|  |
| --- |
|  |
| Single Vehicle  **VRPPDTW问题** |
| C++程序 使用说明 |
|  |
|  |
|  |

|  |
| --- |
| **2018.01** |

目录

[1、问题说明 2](#_Toc504504057)

[2、问题的底层网络结构 2](#_Toc504504058)

[3、模型与算法说明 4](#_Toc504504059)

[4、程序的执行 6](#_Toc504504060)

[5、程序的输入和输出 7](#_Toc504504061)

[5.1 输入文件 7](#_Toc504504062)

[5.2 输出文件 9](#_Toc504504063)

[6、测试数据集 12](#_Toc504504064)

[6.1 1辆车同时服务两位乘客 12](#_Toc504504065)

[6.2 1辆车同时服务两位乘客（无合乘） 13](#_Toc504504066)

[6.3 1辆车2位乘客，一人未被服务 14](#_Toc504504067)

[6.4 1辆车同时服务3名乘客 14](#_Toc504504068)

[7、程序说明 16](#_Toc504504069)

[7.1 g\_ReadInputData() 17](#_Toc504504070)

[7.2 拉格朗日松弛 17](#_Toc504504071)

[7.3 DP算法 18](#_Toc504504072)

# 1、问题说明

本算法适用于单车辆带有乘客接、送、时间窗的VRP问题(VRPPDTW)，利用拉格朗日松弛算法和动态规划求解。模型和算法详述见论文[1]。

输入：物理路网，乘客信息（起讫点、时间窗），车辆信息（能力、时间窗）。

输出：车辆的时空路径。

约束：时间窗约束、车辆能力约束。

目标：总运输费用最小。

# 2、问题的底层网络结构

（1）VRPPDTW网络的节点

在VRPPDTW的网络中有几种类型的节点：运输（物理）节点，乘客接、送节点以及车辆的车站节点。

乘客的时间窗表示在接、送节点（乘客的起始地和目的地）旁边的括号中。例如，接乘客1的时间窗是4-7，那就是说，车辆必须在[4,7]这一时间段内接到乘客1，否则乘客会取消这次服务。同样，车辆必须在[11,14]之间将乘客1送到节点d1。

