湖南省第三届研究生数学建模竞赛

题 目　　智慧考古探测的优化模型

摘　　要：

本文针对考古探测问题，基于题目所给数据信息，根据人工智能理论，确定其为一个搜索问题。其次，运用优化理论对该模型进行了数学描述，即多约束情况下的极小极大问题求解。对问题一的求解，利用智能算法进行编程仿真，给出了完成任务的最短时间。对问题二、三进行求解时，由几何知识推导出最有效的探测车初始布局，结合 CPU 资源调度原理，利用时序表得到行走路线和扫描次序的合理规划。

智慧考古探测的优化模型

1 问题重述

1.1 问题的背景

随着探测技术的发展，探测车在考古探测中得到越来越多的应用。探测车的工作原理类似于海水中的声纳设备，即向地下一定范围内发射一种特殊的“波”，通过接受并分析反射信号来获知地下建筑物情况或地质结构。探测的方式是将地面上的工作区域划分为若干个矩形区域，探测车行走到一个节点停稳后再进行探测，完成本点探测后行走到下一节点。探测过程称为“扫描”，一个节点扫描时间为 12s。

对于大面积的探测工作，需要多台探测车协同工作，为了避免多台探测车的互相干扰，任意 2 辆探测车的扫描时间间隔 *ts* 根据它们的距离 *d* 受到如下图所示的限制。

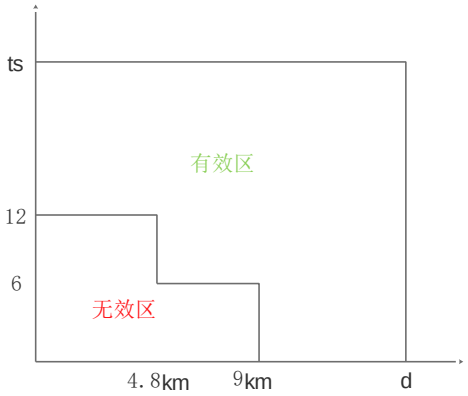

图 1. 扫描限制图

1.2 问题的提出

图 2 是一个区域的探测点分布图，探测车只能沿水平或垂直方向达到下一节点，行走 50m 的时间为 12 秒，走 100m 的时间为 24 秒。请建立数学模型，并解决相应问题。

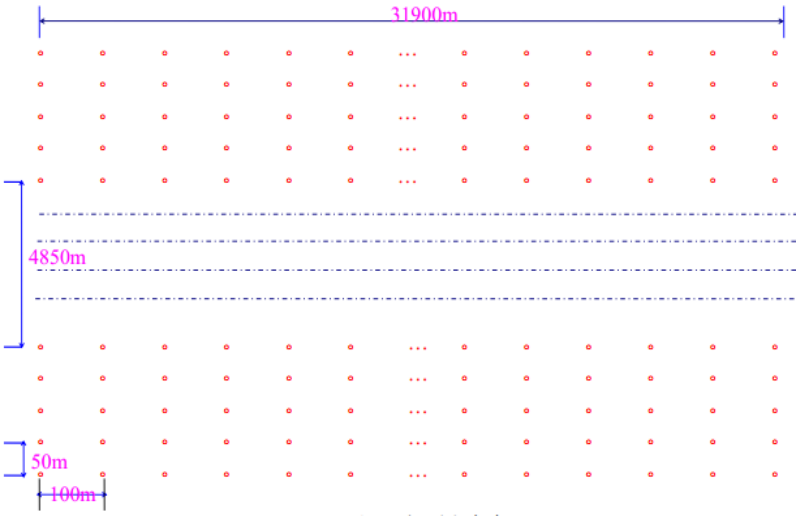

图 2. 探测点分布

1.3 待解决问题

根据以上数据信息，根据建立的模型，解决以下问题：

1、分别计算 12 台、20 台、64 台、80 台探测车的初始布局，行走路线，扫描次序，完成任务所花的最少时间。

2、需要投入多少台的探测车可以使得完成任务的效率最高？给出其初始布局，行走路线，扫描次序，完成任务所花的最少时间。

3、为了方便计算机进行无线远程协调控制，请进一步将问题 1、问题 2 的结果表达成计算机能够处理的数据组织方式，即：任意时刻探测车在何位置？何时计算机向何探测车发布何指令？

2 问题假设与分析

2.1 问题假设

根据待求解问题，在对实际情况进行合理考虑的基础上，为简化模型，对问题做如下假设：

(1)探测车只能在题中所给区域内工作，且不能过河。

(2)同一个探测点上不能同时存在两辆探测车。

(3)每辆探测车的位置不能突变，且探测车只有这几种状态：在工作节点上分为扫描和等待扫描；在移动状态下则只能连续向上下左右四个方向移动，直到到达下一个工作节点。

(4)探测点有这几种状态：节点被探测车占据时分为正被扫描和未被扫描；节点未被探测车占据时分为已扫描和未扫描。且节点只在同一探测车对其连续探测 12s 的情况下才算扫描完成。

2.2 问题分析

对于问题 1 的求解是本次建模所有问题的基础，后面两个问题均建立在对问题 1 思考和求解的基础之上。问题中只包含探测车和工作节点两种对象，二者的所有可能状态分别在问题假设的第 2 点和第 3 点中给出。从给出的数据可以发现，探测车的扫描时间(12s)、连续移动时间(12s or 24s)以及不受干扰的扫描时间间隔(0 or 6s or 12s)均为 6 的整倍数，故可每 6s 对所有对象的状态进行一次更新。

问题 1、2 均是要求在消耗时间最少及效率最高的条件下进行问题的求解。由于每辆探测车有非扫描的空窗期，则应保证每个时间段都有尽可能多的探测车参与扫描任务。对于探测车队初始布局而言，由于探测车能否开启扫描受到与其他探测车之间距离的影响，因此初始布局应尽量平均分布在工作区域内，以尽量减小距离约束条件的影响。

