# 铁路最优路线问题

## 摘 要

本文研究的是在任意两站点之间没有直接到达的铁路或者直达列车票已售罄的情况下，求出两站点之间的最优路线问题。并给出从宜昌出发乘火车到上海、南京、杭州、苏州、无锡旅游最后回到宜昌的最优旅游路线。

针对问题一：将铁路网转化为一个带权有向图，把问题转化成带权有向图中的最短路径问题，建立了以换乘次数最少、乘车时间最短及乘车费用最少为目标的多目标优化模型。用分层序列法、广度优先搜索算法和改进的Dijkstra算法求解，编程求出优先考虑乘车时间和优先考虑乘车费用两种情况下3对始终站之间的最优路线，广度优先搜索算法求解换乘一次的结果如下所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 起点 | 所乘车次 | 换乘站 | 换乘车次 | 终点 | 时间（h.m） | 费用/元 |
| 先考虑时间 | 丹东 | K190/k187 | 镇江 | D3006/D3007 | 宜昌 | 37.40 | 513 |
| 天津 | T5684/T5681 | 北京西 | T27 | 拉萨 | 49.54 | 374 |
| 白城 | K7304 | 长春 | K1056/k1053 | 青岛 | 28.47 | 208 |
| 先考虑费用 | 丹东 | K190/k187 | 南京 | K696/k697 | 宜昌 | 42.12 | 322 |
| 天津 | T5684/T5681 | 北京西 | T27 | 拉萨 | 49.54 | 374 |
| 白城 | 2262 | 天津 | K1056/k1053 | 青岛 | 46.55 | 171 |

针对问题二：根据问题一中任意两城市之间的最优路线方案，建立新的带权有向图，转化为旅行商问题，以乘车时间最少和乘车费用最少为目标的双目标函数，建立0-1规划模型，结合分层序列法，用LINGO编程求出优先考虑乘车时间和优先考虑乘车费用两种情况下的最优旅游路线，如下所示：

优先考虑乘车时间最少时的最优路线：

D3008

D5680

G7362

D3006

G7001

G7002

宜昌 苏州 杭州 上海 南京 无锡 宜昌

D3007

D5677

D3005

优先考虑乘车费用最少时的最优路线：

K698

1230

1228

T7788

K696

K102

宜昌 无锡 上海 苏州 杭州 南京 宜昌

K695

T7785

K697

1229

1227

**关键字：**多目标最优化 广度优先搜索 改进的Dijkstra算法 0-1规划

## 问题重述

### 1.1问题说明

铁路既是社会经济发展的重要载体之一，同时又为社会经济发展创造了前提条件。近几年来，在全社会客运量稳步上升的同时，长期以来铁路承运了大量旅客。相对于其他的运输方式铁路具有时间准确性高、运输能力大、运行比较平稳、安全性高等优点。同时火车也成为了旅途的首选交通运输工具。

虽然目前铁路网络已经比较发达，但是仍然有很多地方之间并没有直接到达的铁路。并且在节假日期间，一些热门路线的火车票总是一票难求。在这种情况下，需要考虑换乘，即先从乘车站到换乘站，再从换乘站到目的站。

### 1.2需要解决的问题

**1）**给出任意两个站点之间的最优铁路路线问题的一般数学模型和算法。若两个站点之间有直达列车，需要考虑直达列车票已售罄情况下最优的换乘方案。根据附录数据，利用你们的模型和算法求出一下起点到终点的最优路线：丹东→宜昌、天津→拉萨、白城→青岛。

**2）**假设你从打算从宜昌出发乘火车到上海、南京、杭州、苏州、无锡旅游最后回到宜昌，请建立相关数学模型，给出整个行程的最优路线。

## 二.模型假设

**假设1**：接受列车晚点不超过一小时。

**假设2**：优先考虑换乘次数。

**假设3：**从起点乘车时所等车的时间忽略不计。

## 三.符号说明

|  |  |
| --- | --- |
| **符号** | **符号说明** |
|  | 站点构成的集合 |
|  | 任意两个站点之间连通的路线构成的集合 |
|  | 起始站点 |
|  | 终到站点 |
|  | 时间权值集合 |
|  | 费用权值集合 |
|  | 任意两个站点之间能否连通构成的邻接矩阵 |
|  | 换乘次数 |
|  | 乘车总费用 |
|  | 乘车总时间 |
|  | 两站之间的里程 |
|  | 普通火车基本票率 |
|  | 动车组票率 |
|  | 高速动车组票率 |
|  | 递远递减率 |
|  | 保险票率 |
|  | 客票发展金 |
|  | 空调票率 |

## 四.问题分析

本文要解决的是铁路换乘最佳路线的选择问题。针对文中提出的问题，在综合考虑换乘次数，乘车时间和乘车费用的前提下，给出任意两个站点之间的最优换乘路线。对问题的具体分析如下：

### 4.1问题一的分析

本题要求给出任意两个站点之间的最优铁路路线问题的一般数学模型和算法，并根据此算法求出题中给出的3对始终站之间的最优路线。在选择乘车路线时一般考虑三个因素即始终站之间的换乘次数、乘车时间和乘车费用。因为铁路列车的交通网络比较复杂，列车车次繁多，很难掌握各列车的发车时间和所经站点，为了减少换乘所带来的麻烦，把换乘次数最少作为选择最优路线的主要目标，由于主观原因，每个人的经济状况和时间空余安排不同，考虑乘车时间和乘车费用时的主次有所不同，以换乘次数最少、乘车时间最少和乘车费用最少为目标作为多目标函数，变换后两个目标函数的优先级分别求解，在转乘时考虑最大接受列车晚点一小时，不晚点情况下，乘客至少有一小时的换乘时间。将铁路网转化为一个带权有向图，将问题转化为带权有向图中的最短路径问题，运用分层序列法将目标函数分层求解，考虑到广度优先搜索算法计算两次转乘以上时算法复杂度太高，计算速度太慢，改进的Dijkstra算法只能输出最优解而不能输出次优解，无法解决高峰期大量缺票的问题，因此分别用广度优先搜索算法和改进的Dijkstra算法求解，用MATLAB编程求出优先考虑乘车时间和优先考虑乘车费用两种情况下的最优路线。

