湖南省人民政府学位委员会办公室
湖南省教育厅学位管理与研究生教育处

2017 湖南省研究生数学建模竞赛参赛承诺书

我们仔细阅读了湖南省研究生数学建模竞赛的竞赛规则.

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们授权湖南省研究生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号是（从组委会提供的试题中选择一项填写）：

我们的队号为（填写完整的队号）：

所属学校（请填写完整的全名）：

参赛队员（打印并签名）：

1. 李成鑫
2. 王梦兰
3. 赖恪

指导教师或指导教师组负责人(打印并签名)：

日期：　　年　　月　　日

评阅编号（由组委会评阅前进行编号）：

2017 湖南省研究生数学建模竞赛

编 号 专 用 页

评阅编号（由组委会评阅前进行编号）：

评阅记录（可供评阅时使用）：

评阅人										
评分										
备注										

湖南省第三届研究生数学建模竞赛

题 目 **基于条件约束的协同探测问题的求解**

摘 要：

智慧考古的协同探测问题主要是在给定探测点分布的情况下，通过建立模型来刻画探测车的初始布局、行走路线、扫描次序、所需时长和探测车数量的关系。进而求解使得探测车完成任务的效率最高时的探测车数量。最后求解出探测车的位置随时间的变化，以及何时向何车发布何指令。本题需要完成上下两个半区，320 列探测点的扫描工作。

问题 1：本题分为两种情况考虑。我们根据半区的探测车数量是否为 320 的约数，建立了整除模型和一般模型。当半区的探测车数量为 320 的约数时，可以利用良好的对称性采用简便的算法来求解整除模型。当半区的探测车数量不是 320 的约数时，需要设计更加复杂的算法来进行求解一般模型。当探测车数量为 12 台时，由于 6 不是 320 的约数，所以一般模型的计算不容易通过编程实现。基于对称以及均分的原则，同时考虑各探测车之间的相互影响，分析并计算得 12 台探测车所需时长为 7062s。其余整除模型的情况均可由计算机编程进行仿真。当探测车数量为 20 台时，所需时长为 5766s。当探测车数量为 64 台时，所需时长为 4818s。当探测车数量为 80 台时，所需时长为 4830s。对于每种情况，我们都给出了具体的扫描方案，包括探测车的初始布局、行走路线、扫描次序等，并画出相应的示意图。

问题 2：探测车完成任务的效率最高时，需要有良好的对称性和均分性，适用整除模型。故我们考虑半区探测车数量为 320 的约数的情况。此时计算量较小，编程较容易实现。通过仿真计算得到，所需时长随着探测车数量的增加而改变，但并非车数越多，时长越短。当车数较少时，车数增加，用时减少；当车数过多时，车数增加，用时增加。所以存在车数的最优值。根据问题 1 的求解过程，我么画出了所需时长随探测车数量变化的曲线图，从而找到用时最短对应的探测车数量。当探测车数量为 128 台时，探测效率最高，完成探测所需的总时间最短，为 4542s。我们给出了具体的扫描方案，包括探测车的初始布局、行走路线、扫描次序等。

问题 3：在运行仿真程序时，分别将问题 1 和问题 2 的求解过程中的数据输出，即可得到结果。输出的结果有时刻（s），探测车序号，坐标，探测车状态。为了简化编程，探测车的每个序号代表两台车，分别代表上下半区，结合坐标即可判定车在哪个半区。当探测车开始扫描某探测点时，状态记为 1。当探测车刚到达某探测点并等待时，状态记为 0。程序见附件二。附件中的 txt 文件为程序的输出结果。附件中的 xlsx 文件为添加了表头信息的结果。

应用本文方法，我们大体实现了协同探测问题的要求。

关键词：条件约束 协同探测

1 问题重述与分析

考古探测通常采用探洞、探沟、探方等实验性挖掘。其专业性强、效率低、有破坏性。现有技术已经能够实现非接触获知地下建筑的位置、大小、形状，这就是“探测车”。

探测车的工作原理像海水中的声呐设备。探测车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。用于浅表探测的探测车的成本越来越小，可实现无人操作，由计算机远程控制，未来甚至可以卫星遥感，因此本题不考虑探测车的成本和使用成本。

