第六届湖南省研究生数学建模竞赛

题 目　基于混合化学反应优化算法的交通信号灯全局优化调控策略设计

摘　　要：

交通信号灯在城市交通中发挥了巨大的作用，面对给定交通路况信息合理设计每个路口交通信号灯调控策略，可以大幅提升道路通行能力，避免交通拥堵。交通规划调控类问题是典型的动态规划问题，每个路口交通信号灯策略的不同调控都会给全局带来影响。这类影响很难用公示给出映射，而且解空间复杂，总共有 8000 个路口和 63968 条路段。为此我们在问题 1 中借助了化学反应算法搜索全局最优解。并且通过数据预处理和创新性地使用数字对单个路口亮绿灯顺序进行编码，解决了问题 1 求解中存在的数据复杂和调控策略难以编码问题，最终迭代得到最佳调控策略存入 result1. txt，最佳策略对应得分为 128522。还对化学反应算法进行了复杂度分析、收敛性分析和鲁棒性分析，虽然随着城市规模增大复杂度会增长的很快，但是我们为复杂城市的交通调控问题提出了可行的解决方案，并且验证了较遗传算法，化学反应算法性能更优。在问题 2 中，我们使用 Dijkstra 算法求解道路故障发生后受影响小车最短路径规划问题；将道路故障发生后局部调整方案建模为小规模寻优问题，综合考虑最终得分和调整范围，利用传统粒子群算法快速求解得到调整后的信号灯调控策略，该策略下按照小车行驶时间计算得分为 135873。比较发现问题 1 中调控策略在故障发生后性能下降 15%左右，调整后调控策略较问题 1 中策略在故障发生后性能提升 27%左右。从而说明了故障发生会使调控策略性能下降，产生资源浪费和调控不适用情况。

关键词：交通灯调控、化学反应算法、粒子群算法、Dijkstra 算法

目　录

一、问题重述与分析

交通信号灯全局优化调控策略研究问题是一个典型的多目标优化问题，题目中具体分为两个问题：

（一）问题 1

问题 1 中给定了全市路网结构、道路信息，以及每辆小车的行驶路线。要求在已知上述信息条件下，确定需要调控的路口数量，为每个路口设计交通信号灯调控策略，对每个路口信号灯亮绿灯的顺序和绿灯持续时间进行设计，最终服务于全局，使每辆小车到达终点时间尽量短，从而实现所有车辆的得分累积和最高。

这是一个常见的优化问题，核心问题在于总共有 200 辆小车，8000 个路口和 63968 条路段，数据量大，维度高，解空间复杂。另外，调控策略建模也有难度，对每个路口调控策略的设计包含两部分内容：亮绿灯的顺序和绿灯持续时间。二者相互独立且共同影响最终策略性能。

针对这类问题，我们首先考虑对数据进行预处理，结合题目已知信息数据，对于这 200 辆车，大部分车行驶路线由 5-30 条路段组成，上限只有 $200\times30=6000$ 左右，也就是说，就算所有小车行驶路线包含 30 条路段且互相不重复，那么最终涉及到的路线也只有 6000 条，远远小于题目中给出的总路线数——63968 条，从而我们可以先对 200 辆车的道路和路口进行筛选汇总，从而大幅度缩小解空间数量级。

其次，对于这类典型多目标优化问题，最常用的算法是智能搜索算法，查阅资料并综合对比各类智能算法，我们决定选取适用度高、性能优越的化学反应算法来进行处理，并针对本题具体情况进行调整改进，优化求解。

至于调控策略建模问题，主要考虑如何将其建模能够利用算法进行求解。绝大部分智能算法涉及到编码和算子操作两部分内容。这个问题难点就主要体现在编码上，为了让算法有效进行，每次需要改变或者交换编码中一个码元，使调控策略发生变化，且能够控制变化范围有时候只发生在单个信号灯亮绿灯情况上，为此我们创新性的提出用数字编码代表信号灯亮绿灯顺序，这样完成了“策略”→“数字”的映射，使得各项编码、算子操作等得以实现，问题能够得到有效地求解。

（二）问题 2

在问题 1 的基础上，小车行驶一段时间后突发道路故障，道路 ebf-ffef 的起点处永久封闭，其他道路正常通行。故障发生后，已经经过该路口小车继续行驶，需要经过但还未经过的小车——受影响小车，需要到达最邻近的终点路口后，按照里程最短原则重新规划行驶路线后继续行驶，这就需要重新调整信号灯调控策略。

该问题较为复杂，包含多个子问题，具体可以分解为下述四个问题：

- 在突发道路故障后，为受影响车辆按照里程最短原则重新规划路线；
- 设计局部调整调控方案的方法；
- 讨论道路突发事件对交通调控方案影响；
- 给出调整后的全局交通信号灯调控方案；

其中第一个子问题只是一个普通最短路径规划问题，且问题较为简单，不涉及路径冲突等复杂因素，唯一的难点是道路故障发生即 $t=30$ s 时全局状态和路网的复杂带来的规划难度。为此，我们在解题过程中设计全局车辆状态表，更新

每一时刻小车当前处于的道路和具体状态——等待红灯、正常行驶、行程结束等等，并且使用 Dijkstra 最短路径算法为受影响的每辆小车规划最短路径。

第二个子问题是设计局部调整的方法，道路故障发生后，新的交通信号灯调控也是一个寻优问题，虽然模型与问题 1 相同，但场景变化后我们需要考虑新的优化目标：交通故障发生后第一要务都是疏散当前路口车辆、降低故障带来影响并且还需要避免该故障造成的影响扩散到很大范围。前半部分可以用重新调控后每辆小车到达终点的“时间短”来表述，具体可以刻画为与行驶时间相关的函数，到达终点行驶时间越短、得分越高；后半部分可以用需要调控的范围来表示，范围越大、得分越低。最终以总得分表示这种调整信号灯调控策略的性能优劣。为了保证故障发生后的“快速反应”，结合前面考虑的“局部调整”，我们选择速度更快的粒子群算法进行求解，在小型问题上，粒子群算法简单易行、收敛快的特点能很好满足我们上述需求。

第三个子问题因为道路故障带来的影响无法通过直接公式刻画，因此我们计划用局部调整前后调控策略性能差异来分析，分别分析故障发生前后原有策略性能下降程度和故障发生后新旧调控策略性能差异。

第四个子问题在前面基础上则更为简单，得到了故障发生后小车新的路线和调整调控策略的方法，带入数据即可得到。对该部分工作的分析在求解子问题三部分便已做出分析，在此部分不再赘述。

二、模型假设

（1）每条道路均为单向通行，连接两个不同的路口。

（2）连接两个路口的道路，同一方向至多有一条。

（3）每辆车行驶状态相互独立、互不影响，行驶速度均匀且相同，可用畅通状态下的通行时间表示一条道路的长度。

（4）每条道路的终点路口设置一个交通灯，交通灯只有红色和绿色 2 种颜色，亮红色时，所有到达该路口的车辆需排队等待绿灯，亮绿色时，车辆可以通过该路口。

（5）交通灯变成绿灯后，同一条道路的终点处，ID 序号较小的车辆排在前面先行，且队列中第一辆车可以无延迟地立刻通过路口。

（6）在任何时刻，每个路口最多只有一个交通灯是绿灯。

（7）全部车辆的行程计划不可改变，且行程经过的路口最多只能经过一次。

（8）忽略车辆的几何尺寸，空间上的差异不会对车辆排队状况造成影响。

（9）初始状态(即 0 时刻)，所有的车辆都在各自路径上第一条道路的终点处，到达终点后，车辆立即被移出。

（10）每个路口处交通灯的调控方案都可以独立设置，不受其他路口的交通灯影响。

三、符号定义与说明

符号	含义说明
i, j	路口序号

k	车辆序号
$s_{i,j}$	以路口 i 为起点，路口 j 为终点的有向道路
I	所有路口集合
V	所有车辆集合
S	所有道路集合
N_I	路口总数
N_V	车辆总数
N_S	道路总数
$node\{\rightarrow i\}$	所有车辆行程中道路指向路口 i 的路口集合
$N_{node\{\rightarrow i\}}$	路口 i 的道路总数
$N_t^{(i)}$	在时刻 t 到达路口 i 的车辆总数
$\Psi_t^{(i)}$	在时刻 t 到达路口 i 的车辆序号集合
$N_{t,s_{i,j}}^{(j)}$	在时刻 t 由同一道路 $s_{i,j}$ 到达路口 j 的车辆总数
$\Psi_{t,s_{i,j}}^{(j)}$	在时刻 t 由同一道路 $s_{i,j}$ 到达路口 j 的车辆序号集合
$\sigma_k^{s_{i,j}}$	车辆 k 到达路口 j 时在同一道路 $s_{i,j}$ 等待绿灯时的排队次序
Θ_k	车辆 k 行程中包含的道路集合
Ξ_k	车辆 k 行程中包含的路口集合
P_k	车辆 k 行程路径的道路总数
T_k	车辆 k 到达终点的时刻
D	总时长

F	车辆按时到达终点的基础得分
M_k	车辆 k 的最终得分
$T_{s_{i,j}}$	车辆通过道路 $s_{i,j}$ 所需时间
$\Omega_{s_{i,j}}$	道路 $s_{i,j}$ 的绿灯所处状态的标识符
$o_{s_{i,j}}$	道路 $s_{i,j}$ 在终点路口 j 的绿灯点亮次序
$T^{s_{i,j}}$	车辆通过道路 $s_{i,j}$ 所需时间
$t_{left}^{o_{s_{i,j}}}$	道路 $s_{i,j}$ 的终点路口 j 处绿灯的剩余持续时间
$t_{duration}^{o_{s_{i,j}}}$	道路 $s_{i,j}$ 在终点路口 j 的绿灯点亮持续时间
$t_k^{(i)}$	车辆 k 在第 i 个路口等待总时长
$t_{k,\delta}^{(i)}$	车辆 k 在第 i 个路口因绿灯点亮次序造成等待的未满一周期时长
$n_{k,period}^{(i)}$	车辆 k 在第 i 个路口等待红绿灯满一个周期的数量
$t_{period}^{(i)}$	第 i 个路口的红绿灯周期时长
$t_{k,crowd}^{(i)}$	车辆 k 因排队拥堵等待前车驶离时长
$n_{k,f}^{s_{j,i}}$	在道路 $s_{j,i}$ 上车辆 k 因排队等候时前方的车辆总数
$n_k^{(i)}$	车辆 k 到达路口 i 的次数
Δt	车辆通过路口所需时长
E_{best}	粒子群局部最优解
ρ_i	路口 i 的扩散范围
Q_{best}	粒子群全局最优解

四、问题 1 模型建立与求解

4.1 交通信号灯调控策略模型

交通灯优化调控问题，是一个多目标优化问题，希望所有车尽可能快的到达终点。题目中已知全城的路网结构、所有车辆的行驶路线以及每条路的行驶时间。在排队、过马路、道路交叉等建模复杂的问题上也已经做出假设简化：

- 当某道路的交通灯为红灯时，所有到达该道路终点的车辆排队等待绿灯，当多辆车在同一道路尽头排队等待绿灯时，车标序号小的先行；
- 当交通灯变为绿灯后，队列中第一辆车可以立即通过路口，没有延迟，每辆车通过路口的时间均为 1 秒；
- 每条道路仅表示从该道路起点到该道路终点的单向道路，连接两个路口的道路，同一方向至多有一条，连接两个路口的双向道用两条方向相反的道路表示，每条道路的中间不会与其他道路交叉。

在本系统中，交通路口由若干条驶入道路与驶出道路构成，不与现实交通系统对应，没有路口右转、左转、直行规则，每个路口连接道路数量不固定，在该路口，每条驶入道路都有一个红绿灯，红绿灯只有“红灯亮”、“绿灯亮”两种状态，绿灯亮则该道路车辆驶入路口，红灯亮则排队等待，如下图所示。

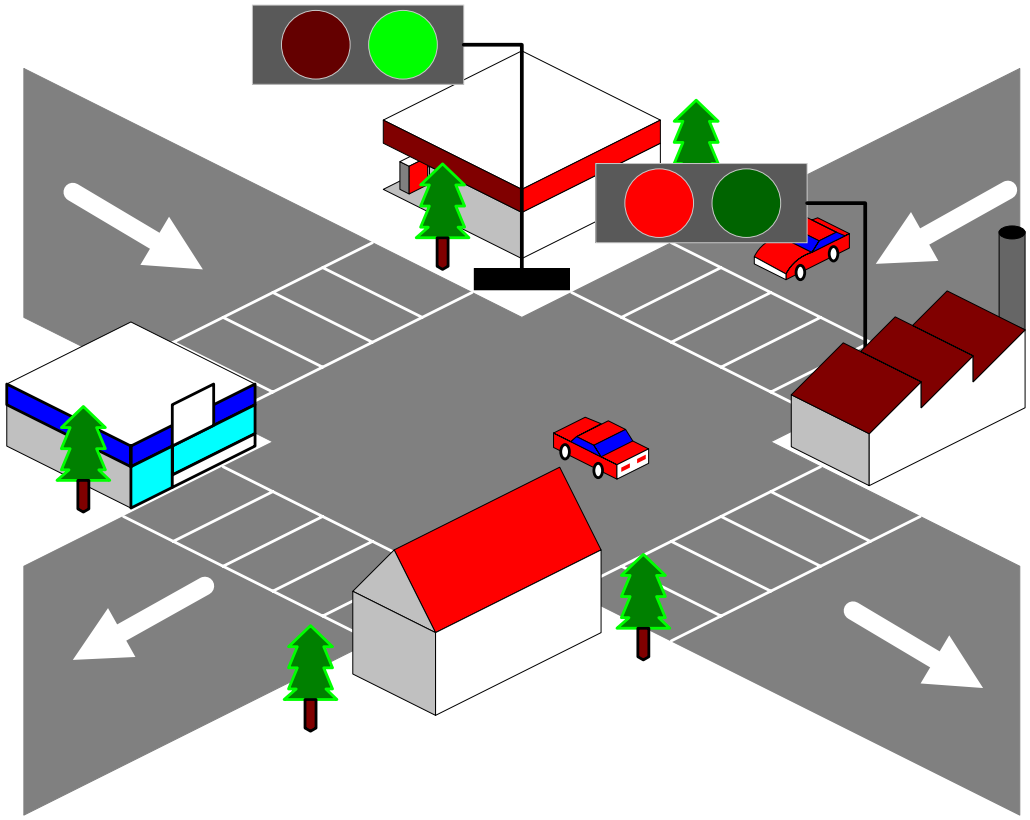

图 1 交通路口示意图

同一路口的若干交通灯组成该路口交通灯优化调控策略，该策略由每个交通灯亮绿灯的顺序和绿灯持续时间构成，若某一路口持续红灯，不亮绿灯，则在交通灯调控策略中依旧有亮绿灯的顺序，但绿灯持续时间为 0。按顺序所有绿灯亮起一遍为一个周期，不同路口调控策略相互独立、互不干扰。最终所有路口的交通灯调控策略组成全局交通灯调控策略，需要注意的是，虽然任意两个路口交通灯调控策略相互独立、互不干扰，但任意一个路口交通灯调控策略的微小变化，

都会影响全局车辆通行情况变化，包括排队时间、等待绿灯时间变化等，并且难以给出影响的数学模型、无法有效预估。

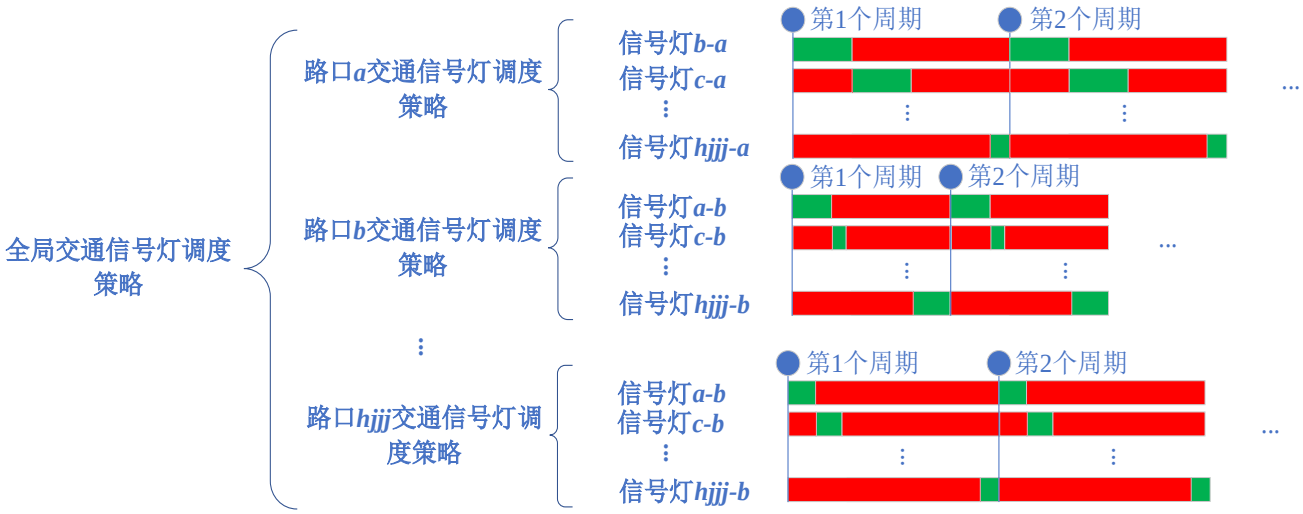

图 2 交通灯策略组成示意图

4.2 小车行驶模型

在给定交通灯策略情况下，本文的优化目标是尽可能提高道路通行能力、减小通行时间。在已知路网情况、道路信息以及每个路口交通灯调控策略后，每个小车按照规定路线从起点行驶到终点的总时间是可以得到的。具体由在路口等待总时间、道路通行总时间、通过路口总时间决定。

4.2.1 道路通行总时间计算

路段内通行总时间是各路段内通行时间的累加，题目中假定车辆行驶速度相同，又因为某一路段的开始路口：每一时刻最多一个绿灯亮起，同一路口多辆车辆排队按照序号先后通行。因此任一时刻，进入路段的车辆最多只有一辆，且每辆车速度相同，不存在道路内拥挤情况。小车到达路段尽头路口，若为绿灯直接通行，若为红灯则排队等待，但不考虑每辆车的几何大小，忽略车辆排队对空间的要求。因此在路段尽头路口排队等待绿灯期间不存在因为道路尽头排队导致空间上的前后使道路内行驶路程变短的问题。因此路段内通行总时间可以简单的由各路段通行时间累加得到。题目中假定车辆行驶速度相同，用畅通状态下的通行时间表示每条道路的长度各路段内通行时间，路段内通行总时间可以表示为：

$$t_1 = \sum_{s_{i,j} \in \Theta_k} T^{s_{i,j}} \tag{1}$$

其中，$T^{s_{i,j}}$ 表示车辆通过道路 $s_{i,j}$ 所需时间。

4.2.2 路口等待总时间计算

路口等红灯总时间是各路段终点路口处等待时间的累加，多辆小车排队总时间也是各路段终点路口处排队时间累加，因为同样发生在路段终点处，多辆小车排队会对红灯等待时间产生影响。针对车辆 k 在第 i 个路口等待总时长 $t_k^{(i)}$，可以分解为因绿灯点亮次序造成等待的未满一周期时长、满整个周期的时长和因排队拥堵等待前车驶离时长，从而可以表示为：

$$
\begin{aligned}
&t_k^{(i)} = t_{k,\delta}^{(i)} + n_{k,period}^{(i)} \cdot t_{period}^{(i)} + +t_{k,crowd}^{(i)} \\
&t_{k,\delta}^{(i)} = t_{left}^{o_{s_{j''',i}}} \cdot \left(\Omega_{s_{i,j}}\right) + \sum_{n_{j'''}} t_{duration}^{o_{s_{j''',i}}},\ j''' \in node\{\rightarrow i\},\ j''' \neq j, j' \\
&t_{k,crowd}^{(i)} = n_{k,f}^{s_{j,i}} \cdot \Delta t \\
&n_{j'''} = \begin{cases} o_{s_{j,i}} - o_{s_{j',i}} - 1 & ,if\ o_{s_{j,i}} > o_{s_{j',i}} \\ N_{node\{\rightarrow i\}} + o_{s_{j,i}} - o_{s_{j',i}} - 1, if\ o_{s_{j,i}} \leq o_{s_{j',i}} \end{cases}
\end{aligned}
\tag{2}
$$

而在具体仿真中，我们需要对每个时刻的路口状况，车辆状态进行动态判断，具体细节可以用下面两段伪代码进行解释：

当 A 来到路口时，是否会出现等待时间的判决过程以及等待时间时长的计算过程：

(a).先分析一条路上车 A 的情况：

Algorithm 1 单路多车等待时间分析策略

Statement: 输出：等待时间 t_jia

```
while 当车辆 A 到路口时 do
    首先分析一条路上车 A 的情况：
    if 当路上只有一辆车 A 时 then
        t_jia =t_jia +0；
    else if 当路上只有两辆车 A,B 时 then
        则可能出现排队情况：
        if 当改路的绿灯持续时间为 1 秒，且 A 为 ID 序号较小的车时 then
            t_jia =t_jia +0；
        else if 当该路的绿灯持续时间为大于等于 2 秒时 then
            t_jia =t_jia +0；
        else if 当该路的绿灯持续时间为 1 秒，且 A 为 ID 序号较大的车时 then
            t_jia =t_jia +t_lu；
        end if
    else if 当路上只有三辆车 A,B,C 时 then
        则可能出现排队情况：
        if 当该路的绿灯持续时间为 1 秒，且 A 为 ID 序号最小的车时 then
            t_jia =t_jia +0；
        else if 当该路的绿灯持续时间为 1 秒，且 A 为 ID 序号中间的车时 then
            t_jia =t_jia +t_lu；
        else if 当该路的绿灯持续时间为 1 秒，且 A 为 ID 序号最大的车时 then
            t_jia =t_jia +t_lu+t_lu；
        else if 当该路的绿灯持续时间为 2 秒，且 A 不为 ID 序号最大的车时 then
            t_jia =t_jia +0；
        else if 当该路的绿灯持续时间为 2 秒，且 A 为 ID 序号最大的车时 then
            t_jia =t_jia +t_lu；
        else if 当该路的绿灯持续时间大于等于 3 秒时 then
            t_jia =t_jia +0；
        end if
    end if
end while
```

(a.a)当路上只有一辆车 A 时，则不会出现排队情况；

(a.b)当路上只有两辆车 A,B 时，则可能出现排队情况：

(a.b.a)当该路的绿灯持续时间为 1 秒时，且 A 为 ID 序号较小的车

时，则不会出现排队情况；

(a.b.b)当该路的绿灯持续时间为大于等于 2 秒时，则不会出现排队情况；

(a.b.c)当该路的绿灯持续时间为 1 秒时，且 A 为 ID 序号较大的车时，则会出现排队情况，且等待时间为一个路口周期；

(a.c)当路上只有三辆车 A,B,C 时，则可能出现排队情况：

(a.c.a)当该路的绿灯持续时间为 1 秒时，且 A 为 ID 序号最小的车时，则不会出现排队情况；

(a.c.b)当该路的绿灯持续时间为 1 秒时，且 A 为 ID 序号中间的车时，则会出现排队情况，且等待时间为一个路口周期；

(a.c.c)当该路的绿灯持续时间为 1 秒时，且 A 为 ID 序号最大的车时，则会出现排队情况，且等待时间为两个路口周期；

(a.c.d)当该路的绿灯持续时间为 2 秒时，且 A 不为 ID 序号最大的车时，则不会出现排队情况；

(a.c.e)当该路的绿灯持续时间为 2 秒时，且 A 为 ID 序号最大的车时，则会出现排队情况，且等待时间为一个路口周期；

(a.c.f)当该路的绿灯持续时间大于等于 3 秒时，则不会出现排队情况；

(b).再分析路口有多条路时 A 的情况：

Algorithm 2 单路口多条路等待时间分析策略

Statement: 输出：等待时间 t_jia

```
while 当车辆 A 到路口时 do
    再分析路口有多条路时 A 的情况:
    先计算该路口所有路的绿灯亮起总持续时间（即周期）t_lu;
    再提取 A 所在路的绿灯在该路口多条路绿灯的亮起顺序 lu_cixv;
    再计算 A 所在路的绿灯亮起持续时间 t_lu_dangqian;
    if lu_cixv==1 then
        t_jia =t_jia +0;
    else
        再计算当前路口从一条路开始到 A 所在路的绿灯亮起持续时间（包括 A 所在路）t_lu_ban;
        再计算当前路口从一条路开始到 A 所在路的绿灯亮起持续时间（不包括 A 所在路）
    t_lu_banjian;
        用当前全局时间 t_quan 进行判决:
        if mod(t_quan,t_lu)< t_lu_banjian then
            t_jia =t_jia +t_lu_banjian-mod(t_quan,t_lu);
        else if mod(t_quan,t_lu) > t_lu_ban then
            t_jia =t_jia +t_lu_banjian+t_lu-mod(t_quan,t_lu);
        else
            t_jia =t_jia+0;
        end if
    end if
end while
```

(b.a)先计算该路口所有路的绿灯亮起总持续时间（即周期）t_lu;

(b.b)再提取 A 所在路的绿灯在该路口多条路绿灯的亮起顺序 lu_cixv;

(b.c)再计算 A 所在路的绿灯亮起持续时间 t_lu_dangqian；

(b.d)对 lu_cixv 进行判决，如果 lu_cixv 为 1，则 A 不会出现等待时间；如果 lu_cixv 为其他，则 A 会出现等待时间；

.....