对于探测车队的行走路线和扫描次序而言，即寻找探测车队最优的状态序列。结合 CPU 资源调度的理论，将其转化为一个寻找探测车队行动序列的问题。

3 建立模型与参数说明

3.1 模型抽象

在设计探测车的行走路线和扫描次序时，我们将探测车视作一个智能 Agent，根据题目的描述先给出探测车的任务环境（PEAS）：

(1)性能度量（Performance）：探测车队用尽可能少的时间和尽可能高的效率完成扫描任务；

(2)环境（Environment）：探测车不受干扰的扫描时间间隔受到它们之间距离的影响；

(3)执行器（Actutors）：完成保持停止及移动（上下左右），开启扫描等行动；

(4)传感器（Sensors）：感知附近的节点是否被扫描，以及探测器之间的距离。

由以上分析，我们可以对任务环境的性质进行如下假设：

(1)有效完全可观察：Agent 的传感器在每个时间点上都能获取探测区域的完整状态并检测所有与行动决策相关的信息。

(2)确定的：环境的下一个状态完全取决于当前状态和 Agent 执行的动作。

(3)静态的：环境在 Agent 计算的时候不会变化。

当然，知道当前的环境状态对探测车的决策而言并不够，所以还需要目标信息来描述想要达到的状况——对区域中的每个探测点进行扫描，即将其视为基于目标的 Agent。则对该模型进行仿真求解时，可以运用搜索方法找到多 Agent 的一组行动序列来完成目标任务。

3.2 数学模型

3.2.1 问题的形式化描述

将整个仿真区域视作一个节点均匀分布的阵列，不考虑处于河流区域内的节点。设置一个节点矩阵 $nodeMat=[(\text{scan},\text{status})_{ij}]_{10\times 320}$，每个元素 $(\text{scan},\text{status})_{ij}$ 在矩阵中的行列数 i 和 j 分别对应工作区域中从上至下的行数和从左至右的列数。$(\text{scan},\text{status})_{ij}$ 中 $scan_{ij}$ 和 $status_{ij}$ 所有状态的含义如下：

$$\text{scan}_{ij}=\begin{cases}0 & \text{节点未被探测车占据}\\ 1 & \text{节点被探测车占据但未被扫描}\\ 2 & \text{节点被 正被探测车扫描}\end{cases} \tag{3.1}$$

$$\text{status}_{ij}=\begin{cases}0 & \text{该探测点已扫描}\\ 1 & \text{该探测点未扫描}\end{cases} \tag{3.2}$$

建立一个包含河流的工作区域的坐标系，坐标原点在左上角。考虑到纵向相邻探测点的间距为横向相邻探测点间距的 2 倍，则节点矩阵中元素 status_{ij} 在坐标系中坐标 (pos_x, pos_y) 与其在矩阵中行列数的关系式如下：

$$(\text{pos\_x},\text{pos\_y})=\begin{cases}(\text{i},2\,\text{j}-1) & i<6\\ (\text{i}+96,2\,\text{j}-1) & i>=6\end{cases} \tag{3.3}$$

工作区域坐标系的简单示例如下图 3 所示。

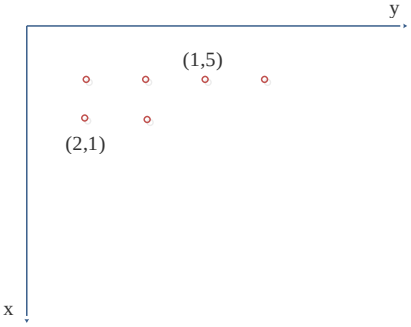

图 3. 工作区域坐标系

设置一个关于探测车的状态向量 $[(\text{position},\text{action},\text{scan})_k]_{n\times 1}$，$(\text{position},\text{action},\text{scan})_k$ 表示第 k 辆车的状态，其中的元素 $position_k$ 表示第 k 辆探测车所在节点的坐标，$action_k$ 为第 k 辆车的移动标志位，$scan_k$ 表示第 k 车的扫描标志位，其可能状态如下：

$$position_k = (\text{pos\_xk}, \text{pos\_yk}) \tag{3.4}$$

$$action_k = \begin{cases} 0 & \text{探测车不移动} \\ 1 & \text{探测车向上移动} \\ 2 & \text{探测车向下移动} \\ 3 & \text{探测车向左移动} \\ 4 & \text{探测车向右移动} \end{cases} \tag{3.5}$$

$$scan_k = \begin{cases} 0 & \text{探测车未扫描} \\ 1 & \text{探测车已扫描6s} \\ 2 & \text{探测车已扫描12s} \end{cases} \tag{3.6}$$

根据上述前提，对问题进行如下的形式化描述：

状态：由工作区域中每个节点的状态及所有探测车的位置、状态确定。

Agent 的初始状态：每辆探测车所在的节点坐标以及本身的状态（未扫描）。

Agent 的可能行动：扫描和等待扫描，向上、下、左、右移动。

转移模型：Agent 的行动所产生的结果。

目标测试：检测所有节点是否被扫描，即节点矩阵中的 $status_{ij}$ 是否全为 0。

路径消耗：完成全部扫描任务所花费的时间。

3.2.2 问题的数学描述

设总共有 n 辆探测车参与任务，t_k 表示第 k 辆探测车完成最后一次扫描任务后所消耗的时间有：

$$t_k = t_{wait} + t_{move} + t_{scan} \tag{3.7}$$

则目标函数为：

$$\min\max(t_k) \quad k = \{1, 2\cdots n\} \tag{3.8}$$

完成全部扫描任务的约束条件为节点矩阵中所有元素为 0，即：

$$\sum status_{ij} = 0 \qquad i = \{1, 2\cdots 10\},\ j = \{1, 2\cdots 320\} \tag{3.9}$$

第 m 辆和第 n 辆探测车之间的距离 d_{ij} 为：

$$d_{mn} = \sqrt{[(pos\_xm - pos\_xn)\times 50]^2 + [(pos\_\text{ym} - pos\_\text{yn})\times 50]^2} \tag{3.10}$$