探测的方式是将地面上的工作区域划分为若干矩形区域，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 12 秒。

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（单位：秒）根据它们的距离 d 受到如图 1 限制。

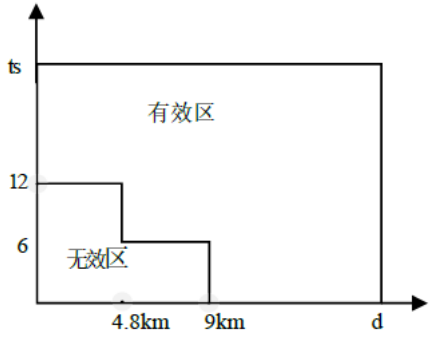

图 1 扫描限制图

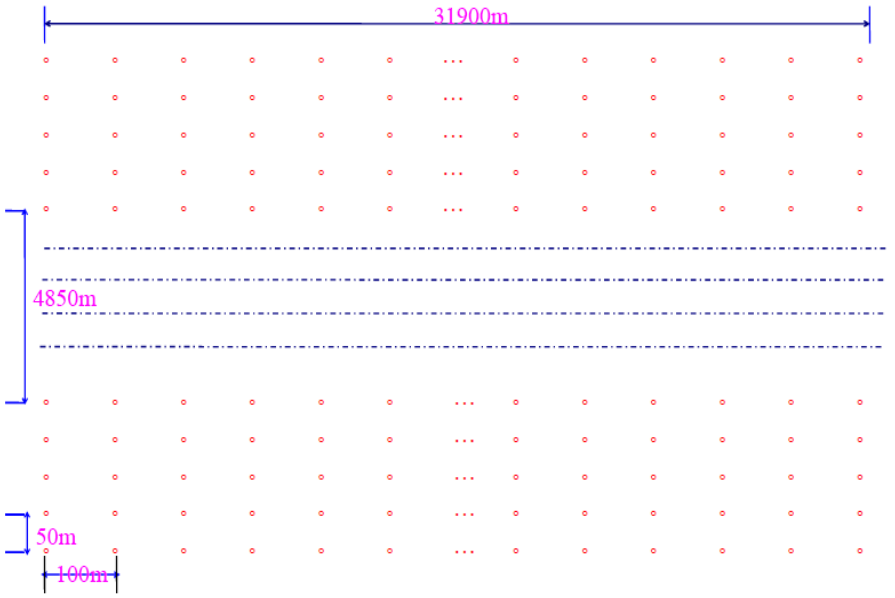

图 2 探测点分布

图 2 是一个区域的探测点分布图，探测车行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，并解决以下问题。

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。

问题 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。

问题 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？

以上是问题的重述。

对于本题，我们有以下分析。

总体上，3 个问题中的探测点分布是给定且相同的。

问题 1 要求通过建立模型来刻画探测车的初始布局、行走路线、扫描次序、所需时长和探测车数量的关系。可以预见的是，所需时长会随着探测车数量的增加而改变，但并不意味着车数越多，时长越短。当车数较少时，车数增加，用时减少；当车数过多时，车数增加，用时增加。所以存在车数的最优值。为了简化计算，探测车的初始布局的上下半区应该是对称的。行走路线走 S 型。扫描次序，我们理解为探测车的扫描次序和扫描的探测点的次序。

问题 2 要求求解使得探测车完成任务的效率最高时的探测车数量。可以根据问题 1 的求解过程，改变探测车数量，画出所需时长随探测车数量变化的曲线图，从而找到用时最短对应的探测车数量。