以下分析 lu_cixv 不等于 1 时的情况：

(b.e)再计算当前路口从一条路开始到 A 所在路的绿灯亮起持续时间（包括 A 所在路）t_lu_ban；

(b.f)再计算当前路口从一条路开始到 A 所在路的绿灯亮起持续时间（不包括 A 所在路）t_lu_banjian；

(b.g)用当前全局时间 t_quan 进行判决

(b.g.a)当 mod(t_quan,t_lu)< t_lu_banjian（即当前时间还没到 A 所在道路的绿灯亮起的时间），则 A 会出现等待时间

t_wait=t_lu_banjian-mod(t_quan,t_lu)（即只需要等到 A 所在路的绿灯亮起，A 就可出发）；

b.g.b 当 mod(t_quan,t_lu) > t_lu_ban(即当前时间超过了 A 所在路的绿灯亮起的时间，但还没过一个完整的周期时间)，则 A 会出现等待时间 t_wait=t_lu_banjian+t_lu-mod(t_quan,t_lu)（即需要等待该轮周期时间结束，并需要等到新周期时 A 所在路的绿灯亮起，A 就可出发）；

综合 1 和 2 的等待时间之和，即为 A 在该路口的等待时间总时长。

4.2.3 全局车辆状态记录

设定全局车辆状态表，在代码仿真中记录全局车辆状态、用于判断“是否排队”、“是否等红灯”等过程，便于上述各模型的代码实现。其记录当前时刻全局车辆状态的能力也为第二问中确定交通故障发生后受影响车辆及车辆调控问题提供支撑。

按照本题中建立各项模型给定车辆状态表更新策略，用于代码中全局工作情况的仿真。如下述伪代码所示：

Algorithm 3 车辆状态变化判决策略

```
if t==0 && cars_state_shu>0 && street_name =”wait” then
    车辆 A 进入排队等待分析；
    if t==0 then
        更新 street_name 为下一条路的路名；
        更新 t 为下一条路的通行时间；
        cars_state_shu-1;
    else if t!=0 then
        更新 street_name 为 wait;
        更新 t 为 t_jia；
    end if
else if street_name==wait $$t==0 then
    更新 street_name 为下一条路的路名；
    更新 t 为下一条路的通行时间；
    cars_state_shu-1;
else if t==0 $$ cars_state_shu==0 then
    cars_state_shu=-1;
    输出 t_quan 为车辆 A 到达终点的时间；
end if
```

下面用例子进一步说明车辆状态表及车辆状态更新策略，对上述伪代码作进一步解释：

(1) 全局车辆状态表：

车辆 A 当前处于的路名	车辆 A 距离切换状态的秒数	车辆 A 剩下要走的路数
cji-ebdi	5	6
wait	4	5
...	...	...
bcf-hcbi	0	-1

示例的全局车辆状态表第一行：车辆 A 当前在路 cji-ebdi 上行驶，还需 5 秒到达路口 ebdi，车辆 A 距离到终点还需要走 6 条路；

示例的全局车辆状态表第二行：车辆 A 当前在路口 ebdi 处于等待（wait）状态，还需要 4 秒结束等待状态进入下一条路，车辆 A 距离到终点还需要走 5 条路；

示例的全局车辆状态表第三行：车辆 A 已经到达了终点 hcbi，还需要 0 秒切换其他状态（即不再切换状态），车辆 A 剩下要走的路数设置为-1（即车辆 A 行程已经结束，立即被移出，不再考虑）。

(2) 全局车辆状态表的更新策略：

(2.1)当车辆 A 距离切换状态的秒数为 0，且车辆 A 剩下要走的路数大于 0，且车辆 A 当前处于的路名不为 wait 时（t==0 && cars_state_shu>0 &&

street_name~="wait"），则进入车辆 A 的排队等待分析中：

(2.1.1)当车辆 A 的等待时间为 0 时，则车辆 A 当前处于的路名变为下一条路的路名，车辆 A 距离切换状态的秒数变为下一条路的通行时间，车辆 A 剩下要走的路数减 1；

(2.1.2)当车辆 A 的等待时间不为 0 时，则车辆 A 当前处于的路名变为 wait，车辆 A 距离切换状态的秒数变为等待时间，车辆 A 剩下要走的路数不变；

(2.2)当车辆 A 当前处于的路名为 wait，且车辆 A 距离切换状态的秒数为 0 时（cars_state(i,1)=="wait" && cars_state(i,2)=="0"），则车辆 A 当前处于的路名变为下一条路的路名，车辆 A 距离切换状态的秒数变为下一条路的通行时间，车辆 A 剩下要走的路数减 1。

(2.3)当车辆 A 距离切换状态的秒数为 0，且车辆 A 剩下要走的路数为 0 时，则车辆 A 剩下要走的路数切换为-1（即车辆 A 行程已经结束，立即被移出，不再考虑），同时输出当前的全局时间作为车辆 A 到达终点的时间。

4.3 全局调控策略优化模型

结合题意要求，我们得到路口信息：从a到$hjjj$，总路口数$N_I=8000$，其中字母$a\sim j$分别表示$0\sim 9$。总时长$D=858$，道路总数$N_S=63968$，车辆总数$N_V=200$，车辆按时到达终点的基本得分$F=250$。

同时，对各路段终点路口处的状态可以建立约束条件：

每辆车的行程经过任一个路口至多一次，即

$$\begin{cases} n_k^{(i)}=1, & \forall i,j\in \Xi_k, s_{i,j}\in \Theta_k \\ n_k^{(i)}=0, & \forall i,j\in I-\Xi_k, s_{i,j}\in S-\Theta_k \end{cases} \tag{3}$$

ID 序号较小的车辆排在前面先行，即

$$\sigma_k^{s_{j,i}}<\sigma_{k'}^{s_{j,i}}, k<k' \text{时}, \forall k,k'\in \Psi_{t,s_{j,i}}^{(i)} \tag{4}$$

在任何时刻，每个路口最多只有一个交通灯是绿灯，即

$$\begin{cases} \sum_j \Omega_{s_{j,i}}=1, \forall t\bigcap \forall j\in node\{\to i\} \\ \Omega_{s_{j,i}}\in\{0,1\} \end{cases} \tag{5}$$

综合上述过程，在给定红绿灯调控策略下，每辆小车按照既定路线从起点到终点的行驶时间为：

$$\begin{cases} T_k=\sum_{s_{i,j}\in\Theta_k} T^{s_{i,j}}+\sum_{i\in\Xi_k} t_k^{(i)}+(P_k-1)\cdot\Delta t \\ \Delta t=1\ \mathrm{s} \end{cases} \tag{6}$$

按照题目规则，特定红绿灯调控策略下，以所有小车得分量化调控策略的优劣，即要求每辆小车尽可能快的到达终点，且不能超出规定的最大行驶时间，依据此规则计算每辆小车的得分规则如下：

$$M_k=\begin{cases}F+\left(D-T_k\right),T_k\le D\\ 0\qquad,T_k>D\end{cases}\tag{7}$$

综上所述，可以得到总的路径优化模型目标函数及其约束条件，从而建立数学模型为：

$$\max\sum_{k=1}^{N_V}M_k$$

$$M_k=\begin{cases}F+\left(D-T_k\right),T_k\le D\\ 0\qquad,T_k>D\end{cases}①$$

$$T_k=t_1+\sum_{i\in\Xi_k}t_k^{(i)}+\left(P_k-1\right)\cdot\Delta t②$$

$$t_1=\sum_{s_{i,j}\in\Theta_k}T^{s_{i,j}}③$$

$$t_k^{(i)}=t_{k,\delta}^{(i)}+n_{k,period}^{(i)}\cdot t_{period}^{(i)}++t_{k,crowd}^{(i)}④$$

$$t_{k,\delta}^{(i)}=t_{left}^{o_{s_{j''',i}}}\cdot\left(\Omega_{s_{i,j}}\right)+\sum_{n_{j'''}}t_{duration}^{o_{s_{j''',i}}},j'''\in node\left\{\to i\right\},j'''\neq j,j'⑤$$

$$t_{k,crowd}^{(i)}=n_{k,f}^{s_{j,i}}\cdot\Delta t⑥$$

$$\Delta t=1\ \mathrm{s}⑦$$

$$n_{j'''}=\begin{cases}o_{s_{j,i}}-o_{s_{j',i}}-1\qquad,if\ o_{s_{j,i}}>o_{s_{j',i}}\\ N_{node\{\to i\}}+o_{s_{j,i}}-o_{s_{j',i}}-1,if\ o_{s_{j,i}}\le o_{s_{j',i}}\end{cases}⑧$$

$$\sigma_k^{s_{j,i}}<\sigma_{k'}^{s_{j,i}},k<k'\text{时},\forall k,k'\in\Psi_{t,s_{j,i}}^{(i)}⑨$$

$$\begin{cases}\sum_j\Omega_{s_{j,i}}=1,\forall t\bigcap\forall j\in node\left\{\to i\right\}\\ \qquad\Omega_{s_{j,i}}\in\left\{0,1\right\}\end{cases}⑩\tag{8}$$

$$\begin{cases}n_k^{(i)}=1,\quad\forall i,j\in\Xi_k,s_{i,j}\in\Theta_k\\ n_k^{(i)}=0,\quad\forall i,j\in I-\Xi_k,s_{i,j}\in S-\Theta_k\end{cases}⑪$$

其中式①表示车辆k的得分，式②表示车辆k到达终点的总时间，式③表示车辆k路段内通行总时间，式④表示车辆k在第i个路口等待总时长，式⑤表示车辆k在第i个路口因绿灯点亮次序造成等待的未满一周期时长，式⑥表示车辆k因排队拥堵等待前车驶离时长，式⑦表示车辆通过路口需时 1 s，式⑧表示在一个周期内，车辆仍需等待的前面的绿灯的数量，式⑨表示 ID 序号较小的车辆排在前面先行，式⑩表示在任何时刻，每个路口最多只有一个交通灯是绿灯，式⑪表示每辆车的行程经过任一个路口至多一次。

4.4 算法求解

4.4.1 算法描述

本文研究的交通信号灯全局优化调控问题是典型的多目标组合优化问题，属于 NP-hard 问题，可以采用智能搜索算法求解该类问题。目前存在多种只能搜索算法，如遗传算法(GA)、粒子群算法(PSO)、蚁群算法(ACO)和化学反应优化算法(CRO)等。相比于遗传算法，化学反应优化算法能避免过早地陷入局部最优，

获取全局近似最优解，同时以更快的速度收敛于最优解。化学反应优化算法最早是在 2010 年由 Lam 和 Li 提出来的，灵感来源于大自然中的化学反应。根据实际应用表明：化学反应算法能够有效地解决多目标优化问题，并且在二次分配问题，网格任务调控问题、人工神经网络训练、资源受限项目调控问题等方面得到广泛的应用。针对交通信号灯全局优化调控问题，本文采用化学反应优化算法求解。

4.4.2 算法框架

化学反应优化算法中，分子具有 3 个必要属性：分子结构（structure，简称 S）、分子势能（potential energy；简称 PE）和分子动能（kinetic energy，简称 KE）。分子结构 S 表示交通信号灯调控策略的一个解空间；分子势能 PE 表示分子结构的稳定性，定义为目标函数的值；分子动能 KE 使得分子势能趋向更高的状态，避免目标函数值过早地陷入局部最优解而继续寻找全局最优解。第 2.1 节~第 2.3 节主要从问题编码、适应度函数和操作算子设计这 3 个角度介绍化学反应优化算法。问题编码定义了调控策略与分子结构之间的关系；适应度函数是评估分子好坏的标准，分子的适应度值越小，代表所求问题的解越优。因此，设计一种好的适应度函数有利于获得最优解；操作算子是算法搜索解空间的关键步骤，不同的操作算子能够改变分子结构，避免算法过早陷入局部最有解，同时搜索全局最优解。第 3 节介绍具体的算法伪代码设计，第 4 节对算法的复杂度和收敛性以及数据量增大后的鲁棒性进行分析。

(1) 问题编码

化学反应优化算法是一种通过模拟化学反应过程搜索最优解的方法，问题解空间用团分子集合表示，分子是由若干个原子编码组成。本文研究的交通信号灯调控策略是由若干个需要调控的路口信号灯调控策略组成的，每个路口交通信号灯调控策略由该路口各信号灯亮绿灯的顺序和绿灯持续时间组成。

(1.1) 单个信号灯编码

对应到编码中，需要将上述内容设计为两组独立的数值来表示。单个路口调控策略包含每个信号灯亮绿灯的顺序和绿灯，绿灯持续时间容易编码，但如何表示每个信号灯亮绿灯的顺序并且还要方便后续算法操作是个难题。为此，我们创新性的提出用区间内数字大小表示每个信号灯亮绿灯的顺序。具体操作为：

为每个信号灯生成一个代表其亮绿灯顺序的限定区间内整数，并按照从小到大排序，作为其亮起绿灯的顺序，对表示亮起绿灯顺序的数相同的信号灯，按照从左至右顺序再次进行排序。

这种创新性的操作使得在本题中化学反应算法的各项分子操作得以实现。下面我们举例对其做进一步说明：

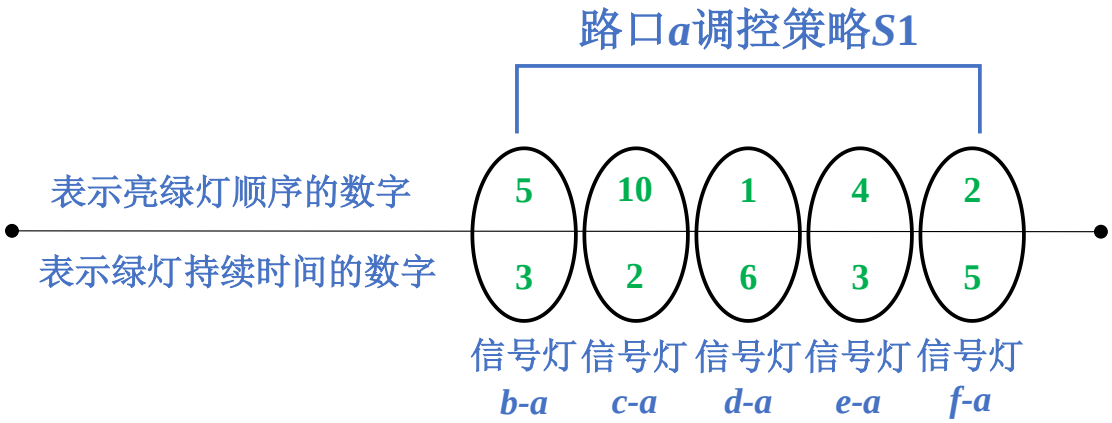

图 3 调控策略示意图 1

在上图所示情况下，表示信号灯亮绿灯数字大小不一，按照从小到大顺序，亮起绿灯顺序为 d-a→f-a→e-a→b-a→c-a，该调控策略 *S*1 具体表示为，一个周期内：

其他信号灯亮红灯，信号灯 d-a 亮 6 秒绿灯；然后其他信号灯亮红灯，信号灯 f-a 亮 5 秒绿灯；然后其他信号灯亮红灯，信号灯 e-a 亮 3 秒绿灯；然后其他信号灯亮红灯，信号灯 b-a 亮 3 秒绿灯；然后其他信号灯亮红灯，信号灯 c-a 亮 2 秒绿灯；

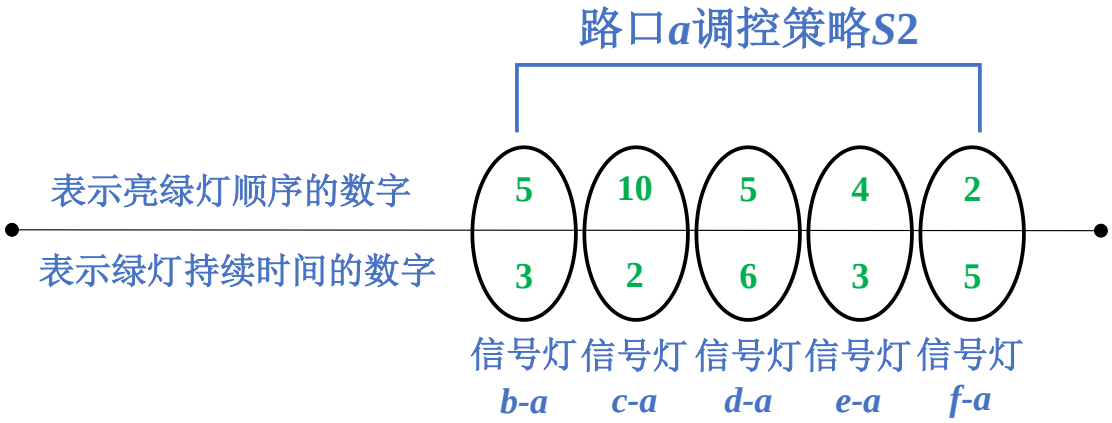

图 4 挑控策略示意图 2

在上图所示情况下，信号灯 b-a 和信号灯 d-a 表示亮绿灯顺序数字相同，因此对这两个信号灯亮绿灯顺序按照从左到右排序为 b-a→d-a，在此基础上再对所有信号灯按照从小到大排序，亮起绿灯顺序为 f-a→e-a→b-a→d-a→c-a，该调控策略 *S*2 具体表示为，一个周期内：

其他信号灯亮红灯，信号灯 f-a 亮 6 秒绿灯；然后其他信号灯亮红灯，信号灯 e-a 亮 3 秒绿灯；然后其他信号灯亮红灯，信号灯 b-a 亮 3 秒绿灯；然后其他信号灯亮红灯，信号灯 d-a 亮 6 秒绿灯；然后其他信号灯亮红灯，信号灯 c-a 亮 2 秒绿灯；

(1.2) 全局调控策略编码

对应到化学反应算法中来，分子团表示全局交通信号灯调控策略，单个分子表示单个路口交通信号灯调控策略，单个分子由若干原子组成，每个原子表示该路口需要进行调控的交通信号灯亮绿灯的顺序及持续时间，由两个数组成，如图 5 所示：

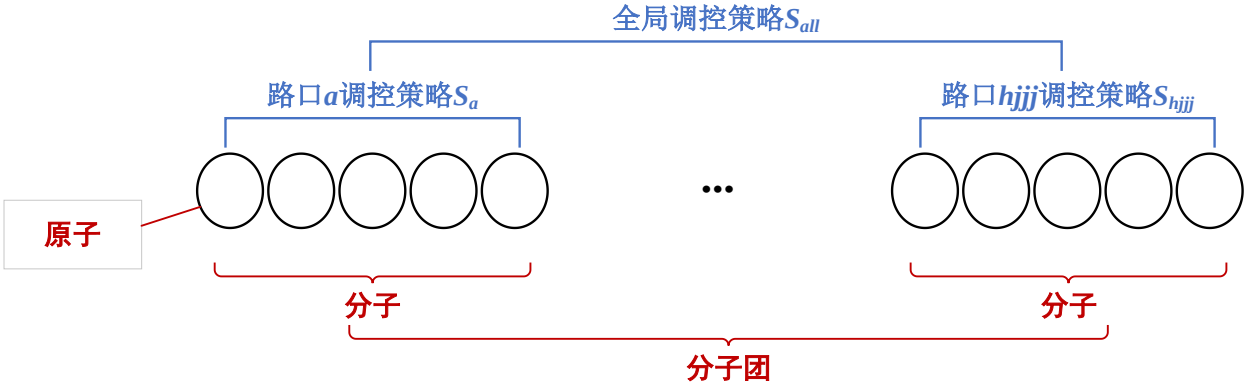

图 5 全局调控策略编码示意图

图中全局调控策略用 S_{all} 表示，包含所有路口的调控策略；单个路口调控策略用 S_a 到 S_{hjjj} 表示，包含该路口所有信号灯的调控；单个信号灯编码后包含两个数值，分别是表示亮绿灯顺序的数字和绿灯持续时间。对应到化学反应算法中来则分别对应分子团、分子、原子三类。

(2) 适应度函数

本文评价交通信号灯调控策略性能的指标是计算每辆小车得分并求和，每辆小车到规定时间内到达终点越早得分越高，超出规定时间到达终点不得分，见式(8)。

适应度函数跟优化目标有关，是评估分子好坏的标准，分子团的适应度值越小，代表所求问题的解越优。因为题中优化目标是希望小车尽快到达终点，利用小车行驶时间计算得到的得分尽可能大，因此可以将分子团的适应度用下式表示：

$$Fitness(S) = -\frac{T - T_{\min}}{T_{\max} - T_{\min}} \tag{9}$$

其中，$T_{\max}$ 和 $T_{\min}$ 分别代表种群中所有分子团对应的利用小车完成时间计算总得分的最大值和最小值。

(3) 操作算子设计

本节介绍化学反应优化算法的 4 种基本操作算子，包括单分子碰撞、单分子分解、分子间碰撞和分子合成。问题编码中介绍过每个分子结构表示一种交通信号灯调控策略，不同的操作算子对分子结构会产生变化，交通信号灯的调控策略也会随之改变。单分子碰撞和分子间碰撞对原分子结构改变较小，旨在原调控策略的邻域空间搜索更优的调控策略；单分子分解和分子合成提供搜索更大解空间的机会，避免调控策略过早地陷入局部最优解。下面详细介绍 4 种基本操作算子的具体设计。

(3.1) 单分子碰撞

单分子碰撞是指单个分子与容器壁发生碰撞并反弹的过程。对于分子 S，发生单分子碰撞后产生新的分子 S'，新分子 S' 与原分子 S 在分子结构上相差不大，表示碰撞过程是在分子 S 的邻域空间搜索最优解。

具体设计图 6 所示：从原分子 S 中随机选择一个原子，改变其内容，剩余部分保留不变。对应到调控策略中，原先的调控策略经过单分子碰撞后，表示该信号灯亮绿灯顺序的数和该信号灯绿灯持续时间变化，产生新的策略。

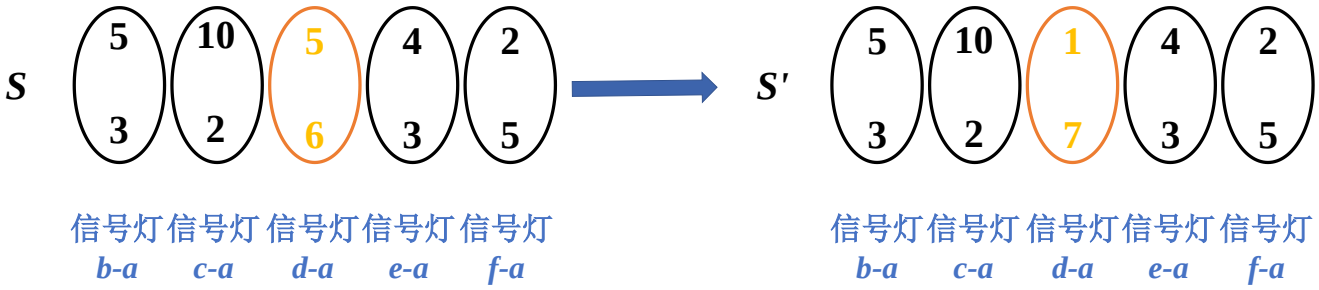

图 6 单分子碰撞示意图

(3.2) 单分子分解

单分子分解是指单个分子与容器壁发生碰撞后分解为两个或多个分子的过程。本文假设分解后产生两个分子。对于分子 S，经过单分子分解后产生新的分子 $S1$ 和 $S2$，新分子 $S1$ 和 $S2$ 与原分子 S 在分子结构上差别较大，表示分解过程是探索更大的解空间，避免结果过早地收敛于局部最优解。

具体设计如图 7 所示：在原分子中随机选择一个分解点 *point*，新分子 $S1$ 保留原分子 S 的 $[start, point]$ 部分的原子，新分子 $S2$ 保留原分子 S 的 $[point, end]$ 部分的原子。$S1$ 和 $S2$ 剩余的原子位随机生成。对应到调控策略中，原调控策略经过单分子分解，随机选择一个交通信号灯作为分解点，将分解点之前和之后部分交通信号灯表示亮绿灯顺序的数值及绿灯持续时间分别保留到两个新分子的对应位置，剩余部分交通信号灯在区间内随机产生表示亮绿灯顺序的数值及绿灯持续时间，从而得到新的调控策略。

图 7 单分子分解示意图

(3.3) 分子间碰撞

分子间碰撞是指两个分子相互碰撞并反弹的过程，对于分子 $S1$ 和 $S2$，发生分子间碰撞后产生新分子 $S1'$ 和 $S2'$，新分子 $S1'$ 和 $S1'$ 与原分子在分子结构上比较相似，表示碰撞过程是在分子 $S1$ 和 $S2$ 的邻域空间搜索最优解。

具体设计如图 8 所示：从原分子中选择两个碰撞点 *point*1 和 *point2*，交换原分子 $S1$ 和 $S2$ 对应碰撞点的原子的值，形成新的分子。对应到调控策略中，原调控策略经过分子间碰撞，随机交换两个调控策略中对应两个交通信号灯的数值，得到新的策略。