第 i 辆和第 j 辆探测车互不干扰的扫描时间间隔 Δt_{ij} 与距离 d_{ij} 的关系如下所示：

$$\Delta t_{mn} = \begin{cases} 12 & d_{mn} < 4800 \\ 6 & 4800 <= d_{mn} <= 9000 \\ 0 & d_{mn} > 9000 \end{cases} \tag{3.11}$$

综合上述分析，该模型能转化为一个求解约束条件下的极小极大问题最优解的问题。

3.3 参数说明

表 1. 参数说明

参数	说明
t_{wait}	探测车在当前节点等待的时间
t_{move}	探测车移动的时间
t_{scan}	探测车扫描当前节点的时间
d_{ij}	第 i 辆和第 j 辆探测车之间的实际距离
Δt_{ij}	第 i 辆和第 j 辆探测车间开启扫描的间隔时间
pos_xk	第 i 辆探测车所在节点的横坐标
pos_yk	第 i 辆探测车所在节点的纵坐标
action	移动标志
status	节点状态标志

4 模型求解与仿真分析

4.1 问题 1 的求解

4.1.1 初始布局

探测车队初始布局尽量均匀分布，由于中间隔了一条宽度为 4.85km 的河流，为尽量降低探测车的扫描受到河对面探测车的影响，由下式计算：

$$\sqrt{4850^2 + y^2} > 9000 \tag{3.12}$$

得到 $y > 7589\text{m}$，即 y 方向需要隔 76 个节点以上。而 y 方向总共只有 320 列，不到 76 的 5 倍，故可将工作区域按 y 轴每 75 个节点一个矩形区域进行划分。对初始布局而言，应尽量减小距离限制对探测车开启扫描的影响。鉴于题目分别要求 12、20、64、80 辆车参与扫描任务，故对初始布局的构思如下：

(1)8 辆探测车均匀分布在工作区域内，初始位置固定为：(1,1)、(102,151)、(1,301)、(102,451)、(106,1)、(5,151)、(106,301)、(5,451)，如下图所示。

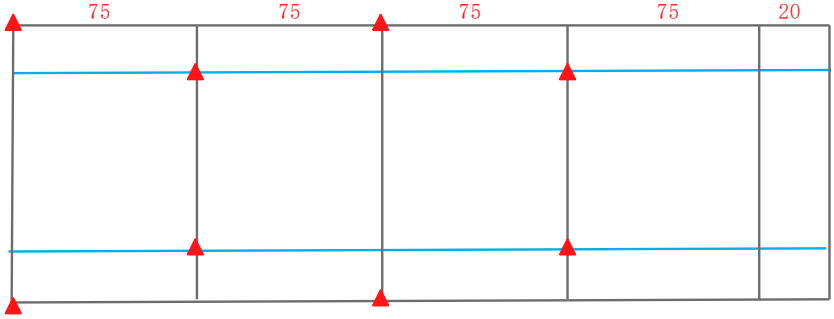

图 4. 初始布局示意图

(2)其余探测车先均分在河的两侧区域，在河流两侧的探测车的坐标分别为(3,pos_y_up)和(104,pos_y_down)，其中，pos_y_up 和 pos_y_down 均沿 y 轴均匀分布。

4.2.2 行走路线和扫描次序

对于行走路线和扫描次序的选取，相当于我们要为探测车队寻找一组合适的状态转移序列。基于之前将探测车队看作多 Agent 的假设，为 Agent 设计基于贪心算法的搜索策略来进行问题求解：

(1)所有满足约束条件且处于可扫描状态的探测车均进行扫描；

(2)探测车对当前所在节点扫描完成后，其行动选择按如下算法进行如下：

Step1 先判断所在列有没有未被扫描的节点，如果有则优先向离当前位置最近的未扫描点移动，否则转向 Step2。

Step2 判断所在行左右各两个节点有没有未被扫描的节点，如果有则优先向离当前位置最近的未扫描点移动，否则转向 Step3。

Step3 比较探测车所在列及该列左侧所有未被扫描的节点的个数与该列右侧所有未被扫描节点的个数，然后往个数较大侧且距离自己最近的未扫描节点移动。

图 5. 算法流程

4.2.3 仿真结果分析

仿真过程，比较了不同初始布局方式对扫描任务的最快扫描完需要的探测车的数量，比较结果见表 2.

表 2 初始布局对完成扫描任务的影响

初始布局	数量/辆	时间/秒
区间随机分布	151	15890
总体随机分布	150	10130
均匀分布	59	9606

初始布局按均匀分布分布效率最高，即耗时最少，动用车辆数量最少。针对问题一，图 6 说明了当车辆数量分别是 12、20、64、80 时不同初始布局对扫描时间的影响。对比之下，均匀分布的布局方式所需时间最少。车辆数量分别是 12、20、64、80 时的行走路线和扫描次序见附录《探测车数量是 12-20-64-80 时的扫描次序和行走路线.xlsx》。探测车数量是 12、20、64、80 时的扫描时间见图 7。

