Single Vehicle

VRPPDTW 问题

C++程序 使用说明

2018.01

目录

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。

1.1 传统的 VRP 问题

VRP 问题在 1959 年首次由 G. Dantzig 和 J. Ramser 提出，传统的 VRP 问题可以概括为：一个车队为一定数量的乘客提供服务，我们需要在车辆的容量限制约束下，找到一系列车辆径路（从停车场出发，完成对乘客的服务后再回到停车场），达到诸如路程最短、成本最小、耗费时间最少等目的。由定义不难看出，旅行商问题（Traveling Saleman Problem,TSP）是 VRP 的特例， TSP 问题已被证明是 NP 难题问题，因此，VRP 也属于 NP 难题。

车辆路线的实际问题包括配送中心配送、公共汽车路线制定、信件和报纸投递、航空和铁路时间表安排、工业废品收集等。

1.2 带有时间窗的 VRP 问题 (VRPTW)

当所有乘客的服务都需在一个给定的时间区间（称为时间窗）内进行，就是带有时间窗的 VRP 问题，这就是一般 VRP 问题的扩展。时间窗可以视为软约束或硬约束。当所有的服务必须在给定的时间窗内进行时，是硬约束；当对时间窗外的服务加以惩罚值时，是软约束。

1.3 带有接、送的 VRP 问题(VRPPD)

传统的 VRP 问题只考虑人或货物“聚集”或“配送”的单一过程，在车辆行驶的过程中，其载客/货量的变化是单调的。而带有接、送的 VRP 问题(VRPPD)

问题集接取和送达服务于一体，需要根据客户的需求在指定的起点和终点进行，既在做路径规划时同时考虑乘客的接送或货物的集货和配送。

1.4 带有接、送、时间窗的 VRP 问题(VRPPDTW)

带有接、送、时间窗的 VRP 问题(VRPPDTW)是 1.3 和 1.4 的整合，所有乘客或货物需要在指定的时间窗内被接到，并在指定时间窗内送至目的地。同时，每个车辆也有自己的离开停车场和回到停车场的约束，以及容量限制约束。本程序正是解决此类问题。

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 d_1。

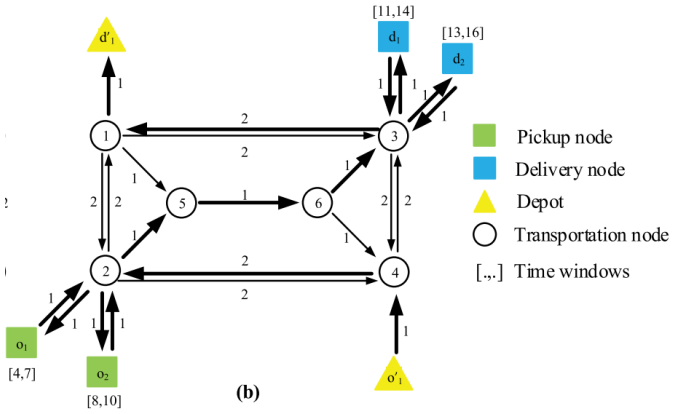


图 1 VRPPDTW 网络的节点

（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。

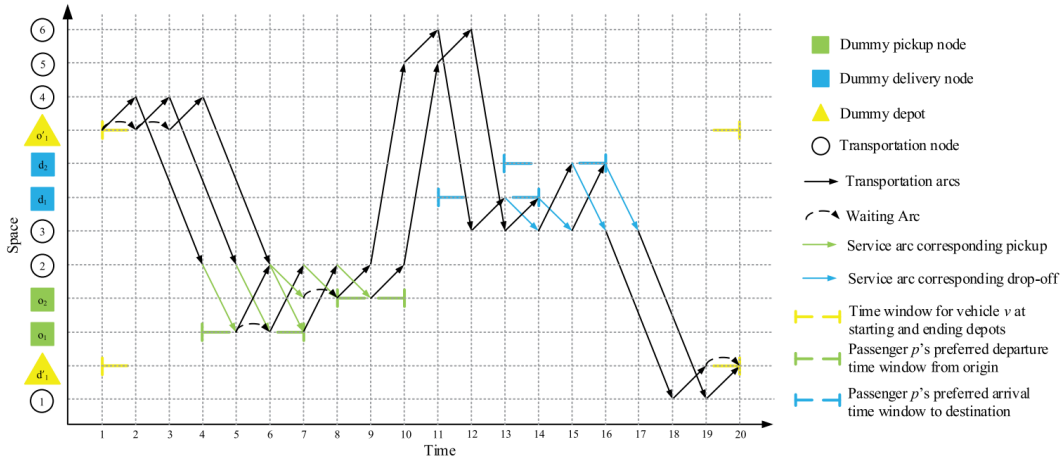


Fig. 2. Shortest paths with node sequence $(o'_1, 4, 2, o_1, 2, o_2, 2, 5, 6, 3, d_1, 3, d_2, 3, 1, d'_1)$ in vehicle 1's space–time network.