问题 3 要求求解出探测车的位置随时间的变化，以及何时向何车发布何指令。分别将问题 1 和问题 2 中，求解过程中的数据输出，即可得到结果。

2 假设说明

1）用于浅表探测的探测车的成本越来越小，可实现无人操作，由计算机远程控制，未来甚至可以卫星遥感，因此本题不考虑探测车的成本和使用成本。

2）假设不考虑探测点的尺寸大小，将探测点视为质点。

3）假设不考虑探测车的耗油量和用电量。

4）假设探测车在工作中不发生意外情况从而导致工作停止，影响进度。

5）假设所有探测车的时钟统一，时间尺度一致。

3 符号说明

符 号	含 义
ts	任意 2 辆探测车的扫描时间间隔（秒）
N	探测车数量（台）
d	任意 2 辆探测车的距离（米）
A_i	下半区的所有探测车
B_i	上半区的所有探测车

4 模型的建立、求解与分析

4.1 问题 1

4.1.1 模型的建立

首先，我们对探测点的分布区域建立坐标系，如图 3 所示。整个探测区域长

31900m，宽 5250m。整个区域可以分为对称的上下两个区域。每个半区有 320 列探测点，每列有 5 个探测点。

记下半区的所有探测车为 $A_i\ (i=1,2,3,\cdots)$，上半区的所有探测车为 $B_i\ (i=1,2,3,\cdots)$。

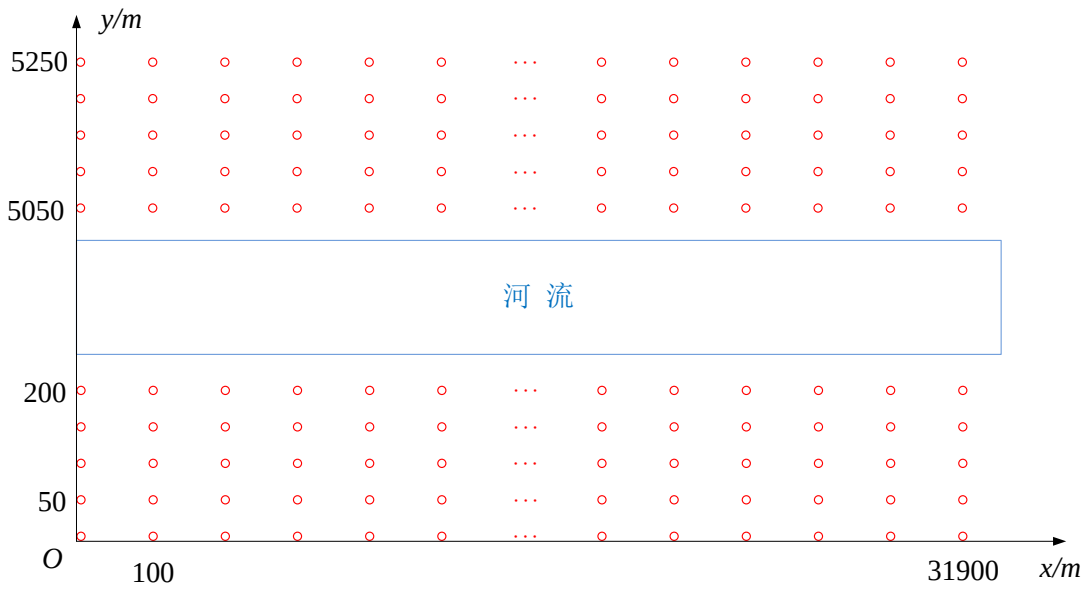

图 3 探测点分布区域建立坐标系

考虑探测车的初始布局。由于长边的长度 31900m 远大于宽边的宽度 200m，所以探测车都应该分布在长边上。同时，可以利用探测区域的对称性简化模型，为了能够同步工作，各探测车应当均匀分布在探测区域的底边上。这里我们做了两个简化处理。对于探测区域，由于上下半区完全对称且为矩形，因此我们将探测车均分给两个半区。对于探测车的初始布局，由题意知探测车会相互影响，要使每台探测车影响的范围最小，应其平均分布在各区域的边缘。

