O(|E|\log|V|)$。经过该算法的计算可以得到图 G 各个节点到其他节点所有最短路径之和。如果图的权值足够准确，即既包含节点之间的运输成本又考虑到节点之间的运输频率等其他因素，就可以将与其他节点最短路径之和最小的那个节点作为整个物流网络的物流中心。其运算结果如下：$\sum_{i=0}^{9} d_{0i}=405$，$\sum_{i=0}^{9} d_{1i}=655$，$\sum_{i=0}^{9} d_{2i}=765$，$\sum_{i=0}^{9} d_{3i}=565$，$\sum_{i=0}^{9} d_{4i}=385$，$\sum_{i=0}^{9} d_{5i}=445$，$\sum_{i=0}^{9} d_{6i}=$

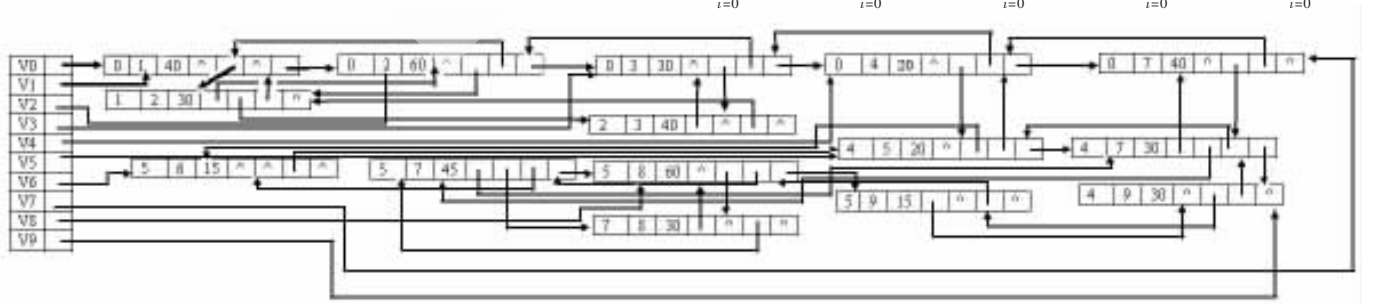

图 5 无向图 G 的链表存储结构

565，$\sum_{i=0}^{9} d_{7i}=515$，$\sum_{i=0}^{9} d_{8i}=710$，$\sum_{i=0}^{9} d_{9i}=540$。

由此，在这个图中就可以根据计算出来的结果来确定节点 V_4 与其他节点最短路径之和最小，因此在无向图 G 中最适合做物流中心的就是 V_4。

3 用改进的 Prim 算法来计算最佳路径

3.1 Prim 算法

在实际的物流过程中，并不是每次送货都需要遍历图中所有的节点，如何在这几个点中找到一条最佳路径从而在最大程度上节省物流成本是各物流公司所要面对的一个问题。Prim 算法是解决上述问题的一个非常有效的方法。假设需要经过 n 个需求点，就需要在无向图中找出 $n-1$ 条边，使包含该 n 个节点的生成树在保持连通的同时还要使权值最小。其步骤如下：

令图 $G=(V,E)$，其生成树的顶点集合为 V_t。

(1)把物流中心节点 V_4 加入到 V_t 中。

(2) 在所有 V_t 与 $V-V_t$ 节点之间寻找一条权值最小的边 $e\in E$ 并将其加入生成树。

(3)把(2)中找到的边 e 对应的属于 $V-V_t$ 的节点 v 加入 V_t 集合，如果 V_t 集合中已有 n 个元素，则算法结束；否则继续执行(2)。

3.2 并行 Prim 算法

在实际工作中，如果货物量比较大或者为了减少整个物流运输的时间等原因，则需要从物流中心同时派出两组以上的运输车辆来完成送货任务。这就需要在原来的 Prim 算法的基础上加以改进。首先要确定物流中心所需要运输车辆的组数，如果是需要两组运输车辆则得出的结果应为 $G_1(V^*_1,E^*_1)$ 和 $G_2(V^*_2,E^*_2)$ 两个子树，其分别对应两组运输车辆的送货路径。其算法分析如下：