### 4.2问题二的分析

本题要求建立数学模型，求出从宜昌出发乘火车到上海、南京、杭州、苏州、无锡最后回到宜昌的最优旅游路线。问题二是以乘车时间最少和乘车费用最少为目标的双目标最优化问题。由问题一可得到任意两城市之间的最优路线方案，将这六个城市的站点作为节点，两城市之间的最优路线作为边，构成一个新的带权有向图，将问题转化为旅行商问题。每个车站只有一条进去的路线和一条出去的路线；除起点和终点外，各个车站之间不能构成封闭的环。由于主观原因，每个人的经济状况和时间空余安排不同，考虑乘车时间和乘车费用时的主次有所不同，因此将这两个目标函数的优先级作以变换分别求解。建立0-1规划模型，用分层序列法将目标函数分层求解，用LINGO编程求出优先考虑乘车时间和优先考虑乘车费用两种情况下的最优路线。

## 五.数据分析与处理

### 5.1车次、站点和车次类型的统计

（1）由于附录中的数据量庞大，车次和站点的冗杂性比较高，利用MATLAB编程处理附录中的数据可以得到：全国共有2867个站点、4683个车次，列车类型共有11中，其中城际高速只存在于北京到天津、上海到金山卫之间，只有少数地区可供选择，且具有速度高价格贵等特点。

（2）各种类型列车的列车车次数如下图所示：

图1 各种类型列车的列车车次数统计

从上图可以看出，新空快速、高速动车、动车组的车次数明显高于其他列车次数，我国目前运营的列车主要是快速空调列车、动车和高速动车，因此选择出游列车乘车方式时，大多数应该是选择这三种。

### 5.2任意两个站点间票价的计算

附录的数据中只给出了起点到各个站点的票价，任意两个站点之间的票价并未给出，根据铁道部公布的信息，票价包括三部分：基本客票票价、附加票票价和其他（保险费、客票发展金等），根据铁道部公布的票价计算方案，可得到各车次的硬座车票价格计算公式如下：

变量的定义：两站之间的里程为，普通火车基本票率为，动车组票率为，高速动车组票率为，区段起点为，区段终点为，递远递减率为，保险票率为，客票发展金为，空调票率为，从起始站点到第个站点的票价为，从起始站点到第个站点的票价为。

普客：

普快：

快速：

新空普客：

新空普快：

新空快速：

新空特快：

动车组：

高速动车：

新空直达：

城际高速：

通过编程计算得到每个车次的列车任意两点之间的票价，计算结果见附录。

## 六.问题一模型的建立与求解

针对问题一，要求用给出的模型和算法求出3对始终站之间的最优路线，首先建立衡量最优路线的三个指标，然后运用分层序列法和广度优先搜索算法求出最优路线。

### 6.1模型的建立

将整个铁路线路网建立成一个带权有向图，将站点作为节点，列车通过两个站点之间的线路作为边，节点集，边集，构成带权有向图，记有向图中由节点，确定的边为。

（1）定义邻接矩阵，若存在同一辆列车经过和，则和邻接，定义邻接矩阵如下：





（2）定义每条边都有若干个行驶时间权值、发车时间权值和乘车费用权值，行驶时间权值为列车从行驶到的行驶时间，发车时间权值为列车从行驶到的列车的出发时间，乘车费用权值为列车从行驶到的费用。记时间权值为，发车时间权值为，费用权值为，若有个车次经过站节点和，则有个时间权值和费用权值，记得时间权值集合和费用权值集合分别为、，有







（3）将从节点到节点的乘车方案表示为：



其中为中转节点,表示边的行驶时间权值、发车时间权值和费用权值分别取、和，同理表示边的行驶时间权值、发车时间权值和费用权值分别取、和。

### 6.1.1目标函数

（1）换乘次数最小，在乘车方案中换乘次数等于中间节点的数目：



（2）乘车所用时间最少，乘车时间=所换乘列车的发车时间—从始点出发的列车的发车时间+乘坐换乘列车的行驶时间：



（3）乘车所花费用最少，乘车费用等于从起始站到换乘站之间的车费加上从换乘站到终点站之间的车费：



### 6.1.2约束条件

（1）换乘时从起点到达换乘站时的时间要小于换乘列车的出发时间，以保证能够顺利换乘车辆：



（2）换乘的站点之间要有路线存在，以保证能够换乘成功：



**综上所述得到问题一的多目标优化模型如下：**





### 6.2模型的求解

### 6.2.1算法分析

（1）分层序列法的算法求解步骤如下所示：

假设换乘次数的优先级最高，乘车时间次优，对其求解：



求得其最优值为。在可行域中，的区域称为最优点集合域，表示为：

。

在内求第二个分目标函数的最优值：





求得其最优值为。的最优点集合域为：

。

在内求的最优值。

（2）广度优先搜索算法：

① 设经过站点的车次集合，经过站点的车次集合为，。

② 若，则从到可以直达，通过对集合进行搜索，得到乘车时间最小的有车次方案集，再通过搜索出费用最小的车次方案集，即为在此优先级下直达的最优解集合。若，则需转乘一次。

③设中所有车次包含的站点的集合为，中所有车次包含的站点集合为，集合，即为中转站点集，通过起点、终点和中转站集合即可得到转乘一次的的方案集，通过对集合进行搜索，得到乘车时间最小的有车次方案集，再通过搜索出费用最小的车次方案集，即为在此优先级下转乘一次的最优解集合。

（3）改进的Dijkstra算法：

以为起始站点，设置计时起点为。

1. 令若，令。
2. 对每个，若用代替，记录，并存下标号。计算，把更新后的顶点的记为，令。
3. 若，则停止算法；若，用，转到第②步计算。算法停止更新完成后，的值即为从到得最短时间，通过依次逆向寻找的前一标号站点得到路径，并通过确定车次。

