|  |
| --- |
|  |
| Single Vehicle  **VRPPDTW问题** |
| GMS程序 使用说明 |
|  |
|  |
|  |

|  |
| --- |
| **2018.03** |

目录

[1、问题的底层网络结构 2](#_Toc507702428)

[2、模型与算法说明 4](#_Toc507702429)

[3、程序的执行 5](#_Toc507702430)

[4、程序的输入和输出 5](#_Toc507702431)

[4.1 输入信息 5](#_Toc507702432)

[4.2 输出信息 6](#_Toc507702433)

[5、测试数据集 8](#_Toc507702434)

[5.1 1辆车同时服务两位乘客 8](#_Toc507702435)

[5.2 1辆车同时服务两位乘客（无合乘） 8](#_Toc507702436)

[5.3 1辆车2位乘客，一人未被服务 9](#_Toc507702437)

[5.4 1辆车同时服务3名乘客 9](#_Toc507702438)

# 1、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。

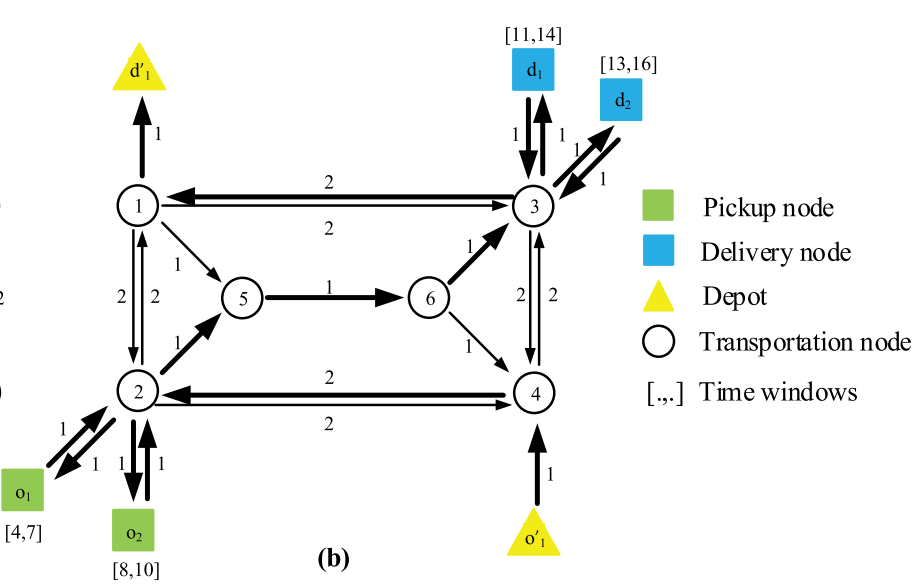


图1 VRPPDTW网络的节点