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。

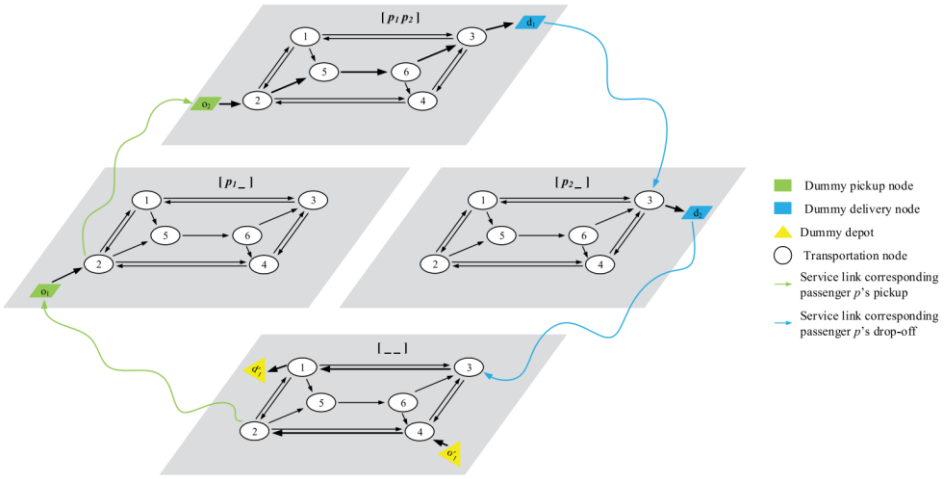
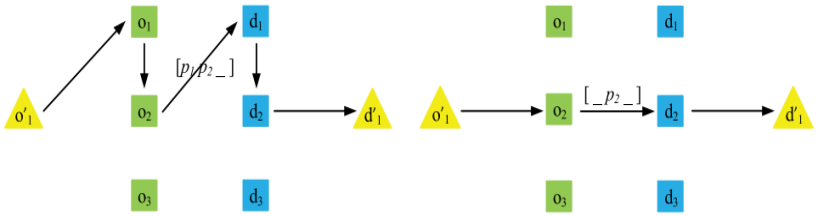


图 3 状态转移



All possible combinations of passenger carrying states.

w/w′	[_ _ _]	[p_1 _ _]	[_ p_2 _]	[_ _ p_3]	[p_1 p_2 _]	[p_1 _ p_3]	[_ p_2 p_3]
[_ _ _]	no change	pickup	pickup	pickup			
[p_1 _ _]	drop-off	no change			pickup	pickup	
[_ p_2 _]	drop-off		no change		pickup		pickup
[_ _ p_3]	drop-off			no change		pickup	pickup
[p_1 p_2 _]		drop-off	drop-off		no change		
[p_1 _ p_3]		drop-off		drop-off		no change	
[_ p_2 p_3]			drop-off	drop-off			no change

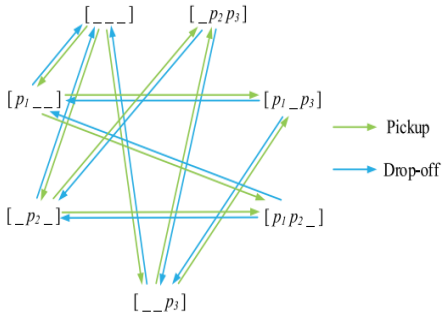


图 4 状态转移

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。

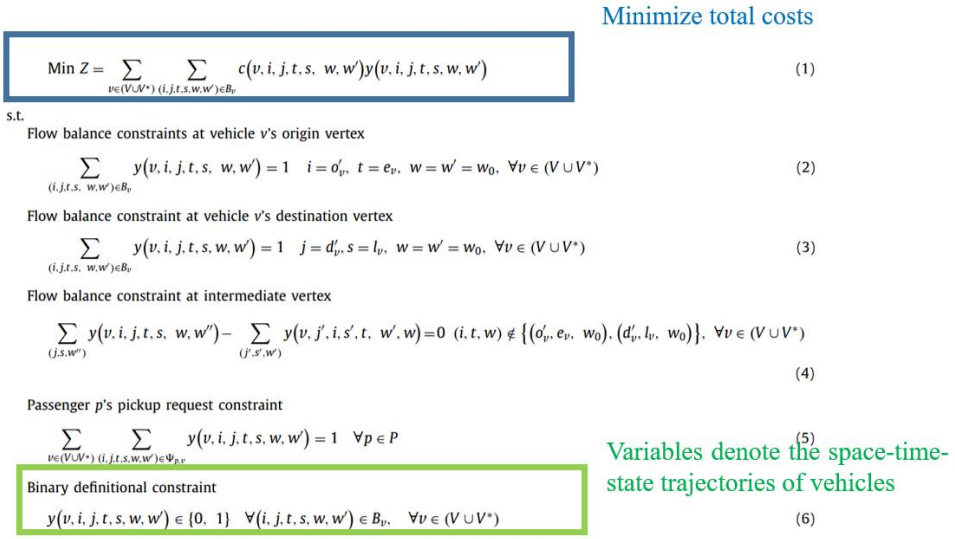


图 5 VRPPDTW 问题的模型

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，其余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。

Lagarangian Relaxation

Lagarangian Multipliers mean passengers' prices

$$L = \sum_{v\in(V\cup V^*)} \sum_{(i,j,t,s,w,w')\in B_v} c(v,i,j,t,s,\ w,w')y(v,i,j,t,s,w,w') + \sum_{p\in P}\lambda(p)\left[\sum_{v\in(V\cup V^*)} \sum_{(i,j,t,s,w,w')\in \Psi_{p,v}} y(v,i,j,t,s,w,w') - 1\right] \quad (7)$$

