考虑出行时间窗的定制公交线路车辆调度方法：http://kns.cnki.net/KCMS/detail/detail.aspx?dbcode=CJFQ&dbname=CJFDLAST2018&filename=ZGGL201805018&uid=WEEvREcwSlJHSldRa1FhdXNXaEd1OFFqN1JLQ0syd1lZUDlTSjhMdURDTT0=$9A4hF\_YAuvQ5obgVAqNKPCYcEjKensW4IQMovwHtwkF4VYPoHbKxJw!!&v=MDk5NDRSTE9mWk9kckZpRGtVYnZMUHlyTVlyRzRIOW5NcW85RWJJUjhlWDFMdXhZUzdEaDFUM3FUcldNMUZyQ1U=

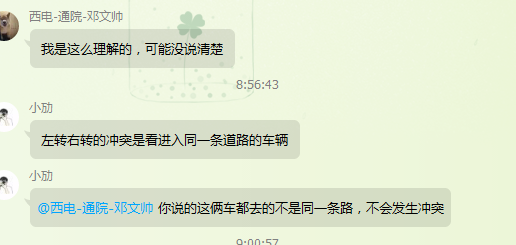
可考虑的思路：

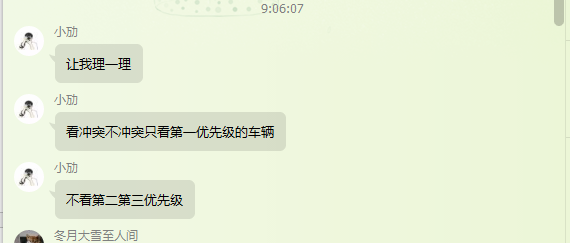
1.分为离线阶段和在线阶段：离线阶段:根据车间路线图提取节点坐标和节点间关系, 建立以节点距离为权值的无向图模型。用该模型和K-最短路径算法, 生成任意两个节点间的K条候补路径集, 并以路径表的形式存储起来, 构成路径库。

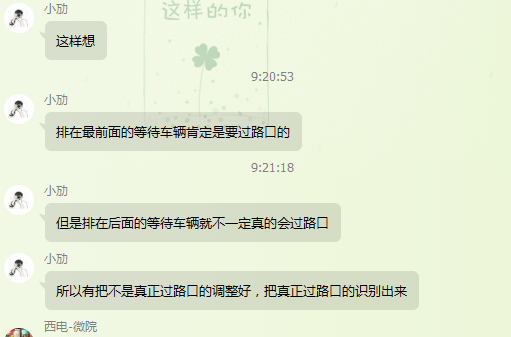
参考：基于多种群遗传算法的多AGV调度，http://kns.cnki.net/KXReader/Detail?TIMESTAMP=636881523229291250&DBCODE=CJFD&TABLEName=CJFDLAST2018&FileName=DZKK201811013&RESULT=1&SIGN=TkR6179KAnsWKhz03ehPqVLqGGM%3d&UID=WEEvREcwSlJHSldRa1FhdXNXaEd1OFFqN1JLQ0syd1lZUDlTSjhMdURDTT0=$9A4hF\_YAuvQ5obgVAqNKPCYcEjKensW4IQMovwHtwkF4VYPoHbKxJw!!&filetitle=%E5%9F%BA%E4%BA%8E%E5%A4%9A%E7%A7%8D%E7%BE%A4%E9%81%97%E4%BC%A0%E7%AE%97%E6%B3%95%E7%9A%84%E5%A4%9AAGV%E8%B0%83%E5%BA%A6\_%E5%AD%9F%E5%86%B2

规则解释：

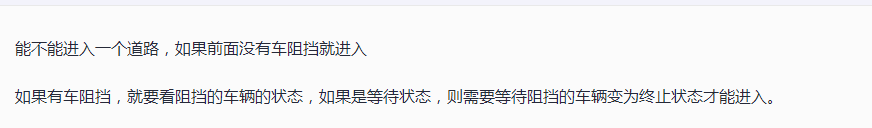
左转右转的冲突是看进入同一条道路的车辆







https://bbs.huaweicloud.com/forum/thread-15007-1-1.html



1. 给每辆车规划N条不同最短路径的思路：

weight = weight\_initial()

adj\_list = {}

paths = {}

for i in range(N):

path = shortest\_path(adj\_list, car\_df)

paths.append(path)