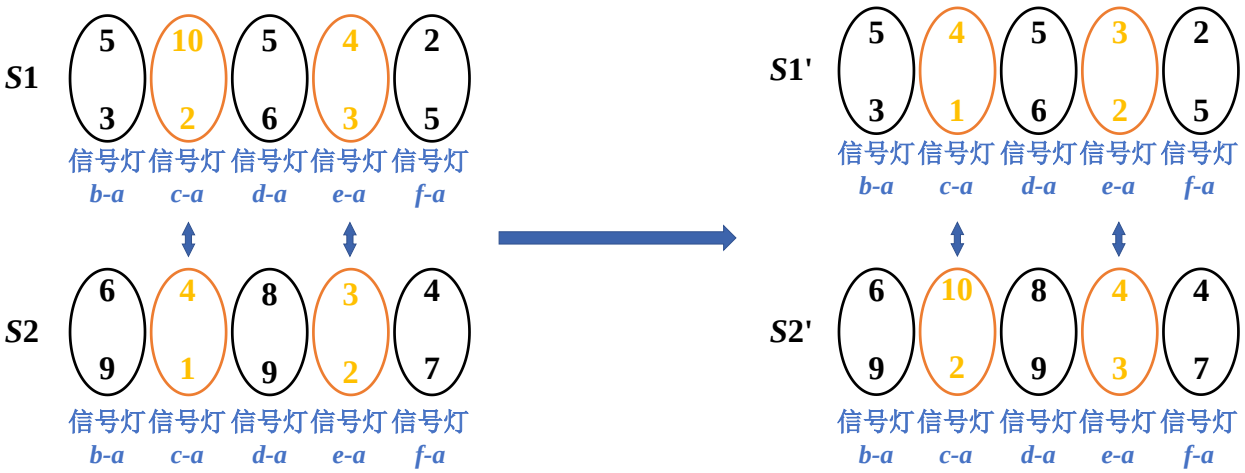

图 8 分子间碰撞示意图

(3.4) 分子合成

分子合成是指两个或多个分子相互碰撞并合成一个分子的过程。本文假设是两个分子进行分子合成操作。对于分子 $S1$ 和 $S2$，经过分子合成后产生新的分子 S，新分子 S 和原分子在分子结构上相差较大，表示合成使得分子种群呈现多样化，扩大解的搜索空间。具体设计如图 9 所示：随机生成一个合成点 $point$，新分子继承原分子 $S1$ 的 $[start, point]$ 部分的原子和原分子 $S2$ 的 $[point, end]$ 部分的原子。对应到调控策略中，原调控策略经过分子合成后，随机选择一个交通信号灯作为合成点将第 1 种调控策略的合成点的前部分信号灯数值和第 2 种调控策略对应合成点的后部分信号灯数值保留，合成后得到新的卸载策略。

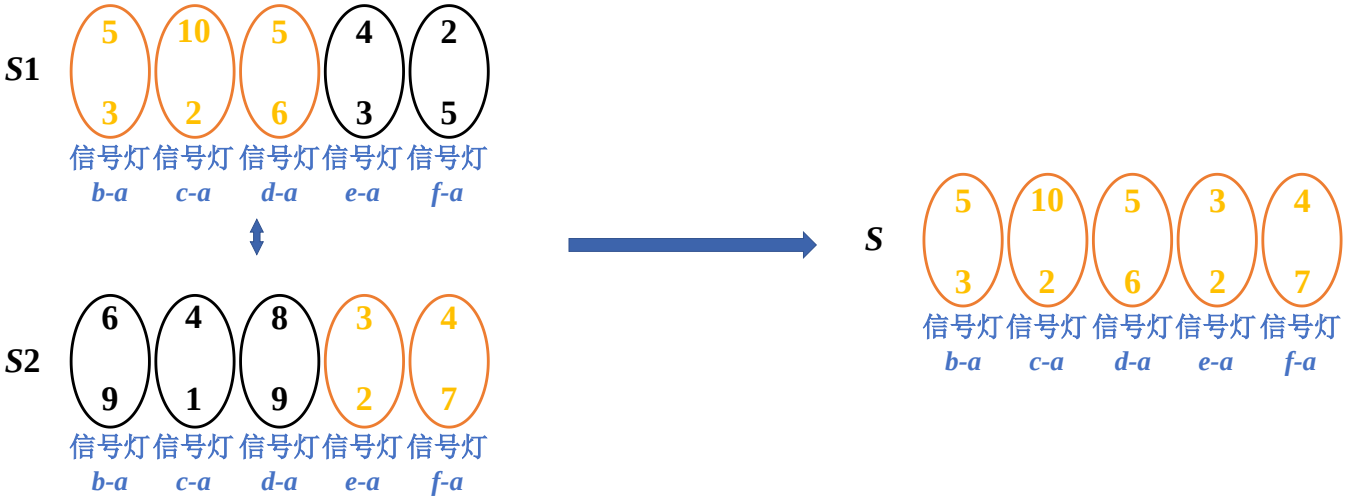

图 9 分子合成示意图

4.4.3 伪代码设计

Algorithm 4 化学反应优化策略

Input: 路口名、路口的路灯数

Output: 路口各路灯的绿灯调度策略

```
初始化 Maxiter,Molecule,KElossrate,InitialKE,Strategies:
Rand(S) //随机生成一个调度策略 S
PE[S]=Fitness(S) //求调度策略 S 的适应度
Strategies← S //将调度策略 S 添加到策略集 Strategies 中
j=1
while j<Maxiter do
    t=random(0,1)
    S← Strategies
    if checkDecomp(S) then
        status← decompose(S,s1,s2) //单分子分解
    else
        status← ineff_coll_on_wall(S,s) //单分子碰撞
        Strategies→ S'
    end if
end while
```

解空间分为两个部分：路口各路灯的绿灯亮起顺序、路口各路灯的绿灯亮起持续时间。

利用上述编码策略我们针对路口各路灯的绿灯亮起顺序和绿灯持续时间采取化学反应优化算法进行优化。其中化学反应分子中的各个原子包含表示绿灯亮起顺序数字和绿灯持续时间两部分内容。

首先初始化分子：

$$S = rowrank\left(\left[1,2,...,ludeng\_shu\right]\right) \tag{10}$$

$$Stratedies \leftarrow S \tag{11}$$

当迭代次数 $j < Maxiter$ 时，开始迭代：

当满足单分子碰撞条件时，则对调控策略进行单分子碰撞，先从策略集 *Stratedies* 中随机选择一个分子 S：

$$S \leftarrow Stratedies \tag{12}$$

然后生成两个随机数 s_i 和 s_j，交换分子 S 中位置为 $S\left(1,s_i\right)$ 和 $S\left(1,s_j\right)$ 的两原子的位置,从而产生一个新的分子 S'，然后将 S' 添加到策略集 *Stratedies* 中，替换旧策略 S。

然后将策略集 *Stratedies* 代入适应度函数公式中求得适应度，如果更优则将策略集 *Stratedies* 更新为最优策略集。通过使用单分子碰撞从而实现较优解的局部寻优，从而加快收敛速度得到局部最优解。

当满足单分子分解条件时，则对调控策略进行单分子分解，先从策略集 *Stratedies* 中随机选择一个分子 S：

$$S \leftarrow Stratedies \tag{13}$$

然后生成一个随机数 s_i，保留分子 S 中随机数 s_i 的前半部分，然后随机交换

S 中后半部分的顺序，从而产生一个新的分子 S'，然后将 S' 添加到策略集 *Stratedies* 中，替换旧策略 S，生成一个新的策略集 *Stratedies*′;保留分子 S 中随机数 s_i 的后半部分，然后随机交换 S 中前半部分的顺序，从而产生一个新的分子 S''，然后将 S'' 添加到策略集 *Stratedies* 中，替换旧策略 S，生成一个新的策略集 *Stratedies*″。

然后将策略集 *Stratedies*′ 和 *Stratedies*″ 代入适应度函数公式中求得适应度，如果更优则将策略集 *Stratedies* 更新为最优策略集。通过使用单分子分解从而实现了分子的全局寻优，避免过早陷入局部最优解。

4.5 算法复杂度、收敛性及鲁棒性分析

4.5.1 算法复杂度分析

设需要设计交通信号灯调控策略的路口个数为 n，单个路口调控策略中涉及到交通信号灯数量为 $N_i, i \in \{1,2,...,n\}$。分子种群的规模为 *PopSize*，最大迭代次数为 *MaxIter*。算法中涉及到的过程主要是采用 CRO 算法做出具体的交通信号灯调控决策。其中，初始化分子种群的时间复杂度为 $O(PopSize \times n)$，单分子碰撞操作的时间复杂度为 $O(1)$，单分子分解操作的时间复杂度为 $O(n)$，分子间碰撞的时间复杂度为 $O(1)$，分子合成的时间复杂度为 $O(n)$，因此，CRO 算法的时间复杂度为 $O\left(PopSize \times n + MaxIter \times n^2\right)$。

4.5.2 算法的收敛性分析

算法的收敛性主要受化学反应算法的基本操作影响。化学反应算法可以建立为一种有限吸收马尔可夫链模型，将每个分子结构表示成一种状态，用 X 表示所有状态空间的集合。对于一种状态，经过基本操作 Λ 后会改变其状态，令 $\Psi_A(x)$ 表示分 x 经过操作 Λ 后可能出现的状态的集合，且 $\Psi_A(x) \subseteq X$，二元组 $\left(X, \left(\Psi_A(x), x \in X\right)\right)$ 能够抽象成一个有向无环图 $G(V,E)$，顶点集 V 用 X 表示，边集 E 则根据 $\Psi_A(x) = \{i \mid (x,i) \in E\}$ 得到。如图 10 所示为经过操作 Λ 后状态的转换过程，可以得出，状态 x_1 经过操作 Λ 可以转换成 x_2, x_3 和 x_4，即 $\Psi_A(x_1) = \{x_2, x_3, x_4\}$，以此类推，$\Psi_A(x_2) = \{x_2, x_5, x_6\}$，$\Psi_A(x_3) = \{x_1, x_2, x_6\}$，$\Psi_A(x_4) = \{x_3, x_5\}$，$\Psi_A(x_5) = \{x_2, x_6\}$，$\Psi_A(x_6) = \{x_2\}$。

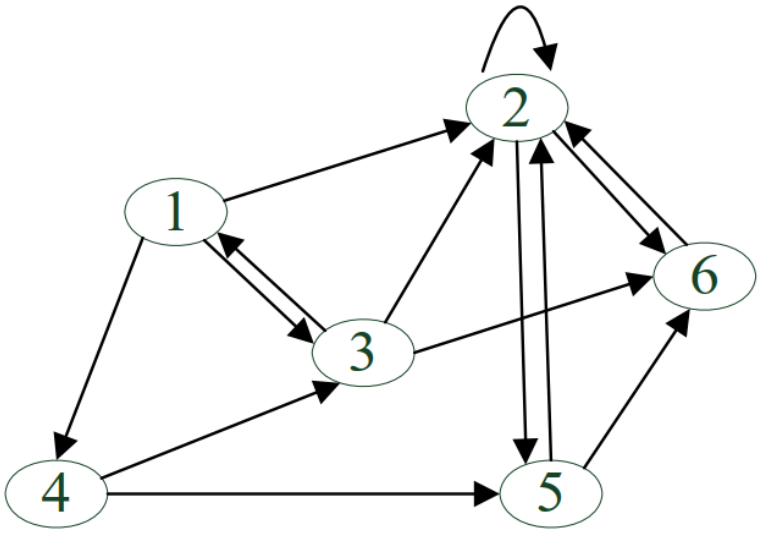

图 10 操作 Λ 下的状态变化

定义 1(最佳可达性)：设 $G(V,E)$ 为 CRO 算法的解图，若 $\forall x \notin X - X_{opt}$，则经过一定的状态转换，至少存在一个最优解 $x_{opt} \in X_{opt}$，使得 $x = x_0 \to x_1 \to x_2 \to ... \to x_I = x_{opt}$，其中，$I$ 为从 x 到 x_{opt} 状态转换的次数，则称 $G(V,E)$ 为最佳可达。

定理 1：CRO 收敛于全局最优解的必要条件是 $G(V,E)$ 最佳可达。

证明：CRO 中每种操作算子都存在一个解图，用 $G(V,E^{owi})$，$G(V,E^{dec})$，$G(V,E^{imi})$，$G(V,E^{syn})$ 分别表示单分子碰撞、单分子分解、分子间碰撞和分子合成的解图，$G(V,E)$ 表示 CRO 算法的解图，则满足 $E = E^{owi} \cup E^{dec} \cup E^{imi} \cup E^{syn}$。假设解图 $G(V,E)$ 不是最佳可达，其解图的任意非最优解都无法通过状态转化到达最优解，则存在解 $x' \in X - X_{opt}$。CRO 算法中初始种群产生的解都是非最优解 x' 的概率为 $1/\|X\|^n$，$\|X\|$ 表示 X 的基，n 为初始种群大小。于是，从 x' 转换到最优解的概率为

$$P\{x_{t+1} \in X_{opt} \mid x_t \notin X_{opt}\} = 0, t \geq 0 \tag{14}$$

则不存在一个从非最优解到最优解的转换过程，因此，产生最优解的概率计算如下:

$$\lim_{t\to\infty} P\left\{x \in X_{opt}\right\} \le 1 - \frac{1}{\|X\|^n} \le 1 \tag{15}$$

由此可得:当$G(V,E)$不满足最佳可达的条件时，CRO 算法不能收敛于全局最优解。证毕。

4.5.3 鲁棒性分析

(1) 算法对城市大小鲁棒性分析

不同城市大小对应路网规模大小不一，随着路网规模的增大，道路、车辆信息增大，需处理数据量增大、解空间维度增大，导致算法收敛速度下降甚至无法收敛。

查阅文献发现，据城市调查统计，直径增大 1 倍的同级城市，经济发展带来的影响导致城市路网规模复杂度增大至少 10 倍以上。由上述复杂度计算公式$O\left(PopSize \times n + MaxIter \times n^2\right)$可得，受分子种群规模 *PopSize* 及最大迭代次数 *MaxIter* 影响，随着城市规模不断增加，使用该算法做交通信号灯调控优化复杂度会提升的很快。

但这种复杂度快速增长是由问题本身因起的，大型城市的信号灯调控或者交通规划问题是一个超多目标复杂组合优化问题，属于 NP-hard 问题，没有算法能很快收敛到最优解。在智能算法领域，其算法能力较其他智能算法有着不错的性能表现。我们对该问题使用遗传算法和化学反应算法，固定迭代 100 次后，其性能曲线如图 11 所示。

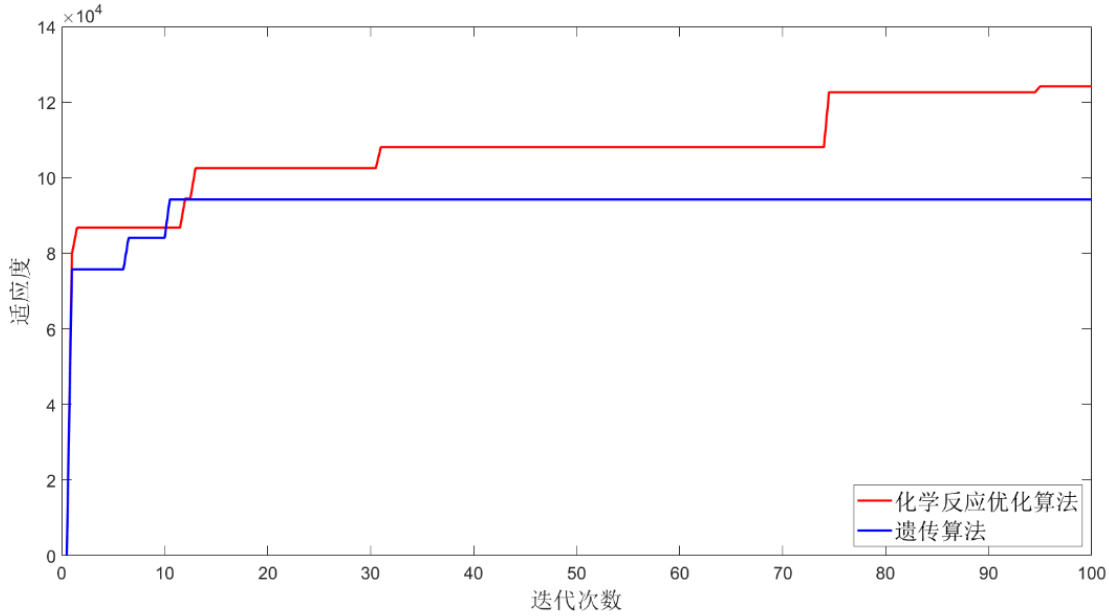

图 11 算法性能对比示意图

从图 11 中可以看到，固定迭代 100 次情况下，较遗传算法相比，化学反应算法能很好避免陷入局部最优并且收敛速度也更快。

(2) 大型复杂城市交通调控解决思路

通过查阅文献了解到，当下对于复杂城市的路网规划、信号灯调控问题，有两种解决思路，一种解决方式是增加资源投入并对问题进行分解，通过大型服务

器的强大算力提升算法性能，并依据城市多中心的特点，将大城市的路网优化、交通调控问题划分为几个小型城市问题进行求解；令一种解决方法是运用机器学习，将问题进一步细分，依据神经网络提取深度特征并进行预测的能力对每个交通路口独立进行调控，依据是当前各个路口的车流量及未来车流量的预测，并且在系统运行过程中不断收集数据、训练模型，做到智能适变。两种解决问题的思路如下图所示：

(2.1) 城市分解

大型复杂城市交通调控问题进行城市分解后可以有效简化并求解。图 12 所示地图是长沙市某一片区卫星航拍图，假设图中所示为某一大型城市，直接对其进行交通规划、调整调控等工作不仅问题复杂且难以实施，存在各种不稳定因素，但通过划分行政片区、或者简单依据山河地形，将其划分为多个小型子城市进行处理，不仅问题简化，而且在落实等方面更有效果，问题细分后也能进一步消除问题过于复杂带来的不确定性因素。

图 12 城市分解示意图

(2.2) 机器学习应用

结合遗传算法和长短时神经网络算法进行智能交通信号灯调控的流程图如图 13 所示，该算法中长短时神经网络用于结合统计信息预测之后时刻的车流量并做路网压力分析，路网压力大小表示每条路的拥堵程度。遗传算法则是在知道每条道路的车流量大小的情况下为交通调控策略进行寻优。通过这种智能寻优和机器预测的方式，可以有效的针对交通情况作出适变，有效进行调控，只是实现起来比较复杂，且在本题中没有数据预测需求，因此在本题中未被采用，但在复杂交通系统的调控中已经有了很多应用。

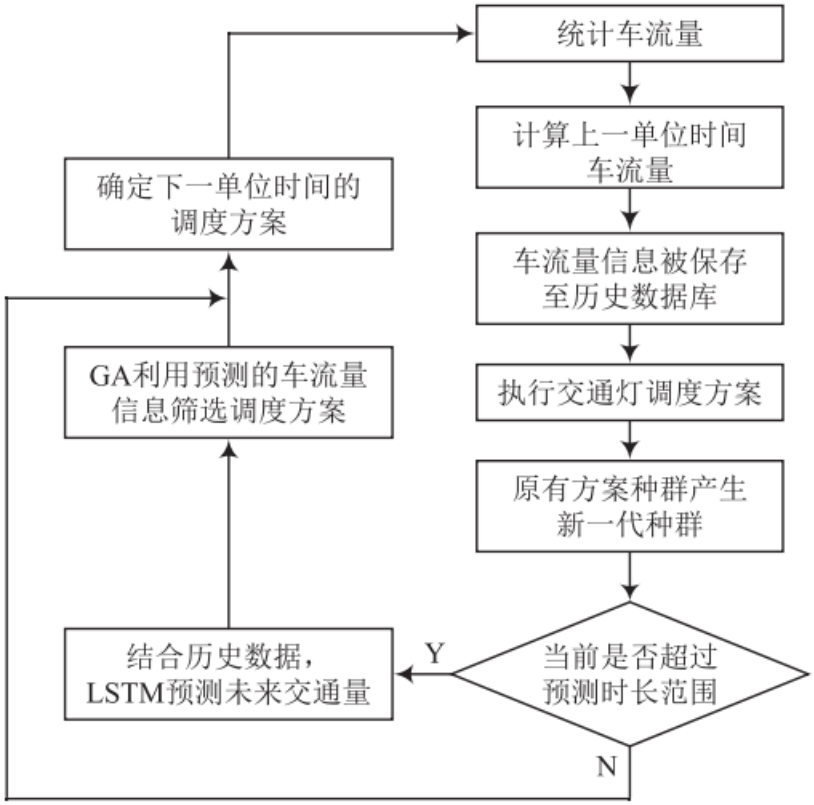

图 13 结合 GA 和 LSTM 的交通灯调度示意图

4.6 结果分析

4.6.1 数据预处理结果

本题涉及数据量较大，包含 200 辆小车行驶路线、8000 个路口和 63968 条路段。为了简化问题我们最开始进行了数据预处理，分析了哪些路段有小车经过及其各自客流量多少；分析了哪些路口有小车经过及该路口需要亮绿灯的路段数。结果如下所示：

经过预处理后，我们发现 200 辆车走过的路段数为 3569 条，其中不重复的路段数总共 3497 条。因此有 72 条路段有多辆车辆经过可能发生排队现象，其中 71 条路会有两辆小车经过，只有 1 条路会有三辆小车经过。

此外，200 辆车总共经过 2577 个路口，需要对这 2577 个路口进行交通信号灯调控设计。

可以看出，相比于最初的 8000 个路口，63968 条路段，预处理之后大大减小了数据纬度，简化了问题，避免了没必要的冗余，加快了解题的速度。

4.6.2 全局交通信号灯调控策略结果与分析

(1) 全局交通信号灯调控策略结果展示

因为本题结果数据量太大，我们按照题目要求将其存储到 result1. txt 中作为附件上传。

通过该全局交通信号灯调控策略，利用所有小车到达终点时间代入得分公式(8)计算总得分，该策略对应的总得分是 128522。

在该策略下具体每辆小车行驶时间及依据行驶时间计算各自得分如下表所示：

小车名称	完成时间	最终得分		小车名称	完成时间	最终得分
C001	141	967		C101	685	423
C002	379	729		C102	213	895
C003	591	517		C103	155	953
C004	410	698		C104	251	857
C005	389	719		C105	663	445
C006	146	962		C106	813	295
C007	299	809		C107	680	428
C008	114	994		C108	519	589
C009	706	402		C109	558	550
C010	425	683		C110	475	633
C011	559	549		C111	410	698
C012	454	654		C112	563	545
C013	295	813		C113	505	603
C014	105	1003		C114	439	669
C015	503	605		C115	372	736
C016	236	872		C116	328	780
C017	144	964		C117	602	506
C018	480	628		C118	85	1023
C019	247	861		C119	486	622
C020	181	927		C120	620	488
C021	189	919		C121	192	916
C022	210	898		C122	703	405
C023	494	614		C123	274	834
C024	495	613		C124	843	265
C025	484	624		C125	258	850
C026	415	693		C126	746	362
C027	754	354		C127	286	822
C028	162	946		C128	565	543
C029	148	960		C129	479	629
C030	342	766		C130	461	647
C031	538	570		C131	210	898
C032	565	543		C132	582	526
C033	574	534		C133	737	371
C034	271	837		C134	427	681
C035	465	643		C135	705	403
C036	714	394		C136	738	370
C037	620	488		C137	174	934
C038	625	483		C138	829	279
C039	134	974		C139	469	639
C040	84	1024		C140	301	807
C041	590	518		C141	664	444
C042	755	353		C142	511	597

C043	116	992		C143	761	347
C044	190	918		C144	189	919
C045	91	1017		C145	244	864
C046	442	666		C146	251	857
C047	339	769		C147	545	563
C048	198	910		C148	614	494
C049	760	348		C149	628	480
C050	736	372		C150	176	932
C051	743	365		C151	743	365
C052	780	328		C152	309	799
C053	825	283		C153	455	653
C054	660	448		C154	777	331
C055	279	829		C155	735	373
C056	332	776		C156	260	848
C057	295	813		C157	152	956
C058	346	762		C158	764	344
C059	648	460		C159	451	657
C060	396	712		C160	265	843
C061	732	376		C161	842	266
C062	565	543		C162	713	395
C063	360	748		C163	678	430
C064	167	941		C164	590	518
C065	547	561		C165	769	339
C066	834	274		C166	192	916
C067	579	529		C167	810	298
C068	343	765		C168	322	786
C069	278	830		C169	541	567
C070	567	541		C170	744	364
C071	615	493		C171	409	699
C072	197	911		C172	127	981
C073	613	495		C173	676	432
C074	845	263		C174	416	692
C075	798	310		C175	450	658
C076	195	913		C176	368	740
C077	253	855		C177	676	432
C078	374	734		C178	755	353
C079	102	1006		C179	215	893
C080	522	586		C180	374	734
C081	901	0		C181	665	443
C082	501	607		C182	340	768
C083	397	711		C183	807	301
C084	404	704		C184	659	449
C085	666	442		C185	485	623

C086	434	674		C186	263	845
C087	285	823		C187	332	776
C088	354	754		C188	293	815
C089	592	516		C189	554	554
C090	536	572		C190	347	761
C091	368	740		C191	211	897
C092	653	455		C192	734	374
C093	815	293		C193	296	812
C094	551	557		C194	450	658
C095	188	920		C195	443	665
C096	411	697		C196	522	586
C097	163	945		C197	478	630
C098	624	484		C198	441	667
C099	754	354		C199	702	406
C100	343	765		C200	253	855