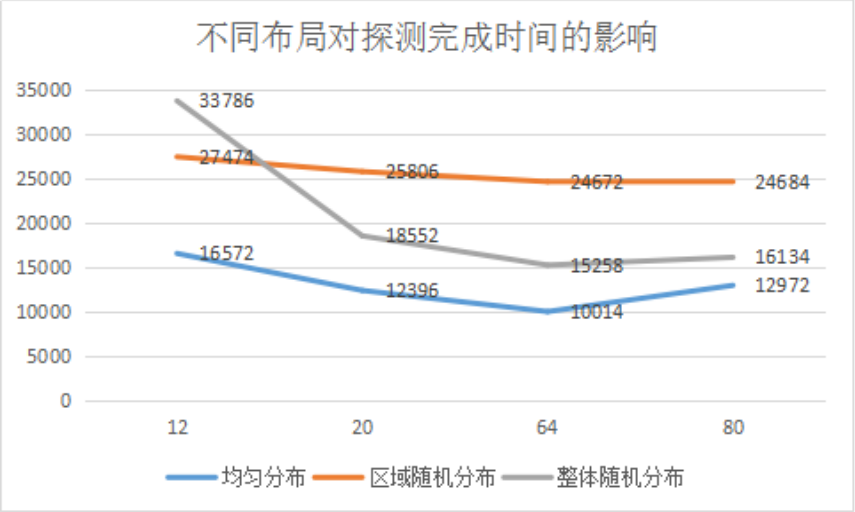

图 6. 初始布局对完成扫描任务的影响

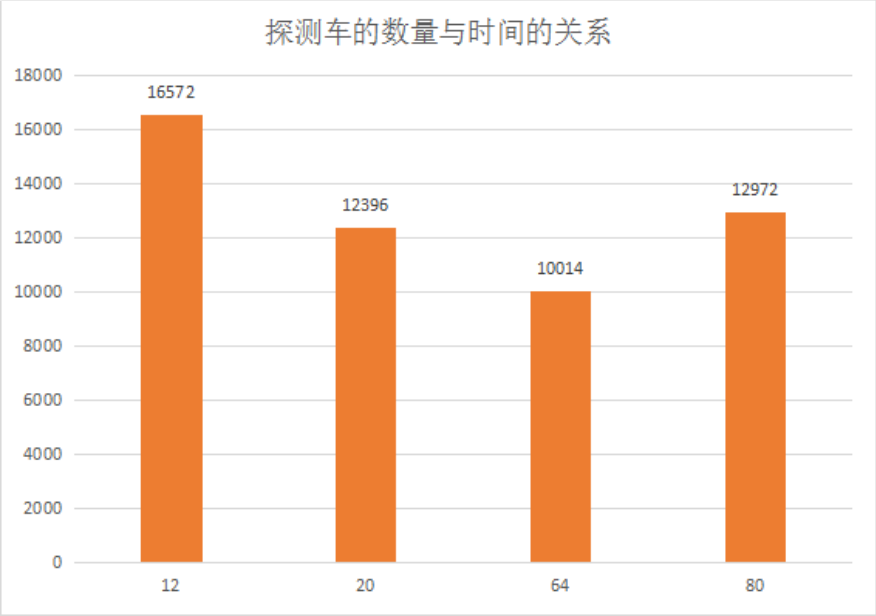

图 7 探测车数量是 12、20、64、80 时的扫描时间

4.2 问题 2 和问题 3 的求解

将时间离散化为 6 秒一个的间隔，通过分析我们可以发现，河两边对应节点的检查时序只需相差 6 秒（d_{ij}>4.8km），即一个时间间隔。由上一节的分析可知，同一时刻探测的车辆数极限情况是五个，它们的坐标分别是(1,1)、(102,151)、(1,301)、(102,451)、(1,601)，画出它们在区域中的排布位置如图 8 所示。

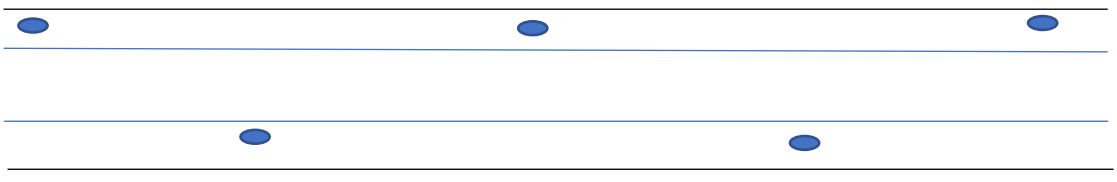

图 8. 五辆探测车同时工作的分布图

以 9000m 为半径，以上述坐标为探测车的初始位置，可以得到它们不受干扰的区域示意图如图 9 所示。

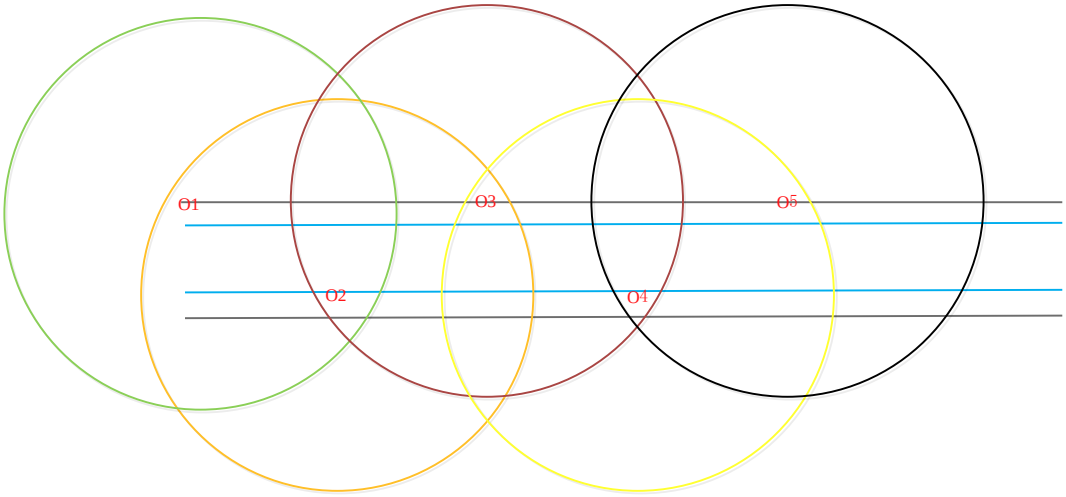

图 9. 同时工作不受干扰的五辆探测车

为保证始终满足约束条件，选择圆行的交点坐标分别为：第一行三个点的坐标为(1,1)、(1,301)、(1,601)；第六行两个点的坐标为分别为(102,151)、(102,451)。又因为上下距离>4800m，所以过了 6 秒后下面也会有 5 个点可以探测。即基本车辆初始化分布如图 10 所示。

