

JD_ 使用方法、相关参数意义

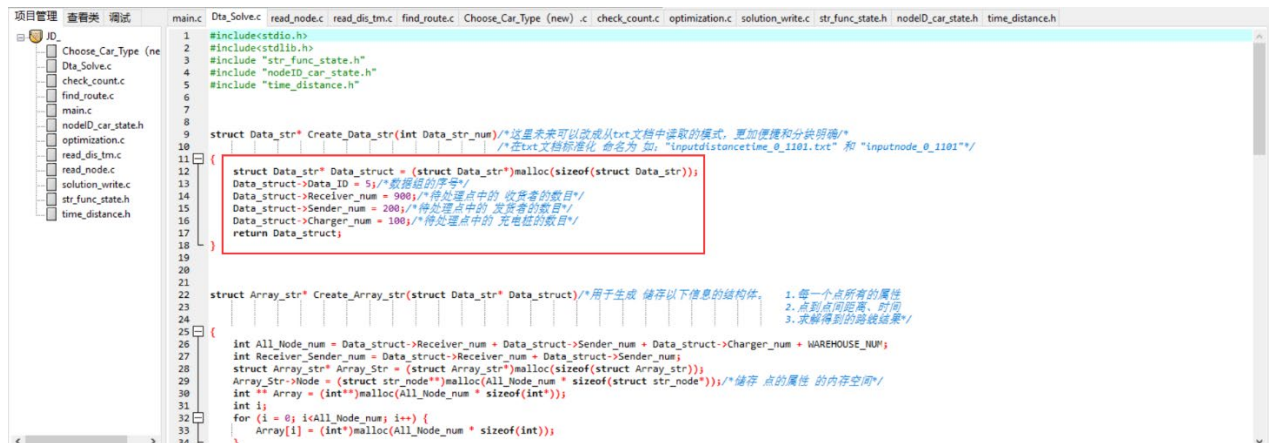
Yxp

2019.02

注意：本说明适用于‘JD 2.6 路线间 含优化 +注释版’，

使用方法：

1. 调用不同数据组进行测试：



```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include "str_func_state.h"
4 #include "nodeID_car_state.h"
5 #include "time_distance.h"
6
7
8
9 struct Data_str* Create_Data_str(int Data_str_num)/*这里未来可以改成从txt文件中读取的模式，更加便捷和方便快捷*/
10 /*在txt文档标准化 命名为 如: "inputdistancetime_0_1101.txt" 和 "inputnode_0_1101.txt"*/
11 {
12     struct Data_str* Data_struct = (struct Data_str*)malloc(sizeof(struct Data_str));
13     Data_struct->Data_ID = 5; /*数据组的序号*/
14     Data_struct->Receiver_num = 900; /*终点节点中的 收货者的数目*/
15     Data_struct->Sender_num = 200; /*终点节点中的 发货者的数目*/
16     Data_struct->Charger_num = 100; /*终点节点中的 充电桩的数目*/
17     return Data_struct;
18 }
19
20
21
22 struct Array_str* Create_Array_str(struct Data_str* Data_struct)/*用于生成 储存以下信息的结构体。 1. 每一个点所有的属性
23 2. 点到点距离、时间
24 3. 求解得到的路线结果*/
25 {
26     int All_Node_num = Data_struct->Receiver_num + Data_struct->Sender_num + Data_struct->Charger_num + WAREHOUSE_NUM;
27     struct Array_str* Array_str = (struct Array_str*)malloc(sizeof(struct Array_str));
28     Array_str->Node = (struct str_node**)malloc(All_Node_num * sizeof(struct str_node)); /*储存 点的属性 的内存空间*/
29     int ** Array = (int**)malloc(All_Node_num * sizeof(int)); /*储存 点的属性 的内存空间*/
30     int i;
31     for (i = 0; i < All_Node_num; i++) {
32         Array[i] = (int*)malloc(All_Node_num * sizeof(int));
33     }
34 }
```

图示为，调用第五组数据；

其中点的情况为：

收货者：900

发货者：200

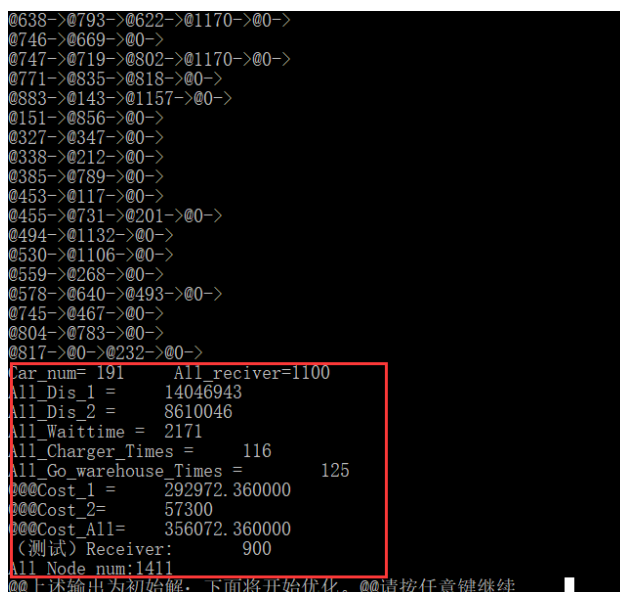
充电桩：100

即程序将自动读取‘inputdistancetime_5_1201.txt’和 ‘inputnode_5_1201.txt’中数据

注：在各数据组中，点的类型及其数目，参见附录一。

2. 输出结果各部分意义：

1) 初始解 相关信息：



```
@638->@793->@622->@1170->@0->
@746->@669->@0->
@747->@719->@802->@1170->@0->
@771->@835->@818->@0->
@883->@143->@1157->@0->
@151->@856->@0->
@327->@347->@0->
@338->@212->@0->
@385->@789->@0->
@453->@117->@0->
@455->@731->@201->@0->
@494->@1132->@0->
@530->@1106->@0->
@559->@268->@0->
@578->@640->@493->@0->
@745->@467->@0->
@804->@783->@0->
@817->@0->@232->@0->
car_num= 191 All_reciver=1100
All_Dis_1 = 14046943
All_Dis_2 = 8610046
All_Waittime = 2171
All_Charger_Times = 116
All_Go_warehouse_Times = 125
@@@Cost_1 = 292972.360000
@@@Cost_2 = 57300
@@@Cost_All= 356072.360000
(测试) Receiver: 900
All_Node_num:1411
@@ 上述输出为初始解；下面将开始优化。@@请按任意键继续...
```

