基于 Dijkstra 算法的轨道小车调度模型

摘 要

随着物联网等技术、智能化、自动化的发展，以轨道小车为核心的智慧医院立体物流传输系统已经在很多地方有了应用。而在这个系统中，如何设计算法合理调度轨道小车、设置变轨器、设置小车寄存区域，以实现各站点间的便利运输并尽可能提高效率，是一个亟待解决且十分重要的问题。基于此大背景，本文提出一种基于 Dijkstra 算法的轨道小车调度模型。

本文首先对整个系统做了建模，根据实际情况和题目要求充分讨论做出了必要假设和简化，并根据距离最小化的数学原理提出了变轨器和寄存区域的设计原则；同时通过设计搜索算法，将题目所给的单向边、转轨器、节点数据整合起来，形成一个邻接矩阵，一张计算机意义下的图，便于后续操作。

其次分别针对区域控制和集中控制建立了最优化模型，在求解此模型的过程中设计控制算法。前者以任务产生时间划分优先级，使用基于 Dijkstra 的贪心算法完成模型求解，即每个任务分配后路径不发生变化，在该任务不与先前任务冲突的情况下规划路径；后者使用分而治之的思想，并调用区域控制的控制算法，保证每个区域中使用时间最短、跨区域时小车等待时间最短，完成算法设计。

再根据随机过程相关内容，以泊松过程的理论为基础完成站点的任务产生算法，并使用泊松过程强度参数表征站点的等级请求负荷；根据问题需要对小车进行描述；选取了小车平均速度、实际时间与理想时间比值、停等次数作为评价指标。结合小车控制算法和任务产生算法，以循环增加最小单位时间、改变任务状态的方式完成离散化仿真，记录多组不同楼层数、不同站点请求负荷等级、不同小车数的仿真结果的影响并分析得出结论：设计的算法能较好完成任务。

最后对模型和算法的优缺点做了一定分析，并提出了未来改进方向。本文提出的基于 Dijkstra 的贪心算法在一定程度上为智慧医院立体物流传输系统设计方案提供了新的参考。

关键词：轨道小车调度，Dijkstra 算法，贪心算法

一、问题重述

为增加医院科室间传输物品的效率，现需要建设一个轨道物流系统，设计一个科学合理的轨道小车方案。已知该医院有一栋 12 层的大楼，每层楼有 8-20 个站点，每个站点可发出调车，发车，寄车指令，小车的行动通过轨道上的传感器受每 2-4 层楼设计一个的区域控制器控制，可以做出前进，后退，变轨三个动作。当小车收到寄车指令时，小车前往每 2-3 层设计的一个小车寄存区域待命。此外，为了满足跨区域的小车调度需求，还需要设计一个专门管理所有区域控制器的集中控制中心，完成区域控制器之间的协调。

我们需要完成的工作由以下五个部分组成：首先根据实际场景指定变轨器与小车寄存区域的设计原则；其次设计区域调度器的调度模型与算法，使轨道小车运行效率尽可能高；第三根据我们设计的算法对一组 4 层楼的轨道铺设方案进行评价，并给出系统在不同等级的请求负荷下的应用效果；第四设计区域控制器与集中控制器协调的调度模型与算法；最后根据设计的两个算法对一组 12 层楼的轨道铺设方案进行评价，并给出系统在不同等级的请求负荷下的应用效果。

二、问题分析

该问题本质上是一个路径规划的问题[5]。对于每个小车而言，它们的运行逻辑如图 1 所示。这样的运行逻辑可以确保距离站点最近的小车以最短距离到达站点并以最短路径完成送货任务，我们可以将之视为图论中的最短路径问题加以分析并解决。当环境中包含多个小车时，我们需要检测小车之间可能发生的路径冲突，并在解决冲突的前提下寻找最优的解决策略。该问题可以通过利用时间窗得到解决。[4,6]

在单个区域的控制问题得到解决后，再考虑解决多区域的综合控制问题。由于题目中指出小车只能在所属区域内运行，不同区域控制器只能控制该区域内的小车，因此跨区域的运输问题可以由在中继点更换小车的方式解决。

将不同区域的交汇点虚拟出一个新的中继站点，当有跨区域运输任务由集中控制器发出时，指令应分配到各个区域控制器。起始端的区域控制器只需解决由发出站点运输至中继站点的区域内运输问题，路线上的区域控制器只需进行中继站点间转移的操作，终端的区域控制器只需解决由中继站点至目标站点的区域内运输问题。该运输模式可将跨区域运输问题变为若干区域内运输问题，问题得到解决。

由于题目假设中提及小车初始位置均在寄存点，且在无任务发配时自动返回寄存点，因此寄存点与各站点之间的距离对运输效率有较大影响。应将寄存点设置于与各站点间距离和最小，且不易造成正常运输拥堵的地点才能使效率最大

化。通过建立到各站点距离之和关于寄存区域位置的函数求极值可计算出寄存区域的理想位置。[1,2]

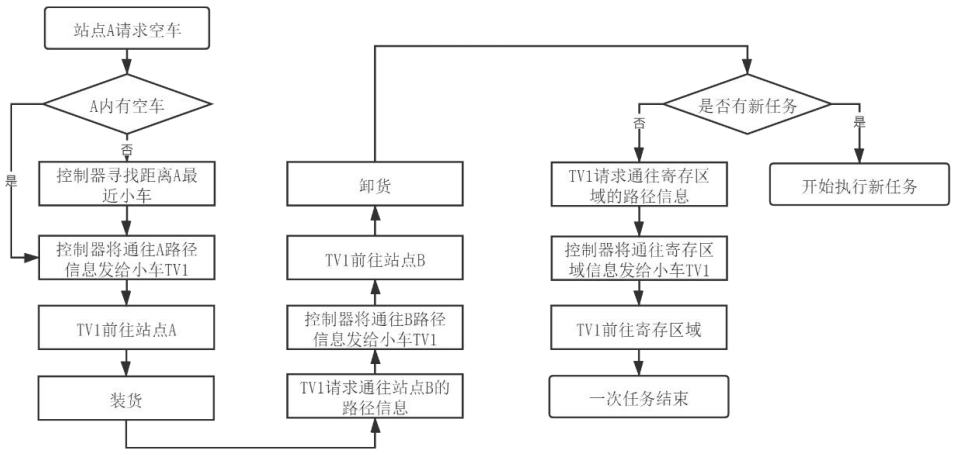

图 1 小车运行逻辑流程图

三、基本假设与变量说明

3.1 基本假设

1）传感器间距几乎为 0，即车辆状态被实时监测；

2）小车体积忽略不计均视为质点，各小车运行速度假设为 0.1m/s，忽略加速和减速过程。在变轨器、转轨器处变轨速度与正常运行速度一致；

3）各站点具有相同的地位，建模过程暂不区分不同站点不同的负荷等级；

4）在初始时刻，所有小车均位于寄存区域。当无指令需要执行时，小车将自动回到寄存区域。各寄存区域小车数量充足，即空闲小车数总大于需要执行的指令数；

5）每层楼具有相同的楼层结构，均匀分布 18 个站点，均采用双轨连接，整体结构示意图如图 2 所示；

6）不同需求产生的时刻总有一定时间差，同一时刻同时生成多个任务为小概率事件不作考虑。

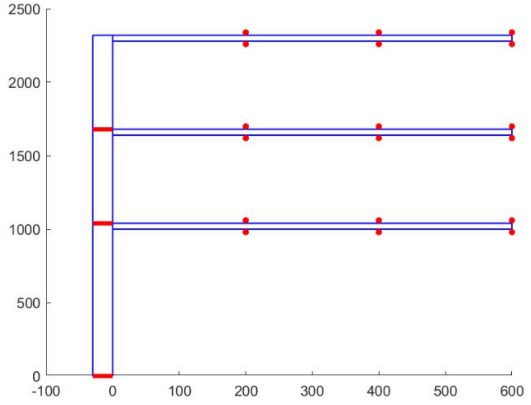

图 2 单层轨道与站点分布结构示意图

3.2 变量说明

表 1 变量说明

符号	变量解释
N	表示一个区域内总站点数
d_i	表示当区域内只有一个寄存区域时第 i 个站点沿轨道到寄存区域的距离
d_{1i}	表示当区域内有两个寄存区域时第 i 个站点沿轨道到寄存区域 1 的距离
d_{2i}	表示当区域内有两个寄存区域时第 i 个站点沿轨道到寄存区域 2 的距离
w_n	表示站点发出的第 n 个任务
W_n	表示收到新任务 w_n 时待完成的任务序列
TV_{w_i}	表示应当执行第 i 个任务的小车
p_{w_i}	执行第 i 个任务的小车装货的时刻
q_{w_i}	执行第 i 个任务的小车卸货的时刻
id	小车序号
t	接到任务的时间
s	起点
e	终点
KS	节点序列（含停等信息）
$flag$	小车状态
λ	请求负荷等级
N	车辆数
l	层数
$\bar{v}$	小车整个过程运行的平均速度
k	实际时间与理想时间比值
n	停等次数