考虑探测车的行走路线。我们分析，探测车平均分布在每个区域的上边界或者下边界，行走路线都以走垂直线为最优。由于整个区域为矩形，探测车又只能沿水平或者垂直方向到达下一个节点，因此最终每台探测车扫描过的区域以近似矩形为最优。又因为相邻探测点的水平距离是垂直距离的两倍，因此行走路线以走垂直路线为最优。

考虑探测车的数量。由于探测点一共有 320 列。所以我们建立整除模型，令每个半区的探测车数量为 320 的约数。此时，每台探测车均分到的探测点的列数刚好为整数。相反地，如果每个半区的探测车数量不是 320 的约数，就会出现某一列探测点由 2 台探测车共同扫描的情况，如图 4。图中探测车数量 N=12，不是 320 的约数。

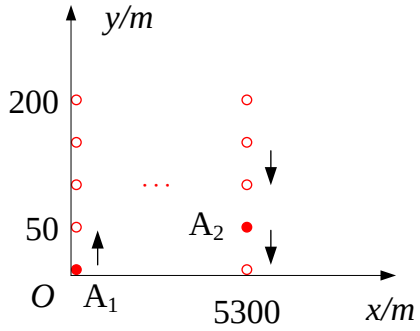

图 4 某一列探测点由 2 台探测车共同扫描情况示意图

由图 4 可知，当每个半区的探测车数量不是 320 的约数时，会有两台探测车先后扫描到同一列的探测点。并且总会出现某个探测车扫描到的点少于其他探测车，则所需时长少，存在空耗现象，要等待其他探测车完成工作，降低了整体效率。所以我们的整除模型选取的每个半区的探测车数量都为 320 的约数。

问题 1 中，探测车数量 20、64、80 都是 640 的约数，因此满足我们的整除模型。记探测车的数量为 N，要求 $N/2$ 为 320 的约数。

我们设计的算法结构如图 5 所示。在算法仿真中，时间以 6s 为间隔，依次往下走。时间每走一步，更新各个探测车的状态。当探测车到达新的探测点，判断在一定距离范围内其他探测车的状态，判断自己是否满足开始扫描的条件，输出 1 为开始扫描，输出 0 为探测车刚到达探测点并等待。

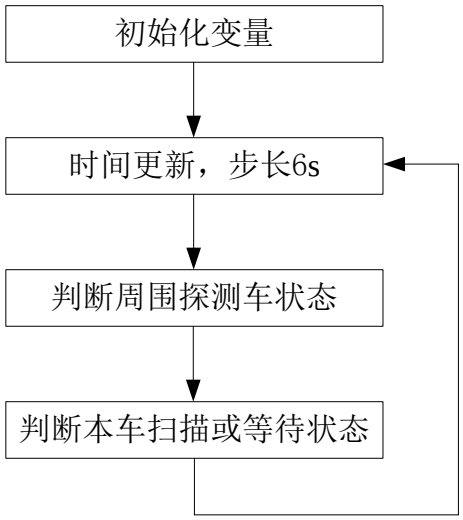

图 5 算法结构图

图 6 为任意 2 个探测车受距离影响的扫描限制的示意图。

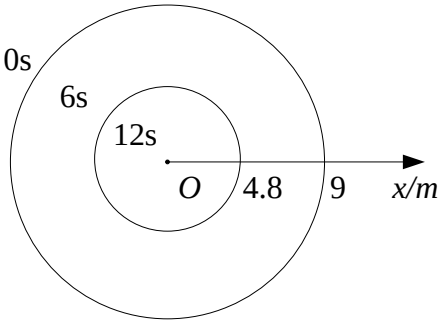

图 6 扫描等待时间随探测车距离变化示意图

4.1.2 模型的求解

先求解探测车数量为 12 台时的一般模型。

在长为 31900m 的探测点分布区域的长边上，有 320 列探测点。将其均分给 12÷2=6 台探测车，每辆车需要扫描 53.333 列，所以每两台车共同扫描一列。考虑前四台探测车，其大略位置如图。由图和图可以看到，A_1 与 B_1，A_1 与 A_2，A_2 与 B_1，三者距离都满足扫描等待时间为 6 秒的情况。假设 A_1 在 $ts=0$ s 开始扫描，则 B_1 应该等待 6s 再开始扫描，A_2 应该等待 12s 再开始扫描，B_2 应该等待 18s 再开始扫描。