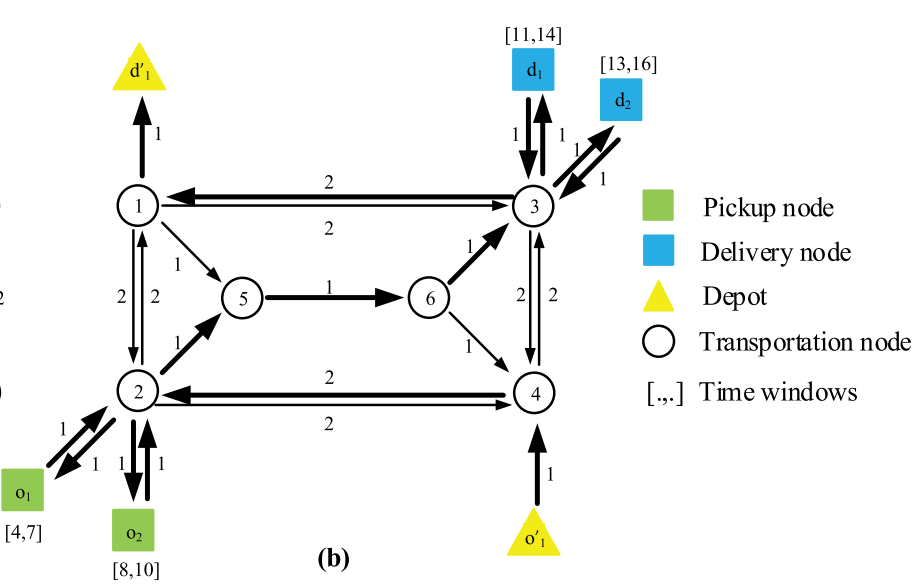


图1 VRPPDTW网络的节点

（2）VRPPDTW网络的弧

因为有不同类型的节点，弧也可以分为3类：运输弧、服务弧和等待弧。

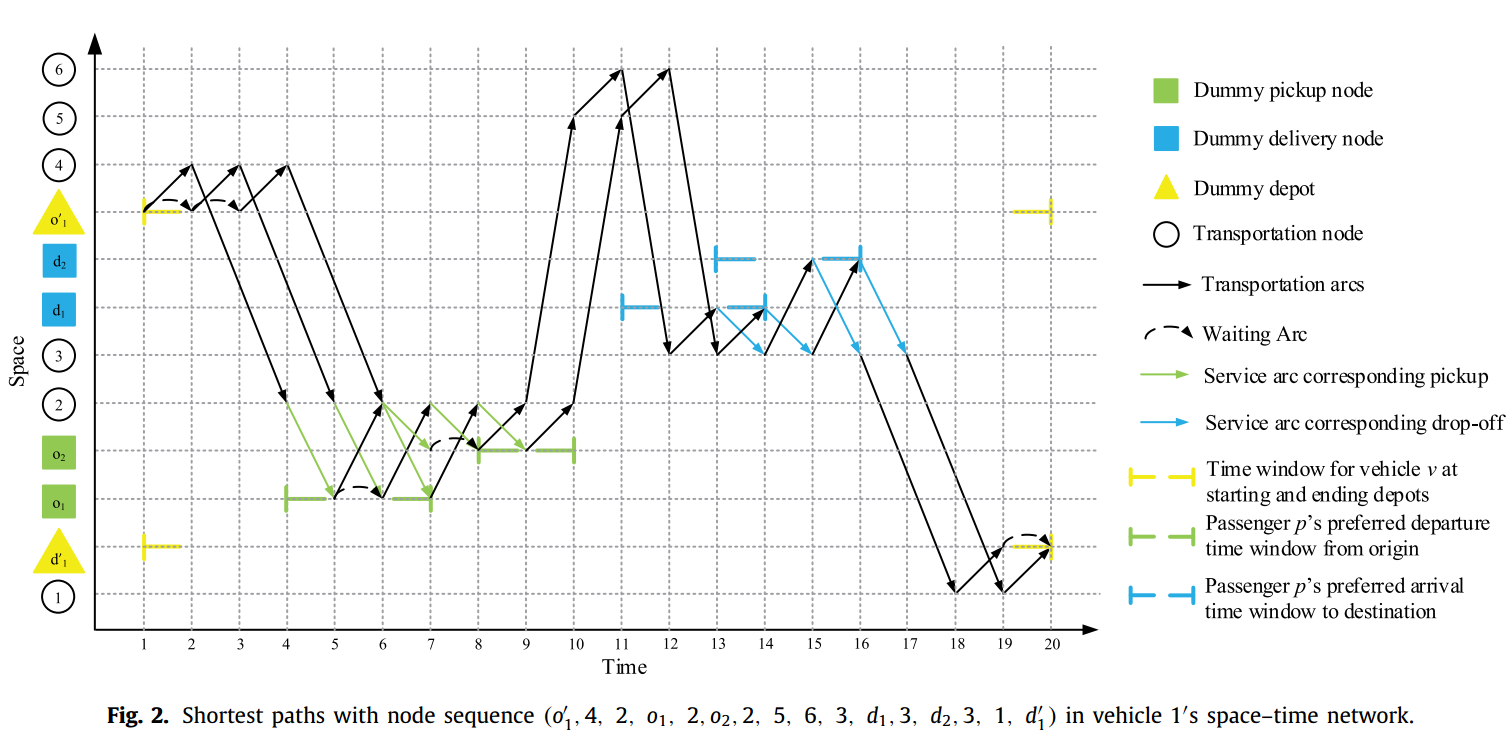


图2 VRPPDTW 网络的弧

①运输弧

运输弧连接相邻的两个运输节点，表示车辆在相应的物理路段上行驶，经过运输弧的乘客和车辆的状态保持不变。

②服务弧

服务弧包括接取弧和送达弧，表示乘客的上下车。由服务弧连接的两个节点分别是接节点和送节点。车辆的乘客状态会因上下车行为而改变。

③等待弧

等待弧表示车辆停车等待以满足时间窗约束。例如，图2中O2的黑色虚线表示车辆在O2等待乘客2到达。一旦乘客2接近该位置，车辆可以立即接他/她。

（3）状态转移

图3描述了车辆载客状态的整个转移过程。从图3的底部可以看到，空车从节点O’1离开，通过物理节点4到节点2准备接乘客1。列车到达节点O1后，就是旅客1的起始点，车辆的状态应该更新为与乘客1已上车，即[p1,\_]。同时，当列车在节点O2接乘客2时，状态将更新为[p1,p2]。此外，当列车到达乘客的到达节点时，乘客应从车辆的乘客状态中移除。最后，车辆的状态必须再次清空。图4详细描述了状态转移。

