轨道交通系统票务清分算法

黄 胜，孟世聪，胡幼华
(华东师范大学 计算机科学技术系，上海 200062)

摘 要：提出了一个实现轨道交通系统票务清分的算法。给出了清分的精确算法，在论证精确算法的不可实现性的基础上，演变出切实可行的票务清分的近似算法，并通过一个清分实例来进一步阐述如何实现清分。

关键词：票务清分算法；最短路径；次短路径
中图法分类号：TP301.6 文献标识码：A 文章编号：1001-3695(2004)06-0104-03

An Algorithm of Income Distribution of the Underground System

HUANG Sheng, MENG Shi-cong, HU You-hua
(Dept. of Computer Science & Technology, East China Normal University, Shanghai 200062, China)

Abstract: An algorithm is presented in this paper for distributing tickets income of the city underground system. To begin with, a precise distributing algorithm is given but afterwards proved unrealizable. Based on such conclusion, an approximate and realizable distributing algorithm is put forward instead with a distributing example to illustrate how it works.

Key words: Algorithm of Tickets Income Distribution; Shortest Path; Second Shortest Path

近年来轨道交通系统规模不断扩大，轨道交通系统将成为一个空前复杂的网状结构，其中各条线路又有不同的营运方。在这样一个复杂的网状结构中，从站点 s 到站点 t，很可能有多种路径，而每条路径很可能经过了若干中转站点，即该路径可能跨过了不同的几条轨道线路(无论走过哪条路径，从 s 点到 t 点的票价 Q 是固定的)。清分的目的便是要把 Q 合理地分配给从 s 到 t 所经过的各条线路的营运方，这是轨道交通营运必须解决的一个重要问题。

1 清分算法的提出和分析

1.1 精确算法

精确算法是指从站点 s 到站点 t，各条所经线路按各自在这次路途中所占的长度来分得票务收入。

设第 p 张交通卡的编号为 N_p，N_p 号交通卡第 q 次使用的花费为 N_{pq}，第 p 张交通卡第 q 次使用所经过的总长度为 L_{pq}，在第 k 条路线上通过的距离为 L_{pqk}，则第 k 条线路所获得的权值 V_k 的计算公式为

$$V_k = \sum_{p=1}^{n}\sum_{q=1}^{m}\left\{L_{pqk}\times\frac{N_{pq}}{L_{pq}}\right\} \tag{1}$$

式中 n 表示卡的总数，m 表示该卡总的使用次数。

第 k 条线路的营运方应分得的收入 InC_k 为

$$InC_k = \text{All-Income}\times\left\{\frac{V_k}{\sum_{i=1}^{t}V_i}\right\} \tag{2}$$

式中 All-Income 为轨道交通系统营运总收入，V_i 为第 i 条线路的权值，t 为总线路数。

为了实现精确算法必须获得每一张卡在各条线路所走过的长度，就需要记录下持卡者此次旅程所经过的站点。这要求在各个换乘站点中增设大量的刷卡机。而这是难以实现的：

(1) 刷卡机是比较昂贵的硬件设备，增设刷卡机必然带来巨额的硬件投入。

(2) 在换乘轨道线路时刷卡，将使人流速度降低，在客流高峰期必将出现拥挤。

(3) 由于人流量很大，即便是只需要记录进站和出站时两点的信息，数据库也需要占用很大存储空间。如果再加上换乘点的信息，则存储空间将成数量级的增加，这必然导致数据库维护备份复杂度的提高和清分数据源安全可靠性的降低。

因此要求提出一种合理可行的方法。

1.2 路径近似算法

设 P_m 是乘客选择第 m 短路径的概率，W_{ij} 表示以站点 i 为起点，j 为终点所获得的收入。

设 $L_{ij,1}$ 表示站点 i 到站点 j 的最短路径，这条路径经过了线段 $|S_1S_2|_1$，$|S_2S_3|_2$，…，$|S_dS_{d+1}|_d$(其中 $|S_dS_{d+1}|$ 表示线段长度，下标表示相应线段所属的线路号分别为 1, 2, …, d)。于是这 d 条线路的所获权值分别为 $(|S_1S_2|_1\times P_1\times W_{ij})\div L_{ij,1}$，$(|S_2S_3|_2\times P_1\times W_{ij})\div L_{ij,1}$，…，$(|S_dS_{d+1}|_d\times P_1\times W_{ij})\div L_{ij,1}$。其中 P_1 表示乘客选择最短路径换乘的概率。