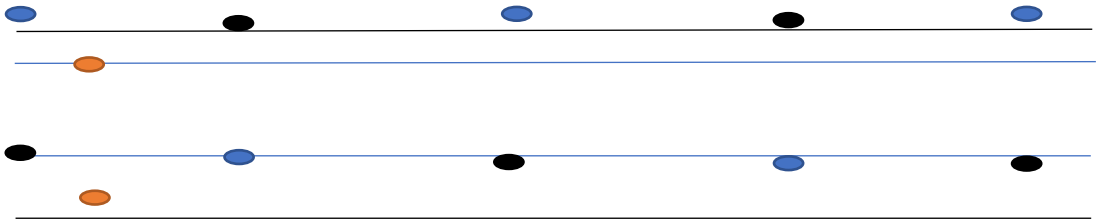

图 10. 五辆探测车交替工作的分布图

蓝色为第一组：(1,1)、(102,151)、(1,301)、(102,451)、(1,601)；黑色为第二组：(102,1)、(1,151)、(102,301)、(1,451)、(102,601)；棕色为第三组。

不失一般性，我们先对分组的时序进行如下分析。设置 A1，B1，C1 初始结点如下表 3。（表中的数据的含义：小数点前后两个数表示所在位置的横纵坐标，标红则表示该节点处于起始位置，箭头代表结点移动方向。）

表 3 优先级轮转

A1		B1		C1	
1.1	1.2	1.3	1.4	1.5	1.6
2.1	2.2	2.3	2.4	2.5	2.6
3.1	3.2	3.3	3.4	3.5	3.6
4.1	4.2	4.3	4.4	4.5	4.6
5.1	5.2	5.3	5.4	5.5	5.6

对于 A5 和 A6，我们考察离区域 1（圆 O1）最近的节点（1,151）作为分析

说明情况。

表 4　A6 行动序列

A6	
1.76	1,77
2,76	2,77
3,76	3,77
4,76	4,77
5,76	5,77

每一列对应于当前时刻结点采取的操作，对一个区域的节点进行时空分析，该状态队列即可作为计算机能接受的序列信息。其中，s—停止，mn—移动 n 次，d—探测。优先级轮转顺序：ABC-ADC-ABC-ADC---

表 5　完整时序图

时 间 / 车辆状态	A1	A6	B	C1	D	
1	1.1 s	1,76w	1.3w	1.5w	1.9w	第一次循环 AB 段
2	1.1d	1,76 s	1,3w	1.5w	1.9w	
3	2.1m	1,76d	1.3s	1.5w	1.9w	
4	2.1m	2,76m	1.3d	1.5w	1.9w	
5	2.1s	2,76m	2.3m	1.5w	1.9w	
6	2.1d	2,76s	2.3m	1.5w	1.9w	
7	3.1m	2,76d	2.3s	1.5w	1.9w	
8	3.1m	3,76m	2.3d	1.5w	1.9w	
9	3.1s	3,76m	3.3m	1.5w	1.9w	
10	3.1d	3,76s	3.3m	1.5w	1.9w	
11	4.1m	3,76d	3.3s	1.5w	1.9w	
12	4.1m	4,76m	3.3d	1.5w	1.9w	
13	4.1s	4,76m	4.3m	1.5w	1.9w	
14	4.1d	4,76s	4.3m	1.5w	1.9w	
15	5.1m	4,76d	4.3s	1.5w	1.9w	
16	5.1m	5,76m	4.3d	1.5w	1.9w	
17	5.1s	5,76m	5.3m	1.5w	1.9w	
18	5.1d	5,76s	5.3m	1.5w	1.9w	
19	5.2m	5,76d	5.3s	1.5w	1.9w	
20	5.2m	5,77m	5.3d	1.5w	1.9w	
21	5.2m	5,77m	5.4m	1.5s	1.9w	第一次循环 CA 段
22	5.2m	5,77m	5.4m	1.5d	1.9w	
23	5.2s	5,77m	5.4m	2.5m	1.9w	
24	5.2d	5,77s	5.4m	2.5m	1.9w	

25	4.2m	5,77d	5.4w	2.5s	1.9w	
26	4.2m	4,77m	5.4w	2.5d	1.9w	
27	4.2s	4,77m	5.4w	3.5m	1.9w	
28	4.2d	4.77s	5.4w	3.5m	1.9w	
29	3.2m	4.77d	5.4w	3.5s	1.9w	
30	3.2m	…	5.4w	3.5d	1.9w	
31	3.2s	…	5.4w	4.5m	1.9w	
32	3.2d	…	5.4w	4.5m	1.9w	
33	2.2m	…	5.4w	4.5s	1.9w	
34	2.2m	…	5.4w	4.5d	1.9w	
35	2.2s	…	5.4w	5.5m	1.9w	
36	2.2d	…	5.4w	5.5m	1.9w	
37	1.2m	…	5.4w	5.5s	1.9w	
38	1.2m	…	5.4w	5.5d	1.9w	
39	1.2s	…	5.4w	5.6m	1.9w	
40	1.2d	…	5.4w	5.6m	1.9w	
41	1.3m	…	5.4s	5.6m	1.9w	第一次循环 BC 段
42	1.3m	…	5.4d	5.6m	1.9w	
43	1.3m	…	4.4m	5.6s	1.9w	
44	1.3m	…	4.4m	5.6d	1.9w	
45	1.4m	…	4.4s	4.6m	1.9w	
46	1.4m	…	4.4d	4.6m	1.9w	
47	1.4m	…	3.4m	4.6s	1.9w	
48	1.4m	…	3.4m	4.6d	1.9w	
49	1.5m	…	3.4s	3.6m	1.9w	
50	1.5m	…	3.4d	3.6m	1.9w	
51	1.5m	…	2.4m	3.6s	1.9w	
52	1.5m	…	2.4m	3.6d	1.9w	
53	1.6m	…	2.4s	2.6m	1.9w	
54	1.6m	…	2.4d	2.6m	1.9w	
55	1.6m	…	1.4m	2.6s	1.9w	
56	1.6m	…	1.4m	2.6d	1.9w	
57	1.7m	…	1.4s	1.6m	1.9w	
58	1.7m	…	1.4d	1.6m	1.9w	
59	1.7m	…	1.5m	1.6s	1.9w	
60	1.7m	…	1.5m	1.6d	1.9w	
61	1,7s	…	1.5m	1.7m	1.9w	第一次循环 AD 段
62	1.7d	…	1.5m	1.7m	1.9w	
63	2.7m	…	1.6m	1.7m	1.9s	
64	2.7m	…	1.6m	1.7m	1.9d	
65	2.7s	…	1.6m	1.8m	2.9m	
66	2.7d	…	1.6m	1.8m	2.9m	