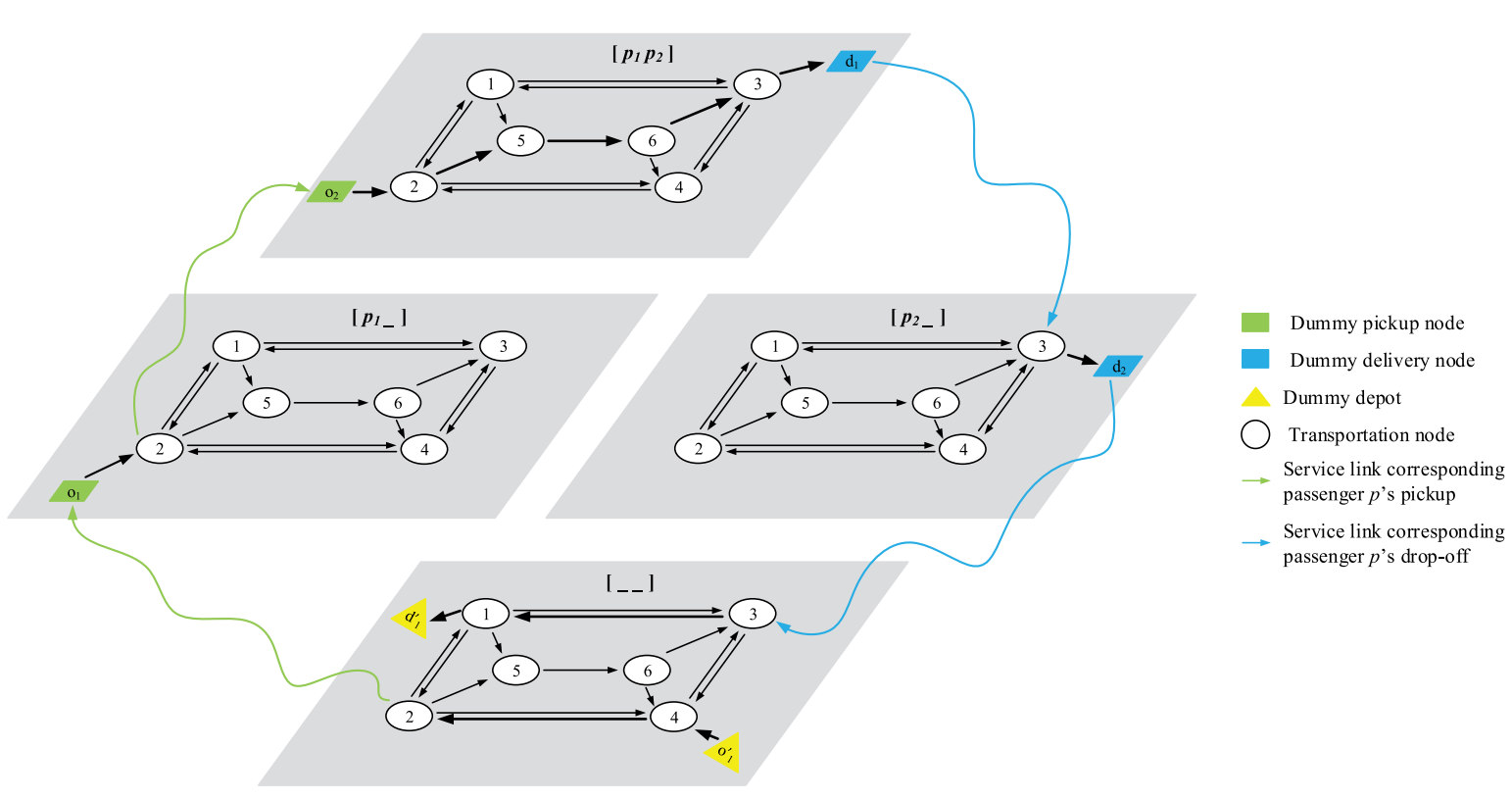


图3 状态转移

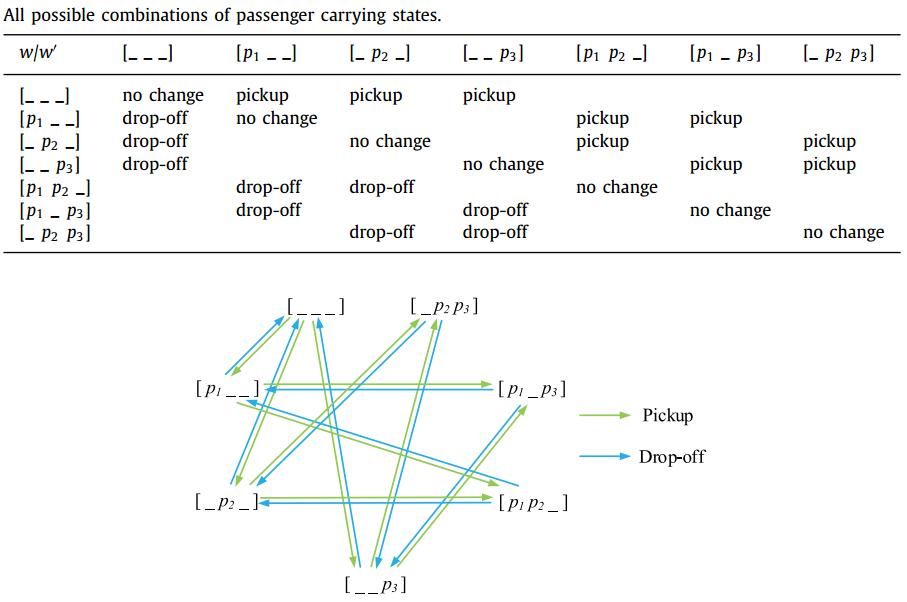
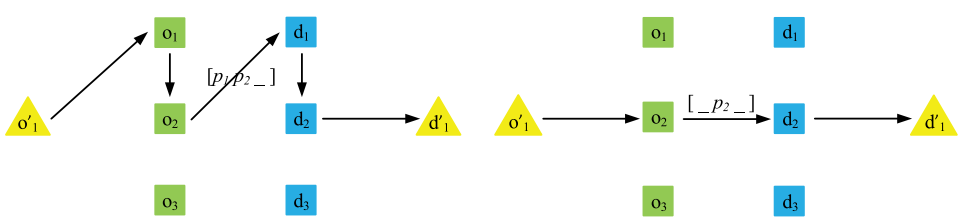


图4 状态转移

# 3、模型与算法说明

表1 符号定义

|  |  |
| --- | --- |
| 参数 | 定义 |
| *i, j* | 表示路段节点 |
| *t, s* | 时间参数 |
| *w, w’* | 状态参数 |
| *(i,t,w)* | VRPPDTW问题的起点 |
| *(i,j,t,s,w,w’)* | 节点 *(i,t,w)* 和 *(j,t,s)*之间的弧段 |
| *y(v,i,j,t,s,w,w’)* | 0-1变量 |

数学模型公式如图5所示。目标函数为车辆总运输成本最少。约束由流量平衡约束、乘客乘车需求约束以及0-1变量定义组成。0-1变量y(v,i,j,t,s,w,w’)表示车流v是否选择路段(i,j,t,s,w,w’)接送乘客或等待乘客。