### 6.3.2模型求解

（1）广度优先搜索算法求解：

用分层序列法和广度优先搜索算法，通过MATLAB编程求解得到在目标函数优先级不同的情况下，题中给出的3对始终站之间的最优路线如下所示：

当乘车所用时间最少的目标函数的优先级大于乘车所花费用最少的目标函数时，最优路线如下所示：

表1.优先考虑乘车时间最少时的最优路线及次优路线

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 起点 | 所乘车次 | 换乘站 | 换乘车次 | 终点 | 时间 | 费用 | 转乘时间 |
| 丹东 | K190/k187 | 镇江 | D3006/D3007 | 宜昌 | 37.40 | 513 | 1.24 |
| 丹东 | K190/K187 | 常州 | D3006/D3007 | 宜昌 | 37.40 | 538 | 17.40 |
| 天津 | T5684/T5681 | 北京西 | T27 | 拉萨 | 49.54 | 374 | 1.40 |
| 天津 | K257 | 郑州 | T264/T265 | 拉萨 | 50.38 | 443 | 1.40 |
| 白城 | K7304 | 长春 | K1056/k1053 | 青岛 | 28.47 | 208 | 12.2 |
| 白城 | K7304 | 长春 | K704/K701 | 青岛 | 28.56 | 208 | 2.15 |

当乘车所花费用最少的目标函数的优先级大于乘车所用时间最少的目标函数时，最优路线如下所示：

表2.优先考虑乘车费用最少时的最优路线及次优路线

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 起点 | 所乘车次 | 换乘站 | 换乘车次 | 终点 | 时间 | 费用 | 转乘时间 |
| 丹东 | K190/k187 | 南京 | K696/k697 | 宜昌 | 42.12 | 322 | 17.40 |
| 丹东 | K190/K187 | 镇江 | K696/k697 | 宜昌 | 66.12 | 333 | 4.2 |
| 天津 | T5684/T5681 | 北京西 | T27 | 拉萨 | 49.54 | 374 | 1.40 |
| 天津 | K7728/K7725 | 北京西 | T27 | 拉萨 | 60.42 | 374 | 1.40 |
| 白城 | 2262 | 天津 | K1056/k1053 | 青岛 | 46.55 | 171 | 12.2 |
| 白城 | 2262 | 天津 | K704/K701 | 青岛 | 47.4 | 171 | 2.15 |

（2）用改进的Dijkstra算法求解：

通过MATLAB编程求解得到在目标函数优先级不同的情况下，题中给出的3对始终站之间的最优路线如下所示：

表3.优先考虑乘车时间最短的最优方案

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 站点 | 丹东 | 沈阳北 | 北京 | 石家庄 | 武汉 | 宜昌 |
| 车次 | K7318 | D606 | K7744/K7741 | G501 | D5876/D5877 | —— |
| 站点 | 天津 | 石家庄北 | 兰州 | 拉萨 | —— | —— |
| 车次 | K548/K545 | Z55 | K917 | —— | —— | —— |
| 站点 | 白城 | 长春 | 沈阳北 | 天津 | 北京南 | 青岛 |
| 车次 | K7304 | G8020 | K1056/K1053 | C2002 | D333 | —— |

此时乘车从丹东到宜昌所用的时间为27小时15分钟，乘车费用为862元；从天津到拉萨所用的时间为50小时25分钟，乘车费用为512元；从白城到青岛所用的时间为27小时42分钟，乘车费用为675元。每次转乘至少有一个小时的过渡时间。

表4. 优先考虑转乘次数时乘车的最优方案

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 起点 | 所乘车次 | 换乘车站 | 换乘车次 | 终点 | 时间（h.m） | 费用（元） |
| 丹东 | K190/k187 | 镇江 | D3006/D3007 | 宜昌 | 37.40 | 513 |
| 天津 | T5684/T5681 | 北京西 | T27 | 拉萨 | 49.54 | 374 |
| 白城 | K7304 | 长春 | K1056/k1053 | 青岛 | 28.47 | 208 |

每次转乘至少有一个小时的过渡时间。对比表1和表4可以看出，此方案与用分层序列法和广度优先搜索算法求出的最优乘车路线一样。

### 6.4问题一的结果分析

（1）由表1和表2可以看出，在优先考虑乘车时间和优先考虑乘车费用这两种情况下，从丹东到宜昌和从白城到青岛的最优路线是不一样的。根据不同的实际情况，旅客可以根据自己的经济情况和时间安排选择适合自己的乘车路线。

（2）在考虑转乘次数最优时,广度优先搜索算法可以快速求得直达和转乘一次的所有方案,但是当转乘次数超过2次时,算法复杂度很高,运算速度很慢。采用改进的Dijkstra算法求解时，以转乘次数最少、乘车时间最短和费用最小为目标都能求出最优解，但是只能输出最优解，无法输出次优解，当最优方案无法执行时，没有备选方案，因此应结合二者的求解方案。

## 七. 问题二模型的建立与求解

### 7.1模型的建立

根据问题一的模型和算法，可以求得任意两个站点之间的的最优路线，将问题二中的六个站点构成新的带权有向图，以六个城市站点作为节点，任意两个城市站点之间的线路作为边，节点集，边集，构成带权有向图，记有向图中中的由节点，确定的边为。

以问题一中的任意两站点之间的最优路线解作为边的权值，每条边都有唯一的时间权值、票价费用权值，分别用和来表示。

建立0-1模型，表示不经过从节点到节点的边，表示经过从从节点到节点的边。

**7.1.1目标函数**

（1）乘车总时间最少，乘车时间等于各条边的时间权值之和：



（2）乘车总费用最少，乘车费用等于各条边的票价费用权值之和：



**7.1.2约束条件**

（1）每个车站只有一条进去的路线：



（2）每个车站只有一条出去的路线：



（3）除起点和终点外，各个车站之间不能构成封闭环：



**综上所述得到问题二的多目标优化模型如下所示：**





### 7.2模型的求解

根据问题二所建立的模型，利用MATLAB编程求得从宜昌出发经过上海、杭州、苏州、无锡最后回到宜昌的最优旅游路线如下：

表5.优先考虑乘车时间最少时的最优旅游路线

|  |  |  |
| --- | --- | --- |
| 站点 | 车次 | 站点 |
| 宜昌 | D3008/D3005 | 苏州 |
| 苏州 | D5680/D5677 | 杭州 |
| 杭州 | G7362 | 上海 |
| 上海 | G7002 | 南京 |
| 南京 | G7001 | 无锡 |
| 无锡 | D3006/D3007 | 宜昌 |