四、模型的建立与求解

4.1 变轨器和寄存区域的设计

4.1.1 变轨器的设计原则

变轨器的作用是便利小车的运行，让小车的运行效率更高，同时减少可能出现的拥堵的状况。因此，一方面，变轨器的设置使得任意两站点之间的通行时间最短，即任意两站点间的距离之和最短；另一方面，变轨器也应设置在容易出现拥堵的路段，用于缓解小车的等待时间。

基于此，我们的变轨器设计原则为

- 变轨器的设置使任意两站点间的距离之和最小
- 变轨器应设置在易拥堵路段用于疏导小车的同行

记系统的站点序列为$\{Site_1, Site_2, ..., Site_n\}$，设置变轨器之前，站点 i 与站点 j 之间的距离记为$d_{ij}\ (1 \quad i \quad n.1 \quad j \quad n)$，设置变轨器之后，站点 i 与站点 j 之间的距离变为$d_{ij}'(1 \quad i \quad n.1 \quad j \quad n)$，则$d_{ij}' - d_{ij}$可以用于刻画设置变轨器对两站点同行时间的变化越大，变轨器的设置对这两个站点的作用就越大。因此，变轨器的设计可归结于优化模型

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} - d_{ij}' \tag{1}$$

对于题目给出的楼层结构，代入模型求解后可得，变轨器应该设置在两个站点之间的轨道上。这样的设计具有现实意义，因为它有效地缓解了不同小车在站点处的拥堵问题。

4.1.2 寄存区域的设计原则

一般情况下每 2-4 个楼层设置一台区域控制器划分为一个区域，每 2-3 个楼层设置一个寄存区域安置空闲小车。因此每个运行区域内可能有 1 或 2 个寄存区域。

寄存区域作为小车的存放处，一方面不应对区域内小车的通行造成困扰，即不应设置在容易发生拥堵的路段；另一方面，为了更快地响应个站点的需求，寄存区域到站点不宜过远，即应设置在距离各站点距离和最近的位置。因此，寄存区域设计原则为

- 寄存区域位置不应设置在容易出现堵塞的路段。
- 寄存区域位置应使其到各站点沿轨道距离之和取得最小值。

a. 当一个区域设置一个寄存区域时

记一个区域内总站点数为 N，第 i 个站点沿轨道到寄存区域的距离为 d_i，则寄存区域的最佳位置应满足

$$\min \sum_{i=1}^{n} d_i \tag{2}$$

对于题目所给的四层楼层结构，计算可得寄存区域位置坐标为(0,0,900)，如图 3 绿色星形标志位置所示。该位置经验证不为易堵塞路段，满足两条设计原则。

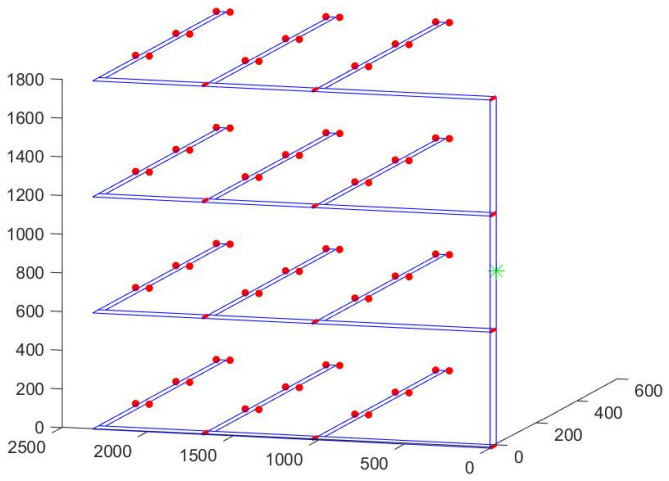

图 3 单个寄存区域设计位置示意图

b. 当一个区域设置两个寄存区域时

同样，记一个区域内总站点数为 N，第 i 个站点沿轨道到寄存区域 I 和寄存区域 II 的距离为 d_{1i}、d_{2i}，则寄存区域的最佳位置应满足

$$\min \sum_{i=1}^{n} \min \left\{ d_{1i}, d_{2i} \right\} \tag{3}$$

对于题目所给的四层楼层结构，计算可得寄存区域 I 和寄存区域 II 坐标可选为(0,0,300)、(0,0,1200)，如图 X 绿色星形标志位置所示。该位置经验证不为易拥堵路段，满足两条设计原则。

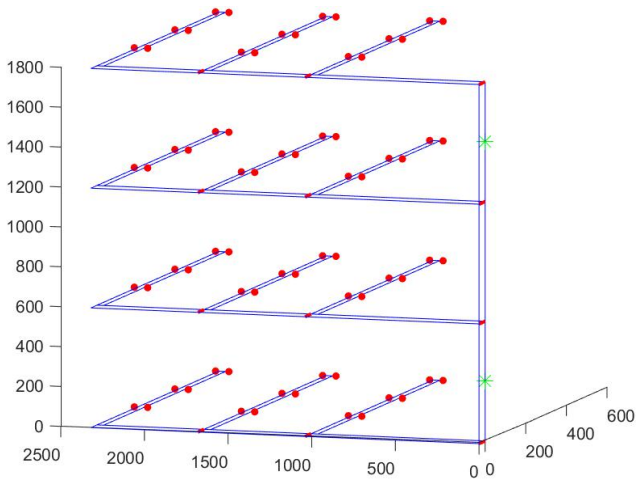

图 4 双寄存区域设计位置示意图

在后续建模过程中，为降低模型复杂度每个运行区域均只设置一个寄存区域，每两层认为是一个运行区域。

4.2 区域控制器的调度模型和算法

4.2.1 模型的准备

a. 轨道建模

轨道建模部分主要讨论不同轨道类型在模型中的处理方式和在运行过程中的规范与限制。根据题目的要求，轨道共分为单轨和双轨两种。对于单轨，小车在单轨运行时应能够实现灵活的正向和反向行驶，但不能允许相向行驶的两辆小车同时通过，因此可抽象为一条无向边；对于双轨，为保证小车的有序运行，我们规定双轨均可采取双向行驶，因此可抽象为两条无向边。示意图如下图所示。

图 5 轨道模型示意图

为实现系统的正常运行，结合实际对小车的在轨行驶作如下规定：

- 小车在轨道上恒定匀速行驶，不考虑加速减速过程；
- 小车在单轨上无法相向通过，需要在节点处进行变轨避让。

b. 站点建模

站点建模部分同样对站点做了分类并设置了各自的规范与限制。根据站点位置和功能，站点共可以被分为四类：一般站点、寄存区域和中继站点。一般站点是指可以作为起点或终点的节点，根据位置不同分为通过站点和边缘站点。通过

站点可引出两条边与两个不同站点相连，而边缘站点只有唯一的对外通路，小车到达该站点后只能进行返回操作。寄存区域是特殊的通过站点，可以容纳小车在其中停靠。中继站点是特殊的、虚拟的边缘站点，在两个相邻区域的交汇点处，我们假定存在一个中继站点，跨区域运输的货物可以在中继站点完成交互。图 8 为示意图。

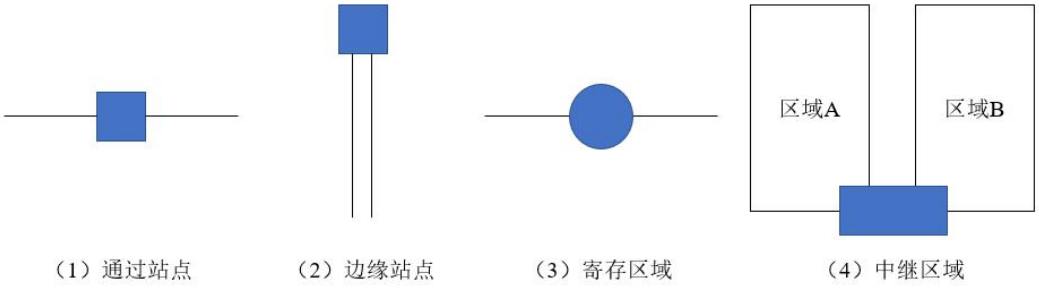

图 6 站点模型示意图

为实现系统的正常运行，结合实际对站点作如下规定：

- 小车无法通过中继站点进入其他区域，可以在中继站点进行装、卸货，由其他区域小车继续运输；
- 小车在站点处的装、卸货时间忽略不计；
- 一次任务的完整流程是，空闲小车到达需求发出站点，再到需求的目标站点，最后返回寄存区域。

4.2.2 区域控制器的调度模型

区域调度器应该整体规划该区域内各小车的运行情况使得系统取得最优解，即使得所有任务完成的总体效率最高，也就是所花的总时间应该最少。由于我们假设了小车均以相同的速度匀速运行，因此可以使用各小车完成任务的总路程替代时间，作为系统调度的优化指标。

为使得区域运行有序且每个需求都能最快得到响应，我们设计区域调度器在每次接受站点新任务时，重新对尚未装货的任务进行整体分配小车、规划路线，而已经装货的小车不再改变路线，不考虑中途换车的问题。

定义任务序列$(w_1, w_2, ..., w_i)$，每个任务对应小车$(TVw_1, TVw_2, ..., TVw_i)$.