同理 $L_{ij,2}$ 表示站点 i 到站点 j 的次短路径，$L_{ij,3}$ 表示站点 i 到站点 j 的次次短路径…。每一条路径又可以为各自所经过的线路分得一个权值。第 k 条线路的权值 $V_{k,1}$ 就是不同路径下通过该线路所得的权值的总和：

$$V_{k,1} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{m=1}^{M}\{|S_gS_h|_kP_mW_{ij}/L_{ij,m}\} \tag{3}$$

式中 $|S_gS_h|_k$ 为从站点 i 到站点 j 的第 m 短的路径中经过线路 k 上的总长度，M 表示从站点 i 到站点 j 的所有路径数，n 表示站点总数。易证在 P_m 精确的情况下 $V_{k,1} = V_k$。

收稿日期：2003-05-27；修返日期：2003-07-06

根据经典概率问题[1,2]可知, P_m 服从单侧近似离散正态分布, 即对绝大部分乘客来讲, 他们往往选择最短或次短路径换乘, 于是只需计算概率最高(记为 P_1)的最短路径和概率次高(记为 P_2)的次短路径加权就可以得到较精确的权值:

$$V_{k,2} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{m=1}^{2}\left\{|S_gS_h|_kP_mW_{ij}/L_{ij,m}\right\} \quad (4)$$

式中 P_1, P_2 可以通过统计调查[3]获得。

在理想情况, 当所有乘客都选择了最短路径时, 得到最简式:

$$V_{k,3} = \sum_{i=1}^{n}\sum_{j=1}^{n}\left\{|S_gS_h|_kW_{ij}/L_{ij,1}\right\} \quad (5)$$

几种方案性能比较如表 1 所示。

表 1　算法可行性对照表

公式编号	时间复杂性要求	描　　述	设备和成本要求
(1)	高	容易推导, 但不可行	最大
(3)	高到难以计算	精确数学模型, 不可行	次大
(4)	较高(N5)	比较精确, 可行	较小
(5)	低(N2)	精确度一般, 可行	最小

由表 1 可以看出, (4) 和(5) 是最可行的两种方法, 表 2 对两种方法作进一步分析。

表 2　可行算法性能对照表

公式编号	站点数	精确度	计算时间
(4)	较少时(20 ~100)	较高	主频为 1000M, 小时级
(4)	较多时(100 ~1000)	较高	主频为 2000M, 十小时级
(5)	较少时(20 ~100)	一般	主频为 1000M, 毫秒级
(5)	较多时(100 ~1000)	一般	主频为 1000M, 秒级

由上述分析可知式(4) 既实际又准确。为了用式(4) 实现清分算法, 关键是求得 W_{ij} 和 $L_{ij,m}$。而为了求得 W_{ij} 和 $L_{ij,m}$ 须求得最短路径和次短路径。

2　最短路径算法和次短路径算法的实现

2.1　Dijkstra 最短路径算法描述

Dijkstra 算法[4~6]是通过分布方法求出最短路径的。可以验证按长度顺序产生最短路径时, 下一条最短路径是由一条已产生的最短路径加上一条边形成。仅利用数组 p 便可以存储最短路径: p[i] 给出从 s 到达 i 的最短路径中顶点 i 的前驱顶点。从 s 到 i 的最短路径是 i- > p[i] - > p[[i]] - >, …, - > s。

为能方便地按长度递增的顺序产生最短路径, 定义 d[i] 为在已产生的最短路径中加入一条最短边的长度, 从而使得扩充的路径到达顶点 i_o。最初, 仅有从 s 到 s 的一条长度为 0 的路径, 这时对于每个顶点 i, d[i] 等于 a[s] [i] (a 是图中的长度邻接矩阵) 。为产生下一条路径, 需要选择还未产生最短路径的下一个节点, 在这些节点中 d 值最小的即为一条路径的终点, 因此有些顶点的 d 值可能会发生变化。如此最后得到 d[i] 的便是 s 到 i 的最短路径的长度。

2.2　Dijkstra 最短路径算法时间复杂性分析

任何最短路径算法必须至少对每一条边检查一次, 因为任何一条边都有可能在最短路径中。由于是使用耗费邻接矩阵来描述图, 仅决定那条边在途中就需 $O(n^2)$ 的时间。因此, 采用这种描述方法的最短路径算法需花费 $O(n^2)$ 的时间。