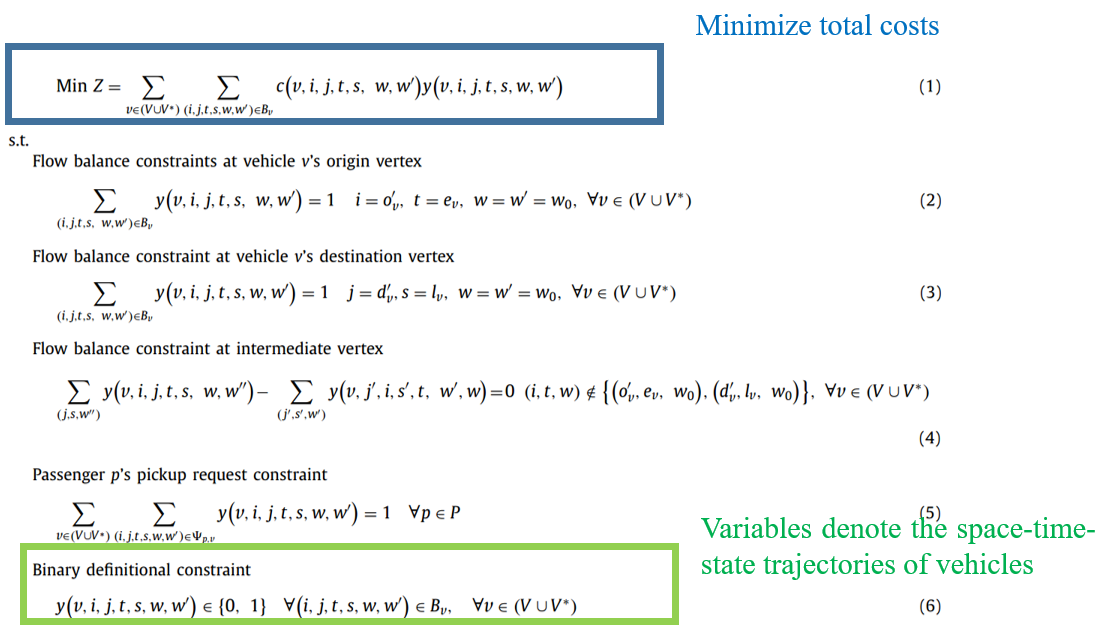


图5 VRPPDTW问题的模型

为了高效地求解模型，利用拉格朗日松弛算法，将乘客乘车需求约束转化为目标函数项。 因此，剩下的约束条件和新的目标函数变为与时间相关的最短路径问题。通过不断更新拉格朗日乘子，用动态规划方法求解子问题，迭代求解模型。新模型如图6所示。拉格朗日乘子可以看作服务乘客的盈利，用来吸引车辆为他们服务。