选择此路线的乘车总时间为20小时8分钟，总费用为1271元。

表6. 优先考虑乘车费用最少时的最优旅游路线

|  |  |  |
| --- | --- | --- |
| 站点 | 车次 | 站点 |
| 宜昌 | K698/K695 | 无锡 |
| 无锡 | 1230/1227 | 上海 |
| 上海 | 1228/1229 | 苏州 |
| 苏州 | T7788/T7785 | 杭州 |
| 杭州 | K102 | 南京 |
| 南京 | K696/K697 | 宜昌 |

选择此路线的总时间为40小时20分钟，总费用为276元。

### 7.3问题二的结果分析

由表5和表6可以看出，当优先考虑乘车时间最少时，乘车的总费用比较高，是优先考虑乘车费用最少方案所花费用的4.6倍；当优先考虑乘车费用最少时，乘车的总时间是前者方案乘车时间的2倍，这与实际情况是相符的，速度快的列车车票价格要比普通列车的车票价格贵，而速度快的列车，乘车时间往往比较少，两者不能兼得，旅行者可以根据自己的实际情况选择适合自己的旅游路线。

## 八.模型的评价

### 8.1模型的优点

（1）在数据处理方面，先将每个站点和车次进行编号，并查询出每种车次的票价计算方法，方便模型的建立与求解。

（2）模型建立比较合理，充分考虑了旅客乘车时对换乘次数、乘车时间和乘车费用的不同需求，建立了三个目标函数，用分层序列法将目标函数分清主次，求出了每种情况下的最优路线。

（3）问题一用两种算法求出最优路线，通过对比可以得出每种算法的优缺点，在实际情况中，旅客可以根据自己的情况加以选择。

（4）所建模型为一般数学模型，运用此模型可以求出题中任意两个站点之间的最佳路线。

### 8.2模型的不足

问题二的模型是基于问题一的结果建立的，问题一结果的准确性会影响到问题二求解结果的准确性。

## 十一.参考文献

[1] 司守奎，孙玺菁.数学建模算法与应用.国防工业出版社.2011.

[2] 马良河，刘言斌，廖大庆.城市公交线路网络图的最短路与乘车路线问题.数学实践与认识.2004.6.

[3] 张秀利，梁迎春，董申.多目标结构模糊化的分层序列法.哈尔滨工业大学出版社.1999 .

## 十二.附录

附录一：任意两个站点之间的车票价格

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 3 | 6 | 10 | 14 | 24 | 28 | 33 | 38 | 45 | 61 | 66 | 72 | 74 | 82 | 84 | 84 | 92 | 93 | 98 | 109 | 122 | 128 | 132 | 132 | 138 | 138 | 140 |
| 0 | 0 | 6 | 9 | 12 | 23 | 28 | 33 | 39 | 48 | 69 | 75 | 83 | 87 | 97 | 101 | 103 | 111 | 114 | 124 | 143 | 163 | 173 | 183 | 184 | 193 | 196 | 200 |
| 0 | 0 | 0 | 5 | 8 | 18 | 23 | 29 | 35 | 43 | 65 | 70 | 79 | 82 | 92 | 97 | 98 | 106 | 109 | 120 | 139 | 158 | 168 | 179 | 180 | 189 | 192 | 196 |
| 0 | 0 | 0 | 0 | 5 | 15 | 20 | 26 | 32 | 40 | 62 | 67 | 75 | 79 | 89 | 94 | 95 | 103 | 106 | 116 | 136 | 155 | 165 | 175 | 177 | 186 | 188 | 193 |
| 0 | 0 | 0 | 0 | 0 | 12 | 17 | 23 | 29 | 37 | 59 | 64 | 72 | 76 | 86 | 90 | 92 | 100 | 103 | 113 | 133 | 152 | 162 | 172 | 174 | 183 | 185 | 190 |
| 0 | 0 | 0 | 0 | 0 | 0 | 7 | 12 | 17 | 26 | 48 | 53 | 61 | 65 | 75 | 79 | 81 | 89 | 92 | 102 | 122 | 141 | 151 | 161 | 163 | 172 | 174 | 178 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 13 | 21 | 43 | 48 | 56 | 60 | 70 | 75 | 76 | 84 | 87 | 97 | 117 | 136 | 146 | 156 | 158 | 167 | 169 | 174 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 15 | 37 | 42 | 51 | 55 | 64 | 69 | 71 | 78 | 82 | 92 | 111 | 130 | 141 | 151 | 152 | 161 | 164 | 168 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 31 | 36 | 45 | 49 | 58 | 63 | 64 | 72 | 76 | 86 | 105 | 124 | 134 | 145 | 146 | 155 | 158 | 162 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 | 28 | 36 | 40 | 50 | 54 | 56 | 64 | 67 | 77 | 97 | 116 | 126 | 136 | 138 | 147 | 149 | 154 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 15 | 18 | 28 | 33 | 34 | 42 | 45 | 56 | 75 | 94 | 104 | 115 | 116 | 125 | 128 | 132 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 13 | 23 | 28 | 29 | 37 | 40 | 50 | 70 | 89 | 99 | 109 | 111 | 120 | 122 | 127 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 15 | 19 | 21 | 29 | 32 | 42 | 61 | 81 | 91 | 101 | 102 | 111 | 114 | 118 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 15 | 17 | 25 | 28 | 38 | 58 | 77 | 87 | 97 | 99 | 108 | 110 | 114 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 8 | 15 | 18 | 28 | 48 | 67 | 77 | 87 | 89 | 98 | 100 | 105 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 11 | 14 | 24 | 43 | 63 | 73 | 83 | 84 | 93 | 96 | 100 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 12 | 22 | 42 | 61 | 71 | 81 | 83 | 92 | 94 | 99 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 14 | 34 | 53 | 63 | 73 | 75 | 84 | 86 | 91 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 31 | 50 | 60 | 70 | 72 | 81 | 83 | 87 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 40 | 50 | 60 | 61 | 70 | 73 | 77 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 30 | 41 | 42 | 51 | 54 | 58 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 21 | 23 | 32 | 34 | 39 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 13 | 22 | 24 | 29 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 12 | 14 | 18 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 13 | 17 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 9 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