2) 优化过程中，结果信息反馈：

```
已经优化！
113->@272->@27->@806->@31->@626->@984->@991->@1013->@981->@1188->@0->
298->@262->@243->@265->@308->@1106->@0->
Car_num= 2      All_reciver=15
All_Dis_1 =      0
All_Dis_2 =      288445
All_Waittime =    23
All_Charger_Times = 2
All_Go_warehouse_Times = 0
@@@Cost_1 =      4047.430000
@@@Cost_2=        600
(测试) Receiver: 15
All Node_num:19
已经优化！
105->@237->@74->@792->@1037->@1035->@963->@1060->@1112->@0->
295->@314->@703->@384->@0->
19->@0->
Car_num= 3      All_recive=13
All_Dis_1 =      141464
All_Dis_2 =      87308
All_Waittime =    81
All_Charger_Times = 1
All_Go_warehouse_Times = 2
@@@Cost_1 =      3000.280000
@@@Cost_2=        900
(测试) Receiver: 13
All Node_num:17
没有优化！
90->@570->@593->@1015->@1032->@1028->@932->@1118->@0->
78->@51->@1131->@0->
```

上述这些点组成的点集合，已经优化

上述这些点组成的点集合，没有优化

3) 最终优化结果 信息反馈：

```
All_Dis_1 =      49514
All_Dis_2 =      201373
All_Waittime =    133
All_Charger_Times = 0
All_Go_warehouse_Times = 1
@@@Cost_1 =      3490.590000
@@@Cost_2=        900
@@@Cost_All=      4390.590000
(测试) Receiver: 15
All Node_num:18
没有优化！
93->@153->@3->@692->@255->@8->@903->@937->@1099->@1085->@1106->@0->
780->@91->@96->@442->@492->@1106->@914->@0->
Car_num= 2      All_reciver=16
All_Dis_1 =      297625
All_Dis_2 =      0
All_Waittime =    26
All_Charger_Times = 2
All_Go_warehouse_Times = 2
@@@Cost_1 =      3629.900000
@@@Cost_2=        600
@@@Cost_All=      4329.900000
(测试) Receiver: 16
All Node_num:20
没有优化！
最终需要成本345658.68
具体已写入：总路线_5.txt 请查看！
请按任意键继续
```

相关参数意义：

1. 代码中注释的写法解释：

- 代码中使用的 ‘/*xxx*/’：表明内容为软件写完后，补注的内容；其主要相关于该部分在整体项目中的特定功能的介绍，以帮助理解整体项目的结构情况。
- 代码中使用的 ‘//xxx’：大多为代码编写时的问题的记录以及对于未来代码改进的思考；该部分可以跳过与不理睬。

2. 外部参数信息参考表

[1]. nodeID_car_state.h

```
/*以下为 有关点的属性、车辆初始状态 的外部参数 以及 限制条件*/

#define WAREHOUSE 0
#define WAREHOUSE_NUM 1
#define First_Receiver 1
#define MAX_CAN_USED_CAR_NUM 500 /*要求单次使用车辆不超过 500 台*/
#define MAX_RECEIVER_ARRAY 4

/*****//车型 及其初始状态*/
#define IVECO 1
#define I_max_volume 12
#define I_max_weight 2
#define I_driving_range 100000
#define TRUCK 2
#define T_max_volume 16
#define T_max_weight 2.5
#define T_driving_range 120000
#define First_ID 0
#define First_use 1
/*****/

/*****//点的类型*/
//#define MAX_CAR_NUM 1000
#define CHARGER_TYPE 4
#define SENDER_TYPE 3
#define RECEIVER_TYPE 2
#define WAREHOUSE_TYPE 1
/*****/

/*****//优化用到 的外部参数*/
#define MAX_RECOMBINANT_NODE_NUM 30 /*混合车辆数上限值*/
#define MAX_GROUP_NUM 2
/*****/
```

[2]. time_distance.h

```
#define START_TIME 480 /*时间限制 下限（最小值）*/
#define FINAL_TIME 1400 /*时间限制 上限（最大值）*/
#define UNLOAD_TIME 30 /*卸货用时*/
#define CHARGE_TIME 30 /*充电用时*/
#define RESET_TIME 60 /*返回仓库再出发 的用时*/
```

3. 函数参数信息参照表：
- 该部分比较复杂；目前请先参见'[str_func_state.h](#)'；以后会专门整理一个函数参数信息参考表。

附录：

附录一

各数据组点的类型及其数目

| 数据集 | 总的点的数目 | Receiver | Sender | Charger |
|-----|--------|----------|--------|---------|
| 5 | 1200 | 900 | 200 | 100 |
| 4 | 1300 | 1000 | 200 | 100 |
| 3 | 1400 | 1100 | 200 | 100 |
| 2 | 1500 | 1200 | 200 | 100 |
| 1 | 1600 | 1300 | 200 | 100 |

附录二

JD Distribution 总体结构图（概要）