Therefore, the new relaxed problem can be written as follows:

$$\text{Min } L \quad (8)$$

s.t.

$$\sum_{(i,j,t,s,\ w,w')\in B_v} y(v,i,j,t,s,\ w,w') = 1 \quad i = o'_v,\ t = e_v,\ w = w' = w_0,\ \forall v \in (V\cup V^*) \quad (9)$$

$$\sum_{(i,j,t,s,\ w,w')\in B_v} y(v,i,j,t,s,w,w') = 1 \quad j = d'_v, s = l_v,\ w = w' = w_0,\ \forall v \in (V\cup V^*) \quad (10)$$

$$\sum_{(j,s,w'')} y(v,i,j,t,s,\ w,w'') - \sum_{(j',s',w')} y(v,j',i,s',t,\ w',w) = 0\ \ (i,t,w) \notin \{(o'_v,e_v,\ w_0),(d'_v,l_v,\ w_0)\},\ \forall v \in (V\cup V^*) \quad (11)$$

$$y(v,i,j,t,s,w,w') \in \{0,\ 1\}\quad \forall (i,j,t,s,w,w') \in B_v,\quad \forall v \in (V\cup V^*) \quad (12)$$

If we further simplify function L, the problem will become a time-dependent least-cost path problem in the constructed state-space-time network. The simplified Lagrangian function L can be written in the following form:

$$L = \sum_{v\in(V\cup V^*)} \sum_{(i,j,t,s,w,w')\in B_v} \xi(v,i,j,t,s,\ w,w')y(v,i,j,t,s,w,w') - \sum_{p\in P}\lambda(p) \quad (13)$$

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。

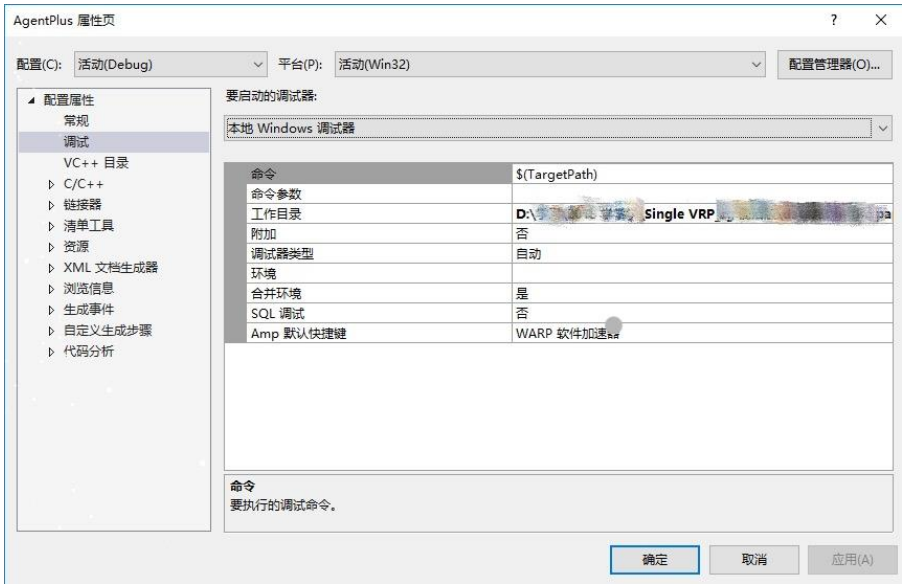


图 7 修改工作目录

（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。

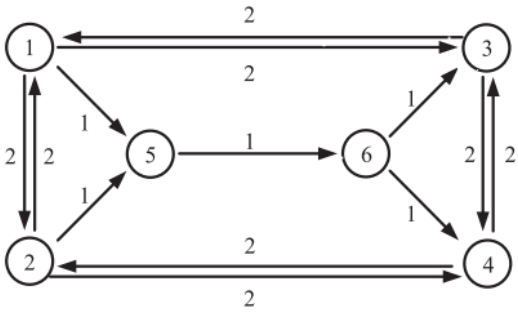


图 8　物理路网

node_id	x	y
1	61.671	71.431
2	60.451	70.882
3	62.22	69.113
4	60.939	68.198
5	62.83	66.063
6	58.194	69.967

图 9 input_node.csv

from_node	to_node_i	direction	length	number_of	speed_lim	lane_capa	link_type	jam_density
1	2	1	0.833	1	25	4950	1	180
1	3	1	0.833	1	25	4950	1	180
1	5	1	0.416	5	25	4950	1	180
2	1	1	0.833	1	25	4950	1	180
2	4	1	0.833	1	25	4950	1	180
2	5	1	0.416	1	25	4950	1	180
3	1	1	0.833	5	25	4950	1	180
3	4	1	0.833	1	25	4950	1	180
4	2	1	0.833	1	25	4950	1	180
4	3	1	0.833	5	25	4950	1	180
5	6	1	0.416	3	25	4950	1	180
6	3	1	0.416	3	25	4950	1	180
6	4	1	0.416	3	25	4950	1	180

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。

agent_id	agent_typ	from_node	to_node_i	departure	departure	arrival_t	arrival_t	capacity	base_profi	VOIVTT_pe	VOWT_per_hour
1	0	2	3	4	3	11	3		50		
2	0	2	3	8	2	13	3		50		
1	1	4	1	0	2	20	0	3		22	0