67	3.7m	…	1.7m	1.8m	2.9s	
68	3.7m	…	1.7m	1.8m	2.9d	
69	3.7s	…	1.7m	1.9m	3.9m	
70	3.7d	…	1.7m	1.9m	3.9m	
71	4.7m	…	1.8m	1.9m	3.9s	
72	4.7m	…	1.8m	1.9m	3.9d	
73	4.7s	…	1.8m	1.10m	4.9m	
74	4.7d	…	1.8m	1.10m	4.9m	
75	5.7m	…	1.9m	1.10m	4.9s	
76	5.7m	…	1.9m	1.10m	4.9d	
77	5.7s	…	1.9m	1.11m	5.9m	
78	5.7d	…	1.9m	1.11m	5.9m	
79	5.8m	…	1.10m	1.11m	5.9s	
80	5.8m	…	1.10m	1.11m	5.9d	
81	5.8m	…	1.10m	1.11s	5.10m	第一次循环 CA 段
82	5.8m	…	1.10m	1.11d	5.10m	
83	5.8s	…	1.11m	2.11m	5.10m	
84	5.8d	…	1.11m	2.11m	5.10m	
85	4.8m	…	1.11m	2.11s	5.10w	
86	4.8m	…	1.11m	2.11d	5.10w	
87	4.8s	…	1.12m	3,11m	5.10w	
88	4.8d	…	1.12m	3.11m	5.10w	
89	3.8m	…	1.12m	3.11s	5.10w	
90	3.8m	…	1.12m	3.11d	5.10w	
91	3.8s	…	1.13m	4,11m	5.10w	
92	3.8d	…	1.13m	4.11m	5.10w	
93	2.8m	…	1.13m	4.11s	5.10w	
94	2.8m	…	1.13m	4.11d	5.10w	
95	2.8s	…	1.14m	5,11m	5.10w	
96	2.8d	…	1.14m	5.11m	5.10w	
97	1.8m	…	1.14m	5.11s	5.10w	
98	1.8m	…	1.14m	5.11d	5.10w	
99	1.8s	…	1.15m	5.12m	5.10w	
100	1.8d	…	1.15m	5.12m	5.10w	
101	1.9m	…	1.15m	5.12m	5.10s	第一次循环 DC 段
102	1.9m	…	1.15m	5.12m	5.10d	
103	1.9m	…	1.15w	5.12s	4,10m	
104	1.9m	…	1.15w	5.12d	4.10m	
105	1.10m	…	1.15w	4,12m	4,10s	
106	1.10m	…	1.15w	4,12m	4.10d	
107	1.10m	…	1.15w	4,12s	3,10m	
108	1.10m	…	1.15w	4.12d	3,10m	
109	1.11m	…	1.15w	3,12m	3,10s	

110	1.11m	…	1.15w	3,12m	3,10d	
111	1.11m	…	1.15w	3,12s	2,10m	
112	1.11m	…	1.15w	3.12d	2,10m	
113	1.12m	…	1.15w	2,12m	2,10s	
114	1.12m	…	1.15w	2,12m	2,10d	
115	1.12m	…	1.15w	2,12s	1,10m	
116	1.12m	…	1.15w	2.12d	1,10m	
117	1.13m	…	1.15w	1,12m	1,10s	
118	1.13m	…	1.15w	1,12m	1,10d	
119	1.13m	…	1.15w	1,12s	1,11m	
120	1.13m	…	1.15w	1.12d	1,11m	c 车最后探测点
121	1.13s	…	1.15w	1.13m	1.11m	第二次循环 AB 段
122	1.13d		1.15w	1.13m	1,11m	
123	2,13m		1.15s	1.13m	1,12m	
124	2,13m		1.15d	1.13m	1,12m	

表 6 探测顺序表

车辆编号|探测节点	第一次	第二次
A	1	2
B	3	4
C	5	6
A	7	8
D	9	10
C	11	12
A	13	14
B	15	16
C	17	18
A	19	
D	20	

所以其总时间为 120*6+228+6+6=960s

从表 3 我们可以看出，B1 与 A6 的初始距离 146>96，所以 1-4 与 5-6 可以相隔 6 秒分别工作，又因为 1234 与 5678 是平行移动的，所以任何时候都可以满足 4 个区域有 75 个节点，因为 75mod（12）=3 所以剩下三个节点，不能重复 6 点序列循环。因此，循环 6 次后，进入第二大阶段。理论分析最短探测完所有点的时间为最后三列的处理方式如下，即 A 车从上到下 B 车从下到上，C 车在中间那个点：