（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。

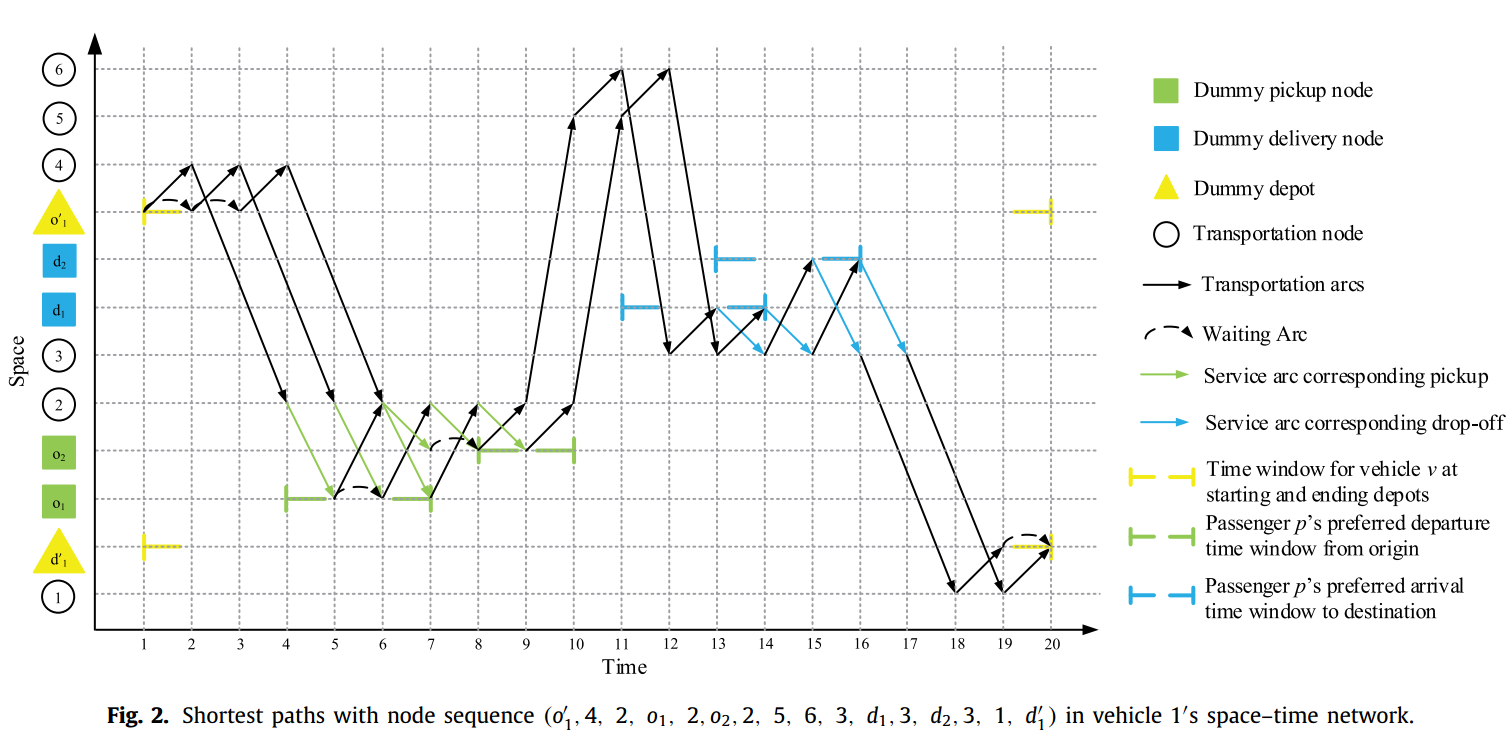


图2 VRPPDTW 网络的弧

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。

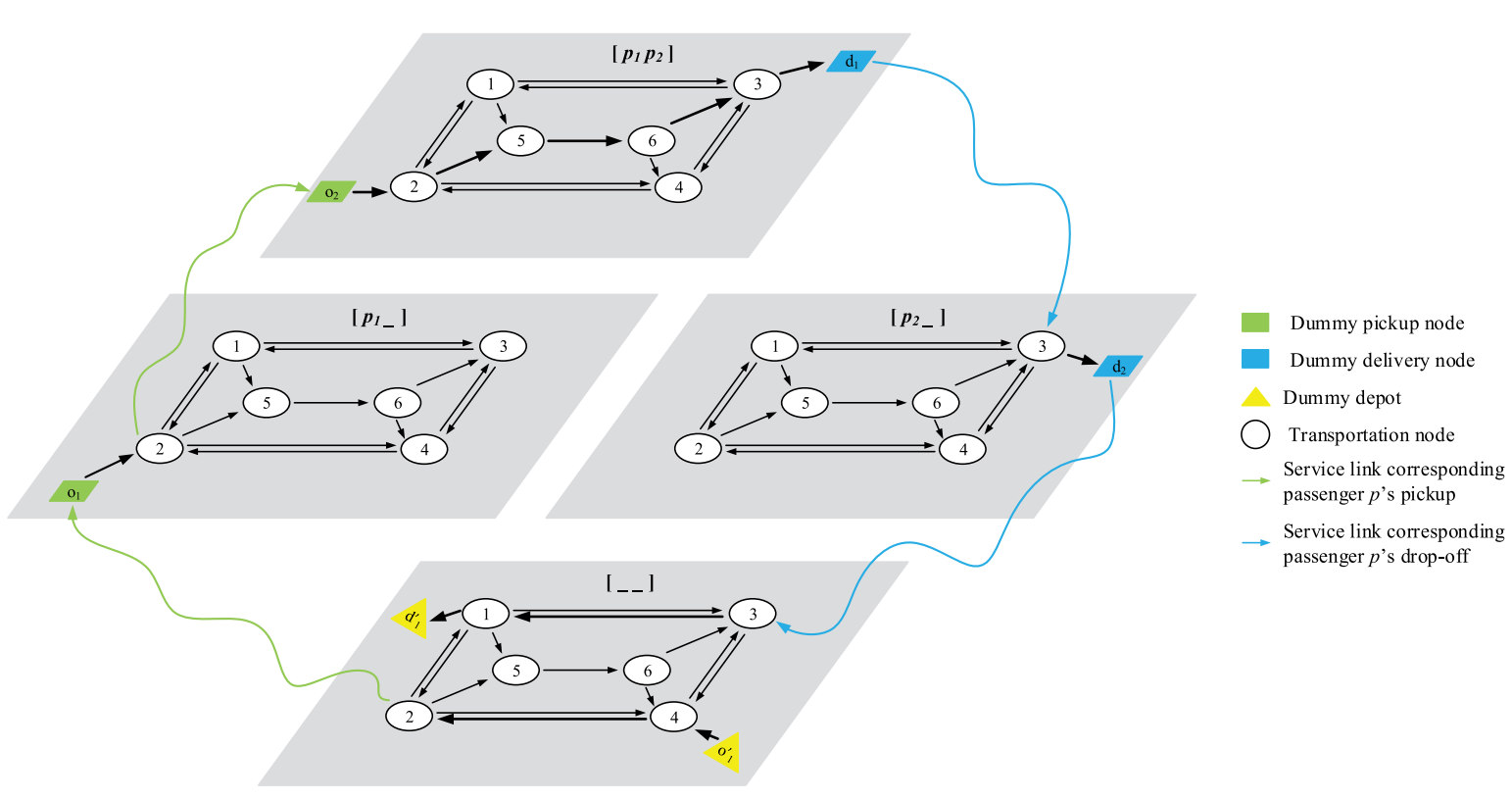


图3 状态转移

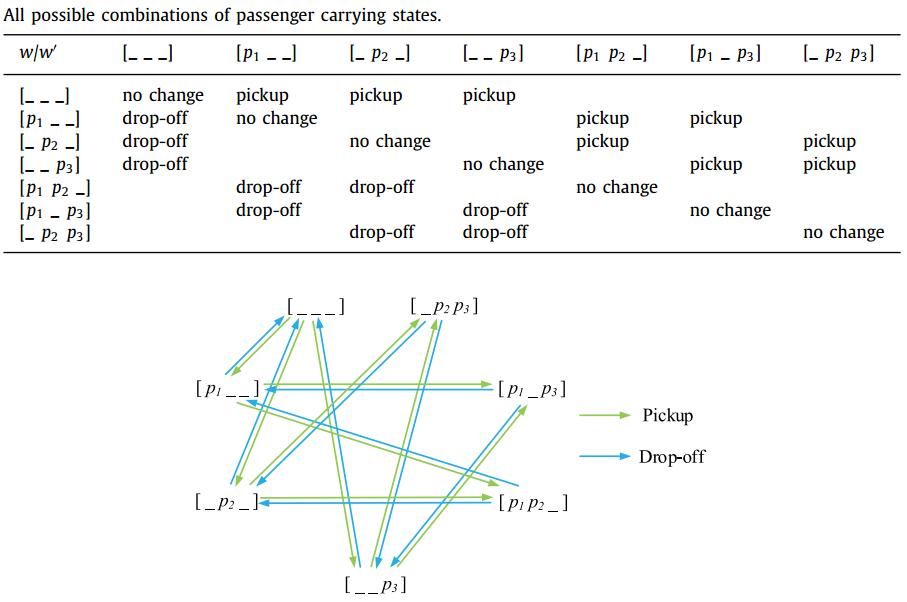
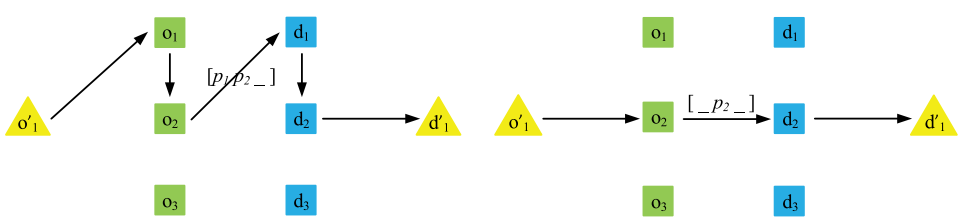