2.3　次短路径算法描述

次短路径的算法实质上就是对 Dijkstra 算法的合理迭代再加以比较选择。运行一次 Dijkstra 算法, 可以得到 s 到 k 的最短路径 L_{sk}, s- > q_{12}, q_{23}, …, $q_{i,i+1}$, …, $q_{n-1,n}$- > k , 其中 $q_{i,i+1}$ 表示最短路径上的一条边。依次在图中分别去掉 q_{12}, q_{23}, …, $q_{i,i+1}$, …, $q_{n-1,n}$ 这些边, 修改长度邻接矩阵 a, 再分别运行 Dijkstra 算法可以得到去掉了第 i 条边后的最短路径 $L_{sk,i}$, 则从 s 点到 k 点次短路径 $LL_{sk} = \min\{L_{sk,i}, i = 0, 1, \cdots, m$, 其中 m 表示了路径 L_{sk} 的中的边数$\}$。

2.4　次短路径算法描述时间复杂性分析

在求解次短路径的过程中有两个阶段要调用 Dijkstra 算法: ①求得两点间的最短路径, 时间是 $O(n^2)$; ②依次去掉最短路径中的边时间 O(n) , 再分别调用 Dijkstra 算法, 时间复杂度为 $O(n^2)$。最后可以得到总的次短路径的时间复杂度为 $O(n^5)$。

3　清分方案的具体实现

3.1　清分方案说明

根据线路图获得线路数据信息; 获得长度邻接矩阵; 获得两两间的最短和次短路径; 从交通卡的使用信息库获得两两站点间的营运收入; 使用式(4) 和式(2) 实现清分。

3.2　清分实例分析

以一张简单的轨道交通线路网示意图(图 1, 来源于上海轨道交通的部分线路) 为例来说明清分方案的实施过程。

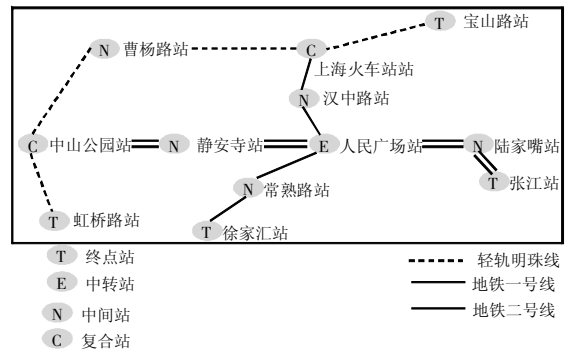

图 1　轨道交通线路网示意图

(1) 算法的数据结构

```
class StationReferenceTable
{ int SeqNO;                    站点的序列号
int RootSeqNO;                  站点的根节点的序列号
int LineNO;                     站点的线路号
int StationCode;                站点的编号
CString StationName;            站点的名称
int StationType;                站点的类型
} StationArray[ ];              站点信息表
int Shortest_Path[ ][ ];        两两站点间最短路径
int AllPathsLength[ ][ ];       两两站点间最短路径的长度
```

其中 StationType 中 T 为终始点, E 为换乘点, N 为普通站点, C 既是换乘点又是终始点。在后两个两维数组中, 第一下标为起始站点序列号, 第二下标为目标站点编号序列号。RootSeqNO 表示具有不同 SeqNO 但是有相同的绝对物理地址的站点序列号, 其值等于这样的站点第一次出现时的 SeqNO; 没有这样性质站点的 RootSeqNO 为 0。如表 3 中 SeqNO 为 8 的人民广场站在第一条线路中已经出现过, 其 SeqNO 为 3, 所以 RootSeqNO 便为3; 而 SeqNO 为 7 的静安寺站没有这样的性质, 其 RootSeqNO 的为 0。

(2) 清分的实现

首先,根据线路图建立表 3(表中线路 1, 2, 3 分别对应地铁一, 二号线和轻轨明珠线)

表 3 站点信息表

线路站点信息表	所属线路名	SeqNO	RootSeqNO	LineNO	StationCode	StationName	StationType
St, 1	地铁一号线	1	0	1	1	上海火车站站	4
St, 2	地铁一号线	2	0	1	2	汉中路站	3
St, 3	地铁一号线	3	0	1	3	人民广场站	2
St, 4	地铁一号线	4	0	1	4	常熟路站	3
St, 5	地铁一号线	5	0	1	5	徐家汇站	1
St, 6	地铁二号线	6	0	2	11	中山公园站	4
St, 7	地铁二号线	7	0	2	12	静安寺站	3
St, 8	地铁二号线	8	3	2	13	人民广场站	2
St, 9	地铁二号线	9	0	2	14	陆家嘴站	3
St, 10	地铁二号线	10	0	2	15	张江站	1
St, 11	轻轨明珠线	11	0	3	21	宝山路站	1
St, 12	轻轨明珠线	12	1	3	22	上海火车站站	4
St, 13	轻轨明珠线	13	0	3	23	曹杨路站	3
St, 14	轻轨明珠线	14	6	3	24	中山公园站	4
St, 15	轻轨明珠线	15	0	3	25	虹桥路站	1