附录二：程序源代码

|  |
| --- |
| 第一问 |
| DataAnaly.m |
| clear,home  %% 导入数据  %[~,~,data] = xlsread('附件一.xls');  [data] = readtable('Train.csv');  [n,m] = size(data);  for i=1:n  data.A\_Time{i}=data.A\_Time{i}(12:end);  data.D\_Time{i}=data.D\_Time{i}(12:end);  end  save data.mat  %% 转换时间  [n] = size(data,1);  for i=1:n  index = strfind(data.A\_Time{i},':');  h = str2double(data.A\_Time{i}(1:index(1)-1));  m = str2double(data.A\_Time{i}(index(1)+1:index(2)-1));  totalM = h\*60 + m;  data.A\_Time\_CurrentMinute(i) = totalM;  data.A\_Time\_TotalMinute(i) = totalM + (data.Day(i)-1)\*24\*60;  end  for i=1:n  index = strfind(data.D\_Time{i},':');  h = str2double(data.D\_Time{i}(1:index(1)-1));  m = str2double(data.D\_Time{i}(index(1)+1:index(2)-1));  totalM = h\*60 + m;  data.D\_Time\_CurrentMinute(i) = totalM;  data.D\_Time\_TotalMinute(i) = totalM + (data.Day(i)-1)\*24\*60;  end  save data2.mat  %% 将每一列车的站点信息放入一行，保存成CVS格式  [n,m] = size(data);  fileID = fopen('trainLine.txt','w');  for i=1:n  if data.S\_No(i)==1  fprintf(fileID,'\n%s,%s',data.ID{i},data.Type{i});  end  fprintf(fileID,',%s',data.Station{i});  end  fclose(fileID);  %% 列车类型与列车编号  trainType = unique(data.Type);  n = size(trainType,1);  typeIndex = cell(n,1);  for i = 1:n  typeIndex{i} = find(strcmp(data.Type,trainType(i)));  end  trainNo = unique(data.ID);  n = size(data,1);  for i=1:n  [~,No] = intersect(trainNo,data.ID{i});  data.NumNo(i) = No;  end  %% 站点编号  station = unique(data.Station);  n = size(data,1);  for i=1:n  [~,No] = intersect(station,data.Station{i});  data.StationNo(i) = No;  end  %% 建立连接矩阵  m = length(station);  startStation = find(data.S\_No==1);  adjMat = zeros(m,m);  n = size(data,1);  for i = 1:n-1  endStation = startStation(find(startStation > i,1))-1;  % [~,s\_No] = intersect(station,data.Station{i});  for j = i+1:endStation  % [~,d\_No] = intersect(station,data.Station{j});  adjMat(data.StationNo(i),data.StationNo(j)) = 1;  end  end  save data3  %% 任意两站点之间的车次  m = length(station);  startStation = find(data.S\_No==1);  trainList = cell(m,m);  n = size(data,1);  for i = 1:n-1  endStation = startStation(find(startStation > i,1))-1;  for j = i+1:endStation  trainList{data.StationNo(i),data.StationNo(j)} = [trainList{data.StationNo(i),data.StationNo(j)},data.NumNo(i)];  end  end  save data4.mat |
| directSearchFun1.m |
| function [ path,pathTime,pathCost ] = directSearchFun1( data,trainList,adjMat,station,s\_name,d\_name )  [~,s\_No] = intersect(station,s\_name);  [~,d\_No] = intersect(station,d\_name);  %% 直达  if isempty(trainList{s\_No,d\_No})  path = [];  else  path = trainList{s\_No,d\_No}';  end  path = [ones(size(path,1),1)\*s\_No path ones(size(path,1),1)\*d\_No];  %% 计算时间  pathTime = zeros(size(path,1),1);  for i=1:size(path,1)  pathTime(i) = timeFun( data,path(i,:) );  end  %% 计算费用  pathCost = zeros(size(path,1),1);  for i=1:size(path,1)  [pathCost(i),~] = pathCostFun( data,path(i,:) );  end  end |
| directSearchFun2.m |
| function [ path,pathTime,pathCost,stopTime ] = directSearchFun2( data,trainList,adjMat,station,s\_name,d\_name )  [~,s\_No] = intersect(station,s\_name);  [~,d\_No] = intersect(station,d\_name);  %% 一次转乘  path=[];  s\_set = find(adjMat(s\_No,:)==1);  d\_set = find(adjMat(:,d\_No)==1);  stop = intersect(s\_set, d\_set);  for i=1:length(stop)  firstarc = trainList{s\_No,stop(i)};  for j = 1:length(firstarc)  secondarc = trainList{stop(i),d\_No};  for k = 1:length(secondarc)  path = [path;[firstarc(j) stop(i) secondarc(k)]];  end  end  end  path = [ones(length(path),1)\*s\_No path ones(length(path),1)\*d\_No];  %% 计算时间  pathTime = zeros(size(path,1),1);  stopTime = cell(size(path,1),1);  for i=1:size(path,1)  [pathTime(i),stopTime{i}] = timeFun( data,path(i,:) );  end  %% 计算费用  pathCost = zeros(size(path,1),1);  for i=1:size(path,1)  [pathCost(i),~] = pathCostFun( data,path(i,:) );  end  end |
| costFun.m |