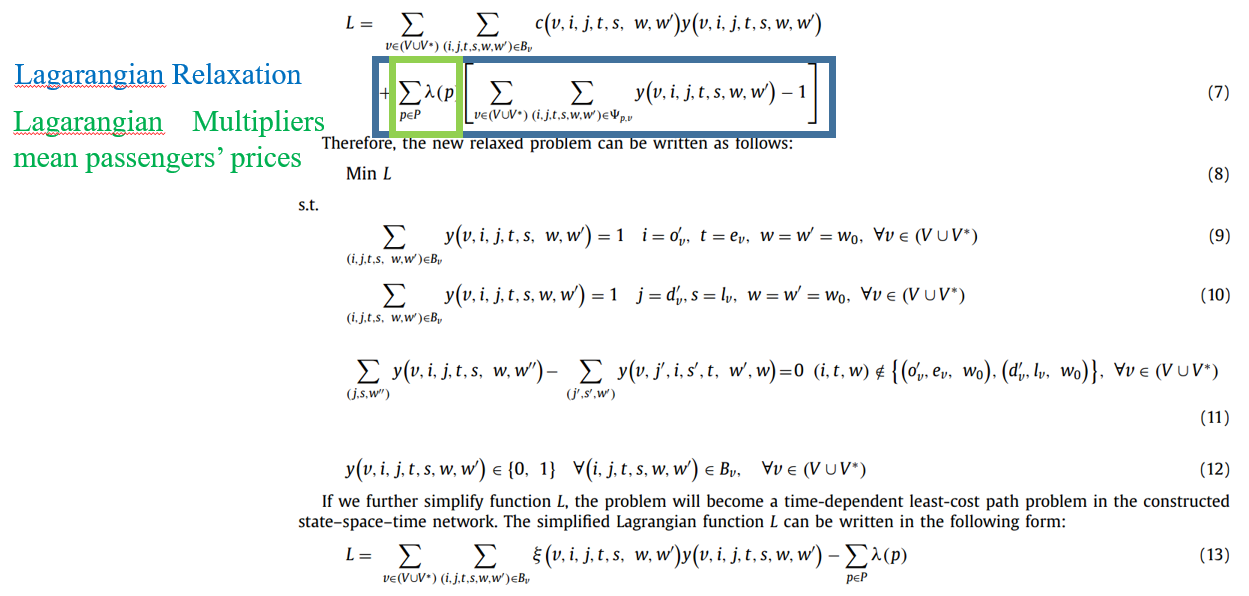


图6拉格朗日松弛模型

# 4、程序的执行

(1) 方式1

双击AgentPlus.sln文件打开程序，在属性设置中将工作目录改为输入数据集路径，如图7所示。点击本地调试器运行程序。

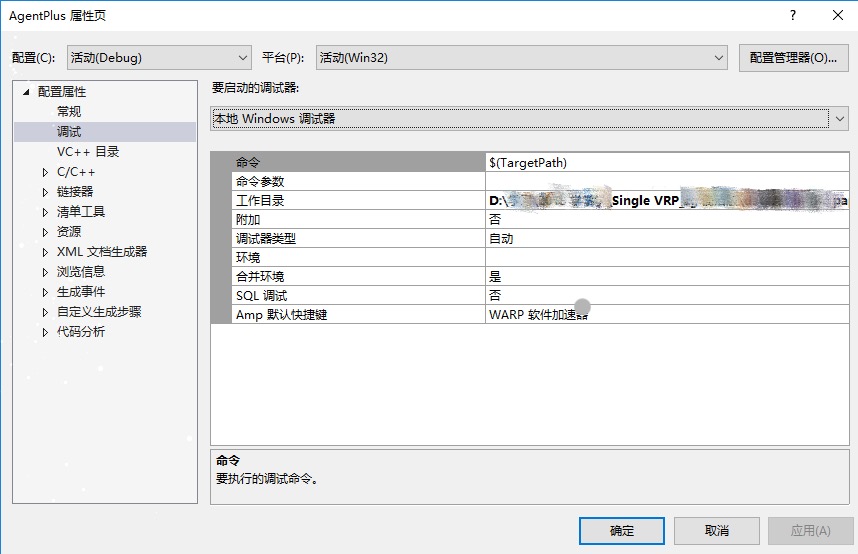


图7 修改工作目录

（2）方式2

将输入文件放入debug文件夹，直接双击exe文件即可运行程序得到结果。

# 5、程序的输入和输出

## 5.1 输入文件

表2 输入数据集

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 数据内容 | 关联文件名 | 重要属性 |
| 1 | 节点信息 | input\_node.csv | 编号 |
| 2 | 线路信息 | input\_link.csv | 端点、长度 |
| 3 | 车辆、乘客信息 | input\_agent.csv | 起讫点、时间窗 |

下面以数据集dataset1为例对输入数据量进行详述。物理路网如图8所示，圆圈内数字表示节点编号，连接线上的数字表示旅行时间。

（1）input\_node.csv

如图9所示，input\_node为物理路网节点信息的输入文件，包括的内容有：node\_id,x,y，node\_id是路网物理节点的编号。x,y 为点的横纵坐标。若在input\_link给出线路长度，x，y可以不填写，亦可利用点的x，y坐标计算线路长度。

（2）input\_link.csv

如图10所示，input\_link为物理路网线路信息的输入文件。包括的主要内容有：

① from\_node\_id：线路起点；

② to\_node\_id：线路终点；

③ direction：线路方向，1为正方向（from\_node→to\_node），-1为反方向（to\_node→from\_node）；

④ length：线路长度；

⑤ speed\_limit\_in\_mph：线路最高限速；

其余参数：number\_of\_lanes，lane\_capacity\_in\_vhc\_per\_hour，jam\_density本程序尚未考虑，用户可根据实际需求进行扩展。

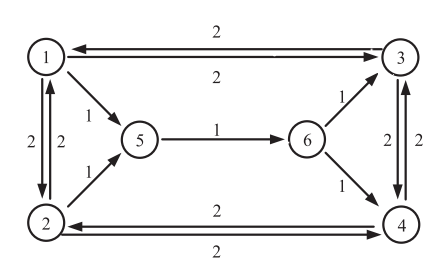
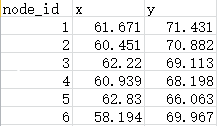
 

图8 物理路网 图9 input\_node.csv

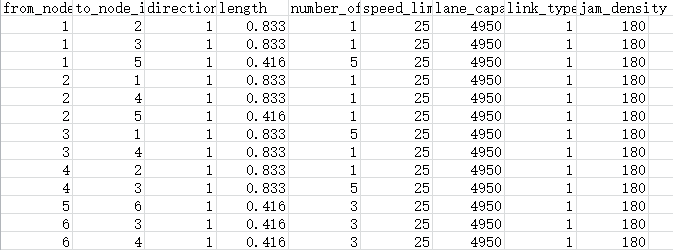


图10 input\_link.csv

（3）input\_agent.csv

如图11所示，input\_agent输入的是乘客和车辆信息。

① agent\_id：乘客/车辆的编号（分别编号）；

② agent\_type：0表示乘客，1表示车辆；

③ from\_node\_id, to\_node\_id：出发点和目的地；

④ departure\_time\_start,departure\_time\_window：最早出发时间和时间窗长度；

⑤ arrival\_time\_start, arrival\_time\_window：最早到达时间和时间窗长度；

⑥ capacity：车辆能力；

⑦ base\_profit：乘客的初始价格（拉格朗日乘子的相反数）。

VOIVTT\_per\_hour和VOWT\_per\_hour分别表示，本程序尚未考虑，用户可根据实际需求进行扩展。

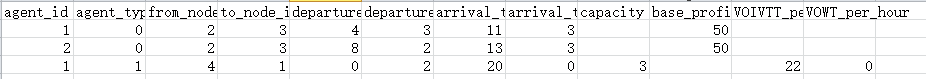


图11 input\_agent.csv

通过以上三个文件的输入，程序可构建出包括虚拟点和虚拟弧的虚拟路网，如图12所示。

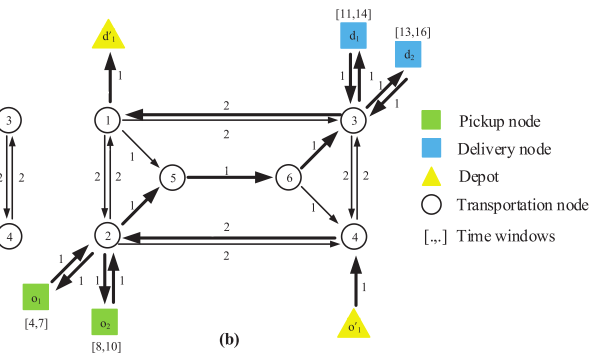


图12 虚拟路网

## 5.2 输出文件

表3 输出文件

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 数据内容 | 关联文件名 | 重要属性 |
| 1 | 节点信息 | output\_node.csv | 编号、属性 |
| 2 | 线路信息 | output\_link.csv | 端点、属性 |
| 3 | 车辆路径 | output\_agent.csv | 迭代次数、时空路径 |
| 4 | 乘客费用 | output\_paxprofitLog.csv | 迭代次数、乘客费用 |
| 5 | 计算结果 | output\_solution.csv | 最小费用、计算时间 |
| 6 | DP计算过程 | PathLog.csv | 时间、状态 |