(2) 化学反应算法性能分析

在前面鲁棒性分析处，我们对比了固定迭代 100 次化学反应算法和遗传算法的性能差距。验证了化学反应算法相比遗传算法能有效地过早避免陷入局部最优情况且收敛速度更快。

在求解过程中我们使用化学反应算法进行求解，限于时间限制和问题复杂度，我们固定迭代 500 后得到的优化结果作为最终优化结果，记录随着迭代次数增加，优化目标——总得分与迭代次数间的趋势图如下所示：

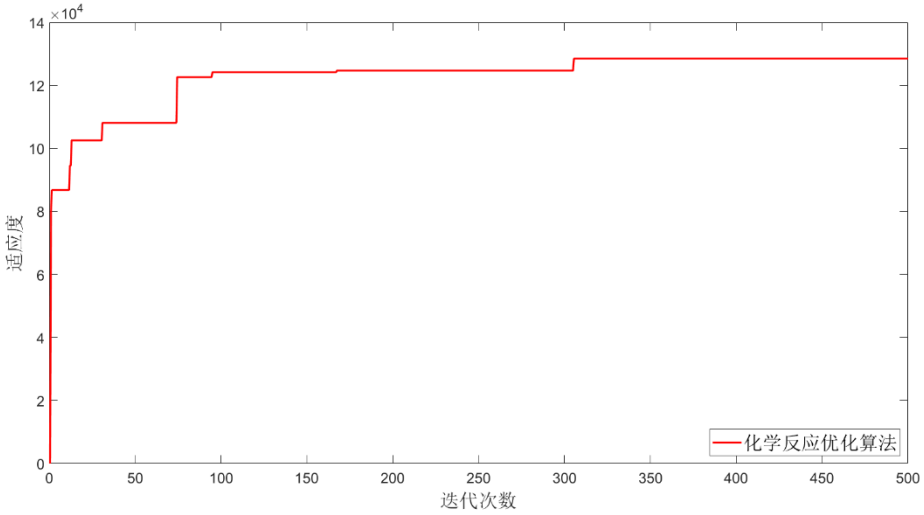

图 14 总得分与迭代次数关系图

可以看到，很多次化学反应算法都避免了陷入局部最优问题，最终优化得到了不错的结果。

五、问题 2 模型建立与求解

在问题重述与分析部分，我们归纳了问题 2 主要包括下述四个任务：

- 在突发道路故障后，为受影响车辆按照最短路径原则重新规划路线；
- 讨论道路突发事件对交通调控方案影响；

- 设计局部调整调控方案的方法；
- 给出调整后的全局交通信号灯调控方案；

其中第一项任务是最短路径问题，主要遇到的困难是数据量大，8000 个路口和 63968 条路段，无法人为给出最短路径，为此借助最短路径搜索算法求解得到最短路径；

第二项、第三项任务主要涉及城市交通突发服务中断问题研究，城市交通突发服务中断后，对事件影响的辐射判断以及针对此问题的调整策略需要在尽量短的时间内完成，否则容易发生大范围交通堵塞，增加交通安全风险，造成乘客严重延误等恶劣影响以及次生危害。因此，城市交通突发服务中断后，对影响的界定和现有交通的调整具有范围小、响应快的需求，对性能的需求反而可以退而求其次。为此，体现在算法中就是迭代收敛尽量快，不追求广泛的全局最优，只需要能解决燃眉之急，避免次生灾害发生。

第四项任务需要新的交通信号灯调控方案，在完成城市交通突发服务中断问题研究后，给出局部调整方法，便可得到。虽然把发生突发事件后路网交通及车辆新行驶路线作为初始值，可以利用问题 1 中全局优化调控算法得到新的全局优化调控策略，但是所需迭代次数多，时间开销大，考虑现实需求无法满足“快速响应”的需求，因此在新的交通信号灯调控方案产生过程，我们专门为突发事件后的原有调控策略调整方案设计了能快速响应的新算法，做到快速收敛、快速响应。

5.1 最短路径问题求解

针对最短路径问题，目前常用的方法包括 Dijkstra 算法、A-Star 算法、Floyd 算法和 Bellman-Ford 算法等。其中，Dijkstra 算法凭借寻优速度快、代码实现简单等特点，在路径优化、节点调控等领域得到了广泛应用。其核心思想在于以起点为中心向外层逐步扩展，求解从起点到其他所有顶点的最短路径，直到扩展到终点。

Dijkstra 算法是基于贪心思想实现的，利用这种类似广度优先搜索的方法来解决赋权图的单源最短路径问题，即指定一个点（源点）到其余各个顶点的最短路径。在本题中车辆行驶的交通网络属于拓扑结构相对固定、路径权值相对稳定的网络，因此我们可以采用 Dijkstra 算法来找出道路阻断后，受影响的车辆里程最短的路径。

考虑一个带权重的有向图$G(V,E)$，其中V 为所有顶点的集合，E 为存在连接指向关系的所有顶点之间的连线的边的集合，我们定义v_s 为源起点，v_e 为终点，$d_{ij}^{\min}$ 为从顶点 i 出发，到达顶点 j 的最短距离，m_{ij} 为顶点 i 到顶点 j 的有向路径，S 为已经求出的由起点 v_s 出发，到达的路径最短的点的集合，U 则是还未求出最短路径的顶点集合。如果存在由顶点 i 指向顶点 j 的路径，则等于从顶点 i 出发，到达顶点 j 的距离，否则定义 $d_{ij}=\infty$，且 l_{ij} 不存在。初始化时，$S=\{v_s\}$，定义起点 v_s 到自身的最短路径表示为 m_{ss}，相对应的最短距离 $d_{ij}^{\min}=d_{ss}=0$。

依据 Dijkstra 算法的基本思想，起点 v_s 到达终点 v_e 之间最短路径的搜索过程可以通过以下步骤实现。

第 1 步，找到起点 v_s 所有单跳可达的相邻顶点，从里面找出距离最短的相邻顶点(不妨设其为 v_1)，将其放入顶点集合 S 中，即有 $S=\{v_s,v_1\}$，则起点 v_s 到顶点 v_1 的最短路径距离为 $d_{ij}^{\min}=d_{s1}$，最短路程为 m_{s1}。

第 2 步，找出集合 S 中的顶点 $\{v_s,v_1\}$ 所有单跳即可到达的相邻顶点，我们不妨设其为 v_2 和 v_3。以 v_2 为例，起点 v_s 到顶点 v_2 的路径分别为 m_{s2} 或 $m_{s1}\to m_{12}$，相应的距离为 d_{s2} 或者 $d_{s1}\to d_{12}$。同样地，我们可以找出起点 v_s 到顶点 v_3 的路径和距离。

第 3 步，如果起点 v_s 到顶点 v_2 的最短路径距离小于起点 v_s 到顶点 v_3 的最短路径距离，则将具有最短距离的顶点 v_2 加入集合 S，且有 $d_{ij}^{\min}=\min\{\min\{d_{s2},d_{s1}+d_{12}\},\min\{d_{s3},d_{s1}+d_{13}\}\}$。此时有 $S=\{v_s,v_1,v_2\}$，起点 v_s 到顶点 v_2 的最短距离为 $d_{s2}^{\min}$。

第 4 步，依此类推，迭代遍历集合 U 中其他所有顶点并更新集合 S，直至终点 v_e 被加入到集合 S 中时，算法结束。

以图 15 的有向图为例，阐述一下 Dijkstra 算法计算从节点 1 到节点 5 的最短路径及距离的具体过程。

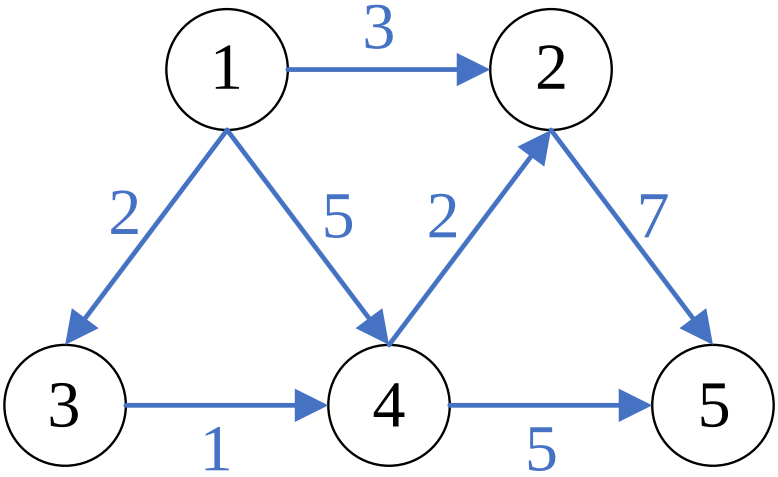

图 15 带权值的有向图

为了便于表达，我们不妨对图中的顶点赋予两个标号 (l_j,k_j)，第一个符号 l_j 表

示用算法计算得到的从起点v_s到v_j的最短路径的长度,第二个标号k_j表示在v_s到v_j的最短路径上的v_j前面一个相邻顶点的下标,用c_{ij}表示从顶点i到顶点j的有向路径的边权值，从而找到v_s到v_j的最短路径以及最短路径的距离。

解：

① 给起点v_1标号$(0,1)$，即v_1为起点，$d_{11}=0$。

② 此时，$S=\{v_1\}$，集合U中起点v_1单跳可达的顶点有v_2，v_3和v_4，则有

$$d_{12}=l_1+c_{12}=0+3=3$$
$$d_{13}=l_1+c_{13}=0+2=2$$
$$d_{14}=l_1+c_{14}=0+5=5$$
$$\min\{d_{12},d_{13},d_{14}\}=d_{13}=2$$

则我们给v_3标号$(2,1)$，即v_1到v_3最短路径距离为 2，该路径中v_3前一个顶点是v_1。

③ 此时，$S=\{v_1,v_3\}$，$U=\{v_2,v_4,v_5\}$，顶点v_3单跳可达的顶点为v_4，则

$$d'_{14}=l_3+c_{34}=2+1=3$$
$$\min\{d_{12},d_{14},d'_{14}\}=d_{12}=d'_{14}=3$$

则我们给v_2标号$(3,1)$，即v_1到v_2最短路径距离为 3，该路径中v_2前一个顶点是v_1；给v_4标号$(3,3)$，即v_1到v_4最短路径距离为 3，该路径中v_4前一个顶点是v_3。

④ 此时，$S=\{v_1,v_2,v_3,v_4\}$，$U=\{v_5\}$，此时有

$$d'_{15}=l_2+c_{25}=3+7=10$$
$$d''_{15}=l_4+c_{45}=3+5=8$$
$$\min\{d'_{15},d''_{15}\}=d''_{15}=8$$

我们给v_5标号$(8,4)$，即v_1到v_5最短路径距离为 8，该路径中v_5前一个顶点是v_4。

⑤ 此时，$S=\{v_1,v_2,v_3,v_4,v_5\}$，$U=\varnothing$，计算结束，我们得到最优结果如图 16。从起点$v_1$到终点$v_5$的最短路径为$m_{15}=\{v_1\to v_3\to v_4\to v_5\}$，最短距离为

$d_{15}^{\min}=8$。

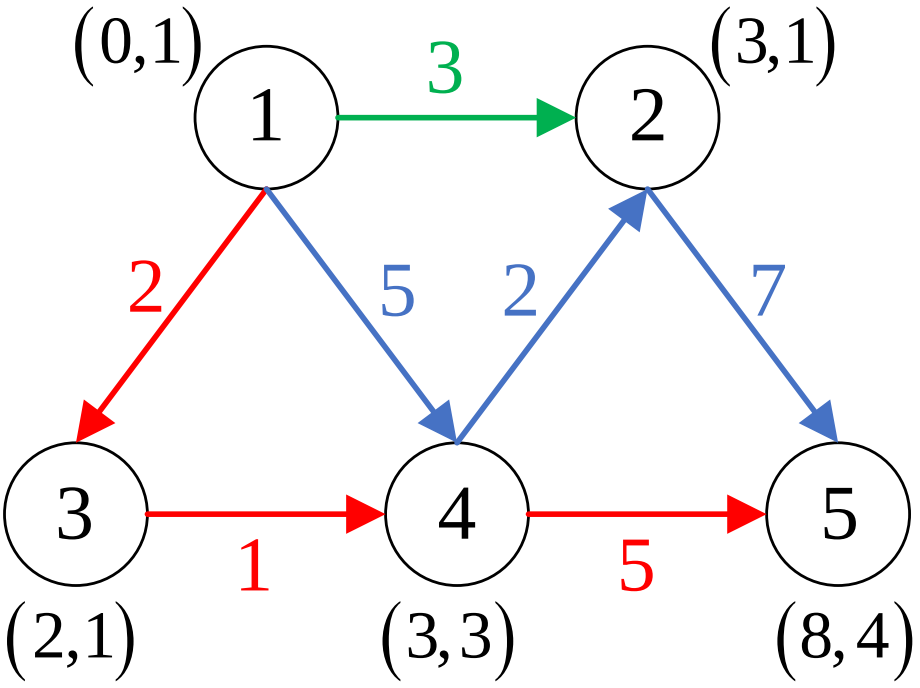

图 16 最短路径结果示意图

同时我们利用仿真可以得到相同的结果如图 17：

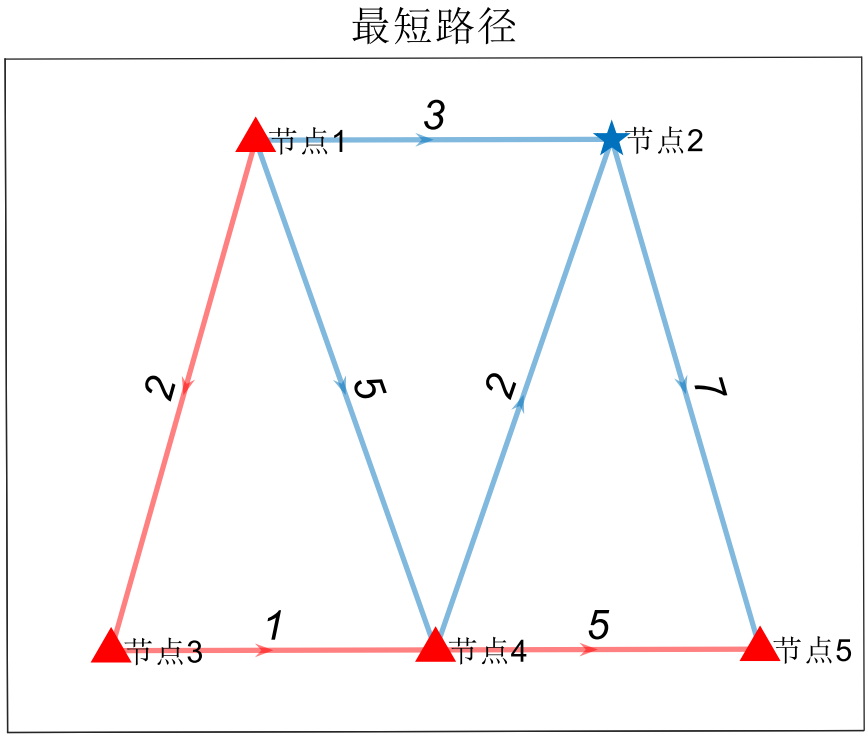

图 17 最短路径仿真结果图

在 $t=30$ s时，道路 ebf-ffef 即（即路口 415-路口 5545，对应 txt 文件中第 14424 条路）的起点处发生道路设施故障，道路 ebf-ffef 永久封闭。此时我们获

取题目中各车状态，并对所有车辆的行程路径信息进行筛选，最终得到受影响的车辆共有 3 辆，序号分别为 C16，C35，C50。且三者均未走出第 1 条道路，因此只需规划 3 辆车从最近的路口到达终点的最短路径，即分别找出 5908-1749，3948-2337，3992-1957 的最短路径。

5.2 城市交通突发服务中断问题求解

5.2.1 模型建立

问题 2 与问题 1 模型相似，交通信号灯策略表示以及在既定路线下小车行驶时间计算完全相同，但最终优化目标有所区别：

问题 1 中只需要所有小车尽可能早点到达终点即可，但是在问题 2 中，面对城市交通突发服务中断背景，一方面需要优化调整后小车尽可能早的到达终点——即交通畅通、减小突发事件带来影响；另一方面还需要该突发事件影响扩散尽可能小，即涉及到的优化调整范围尽可能小，这里用影响扩散图来进一步说明：

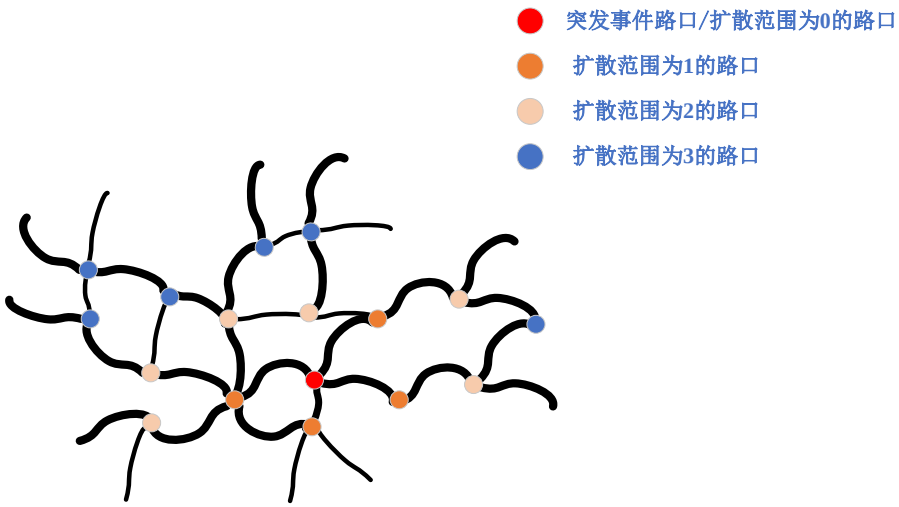

图 18 影响扩散范围图

特定红绿灯调控策略下，以所有小车得分量化调控策略的优劣，即，本题优化目标为：

$$\max\left(\sum_{k=1}^{N_V} M_k - \sum \rho_i\right) \tag{16}$$

一方面是累加分最大，另一方面扩散范围最小，也就是总目标=车辆总分-扩散程度。

5.2.2 算法求解

(1) 算法介绍

与其他进化寻优算法相类似，粒子群算法（PSO）算法通过个体间的协作与竞争，实现多维空间中最优解的搜索。

粒子群算法初始化为一群随机粒子（随机解），在 D 维搜索空间中，第 $i, i=1,2,...,m$ 个粒子在 D 维空间中的位置表示为 $x_i=(x_{i1},x_{i2},...,x_{iD})$，第 i 个粒子经历过的最好位置（即有最好适应度）记为 $P_i=(p_{i1},p_{i2},...,p_{iD})$，每个粒子的飞行速度为 $V_i=(v_{i1},v_{i2},...,v_{iD})$。在整个群体中，所有粒子经历过的最好位置为

$P_g = \left(p_{g1}, p_{g2}, ..., p_{gD}\right)$。每一代粒子根据式(17)(18)更新自己的速度和位置：

$$v_{id} = \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{17}$$

$$x_{id} = x_{id} + v_{id} \tag{18}$$

其中：ω为惯性权重；c_1和c_2为学习因子，通常，$c_1 = c_2 = 2$；r_1和r_2是$[0,1]$的随机数。更新过程中，粒子每一维的位置、速度均被限制在允许的范围之内。

基本 PSO 算法需要用户确定的参数不多，操作简单、使用方便，在求解小型优化问题时收敛快速，后续一些研究改进主要是提高该算法在复杂大型问题上的收敛速度。

(2) 算法框架

粒子群算法工作流程如下所示：

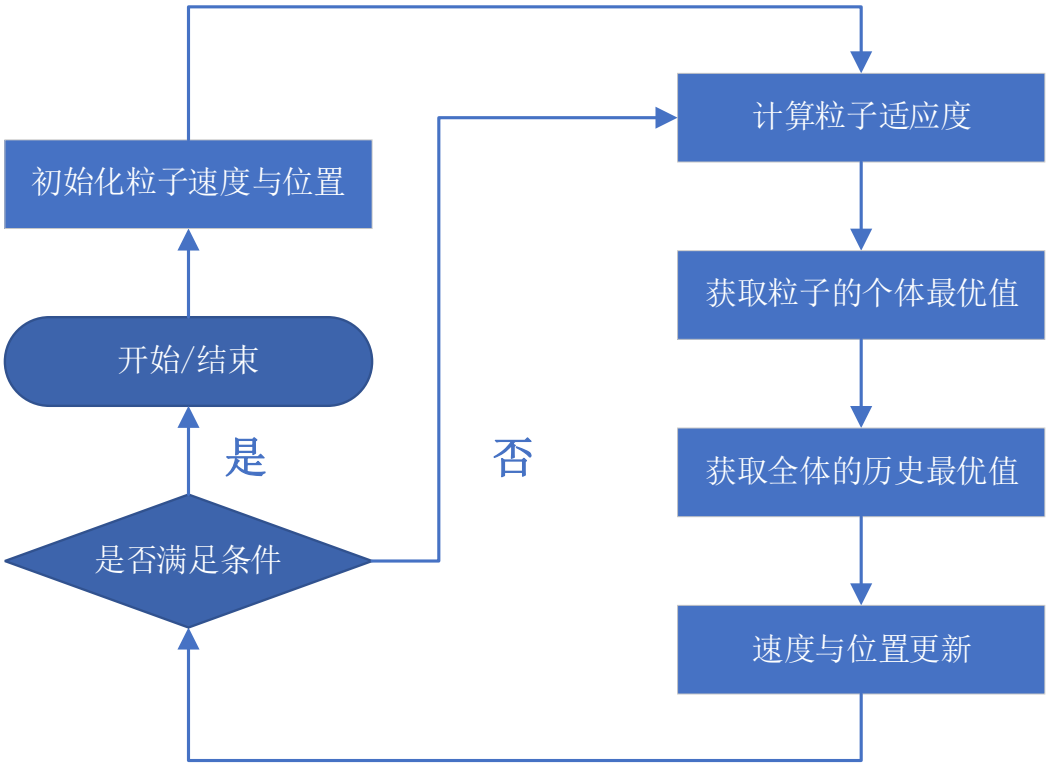

图 19 粒子群算法流程图

将问题模型应用到粒子群算法中，主要涉及到粒子编码和粒子适应度计算两方面。

(2.1) 粒子编码

粒子群算法通过设计一种无质量的粒子来模拟鸟群中的鸟，粒子仅具有两个属性：速度和位置，速度代表移动的快慢，位置代表移动的方向。

每个粒子表示一种调控策略，将调控策略编码为粒子 P 的过程如下：

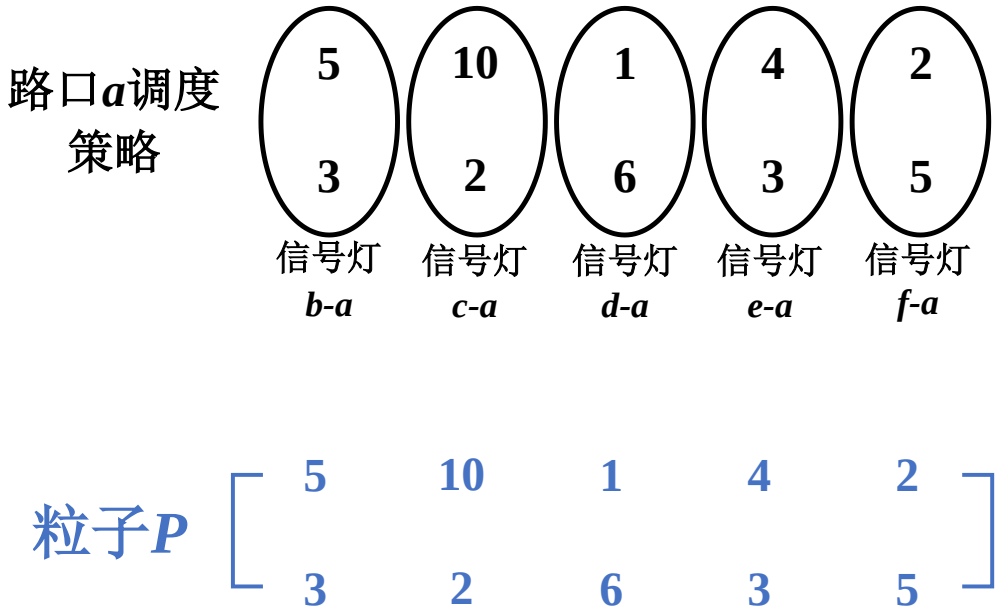

图 20 粒子编码示意图

(2.2) 粒子适应度计算

粒子适应度体现该调控策略带来的交通优化性能表现和影响范围确定。具体计算取决于模型优化目标，以适应度最大为目标，粒子适应度计算同式(16)。

(3) 伪代码设计

Algorithm 5 粒子群算法优化策略

```
初始化粒子速度和位置：
pop_x(chixv,1,i)=(xlukou(1,i))_xinxi(chixv,2))
pop_v_xlukou(1,i)(chixv,1)=rand(1,1)*3
while 满足迭代条件时 do
    对粒子的速度进行更新：
    pop_v_xlukou(1,i)(chixv,1)=w*pop_v_xlukou(chixv,1)+c1*rand(1,1)*3*((xlukou(1,i)_xinxi)(
    chixv,2)-pop_x(chixv,1))
    并对粒子的速度进行限幅处理：
    if pop_v_xlukou(chixv,1)<-3 then
        pop_v_xlukou(chixv,1)=mod(pop_v_xlukou(chixv,1),-3)
    else if pop_v_xlukou(chixv,1) >3 then
        pop_v_xlukou(chixv,1)=mod(pop_v_xlukou(chixv,1),3)
    end if
    为了防止粒子速度始终为 0
    if pop_v_xlukou(chixv,1)==0 then
        pop_v_xlukou(chixv,1)=rand(1,1)*3
    end if
    然后是对粒子的位置进行更新：
    pop_x(chixv,1,i)=pop_x(chixv,1,i)+pop_v_xlukou(1,i)(chixv,1)
    并对粒子位置进行限幅处理：
    if pop_x(chixv,1,i)<0 || pop_x(chixv,1,i)>5 then
        pop_x(chixv,1,i)=mod(abs(pop_x(chixv,1,i)),5)
    end if
    为了防止粒子位置始终为 0：
    if pop_x(chixv,1,i)==0  then
        pop_x(chixv,1,i)=round(rand(1,1)*4)+1
    end if
end while
```