图4 状态转移

# 2、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, wp* | 状态参数 |
| *(i,t,w)* | VRPPDTW问题的起点 |
| *(i,j,t,s,w,wp)* | 节点 *(i,t,w)* 和 *(j,t,s)*之间的弧段 |
| *y(v,i,j,t,s,w,wp)* | 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,wp)表示车流v是否选择路段(i,j,t,s,w,wp)接送乘客或等待乘客。

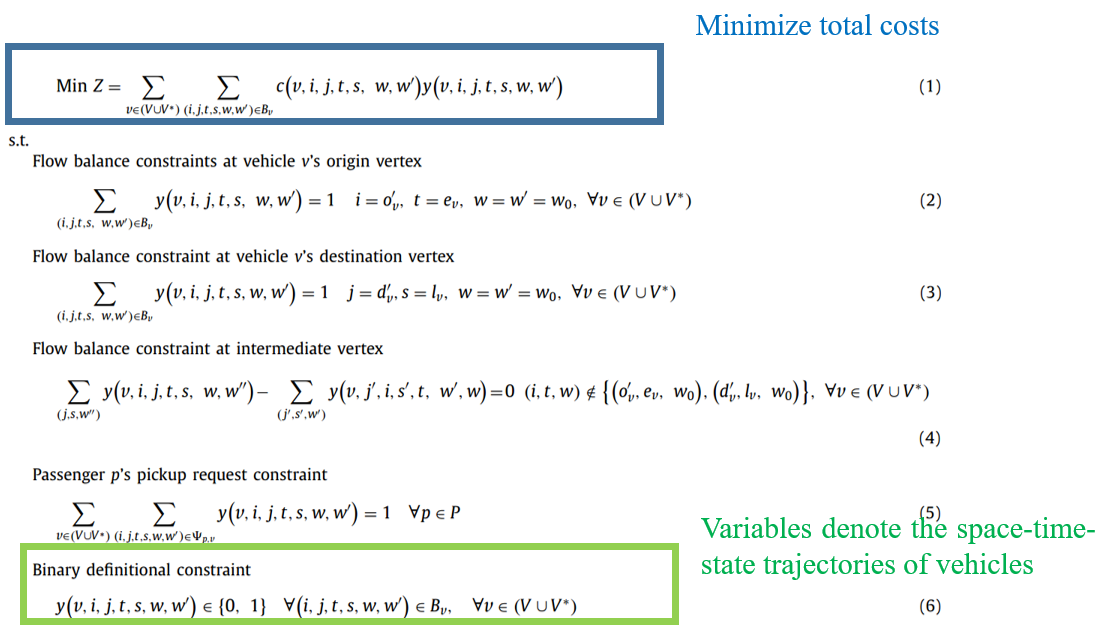


图5 VRPPDTW问题的模型

# 3、程序的执行

双击VRP\_eg.2.gms文件打开程序

# 4、程序的输入和输出

## 4.1 输入信息

表2 输入数据集

|  |  |  |
| --- | --- | --- |
| 编号 | 数据内容 | 参数名称 |
| 1 | 物理节点信息 | i/i\_source(i)/ i\_sink(i) |
| 2 | 路网连接 | arcs(i,j,t,s)，travel\_cost(k,i,j,t,s) |
| 3 | 时间窗信息 | win(t,n) |
| 4 | 乘客信息 | p/ p\_origin(i,t)/ p\_destination(i,t) |
| 5 | 车辆信息 | k |
| 6 | 状态信息 | w，w\_trans(w,wp) |

下面对输入数据量进行详述。所构建的虚拟网络如图6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）i/i\_source(i)/ i\_sink(i)

i为节点数， i\_source(i)为所出车场，i\_sink(i)为所入车场。

（2）win(t,n)

t表示时间窗所包含的各时间点，n表示时间窗序号，win(t,n)的值表示该时间窗对应的节点。时间窗t取值如图中方括号所示。



图6 虚拟路网

（3）arcs(i,j,t,s)

两点间的连接由此定义，其中包括起止节点编号，已经走行该弧段所需时间，其值取为1。

（4） p/ p\_origin(i,t)/ p\_destination(i,t)

p为乘客数量，p\_origin(i,t)和 p\_destination(i,t)分别表示乘客的出行起点和终点。

（5）k

其值为车辆数目。