其次,在已知两两站点间距的基础上对表 3 进行两边夹搜索,获得长度邻接矩阵,如表 4 所示。

表 4 邻接矩阵表

邻接矩阵	st. 1	st. 2	st. 3	st. 4	st. 5	st. 6	st. 7	st. 8	st. 9	st. 10	st. 11	st. 12	st. 13	st. 14	st. 15
st. 1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	3	-1	-1
st. 2	1	0	1	-1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	-1
st. 3	-1	1	0	1	-1	-1	2	0	2	-1	-1	-1	-1	-1	-1
st. 4	-1	-1	1	0	1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
st. 5	-1	-1	-1	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
st. 6	-1	-1	-1	-1	-1	0	2	-1	-1	-1	-1	-1	2	0	2
st. 7	-1	-1	2	-1	-1	2	0	2	-1	-1	-1	-1	-1	2	-1
st. 8	-1	1	0	1	-1	-1	2	0	2	-1	-1	-1	-1	-1	-1
st. 9	-1	-1	2	-1	-1	-1	-1	2	0	1	-1	-1	-1	-1	-1
st. 10	-1	-1	-1	-1	-1	-1	-1	-1	1	0	-1	-1	-1	-1	-1
st. 11	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	1	-1	-1	-1
st. 12	0	1	-1	-1	-1	-1	-1	-1	-1	-1	1	0	3	-1	-1
st. 13	3	-1	-1	-1	-1	2	-1	-1	-1	-1	-1	3	0	2	-1
st. 14	-1	-1	-1	-1	-1	0	2	-1	-1	-1	-1	-1	2	0	2
st. 15	-1	-1	-1	-1	-1	2	-1	-1	-1	-1	-1	-1	-1	2	0

再次,获得两两之间的最短路径和次短路径。

①最短路径加权

分别以 1, 2, …, 15 为起始点调用 Dijstra 算法。在第 r 次调用后就会得到两个数组 p^r(存储以各站点到站点 r 的最短路径的前导节点) 和 d^r(存储各站点到站点 r 的最短路径的长度)。把这两个数组分别复制到 Shortest _ Path[r] [ ] 和 AllPathsLength[r] [ ] 中(表 5 和表 6 分别表示最终的 Shortest _ Path 和 AllPathsLength 的值)。根据这两张表便可以得到 $L_{ij,1}$ 和 $|S_gS_{h|k}: L_{ij,1}$ = AllPathsLength[i] [j] , 而 $|S_gS_{h|k}$ 是在查询 Shortest _ Path 获得最短路径的过程中得到的。

表 5 两两最短路径表

两两最短路径表	st. 1	st. 2	st. 3	st. 4	st. 5	st. 6	st. 7	st. 8	st. 9	st. 10	st. 11	st. 12	st. 13	st. 14	st. 15
st. 1	0	1	2	3	4	13	3	2	3	9	1	0	1	13	14
st. 2	2	0	2	3	4	7	3	2	3	9	1	2	1	7	6
st. 3	2	3	0	3	4	7	3	0	3	9	1	2	1	7	6
st. 4	2	3	4	0	4	7	3	4	3	9	1	2	1	7	6
st. 5	2	3	4	5	0	7	3	4	3	9	1	2	1	7	6
st. 6	13	3	7	3	4	0	6	7	3	9	1	13	6	0	6
st. 7	2	3	7	3	4	7	0	7	3	9	1	2	6	7	6
st. 8	2	8	2	8	4	7	8	0	8	9	1	2	1	7	6
st. 9	2	3	9	3	4	7	3	9	0	9	1	2	1	7	6
st. 10	2	3	9	3	4	7	3	9	10	0	1	2	1	7	6
st. 11	11	1	2	3	4	13	3	2	3	9	0	11	1	13	14
st. 12	2	12	2	3	4	13	3	2	3	9	12	0	12	13	14
st. 13	13	1	2	3	4	13	6	2	3	9	1	13	0	13	6
st. 14	13	3	7	3	4	7	14	7	3	9	1	13	14	0	14
st. 15	13	3	7	3	4	15	6	7	3	9	1	13	6	15	0

假设 i = 1(上海火车站), j = 7(静安寺站), 查询过程如下:

