**车辆智能调度算法文档（基于c++）**

**1、实现目标**

通过给定的数据（包含订单信息、客户信息、地图信息、车辆信息），得到最优的装车方案以及车辆运行路线方案。

**2、数据预处理**

首先调用nlohmann库（用于读取JSON数据以及相关操作）。

在函数read\_data中实现数据预处理：

·读取数据中的限制条件

·处理订单和站点相关信息

·通过处理好的订单来得到所需的车辆类型以及数量

·通过地图信息得到距离以及路程消耗信息。

1. **算法实现**

数据预处理

进行初始化，得到初始种群

计算装载fitness

进行路径规划，多次迭代得到最优的路径

得到总的fitness

对种群进行多次迭代，得到最优fitness的种群

输出结果

**3.1对于订单装车使用随机初始化以及聚类初始化**

（1）若车辆数量大于站点数量，使用随机初始化。把处理好的订单信息随机装车，用一个二维矩阵来存储，行数代表车辆数，列数代表订单数，元素为1则代表此订单（此列）装载到此车（此行）上，否则为0。

（2）若车辆数量小于站点数量则使用聚类初始化。

·首先剔除初始出发地站点。

·根据经纬度对站点进行聚类（使用dkm库）

·根据聚类结果给车辆分配订单

**3.2计算车辆装载惩罚**

·若车辆上装载订单的重量、体积、件数超过车辆限制则，则计算其相应的惩罚。

·c\_load: 件数、体积、重量惩罚系数

**3.3进行车辆行进路径规划**

通过以下参数：

·START\_FEES: 起步价

·KM\_FEES: 每公里运费

计算路程花费，作为适应度（fitness）。

然后通过以下参数：

·N\_GENERATIONS: 迭代次数

·RATE\_M: 变异系数

·RATE\_C: 交叉系数

·POP\_SIZE: 种群数量

对初始种群进行迭代运算，找到最优（即fitness最小的）车辆路线规划。

**3.4进行订单装载车辆的规划**

通过前面装载惩罚和路径花费的计算可以得到此种订单分配车辆方案的总fitness，通过交叉和变异操作对方案进行改变，多次迭代计算得到最优的方案。

1. **结果输出**

{

"dispatchLimit": [ # 调度限制条件（列表）

1,

2

],

"dispatchMode": "2", # 调度模式 （str）

"dispatchRule": [ # 调度规则 （列表）

1

],

"requiredVehicles": { # 所需车辆信息

"10000035": { # 车辆类型vehicleType: {}

"plateNumber": [ # 车牌号（按车型调度时，车牌号为伪号码）

"沪A12345"

],

"quantity": 1 # 数量

},

"10000033": {

"plateNumber": [

"湘A66688",

"湘A88812",

"苏A123456"

],

"quantity": 3

},

"10000029": {

"plateNumber": [

"鄂A12342"

],

"quantity": 1

}

},

"routesInfo": [ # 车辆路线信息

{

"vehicleType": "10000029", # 车型

"vehiclePlate": "鄂A12342", # 车牌

"consigneeRoute": [ # 站点顺序（列表）

{

"consigneeId": "1000000037", # 从仓库开始

"deliveryOrder": 0,

"longitudeLatitude": [

"114.23382921",

"30.58007107"

],

"shipBillIds": []

},

{

"consigneeId": "1000000048",

"deliveryOrder": 1,

"longitudeLatitude": [

"121.478377",

"31.231787"

],

"shipBillIds": [ # 订单编号

"YD1000000234",

"YD1000000235",

]

}

]

},