| function [ cost ] = costFun( data,s\_No,d\_No,trainID )  %costFun 计算两个站点之间某车次的票价  % data 为列车数据库  % s\_No 为出发点编号  % d\_No 为目的地编号  % trainNo 为车次编号  % cost 为票价  % s\_No = 933; d\_No = 1065; trainID = 1154;  s = find(data.NumNo==trainID&data.StationNo==s\_No);  d = find(data.NumNo==trainID&data.StationNo==d\_No);  if data.S\_No(s)==1&&(~isempty(data.P1(d)))&&(~isnan(data.P1(d)))  cost = data.P1(d);  return  end  if data.S\_No(s)==1&&(~isempty(data.P3(d)))&&(~isnan(data.P3(d)))  cost = data.P3(d);  return  end  x = data.Distance(d) - ...  data.Distance(s);  index = find(data.NumNo==trainID,1);  if strcmp(data.Type{index},'城际高速')  cost = min(table2array(data(d,9:12))) - min(table2array(data(s,9:12)));  cost =round(cost);  return  end  if strcmp(data.Type{index},'动车组')  cost = 0.3085\*x;  cost =round(cost);  return  end  if strcmp(data.Type{index},'高速动车')  cost = 0.485\*x;  cost =round(cost);  return  end  if strcmp(data.Type{index},'新空直达')    cost = min(table2array(data(d,9:12))) - min(table2array(data(s,9:12)));  cost =round(cost);  return  end  sigma = 1;  if strcmp(data.Type{index},'普客')  sigma = sigma+0.02;  end  if strcmp(data.Type{index},'普快')  sigma = sigma+0.02+0.2;  end  if strcmp(data.Type{index},'新空普快')  sigma = sigma+0.02+0.2+0.25;  end  if strcmp(data.Type{index},'新空普客')  sigma = sigma+0.02+0.25;  end  if strcmp(data.Type{index},'快速')  sigma = sigma+0.02+0.4;  end  if strcmp(data.Type{index},'新空快速')  sigma = sigma+0.02+0.4+0.25;  end  if strcmp(data.Type{index},'新空特快')  sigma = sigma+0.02+0.4+0.25;  end  eta = 0.05861;  if 0<x&&x>200  cost = eta\*x\*sigma+1;  elseif 200>x&&x<=500  cost = eta\*(200+(x-200)\*0.9)\*sigma+1;  elseif 500>x&&x<=1000  cost = eta\*(200+300\*0.9+(x-500)\*0.8)\*sigma+1;  elseif 1000>x&&x<=1500  cost = eta\*(200+300\*0.9+500\*0.8+(x-1000)\*0.7)\*sigma+1;  elseif 1500>x&&x<=2500  cost = eta\*(200+300\*0.9+500\*0.8+500\*0.7+(x-1500)\*0.6)\*sigma+1;  else  cost = eta\*(200+300\*0.9+500\*0.8+500\*0.7+1000\*0.6+(x-2500)\*0.5)\*sigma+1;  end  cost =round(cost);  end |
| timeFun.m |
| function [ totalTime,stopTime,startTime ] = timeFun( data,sigpath )  %timeFun 计算路径方案的时间  % data 列车数据  % path 路径，格式为起点-车次-中转站-车次-终点  % totalTime 方案的总时间  % stopTime 方案的中间停留时间  % startTime 方案的出发时间  n = (length(sigpath)-1)/2;  section = zeros(n,2);  sectionTime = zeros(n,1);  stopTime = zeros(n-1,1);  for i=1:n  temp = sigpath(i\*2-1:i\*2+1);  m = find(data.NumNo==temp(2)&data.StationNo==temp(1));  n = find(data.NumNo==temp(2)&data.StationNo==temp(3));  section(i,:) = [m n];  sectionTime(i) = data.A\_Time\_TotalMinute(n)-data.D\_Time\_TotalMinute(m);  end  if n>1  n = (length(sigpath)-1)/2;  for i=1:n-1  s\_stop = data.A\_Time\_CurrentMinute(section(i,2));  d\_stop = data.D\_Time\_CurrentMinute(section(i+1,1));  if d\_stop<s\_stop  stopTime(i) = d\_stop + 24\*60 - s\_stop;  else  stopTime(i) = d\_stop - s\_stop;  end  end  else  stopTime = 0;  end  totalTime = sum(sectionTime) + sum(stopTime);  startTime = data.D\_Time(section(1,1));  end |
| bestPathFun.m |
| function [ bestIndex ] = bestPathFun( pathCost,pathTime,type )  %bestPathFun 将方案按照一定的优先级进行排序  % path 所有方案  % type 优先级类型：1为时间优先，2为费用优先  % bestPath 方案的排序  if type==1  [~,bestIndex]=sort(pathTime);  end  if type==2  [~,bestIndex]=sort(pathCost);  end  end |
| displayFun.m |
| function displayFun( path,pathCost,pathTime,station,trainNo )  %displayFun 输出方案  for i=1:length(path)  if mod(i,2)==1  fprintf('%s-',station{path(i)})  else  fprintf('%s-',trainNo{path(i)})  end  end  fprintf('%d元-',pathCost)  h = floor(pathTime/60);  m = mod(pathTime,60);  fprintf('%d小时%d分钟\n',h,m)  end |
| main.m |