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。

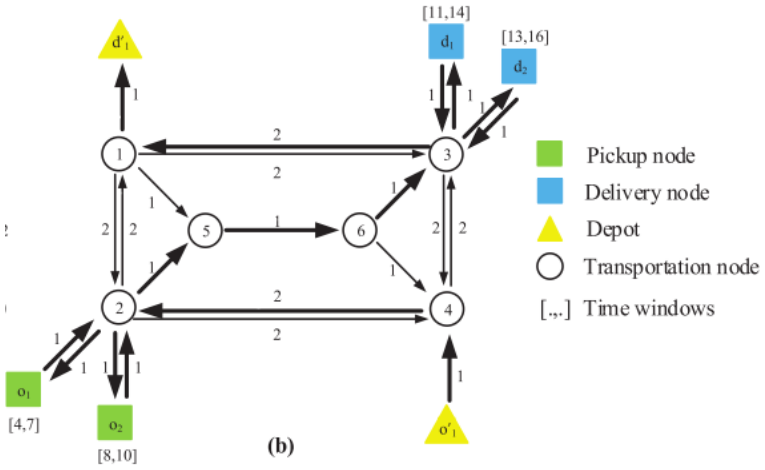


图 12　虚拟路网

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.

node_id	node_type
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	2
9	1
10	2
11	3
12	3

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。

from_node_	to_node_	link_type
1	2	0
1	3	0
1	5	0
2	1	0
2	4	0
2	5	0
3	1	0
3	4	0
4	2	0
4	3	0
5	6	0
6	3	0
6	4	0
2	7	1
7	2	1
3	8	-1
8	3	-1
2	9	2
9	2	2
3	10	-2
10	3	-2
11	4	100
1	12	101

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。

LR_iterat	Stepsize	path_node_seq	path_ime_sequence
initializ		11;4;2;1;12;	0;1;3;5;6;
1	1	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
2	0.5	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
3	0.333333	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
4	0.25	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
5	0.2	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
6	0.166667	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
7	0.142857	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
8	0.125	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
9	0.111111	11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;
Upperbour		11;4;2;7;2;9;9;2;5;6;3;8;3;10;3;1;12;	0;1;3;4;5;6;8;9;10;11;12;13;14;15;16;18;19;

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。

iteratior	stepSize	p1' profi	p2's profi	p3's profi	p4's profi	p5's profi	p6's profi	p7's profi	p8's profi	p9's profi	p10's pr	LowerBoundCost
initializ		2	2	0	0	0	0	0	0	0	0	4
1	1	12	12	0	0	0	0	0	0	0	0	16
2	0.5	12	12	0	0	0	0	0	0	0	0	16
3	0.333333	12	12	0	0	0	0	0	0	0	0	16
4	0.25	12	12	0	0	0	0	0	0	0	0	16
5	0.2	12	12	0	0	0	0	0	0	0	0	16
6	0.166667	12	12	0	0	0	0	0	0	0	0	16
7	0.142857	12	12	0	0	0	0	0	0	0	0	16
8	0.125	12	12	0	0	0	0	0	0	0	0	16
9	0.111111	12	12	0	0	0	0	0	0	0	0	16

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。

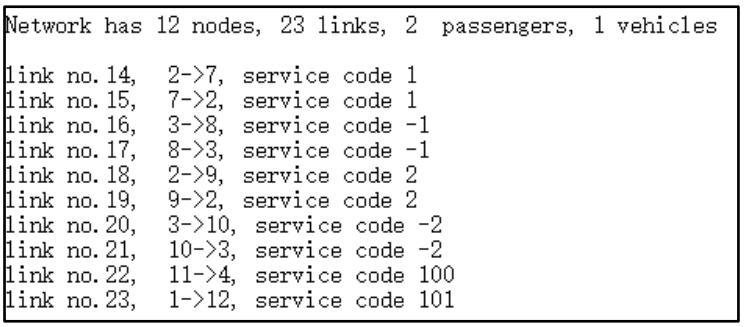
（6）Debug.txt

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；



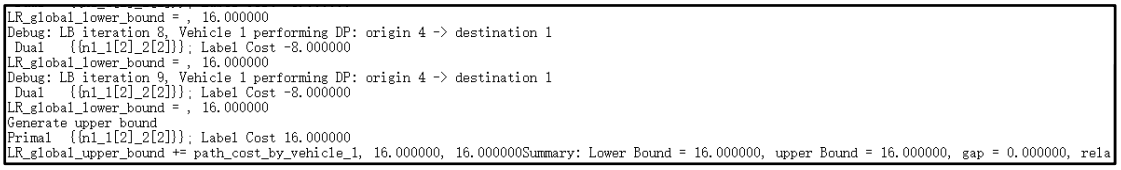


图 17 Debug.txt

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]）。

agent_id	agent_typ	from_node	to_node_i	departure	departure	arrival_t	arrival_t	capacity	base_prof
1	0	2	6	5	2	9	3		1
2	0	5	3	8	2	11	3		1
1	1	4	1	1	0	120	1	2	