## 4.2 输出信息

表3 输出文件

|  |  |  |
| --- | --- | --- |
| 编号 | 数据内容 | 名称 |
| 1 | 最小费用 | z |
| 2 | 车辆路径 | y(k,y,j,t,s,w,wp) |

在程序的输入中，我们通过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，构建出虚拟节点。

运行程序后，依次输出以下信息，如图7所示：



图7 输出信息

# 5、测试数据集

## 5.1 1辆车同时服务两位乘客

物理路网不变

按照以下步骤更改代码：

1. 修改时间窗乘客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]）。
2. 修改路网连接arcs(i,j,t,s)，修改虚拟节点与物理节点的连接。
3. 修改乘客出行起终点。

车辆运行路线如下图8所示。

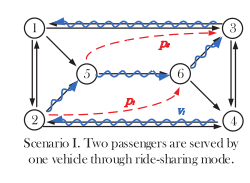


图8 算例1车辆走行路径

## 5.2 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]）。

该代码虚拟网络基于此例，可直接运行。

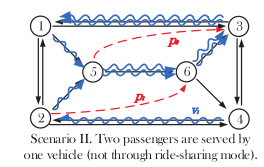


图9 算例2车辆走行路径

## 5.3 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]）。

1. 修改乘客数量为1\*3
2. 修改时间窗
3. 修改状态参数，将注解中的\*去除即可
4. 修改路网链接arcs(i,j,t,s)，修改虚拟节点与物理节点的连接。
5. 修改乘客出行起终点。

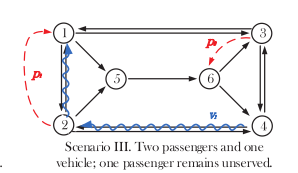


图10 算例3车辆走行路径

## 5.4 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]）。

1. 修改乘客数量为1\*3
2. 修改时间窗
3. 修改状态参数，将注解中的\*去除即可
4. 修改路网链接arcs(i,j,t,s)，修改虚拟节点与物理节点的连接。

⑤ 修改乘客出行起终点

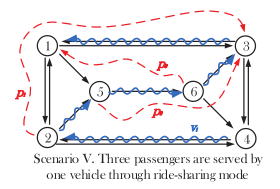


图21 算例4车辆走行路径

[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.