（1）output\_node.csv

在程序的输入中，我们通过input\_node.csv输入了物理网络的节点信息，然而在模型的构建和算法的实现中，还需要添加虚拟节点：乘客接、送节点以及车辆的车站节点。根据输入的input\_node和input\_agent文件，程序构建出虚拟节点。

①node\_id：节点编号。编号顺序依次为物理节点、乘客1接取节点、乘客1送达节点、乘客2接取节点、乘客2送达节点……、车辆出发节点、车辆终到节点。

②node\_type：节点属性。0表示物理节点，1表示乘客接取节点，2为乘客送达节，3表示辆车的出发到达节点。

根据之前的输入，我们得到的output\_node.csv如图13所示。其中，点1-6为物理节点，点7-8为乘客1的接取送达节点，对应图12中的O1和D1，点9-10为乘客2的接取送达节点，对应o2和d2，点11-12 对应o’1和d’1.

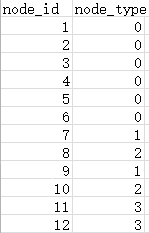
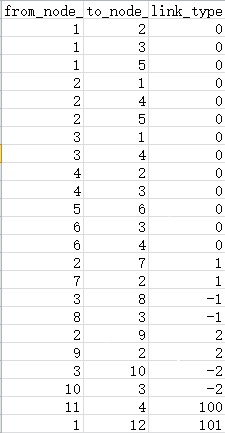


图13 output\_node.csv

（2）output\_link.csv

同理，除既有的输入物理线路之外，程序会根据input\_link和input\_agent构建虚拟弧，并输出output\_link.csv文件。 图14 output\_link.csv

① from\_node\_id 和to\_node\_id： 分别表示弧的起点和终点。

② link\_type：0表示物理弧，1表示乘客1的接取弧，既图12中点2与O1间对应的弧，-1表示乘客1的送达弧，既点3与d1的间对应的弧。同理，属性为2和-2的弧分别表示乘客2的接取、送达弧。100表示车辆出场弧，既o’1-点4对应的弧，101表示车辆回场弧，既点1-d’1对应的弧。