Input_agent.csv

得到结果：

LR_iterat	Stepsize	path_node_seq	path_ime_sequence
initializ		11;4;2;1;12;	1;2;4;6;7;
1	1	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
2	0.5	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
3	0.333333	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
4	0.25	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
5	0.2	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
6	0.166667	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
7	0.142857	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
8	0.125	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
9	0.111111	11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
Upperbour		11;4;2;7;2;5;9;5;6;8;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;

Output_agent.csv

车辆运行路线如下图 18 所示。

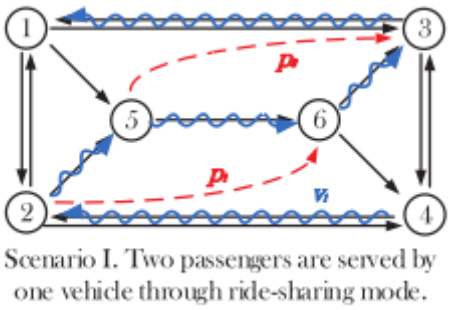


图 18 算例 1 车辆走行路径

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]）。

agent_id	agent_typ	from_node	to_node_i	departure	departure	arrival_t	arrival_t	capacity	base_prof
1	0	2	6	5	2	9	3		1
2	0	5	3	16	3	21	3		1
1	1	4	1	1	0	120	1	2	

Input_agent.csv

LR_iterat	Stepsize	path_node_seq	path_ime_sequence
initializ		11;4;2;1;12;	1;2;4;6;7;
1	1	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
2	0.5	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
3	0.333333	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
4	0.25	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
5	0.2	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
6	0.166667	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
7	0.142857	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
8	0.125	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
9	0.111111	11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;
Upperbour		11;4;2;7;2;5;6;8;6;3;1;5;9;9;5;6;3;10;3;1;12;	1;2;4;5;6;7;8;9;10;11;13;14;15;16;17;18;19;20;21;23;24;

output_agent.csv

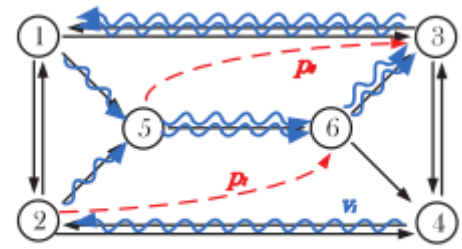


图 19 算例 2 车辆走行路径

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]）。

agent_id	agent_typ	from_node	to_node_i	departure_	departure	arrival_t	arrival_t	capacity	base_prof
1	0	2	1	4	1	8	2		1
2	0	3	6	3	2	11	3		1
1	1	4	1	1	0	120	1	2	

Input_agent.csv

当我们将 Constant 取 10，初始步长为 1 对乘客乘子进行更新，计算得到以下结果：

iteratio	stepSize	p1'	profip2's	profip3'	p4	p5	p6	p7'	p8'	p9'	p10	LowerBoundCost
0	1	1	1	0	0	0	0	0	0	0	0	7
1	1	11	11	0	0	0	0	0	0	0	0	20.3
2	0.5	11	16	0	0	0	0	0	0	0	0	25.3
3	0.333333	11	19.33333	0	0	0	0	0	0	0	0	25.6
4	0.25	13.5	19.33333	0	0	0	0	0	0	0	0	28.1
5	0.2	15.5	19.33333	0	0	0	0	0	0	0	0	28.63334
6	0.166667	15.5	21	0	0	0	0	0	0	0	0	30.1
7	0.142857	16.92857	21	0	0	0	0	0	0	0	0	30.3
8	0.125	16.92857	22.25	0	0	0	0	0	0	0	0	31.52857
9	0.111111	18.03968	22.25	0	0	0	0	0	0	0	0	31.55
10	0.1	18.03968	23.25	0	0	0	0	0	0	0	0	32.55
11	0.1	18.03968	24.25	0	0	0	0	0	0	0	0	32.63968
12	0.1	19.03968	24.25	0	0	0	0	0	0	0	0	33.55
13	0.1	19.03968	25.25	0	0	0	0	0	0	0	0	33.63968
14	0.1	20.03968	25.25	0	0	0	0	0	0	0	0	34.55
15	0.1	20.03968	26.25	0	0	0	0	0	0	0	0	34.63968
16	0.1	21.03968	26.25	0	0	0	0	0	0	0	0	35.55
17	0.1	21.03968	27.25	0	0	0	0	0	0	0	0	35.63968

LR_iterat	Stepsize	path_node_seq	path_ime_sequence
0	1	11;4;2;1;12;	1;2;4;6;7;
1	1	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
2	0.5	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
3	0.333333	11;4;3;9;3;1;5;6;10;6;3;1;12;	1;2;4;5;6;8;9;10;11;12;13;15;16;
4	0.25	11;4;3;9;3;1;5;6;10;6;3;1;12;	1;2;4;5;6;8;9;10;11;12;13;15;16;
5	0.2	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
6	0.166667	11;4;3;9;3;1;5;6;10;6;3;1;12;	1;2;4;5;6;8;9;10;11;12;13;15;16;
7	0.142857	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
8	0.125	11;4;3;9;3;1;5;6;10;6;3;1;12;	1;2;4;5;6;8;9;10;11;12;13;15;16;
9	0.111111	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
10	0.1	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
11	0.1	11;4;3;9;3;1;5;6;10;6;3;1;12;	1;2;4;5;6;8;9;10;11;12;13;15;16;
12	0.1	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
13	0.1	11;4;3;9;3;1;5;6;10;6;3;1;12;	1;2;4;5;6;8;9;10;11;12;13;15;16;
14	0.1	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
15	0.1	11;4;3;9;3;1;5;6;10;6;3;1;12;	1;2;4;5;6;8;9;10;11;12;13;15;16;
16	0.1	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
17	0.1	11;4;3;9;3;1;5;6;10;6;3;1;12;	1;2;4;5;6;8;9;10;11;12;13;15;16;
18	0.1	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;