设小车TVw_i位置关于时间的函数为

$$X_{TVwi}(t) \tag{4}$$

记该小车装货时刻为p_{w_i}，卸货时刻为q_{w_i}，则小车完成任务的路程为

$$S_{TVw_i}(t)=\int_0^{q_{w_i}}\left|X_{TVw_i}'(t)\right|dt \tag{5}$$

根据设计，区域控制器在每次接受新任务时，进行计算、重新规划路线和分发指令。记在t_n时刻接收到新任务w_n，此时刻待完成任务序列为

$$W_n=\left(w_{n_1},w_{n_2},w_n\right),0<n_1<n_2<...<n \tag{6}$$

则区域寄存器在t_n时刻的规划目标为下面最优化模型

$$\min\sum_{j\in D_n}S_{TVj} \tag{7}$$

s.t.

①$D_n\subseteq W_n$

②$t_i+p_{w_i}>t_n,\forall w_i\in D_n$

③$X_{TVw_a}'(t_0)=X_{TVw_b}'(t_0),\ if\ X_{TVw_a}(t_0)=X_{TVw_b}(t_0)$

$\forall w_a,w_b\in W_n(a<b),\max(t_a,t_b)<t_0<\min(t_a+q_{w_a},t_b+q_{w_b})$

依据模型，区域寄存器每次计算均使得当前待完成任务取得最优方案，因此系统在整个运行过程中的策略为近似最优策略。

4.2.3 区域控制器的调度算法

求解该模型，可使用基于 Dijkstra 的贪心算法。即每当接受站点新任务时，利用 Dijkstra 算法求解该小车在不与其他小车发生碰撞条件下的最优路径，而不更改其余小车的既定路线。整个求解算法流程如下。

Scheduling(MissionList, MissionSet=[], i)

1. 初始化任务集，为空；初始化时间，记当前为零时刻；初始化区域 i 的相关信息

2. 跳转至下一时刻，更新任务集中所有任务状态

(1) 若状态为由 0 或 3 变为 1

M1=ShortestPath(PreStart, Start, MissionSet)

此新任务 M1 加入任务集

(2) 若状态由 1 变为 2

删除上一任务 M1

M2=ShortestPath(Start, End, MissionSet)

此新任务 M2 加入任务集

(3) 若状态由 2 变为 3

删除上一任务 M2

M3=ShortestPath(End, StoragePlace, MissionSet)

此新任务 M3 加入任务集

(4) 若状态由 3 变为 0

删除上一任务 M3

3. 查看 MissionList，判断当前是否有新任务，若无则转到 2；若有则转到 4；若已执行完则退出

4. 判断离起点 Start 最近的空闲小车（返程中小车或寄存区小车），位置记为 PreStart，跳转 2

ShortestPath(Start, End, MissionSet)

1. 使用 Dijkstra 算法寻找 Start 到 End 的最短路径形成任务 M，记录完成时间

2. 检测新任务 M 与任务集 MissionSet 中的任务是否有冲突，若无则跳转 4；否则转到 3

3. 在冲突发生前的节点执行新任务的小车等待，即增加邻接矩阵对应边的权重。超出递归深度则停止，选取最短时间及对应最短路径的任务 M，转到 4；否则转到 2

4. 将新任务其加入任务集

其中 Dijkstra 算法流程如下[3]：

Dijkstra.(G,w,s)

```
1 INITIALIZATION-SINGLE-SOURCE(G,s)
2 S=∅
3 Q=G.V
4 while Q≠∅            //Q 最小优先队列保存节点集合
5    u=EXTRACT-MIN(Q)
6    S=S∪{u}
7    for each vertex v∈G.Adj[u]
8         RELAX(u,v,w)
```

4.3 区域控制器的仿真与评价

4.3.1 仿真环境的准备

a. 新任务的随机产生模型

系统参数及小车控制策略已在以上部分充分讨论，此部分解决各站点新任务的随机生成问题。

在现实生活中，当一个随机事件以固定的平均瞬时速率λ（或称密度）随机且独立地出现时，那么这个事件在单位时间（面积或体积）内出现的次数或个数就近似地服从泊松分布$P(\lambda)$[9]。

设随机过程$\{X(t),t\geq 0\}$是一个计数过程[7]，满足：

(1) $X(0)=0$

(2) $X(t)$是独立增量过程

(3)对任一长度为t的区间中事件的个数服从参数为λt $(\lambda>0)$的泊松分布，即对一切$s,t\geq 0$，有

$$P\{X(t+s)-X(s)=k\}=\frac{(\lambda t)^k}{k!}e^{-\lambda t}(k=0,1,2,\ldots) \tag{8}$$

则称$X(t)$为具有参数λ的泊松过程。

而需求的产生过程也可以认为是一个泊松过程。

可以证明[8]，给定时间T，记$[0,T]$内时间发生的次数是$N=P(\lambda t)$，在已知T内时间发生了N次的条件下，若生成N个独立的$(0,1)$均匀分布随机数$U_1,U_2,U_3,\ldots,U_N$，从小到大排序为$U_{(1)},U_{(2)},U_{(3)},\ldots,U_{(N)}$，则

$$(TU_{(1)},TU_{(2)},TU_{(3)},\ldots,TU_{(N)}) \tag{9}$$

即为时间 T 之前的所有事件到来时间。

基于此，可设计各站点新任务的随机生成算法：

初始化泊松过程强度λ和预计模拟时间 t

根据泊松分布概率公式生成（伪）随机数 N

生成 N 个在(0,T)均匀分布的随机数，并排序$t_1,t_2,t_3,\ldots,t_N$

生成 N 个随机数对$(s_1,e_1),(s_2,e_2),(s_3,e_3),\ldots,(s_N,e_N)$其中$s_i\neq e_i(i=1,2,3,\ldots N)$

得到任务集为$[(t_1,s_1,e_1),(t_2,s_2,e_2),\ldots,(t_N,s_N,e_N)]$

b. 小车的描述

当一辆小车接到任务时，最优路径即刻生成，对这样刚接到任务的小车描述如表 4。

表 4 小车描述

符号	描述
id	小车序号
t	接到任务的时间
s	起点
e	终点
KS	节点序列（含停等信息）
flag	小车状态

其中小车状态有 4 种：在寄存区、从寄存区（返程时）前往任务起点、从任务起点到任务终点、从任务终点返回寄存区。这四种状态分别用 0，1，2，3 来表示。

c. 变量与评价指标的选取

首先，需要选取 3 个变量：请求负荷等级、车辆数、层数，如表 5。考察这三个变量将会对结果产生的影响。请求负荷等级也即单位时间产生任务多少的程度，其越大也意味着产生拥堵的可能性越大；车辆数不同，任务的完成情况也不相同，事实上有两种极端情况，当只有一辆小车时问题变为一个“邮差问题”，当小车数量足够时问题变为冲突避免的多目标优化问题，其余情况为一般情况；不同层数意味着不同的结构、寄存区数目、区域数，根据题目提供数据，层数仅为 4 和 12，在 4.3 区域控制器的仿真与评价中层数为 4。

表 5 影响变量

符号	描述	取值
λ	请求负荷等级	$(0,+\infty)$
N	车辆数	N_+
l	层数	4,12

其次，为了评价仿真结果，我们需要选取合适的指标：小车平均速度、实际时间与理想时间比值、停等次数，如表 6。在理想情况下，小车速度为 0.1m/s。而请求负荷等级较高时因为停等，小车的平均速度将下降；同样，因为停等、重新路径规划，实际时间与理想时间比值也将大于 1，且越大意味着拥堵越严重；请求负荷等级较高时停等次数将增加，其也是拥堵程度的一个指标。

表 6 评价指标

符号	描述
$\bar{v}$	小车整个过程运行的平均速度
k	实际时间与理想时间比值
n	停等次数

4.3.2 仿真结果与评价

结合系统参数、小车控制策略、任务生成算法等内容，可以编写程序进行仿真。代码详见附录。下面展示仿真结果的统计信息及其评价。

a. 平均速度$\bar{v}$与λ、N的关系

表 7 平均速度与λ、N的关系

N/λ	0.01	0.1	0.2	0.5	1
1	0.1	0.1	0.1	0.1	0.1
2	0.1	0.098	0.097	0.096	0.096
10	0.1	0.096	0.088	0.085	0.082
50	0.1	0.096	0.086	0.083	0.081
200	0.1	0.096	0.086	0.082	0.081