考虑探测车 A_2 在第 54 列的具体位置。第 54 列的 5 个探测点由 A_1 与 A_2 进行扫描。由于 A_2 等待了 12s 才开始扫描，故 A_2 应该扫描 2 个点，A_1 应该扫描 3 个点。

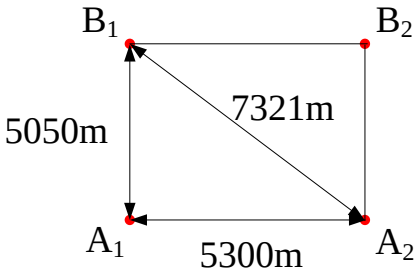

图 7 前 4 台探测车距离示意图

12 台探测车 A_1~A_6、B_1~B_6 的初始布局如图 8 所示。

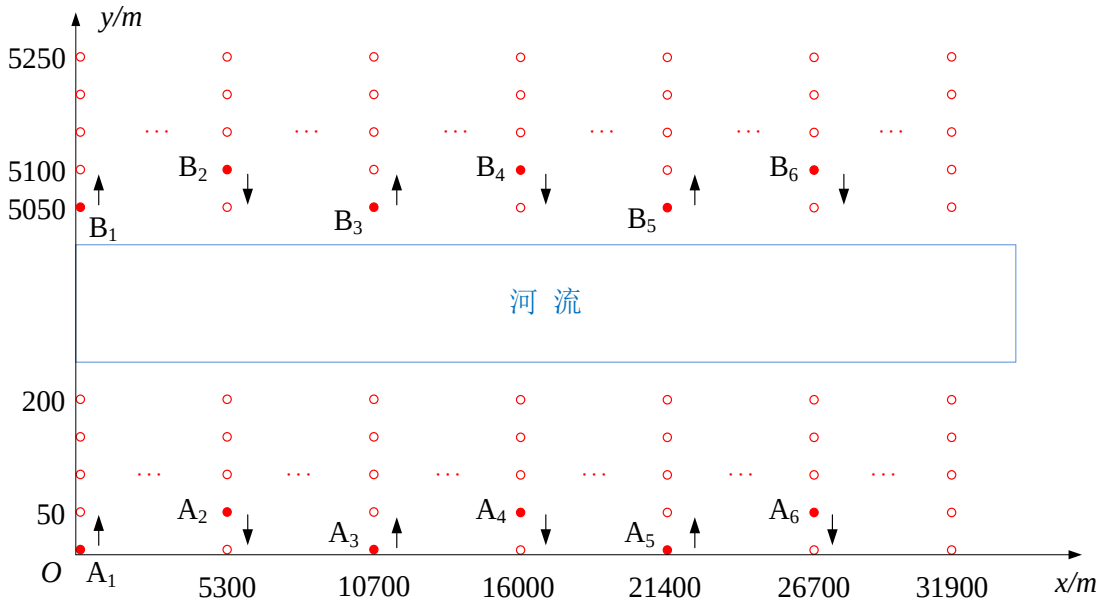

图 8 12 台探测车的初始布局

结合图 8，可得 12 台探测车的初始布局的坐标，如表 1 所示。

表 1 12 台探测车初始布局坐标

探测车序号	A_1	A_2	A_3
初始布局坐标	(0,0)	(5300,50)	(10700,0)
探测车序号	A_4	A_5	A_6

初始布局坐标	(16000,50)	(21400,0)	(26700,50)
探测车序号	B_1	B_2	B_3
初始布局坐标	(0,5050)	(5300,5100)	(10700,5050)
探测车序号	B_4	B_5	B_6
初始布局坐标	(16000,5100)	(21400,5050)	(26700,5100)