Output_ paxprofitLog.csv　　Output_agent.csv

未被服务的乘客会提高自己的 profit 来吸引车辆，因此得到的结果的，车辆在乘客 1 和乘客 2 之间“摇摆不定。车辆服务乘客 1 的成本是 9，服务乘客 2 的成本是 14，只有当乘客 2 的出价比乘客 1 多出 5 以上时，车辆才会服务乘客 2。

然而，在原问题中（不含拉格朗日乘子），由于车辆服务乘客 2 的成本高于乘客 1，因此服务乘客 1 才是最优解。在上例中我们未得到最优解的原因是 Constant 选得过大，使得乘客 1 和乘客 2 之间本来存在的成本差异变得不明显。

当我们将 Constant 改为 3，初始步长为 1 进行计算，得到的结果为车辆服务乘客 1，因此在解决实际问题时，合理地设计拉格朗日乘子的迭代步长很重要。

iteration	stepSize	p1'	profip2's	profp3'	p4'	p5'	p6	p7	p8'	p9'	p10	LowerBoundCost
0	1	1	1	0	0	0	0	0	0	0	0	7
1	1	4	4	0	0	0	0	0	0	0	0	13
2	0.5	5.5	5.5	0	0	0	0	0	0	0	0	14.8
3	0.333333	5.5	6.5	0	0	0	0	0	0	0	0	15.8
4	0.25	5.5	7.25	0	0	0	0	0	0	0	0	16.55
5	0.2	5.5	7.85	0	0	0	0	0	0	0	0	17.15
6	0.166667	5.5	8.35	0	0	0	0	0	0	0	0	17.65
7	0.142857	5.5	8.778572	0	0	0	0	0	0	0	0	18.07857
8	0.125	5.5	9.153572	0	0	0	0	0	0	0	0	18.45357
9	0.111111	5.5	9.486905	0	0	0	0	0	0	0	0	18.78691

Output_ paxprofitLog.csv

LR_iterat	Stepsize	path_node_seq	path_ime_sequence
0	1	11;4;2;1;12;	1;2;4;6;7;
1	1	11;4;2;1;12;	1;2;4;6;7;
2	0.5	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
3	0.333333	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
4	0.25	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
5	0.2	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
6	0.166667	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
7	0.142857	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
8	0.125	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
9	0.111111	11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;
Upperbour		11;4;2;7;2;1;8;1;12;	1;2;4;5;6;8;9;10;11;

Output_agent.csv

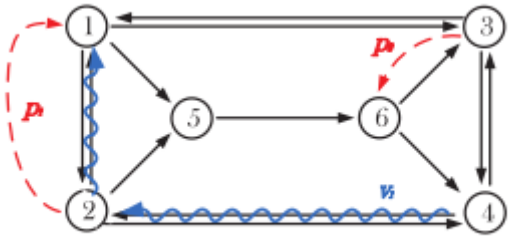


图 20 算例 3 车辆走行路径

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]）。

agent_id	agent_typ	from_node	to_node_i	departure	departure	arrival_t	arrival_t	capacity	base_prof
1	0	2	3	4	3	13	3		1
2	0	5	3	7	3	14	4		1
3	0	6	1	10	3	19	4		1
1	1	4	1	1	0	120	1	10	

Input_agent.csv

LR_iterat	Stepsize	path_node_seq	path_ime_sequence
initializ		13;4;2;1;14;	1;2;4;6;7;
1	1	13;4;2;5;9;9;5;6;11;6;3;10;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;16;17;18;19;
2	0.5	13;4;2;7;2;5;9;5;6;3;8;3;10;3;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;17;18;
3	0.333333	13;4;2;7;2;5;6;11;11;6;3;8;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;16;17;18;19;
4	0.25	13;4;2;7;2;5;9;5;6;11;6;3;8;3;10;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;16;17;19;20;21;22;
5	0.2	13;4;2;7;2;5;9;5;6;11;6;3;8;3;10;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;16;17;19;20;21;22;
6	0.166667	13;4;2;7;2;5;9;5;6;11;6;3;8;3;10;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;16;17;19;20;21;22;
7	0.142857	13;4;2;7;2;5;9;5;6;11;6;3;8;3;10;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;16;17;19;20;21;22;
8	0.125	13;4;2;7;2;5;9;5;6;11;6;3;8;3;10;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;16;17;19;20;21;22;
9	0.111111	13;4;2;7;2;5;9;5;6;11;6;3;8;3;10;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;16;17;19;20;21;22;
Upperbour		13;4;2;7;2;5;9;5;6;11;6;3;8;3;10;3;1;12;1;14;	1;2;4;5;6;7;8;9;10;11;12;13;14;15;16;17;19;20;21;22;