| % 主程序  % 输出欢迎界面  fprintf('欢迎使用铁路查询系统！\n请输入起始站与终点站\n')  % 用户输入数据  s\_name = input('起始站：','s');  while ~sum(strcmp(station,s\_name))  fprintf('Sorry，没有找到此站点，请重新输入！\n')  s\_name = input('起始站：','s');  end  d\_name = input('终点站：','s');  while ~sum(strcmp(station,d\_name))  fprintf('Sorry，没有找到此站点，请重新输入！\n')  d\_name = input('终点站：','s');  end  type = input('输入优先级，时间优先输入1，费用优先输入2：');  path = cell(2,1);  pathTime = cell(2,1);  pathCost = cell(2,1);  % 直达方案  [ path{1},pathTime{1},pathCost{1} ] = directSearchFun1(...  data,trainList,adjMat,station,s\_name,d\_name );  goOn = 0;  if isempty(path{1})  fprintf('%s到%s ',s\_name,d\_name)  fprintf('没有直达车次 为您推荐如下转乘方案：\n')  goOn = 1;  else  [ bestIndex{1} ] = bestPathFun( pathCost{1},pathTime{1},type );  for i=1:length(path{1})  n = bestIndex{1}(i);  displayFun( path{1}(n,:),pathCost{1}(n),pathTime{1}(n),station,trainNo )  end  end  % 转乘一次  goOn = input('是否继续搜索一次转乘方案，若是输入1，否则输入0：');  while goOn~=1&&goOn~=0  goOn = input('是否继续搜索一次转乘方案，若是输入1，否则输入0：');  end  if goOn == 1  [ path{2},pathTime{2},pathCost{2},stopTime ] = directSearchFun2(...  data,trainList,adjMat,station,s\_name,d\_name );  if isempty(path{2})  fprintf('%s到%s ',s\_name,d\_name)  fprintf('没有一次转乘方案，无法搜索，Sorry\n')  else  [ bestIndex{2} ] = bestPathFun( pathCost{2},pathTime{2},type );  for i=1:length(path{2})  n = bestIndex{2}(i);  displayFun( path{2}(n,:),pathCost{2}(n),pathTime{2}(n),station,trainNo )  h = floor(stopTime{i}./60);  m = mod(stopTime{i},60);  fprintf('中转站等待时间：%d小时%d分钟\n',h,m)  end  end  end |
| oneBestPathFun.m |
| function [ bestPath,cost,time ] = oneBestPathFun( data,trainList,adjMat,station,s\_name,d\_name,type )  %oneBestPathFun 输出一条最优路线  [ path,pathTime,pathCost ] = directSearchFun1( data,trainList,adjMat,station,s\_name,d\_name );  if type==1  [directMin,directIndex] = min(pathTime);  bestPath = path(directIndex,:);  time = pathTime(directIndex);  cost = pathCost(directIndex);  else  [directMin,directIndex] = min(pathCost);  bestPath = path(directIndex,:);  time = pathTime(directIndex);  cost = pathCost(directIndex);  end  end |
| pathCostFun.m |
| function [ totalCost,cost ] = pathCostFun( data,sigpath )  %pathCostFun 计算路径方案的总费用  % path为路径方案  % totalCost为总费用  % cost为各车次票价  n = (length(sigpath)-1)/2;  cost = zeros(n,1);  for i=1:n  cost(i) = costFun( data,sigpath(i\*2-1),sigpath(i\*2+1),sigpath(i\*2) );  end  totalCost = sum(cost);  end |
| startStationMFile.m |
| m = length(station);  startStation = find(data.S\_No==1);  trainList = cell(m,m);  n = size(data,1);  for i = 1:n-1  endStation = startStation(find(startStation > i,1))-1;  for j = i+1:endStation  trainList{data.StationNo(i),data.StationNo(j)} = [trainList{data.StationNo(i),data.StationNo(j)},data.NumNo(i)];  end  end  save data4.mat |
| dijkstraMain.m |
| tic  s = find(strcmp(station,'上海'));  d = find(strcmp(station,'苏州'));  beginH = 8;  beginM = 0;  beginTime = beginH\*60 + beginM;  [ t,p,q ] = dijkstraTurnFun( adjMat,C,TS,s,d,beginTime,60 );  toc  p(s) = 0;  path = graphpred2path(p',d);  n = size(path,2)\*2-1;  fullpath = zeros(1,n);  % fullpath(1) = path(1);  % fullpath(2) = trainList{path(1),path(2)}(q(path(2)));  % fullpath(3) = path(2);  % fullpath(4) = trainList{path(2),path(3)}(q(path(3)));  for i = 1:n  if mod(i,2)==1  fullpath(i) = path((i+1)/2);  else  fullpath(i) = trainList{path(i/2),path(i/2+1)}(q(path(i/2+1)));  end  end  pathCost = pathCostFun( data,fullpath );  [ totalTime,stopTime,startTime ] = timeFun( data,fullpath );  displayFun( fullpath,pathCost,totalTime,station,trainNo )  disp(stopTime)  % s\_No = 1579;  % d\_No = 933;  % station(s\_No)  % station(d\_No)  % train = trainNo(trainList{s\_No,d\_No}(q(d\_No))) |
| DataPossess.m |
| %% 任意两站点之间的车次  m = length(station);  startStation = find(data.S\_No==1);  trainList = cell(m,m);  C = cell(m,m);  TS = cell(m,m);  r = zeros(m,m);  n = size(data,1);  h = waitbar(0,'Please wait...');  for i = 1:n-1  endStation = startStation(find(startStation > i,1))-1;  for j = i+1:endStation  s\_No = data.StationNo(i);  d\_No = data.StationNo(j);  sigpath = data.NumNo(i);  trainList{s\_No,d\_No} = [trainList{s\_No,d\_No},sigpath];  %[ totalTime,~,startTime ] = timeFun( data,[s\_No,sigpath,d\_No] );  C{s\_No,d\_No} = [C{s\_No,d\_No},data.A\_Time\_TotalMinute(j)-data.D\_Time\_TotalMinute(i)];  TS{s\_No,d\_No} = [TS{s\_No,d\_No},data.D\_Time\_CurrentMinute(i)];  r(s\_No,d\_No) = r(s\_No,d\_No) + 1;  end  waitbar(i/n)  end  close(h)  save data6.mat  DG = sparse(adjMat);  minTurn = graphallshortestpaths(DG); |
| dijkstraFun.m |