可见，当小车数量增加时，若请求负荷等级小，平均速度几乎不变，否则有下降的趋势，但小车数量足够多时平均速度趋于稳定；当请求负荷等级增大时，若车辆少，平均速度几乎不变，相反，若车辆多，平均速度有不断下降的趋势。

b. 实际时间与理想时间比值k与λ、N的关系

表 8 实际时间与理想时间比值与λ、N的关系

N/λ	0.01	0.1	0.2	0.5	1
1	1	1	1	1	1
2	1	1.03	1.04	1.05	1.07
10	1	1.06	1.14	1.19	1.23
50	1	1.06	1.16	1.21	1.24
200	1	1.06	1.17	1.22	1.24

可见，当小车数量增加时，若请求负荷等级小，实际时间与理想时间比值几乎不变，否则有增大的趋势，但小车数量足够多时比值趋于稳定；当请求负荷等级增大时，若车辆少，比值几乎不变，相反，若车辆多，比值有不断上升的趋势。

c. 停等次数n与λ、N的关系

表 9 停等次数与λ、N的关系

N/λ	0.01	0.1	0.2	0.5	1
1	0	0	0	0	0
2	0	0.12	1.31	1.36	1.40
10	0	0.13	2.42	5.67	5.88
50	0	0.13	2.45	6.22	13.65
200	0	0.14	2.45	6.25	14.94

可见，当小车数量增加时，若请求负荷等级小，停等次数几乎不变，否则有增大的趋势，但小车数量足够多时停等次数趋于稳定；当请求负荷等级增大时，若车辆少，停等次数几乎不变，相反，若车辆多，停等次数有不断上升的趋势。

4.4 集中控制器的调度模型和算法

4.4.1 集中控制器的调度模型

由于各区域内的小车运行相互独立，由区域控制器进行分别调度，集中控制器需要考虑的是跨区域运输的小车调配问题。

在设计中我们通过中继站点将不同区域分而治之，跨区域运输问题简化为“起始站点——若干中继站点——目标站点”的分步运输问题。

(1)起始站点——中继站点

考虑任意两个相邻的区域 A 和区域 B，计算由区域 A 中的站点 $Site_A$ 到区域 A 与区域 B 的中继站点 $Site_{(A,B)}$ 的距离。由于小车的调配需要时间，因此该距离为区域 A 调配一小车到 $Site_A$ 加上从 $Site_A$ 到 $Site_{(A,B)}$ 的距离与区域 B 调配一小车到 $Site_{(A,B)}$ 距离的最大值，记为

$$S_1(A,B) \tag{10}$$

(2)中继站点——中继站点

考虑任意相邻的区域 A 和区域 B，相邻的区域 B 和区域 C，计算由区域 A 与区域 B 的中继站点 $Site_{(A,B)}$ 到区域 B 与区域 C 的中继站点 $Site_{(B,C)}$ 的距离。同(1)理，该距离为从 $Site_{(A,B)}$ 到 $Site_{(B,C)}$ 的距离与区域 C 调配一小车到 $Site_{(B,C)}$ 距离的最大值，记为

$$S_2(B,C) \tag{11}$$

(3)中继站点——目标站点

考虑任意两个相邻的区域 B 和区域 C，计算由区域 B 与区域 C 的中继站点 $Site_{(B,C)}$ 到区域 C 中的站点 $Site_C$ 的距离。由于小车的调配需要时间，因此该距离为区域 C 调配一小车到 $Site_{(B,C)}$ 加上从 $Site_{(B,C)}$ 到 $Site_C$ 的距离，记为

$$S_3(C) \tag{12}$$

这样，从区域 A 中的站点 $Site_A$ 到区域 C 中的站点 $Site_C$ 之间的距离表示为

$$S(Site_A, Site_C) = S_1(A,B) + S_2(B,C) + S_3(C) \tag{13}$$

进而，对于任意跨区域任务 w_i，实现从区域 IN 的站点 $Site_{IN}$ 到区域 OUT 的

站点 $Site_{OUT}$，途径区域为$\left(Site_1, Site_2, ...Site_r\right)$，则任务 w_i 的总路程为

$$S_{w_i} = S_1\left(IN, A_1\right) + \sum_{k=1}^{r-1} S_2\left(A_i, A_{i+1}\right) + S_3\left(OUT\right) \tag{14}$$

记集中控制器在 t_n 时刻接收到新的跨区域任务 w_n'，此时刻待完成任务序列为

$$W_n' = \left(w_{n_1}', w_{n_2}', w_n'\right), 0 < n_1 < n_2 < ... < n \tag{15}$$

则集中寄存器在 t_n 时刻的规划目标为下面最优化模型

$$\min \sum_{w_j \in W_n'} S_{w_j} \tag{16}$$

根据模型，集中控制器的主要优化目标有两个方面。一方面是跨区域的路径，即途径区域序列$\left(Site_1, Site_2, ...Site_r\right)$，另一方面则是各区域内部的调度方案。

4.4.2 集中控制器的调度算法

求解该模型，可结合分治思想运用基于 Dijkstra 的贪心算法。即分区域指派任务，且每当接受站点新任务时，利用 Dijkstra 算法求解该小车在不与其他小车发生碰撞条件下的最优路径，而不更改其余小车的既定路线。参考 4.2.3，整个求解算法流程如下。

CenteredScheduling(MissionList, MissionSet=[])

1. 初始化时间，记当前为零时刻
2. 跳转至下一时刻，检查当前是否有任务；若所有任务都以执行完毕则退出
   (1) 若有任务 M，则判断是否为跨区域调度
      ① 若是，则跳转 3
      ② 若否，其仅为区域 i 的调度，则
         M_0=Scheduling(MissionList, MissionSet, i)
         M_0 加入任务集
   (2) 若无，则回到 2
3. 将任务拆分，M_1=从起始区域(i_1)到中继节点 1，M_2=中继节点 1 到中继节点 2(in i_2)，…，M_n=中继节点 n-1 到中继节点 n(in i_n)，M_{n+1}=中继节点 n 到目标区域(in i_{n+1})
   (1) M_1=Scheduling(M_1, MissionSet, i_1)，加入任务集

(2) M_2=Scheduling(M_2, MissionSet, i_2)，加入任务集

……

(n) M_{n+1}=Scheduling(M_{n+1}, MissionSet, i_{n+1})，加入任务集

4. 跳转 2

Scheduling(MissionList, MissionSet=[], i)

1. 初始化任务集，为空；初始化时间，记当前为零时刻；初始化区域 i 的相关信息

2. 跳转至下一时刻，更新任务集中所有任务状态

(1) 若状态为由 0 或 3 变为 1

M1=ShortestPath(PreStart, Start, MissionSet)

此新任务 M1 加入任务集

(2) 若状态由 1 变为 2

删除上一任务 M1

M2=ShortestPath(Start, End, MissionSet)

此新任务 M2 加入任务集

(3) 若状态由 2 变为 3

删除上一任务 M2

M3=ShortestPath(End, StoragePlace, MissionSet)

此新任务 M3 加入任务集

(4) 若状态由 3 变为 0

删除上一任务 M3

3. 查看 MissionList，判断当前是否有新任务，若无则转到 2；若有则转到 4；若已执行完则退出

4. 判断离起点 Start 最近的空闲小车（返程中小车或寄存区小车），位置记为 PreStart，跳转 2

ShortestPath(Start, End, MissionSet)

1. 使用 Dijkstra 算法寻找 Start 到 End 的最短路径形成任务 M，记录完成时间

2. 检测新任务 M 与任务集 MissionSet 中的任务是否有冲突，若无则跳转 4；否则转到 3

3. 在冲突发生前的节点执行新任务的小车等待，即增加邻接矩阵对应边的权重。超出递归深度则停止，选取最短时间及对应最短路径的任务 M，转到 4；否

则转到 2

4. 将新任务其加入任务集

其中 Dijkstra 算法流程如下[3]：

Dijkstra.(G,w,s)
1 INITIALIZATION-SINGLE-SOURCE(G,s)
2 S=∅
3 Q=G.V
4 while Q≠∅ //Q 最小优先队列保存节点集合
5 u=EXTRACT-MIN(Q)
6 S=S∪{u}
7 for each vertex v∈G.Adj[u]
8 RELAX(u,v,w)