(1) 在各个节点到其他节点所有最短路径之和中找出数值最大的那个节点(在图 G 中为 V_2)。

表 1 各节点到其他节点所有最短路径之和

V0	V1	V2	V3	V4	V5	V6	V7	V8	V9
405	655	765	565	385	445	565	515	710	540

(2)找出 V_2 所有邻边中权值最小的一条边 E_{12}，并将此边加入到结果子树 G_1 的边集 E^*_1 中，同时将 V_1、V_2 加入到子树 G_1 的点集 V^*1 中；然后用 Prim 算法找出 V_1、V_2 到物流中心 V_4 的最短路径 $V_2--V_1--V_0--V_4$，并将该路径上的所有节点和边均加入子树 G_1 中。

(3) 根据表 1 选择除物流中心节点外到其他节点所有最短路径之和最小的节点，判断该节点以及其临界节点是否在 V^*_1 中，若在，则接着去寻找次小值的节点继续判断。如果不在则将此节点的邻边中权值最小的一条边加入到结果子树 G_2 的边集 E^*_2 中；然后用 Prim 算法找出节点到物流中心的最短路径，并将该路径上的所有节点和边均加入子树 G_2 中。

在图 G 中，除物流中心节点外到其他节点所有最短路径之和最小的节点是 V_0，但是 V_0 已经存在于 V^*_1 中了，因此需要找次小值 V_5 继续判断，发现 V_5 以及其邻节点均不在 V^*_1 中，因此可将 V_5 的最小临边 E_{59} 加入到 G_2 的边集 E^*_2 中，将 V_5、V_9 加入到 G_2 的点集 V^*_1 中；然后找出 V_5、V_9~V_4 的最短路径 $V_9--V_5--V_4$，并将其加入到子树 G_2 中。

这里如果需要增加第三组运输车辆时，就按照步骤(3) 所描述的在剩下的节点中继续寻找合适的点去构建子树 G_3。

(4)针对剩余的孤立点连接到两个子树的距离，判断其加入到各自最近子树后子树的总路径长度的增加值，取较小者将其加入子树。

在本例中还有 V_3、V_6、V_7、V_8 4 个节点，分别判断其加入 G_1 或 G_2 后路径长度的增加值。V_3 离 G_1 近，则加入后路径长度增加 30，V_6 离 G_2 近，则加入后路径长度增加 15，V_7 离 G_1、G_2 的距离一样都是 30，V_8 离 G_2 近加入后路径长度增加 60。因此这里选择先将 V_6 加入 G_2。

(5)重复步骤(4)直到图 G 中无任何孤立点存在。

经过上述算法，得到了 G_1、G_2 两个子树即两条送货路线。

本文介绍了如何利用贪婪技术在一个物流网络的各个节点之中，找出适合作为物流中心的节点，并提出了一种经过改进的并行 Prim 算法，使其在整个物流网络中可以找出两条以上的物流送货线路来提高整个物流运输的工作效率，使其在一定程度上减少物流损耗。但本文所提出的算法改进还存在很多不足之处，今后的工作是要改进算法，进一步考虑运输路线的回路等问题的解决。

参考文献

[1] 章海峰.进口物资中转运输选址-分配问题[D].武汉：华中科技大学，2006.

[2] 黄冬梅，张岭，韩彦岭.并行搜救算法在确定灾后搜救路线中的应用[J].计算机应用研究，2011，28(2)：472-480.

[3] LEVITIN A. Introduction to the design and analysis of algorithms[M].潘彦译.北京：清华大学出版社，2010.

[4] 江波，张黎.基于 Prim 算法的最小生成树优化研究[J].计算机工程与设计，2009，30(13)：3244-3247.

[5] OMRAN M G, ENGELBRECHT A P, SALMAN A. Particle swarm optimization method for image clustering[J]. International Journal on Pattern Recognition and Artificial, 2005,19(3):297-322.

[6] SALEH B, SADOUN B. Design and implementation of a GIS system for planning [J]. International Journal on Digital Libraries, 2006,6(2):210-218.

(收稿日期：2011-08-04)

作者简介：

任文轩，男，1983 年生，硕士，实验师，主要研究方向：计算机软件理论与多媒体技术。