| function [ t,p,q ] = dijkstraFun( adjMat,C,TS,s,d,beginTime,stopTime )  %dijkstraFun 改进的Dijkstra算法  % adjMat 邻接矩阵  % c 节点之间的行驶时间矩阵  % TS 出发时间  N = size(adjMat,1);  Node = 1:N; % 节点集合  t = zeros(N,1); % 到达节点u的最短时间  %dayT = zeros(N,1); % 到达节点u时的当天时间  p = zeros(N,1); % s到u最短路径上的u的前一个节点  q = zeros(N,1); % s到u最短路径上p(u)到u出发时间的索引  %S = [];  %s = 1; % 设置出发点  % 初始化  t(s) = beginTime;  %dayT(s) = 0;  p(s) = nan;  q(s) = nan;  S = s;  for v=setdiff(Node,s)  t(v) = inf;  p(v) = nan;  q(v) = nan;  end  while ~isempty(S)  [~,index] = min(t(S));  u = S(index);  S = setdiff(S,u);  for v = find(adjMat(u,:))    for i = 1:length(TS{u,v})  temp = TS{u,v}(i);  while temp<t(u)+stopTime;  temp = temp + 24\*60;  end  if (t(v)>temp+C{u,v}(i))  p(v) = u;  q(v) = i;  t(v) = temp + C{u,v}(i);  S = union(S,v);  end  end    end  if u == d  return  end  end  end |
| dijkstraTurnFun.m |
| function [ t,p,q,turn ] = dijkstraTurnFun( adjMat,C,TS,s,d,beginTime,stopTime )  %dijkstraFun 改进的Dijkstra算法  % adjMat 邻接矩阵  % c 节点之间的行驶时间矩阵  % TS 出发时间  N = size(adjMat,1);  Node = 1:N; % 节点集合  t = zeros(N,1); % 到达节点u的最短时间  turn = zeros(N,1); % 到达节点u的最小转乘次数  %dayT = zeros(N,1); % 到达节点u时的当天时间  p = zeros(N,1); % s到u最短路径上的u的前一个节点  q = zeros(N,1); % s到u最短路径上p(u)到u出发时间的索引  %S = [];  %s = 1; % 设置出发点  % 初始化  t(s) = beginTime;  turn(s) = 0;  %dayT(s) = 0;  p(s) = nan;  q(s) = nan;  S = s;  for v=setdiff(Node,s)  t(v) = inf;  turn(v) = inf;  p(v) = nan;  q(v) = nan;  end  while ~isempty(S)  tAndTurn = [t(S),turn(S)];  [~,index] = sortrows(tAndTurn,[2,1]);  index = index(1);  %[~,index] = min(t(S));  %[~,index] = min(turn(S));  u = S(index);  S = setdiff(S,u);  for v = find(adjMat(u,:))  for i = 1:length(TS{u,v})  temp = TS{u,v}(i);  while temp<t(u)+stopTime;  temp = temp + 24\*60;  end  if (turn(v)>turn(u)+1)||((turn(v)==turn(u)+1)&&(t(v)>temp+C{u,v}(i)))%(t(v)>temp+C{u,v}(i))  p(v) = u;  q(v) = i;  t(v) = temp + C{u,v}(i);  turn(v) = turn(u) + 1;  S = union(S,v);  end  end  end  if u == d  return  end  end  end |
| 第二问 |
| main2.m |
| travel = {'宜昌东','上海','南京','杭州','苏州','无锡'};  n = length(travel);  TimeFirstBestPath=cell(n,n);  TimeFirstCost = ones(n,n)\*inf;  TimeFirstTime = ones(n,n)\*inf;  type = 1;  for i=1:n  for j=1:n  if i==j  continue  end  [ TimeFirstBestPath{i,j},TimeFirstCost(i,j),TimeFirstTime(i,j) ] = oneBestPathFun( data,trainList,adjMat,station,travel{i},travel{j},type );  end  end  CostFirstBestPath=cell(n,n);  CostFirstCost = ones(n,n)\*inf;  CostFirstTime = ones(n,n)\*inf;  type = 2;  for i=1:n  for j=1:n  if i==j  continue  end  [ CostFirstBestPath{i,j},CostFirstCost(i,j),CostFirstTime(i,j) ] = oneBestPathFun( data,trainList,adjMat,station,travel{i},travel{j},type );  end  end |
| WENTI2.lg4 |
| MODEL:  SETS:  CITY / 1.. 6/: U; ! U( I) = sequence no. of city;  LINK( CITY, CITY):  DIST, ! The distance matrix;  X; ! X( I, J) = 1 if we use link I, J;  ENDSETS  DATA: !Distance matrix, it need not be symmetric;  DIST =  0 1064 767 1215 452 435  1016 0 99 90 25 42  774 99 0 228 71 55  1159 86 216 0 106 123  428 24 72 105 0 15  411 41 55 123 14 0;  ! 0 702 454 842 2396 1196  702 0 324 1093 2136 764  454 324 0 1137 2180 798  842 1093 1137 0 1616 1857  2396 2136 2180 1616 0 2900  1196 764 798 1857 2900 0;  ENDDATA  !The model:Ref. Desrochers & Laporte, OR Letters,  Feb. 91;  N = @SIZE( CITY);  MIN = @SUM( LINK: DIST \* X);  @FOR( CITY( K):  ! It must be entered;  @SUM( CITY( I)| I #NE# K: X( I, K)) = 1;  ! It must be departed;  @SUM( CITY( J)| J #NE# K: X( K, J)) = 1;  ! Weak form of the subtour breaking constraints;  ! These are not very powerful for large problems;  @FOR( CITY( J)| J #GT# 1 #AND# J #NE# K:  U( J) >= U( K) + X ( K, J) -  ( N - 2) \* ( 1 - X( K, J)) +  ( N - 3) \* X( J, K)  );  );  ! Make the X's 0/1;  @FOR( LINK: @BIN( X));  ! For the first and last stop we know...;  @FOR( CITY( K)| K #GT# 1:  U( K) <= N - 1 - ( N - 2) \* X( 1, K);  U( K) >= 1 + ( N - 2) \* X( K, 1)  );  END |

附录三：丹东到宜昌优先考虑乘车时间时的路线：



附录四：丹东到宜昌优先考虑乘车费用时的路线：



附录五：天津到拉萨优先考虑乘车时间时的路线：



附录六：天津到拉萨优先考虑乘车费用时的路线：



附录七：白城到青岛优先考虑乘车时间时的路线：



附录八：白城到青岛优先考虑乘车费用时的路线：