4.5 集中控制器的仿真与评价

新任务随机产生模型、小车的描述、变量与指标的选取详见 4.3.1 仿真环境的准备。新任务随机产生基于泊松过程；小车的描述含小车序号、接到任务的时间、起点、终点、节点序列（含停等信息）、小车状态；需要选取 3 个变量，请求负荷等级、车辆数、层数，在 4.5 集中控制器的仿真与评价中，车辆数取足够，层数为 12；评价指标为小车平均速度、实际时间与理想时间比值、停等次数。下面展示仿真结果的统计信息及其评价。

a. 平均速度$\bar{v}$与λ的关系

表 10 平均速度与λ的关系

λ	0.01	0.1	0.2	0.5	1
$\bar{v}$	0.1	0.099	0.091	0.088	0.082

可见，当请求负荷等级增大时，平均速度有不断下降的趋势。

b. 实际时间与理想时间比值k与λ的关系

表 11 实际时间与理想时间比值与λ的关系

λ	0.01	0.1	0.2	0.5	1
k	1	1.07	1.17	1.24	1.35

可见，当请求负荷等级增大时，实际时间与理想时间比值有不断上升的趋势。

c. 停等次数n与λ的关系

表 12 停等次数与λ的关系

λ	0.01	0.1	0.2	0.5	1
n	0	0.22	2.89	7.03	16.34

可见，当请求负荷等级增大时，停等次数有不断上升的趋势。

五、结论

1. 成功对整个系统做了建模，根据实际情况和题目要求充分讨论作出了必要假设和简化，并根据数学原理提出了变轨器和寄存区域的设计原则；同时通过设计搜索算法，将题目所给的单向边、转轨器、节点数据整合起来，形成了一个邻接矩阵，一张计算机意义下的图。

2. 分别针对区域控制和集中控制建立了最优化模型，在求解此模型的过程中设计控制算法。前者以任务产生时间划分优先级，使用基于 Dijkstra 的贪心算法完成模型求解，后者使用分而治之的思想，并调用区域控制的控制算法，保证每个区域中使用时间最短、跨区域时小车等待时间最短，成功完成了算法设计。

3. 根据随机过程相关内容，以泊松过程的理论为基础完成站点的任务产生算法，并使用泊松过程强度参数表征站点的等级请求负荷。结合小车控制算法和任务产生算法，以循环增加最小单位时间、改变任务状态的方式完成了离散化仿真，记录了多组不同楼层数、不同站点等级请求负荷的仿真结果。

4. 通过对选取的三个指标（小车平均速度、实际时间与理想时间比值、停等次数）进行分析，能够看到算法能够在不同请求负荷等级下较好地完成任务，达到缓解拥堵、提高效率的作用。

六、模型的评价与改进

6.1 优点

- 系统建模合理，所有题目数据都未被忽略
- 算法简单，解释性强，易设计
- 贪心算法，相比其他大部分算法能较快够找到解决方案
- 从原理上分析，模型接近最优
- 有一定创新型，为领域内的相关研究提供新思路

6.2 缺点

- 使用 python 进行离散系统的仿真，运行十分缓慢
- 效率可能不及其他启发式算法
- 某些假设可能过于理想化，不完全符合实际
- 最优路径搜索过程中的最大递归深度不易掌握，太大可能爆栈，太小可能找不到最优方案

6.3 改进方向

- 尝试使用启发式算法，降低时间复杂度
- 在原数据的基础上根据算法直接仿真，找到最大可能出现拥挤的点，增设变轨器继续仿真
- 使用专业的仿真软件进行仿真，既可以提高精度，也可以减少设计难度

七、参考文献

[1] 姜启源. 数学模型(第二版)[M]. 高等教育出版社, 1987.

[2] 司守奎, 孙玺菁. 数学建模算法与应用[M]. 国防工业出版社, 2011.

[3] ThomasH.Cormen, 科曼, Leiserson,等. 算法导论[M]. 机械工业出版社, 2006.

[4] 张中. 医用轨道物流传输系统调度规划研究与仿真系统设计[D]. 南京航空航天大学.

[5] 杨凉山. 动态车辆路径规划和调度问题的研究[D]. 同济大学, 2004.

[6] Psaraftis H N . A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem[J]. Transportation Science, 1980, 14(2):130-154.

[7] 林元烈. 应用随机过程[M]. 清华大学出版社, 2002.

[8] 张晓军, 陈良均. 随机过程及应用习题集[M]. 清华大学出版社, 2011.

[9] 徐全智, 吕恕. 概率论与数理统计[M]. 高等教育出版社, 2010.

八、附件

8.1 附件清单

附件一 信息提取 Python 程序

附件二 仿真 Python 程序

附件三 可视化 matlab 程序

8.2 附件

附件一 信息提取 Python 程序

```
#InfoGet.py
    from math import*
import networkx as nx
```

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
def DisCal(a,b):
    return sqrt((a[0]-b[0])**2+(a[1]-b[1])**2+(a[2]-b[2])**2)
def add(a,b):
    return [a[i]+b[i] for i in range(3)]
def cor2str(cor):
    return str(cor[0])+','+str(cor[1])+','+str(cor[2])

f=open('node4.txt')
id_cor=dict()
cor_id=dict()
id_m=dict()
for line in f:
    ele=[]
    num=[]
    for letter in line:
        if letter!=' ':
            num.append(letter)
        if letter==' ' and num!=[]:
            ele.append(int(''.join(num)))
            num=[]
    if num!=[]:
        ele.append(int(''.join(num)))
    id_cor[ele[0]]=ele[1:4]
keys=list(id_cor.keys())
print('node',len(keys))
value=list(id_cor.values())
```

```
for key in keys:
    cor=id_cor[key]
    cor_id[cor2str(cor)]=key


f=open('edge4.txt')
edge=[]
for line in f:
    ele=[]
    num=[]
    for letter in line:
        if letter!=' ':
            num.append(letter)
        if letter==' ' and num!=[]:
            ele.append(int(''.join(num)))
            num=[]
    if num!=[]:
        ele.append(int(''.join(num)))
    edge.append(ele)
print('edge',len(edge))
f=open('changer4.txt')
changer=[]
for line in f:
    ele=[]
    num=[]
    for letter in line:
        if letter!=' ':
            num.append(letter)
        if letter==' ' and num!=[]:
            ele.append(int(''.join(num)))
            num=[]
```

```
        if num!=[]:
            ele.append(int(''.join(num)))
        changer.append(ele)
print('changer',len(changer))
'''
print(edge)
print(len(edge))
flag=True
numb=0
for i in range(len(edge)-1):
        if edge[i][1]!=edge[i+1][0]:
            numb+=1
            flag=False
print(flag,numb)
'''


Sites=[]
ChangerP=[]
straight=[]
nodes=[]
for i in changer:
        ChangerP.append(i[0])
        ChangerP.append((i[1]))

for i in range(len(edge)-1):
        if i==0:
            s0=edge[0]
        a = edge[i][0]
        b = edge[i][1]
```

```
        c = edge[i + 1][1]
        x1 = id_cor[b][0] - id_cor[a][0]
        x2 = id_cor[c][0] - id_cor[b][0]
        y1 = id_cor[b][1] - id_cor[a][1]
        y2 = id_cor[c][1] - id_cor[b][1]
        z1 = id_cor[b][2] - id_cor[a][2]
        z2 = id_cor[c][2] - id_cor[b][2]
        if edge[i+1][0]==edge[i][1] and x1*x2+y1*y2+z1*z2!=0:
            s0.append(edge[i+1][1])
        else:
            straight.append(s0)
            s0=edge[i+1]
straight.append(s0)
count=0
print(len(straight))
stu=[]
for i in straight:
    stu.append(i)
for i in range(len(straight)):
    s0=stu[i]
    xl=[id_cor[s0[-1]][i]-id_cor[s0[0]][i] for i in range(3)]
    dis=int(DisCal(id_cor[s0[0]],id_cor[s0[-1]]))
    if dis==600 and xl[0]!=0:
        xld3=[d//3 for d in xl]
        x1=add(id_cor[s0[0]],xld3)
        x2=add(id_cor[s0[0]],2*xld3)
        Sites.append(id_cor[s0[0]])
        Sites.append(x1)
        Sites.append(x2)
        straight.append([s0[0],s0[1]])
```

```
            straight.append([s0[1],s0[2]])
            straight.append([s0[2],s0[3]])
            straight.remove(s0)

for i in changer:
    straight.append([i[0],i[1]])

for i in Sites:
    nodes.append(i)
print('sites',len(Sites))
for i in ChangerP:
    nodes.append(id_cor[i])
stu=[]
for i in straight:
    stu.append(i)
for i in stu:
    extr=[i[0]]
    for j in range(1,len(i)-1):
        if id_cor[i[j]] in nodes:
            extr.append(i[j])
    if len(extr)!=1:
        extr.append(i[-1])
        for j in range(len(extr)-1):
            straight.append([extr[j],extr[j+1]])
        straight.remove(i)
for i in range(len(straight)):
    s0=straight[i]
    if len(s0)!=2:
        straight[i]=[s0[0],s0[-1]]
'''
```