解空间分为两个部分：路口各路灯的绿灯亮起顺序、路口各路灯的绿灯亮起持续时间。

主要针对路口各路灯的绿灯亮起持续时间采取粒子群算法进行优化，其中粒子的位置为绿灯亮起持续时间。

首先初始化粒子群的规模 m 和最大迭代次数 M ，同时初始化设置粒子群算法的惯性权重和学习因子。接着通过式(19)、(20)初始化生成粒子的初始位置和速度。然后是剩余初始群体的产生，针对绿灯亮起持续时间采用均匀分布的随机分布函数产生粒子速度 $pop_{v\_xlukou(1,i)}(chixv,1)$，再通过式子(19)得到粒子的位置。初始化完毕之后，针对每个粒子，根据式子(16)计算其适应度并将其作为初始的个体极值 E_{best} 和全局极值 Q_{best}。

然后进入循环启发式搜索过程：根据离散粒子群算法式子(21)、(22)更新粒子的速度和位置。然后根据绿灯亮起持续时间策略通过式(16)计算粒子适应度，如果适应度 E 优于个体极值 E_{best}，则更新 E_{best} 的位置。如果适应度 E 优于全局极值 Q_{best}，则更新 Q_{best} 的位置（同时在更新的过程中对粒子的速度和位置进行处理

从而限制其幅度）。最终通过多轮迭代之后，输出最优位置Q_{best}作为绿灯亮起的持续时间，从而得到了优化。

$$pop_x(chixv,1,i)=xlukou(1,i)\_xinxi(chixv,2) \tag{19}$$

$$pop_{v\_xlukou(1,i)}(chixv,1)=3*rand(1,1) \tag{20}$$

$$\begin{aligned}&pop_{v\_xlukou(1,i)}(chixv,1)=w*pop_{v\_xlukou(1,i)}(chixv,1)+c_1*3*rand(1,1)\\&*\left(\left(xlukou(1,i)\_xinxi\right)(chixv,2)-pop_x(chixv,1)\right)\end{aligned} \tag{21}$$

$$pop_x(chixv,1,i)=pop_x(chixv,1,i)+pop_{v\_xlukou(1,i)}(chixv,1) \tag{22}$$

5.3 结果分析

5.3.1. 故障发生时状态读取

故障发生时——即 30s 时对全局车辆状态的读取是后续最短路径求解、调整调控策略的基础。

以小车 C001 举例说明全局车辆状态信息表含义：

小车名称	所处路段	剩余时间	剩余路段数量
C001	fib−eggb	28	3

该表表示小车 C001A 当前在路 fib−eggb 上行驶，还需 28 秒到达路口 eggb，车辆 C001 距离到终点还需要经过 3 条路段；

在 30s 时全局车辆状态信息如下表所示：

小车名称	所处路段	剩余时间	剩余路段数量		小车名称	所处路段	剩余时间	剩余路段数量
C001	fib−eggb	28	3		C101	geic−ghbf	0	29
C002	gebf−ebeb	22	9		C102	bja−ebie	29	7
C003	ddbj−did	0	23		C103	dig−jde	0	6
C004	ciac−hbii	3	12		C104	fgab−ehcj	38	8
C005	ieh−ehjb	29	16		C105	bj−fcdg	16	24
C006	edgb−ggeh	1	3		C106	cbg−cg	29	25
C007	gfib−hgfa	0	9		C107	hijg−ddai	10	22
C008	dehd−ccb	31	3		C108	ciee−hcc	9	16
C009	bb−cbga	6	24		C109	dei−jhc	0	18
C010	hfi−hb	13	14		C110	wait	4	20
C011	ggji−ide	17	22		C111	bfga−cbjg	13	14
C012	bgbf−ijf	15	16		C112	edhb−jgf	19	21
C013	hchc−eghc	11	8		C113	dgjb−heej	10	14
C014	egg−ge	27	2		C114	eajb−egia	10	14
C015	cgbg−bcb	1	18		C115	faa−fcee	4	13
C016	dbh−fjai	0	8		C116	bbb−hbe	17	15
C017	fhbj−bg	37	2		C117	ccb−heaa	6	25
C018	cbcb−jjh	5	20		C118	ejcg−dchj	25	2

C019	eadc−cjdf	9	9		C119	bh−djbc	39	16
C020	cchd−cbcb	31	7		C120	bgch−ciie	3	21
C021	jdd−jf	12	6		C121	hffc−cce	3	5
C022	cjci−gdbc	6	6		C122	dijc−bgfj	22	21
C023	edie−chf	11	18		C123	fjje−djbd	7	10
C024	cbea−ijf	10	16		C124	cadg−fedi	3	25
C025	chcg−gd	11	13		C125	bdhd−jbb	8	8
C026	dhh−cce	24	16		C126	f−daj	11	26
C027	gddf−bdec	35	23		C127	bifa−dj	19	7
C028	ehfb−edgd	19	5		C128	c−hf	2	20
C029	wait	1	7		C129	gfee−ccgd	9	19
C030	efef−bebc	10	14		C130	bieb−jaa	4	17
C031	cbfj−fdag	25	21		C131	he−dgi	22	8
C032	bgbh−cee	15	22		C132	hgaj−bhdf	11	19
C033	ibi−ba	12	22		C133	bdbi−bfad	5	26
C034	icg−feag	28	7		C134	bah−ccg	11	15
C035	bch−djei	11	20		C135	gbjf−bfh	17	26
C036	fbhe−if	4	29		C136	ficc−egad	14	25
C037	eeja−fffd	32	24		C137	cah−bhg	23	4
C038	bid−fd	17	22		C138	djb−bdfg	16	27
C039	ga−bgdj	9	4		C139	dadb−bgah	34	16
C040	gch−haab	10	3		C140	eccc−bhfg	12	10
C041	ghhc−ebji	10	26		C141	cgih−cdd	0	28
C042	beb−cgja	20	26		C142	gjae−hahb	17	17
C043	febi−dbgb	33	3		C143	fejd−hc	12	29
C044	gcjj−bade	39	5		C144	fedj−fah	26	7
C045	cbii−heeb	16	2		C145	fcjj−cfdd	18	8
C046	giei−fhba	3	14		C146	ehdg−ied	5	9
C047	gbdj−ecej	2	14		C147	chjf−def	0	23
C048	gdji−chdh	40	4		C148	hjji−bij	0	24
C049	cbdj−chaa	13	27		C149	bbfj−bj	9	18
C050	gg−djjc	9	27		C150	djhf−chec	13	4
C051	eiih−fgci	8	29		C151	ebga−hifj	29	28
C052	icf−gbai	10	28		C152	bd−babh	3	11
C053	ide−hhgc	10	24		C153	edig−bd	4	18
C054	fefd−dedi	13	25		C154	cjei−ea	4	27
C055	dfbf−bhia	32	7		C155	bgfd−bg	3	23
C056	eihj−egea	12	11		C156	cjf−dejb	36	10
C057	wait	12	11		C157	beje−bgji	14	5
C058	fjdi−gbjh	13	10		C158	djf−ffdi	5	26
C059	bfij−gcic	14	23		C159	efdi−gc	22	14
C060	fbg−gch	27	12		C160	djfj−dd	8	8
C061	gde−gcfi	4	28		C161	cg−eebi	17	26

C062	hfee−dgfb	37	25		C162	cjhc−chgh	2	25
C063	fgi−cfg	20	15		C163	fajj−geej	5	27
C064	ccgg−ccaf	13	3		C164	bf−cbe	16	20
C065	dc−bfgc	1	16		C165	fjde−baje	4	25
C066	ddfg−dhh	23	27		C166	feib−bidg	21	6
C067	hb−fb	1	21		C167	ccgg−hgaj	33	28
C068	ibc−hjed	18	11		C168	bdg−jb	7	12
C069	cijh−chcc	32	10		C169	bbgg−bjdd	4	19
C070	fbgj−ggdc	15	20		C170	fgij−dhj	2	25
C071	gejh−hb	26	22		C171	wait	2	16
C072	bfdc−jfa	18	6		C172	caja−gegf	1	4
C073	diff−ggcc	29	22		C173	cegg−bfga	20	24
C074	bbfh−cfef	5	25		C174	hbe−beej	3	16
C075	hch−e	43	26		C175	bbcc−fadg	12	17
C076	bhaa−hddc	17	4		C176	ebfi−hdje	1	14
C077	dbdc−eceh	28	8		C177	hgbh−eejc	20	22
C078	egf−hjef	7	12		C178	bbcj−efha	9	26
C079	bcje−hbgc	2	2		C179	bhe−ddbj	13	6
C080	gadf−ggcg	15	17		C180	bjgg−bdcj	16	14
C081	bje−gdej	7	27		C181	ejai−gija	33	27
C082	geej−bfeh	22	22		C182	fich−febg	23	10
C083	baea−ehea	6	15		C183	ddhc−fcg	7	28
C084	hhhf−eihh	26	13		C184	efc−jbf	9	25
C085	ebja−igi	11	25		C185	cdbj−bdgf	27	15
C086	ccia−bcjg	0	18		C186	bdac−cdbf	6	9
C087	eech−hegh	31	8		C187	efia−fgd	3	12
C088	bdad−gcb	19	13		C188	fcbh−cheg	20	10
C089	eajb−edah	14	21		C189	gdda−g	11	19
C090	bcg−ccb	21	17		C190	ggde−cjhf	1	13
C091	jg−dgef	9	12		C191	bbfi−bebh	30	6
C092	fcf−fcg	6	25		C192	fig−hffb	1	26
C093	fgdc−gbfg	4	22		C193	ddab−dei	9	9
C094	bhca−fedc	1	22		C194	bggg−cfae	36	19
C095	bicg−cgcc	18	6		C195	efea−cbja	25	16
C096	hdgj−gifb	12	16		C196	egjj−ffdg	21	20
C097	bcaj−hjdh	19	5		C197	cjii−ib	23	14
C098	dggg−hjbb	1	19		C198	de−gcfc	0	15
C099	fh−dbij	3	28		C199	gicc−gcjd	40	23
C100	bbcc−fadg	16	12		C200	fdai−fjab	1	8

5.3.2 最短路径结果

道路故障发生后，需要改变行驶路径的车辆是小车 C016、C035 和 C050。根据最短路径算法求解得到的行驶路径如下所示：

小车编号	经过路径数	经过路径							
C016	7	dbh-fjai	fjai-ebje	ebje-bheb	bheb-bch	bch-cd	cd-f	f-bhej	
C035	8	bch-djei	djei-b	b-fegd	fegd-bbif	bbif-cace	cace-ffjj	ffjj-cgci	cgci-cddh
C050	6	gg-djjc	djjc-bc	bc-ba	ba-ijh	ijh-bhi	bhi-bjfh		

图 21 需要调整的车辆最短路径结果图

5.3.3 调整后的调控策略

道路事故发生后的交通信号灯调控策略因为数据量太大，同样难以在正文中展示，我们按照题目要求将其存储到 result2. txt 中作为附件上传。通过该全局交通信号灯调控策略，利用所有小车到达终点时间代入式(8)计算总得分，该策略对应的总得分是 135873。

5.3.4 此类事件对交通灯调控方案的影响

首先理论分析，在已经规划好的交通系统中，突发道路故障引起车辆行驶路线改变，故障发生路口临近的路线车流量及附近路口交通拥堵情况都会改变，针对原有车辆行驶路线设计的交通信号灯调控系统不再体现“最佳”。需要进行调整使之重新适应当前交通状况。

其次我们通过对比本题中突发故障发生前后针对不同行驶路线情况，我们问题 1 中设计的全局优化算法性能表现以及对比突发故障发生后，调整前后的全局交通信号灯调控策略性能表现分析此类事件对交通灯调控方案的影响。

在两次对比中，调控算法性能表现均通过该策略下小车行驶时间计算得分大小来体现，总得分越大，性能越好。两次对比情况具体如下图所示：

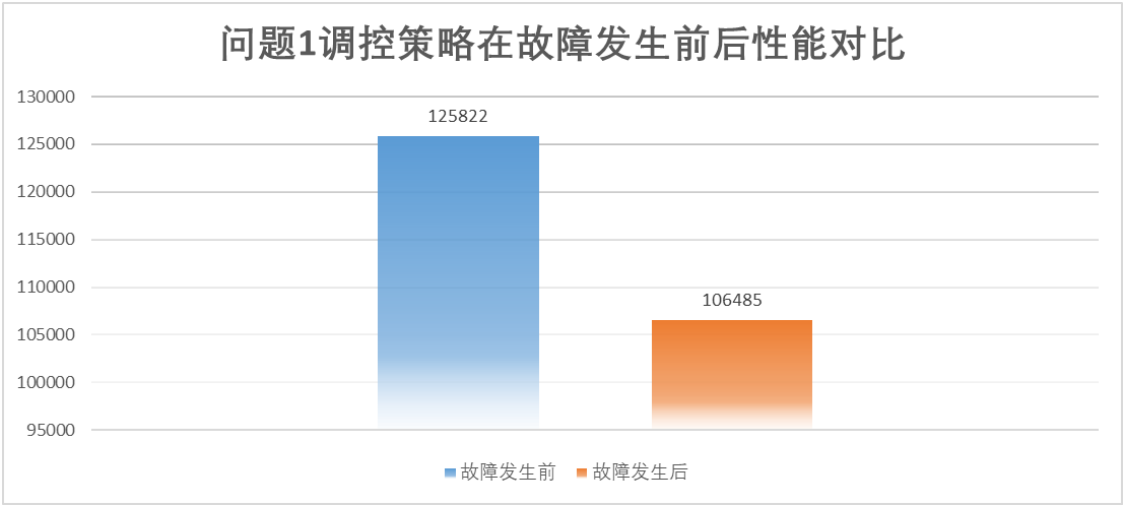

图 22 问题 1 调控策略故障发生前后性能对比图

由上图对比可得，突发故障使原有调控策略性能下降 15%左右，分析主要原因是因为故障发生后，一些信号灯在原有策略中会亮绿灯，但故障发生后实际路口已经没有车辆通过，造成资源浪费，另一方面为受影响小车重新规划的路线，有些在之前的调控策略中一直红灯，但是重新规划路线后实际有小车通过，导致这些小车无法到达终点、得分为 0。

六、模型评价与改进

（一）第 1 问

交通信号灯全局优化调控策略是一个典型多目标优化问题，全局优化调控策

略由各个路口信号灯相互独立的调控策略组成，每个路口信号灯调控策略包括各路口信号灯亮绿灯顺序和绿灯持续时间。而优化目标则是希望按照既定路线行驶的小车尽量快的到达终点，题目中给出了由“尽量快”到数学表示的公式，即单个小车得分。行驶时间越短，得分越高。因此就第 1 问来说解决问题难点在于数据的复杂度以及调控策略到算法编码的转化方式。

面对数据复杂问题：

我们首先通过数据预处理发现总共 8000 个路口只有 2577 个路口有车辆经过、需要进行调控；63968 条路段只有 3497 条路段有小车经过，从而大大缩小了解控间的维度及范围。此外通过查阅文献对比分析，我们了解到较为先进且在复杂解空间问题上，化学反应算法能够有效地解决多目标优化问题，并且在二次分配问题，网格任务调控问题、人工神经网络训练、资源受限项目调控问题等方面得到广泛的应用。且较传统遗传算法收敛速度快、表现优异的化学反应算法。因此通过上述两种手段，我们将道路交通信号灯全局调控这个典型 NP-hard 问题简化并求解，还在小规模数据上将其与其他几种传统智能算法进行性能对比，并分析其计算复杂度与收敛性，进一步证明了模型算法的有效性。

面对调控策略难以转化问题：

单个路口调控策略包含每个信号灯亮绿灯的顺序和绿灯 ，绿灯持续时间容易编码，但如何表示每个信号灯亮绿灯的顺序并且还要方便后续算法操作是个难题。为此，我们创新改进提出用区间内数字大小表示每个信号灯亮绿灯的顺序举例详细说明。这种创新性的操作使得在本题中化学反应算法的各项分子操作得以实现，并最终求出最优解。

最后，我们把优化结果按照题目要求存储到 result1.txt 文件中并在论文中展示部分进行示例。

（二）第 2 问

第二问涉及多个问题，最主要的是突发道路故障后，受影响小车最短路径规划问题以及已有调控策略调整方法。剩下两个问题突发道路故障的影响以及新的调控策略生成可以通过上面两个问题求解后得到。

小车最短路径规划问题：

小车最短路径规划问题是一个传统老问题，唯一的难点是 30 s 时全局状态掌握以及数据量较大带来的问题难度增加。本题中对 30 s 时全局状态掌握主要取决于建模环节及仿真环节的逻辑通顺，在对问题分析透彻及代码编写逻辑顺畅基础上，令系统在$t = 30$ s时暂停，输出各项变量，便可以轻松得到此时全局状态以及受影响小车状态。

最后已知起点、终点和道路结构的最短路径规划问题我们使用传统 Dijkstra 算法进行单源最短路径问题求解并将结果按照题目要求存储到 newcars.txt 文件中。

已有调控策略调整方法：

突发道路故障后，联系实际最重要的是快速规划新的交通信号灯工作策略，疏散故障路段车流量，避免次生灾害和社会不稳定隐患滋生。又因为交通调控策略变化会给整个系统带来不稳定因素，因此面对突发道路故障，我们希望把该事件带来的影响缩小到周围几个路口，避免全局调控带来的“开销大、不稳定”因素。

为此，在交通策略模型与小车行驶时间模型与问题 1 相同的情况下，我们综合影响范围和调整后调控策略性能综合设计优化目标函数，针对小规模寻优问题

选择传统粒子群算法进行求解。粒子群算法的特点就是迅速，且在求解小型问题时与其他算法、甚至是改进的粒子群算法性能相差不大，反而难度、速度上表现更优，因此在该问题上我们选择了粒子群算法进行求解，做出快速反应。并且结果证明我们提出的模型策略能够有效减少车辆在路口处的排队长度和等候时间，达到了缓解交通拥堵的目标。

突发道路故障的影响：

突发道路故障影响很难用数学公式去推导，在实际生活中可以通过搜集数据做拟合回归且拟合后的函数只适用于搜集数据的特定场景。但在该题中很明显无法这样做，因此我们通过比较突发道路故障事件前后，调整前的调控策略和调整后的调控策略性能差异来进行分析。

新的调控策略生成

最终新的调控策略生成通过上述调整方法代入数据即可得到，在此不再赘述。

（三）展望

至此两个问题解答完毕，我们将类似问题的可以进一步研究方向进行展望：

当今是智能时代，机器学习、人工智能发展如火如荼，并且在多个研究领域得到了进一步改进和发展。传统理论知识对问题的建模总是离不开各种各样的假设，越是简单的模型需要的假设就越多、越苛刻。就以无线通信信道建模举例：基础理论中教授的各种确定性理论公式推导计算都是在理想的高斯白噪声信道下进行的，稍微考虑现实生活中的衰落和多径延时，建模为瑞利信道和莱斯信道之后，很多原有的公式不再适用，一些计算结果只能以概率、分布来表示。

对应到本题中，在路段终点处排队不考虑小车空间大小等一系列假设，都帮助我们很容易的理解问题、进行求解，但是推广应用之后会有各种不适应情况。

因此，“研究总要应用”，与实际生活场景结合的越多，建立准确模型难度就越高，求解就越困难。为此，新兴的机器学习可以很好地解决这类问题，机器学习以数据驱动模型，挖掘事物、逻辑之间的深度特征联系，不再受限于模型的各项假设，在很多领域的实际应用中都取得了不错的效果和反响。

结合本题来说，考虑使用机器学习，智能分析当前物流情况和对未来交通物流进行预测，在解决这类确定性问题上性能不一定能提高多少——原因主要是我们设置了一系列假设来简化模型问题难度，但是在这个问题的推广应用上，实际交通物流中，机器学习、神经网络的应用，不仅理论上能起到更好作用，一些实践也已经证明了这类算法的强大能力和有效性。

此外，此类交通调控策略问题，不仅仅是一个多目标寻优问题，问题本身也具备一定实际背景，交通领域关系民生经济发展，交通领域的研究广度深度都有着不错的基础，一些交通领域先进研究成果的带入应用，在解决此类问题上会产生出其不意的效果，比如我们在解题过程中考虑过的对应“绿波时速”概念引入“绿波策略”，结合小车行驶过程考虑相邻交通信号灯相关性，以信号灯相关性为优化目标进行设计，让每辆小车“畅通无阻”，理论上是可以达到所需的最短时间。但是苦于搜集到的文献不多、以及难以实现问题，最终我们放弃了这两种算法的使用，选择了更为稳定的智能搜索算法进行求解，希望在未来有机会在此方向做进一步研究。

参考文献

[1] 刘伟,黄宇成,杜薇,王伟.移动边缘计算中资源受限的串行任务卸载策略[J].软

件学报,2020,31(06):1889-1908.
[2] 李端,钱富才,李力,高建军.动态规划问题研究[J].系统工程理论与实践,2007(08):56-64.
[3] 高丽,曾庆良,范文慧.多学科设计优化中的智能算法比较[J].计算机应用研究,2008(01):93-95.
[4] 唐贤伦,庄陵,李银国,曹长修.混合粒子群优化算法优化前向神经网络结构和参数[J].计算机应用研究,2007(12):91-93.
[5] 李金澎,丁博,王雨,吕思浓.基于 GA 和 LSTM 的智能交通灯调控方法[J].物联网技术,2019,9(12):50-54.
[6] 王鼎湘,李茂军.基于车流量的交通灯智能控制算法[J].计算机应用与软件,2015,32(06):241-244.
[7] 陈秋莲,郑以君,蒋环宇,陈燕.基于神经网络改进粒子群算法的动态路径规划[J].华中科技大学学报(自然科学版),2021,49(02):51-55.
[8] 李远平,蔡远利,李济生.基于改进粒子群算法的月地转移轨道优化[J].工程力学,2020,37(03):238-244.
[9] 肖华军,柴子力,张超勇,孟磊磊,任亚平,梅慧文.基于混合化学反应算法的柔性作业车间调控[J].计算机集成制造系统,2018,24(09):2234-2245.
[10] 吴高超. 基于粒子群算法的路径规划问题研究[D].燕山大学,2016.
[11] 晏勇,雷航,赵晓雨,梁潘.基于无线传感器网络的自适应交通信号灯控制系统[J].实验室研究与探索,2018,37(12):90-93+191.
[12] 张东波,林永杰,卢凯,首艳芳,徐建闽.基于云计算的大规模交通路网的最短路径算法[J].华南理工大学学报(自然科学版),2018,46(12):139-146.
[13] 赵卫绩,巩占宇,王雯,樊守芳.几种经典的最短路径算法比较分析[J].赤峰学院学报(自然科学版),2018,34(12):47-49.
[14] 张伟,谢源海,王亚刚.基于收敛性分析的偏差粒子群算法及 PID 仿真应用[J].控制工程,2021,28(07):1466-1473.
[15] 李林. 连续流绿波交通控制理论与方法研究[D].华南理工大学,2011.
[16] 赵彪. 面向全路网交通流优化的交通灯调控方法研究[D].华东师范大学,2016.
[17] 邵明莉,曹鹗,胡铭,章玥,陈闻杰,陈铭松.面向优先车辆感知的交通灯优化控制方法[J].软件学报,2021,32(08):2425-2438.
[18] 朱佳莹,高茂庭.融合粒子群与改进蚁群算法的 AUV 路径规划算法[J].计算机工程与应用,2021,57(06):267-273.
[19] 王玉,申铉京,周昱洲,林鸿斌.一种求解交通网络中最短路径问题的人工蜂群算法[J].吉林大学学报(理学版),2021,59(05):1144-1150.
[20] 王云. 应对城市轨道交通突发服务中断的接驳公交应急优化研究[D].北京交通大学,2017.
[21] Yun Wang,Xingquan Zuo.An Effective Cloud Workflow Scheduling Approach Combining PSO and Idle Time Slot-Aware Rules[J].IEEE/CAA Journal of Automatica Sinica,2021,8(05):1079-1094.
[22] Wenbin Hu, Huan Wang, Liping Yan, and Bo Du. 2016. A swarm intelligent method for traffic light scheduling: application to real urban traffic networks. Applied Intelligence 44, 1 (January 2016), 208–231. DOI:https://doi-org-s.nudtproxy.yitlink.com/10.1007/s10489-015-0701-y.
[23] Cui M, Levinson D. Shortest paths, travel costs, and traffic. Environment and

Planning B: Urban Analytics and City Science. 2021;48(4):828-844. doi:10.1177/2399808319897619.
[24]刘峰博,周庭梁,王小敏.城市轨道交通故障下客流分布计算及评估方法[J].西南交通大学学报,2021,56(05):921-927+966.
[25]李晓刚,王伟,贾庆东.城市轨道交通故障处置辅助决策系统关键技术研究[J].自动化仪表,2021,42(06):43-47.
[26]张添豪. 基于交通流量预测的动态干道绿波信号协调控制及系统设计[D].南通大学,2020.