该测试数据集输出的out\_link.csv 如图14所示。

注：程序将虚拟弧的travel time 均设为1，可根据需要进行修改。

（3）output\_agent.csv

output\_agent.csv输出每一次迭代对应的时空路径，如图15所示。

① LR\_iteration：拉格朗日算法迭代次数；

② Stepsize：迭代步长；

③ path\_node\_seq：车辆行驶路径；

④ path\_ime\_sequence：车辆到达路径中各节点的时间；

⑤ upperbound：上界的计算结果。

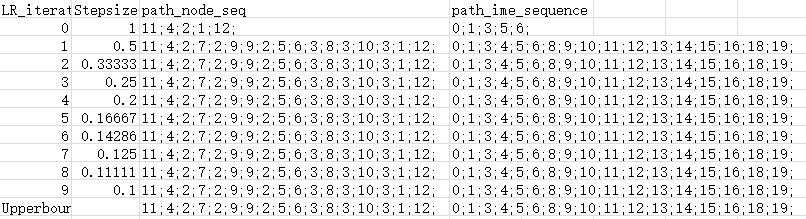


图15 Output agent.csv

（4）output\_paxprofitLog.csv

output\_ paxprofitLog.csv输出每一次迭代对应的乘客利润（模型中拉格朗日乘子的相反数）。如图16所示。

① iteration：迭代代数；

② stepSize：迭代步长；

③ p1' profit- p10' profit：在每一次迭代中各乘客对应的利润；

④ LowerBoundCost：下界计算得到的费用。

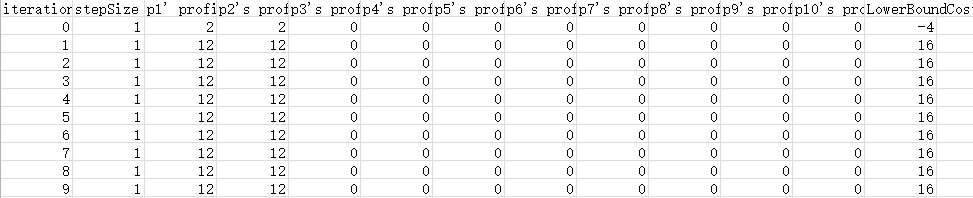


图16 Output\_paxprofitLog.csv

（5）PathLog.csv

用DP算法计算LR上界时，车辆最短路计算过程记录。

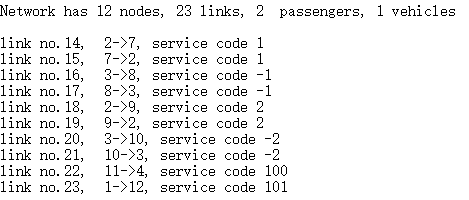
（6）Debug.txt

Debug.txt如图17所示，文件依次输出以下信息：

① 加入input\_agent信息后，生成的虚拟路网节点数量弧数量以及乘客和车辆数量；

② 生成虚拟弧的端点和属性；

③ 在LR和DP计算中的更新信息；



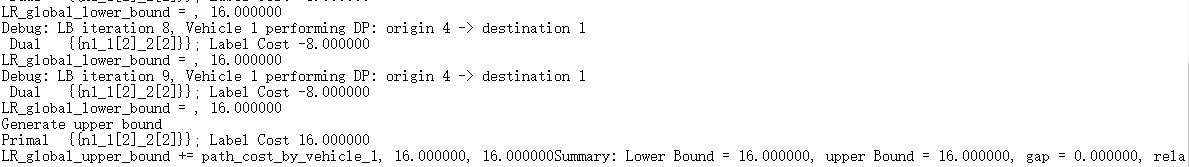


图17 Debug.txt

# 6、测试数据集

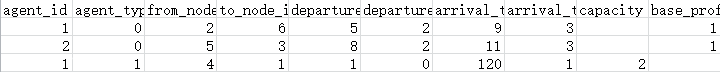
测试数据集储存在dataset文件夹中，6.1-6.4分别对应1v2p-share, 1v2p-seperate ,1v2p-1unserved, 1v3p-share。

## 6.1 1辆车同时服务两位乘客

物理路网不变，input\_node.csv和input\_link.csv文件和5.1示例相同。

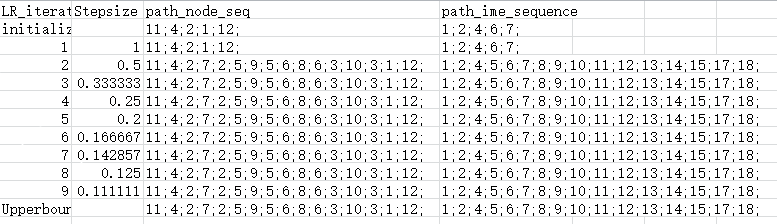
按照以下情景更改input\_agent.csv文件：

乘客1从节点2（时间窗 [5, 7]）前往节点6（时间窗 [9,12]），乘客2从节点5（时间窗 [8, 10]）前往节点3（时间窗 [11, 14]）。车辆1从节点4（时间窗 [1, 30]）前往到节点1（时间窗 [1, 30]）。



Input\_agent.csv

得到结果：



Output\_agent.csv

车辆运行路线如下图18所示。

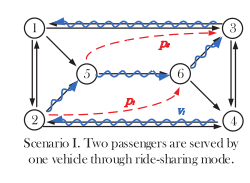
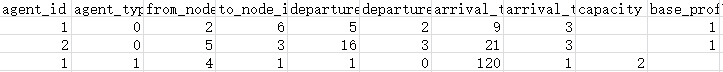


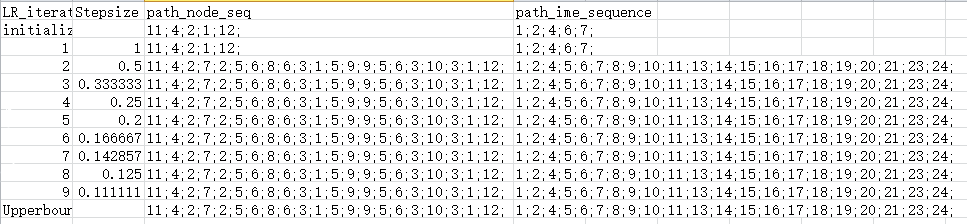
图18 算例1车辆走行路径

## 6.2 1辆车同时服务两位乘客（无合乘）

乘客1从节点2（时间窗 [5, 7]）前往节点6（时间窗 [9,12]），乘客2从节点5（时间窗 [16, 19]）前往节点3（时间窗 [21, 24]）。车辆1从节点4（时间窗 [1, 30]）前往到节点1（时间窗 [1, 30]）。



Input\_agent.csv



output\_agent.csv

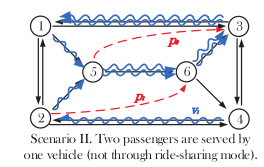
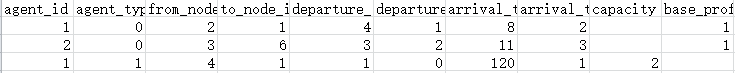


图19 算例2车辆走行路径

## 6.3 1辆车2位乘客，一人未被服务

乘客1从节点2（时间窗 [4, 5]）前往节点1（时间窗 [9,12]），乘客2从节点3（时间窗 [3, 5]）前往节点6（时间窗 [11, 14]）。车辆1从节点4（时间窗 [1, 30]）前往到节点1（时间窗 [1, 30]）。



Input\_agent.csv

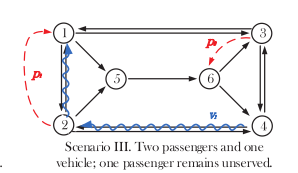
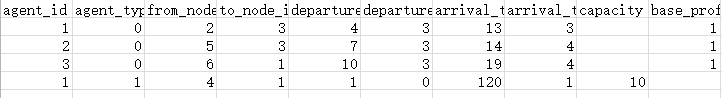