```
while True:
    flag=0
    stn=[]
    for i in straight:
        stn.append(i)
    for i in straight:
        for j in straight:
            if i!=j and i[1]==j[0] and (id_cor[i[1]] not in nodes):
                stn.append([i[0],j[1]])
                stn.remove(i)
                stn.remove(j)
                flag=1
                break
        if flag==1:
            break
    straight=stn
    if lstr==len(stn):
        break
    else:
        lstr = len(stn)
print('lstr',lstr)
'''
print('new_edge',len(straight))



l=len(nodes)
print('nodes',l)

#nodes  站点+变轨器端点
#rnode  所有节点
```

```
rnode=[]
for i in straight:
    if i[0] not in rnode:
        rnode.append(i[0])
    if i[1] not in rnode:
        rnode.append(i[1])

flag=0
print(rnode)
for i in rnode:
    if i==100001:
        flag=1
        break
print(flag)
for i in range(len(rnode)):
    id_m[rnode[i]]=i

matrix=[[10000 for i in range(len(rnode))] for i in range(len(rnode))]
for s0 in straight:
    a=s0[0]
    b=s0[1]
    cor1=id_cor[a]
    cor2=id_cor[b]
    dis=int(DisCal(cor1,cor2))
    matrix[id_m[a]][id_m[b]] = dis
    matrix[id_m[b]][id_m[a]] = dis
print('newnode',len(matrix))
print(matrix)

#定义坐标轴
```

```
fig = plt.figure()
ax1 = plt.axes(projection='3d')
#ax = fig.add_subplot(111,projection='3d')   #这种方法也可以画多个子图
for i in straight:
    cor1=id_cor[i[0]]
    cor2=id_cor[i[1]]
    x1=[cor1[0],cor2[0]]
    y1=[cor1[1],cor2[1]]
    z1=[cor1[2],cor2[2]]
    ax1.plot3D(x1,y1,z1,'gray')       #绘制空间曲线
plt.show()
G = nx.Graph()
Matrix = np.array(matrix)
edges=0
for i in range(len(Matrix)):
    for j in range(len(Matrix)):
        if Matrix[i,j] != 10000:
            G.add_edge(i, j)
            edges+=1
print(edges/2)
nx.draw(G,node_size = 20, pos = nx.spring_layout(G))
plt.show()
```

附件二 仿真 Python 程序

```
from math import*
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
from random import choice
import seaborn as sns
```

```
from mpl_toolkits.mplot3d import Axes3D
from copy import *
def DisCal(a,b):
    return sqrt((a[0]-b[0])**2+(a[1]-b[1])**2+(a[2]-b[2])**2)
def add(a,b):
    return [a[i]+b[i] for i in range(3)]
def cor2str(cor):
    return str(cor[0])+','+str(cor[1])+','+str(cor[2])
def dijkstra(distmat,a):
    distMat=deepcopy(distmat)
    maxdis = max([max(b) for b in distMat])
    road=dict()
    l=len(distMat)
    visit=set()
    visit.add(a)
    unvisit=set()
    for i in range(l):
        if distMat[a][i]==0:
            road[i]=[i]
            continue
        if distMat[a][i]!=maxdis:
            road[i]=[a,i]
    for i in range(l):
        if i==a:
            continue
        unvisit.add(i)
    while len(unvisit)!=0:
        uv=list(unvisit)
        id=uv[0]
        minimum=distMat[a][id]
```

```
            for i in uv:
                if distMat[a][i]<minimum:
                    minimum=distMat[a][i]
                    id=i
            visit.add(id)
            unvisit.remove(id)
            uv=list(unvisit)
            for i in uv:
                if distMat[a][id]+distMat[id][i]<distMat[a][i]:
                    distMat[a][i] = distMat[a][id] + distMat[id][i]
                    distMat[i][a] = distMat[a][id] + distMat[id][i]
                    roadid=road[id].copy()
                    roadid.append(i)
                    road[i]=roadid
        return road
def dijkstra2(distmat,a,destination):
        distMat=deepcopy(distmat)
        maxdis = max([max(b) for b in distMat])
        road=dict()
        l=len(distMat)
        visit=set()
        visit.add(a)
        unvisit=set()
        for i in range(l):
            if distMat[a][i]==0:
                road[i]=[i]
                continue
            if distMat[a][i]!=maxdis:
                road[i]=[a,i]
        for i in range(l):
```

```
            if i==a:
                continue
            unvisit.add(i)
        id=-1
        while len(unvisit)!=0 and destination!=id:
            uv=list(unvisit)
            id=uv[0]
            minimum=distMat[a][id]
            for i in uv:
                if distMat[a][i]<minimum:
                    minimum=distMat[a][i]
                    id=i
            visit.add(id)
            unvisit.remove(id)
            uv=list(unvisit)
            for i in uv:
                if distMat[a][id]+distMat[id][i]<distMat[a][i]:
                    distMat[a][i] = distMat[a][id] + distMat[id][i]
                    distMat[i][a] = distMat[a][id] + distMat[id][i]
                    roadid=road[id].copy()
                    roadid.append(i)
                    road[i]=roadid
        return road[destination]

#设速度为 1dm/s
#根据邻接矩阵生成距离矩阵
def disMatGet(matrix):
        distmat=deepcopy(matrix)
        l = len(distmat)
        road = dict()
```

```
    for i in range(l):
        print(i)
        roadi = dijkstra(distmat, i)
        for key in roadi.keys():
            if i == key:
                continue
            road[str(i)+','+str(key)] = roadi[key]
    return road
def find_all_path(distmat, visited, start, end):
    if start==end:
        return [[end]]
    newvisi=deepcopy(visited)
    newvisi.append(start)
    all_path=[]
    nextn=[]
    l=len(distmat)
    for i in range(l):
        if distmat[start][i]!=10000 and distmat[start][i]!=0 and (i not in newvisi):
            nextn.append(i)
    for i in nextn:
        newpath=find_all_path(distmat, newvisi, i, end)
        for i in newpath:
            path = [start]
            for j in i:
                path.append(j)
            if len(path)!=1:
                all_path.append(path)
    return all_path
#以 numbda 的强度产生时长 t 内发生的任务
def missionGen(Sites_id,nambda,time):
```

```
    n = np.random.poisson(nambda * time)
    # 生成 n 个[0, T]均匀分布随机数并排序
    t = [int(i) for i in np.sort(np.random.random(n) * time)]
    mission=[]
    for i in range(len(t)):
        s=choice(Sites_id)
        e=choice(Sites_id)
        while s==e:
            e=choice(Sites_id)
        mission.append([t[i],s,e])
    return mission
#侦测冲突
def confliDetec(tspan, road1, road2, rnode, id_cor):
    #tspan 轨迹二滞后轨迹一的时间
    state1=[]
    state2=[]
    if tspan<0:
        for i in range(abs(tspan)):
            state1.append([-1, -1])
    else:
        for i in range(abs(tspan)):
            state2.append([-1, -1])
    for i in range(len(road1)-1):
        time=int(DisCal(id_cor[rnode[road1[i]]],id_cor[rnode[road1[i+1]]]))
        for j in range(time):
            state1.append([road1[i],road1[i+1]])
    for i in range(len(road2)-1):
        time=int(DisCal(id_cor[rnode[road2[i]]],id_cor[rnode[road2[i+1]]]))
        for j in range(time):
            state2.append([road2[i],road2[i+1]])
```

```
    judge = False
    for i in range(min(len(state1),len(state2))):
        s1=state1[i]
        s2=state2[i]
        if s1[0]==s2[1] and s1[1]==s2[0]:
            st=s2[0]
            p = 0
            k=state1[i]
            for j in range(i,min(len(state1),len(state2))):
                if state1[j]==k:
                    print(state1[j])
                    p+=1
                else:
                    break
            judge=True
            break
    if judge==False:
        return [judge]
    else:
        return [judge,p,st]
#模拟，形成最佳路径
def pathdis(path,distmat):
    pathd=0
    for i in range(len(path)-1):
        pathd+=distmat[path[i]][path[i+1]]
    return pathd
def pathPlan(misAgoing,matrix,a,b):
    road_ab=dijkstra2(matrix,a,b)
    mintime=int(road_ab/10)
    for i in misAgoing:
```