附录

问题一代码：

```
%对数据进行初步的评估，从而得出 % lushu.mat 没有重复的路、
%lu_x.mat    重复的路、%xlukou.mat   没有重复的路口

clc;
clear all;
load('cars.mat')
daiqiu=[];
lu_x=[];
for i=1:200
xx=cars(i,2);
u=str2num(table2array(xx));
for k=1:u

    lu_x=[lu_x,table2array(cars(i,k+2))];
end

for j=1:u-1
    x=cars(i,j+3);
y=table2array(x);
z=strsplit(y,'-');
a=z(1,1);
b=z(1,2);
daiqiu=[daiqiu,a,b];
```

```
end
end
xlukou=unique(daiqiu);
lushu=unique(lu_x);
save('lushu.mat','lushu')          % lushu.mat 没有重复的路
save('lu_x.mat','lu_x')            %lu_x.mat    重复的路
save('xlukou.mat','xlukou')        %xlukou.mat   没有重复的路口
```

%对数据进行进一步的分析，从而知道路中重复路的个数

```
clc;
clear all;
load('lushu.mat');
load('lu_x.mat');
load('xlukou.mat');
for i=1:3569
    jj=1;
    for j=i+1:3569
        if strcmp(lu_x(1,i),lu_x(1,j))
            jj=jj+1;
        end
    end
    lu_x(2,i)=jj;
end
save('lu_xshu2.mat','lu_x');       %lu_xshu2.mat    重复的路两行（第二行是路数）
```

%数据的进一步处理，从而知道了每个路口指向的路数   %xlukou_jishu.mat 路口（统计了指向终点的路数）

```
clc;
clear all;
```

```
load('lushu.mat')
load('xlukou.mat')
for  i=1:2577
     xlukou(2,i)=0;
end


for  i=1:3497
     a=lushu(1,i);
     z=strsplit(a,'-');
     b=z(1,2);


     for  j=1:2577
     if  strcmp(b,xlukou(1,j))
          bbb=str2num(xlukou(2,j));
          bbb=bbb+1;
          xlukou(2,j)=bbb;
     end
     end
end


save('xlukou_jishu.mat','xlukou');     %xlukou_jishu.mat 路口（统计了指
```

向终点的路数）

```
%策略生成区


clc;
clear  all;
load('xlukou_jishu.mat  ');
load('lushu.mat');
%a=getfield(lukou_xinxi(1:2),  'a');          获取 struct 信息的函数
%  for  i=1:2577
%       lukou_xinxi=struct(xlukou(1,1),);
%a=xlukou(1,1);
```

```
for  i=1:2577
    eval([num2str(xlukou(1,i)),'_xinxi=[];']);


    for  j=1:3497
        aa=lushu(1,j);
        aaa=strsplit(aa,'-');
        bbb=num2str(aaa(1,2));
        if  strcmp(bbb,xlukou(1,i))

eval([num2str(xlukou(1,i)),'_xinxi=[',num2str(xlukou(1,i)),'_xinxi;aa
,round(rand(1,1)*4)+1];']);


        end
    end
    eval(['save("D:\桌面\省数模\A 题\代码\路口的具体路数数组群
\',num2str(xlukou(1,i)),'_xinxi.mat","',num2str(xlukou(1,i)),'_xinxi
")']);
    fprintf("在学啦在学啦，还有%f 个\n",2577-i);


end
```

```
%%%求得分（考虑了排队）


clc;
clear  all;
load('cars.mat')
load('streets.mat')


streets_str=table2array(streets);
```

```
cars_str=table2array(cars);


%初始化
cars_state=[];
car_t=zeros(200,1);                %car_t 为车的全局时间
for  i=1:200
     [m,n]=find(streets_str==num2str(cars_str(i,3)));
     cars_str_shu=str2num(cars_str(i,2))-1;


cars_state=[cars_state;cars_str(i,3),streets_str(m,4),num2str(cars_str_shu)]
;
end
t_quan=0;       %全局时间
%%%
cars_zaiwei=200;     %200 个车都还在


while(cars_zaiwei>0)



for  i=1:200       %往前推动了 1s
     car_state_3=str2num(cars_state(i,3));
     if  car_state_3~=-1


     cars_state(i,2)=num2str(str2num(cars_state(i,2))-1);
     end


end
```

```
t_quan=t_quan+1;


% %%寻找第 30 秒时各车的状态
% if t_quan==30
%     fprintf("找到虎虎啦!!!! ");
%     save('cars_state_30.mat','cars_state');
% end


for i=1:200
    t=str2num(cars_state(i,2));
    cars_state_shu=str2num(cars_state(i,3));
    street_name=cars_state(i,1);
    if t==0 && cars_state_shu>0 && street_name~="wait" %进入判决
        % t_jia=Panjue(i,cars_state,t_quan);


        t_jia=0;
    street_name=cars_state(i,1);




    lukou_chuli=strsplit(street_name,'-');
    lukou_name=lukou_chuli(1,2);
    %搜索路口具体情况
    eval(['load("D:\桌面\省数模\A 题\代码\路口的具体路数数组群
\',num2str(lukou_name),'_xinxi.mat")']);
    %根据路口的交通灯数进行分类讨论
    eval(['size=numel(',num2str(lukou_name),'_xinxi)/2;']);    %如何
求 string 组的大小
```

```
if  size>1          %路灯为 1 时
    t_lu=0;
    t_lu_ban=0;
    %要判断是否排队
    [m,n]=find(cars_str==cars_state(i,1));
    size_m=length(m);
    if  size_m==2   %则可能会排队
        for  jj=1:size
eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
总时间
            t_lu=t_lu+t;
        end
eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序
eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']);  %t_lu_dangqian 为当前路的持续时间
        if  strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))  &&
cars_state(m(1,1),2)==cars_state(m(2,1),2)
            %则可能排队
            if  t_lu_dangqian==1
                %在判断当前车是小还是大
                m_max=max(m);
                if  i==m_max
                    %则等待一个周期
                    t_jia=t_jia+t_lu;
                end
            end
        end
    elseif  size_m==3  %则可能会排队
        for  jj=1:size
```

```
eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);    %t_lu 为总时间
                t_lu=t_lu+t;
            end

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2));']); %t_lu_dangqian 为当前路的持续时间
            if strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))||strcmp(cars_state(m(1,1),1),cars_state(m(3,1),1))||strcmp(cars_state(m(2,1),1),cars_state(m(3,1),1))
                %则可能排队

                %判断是两个还是三个
                if strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1)) && cars_state(m(1,1),2)==cars_state(m(2,1),2)
                    %则可能排队
                    if t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m(1,1),m(2,1));
                        if i==m_max
                            %则等待一个周期
                            t_jia=t_jia+t_lu;
                        end
                    end
                elseif strcmp(cars_state(m(1,1),1),cars_state(m(3,1),1)) && cars_state(m(1,1),2)==cars_state(m(3,1),2)
                    %则可能排队
                    if t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m(1,1),m(3,1));
                        if i==m_max
                            %则等待一个周期
```

```
                t_jia=t_jia+t_lu;
            end
        end
    elseif
strcmp(cars_state(m(2,1),1),cars_state(m(3,1),1)) &&
cars_state(m(2,1),2)==cars_state(m(3,1),2)
        %则可能排队
        if t_lu_dangqian==1
            %在判断当前车是小还是大
            m_max=max(m(2,1),m(3,1));
            if i==m_max
                %则等待一个周期
                t_jia=t_jia+t_lu;
            end
        end
    elseif
strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))&&strcmp(cars_state(m
(1,1),1),cars_state(m(3,1),1))&&strcmp(cars_state(m(2,1),1),cars_stat
e(m(3,1),1))&& cars_state(m(1,1),2)==cars_state(m(3,1),2)&&
cars_state(m(2,1),2)==cars_state(m(3,1),2)
        %则可能排队
        if t_lu_dangqian==1
            %在判断当前车是小还是大
            m_max=max(m);
            m_min=min(m);
            if i~=m_max && i~=m_min
                %则等待一个周期
                t_jia=t_jia+t_lu;
            elseif i==m_max
                t_jia=t_jia+t_lu+t_lu;
            end
        elseif t_lu_dangqian==2
            m_max=max(m);
            if i==m_max
                t_jia=t_jia+t_lu;
            end
        end
    end
```

```
            end
        end


        for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
总时间
        t_lu=t_lu+t;
        end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']);  %t_lu_dangqian 为当前路的持续时间
        if  lu_cixv~=1
          for  jjj=1:lu_cixv

eval(['ttt=str2num(',num2str(lukou_name),'_xinxi(jjj,2));']);
            t_lu_ban=t_lu_ban+ttt;                %t_lu_ban 为算上当前路后的
时间
        end
        t_lu_banjian=t_lu_ban-t_lu_dangqian;     %t_lu_banjian 为不算上当前
路的时间
        if  mod(t_quan,t_lu)<  t_lu_banjian
            t_jia=t_jia+(t_lu_banjian-mod(t_quan,t_lu));
        elseif   mod(t_quan,t_lu)  >  t_lu_ban
            t_jia=t_jia+(t_lu_banjian+t_lu-mod(t_quan,t_lu));
        end
        end
    end


%           if  t_jia<0
%           fprintf("11 找到虎虎啦!!!!!!  \n");
```

```
%           end


        if  t_jia==0
            %更新状态

            %fprintf("会有虎虎啦!!!!!! \n");

            t=str2num(cars_str(i,2))+2;
            cars_state(i,1)=cars_str(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state(i,1)));
            cars_state(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state(i,3));
            cars_state(i,3)=num2str(cars_state_3-1);
        else
            cars_state(i,1)="wait";
            cars_state(i,2)=num2str(t_jia);
        end


    elseif  street_name=="wait"
        car_state_2=str2num(cars_state(i,2));
%           if  cars_state(i,1)=="wait"  &&   car_state_2<0
%               fprintf("22 找到虎虎啦!!!! ");
%           end


        if  cars_state(i,1)=="wait"  &&  cars_state(i,2)=="0"
            %fprintf("虎虎呢？？？？\n");

            %更新状态

            t=str2num(cars_state(i,2))+2;
            cars_state(i,1)=cars_str(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state(i,1)));
            cars_state(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state(i,3));
            cars_state(i,3)=num2str(cars_state_3-1);
        end
    %elseif  t==0  &&  cars_state_shu==0
%           elseif   cars_state_shu==0
%               fprintf()
%           car_t(i,1)=t_quan;
```

```
    end
end


 for  i=1:200
        cars_3=str2num(cars_state(i,3));
        car_state_2=str2num(cars_state(i,2));
        if  cars_3==0  &&  car_state_2==0
            car_t(i,1)=t_quan;
            cars_state(i,3)="-1";
            cars_zaiwei=cars_zaiwei-1;
        end
 end
    fprintf("猫炸啦猫炸啦,第%f秒,",t_quan);

    fprintf("第%f个猫咪转圈圈\n",cars_zaiwei);



end


%求总分

T=max(car_t);
for  i=1:200
    if  car_t(i,1)>858
        car_tt(i,1)=0;
    else
        car_tt(i,1)=250+858-car_t(i,1);
    end
end
Zongfeng=sum(car_tt);




%%%第三十秒状态提取区
```

```
clc;
clear  all;
load('cars.mat')
load('streets.mat')


streets_str=table2array(streets);
cars_str=table2array(cars);


%初始化
cars_state=[];
car_t=zeros(200,1);                %car_t 为车的全局时间
for  i=1:200
    [m,n]=find(streets_str==num2str(cars_str(i,3)));
    cars_str_shu=str2num(cars_str(i,2))-1;


cars_state=[cars_state;cars_str(i,3),streets_str(m,4),num2str(cars_str_shu)]
;
end
t_quan=0;       %全局时间
%%%
cars_zaiwei=200;     %200 个车都还在


while(cars_zaiwei>0)


for  i=1:200       %往前推动了 1s
    car_state_3=str2num(cars_state(i,3));
    if  car_state_3~=-1
```

```
        cars_state(i,2)=num2str(str2num(cars_state(i,2))-1);
        end


end




t_quan=t_quan+1;


%%寻找第 30 秒时各车的状态
if  t_quan==30
        fprintf("找到虎虎啦!!!!  \n");
        save('cars_state_30.mat','cars_state');
        fprintf("应该有了吧\n");
end


for  i=1:200
        t=str2num(cars_state(i,2));
        cars_state_shu=str2num(cars_state(i,3));
        street_name=cars_state(i,1);
        if  t==0  &&  cars_state_shu>0  &&  street_name~="wait"  %进入判决
            %  t_jia=Panjue(i,cars_state,t_quan);


             t_jia=0;
        street_name=cars_state(i,1);




        lukou_chuli=strsplit(street_name,'-');
        lukou_name=lukou_chuli(1,2);
        %搜索路口具体情况
```

```
    eval(['load("D:\桌面\省数模\A 题\代码\最优结果区
\',num2str(lukou_name),'_xinxi_zui.mat")']);

eval([num2str(lukou_name),'_xinxi=',num2str(lukou_name),'_xinxi_zui;'
]);
    %根据路口的交通灯数进行分类讨论

    eval(['size=numel(',num2str(lukou_name),'_xinxi)/2;']);    %如何
求 string 组的大小



    if  size>1          %路灯为 1 时
        t_lu=0;
        t_lu_ban=0;
        %要判断是否排队
        [m,n]=find(cars_str==cars_state(i,1));
        size_m=length(m);
        if  size_m==2   %则可能会排队
            for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);    %t_lu 为
总时间
                t_lu=t_lu+t;
            end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']); %t_lu_dangqian 为当前路的持续时间
            if  strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))  &&
cars_state(m(1,1),2)==cars_state(m(2,1),2)
                %则可能排队
```

```
                if  t_lu_dangqian==1
                    %在判断当前车是小还是大
                    m_max=max(m);
                    if  i==m_max
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    end
                end
            end
        elseif  size_m==3  %则可能会排队
            for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
总时间
                t_lu=t_lu+t;
            end

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']);  %t_lu_dangqian 为当前路的持续时间
            if
strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))||strcmp(cars_state(m
(1,1),1),cars_state(m(3,1),1))||strcmp(cars_state(m(2,1),1),cars_stat
e(m(3,1),1))
                %则可能排队
                %判断是两个还是三个
                if  strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))
&&  cars_state(m(1,1),2)==cars_state(m(2,1),2)
                    %则可能排队
                    if  t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m(1,1),m(2,1));
                        if  i==m_max
                            %则等待一个周期
```

```
                    t_jia=t_jia+t_lu;
                end
            end
        elseif
strcmp(cars_state(m(1,1),1),cars_state(m(3,1),1)) &&
cars_state(m(1,1),2)==cars_state(m(3,1),2)
            %则可能排队
            if t_lu_dangqian==1
                %在判断当前车是小还是大
                m_max=max(m(1,1),m(3,1));
                if i==m_max
                    %则等待一个周期
                    t_jia=t_jia+t_lu;
                end
            end
        elseif
strcmp(cars_state(m(2,1),1),cars_state(m(3,1),1)) &&
cars_state(m(2,1),2)==cars_state(m(3,1),2)
            %则可能排队
            if t_lu_dangqian==1
                %在判断当前车是小还是大
                m_max=max(m(2,1),m(3,1));
                if i==m_max
                    %则等待一个周期
                    t_jia=t_jia+t_lu;
                end
            end
        elseif
strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))&&strcmp(cars_state(m
(1,1),1),cars_state(m(3,1),1))&&strcmp(cars_state(m(2,1),1),cars_stat
e(m(3,1),1))&& cars_state(m(1,1),2)==cars_state(m(3,1),2)&&
cars_state(m(2,1),2)==cars_state(m(3,1),2)
            %则可能排队
            if t_lu_dangqian==1
                %在判断当前车是小还是大
                m_max=max(m);
```

```
                    m_min=min(m);
                    if i~=m_max && i~=m_min
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    elseif i==m_max
                        t_jia=t_jia+t_lu+t_lu;
                    end
                elseif t_lu_dangqian==2
                    m_max=max(m);
                    if i==m_max
                        t_jia=t_jia+t_lu;
                    end
                end
            end
        end


        for jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);    %t_lu 为
总时间
        t_lu=t_lu+t;
        end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']); %t_lu_dangqian 为当前路的持续时间
        if lu_cixv~=1
          for jjj=1:lu_cixv

eval(['ttt=str2num(',num2str(lukou_name),'_xinxi(jjj,2));']);
            t_lu_ban=t_lu_ban+ttt;              %t_lu_ban 为算上当前路后的
时间
        end
```

```
        t_lu_banjian=t_lu_ban-t_lu_dangqian;     %t_lu_banjian 为不算上当前
路的时间
        if  mod(t_quan,t_lu)<  t_lu_banjian
            t_jia=t_jia+(t_lu_banjian-mod(t_quan,t_lu));
        elseif   mod(t_quan,t_lu)  >  t_lu_ban
            t_jia=t_jia+(t_lu_banjian+t_lu-mod(t_quan,t_lu));
        end
        end
    end


%           if  t_jia<0
%           fprintf("11 找到虎虎啦!!!!!!  \n");
%           end


        if  t_jia==0
            %更新状态

            %fprintf("会有虎虎啦!!!!!!  \n");

            t=str2num(cars_str(i,2))+2;
            cars_state(i,1)=cars_str(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state(i,1)));
            cars_state(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state(i,3));
            cars_state(i,3)=num2str(cars_state_3-1);
        else
            cars_state(i,1)="wait";
            cars_state(i,2)=num2str(t_jia);
        end


    elseif  street_name=="wait"
        car_state_2=str2num(cars_state(i,2));
%           if  cars_state(i,1)=="wait"  &&   car_state_2<0
%               fprintf("22 找到虎虎啦!!!!  ");
%           end
```

```
        if  cars_state(i,1)=="wait"  &&  cars_state(i,2)=="0"
            %fprintf("虎虎呢？？？？\n");

            %更新状态
            t=str2num(cars_str(i,2))+2;
            cars_state(i,1)=cars_str(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state(i,1)));
            cars_state(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state(i,3));
            cars_state(i,3)=num2str(cars_state_3-1);
        end
    %elseif  t==0  &&  cars_state_shu==0
%           elseif   cars_state_shu==0
%               fprintf()
%           car_t(i,1)=t_quan;









    end
end


 for  i=1:200
        cars_3=str2num(cars_state(i,3));
        car_state_2=str2num(cars_state(i,2));
        if  cars_3==0  &&  car_state_2==0
            car_t(i,1)=t_quan;
            cars_state(i,3)="-1";
            cars_zaiwei=cars_zaiwei-1;
        end
 end
    fprintf("猫炸啦猫炸啦,第%f秒,",t_quan);

    fprintf("第%f个猫咪转圈圈\n",cars_zaiwei);



end
```

```
%求总分
T=max(car_t);
for  i=1:200
      if  car_t(i,1)>858
            car_tt(i,1)=0;
      else
            car_tt(i,1)=250+858-car_t(i,1);
      end
end
Zongfeng=sum(car_tt);




clc;
clear  all;


%%%设置参数

P=90;     %P 为循环次数

zDie_dai=[];            %  Die_dai 为迭代的过程展示值

%Zui_youzongfeng=0;        %最优总分的值

load('D:\桌面\省数模\A 题\代码\最优结果区\Zui_youzongfeng.mat');

load('xlukou_jishu.mat  ');
load('lushu.mat');
Pmax=P;

c1=2;     %学习因子

w=0.4;     %权重因子

load('xlukou.mat')




%     1.初始化策略（待改进）
```

```
%    绿灯持续时间的范围为[1，3]秒
for  i=1:2577
    eval([num2str(xlukou(1,i)),'_xinxi=[];']);


    for  j=1:3497
        aa=lushu(1,j);
        aaa=strsplit(aa,'-');
        bbb=num2str(aaa(1,2));
        if  strcmp(bbb,xlukou(1,i))

eval([num2str(xlukou(1,i)),'_xinxi=[',num2str(xlukou(1,i)),'_xinxi;aa
,round(rand(1,1)*2)+1];']);


        end
    %fprintf("猫猫持续生成解空间中,还剩%f组\n",2577-i);
    end
    %对各解空间的行进行了一次随机打乱次序
    eval(['rowrank  =
randperm(length(',num2str(xlukou(1,i)),'_xinxi(:))/2);']);  % 随机打
乱的数字，从 1~行数打乱
    eval([num2str(xlukou(1,i)),'_xinxi
=',num2str(xlukou(1,i)),'_xinxi(rowrank,  :);']);%%按照 rowrank 打乱
矩阵的行数


    fprintf("猫猫持续生成初始解空间中,还剩%f组\n",2577-i);
    end


%进入循环
while(P>0)
```

```
%%% 1.1 解空间优化区
for i=1:2577



%对各解空间的行进行了一次随机打乱次序
%eval(['rowrank =
randperm(length(',num2str(xlukou(1,i)),'_xinxi(:))/2);']); % 随机打乱的数字，从1~行数打乱
%eval([num2str(xlukou(1,i)),'_xinxi
=',num2str(xlukou(1,i)),'_xinxi(rowrank, :);']);%%按照rowrank打乱矩阵的行数



%单分子碰撞区

eval(['dengzongchang=length(',num2str(xlukou(1,i)),'_xinxi(:))/2;']);


    if dengzongchang~=1
    %则进行单分子碰撞
        deng=randperm(dengzongchang);
        lvdengcixv=deng(1,1:2);

eval(['lvdengzhongjian=',num2str(xlukou(1,i)),'_xinxi(lvdengcixv(1,1),:);']);

eval([num2str(xlukou(1,i)),'_xinxi(lvdengcixv(1,1),:)=',num2str(xlukou(1,i)),'_xinxi(lvdengcixv(1,2),:)']);

eval([num2str(xlukou(1,i)),'_xinxi(lvdengcixv(1,2),:)=lvdengzhongjian;']);
```

```
    end

    %持续时间区（粒子群）

eval(['lvdengzongchang=length(',num2str(xlukou(1,i)),'_xinxi(:))/2;']);
%初始化速度和位置
pop_v=zeros(lvdengzongchang,1);
pop_x=zeros(lvdengzongchang,1);
eval(['load("D:\桌面\省数模\A 题\代码\最优结果区
\',num2str(xlukou(1,i)),'_xinxi_zui.mat");']);
for  chixv=1:lvdengzongchang

eval(['pop_x(chixv,1)=str2num(',num2str(xlukou(1,i)),'_xinxi(chixv,2));']);
    %eval(['pop_x_jiu(chixv,1)=str2num(',num2str(xlukou(1,i)),'_xinxi(chixv,2));']);

eval(['pop_v_',num2str(xlukou(1,i)),'(chixv,1)=rand(1,1)*3;']);
end
%进行粒子的更新
for  chixv=1:lvdengzongchang

eval(['pop_v_',num2str(xlukou(1,i)),'(chixv,1)=w*pop_v_',num2str(xlukou(1,i)),'(chixv,1)+c1*rand(1,1)*3*(str2num(',num2str(xlukou(1,i)),'_xinxi_zui(chixv,2))-pop_x(chixv,1));']);
    %对 pop_v 进行限幅
    eval(['pop_v=pop_v_',num2str(xlukou(1,i)),'(chixv,1)']);
    if  pop_v<-3
    pop_v=mod(pop_v,-3);
    elseif  pop_v>3
    pop_v=mod(pop_v,3);
    end
    if  pop_v==0
        pop_v=rand(1,1)*3;
    end
```

```
eval([ 'pop_x(chixv,1)=round(pop_x(chixv,1)+pop_v_',num2str(xlukou(1,
i)),'(chixv,1));']);

    %对 pop_x 进行限幅度

    if  pop_x(chixv,1)<0  ||  pop_x(chixv,1)>5
        pop_x(chixv,1)=mod(abs(pop_x(chixv,1)),5);
    end
   if  pop_x(chixv,1)==0
        pop_x(chixv,1)=round(rand(1,1)*4)+1;
    end



eval([num2str(xlukou(1,i)),'_xinxi(chixv,2)=num2str(pop_x(chixv,1));'
]);
end

    %fprintf("猫猫持续生成解空间中,还剩%f 组\n",2577-i);

    end

%   1.2  算子改进（对策略进行变换）1.1 为 1 的优化版

%   2.求总分

load('cars.mat')
load('streets.mat')
```

```
streets_str=table2array(streets);
cars_str=table2array(cars);


%初始化
cars_state=[];
car_t=zeros(200,1);                %car_t 为车的全局时间
for  i=1:200
    [m,n]=find(streets_str==num2str(cars_str(i,3)));
    cars_str_shu=str2num(cars_str(i,2))-1;


cars_state=[cars_state;cars_str(i,3),streets_str(m,4),num2str(cars_str_shu)]
;
end
t_quan=0;       %全局时间
%%%
cars_zaiwei=200;     %200 个车都还在


while(cars_zaiwei>0)



for  i=1:200       %往前推动了 1s
    car_state_3=str2num(cars_state(i,3));
    if  car_state_3~=-1


    cars_state(i,2)=num2str(str2num(cars_state(i,2))-1);
    end


end
```

```
t_quan=t_quan+1;


for  i=1:200
     t=str2num(cars_state(i,2));
     cars_state_shu=str2num(cars_state(i,3));
     street_name=cars_state(i,1);
     if  t==0  &&  cars_state_shu>0  &&  street_name~="wait"  %进入判决




          t_jia=0;
     street_name=cars_state(i,1);




     lukou_chuli=strsplit(street_name,'-');
     lukou_name=lukou_chuli(1,2);
     %搜索路口具体情况


     %eval(['load("D:\桌面\省数模\A 题\代码\路口的具体路数数组群
\',num2str(lukou_name),'_xinxi.mat")']);


     %根据路口的交通灯数进行分类讨论

     eval(['size=numel(',num2str(lukou_name),'_xinxi)/2;']);     %如何
求 string 组的大小


     if  size>1          %路灯为 1 时
          t_lu=0;
          t_lu_ban=0;
```

```
%要判断是否排队
[m,n]=find(cars_str==cars_state(i,1));
size_m=length(m);
if  size_m==2   %则可能会排队
    for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
```