Shortest _ Path[1] [7] = 3, Shortest _ Path[1] [3] = 2, Shortest _ Path[1] [2] = 1 = i, 停止查询。于是最短路径为 1- > 2- > 3- > 7(即上海火车站- > 汉中路站- > 人民公园站- > 静安寺站), 长度为 AllPathsLength[1] [7] = $L_{17.1}$ = 4。其中线段 $\overrightarrow{S_1S_2}$, $\overrightarrow{S_2S_3}$, $\overrightarrow{S_3S_7}$ 分别属于线路 1, 线路 1, 线路 2。查询表 6 可得在 i = 1, j = 7 时, $|S_1S_2|_1$ = 1, $|S_2S_3|_1$ = 1, $|S_3S_7|_2$ = 2。依此类推, 让 i, j 分别取完所有的值, 便可以得到所有需要的 $L_{ij,1}$ 和 $|S_gS_{h|k}$。

表 6 两两最短路径的长度

两两最短路径长度表	st. 1	st. 2	st. 3	st. 4	st. 5	st. 6	st. 7	st. 8	st. 9	st. 10	st. 11	st. 12	st. 13	st. 14	st. 15
st. 1	0	1	2	3	4	5	4	2	4	5	1	0	3	5	7
st. 2	1	0	1	2	3	5	3	1	3	4	2	1	4	5	7
st. 3	2	1	0	1	2	4	2	0	2	3	3	2	5	4	6
st. 4	3	2	1	0	1	5	3	1	3	4	4	3	6	5	7
st. 5	4	3	2	1	0	6	4	2	4	5	5	4	7	6	8
st. 6	5	5	4	5	6	0	2	4	6	7	7	5	2	0	2
st. 7	4	3	2	3	4	2	0	2	4	5	5	4	4	2	4
st. 8	2	1	2	1	2	4	2	0	2	3	3	2	5	4	6
st. 9	4	3	2	3	4	6	4	2	0	1	5	4	7	6	8
st. 10	5	4	3	4	5	7	5	3	1	0	6	5	8	7	9
st. 11	1	2	3	4	5	6	5	3	5	6	0	1	4	6	8
st. 12	2	1	2	3	4	5	4	2	4	5	1	0	3	5	7
st. 13	3	4	5	6	7	2	4	5	7	8	4	3	0	2	4
st. 14	5	5	4	5	6	4	2	4	6	7	7	5	2	0	2
st. 15	7	7	6	7	8	2	4	6	8	9	9	7	4	2	0

②次短路径加权

根据次短路径算法,就可以得到相应的次短路径信息表,其加权的算法与最短路径加权的算法相类似,不再细述。

最后综合所获数据,实现清分。

通过以上的几步演算便得到了实现清分必不可少的数据 $L_{ij,m}$(m = 1, 2) 和 $|S_gS_h|_k$。两两站点间的营运收入 W_{ij} 可以从交通卡使用信息库直接获得。而另一个重要数据 P_m 可以根据经典概率模型通过概率统计求得[1~3]。把这些数据代入式(4), 得到各线路的权值, 最后再代入式(2), 便得到了各线路应分得的收入金额,实现清分。

4 结束语

近年来各大城市轨道交通系统规模不断扩大,票务清分已经成为轨道交通营运必须解决的一个重要问题。本文针对这一实际问题给出了一种实现票务清分的算法。该算法兼顾了准确性和可行性,有较高的实用价值,为实现城市轨道交通系统的票务清分问题建立了科学的算法模型。笔者以 VC + + 为平台开发出了相应的清分软件。

参考文献:

[1] 魏宗舒, 等. 概率论与数理统计教程[M] . 北京: 高等教育出版社, 2001.

[2] 美] 斯皮格尔, 等. 概率与统计[M] . 孙山, 戴中维. 北京: 科学出版社, 2002.

[3] 美] S· 伯恩斯坦, R· 伯恩斯坦. 统计学原理(下册) ——推断性统计学[M] . 史道济. 北京: 科学出版社, 2002.

[4] artaj Sahni. Data Structure, Algorithms, and Applications in C + + [M] . 北京: 机械工业出版社, 2001.

[5] han F B. Three Fastest Shortest Path Algorithms on Real Road Networks[J] . Journal of Geographic Information and Decision Analysis, 1997, 1(1) : 69-72.

[6] arbehenn, Michael. Note on the Complexity of Dijkstra 's Algorithm for Graphs with Weighted Vertices[J] . IEEE Transactions on Computers, 1998, 47(2) : 263-266.

作者简介:

黄胜(1982-) , 男, 四川成都人, 本科生, 研究方向为计算机应用与算法; 孟世聪(1982-) , 男, 安徽芜湖人, 本科生, 研究方向为计算机应用与算法; 胡幼华(1946-) , 女, 浙江平湖人, 教授, 博士生导师, 主要研究方向为系统分析与集成。