# 修改邻接表中边的权重，增大在path中使用过的边的权重

adj\_list = adl\_update(adj\_list, path)

return paths

函数adj\_list = adl\_update(adj\_list, path)中权重更新的设计思路：

For edge in all:

# 得到该条边在所规划的路径中出现的次数

num = use\_time(edge, paths)

Weight += lambda \* num^2 #lambda为权重因子，初始时暂取0.02

1. 权重初始化函数的设计：weight = weight\_initial()

# 最短路径是使所有边的权重之和最小

weight = lambda1 \* (s/v) + lambda2 \* (1/n) + lambda3 \* X….

其中lambda表示权重因子，(s/v)表示该条边长度除以最大通行速度，即表示该条边的通行时间; n表示该条边对应道路的车道数，即车道数越大，(1/n)值越小，表示通行耗时越短; X表示其他要考虑的因素。

对于权重的修改是为了规划出不同的路径，也可以考虑直接对地图进行剪枝操作，然后再进行路径规划，剪枝后的规划也可以得到不同路径，并且规划速度更短，但是可能会出现无法得到可行路径的问题。

3．根据出发时刻分布，以及车的数量对路径进行规划：首先得到车出发的计划时刻分布，对这些不同发车时刻的车按照一定比例分割为N份，分别对这N份车进行路径规划。在对每份车进行路径规划时，每得到一辆车的路径，就更新更新一次邻接表中的权重，思路类似1中邻接表的更新adj\_list = adl\_update(adj\_list, path)，这样做的目的是争取让地图中的每条路都被用到，从而使地图道路的利用率最高。

4. 道路当前时刻负荷率的设定：

miu = (car\_in\_road \* 1) / road\_all\_space

road\_all\_space：表示所有道路的每条车道的长度之和，即可以用来表征当前地图的负载能力。计算公式是：

for road in roads:

for channel in road:

road\_all\_space += channel.getLength()

通过控制负荷率小于一个值，来防止锁死情况的发生，即当负荷率高于某个值时，暂停发车。

5 . Hamilton问题：<https://github.com/michzio/Python---Public-Transport-Shortest-Path-Dijsktra-and-Hamilton-Cycle-Search/tree/master/python>

<https://github.com/zxlim123/Hamiltonian-Path/blob/master/HamiltonianPath.py>

50个城市路径问题：

<https://github.com/TheMalaka/Ensemble-Cheapest-Insertion-Heuristic-to-Solve-Hamilton-Path-Problem>

Java 基因遗传算法求取hamilton路径：

<https://github.com/mohamedalamicom/HamiltonianCircuit/blob/master/js/genetic-algorithm.js>

C++实现：

<https://github.com/samarth-p/Euler-and-Hamiltonian-Path/blob/master/DMS%20program.cpp>

https://github.com/Guber/Hamiltonian

复杂度最低的C++实现：Christofides algorithm in plane. 平面克里斯托费尔德斯算法

<https://github.com/VladX/Euclidean-Hamiltonian-Path>

重要的python实现：https://github.com/kriths/square-sum-problem

https://github.com/anmolg1024/AdvancedTSP/blob/master/AdvancedTSP.py

其他博客：

<https://blog.csdn.net/baimafujinji/article/details/49687061>

<https://blog.csdn.net/zhangyifei521/article/details/53283028>

https://blog.csdn.net/zhb67473517/article/details/80364476

首先考虑Hamilton路径是否存在的问题

https://max.book118.com/html/2017/0706/120636894.shtm

6.思路：先根据每条道路的通行能力，即之前的权重函数，对通行能力较差的节点剔除，然后求剩余节点间的Hamilton路径（大圆环），在考虑将剔除的节点接回至Hamilton路径中（再构成一个小圆环）