ouput_agent.csv

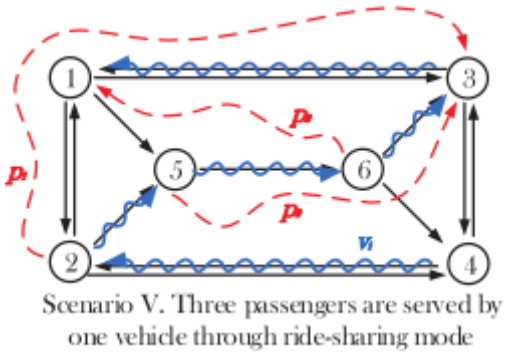
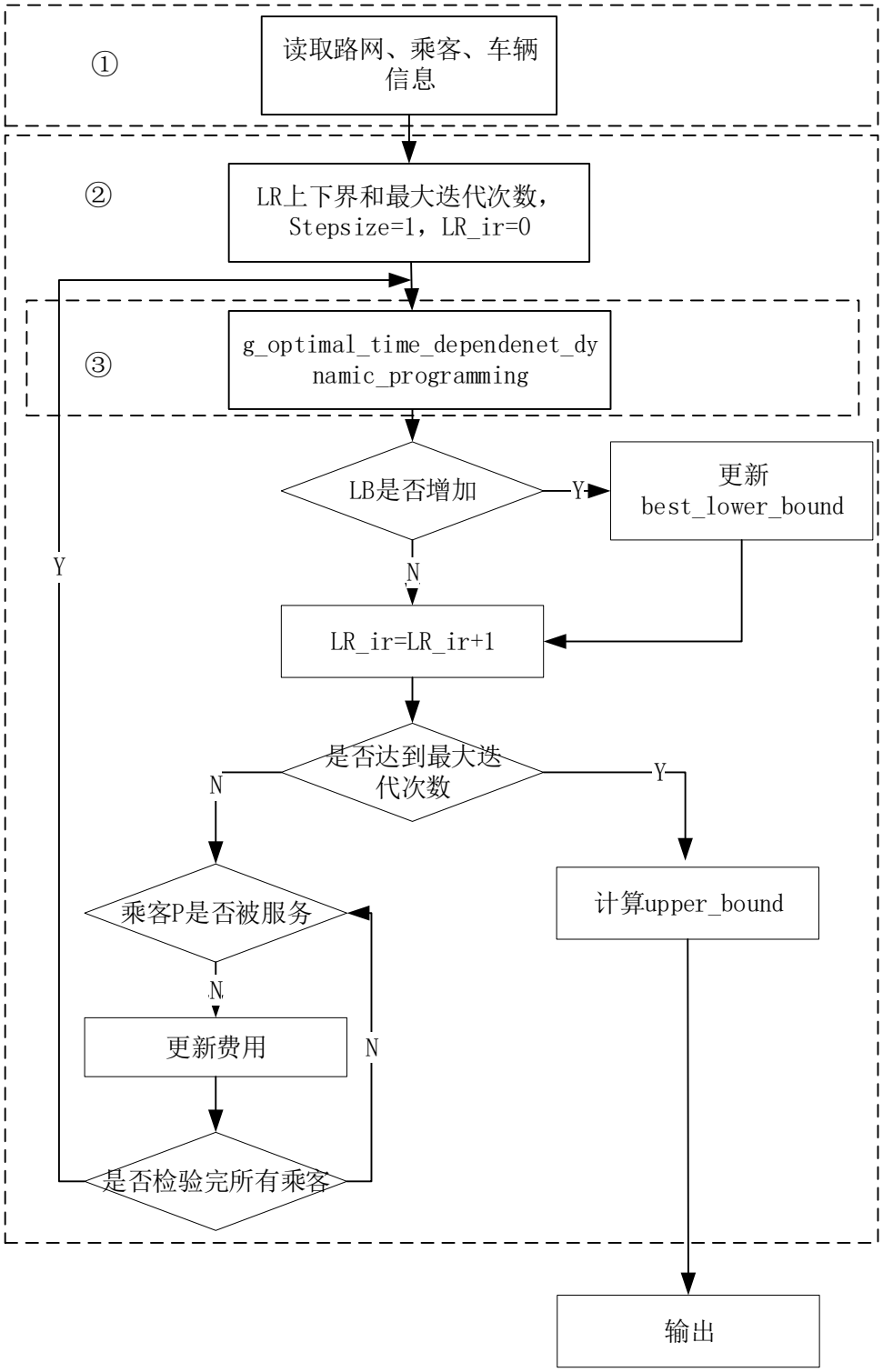