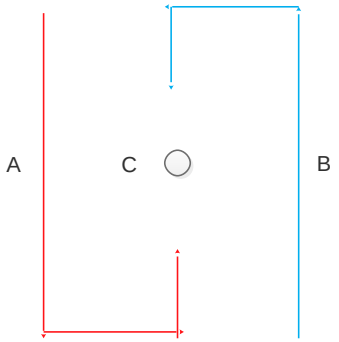

图 11. 最后三列节点的扫描图

将问题分为 10 个区域，前 8 个区域每个有 375 个点，最后一个区域只有 100 个点，故总时间 360*12+180+6=4506 秒。

表 7 第二阶段的扫描序列

第二阶段 处理剩余的3列 从t=120*6=720开始处理						只需要三辆车ABC	
721	1.73s	5.75w	3.74w				
722	1.73d	5.75w	3.74w	只需要三辆车ABC			
723	2,73m	5.75s	3.74w				
724	2,73m	5.75d	3.74w				
725	2.73s	4.75m	3.74w				
726	2.73d	4.75m	3.74w				
727	3.73m	4.75s	3.74w				
728	3.73m	4.75d	3.74w				
729	3.73s	3.75m	3.74w				
730	3.73d	3.75m	3.74w				
731	4.73m	3.75s	3.74w				
732	4.73m	3.75d	3.74w				
733	4.73s	2.75m	3.74w				
734	4.73d	2.75m	3.74w				
735	5.73m	2.75s	3.74w				
736	5.73m	2.75d	3.74w				
737	5.73s	1.75m	3.74w				
738	5.73d	1.75m	3.74w				
739	5.74m	1.75s	3.74w				
740	5.74m	1.75d	3.74w				
741	5.74m	1.74m	3.74s				
742	5.74m	1.74m	3.74d				
743	5.74s	1.74m					
744	5.74d	1.74m					
745	4.74m	1.74s					
746	4.74m	1.74d					
747	4.74s	2.74m					
748	4.74d	2.74m					
749		2.74s					
750		2.74d					

5 参考文献

[1]唐洪英，周敏. 基于分层分次，贪心算法的排课系统的设计与实现[J]. 微计算机信息，2006 (01X): 237-240.

[2]陈建中，刘大有. 智能 Agent 建模的一种模板结构[J]. 计算机研究与发展，1999，36(10): 1164-1168.

[3] 王海鹰，张乃良. 带约束条件的离散 Minimax 问题的区间极大熵方法[J]. 高校应用数学学报 (A 辑), 2000, 15(3): 369-376.