```
            if confliDetec(t,i,road_ab)==True:
                find()
                Matrix=deepcopy(matrix)
                pathPlan()
def simulation(mission,distmat,time,rnode,id_cor):
    misAgoing = []
    avalable = [1]
    for i in range(time):
        for misa in misAgoing:
            misa[-2]+=10
            misa[-1]-=10
            if misa[-1]<=0:
                misAgoing.remove(misa)
        print(i+1)
        print(misAgoing)
        for mis in mission:
            if mis[0]==i:
                paths=find_all_path(distmat,[],mis[1],mis[2])
                print('path searched.')
                pathd=[pathdis(path,distmat) for path in paths]
                for j in range(len(paths)):
                    path=paths[j]
                    minextra=10000000
                    for misa in misAgoing:
                        cd=confliDetec(misa[-2],misa[1],path,rnode,id_cor)
                        if len(cd)==1:
                            continue
                        else:
                            print('conflict searched.')
                            if minextra>min(minextra,cd[1]):
```

```
                                st=cd[2]
                                minextra=min(minextra,cd[1])
                        if minextra==10000000:
                                continue
                        else:
                                paths[j].append([st,minextra])
                                pathd[j]+=minextra
                minl=10000000
                idex=0
                for j in range(len(pathd)):
                        if pathd[j]<minl:
                                minl=pathd[j]
                                idex=j
                p=paths[idex]
                if isinstance(p[-1],list):
                        mis.insert(1,p[0:-1])
                else:
                        mis.insert(1, p)
                del mis[2:4]
                mis.append(0)
                mis.append(pathd[idex]-0)
                misAgoing.append(mis)



f=open('node4.txt')
id_cor=dict()
cor_id=dict()
id_m=dict()
for line in f:
        ele=[]
```

```
        num=[]
        for letter in line:
            if letter!=' ':
                num.append(letter)
            if letter==' ' and num!=[]:
                ele.append(int(''.join(num)))
                num=[]
        if num!=[]:
            ele.append(int(''.join(num)))
        id_cor[ele[0]]=ele[1:4]
keys=list(id_cor.keys())
deposit=[]
for i in range(1,max(keys)):
        if len(deposit)==2:
            break
        if i not in keys:
            deposit.append(i)
print(deposit)
print('node',len(keys))
value=list(id_cor.values())

for key in keys:
        cor=id_cor[key]
        cor_id[cor2str(cor)]=key

f=open('edge4.txt')
edge=[]
for line in f:
        ele=[]
        num=[]
```

```
        for letter in line:
            if letter!=' ':
                num.append(letter)
            if letter==' ' and num!=[]:
                ele.append(int(''.join(num)))
                num=[]
        if num!=[]:
            ele.append(int(''.join(num)))
        edge.append(ele)
print('edge',len(edge))
f=open('changer4.txt')
changer=[]
for line in f:
        ele=[]
        num=[]
        for letter in line:
            if letter!=' ':
                num.append(letter)
            if letter==' ' and num!=[]:
                ele.append(int(''.join(num)))
                num=[]
        if num!=[]:
            ele.append(int(''.join(num)))
        changer.append(ele)
print('changer',len(changer))
'''
print(edge)
print(len(edge))
flag=True
numb=0
```

```
for i in range(len(edge)-1):
    if edge[i][1]!=edge[i+1][0]:
        numb+=1
        flag=False
print(flag,numb)
'''

Sites=[]
ChangerP=[]
straight=[]
nodes=[]
for i in changer:
    ChangerP.append(i[0])
    ChangerP.append((i[1]))

for i in range(len(edge)-1):
    if i==0:
        s0=edge[0]
    a = edge[i][0]
    b = edge[i][1]
    c = edge[i + 1][1]
    x1 = id_cor[b][0] - id_cor[a][0]
    x2 = id_cor[c][0] - id_cor[b][0]
    y1 = id_cor[b][1] - id_cor[a][1]
    y2 = id_cor[c][1] - id_cor[b][1]
    z1 = id_cor[b][2] - id_cor[a][2]
    z2 = id_cor[c][2] - id_cor[b][2]
    if edge[i+1][0]==edge[i][1] and x1*x2+y1*y2+z1*z2!=0:
        s0.append(edge[i+1][1])
    else:
```

```
            straight.append(s0)
            s0=edge[i+1]
straight.append(s0)
count=0

stu=[]
for i in straight:
    stu.append(i)
for i in range(len(straight)):
    s0=stu[i]
    xl=[id_cor[s0[-1]][i]-id_cor[s0[0]][i] for i in range(3)]
    dis=int(DisCal(id_cor[s0[0]],id_cor[s0[-1]]))
    if dis==600 and xl[0]!=0:
        xld3=[d//3 for d in xl]
        x1=add(id_cor[s0[0]],xld3)
        x2=add(id_cor[s0[0]],2*xld3)
        Sites.append(id_cor[s0[0]])
        Sites.append(x1)
        Sites.append(x2)
        straight.append([s0[0],s0[1]])
        straight.append([s0[1],s0[2]])
        straight.append([s0[2],s0[3]])
        straight.remove(s0)

for i in changer:
    straight.append([i[0],i[1]])

for i in Sites:
    nodes.append(i)
print('sites',len(Sites))
```

```
for i in ChangerP:
    nodes.append(id_cor[i])
stu=[]
for i in straight:
    stu.append(i)
for i in stu:
    extr=[i[0]]
    for j in range(1,len(i)-1):
        if id_cor[i[j]] in nodes:
            extr.append(i[j])
    if len(extr)!=1:
        extr.append(i[-1])
        for j in range(len(extr)-1):
            straight.append([extr[j],extr[j+1]])
        straight.remove(i)
for i in range(len(straight)):
    s0=straight[i]
    if len(s0)!=2:
        straight[i]=[s0[0],s0[-1]]
'''
while True:
    flag=0
    stn=[]
    for i in straight:
        stn.append(i)
    for i in straight:
        for j in straight:
            if i!=j and i[1]==j[0] and (id_cor[i[1]] not in nodes):
                stn.append([i[0],j[1]])
                stn.remove(i)
```

```
                        stn.remove(j)
                        flag=1
                        break
                if flag==1:
                    break
            straight=stn
            if lstr==len(stn):
                break
            else:
                lstr = len(stn)
        print('lstr',lstr)
        '''
        dep1=deposit[0]
        id_cor[dep1]=[0,0,900]
        cor_id[cor2str([0,0,900])]=dep1
        for i in straight:
            cor1=id_cor[i[0]]
            cor2=id_cor[i[1]]
            x1=cor1[0]-0
            y1=cor1[1]-0
            z1=cor1[2]-900
            x2=cor2[0]-0
            y2=cor2[1]-0
            z2=cor2[2]-900
            if x1*y2-x2*y1==0 and x1*z2-x2*z1==0 and y1*z2-y2*z1==0 and
x1*x2+y1*y2+z1*z2<0:
                straight.append([cor_id[cor2str(cor1)],dep1])
        print('new_edge',len(straight))
```

```
l=len(nodes)
print('nodes',l)

#nodes  站点+变轨器端点
#rnode  所有节点
rnode=[]
for i in straight:
    if i[0] not in rnode:
        rnode.append(i[0])
    if i[1] not in rnode:
        rnode.append(i[1])

for i in range(len(rnode)):
    id_m[rnode[i]]=i

matrix=[[10000 for i in range(len(rnode))] for i in range(len(rnode))]
for s0 in straight:
    a=s0[0]
    b=s0[1]
    cor1=id_cor[a]
    cor2=id_cor[b]
    dis=int(DisCal(cor1,cor2))
    matrix[id_m[a]][id_m[b]] = dis
    matrix[id_m[b]][id_m[a]] = dis
Sites_id=[id_m[cor_id[cor2str(i)]] for i in Sites]

mission=missionGen(Sites_id,0.2,500)
for i in mission:
    print(i)
```

simulation(mission,matrix,500,rnode,id_cor)

附件三 可视化 matlab 程序