图 21 算例 4 车辆走行路径

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。

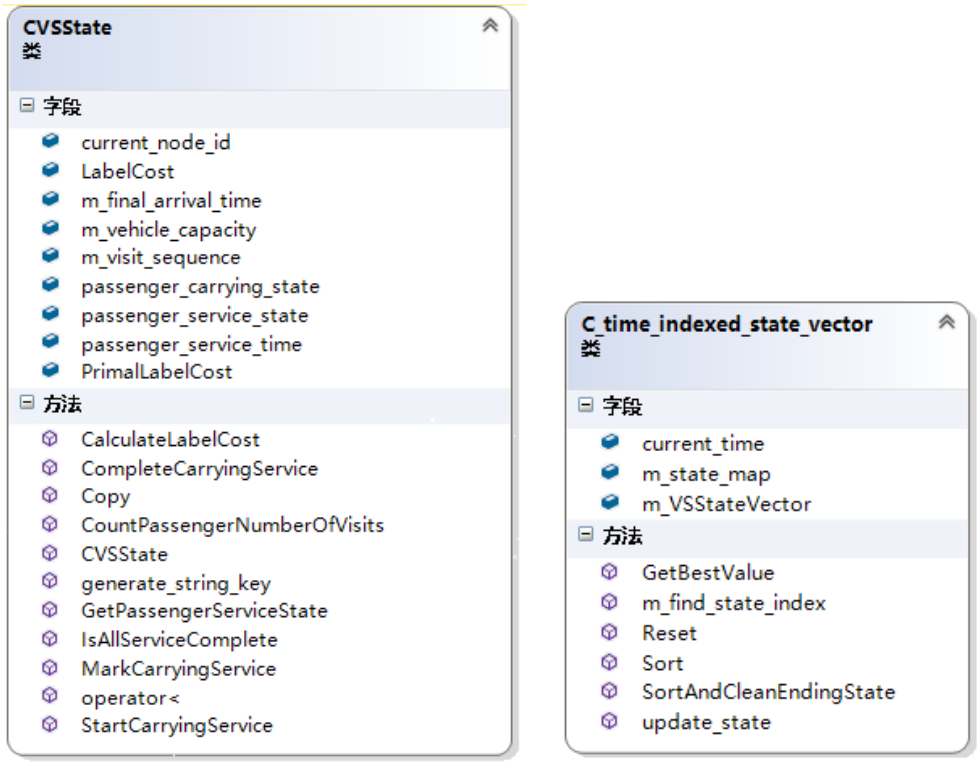
7.1 g_ReadInputData()

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 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、费用等信息。

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。

我们采用前向动态规划来计算一辆车最短路径的最小成本。步骤如下：

```
For vehicle v
    For time t
        for state w //当前时间 t 前 BestKSize 个状态
            for outbound of current w's node i
            calculate labelCost of (i,t,w):
            if(the state does not exist)
               create it
            else if(the label cost of the temp state < label cost of the existing state)
                update the cost of the state
```

在 *for outbound of current w's node i* 中，分以下几类情况：

（1）对于可能发生状态变化的：

①判断是否超出时间窗结束时间；

② feasible state transitions 发生状态变化:

```
if ((to_node_type == 1 && pElement->GetPassengerServiceState(to_node_passenger_id) == 0)//pickup
    || (to_node_type == 2 && pElement->GetPassengerServiceState(to_node_passenger_id) == 1))  // delivery)
```

a. next_time 小于 pickup 时间窗的开始时间；

b. next_time 小于 delivery 时间窗的开始时间；

c. next_time 整好处于 pickup 或 delivery 时间窗内。

（2）到达最终状态

（3）普通 physical link

//根据以上各种情形分类，对 CVSState 进行更新

```
CVSState new_element;
new_element.Copy(pElement);
new_element.MarkCarryingService(to_node_passenger_id, to_node_type, next_time);

new_element.current_node_id = to_node;
new_element.passenger_service_state[to_node_passenger_id] = to_node_type;
new_element.m_visit_time_sequence.push_back(next_time);
new_element.m_visit_sequence.push_back(to_node);

new_element.CalculateLabelCost(vehicle_id);

int link_no = g_outbound_link_no[from_node][i];

g_time_dependent_state_vector[vehicle_id][next_time].update_state(new_element);
}
```

CVSState 更新

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);
```

其中，CalculateLabelCost 根据目标函数来确定，见 `void CalculateLabelCost(int vehicle_id)`

本程序中，LabelCost 包括三部分：

① vehicle transportation cost：费用为 1；

② vehicle waiting cost：费用为 0.5；

③ passenger waiting cost: 费用为 0.3。

各部分的费用可以根据需要自行设定。

参数 DualCostFlag 说明：

当计算 LR 下界，调用 DP 算法时，参数 DualCostFlag=1，path_cost_by_vehicle_v 的返回值为 LabelCost，即模型中的 $cy-\lambda y$。

当计算 LR 上界，调用 DP 算法时，参数 DualCostFlag=0，path_cost_by_vehicle_v 的返回值为 primalLabelCost，即模型中的 cy。

7.3 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

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。

说明：在计算上下界调用 DP 算法时，返回的 path_cost_by_vehicle_v 不同，下界 path_cost_by_vehicle_v 的返回值是 $cy-\lambda y$，所以我们还需要加 $\sum\lambda$ 来计算下界。

上界 path_cost_by_vehicle_v 的返回值是 cy，我们只需要判断是否有乘客没有被服务，是否需要出动虚拟车。

8、Tips

VRPPDTW 的求解包含了多种运筹学基本问题和算法，熟悉以下子问题并融会贯通，有助于更好地理解 VRPPDTW 问题。

（1）运输问题；

（2）指派问题（assign passenger to vehicle）；

（3）背包问题（Knapsack problem）：当车辆能力不能满足所有乘客需求时，接哪些乘客才能使利润最大化？

（4）time-dependent shortest path；

（5）拉格朗日松弛算法（Lagrangian relaxation (LR) solution framework）：将原问题转化为 time-dependent shortest path problem；

（6） 动态规划（Dynamic Programming）：用来解决 time-dependent shortest path problem；

（7）Column generation (with given based price)。

[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.