总时间

```
            t_lu=t_lu+t;
        end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']);  %t_lu_dangqian 为当前路的持续时间
        if  strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))  &&
cars_state(m(1,1),2)==cars_state(m(2,1),2)
            %则可能排队
            if  t_lu_dangqian==1
                %在判断当前车是小还是大
                m_max=max(m);
                if  i==m_max
                    %则等待一个周期
                    t_jia=t_jia+t_lu;
                end
            end
        end
    elseif  size_m==3  %则可能会排队
        for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
```

总时间

```
                t_lu=t_lu+t;
            end

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']); %t_lu_dangqian 为当前路的持续时间

            if
strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))||strcmp(cars_state(m
(1,1),1),cars_state(m(3,1),1))||strcmp(cars_state(m(2,1),1),cars_stat
e(m(3,1),1))

                %则可能排队

                %判断是两个还是三个

                if  strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))
&&  cars_state(m(1,1),2)==cars_state(m(2,1),2)

                    %则可能排队

                    if  t_lu_dangqian==1

                        %在判断当前车是小还是大

                        m_max=max(m(1,1),m(2,1));
                        if  i==m_max

                            %则等待一个周期

                            t_jia=t_jia+t_lu;
                        end
                    end
                elseif
strcmp(cars_state(m(1,1),1),cars_state(m(3,1),1))  &&
cars_state(m(1,1),2)==cars_state(m(3,1),2)

                    %则可能排队

                    if  t_lu_dangqian==1

                        %在判断当前车是小还是大

                        m_max=max(m(1,1),m(3,1));
                        if  i==m_max

                            %则等待一个周期

                            t_jia=t_jia+t_lu;
                        end
                    end
                elseif
```

```
strcmp(cars_state(m(2,1),1),cars_state(m(3,1),1)) &&
cars_state(m(2,1),2)==cars_state(m(3,1),2)
                    %则可能排队
                    if t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m(2,1),m(3,1));
                        if i==m_max
                            %则等待一个周期
                            t_jia=t_jia+t_lu;
                        end
                    end
                elseif
strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))&&strcmp(cars_state(m
(1,1),1),cars_state(m(3,1),1))&&strcmp(cars_state(m(2,1),1),cars_stat
e(m(3,1),1))&& cars_state(m(1,1),2)==cars_state(m(3,1),2)&&
cars_state(m(2,1),2)==cars_state(m(3,1),2)
                    %则可能排队
                    if t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m);
                        m_min=min(m);
                        if i~=m_max && i~=m_min
                            %则等待一个周期
                            t_jia=t_jia+t_lu;
                        elseif i==m_max
                            t_jia=t_jia+t_lu+t_lu;
                        end
                    elseif t_lu_dangqian==2
                        m_max=max(m);
                        if i==m_max
                            t_jia=t_jia+t_lu;
                        end
                    end
                end
            end
        end
```

```
        for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
```

总时间

```
        t_lu=t_lu+t;
        end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
```

%lu_cixv 为路名的顺序

```
eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']);  %t_lu_dangqian 为当前路的持续时间
        if  lu_cixv~=1
          for  jjj=1:lu_cixv

eval(['ttt=str2num(',num2str(lukou_name),'_xinxi(jjj,2));']);
              t_lu_ban=t_lu_ban+ttt;                %t_lu_ban 为算上当前路后的
```

时间

```
        end
        t_lu_banjian=t_lu_ban-t_lu_dangqian;     %t_lu_banjian 为不算上当前
```

路的时间

```
        if  mod(t_quan,t_lu)<  t_lu_banjian
            t_jia=t_jia+(t_lu_banjian-mod(t_quan,t_lu));
        elseif   mod(t_quan,t_lu)  >  t_lu_ban
            t_jia=t_jia+(t_lu_banjian+t_lu-mod(t_quan,t_lu));
        end
        end
    end


%           if  t_jia<0
%           fprintf("11 找到虎虎啦!!!!!!  \n");
%           end
```

```
        if  t_jia==0
            %更新状态

            %fprintf("会有虎虎啦!!!!!! \n");

            t=str2num(cars_str(i,2))+2;
            cars_state(i,1)=cars_str(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state(i,1)));
            cars_state(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state(i,3));
            cars_state(i,3)=num2str(cars_state_3-1);
        else
            cars_state(i,1)="wait";
            cars_state(i,2)=num2str(t_jia);
        end


    elseif  street_name=="wait"
        car_state_2=str2num(cars_state(i,2));
        if  cars_state(i,1)=="wait"  &&   car_state_2<0
            fprintf("22 找到虎虎啦!!!! ");

        end


        if  cars_state(i,1)=="wait"  &&  cars_state(i,2)=="0"
            %fprintf("虎虎呢？？？？\n");

            %更新状态

            t=str2num(cars_str(i,2))+2;
            cars_state(i,1)=cars_str(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state(i,1)));
            cars_state(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state(i,3));
            cars_state(i,3)=num2str(cars_state_3-1);
        end
    %elseif  t==0  &&  cars_state_shu==0
    %         elseif   cars_state_shu==0
    %              fprintf()
    %         car_t(i,1)=t_quan;
```

```
    end
end


 for  i=1:200
        cars_3=str2num(cars_state(i,3));
        car_state_2=str2num(cars_state(i,2));
        if  cars_3==0  &&  car_state_2==0
            car_t(i,1)=t_quan;
            cars_state(i,3)="-1";
            cars_zaiwei=cars_zaiwei-1;
        end
 end
    fprintf("猫炸啦猫炸啦,第%f秒,",t_quan);

    fprintf("第%f个猫咪转圈圈,",cars_zaiwei);

    fprintf("第%f轮迭代\n",Pmax-P+1);



end


%求总分

T=max(car_t);
for  i=1:200
    if  car_t(i,1)>858
        car_tt(i,1)=0;
    else
        car_tt(i,1)=250+858-car_t(i,1);
    end
end
Zongfeng=sum(car_tt);          %Zongfeng   即为当前解空间的总分值




%  %存一下迭代的过程
```

```
% %Die_dai=[];          %  Die_dai 为迭代的过程展示值
% Die_dai=[Die_dai;P,Zongfeng];



%   3.判断，存优
if Zongfeng > Zui_youzongfeng
    Zui_youzongfeng=Zongfeng;

    %存到最优结果区

    save('D:\桌面\省数模\A 题\代码\最优结果区
\Zui_youzongfeng.mat','Zui_youzongfeng');
    for i=1:2577
eval([num2str(xlukou(1,i)),'_xinxi_zui=',num2str(xlukou(1,i)),'_xinxi'
]);
        eval(['save("D:\桌面\省数模\A 题\代码\最优结果区
\',num2str(xlukou(1,i)),'_xinxi_zui.mat","',num2str(xlukou(1,i)),'_x
inxi_zui")']);


    end
end


for i=1:2577
%单分子分解

%单分子分解
```

```
eval([ 'zdengzongchang=length(',num2str(xlukou(1,i)),'_xinxi(:))/2;'])
;


if  zdengzongchang>3
    z_suiji=round(rand(1,1)*(zdengzongchang-2)+2);
    if  z_suiji>zdengzongchang
        z_suiji=zdengzongchang;
    end

eval([ 'kouzi_qian=',num2str(xlukou(1,i)),'_xinxi(1:z_suiji,:);'])

    %eval([ 'kouzi_qian=h_xinxi(1:z_suiji,:);'])


    eval([ 'rowrank  =  randperm(length(kouzi_qian(:))/2);'])  % 随机打
乱的数字，从 1~行数打乱

    eval([ 'kouzi_qian=kouzi_qian(rowrank,  :);'])%%按照 rowrank 打乱矩
阵的行数
    for  jjjjj=1:z_suiji
        %zh_xinxi(i,:)=kouzi_qian(i,:);

eval([num2str(xlukou(1,i)),'_xinxi(jjjjj,1)=kouzi_qian(jjjjj,1);'])

eval([num2str(xlukou(1,i)),'_xinxi(jjjjj,2)=kouzi_qian(jjjjj,2);'])
    end
    clear  kouzi_qian;
end


    %持续时间区（粒子群）


eval([ 'lvdengzongchang=length(',num2str(xlukou(1,i)),'_xinxi(:))/2;'])
;

%初始化速度和位置

pop_v=zeros(lvdengzongchang,1);
pop_x=zeros(lvdengzongchang,1);
```

```
eval(['load("D:\桌面\省数模\A 题\代码\最优结果区
\',num2str(xlukou(1,i)),'_xinxi_zui.mat");']);
for  chixv=1:lvdengzongchang

eval(['pop_x(chixv,1)=str2num(',num2str(xlukou(1,i)),'_xinxi(chixv,2)
);']);
    %eval(['pop_x_jiu(chixv,1)=str2num(',num2str(xlukou(1,i)),'_xinxi
(chixv,2));']);

eval(['pop_v_',num2str(xlukou(1,i)),'(chixv,1)=rand(1,1)*3;']);
end
%进行粒子的更新
for  chixv=1:lvdengzongchang

eval(['pop_v_',num2str(xlukou(1,i)),'(chixv,1)=w*pop_v_',num2str(xl
ukou(1,i)),'(chixv,1)+c1*rand(1,1)*3*(str2num(',num2str(xlukou(1,i)
),'_xinxi_zui(chixv,2))-pop_x(chixv,1));']);
    %对 pop_v 进行限幅
    eval(['pop_v=pop_v_',num2str(xlukou(1,i)),'(chixv,1)']);
    if  pop_v<-3
    pop_v=mod(pop_v,-3);
    elseif  pop_v>3
    pop_v=mod(pop_v,3);
    end
    if  pop_v==0
        pop_v=rand(1,1)*3;
    end


eval(['pop_x(chixv,1)=round(pop_x(chixv,1)+pop_v_',num2str(xlukou(1,
i)),'(chixv,1));']);
    %对 pop_x 进行限幅度
    if  pop_x(chixv,1)<0  ||  pop_x(chixv,1)>5
        pop_x(chixv,1)=mod(abs(pop_x(chixv,1)),5);
    end
   if  pop_x(chixv,1)==0
        pop_x(chixv,1)=round(rand(1,1)*4)+1;
    end
```

```
eval([num2str(xlukou(1,i)),'_xinxi(chixv,2)=num2str(pop_x(chixv,1));'
]);
end
end
```

```
    %fprintf("猫猫持续生成解空间中,还剩%f组\n",2577-i);
```

```
%    1.2  算子改进（对策略进行变换）1.1 为 1 的优化版
```

```
%    2.求总分
load('cars.mat')
load('streets.mat')

streets_str=table2array(streets);
cars_str=table2array(cars);

%初始化
cars_state=[];
car_t=zeros(200,1);                %car_t 为车的全局时间
for  i=1:200
    [m,n]=find(streets_str==num2str(cars_str(i,3)));
```

```
        cars_str_shu=str2num(cars_str(i,2))-1;


cars_state=[cars_state;cars_str(i,3),streets_str(m,4),num2str(cars_str_shu)]
;
end
t_quan=0;       %全局时间
%%%
cars_zaiwei=200;     %200 个车都还在


while(cars_zaiwei>0)




for  i=1:200       %往前推动了 1s
        car_state_3=str2num(cars_state(i,3));
        if  car_state_3~=-1


        cars_state(i,2)=num2str(str2num(cars_state(i,2))-1);
        end


end




t_quan=t_quan+1;


for  i=1:200
        t=str2num(cars_state(i,2));
        cars_state_shu=str2num(cars_state(i,3));
        street_name=cars_state(i,1);
        if  t==0  &&  cars_state_shu>0  &&  street_name~="wait"  %进入判决
```

```
        t_jia=0;
    street_name=cars_state(i,1);



    lukou_chuli=strsplit(street_name,'-');
    lukou_name=lukou_chuli(1,2);
    %搜索路口具体情况


    %eval(['load("D:\桌面\省数模\A 题\代码\路口的具体路数数组群
\',num2str(lukou_name),'_xinxi.mat")']);


    %根据路口的交通灯数进行分类讨论

    eval(['size=numel(',num2str(lukou_name),'_xinxi)/2;']);     %如何
求 string 组的大小


    if  size>1          %路灯为 1 时
        t_lu=0;
        t_lu_ban=0;
        %要判断是否排队
        [m,n]=find(cars_str==cars_state(i,1));
        size_m=length(m);
        if  size_m==2   %则可能会排队
            for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
总时间
```

```
                t_lu=t_lu+t;
            end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']); %t_lu_dangqian 为当前路的持续时间

            if strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1)) &&
cars_state(m(1,1),2)==cars_state(m(2,1),2)
                %则可能排队
                if t_lu_dangqian==1
                    %在判断当前车是小还是大
                    m_max=max(m);
                    if i==m_max
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    end
                end
            end
        elseif size_m==3 %则可能会排队
            for jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);    %t_lu 为
总时间
                t_lu=t_lu+t;
            end

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']); %t_lu_dangqian 为当前路的持续时间

            if
strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))||strcmp(cars_state(m
(1,1),1),cars_state(m(3,1),1))||strcmp(cars_state(m(2,1),1),cars_stat
e(m(3,1),1))
                %则可能排队
```

```
                    %判断是两个还是三个
                    if  strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))
&&  cars_state(m(1,1),2)==cars_state(m(2,1),2)
                        %则可能排队
                        if  t_lu_dangqian==1
                            %在判断当前车是小还是大
                            m_max=max(m(1,1),m(2,1));
                            if  i==m_max
                                %则等待一个周期
                                t_jia=t_jia+t_lu;
                            end
                        end
                    elseif
strcmp(cars_state(m(1,1),1),cars_state(m(3,1),1))  &&
cars_state(m(1,1),2)==cars_state(m(3,1),2)
                        %则可能排队
                        if  t_lu_dangqian==1
                            %在判断当前车是小还是大
                            m_max=max(m(1,1),m(3,1));
                            if  i==m_max
                                %则等待一个周期
                                t_jia=t_jia+t_lu;
                            end
                        end
                    elseif
strcmp(cars_state(m(2,1),1),cars_state(m(3,1),1))  &&
cars_state(m(2,1),2)==cars_state(m(3,1),2)
                        %则可能排队
                        if  t_lu_dangqian==1
                            %在判断当前车是小还是大
                            m_max=max(m(2,1),m(3,1));
                            if  i==m_max
                                %则等待一个周期
                                t_jia=t_jia+t_lu;
```

```
                    end
                end
            elseif
strcmp(cars_state(m(1,1),1),cars_state(m(2,1),1))&&strcmp(cars_state(m
(1,1),1),cars_state(m(3,1),1))&&strcmp(cars_state(m(2,1),1),cars_stat
e(m(3,1),1))&&  cars_state(m(1,1),2)==cars_state(m(3,1),2)&&
cars_state(m(2,1),2)==cars_state(m(3,1),2)
                %则可能排队
                if  t_lu_dangqian==1
                    %在判断当前车是小还是大
                    m_max=max(m);
                    m_min=min(m);
                    if  i~=m_max  &&  i~=m_min
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    elseif  i==m_max
                        t_jia=t_jia+t_lu+t_lu;
                    end
                elseif  t_lu_dangqian==2
                    m_max=max(m);
                    if  i==m_max
                        t_jia=t_jia+t_lu;
                    end
                end
            end
        end
    end


    for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
总时间
    t_lu=t_lu+t;
    end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序
```

```
eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']); %t_lu_dangqian 为当前路的持续时间
        if lu_cixv~=1
          for jjj=1:lu_cixv

eval(['ttt=str2num(',num2str(lukou_name),'_xinxi(jjj,2));']);
            t_lu_ban=t_lu_ban+ttt;          %t_lu_ban 为算上当前路后的
时间
        end
        t_lu_banjian=t_lu_ban-t_lu_dangqian;    %t_lu_banjian 为不算上当前
路的时间
        if mod(t_quan,t_lu)< t_lu_banjian
            t_jia=t_jia+(t_lu_banjian-mod(t_quan,t_lu));
        elseif  mod(t_quan,t_lu) > t_lu_ban
            t_jia=t_jia+(t_lu_banjian+t_lu-mod(t_quan,t_lu));
        end
        end
    end


%           if t_jia<0
%           fprintf("11 找到虎虎啦!!!!!! \n");
%           end


        if t_jia==0
            %更新状态

            %fprintf("会有虎虎啦!!!!!! \n");

            t=str2num(cars_str(i,2))+2;
            cars_state(i,1)=cars_str(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state(i,1)));
            cars_state(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state(i,3));
            cars_state(i,3)=num2str(cars_state_3-1);
        else
```

```
            cars_state(i,1)="wait";
            cars_state(i,2)=num2str(t_jia);
        end


    elseif street_name=="wait"
        car_state_2=str2num(cars_state(i,2));
        if cars_state(i,1)=="wait" && car_state_2<0
            fprintf("22 找到虎虎啦!!!! ");
        end


        if cars_state(i,1)=="wait" && cars_state(i,2)=="0"
            %fprintf("虎虎呢？？？？\n");

            %更新状态
            t=str2num(cars_str(i,2))+2;
            cars_state(i,1)=cars_str(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state(i,1)));
            cars_state(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state(i,3));
            cars_state(i,3)=num2str(cars_state_3-1);
        end
    %elseif t==0 && cars_state_shu==0
%           elseif  cars_state_shu==0
%               fprintf()
%           car_t(i,1)=t_quan;




    end
end


 for i=1:200
        cars_3=str2num(cars_state(i,3));
        car_state_2=str2num(cars_state(i,2));
        if cars_3==0 && car_state_2==0
            car_t(i,1)=t_quan;
```

```
                cars_state(i,3)="-1";
                cars_zaiwei=cars_zaiwei-1;
            end
 end
    fprintf("猫炸啦猫炸啦,第%f秒,",t_quan);

    fprintf("第%f个猫咪转圈圈,",cars_zaiwei);

    fprintf("第%f轮迭代\n",Pmax-P+1);


end


%求总分

T=max(car_t);
for  i=1:200
    if  car_t(i,1)>858
        car_tt(i,1)=0;
    else
        car_tt(i,1)=250+858-car_t(i,1);
    end
end
Zongfeng=sum(car_tt);          %Zongfeng  即为当前解空间的总分值




%  %存一下迭代的过程

%  %Die_dai=[];          %  Die_dai 为迭代的过程展示值

%  Die_dai=[Die_dai;P,Zongfeng];




%    3.判断，存优

if  Zongfeng  >  Zui_youzongfeng
    Zui_youzongfeng=Zongfeng;
```

```
    %存到最优结果区


    save('D:\桌面\省数模\A 题\代码\最优结果区
\Zui_youzongfeng.mat','Zui_youzongfeng');
    for  i=1:2577

eval([num2str(xlukou(1,i)),'_xinxi_zui=',num2str(xlukou(1,i)),'_xinxi'
]);

        eval(['save("D:\桌面\省数模\A 题\代码\最优结果区
\',num2str(xlukou(1,i)),'_xinxi_zui.mat","',num2str(xlukou(1,i)),'_x
inxi_zui")']);




    end
end




P=P-1;




%存一下迭代的过程

%Die_dai=[];          %  Die_dai 为迭代的过程展示值

zDie_dai=[zDie_dai;P,Zui_youzongfeng];
```

```
end

fprintf("最终的最高得分为%f\n",Zui_youzongfeng);

fprintf("完结撒花\n");
```