```
road=[      100001      100002
     100002      100003
       1001        1002
       1002        1003
       1003        1004
       1004        1005
       1005        1006
       1006        1007
       1007        1008
       1008        1009
       1009        1010
       1010        1011
       1004        1011
       1011        1012
       1012        1013
       1013        1014
       1014        1015
       1015        1016
       1016        1017
       1017        1018
       1018        1019
       1019        1020
       1020        1021
       1014        1021
       1021        1022
       1022        1023
       1023        1024
```

1024	1025
1025	1026
1026	1027
1027	1028
1028	1029
1029	1030
1030	1031
1024	1031
1031	1513
1513	1512
1512	1511
1511	1510
1510	1509
1509	1508
1508	1507
1507	1506
1506	1505
1505	1504
1504	1503
1503	1502
1502	1501
100001	1001
1501	101001
100003	100004
100004	100005
100005	100006
2001	2002
2002	2003
2003	2004
2004	2005

2005	2006
2006	2007
2007	2008
2008	2009
2009	2010
2010	2011
2004	2011
2011	2012
2012	2013
2013	2014
2014	2015
2015	2016
2016	2017
2017	2018
2018	2019
2019	2020
2020	2021
2014	2021
2021	2022
2022	2023
2023	2024
2024	2025
2025	2026
2026	2027
2027	2028
2028	2029
2029	2030
2030	2031
2024	2031
2031	2513

2513	2512
2512	2511
2511	2510
2510	2509
2509	2508
2508	2507
2507	2506
2506	2505
2505	2504
2504	2503
2503	2502
2502	2501
100004	2001
2501	101004
100006	100007
100007	100008
100008	100009
3001	3002
3002	3003
3003	3004
3004	3005
3005	3006
3006	3007
3007	3008
3008	3009
3009	3010
3010	3011
3004	3011
3011	3012
3012	3013

3013 3014

3014 3015

3015 3016

3016 3017

3017 3018

3018 3019

3019 3020

3020 3021

3014 3021

3021 3022

3022 3023

3023 3024

3024 3025

3025 3026

3026 3027

3027 3028

3028 3029

3029 3030

3030 3031

3024 3031

3031 3513

3513 3512

3512 3511

3511 3510

3510 3509

3509 3508

3508 3507

3507 3506

3506 3505

3505 3504

3504	3503
3503	3502
3502	3501
100007	3001
3501	101007
100009	100010
4001	4002
4002	4003
4003	4004
4004	4005
4005	4006
4006	4007
4007	4008
4008	4009
4009	4010
4010	4011
4004	4011
4011	4012
4012	4013
4013	4014
4014	4015
4015	4016
4016	4017
4017	4018
4018	4019
4019	4020
4020	4021
4014	4021
4021	4022
4022	4023

4023	4024
4024	4025
4025	4026
4026	4027
4027	4028
4028	4029
4029	4030
4030	4031
4024	4031
4031	4513
4513	4512
4512	4511
4511	4510
4510	4509
4509	4508
4508	4507
4507	4506
4506	4505
4505	4504
4504	4503
4503	4502
4502	4501
100010	4001
4501	101010
101002	101001
101003	101002
101004	101003
101005	101004
101006	101005
101007	101006

```
        101008      101007
        101009      101008
        101010      101009];

changer=[
            100001      101001
          1011          1505
          1021          1509
        100004      101004
          2011          2505
          2021          2509
        100007      101007
          3011          3505
          3021          3509
        100010      101010
          4011          4505
          4021          4509]

index=[
            100001          0           0           0
        100002              0           0         200
        100003              0           0         400
          1001              0         400           0
          1002              0         600           0
          1003              0         800           0
          1004              0        1000           0
          1005            200        1000           0
          1006            400        1000           0
          1007            600        1000           0
          1008            600        1040           0
```

1009	400	1040	0
1010	200	1040	0
1011	0	1040	0
1012	0	1240	0
1013	0	1440	0
1014	0	1640	0
1015	200	1640	0
1016	400	1640	0
1017	600	1640	0
1018	600	1680	0
1019	400	1680	0
1020	200	1680	0
1021	0	1680	0
1022	0	1880	0
1023	0	2080	0
1024	0	2280	0
1025	200	2280	0
1026	400	2280	0
1027	600	2280	0
1028	600	2320	0
1029	400	2320	0
1030	200	2320	0
1031	0	2320	0
1513	-30	2320	0
1512	-30	2280	0
1511	-30	2080	0
1510	-30	1880	0
1509	-30	1680	0
1508	-30	1640	0
1507	-30	1440	0

1506	-30	1240	0
1505	-30	1040	0
1504	-30	1000	0
1503	-30	800	0
1502	-30	600	0
1501	-30	400	0
100004	0	0	600
100005	0	0	800
100006	0	0	1000
2001	0	400	600
2002	0	600	600
2003	0	800	600
2004	0	1000	600
2005	200	1000	600
2006	400	1000	600
2007	600	1000	600
2008	600	1040	600
2009	400	1040	600
2010	200	1040	600
2011	0	1040	600
2012	0	1240	600
2013	0	1440	600
2014	0	1640	600
2015	200	1640	600
2016	400	1640	600
2017	600	1640	600
2018	600	1680	600
2019	400	1680	600
2020	200	1680	600
2021	0	1680	600

2022	0	1880	600
2023	0	2080	600
2024	0	2280	600
2025	200	2280	600
2026	400	2280	600
2027	600	2280	600
2028	600	2320	600
2029	400	2320	600
2030	200	2320	600
2031	0	2320	600
2513	-30	2320	600
2512	-30	2280	600
2511	-30	2080	600
2510	-30	1880	600
2509	-30	1680	600
2508	-30	1640	600
2507	-30	1440	600
2506	-30	1240	600
2505	-30	1040	600
2504	-30	1000	600
2503	-30	800	600
2502	-30	600	600
2501	-30	400	600
100007	0	0	1200
100008	0	0	1400
100009	0	0	1600
3001	0	400	1200
3002	0	600	1200
3003	0	800	1200
3004	0	1000	1200

3005	200	1000	1200
3006	400	1000	1200
3007	600	1000	1200
3008	600	1040	1200
3009	400	1040	1200
3010	200	1040	1200
3011	0	1040	1200
3012	0	1240	1200
3013	0	1440	1200
3014	0	1640	1200
3015	200	1640	1200
3016	400	1640	1200
3017	600	1640	1200
3018	600	1680	1200
3019	400	1680	1200
3020	200	1680	1200
3021	0	1680	1200
3022	0	1880	1200
3023	0	2080	1200
3024	0	2280	1200
3025	200	2280	1200
3026	400	2280	1200
3027	600	2280	1200
3028	600	2320	1200
3029	400	2320	1200
3030	200	2320	1200
3031	0	2320	1200
3513	-30	2320	1200
3512	-30	2280	1200
3511	-30	2080	1200

3510	-30	1880	1200
3509	-30	1680	1200
3508	-30	1640	1200
3507	-30	1440	1200
3506	-30	1240	1200
3505	-30	1040	1200
3504	-30	1000	1200
3503	-30	800	1200
3502	-30	600	1200
3501	-30	400	1200
100010	0	0	1800
4001	0	400	1800
4002	0	600	1800
4003	0	800	1800
4004	0	1000	1800
4005	200	1000	1800
4006	400	1000	1800
4007	600	1000	1800
4008	600	1040	1800
4009	400	1040	1800
4010	200	1040	1800
4011	0	1040	1800
4012	0	1240	1800
4013	0	1440	1800
4014	0	1640	1800
4015	200	1640	1800
4016	400	1640	1800
4017	600	1640	1800
4018	600	1680	1800
4019	400	1680	1800

4020	200	1680	1800
4021	0	1680	1800
4022	0	1880	1800
4023	0	2080	1800
4024	0	2280	1800
4025	200	2280	1800
4026	400	2280	1800
4027	600	2280	1800
4028	600	2320	1800
4029	400	2320	1800
4030	200	2320	1800
4031	0	2320	1800
4513	-30	2320	1800
4512	-30	2280	1800
4511	-30	2080	1800
4510	-30	1880	1800
4509	-30	1680	1800
4508	-30	1640	1800
4507	-30	1440	1800
4506	-30	1240	1800
4505	-30	1040	1800
4504	-30	1000	1800
4503	-30	800	1800
4502	-30	600	1800
4501	-30	400	1800
101001	-30	0	0
101002	-30	0	200
101003	-30	0	400
101004	-30	0	600
101005	-30	0	800

```
        101006          -30             0           1000
        101007          -30             0           1200
        101008          -30             0           1400
        101009          -30             0           1600
        101010          -30             0           1800]

    rl=length(road)
    for i=1:rl
        a=road(i,1);
        b=road(i,2);
        ida=find(index(:,1)==a);
        idb=find(index(:,1)==b);
        x=[index(ida,2),index(idb,2)];
        y=[index(ida,3),index(idb,3)];
        z=[index(ida,4),index(idb,4)];
    %       patch(x,y,z,'edgecolor','flat','facecolor','none')

quiver3(index(ida,2),index(ida,3),index(ida,4),index(idb,2)-index(ida,2),index(idb,3)-index(ida,3),
index(idb,4)-index(ida,4))
        %plot3([index(ida,2),index(idb,2)],[index(ida,3),index(idb,3)],[index(ida,4),index(idb,4)]
);
        hold on
    end
    for i=1:length(changer)
        a=changer(i,1);
        b=changer(i,2);
        ida=find(index(:,1)==a);
        idb=find(index(:,1)==b);
        x=[index(ida,2),index(idb,2)];
        y=[index(ida,3),index(idb,3)];
```

```
        z=[index(ida,4),index(idb,4)];
    %       patch(x,y,z,'edgecolor','flat','facecolor','none')

plot3([index(ida,2),index(idb,2)],[index(ida,3),index(idb,3)],[index(ida,4),index(idb,4)],'b');
        hold on
    end
```