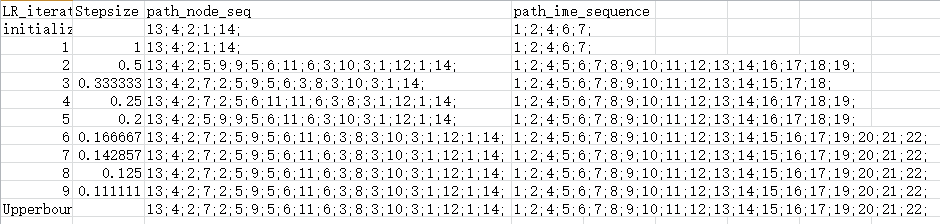
图20 算例3车辆走行路径

## 6.4 1辆车同时服务3名乘客

乘客1从节点2（时间窗 [4, 7]）前往节点3（时间窗 [13,16]），乘客2从节点5（时间窗 [7, 10]）前往节点3（时间窗 [14, 18]），乘客3从节点4（时间窗[10,13]）前往节点1（时间窗[19,23]）。车辆1从节点4（时间窗 [1, 30]）前往到节点1（时间窗 [1, 30]）。



Input\_agent.csv



ouput\_agent.csv

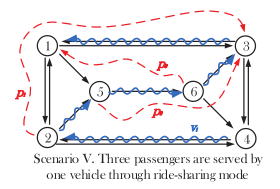


图21 算例4车辆走行路径

# 7、程序说明



程序包括三个主要部分，如流程图中三个虚线框中内容所示。

## 7.1 g\_ReadInputData()

读取input\_node.csv、input\_link.csv中物理节点和路网信息。

读取input\_agent.csv中的乘客，和车辆信息，生成虚拟节点和虚拟弧，构建出完整的虚拟路网。

## 7.2 拉格朗日松弛

g\_Optimization\_Lagrangian\_Method\_Vehicle\_Routing\_Problem\_Simple\_Variables（）即为拉格朗日松弛算法。

主要变量：

g\_best\_upper\_bound；g\_best\_lower\_bound 储存LR算法最终的最优上下界；

LR\_global\_upper\_bound , LR\_global\_lower\_bound每一次迭代中，用来储存上下界；

① 设置上下界和迭代次数，初始步长；

② 调用DP算法求解最短路问题

③ 计算当前迭代中的下界：

LR\_global\_lower\_bound += path\_cost\_by\_vehicle\_v

if passenger p is served：

LR\_global\_lower\_bound += g\_passenger\_base\_profit[p]

如果该次迭代中LR\_global\_lower\_bound优于g\_best\_lower\_bound，则更新

④ 对未被服务的乘客，增加他的profit以吸引车辆对其提供服务：

g\_passenger\_base\_profit[p] -= constant\*StepSize \* (g\_passenger\_number\_of\_visits[p] - 1)，回到步骤2直到完成迭代次数。

⑤ 当完成设置的迭代次数时，根据最后依次迭代的乘客费用，调用DP算法求解上界。

LR\_global\_upper\_bound += path\_cost\_by\_vehicle\_v;

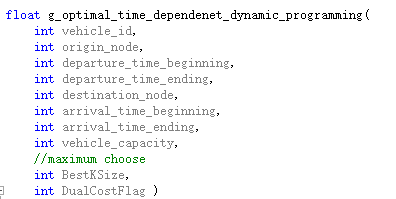
如果有乘客没有被服务，上界+30，（可以理解为出动虚拟车，虚拟车费用为30）：

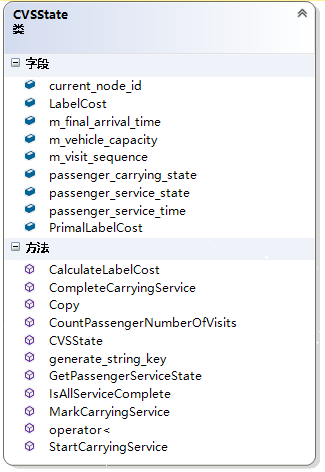
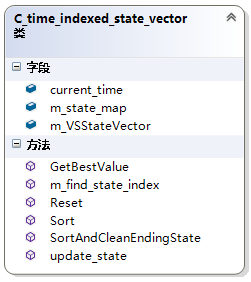
LR\_global\_upper\_bound += 30

⑥ 计算gap。

## 7.3 DP算法

程序利用前向动态规划算法求解车辆最短路径。动态规划的原理是：每个阶段的最优状态可以从之前某个阶段的某个或某些状态直接得到而不管之前这个状态是如何得到的。



在DP算法中，需要用到CVSState和C\_time\_indexed\_state\_vector两个类，类视图如图所示，

CVSState用来更新、记录车辆的位置、状态、费用。

C\_time\_indexed\_state\_vector用来对每个时间t上不同状态费用进行排序，在动态规划算法中，随着t的推移，每一个时间t存在的状态会越来越多，到达中间时（状态最多的部分），可能会发生数据爆炸使得无法求解。

因此在动态规划算法中我们，引入了参数**BestKSize**。通过对每一个时间t的状态的labelcost进行排序，只保留前BestKSize个状态，可以有效地控制问题规模。

最终DP返回车辆到达最终状态所需要的最小费用。

return g\_ending\_state\_vector[vehicle\_id].GetBestValue(DualCostFlag, vehicle\_id);

**参数：DualCostFlag**

当计算LR下界，调用DP算法时，参数DualCostFlag=1，path\_cost\_by\_vehicle\_v的返回值时LabelCost，即模型中的。

当计算LR上界，调用DP算法时，参数DualCostFlag=0，path\_cost\_by\_vehicle\_v的返回值时primalLabelCost，即模型中的cy。

[1] Mahmoudi M, Zhou X. Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state–space–time network representations[J]. Transportation Research Part B, 2016, 89(N°2,):19-42.