附录

```
%输入为车辆数 n,n 为 4 的倍数。
function [Save_workNum,route,status,node,gt]=diapatch(n)
%% 初始化各项参数，以及点的坐标。
posx=zeros(n,1);
posy=zeros(n,1);
py=zeros(10,320);%坐标 y
px=zeros(10,320);%坐标 x
xa=zeros(n,1);
ya=zeros(n,1);

numberMax_work=0;%任意时刻在扫描状态的小车最大数量  1<=numberMax_work<=5;

%初始化每个点的坐标
for i=1:10
    for j=1:320
       if i<6
            px(i,j)=i;
            py(i,j)=j*2-1;
       else
           px(i,j)=i+96;
           py(i,j)=j*2-1;
       end
    end
end
%状态矩阵，每六秒改变一次状态
%状态有，每一时刻的，x 坐标，y 坐标，移动方向，探测阶段 0，1，2；0 2 1 0
time=32000;
status=zeros(n,4,time);
Save_workNum=zeros(time,1);
%路径矩阵
route=zeros(n,2,3200);
%% 初始化 n 个结点，先初始化 8 个，再初始化其余结点。
status=initialPoint(n,status,1);
%% 路径初始化
for i=1:n
    route(i,1,1)=status(i,1,1);
    route(i,2,1)=status(i,2,1);
    posx(i)=status(i,1,1);
    posy(i)=status(i,2,1);
end
```

```
p1=0.4;
p2=0.4;
p3=0.1;
p4=0.1;
node=ones(10,320);

discover=zeros(n,1);%操作状态 0 1 2
direction=zeros(n,1);%移动方向
%两种初始化方案，一种 1,1；102,150；1,299;102,448；|||106,1；5,150；106,299;5,448 剩余点四个一组
%1,1;1,161;1,321,1;480;
%102,1;102,161;102,321,102;480
gt=1;%时间 一个单位 6 秒

while sum(sum(node)) && gt<time

for i=1:n
    t=gt;
%% 先判断它是出于坐标点还是移动状态。
%处于探测状态
    if xa(i)==0&&ya(i)==0
    %再判断探测车是处于探测 1 阶段还是探测 2 阶段还是无等待状态；0->2->1->0,0 是等待状态

        if discover(i)==2
            t=t+1;
            discover(i)=discover(i)-1;
            status(i,1,t)=posx(i) ;
            status(i,2,t)=posy(i) ;
            status(i,3,t)=direction(i) ;
            status(i,4,t)=discover(i);
        continue;
    else if discover(i)==1

          nodmat=inverse(status(i,1,t),status(i,2,t));
          discover(i)=discover(i)-1;
          node(nodmat(1),nodmat(2))=0;
          t=t+1;
          status(i,1,t)=posx(i) ;
          status(i,2,t)=posy(i) ;
          status(i,3,t)=direction (i);
          status(i,4,t)=discover(i);
           continue;
        end
       end
```

```
    end
 %% 如果该点已被检测，则向下一个方向移动
 nodmat=inverse(status(i,1,t),status(i,2,t));

% a=nodmat(1,1)
% b=nodmat(2,1)
% if(nodmat(1,1)==0)
% nodmat(1,1)=1;
% posx(i)=1;
% end
% if a==0
% d=direction(i)
% i
%
%  x0=status(i,1,t-1)
%  y0=status(i,2,t-1)
%  x=status(i,1,t)
%  y=status(i,2,t)
% m_node=node;
% gt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% figure(1)
%  if i==1
%  plot(status(i,2,gt),-status(i,1,gt),'r*');
%  hold on;
%  pause(0.01);
% % %  else
% % %  plot(status(i,2,gt),-status(i,1,gt),'b*');
% % %  hold on;
%  %pause(0.01);
%  end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %   pause(0.01);
% % end
% for ttime=1:1:gt
%  p11(ttime)=status(i,1,ttime);
%  x=status(i,1,ttime);
%  p12(ttime)=status(i,2,ttime);
%  plot(status(i,1,ttime),status(i,2,ttime),'b*');
%  if status(i,1,ttime)==1&&status(i,2,ttime)==5
%     count=count+1;
%  end
%  hold on
%
```

```
%    pause(0.01);
% end
% %  plot(p12,p11);
% %  hold on
% end
%         if x==0
%           x=0
%         end


        if node(nodmat(1,1),nodmat(2,1))==0

         %如果不处于移动状态，则判断移动方向
          if xa(i)==0&&ya(i)==0% xa:-2 -1 0 1 2   ya: -3 -2 -1 0 1 2 3
         %direction(i)=getd(posx(i),posy(i),p1,p2,p3,p4,direction(i),node);
         direction(i)=Move(posx(i),posy(i),node);
         %边界方向限制，出现错误强制反向
         if
nodmat(1,1)==1||nodmat(1,1)==5||nodmat(1,1)==6||nodmat(1,1)==10||nodmat(2,1)==1||nodmat(2,1)
==320
             if (nodmat(1,1)==1||nodmat(1,1)==6)&&direction(i)==1
                 direction(i)=3;
             end

             if (nodmat(1,1)==5||nodmat(1,1)==10)&&direction(i)==3
                 direction(i)=1;
             end

             if nodmat(2,1)==1&&direction(i)==4
                  direction(i)=2;
             end

             if nodmat(2,1)==320&&direction(i)==2
                  direction(i)=4;
             end
         end

          switch direction(i)
              case 3,
                 if xa(i)<1
                    xa(i)=xa(i)+1;
                 else
                    xa(i)=0;
                    posx(i)=posx(i)+1;
```

```
                end

            case 2,
                if ya(i)<3
                    ya(i)=ya(i)+1;
                else
                    ya(i)=0;
                    posy(i)=posy(i)+2;
                end

            case 1,
                if xa(i)>-1
                    xa(i)=xa(i)-1;
                else
                    xa(i)=0;
                    posx(i)=posx(i)-1;
                end
          case 4
                if ya(i)>-3
                    ya(i)=ya(i)-1;
                else
                    ya(i)=0;
                   posy(i)=posy(i)-2;
                end
            otherwise
         end
          t=t+1;
             status(i,1,t)=posx(i) ;
             status(i,2,t)=posy(i) ;
             status(i,3,t)=direction(i);
             status(i,4,t)=discover(i);

             continue;
%         a1=direction(5);
%          a2=ya(5);
       %如果处于移动状态则继续移动
          else
             %c= direction(i)
              if ya(i)>0
                  direction(i)=2;
              end
                 if ya(i)<0
                      direction(i)=4;
                 end
```

```
            if xa(i)>0
                 direction(i)=3;
            end
                if xa(i)<0
                    direction(i)=1;
                end
switch direction(i)

case 3,
     if xa(i)<1
           xa(i)=xa(i)+1;
         else
           xa(i)=0;
           posx(i)=posx(i)+1;
     end

case 2,
        if ya(i)<3
           ya(i)=ya(i)+1;
        else
            ya(i)=0;
            posy(i)=posy(i)+2;
        end

 case 1,
        if xa(i)>-1
           xa(i)=xa(i)-1;
        else
           xa(i)=0;
           posx(i)=posx(i)-1;
        end
  case 4,
        if ya(i)>-3
           ya(i)=ya(i)-1;
        else
           ya(i)=0;
          posy(i)=posy(i)-2;
        end

   end

      t=t+1;
      status(i,1,t)=posx(i) ;
      status(i,2,t)=posy(i) ;
```

```
            status(i,3,t)=direction(i);
            status(i,4,t)=discover(i);
            continue;

        end

            % c= direction(i)

    end
%% 如果该点未被访问过，则判断它周围是否有可以影响到它的点
              if   node(nodmat(1),nodmat(2))==1
                  label=0;%周围是否不男足约束
                  for j= 1:n;
                     if i~=j
                        if discover(j)==1
                          if  sqrt((posx(i)-posx(j)).^2+(posy(i)-posy(j)).^2)<96
                              label=1;  %不满足
                              direction(i)=0;
                              discover(i)=0;
                              t=t+1;
                              status(i,1,t)=posx(i) ;
                                status(i,2,t)=posy(i) ;
                                status(i,3,t)=direction(i);
                               status(i,4,t)=discover(i);
                              break;
                          end
                        end

                         if discover(j)==2

                          if sqrt((posx(i)-posx(j)).^2+(posy(i)-posy(j)).^2)<180
                              label=1;
                              direction(i)=0;
                               t=t+1;
                               discover(i)=0;
                            status(i,1,t)=posx(i) ;
                           status(i,2,t)=posy(i) ;
                           status(i,3,t)=direction(i);
                           status(i,4,t)=discover(i);
                              break;
                          end
                         end
                     end
                  end
```

```
                    if label==0
                       discover(i)=2;
                    end
                end


  %% 状态更新
  if label==0
    t=t+1;
     status(i,1,t)=posx(i) ;
     status(i,2,t)=posy(i) ;
     status(i,3,t)=direction(i);
     status(i,4,t)=discover(i);
  end

end
  gt=gt+1;

%% 计算任意时刻在扫描的车的数量
for s_num=1:n
          if  status(s_num,4,t)>0
               numberMax_work=numberMax_work+1;
          end
end
  Save_workNum(gt)=numberMax_work;
  numberMax_work=0;%重新计数
end
% 状态矩阵的有效维度
% tlabel=find(status(1,1,:)==0, 1 );
% for i=2:n
%    temple=find(status(i,1,:)==0, 1 );
%    tlabel=min(tlabel,temple);
% end
 tlabel =32000;
for i=1:n
    k=1;
    for j=1:tlabel

      if route(i,1,k)~=status(i,1,j)
        k=k+1;
          route(i,1,k)=status(i,1,j);

      end
       if route(i,2,k)~=status(i,2,j)
```

```
            k=k+1;
                route(i,2,k)=status(i,2,j);

            end
        end

end
```