问题二代码:

```
%%% 根据给的 newpath 生成新的路口和路线数的总表
clc;
clear all;
load('newpath.mat');
load('xlukou.mat');
uu=length(newpath(:));
for i=1:2577
    lukou_name=xlukou(1,i);
    eval(['load("D:\桌面\省数模\A 题\问题 2 代码\路口的具体路数数组
群\',num2str(lukou_name),'_xinxi.mat")']);
end

for i=1:uu
    zz=newpath(1,i);
    zzz=strsplit(zz,'-');
    zzzz=zzz(1,2);
    yyy=find(xlukou==zzzz);
    if isempty(yyy)
        %创建新的 lukou_xinxi
        xlukou=[xlukou,zzzz];
        eval(['num2str(zzzz),'_xinxi=[zz,round(rand(1,1)*2)+1];']);
    else
```

```
        %在原路口添加新的行
        lukou_name=xlukou(1,yyy);

eval([num2str(lukou_name),'_xinxi=[',num2str(lukou_name),'_xinxi;zz,round(rand(1,1)*2)+1];']);
    end



end


lukou_length=length(xlukou(:));
for  i=1:lukou_length
    lukou_name=xlukou(1,i);
    eval(['save("D:\桌面\省数模\A 题\问题 2 代码\更新的路口状态
\',num2str(lukou_name),'_xinxi.mat","',num2str(xlukou(1,i)),'_xinxi"
)']);
    fprintf("在学啦在学啦，还剩%f 个\n",lukou_length-i);
end


save('D:\桌面\省数模\A 题\问题 2 代码\xlukou_gengxin.mat','xlukou');



%%%产生第一个最优结果并保存
lukou_length=length(xlukou(:));
for  i=1:lukou_length
    lukou_name=xlukou(1,i);

eval([num2str(lukou_name),'_xinxi_zui=',num2str(lukou_name),'_xinxi;'
]);
    eval(['save("D:\桌面\省数模\A 题\问题 2 代码\最优结果区
\',num2str(lukou_name),'_xinxi_zui.mat","',num2str(lukou_name),'_xi
nxi_zui")']);
```

```
    fprintf("在学啦在学啦，还剩%f个\n",lukou_length-i);
end



clc;
clear  all;
load('cars_state_new.mat');
load('cars_str_new.mat');
cars_state_new=cars_state;
cars_str_new=cars_str;


%%%设置参数

P=100;     %P 为循环次数

zDie_dai=[];            %  Die_dai 为迭代的过程展示值

%  Zui_youzongfeng=0;         %最优总分的值

%  save('D:\桌面\省数模\A 题\问题 2 代码\最优结果区
\Zui_youzongfeng.mat','Zui_youzongfeng');


load('D:\桌面\省数模\A 题\问题 2 代码\最优结果区
\Zui_youzongfeng.mat');
load('xlukou_jishu.mat  ');
load('lushu.mat');
Pmax=P;
c1=2;     %学习因子

w=0.4;     %权重因子

%load('xlukou.mat')
load('xlukou_gengxin.mat')


lukou_length_gengxin=length(xlukou);
%     1.初始化策略（待改进）
```

```
%    绿灯持续时间的范围为[1，3]秒
lukou_length_gengxin=length(xlukou);
for  i=1:lukou_length_gengxin
     lukou_name=xlukou(1,i);
     eval(['load("D:\桌面\省数模\A 题\问题 2 代码\更新的路口状态
\',num2str(lukou_name),'_xinxi.mat")']);
end


%进入循环
while(P>0)


     %%% 1.1 解空间优化区
     for  i=1:lukou_length_gengxin




     %对各解空间的行进行了一次随机打乱次序
     %eval(['rowrank  =
randperm(length(',num2str(xlukou(1,i)),'_xinxi(:))/2);']); % 随机打
乱的数字，从 1~行数打乱
     %eval([num2str(xlukou(1,i)),'_xinxi
=',num2str(xlukou(1,i)),'_xinxi(rowrank, :);']);%%按照 rowrank 打乱
矩阵的行数


     %单分子碰撞区

eval(['dengzongchang=length(',num2str(xlukou(1,i)),'_xinxi(:))/2;'])
;
```

```
if  dengzongchang~=1
%则进行单分子碰撞
    deng=randperm(dengzongchang);
    lvdengcixv=deng(1,1:2);

eval(['lvdengzhongjian=',num2str(xlukou(1,i)),'_xinxi(lvdengcixv(1,1),:);']);

eval([num2str(xlukou(1,i)),'_xinxi(lvdengcixv(1,1),:)=',num2str(xlukou(1,i)),'_xinxi(lvdengcixv(1,2),:)']);

eval([num2str(xlukou(1,i)),'_xinxi(lvdengcixv(1,2),:)=lvdengzhongjian;']);
    end


%持续时间区（粒子群）

eval(['lvdengzongchang=length(',num2str(xlukou(1,i)),'_xinxi(:))/2;']);
%初始化速度和位置
pop_v=zeros(lvdengzongchang,1);
pop_x=zeros(lvdengzongchang,1);
eval(['load("D:\桌面\省数模\A 题\问题 2 代码\最优结果区
\',num2str(xlukou(1,i)),'_xinxi_zui.mat");']);
for  chixv=1:lvdengzongchang

eval(['pop_x(chixv,1)=str2num(',num2str(xlukou(1,i)),'_xinxi(chixv,2));']);
    %eval(['pop_x_jiu(chixv,1)=str2num(',num2str(xlukou(1,i)),'_xinxi(chixv,2));']);

eval(['pop_v_',num2str(xlukou(1,i)),'(chixv,1)=rand(1,1)*3;']);
end
%进行粒子的更新
for  chixv=1:lvdengzongchang
```

```
eval([ 'pop_v_',num2str(xlukou(1,i)),'(chixv,1)=w*pop_v_',num2str(xlukou(1,i)),'(chixv,1)+c1*rand(1,1)*3*(str2num(',num2str(xlukou(1,i)),'_xinxi_zui(chixv,2))-pop_x(chixv,1));']);

    %对 pop_v 进行限幅

    eval([ 'pop_v=pop_v_',num2str(xlukou(1,i)),'(chixv,1)']);
    if  pop_v<-3
    pop_v=mod(pop_v,-3);
    elseif  pop_v>3
    pop_v=mod(pop_v,3);
    end
    if  pop_v==0
        pop_v=rand(1,1)*3;
    end



eval([ 'pop_x(chixv,1)=round(pop_x(chixv,1)+pop_v_',num2str(xlukou(1,i)),'(chixv,1));']);

    %对 pop_x 进行限幅度

    if  pop_x(chixv,1)<0  ||  pop_x(chixv,1)>5
        pop_x(chixv,1)=mod(abs(pop_x(chixv,1)),5);
    end
   if  pop_x(chixv,1)==0
        pop_x(chixv,1)=round(rand(1,1)*4)+1;
    end



eval([num2str(xlukou(1,i)),'_xinxi(chixv,2)=num2str(pop_x(chixv,1));']);
end




    %fprintf("猫猫持续生成解空间中,还剩%f 组\n",2577-i);

    end
```

```
%    1.2  算子改进（对策略进行变换）1.1 为 1 的优化版




%    2.求总分
load('cars.mat')
load('streets.mat')


streets_str=table2array(streets);
%cars_str=table2array(cars);


%初始化
%cars_state=[];
load('cars_state_new.mat');
cars_state_new=cars_state;


car_t=zeros(200,1);                %car_t 为车的全局时间
%  for  i=1:200
%        [m,n]=find(streets_str==num2str(cars_str_new(i,3)));
%        cars_str_shu=str2num(cars_str_new(i,2))-1;
%
%        %cars_state=[cars_state;cars_str_new(i,3),streets_str(m,4),num2str(c
ars_str_shu)];
%  end
t_quan=0;       %全局时间
%%%
cars_zaiwei=200;     %200 个车都还在


while(cars_zaiwei>0)
```

```
% for i=1:200      %往前推动了 1s
%       car_state_3=str2num(cars_state_new(i,3));
%       if car_state_3~=-1
%
%       cars_state_new(i,2)=num2str(str2num(cars_state_new(i,2))-1);
%       end
%
% end
%
%
%
% t_quan=t_quan+1;


for i=1:200
    t=str2num(cars_state_new(i,2));
    cars_state_shu=str2num(cars_state_new(i,3));
    street_name=cars_state_new(i,1);
    if t==0 && cars_state_shu>0 && street_name~="wait" %进入判决




        t_jia=0;
    street_name=cars_state_new(i,1);




    lukou_chuli=strsplit(street_name,'-');
    lukou_name=lukou_chuli(1,2);
    %搜索路口具体情况



    %eval(['load("D:\桌面\省数模\A 题\代码\路口的具体路数数组群
\',num2str(lukou_name),'_xinxi.mat")']);
```

```
%根据路口的交通灯数进行分类讨论
eval(['size=numel(',num2str(lukou_name),'_xinxi)/2;']);    %如何求 string 组的大小

if  size>1          %路灯为 1 时
    t_lu=0;
    t_lu_ban=0;
    %要判断是否排队
    [m,n]=find(cars_str_new==cars_state_new(i,1));
    size_m=length(m);
    if  size_m==2   %则可能会排队
        for  jj=1:size
eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);    %t_lu 为总时间
            t_lu=t_lu+t;
        end
eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序
eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2));']); %t_lu_dangqian 为当前路的持续时间
        if
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(2,1),1)) && cars_state_new(m(1,1),2)==cars_state_new(m(2,1),2)
            %则可能排队
            if  t_lu_dangqian==1
                %在判断当前车是小还是大
```

```
                    m_max=max(m);
                    if  i==m_max
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    end
                end
            end
        elseif  size_m==3  %则可能会排队
            for  jj=1:size
eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
总时间
                t_lu=t_lu+t;
            end
eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']);  %t_lu_dangqian 为当前路的持续时间
            if
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(2,1),1))||strcmp(ca
rs_state_new(m(1,1),1),cars_state_new(m(3,1),1))||strcmp(cars_state_n
ew(m(2,1),1),cars_state_new(m(3,1),1))
                %则可能排队
                %判断是两个还是三个
                if
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(2,1),1))  &&
cars_state_new(m(1,1),2)==cars_state_new(m(2,1),2)
                    %则可能排队
                    if  t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m(1,1),m(2,1));
                        if  i==m_max
                            %则等待一个周期
                            t_jia=t_jia+t_lu;
                        end
```

```
                end
            elseif
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(3,1),1)) &&
cars_state_new(m(1,1),2)==cars_state_new(m(3,1),2)
                %则可能排队
                if t_lu_dangqian==1
                    %在判断当前车是小还是大
                    m_max=max(m(1,1),m(3,1));
                    if i==m_max
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    end
                end
            elseif
strcmp(cars_state_new(m(2,1),1),cars_state_new(m(3,1),1)) &&
cars_state_new(m(2,1),2)==cars_state_new(m(3,1),2)
                %则可能排队
                if t_lu_dangqian==1
                    %在判断当前车是小还是大
                    m_max=max(m(2,1),m(3,1));
                    if i==m_max
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    end
                end
            elseif
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(2,1),1))&&strcmp(ca
rs_state_new(m(1,1),1),cars_state_new(m(3,1),1))&&strcmp(cars_state_n
ew(m(2,1),1),cars_state_new(m(3,1),1))&&
cars_state_new(m(1,1),2)==cars_state_new(m(3,1),2)&&
cars_state_new(m(2,1),2)==cars_state_new(m(3,1),2)
                %则可能排队
                if t_lu_dangqian==1
                    %在判断当前车是小还是大
                    m_max=max(m);
                    m_min=min(m);
```

```
                    if  i~=m_max  &&  i~=m_min
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    elseif  i==m_max
                        t_jia=t_jia+t_lu+t_lu;
                    end
                elseif  t_lu_dangqian==2
                    m_max=max(m);
                    if  i==m_max
                        t_jia=t_jia+t_lu;
                    end
                end
            end
        end
    end


    for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
总时间
    t_lu=t_lu+t;
    end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']);  %t_lu_dangqian 为当前路的持续时间
    if  lu_cixv~=1
      for  jjj=1:lu_cixv

eval(['ttt=str2num(',num2str(lukou_name),'_xinxi(jjj,2));']);
        t_lu_ban=t_lu_ban+ttt;                %t_lu_ban 为算上当前路后的
时间
    end
    t_lu_banjian=t_lu_ban-t_lu_dangqian;     %t_lu_banjian 为不算上当前
```

路的时间

```
        if  mod(t_quan,t_lu)<  t_lu_banjian
            t_jia=t_jia+(t_lu_banjian-mod(t_quan,t_lu));
        elseif   mod(t_quan,t_lu)  >  t_lu_ban
            t_jia=t_jia+(t_lu_banjian+t_lu-mod(t_quan,t_lu));
        end
        end
    end


        if  t_jia<0
        fprintf("11 找到虎虎啦!!!!!!  \n");
        end


        if  t_jia==0
            %更新状态

            %fprintf("会有虎虎啦!!!!!!  \n");
            t=str2num(cars_str_new(i,2))+2;
            cars_state_new(i,1)=cars_str_new(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state_new(i,1)));
            cars_state_new(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state_new(i,3));
            cars_state_new(i,3)=num2str(cars_state_3-1);
        else
            cars_state_new(i,1)="wait";
            cars_state_new(i,2)=num2str(t_jia);
        end


    elseif  street_name=="wait"
        car_state_2=str2num(cars_state_new(i,2));
        if  cars_state_new(i,1)=="wait"  &&   car_state_2<0
            fprintf("22 找到虎虎啦!!!!  ");
        end


        if  cars_state_new(i,1)=="wait"  &&  cars_state_new(i,2)=="0"
            %fprintf("虎虎呢？？？？\n");
```

```
            %更新状态
            t=str2num(cars_str_new(i,2))+2;
            cars_state_new(i,1)=cars_str_new(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state_new(i,1)));
            cars_state_new(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state_new(i,3));
            cars_state_new(i,3)=num2str(cars_state_3-1);
        end
    %elseif  t==0  &&  cars_state_shu==0
%           elseif   cars_state_shu==0
%               fprintf()
%           car_t(i,1)=t_quan;




    end
end


 for  i=1:200
        cars_3=str2num(cars_state_new(i,3));
        car_state_2=str2num(cars_state_new(i,2));
        if  cars_3==0  &&  car_state_2==0
            car_t(i,1)=t_quan;
            cars_state_new(i,3)="-1";
            cars_zaiwei=cars_zaiwei-1;
        end
 end
    fprintf("猫炸啦猫炸啦,第%f秒,",t_quan);

    fprintf("第%f个猫咪转圈圈,",cars_zaiwei);

    fprintf("第%f轮迭代\n",Pmax-P+1);



    for  i=1:200      %往前推动了 1s
    car_state_3=str2num(cars_state_new(i,3));
    if  car_state_3~=-1
```

```
        cars_state_new(i,2)=num2str(str2num(cars_state_new(i,2))-1);
        end
        end
        t_quan=t_quan+1;


end
```

```
%求总分
T=max(car_t);
for  i=1:200
        if  car_t(i,1)>858
                car_tt(i,1)=0;
        else
                car_tt(i,1)=250+858-car_t(i,1);
        end
end
Zongfeng=sum(car_tt);              %Zongfeng   即为当前解空间的总分值




%  %存一下迭代的过程
%  %Die_dai=[];              %   Die_dai 为迭代的过程展示值
%  Die_dai=[Die_dai;P,Zongfeng];




%     3.判断，存优
if  Zongfeng  >  Zui_youzongfeng
        Zui_youzongfeng=Zongfeng;
```

```
    %存到最优结果区


    save('D:\桌面\省数模\A 题\问题 2 代码\最优结果区
\Zui_youzongfeng.mat','Zui_youzongfeng');
    for  i=1:lukou_length_gengxin

eval([num2str(xlukou(1,i)),'_xinxi_zui=',num2str(xlukou(1,i)),'_xinxi'
]);

        eval(['save("D:\桌面\省数模\A 题\问题 2 代码\最优结果区
\',num2str(xlukou(1,i)),'_xinxi_zui.mat","',num2str(xlukou(1,i)),'_x
inxi_zui")']);




    end
end



for  i=1:lukou_length_gengxin
%单分子分解

%单分子分解


eval(['zdengzongchang=length(',num2str(xlukou(1,i)),'_xinxi(:))/2;']
);


if  zdengzongchang>3
    z_suiji=round(rand(1,1)*(zdengzongchang-2)+2);
    if  z_suiji>zdengzongchang
        z_suiji=zdengzongchang;
    end
```

```
eval(['kouzi_qian=',num2str(xlukou(1,i)),'_xinxi(1:z_suiji,:);']);

    %eval(['kouzi_qian=h_xinxi(1:z_suiji,:);']);

    eval(['rowrank = randperm(length(kouzi_qian(:))/2);']); % 随机打乱的数字，从 1~行数打乱

    eval(['kouzi_qian=kouzi_qian(rowrank, :);']);%%按照 rowrank 打乱矩阵的行数

    for jjjjj=1:z_suiji
        %zh_xinxi(i,:)=kouzi_qian(i,:);

eval([num2str(xlukou(1,i)),'_xinxi(jjjjj,1)=kouzi_qian(jjjjj,1);']);

eval([num2str(xlukou(1,i)),'_xinxi(jjjjj,2)=kouzi_qian(jjjjj,2);']);
    end
    clear kouzi_qian;
end

    %持续时间区（粒子群）

eval(['lvdengzongchang=length(',num2str(xlukou(1,i)),'_xinxi(:))/2;']);

%初始化速度和位置

pop_v=zeros(lvdengzongchang,1);
pop_x=zeros(lvdengzongchang,1);

eval(['load("D:\桌面\省数模\A 题\问题 2 代码\最优结果区

\',num2str(xlukou(1,i)),'_xinxi_zui.mat");']);
for chixv=1:lvdengzongchang

eval(['pop_x(chixv,1)=str2num(',num2str(xlukou(1,i)),'_xinxi(chixv,2));']);
    %eval(['pop_x_jiu(chixv,1)=str2num(',num2str(xlukou(1,i)),'_xinxi(chixv,2));']);
```

```
eval([ 'pop_v_',num2str(xlukou(1,i)),'(chixv,1)=rand(1,1)*3;']);
end
%进行粒子的更新
for  chixv=1:lvdengzongchang

eval([ 'pop_v_',num2str(xlukou(1,i)),'(chixv,1)=w*pop_v_',num2str(xl
ukou(1,i)),'(chixv,1)+c1*rand(1,1)*3*(str2num(',num2str(xlukou(1,i)
),'_xinxi_zui(chixv,2))-pop_x(chixv,1));']);

    %对 pop_v 进行限幅

    eval([ 'pop_v=pop_v_',num2str(xlukou(1,i)),'(chixv,1)']);
    if  pop_v<-3
    pop_v=mod(pop_v,-3);
    elseif  pop_v>3
    pop_v=mod(pop_v,3);
    end
    if  pop_v==0
        pop_v=rand(1,1)*3;
    end



eval([ 'pop_x(chixv,1)=round(pop_x(chixv,1)+pop_v_',num2str(xlukou(1,
i)),'(chixv,1));']);

    %对 pop_x 进行限幅度

    if  pop_x(chixv,1)<0  ||  pop_x(chixv,1)>5
        pop_x(chixv,1)=mod(abs(pop_x(chixv,1)),5);
    end
   if  pop_x(chixv,1)==0
        pop_x(chixv,1)=round(rand(1,1)*4)+1;
    end



eval([num2str(xlukou(1,i)),'_xinxi(chixv,2)=num2str(pop_x(chixv,1));'
]);
end
end
```

```
%fprintf("猫猫持续生成解空间中,还剩%f组\n",2577-i);
```

```
%    1.2  算子改进（对策略进行变换）1.1为1的优化版
```

```
%    2.求总分
load('cars.mat')
load('streets.mat')

streets_str=table2array(streets);
%cars_str=table2array(cars);

%初始化
%cars_state_new=[];
load('cars_state_new.mat');
cars_state_new=cars_state;

car_t=zeros(200,1);              %car_t为车的全局时间
%  for  i=1:200
%       [m,n]=find(streets_str==num2str(cars_str_new(i,3)));
%       cars_str_shu=str2num(cars_str_new(i,2))-1;
%
%
cars_state_new=[cars_state_new;cars_str_new(i,3),streets_str(m,4),num2str
(cars_str_shu)];
```

```
% end
t_quan=0;       %全局时间
%%%
cars_zaiwei=200;     %200 个车都还在


while(cars_zaiwei>0)




% for i=1:200       %往前推动了 1s
%       car_state_3=str2num(cars_state_new(i,3));
%       if car_state_3~=-1
%
%       cars_state_new(i,2)=num2str(str2num(cars_state_new(i,2))-1);
%       end
%
% end
%
%
%
% t_quan=t_quan+1;


for i=1:200
    t=str2num(cars_state_new(i,2));
    cars_state_shu=str2num(cars_state_new(i,3));
    street_name=cars_state_new(i,1);
    if t==0 && cars_state_shu>0 && street_name~="wait" %进入判决




        t_jia=0;
    street_name=cars_state_new(i,1);
```

```
lukou_chuli=strsplit(street_name,'-');
lukou_name=lukou_chuli(1,2);
%搜索路口具体情况

%eval(['load("D:\桌面\省数模\A 题\问题 2 代码\路口的具体路数数组群\',num2str(lukou_name),'_xinxi.mat")']);

%根据路口的交通灯数进行分类讨论
eval(['size=numel(',num2str(lukou_name),'_xinxi)/2;']);    %如何求 string 组的大小

if  size>1          %路灯为 1 时
    t_lu=0;
    t_lu_ban=0;
    %要判断是否排队
    [m,n]=find(cars_str_new==cars_state_new(i,1));
    size_m=length(m);
    if  size_m==2   %则可能会排队
        for  jj=1:size
eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);    %t_lu 为总时间
            t_lu=t_lu+t;
        end
eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序
eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
```

```
;']); %t_lu_dangqian 为当前路的持续时间
            if
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(2,1),1)) &&
cars_state_new(m(1,1),2)==cars_state_new(m(2,1),2)
                %则可能排队
                if t_lu_dangqian==1
                    %在判断当前车是小还是大
                    m_max=max(m);
                    if i==m_max
                        %则等待一个周期
                        t_jia=t_jia+t_lu;
                    end
                end
            end
        elseif size_m==3 %则可能会排队
            for jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']); %t_lu 为
总时间
                t_lu=t_lu+t;
            end

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']); %t_lu_dangqian 为当前路的持续时间
            if
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(2,1),1))||strcmp(ca
rs_state_new(m(1,1),1),cars_state_new(m(3,1),1))||strcmp(cars_state_n
ew(m(2,1),1),cars_state_new(m(3,1),1))
                %则可能排队

                %判断是两个还是三个
                if
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(2,1),1)) &&
cars_state_new(m(1,1),2)==cars_state_new(m(2,1),2)
                    %则可能排队
```

```
                    if  t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m(1,1),m(2,1));
                        if  i==m_max
                            %则等待一个周期
                            t_jia=t_jia+t_lu;
                        end
                    end
                elseif
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(3,1),1))  &&
cars_state_new(m(1,1),2)==cars_state_new(m(3,1),2)
                    %则可能排队
                    if  t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m(1,1),m(3,1));
                        if  i==m_max
                            %则等待一个周期
                            t_jia=t_jia+t_lu;
                        end
                    end
                elseif
strcmp(cars_state_new(m(2,1),1),cars_state_new(m(3,1),1))  &&
cars_state_new(m(2,1),2)==cars_state_new(m(3,1),2)
                    %则可能排队
                    if  t_lu_dangqian==1
                        %在判断当前车是小还是大
                        m_max=max(m(2,1),m(3,1));
                        if  i==m_max
                            %则等待一个周期
                            t_jia=t_jia+t_lu;
                        end
                    end
                elseif
strcmp(cars_state_new(m(1,1),1),cars_state_new(m(2,1),1))&&strcmp(ca
rs_state_new(m(1,1),1),cars_state_new(m(3,1),1))&&strcmp(cars_state_n
ew(m(2,1),1),cars_state_new(m(3,1),1))&&
```

```
cars_state_new(m(1,1),2)==cars_state_new(m(3,1),2)&&
cars_state_new(m(2,1),2)==cars_state_new(m(3,1),2)
                        %则可能排队
                        if  t_lu_dangqian==1
                            %在判断当前车是小还是大
                            m_max=max(m);
                            m_min=min(m);
                            if  i~=m_max  &&  i~=m_min
                                %则等待一个周期
                                t_jia=t_jia+t_lu;
                            elseif  i==m_max
                                t_jia=t_jia+t_lu+t_lu;
                            end
                        elseif  t_lu_dangqian==2
                            m_max=max(m);
                            if  i==m_max
                                t_jia=t_jia+t_lu;
                            end
                        end
                    end
                end
            end


            for  jj=1:size

eval(['t=str2num(',num2str(lukou_name),'_xinxi(jj,2));']);     %t_lu 为
总时间
            t_lu=t_lu+t;
            end

eval(['lu_cixv=find(',num2str(lukou_name),'_xinxi==street_name);']);
%lu_cixv 为路名的顺序

eval(['t_lu_dangqian=str2num(',num2str(lukou_name),'_xinxi(lu_cixv,2))
;']);  %t_lu_dangqian 为当前路的持续时间
            if  lu_cixv~=1
```

```
            for  jjj=1:lu_cixv

eval(['ttt=str2num(',num2str(lukou_name),'_xinxi(jjj,2));']);
                t_lu_ban=t_lu_ban+ttt;                %t_lu_ban 为算上当前路后的
时间
            end
            t_lu_banjian=t_lu_ban-t_lu_dangqian;     %t_lu_banjian 为不算上当前
路的时间
            if  mod(t_quan,t_lu)<  t_lu_banjian
                t_jia=t_jia+(t_lu_banjian-mod(t_quan,t_lu));
            elseif   mod(t_quan,t_lu)  >  t_lu_ban
                t_jia=t_jia+(t_lu_banjian+t_lu-mod(t_quan,t_lu));
            end
            end
        end


%             if  t_jia<0
%             fprintf("11 找到虎虎啦!!!!!!  \n");
%             end


            if  t_jia==0
                %更新状态

                %fprintf("会有虎虎啦!!!!!!  \n");
                t=str2num(cars_str_new(i,2))+2;
                cars_state_new(i,1)=cars_str_new(i,t-cars_state_shu+1);
                [m,n]=find(streets_str==num2str(cars_state_new(i,1)));
                cars_state_new(i,2)=streets_str(m,4);
                cars_state_3=str2num(cars_state_new(i,3));
                cars_state_new(i,3)=num2str(cars_state_3-1);
            else
                cars_state_new(i,1)="wait";
                cars_state_new(i,2)=num2str(t_jia);
            end
```

```
    elseif  street_name=="wait"
        car_state_2=str2num(cars_state_new(i,2));


        if  cars_state_new(i,1)=="wait"  &&   car_state_2<0
            fprintf("22 找到虎虎啦!!!!  \n");
        end


        if  cars_state_new(i,1)=="wait"  &&  cars_state_new(i,2)=="0"
            %fprintf("虎虎呢？？？？\n");

            %更新状态
            t=str2num(cars_str_new(i,2))+2;
            cars_state_new(i,1)=cars_str_new(i,t-cars_state_shu+1);
            [m,n]=find(streets_str==num2str(cars_state_new(i,1)));
            cars_state_new(i,2)=streets_str(m,4);
            cars_state_3=str2num(cars_state_new(i,3));
            cars_state_new(i,3)=num2str(cars_state_3-1);
        end
    %elseif  t==0  &&  cars_state_shu==0
%           elseif   cars_state_shu==0
%               fprintf()
%           car_t(i,1)=t_quan;




    end
end


 for  i=1:200
        cars_3=str2num(cars_state_new(i,3));
        car_state_2=str2num(cars_state_new(i,2));
        if  cars_3==0  &&  car_state_2==0
            car_t(i,1)=t_quan;
            cars_state_new(i,3)="-1";
            cars_zaiwei=cars_zaiwei-1;
        end
```

```
 end
    fprintf("猫炸啦猫炸啦,第%f秒,",t_quan);
    fprintf("第%f个猫咪转圈圈,",cars_zaiwei);
    fprintf("第%f轮迭代\n",Pmax-P+1);

    for  i=1:200      %往前推动了 1s
    car_state_3=str2num(cars_state_new(i,3));
    if  car_state_3~=-1

    cars_state_new(i,2)=num2str(str2num(cars_state_new(i,2))-1);
    end
    end
    t_quan=t_quan+1;

end

%求总分
T=max(car_t);
for  i=1:200
    if  car_t(i,1)>858
        car_tt(i,1)=0;
    else
        car_tt(i,1)=250+858-car_t(i,1);
    end
end
Zongfeng=sum(car_tt);          %Zongfeng  即为当前解空间的总分值

%  %存一下迭代的过程
%  %Die_dai=[];          %  Die_dai 为迭代的过程展示值
```

```
% Die_dai=[Die_dai;P,Zongfeng];



%    3.判断，存优
if Zongfeng > Zui_youzongfeng
    Zui_youzongfeng=Zongfeng;


    %存到最优结果区


    save('D:\桌面\省数模\A 题\问题 2 代码\最优结果区
\Zui_youzongfeng.mat','Zui_youzongfeng');
    for i=1:lukou_length_gengxin

eval([num2str(xlukou(1,i)),'_xinxi_zui=',num2str(xlukou(1,i)),'_xinxi'
]);

        eval(['save("D:\桌面\省数模\A 题\问题 2 代码\最优结果区
\',num2str(xlukou(1,i)),'_xinxi_zui.mat","',num2str(xlukou(1,i)),'_x
inxi_zui")']);



    end
end



P=P-1;
```

```
%存一下迭代的过程
%Die_dai=[];            %  Die_dai 为迭代的过程展示值
zDie_dai=[zDie_dai;P,Zui_youzongfeng];

end

fprintf("最终的最高得分为%f\n",Zui_youzongfeng);
fprintf("完结撒花\n");
```