12 台探测车的行走路线，如图 9 所示。探测车均走 S 型路线。

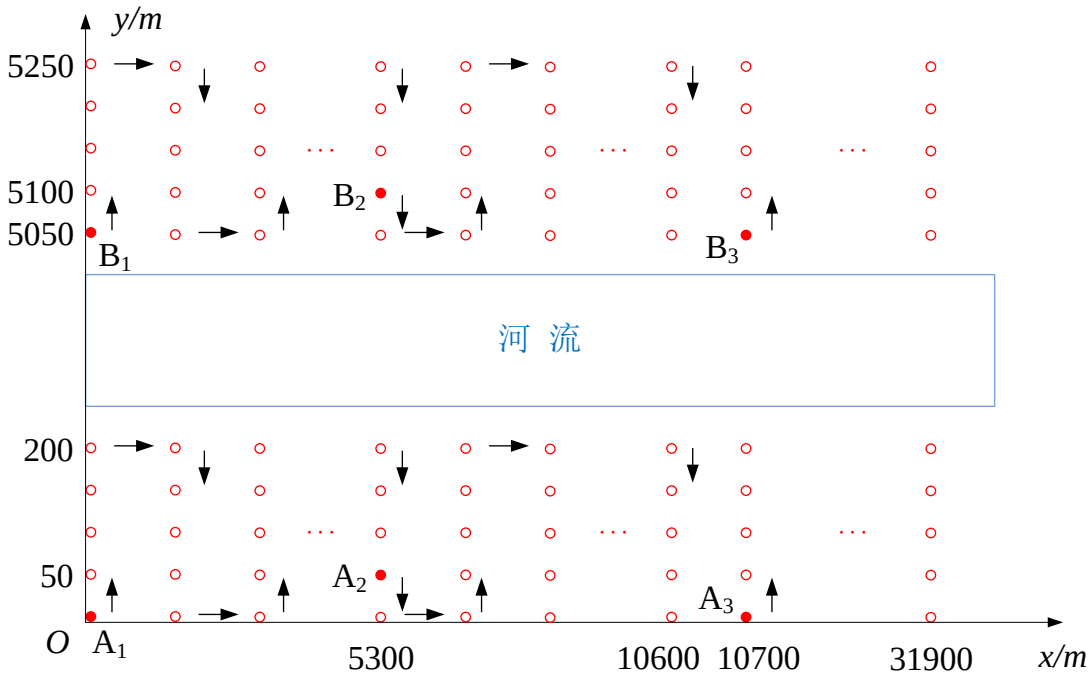

图 9　12 台探测车的行走路线

将 12 台探测车 A_1~A_6、B_1~B_6 分为 6 组，分别是 A_1 与 A_2，A_3 与 A_4，A_5 与 A_6，B_1 与 B_2，B_3 与 B_4，B_5 与 B_6。由图 9 可知，6 组的初始布局与扫描路线类似。

表 2　12 台探测车扫描开始时刻表

探测车序号	A_1	A_2	A_3	A_4	A_5	A_6	B_1	B_2	B_3	B_4	B_5	B_6
开始扫描时刻/s	0	12	0	12	0	12	6	18	6	18	6	18

由图 9 和表 2 计算得到每辆探测车的所需时长。

A_1，A_3，A_5 完成任务所需时长为

$$53\times 24+9\times 12\times 53+5\times 12=7056\,\mathrm{s}$$

A_2，A_4，A_6 完成任务所需时长为

$$53\times 24+9\times 12\times 53+3\times 12+12=7044\,\mathrm{s}$$

B_1，B_3，B_5 完成任务所需时长为

$$53\times 24+9\times 12\times 53+5\times 12+6=7062\,\mathrm{s}$$

B_2，B_4，B_6 完成任务所需时长为

$$53\times 24+9\times 12\times 53+3\times 12+18=7050\,\mathrm{s}$$

所以，当探测车数量 N=12 时，所需时长为 7062s。

求解完探测车数量为 12 台时的一般模型，再求解整除模型，可以用程序进行仿真计算。

程序输出的结果在附件中。格式如表。在程序中，探测车序号没有对上下半区进行区分，使用了相同的数字，即输出的结果中有 2 个 1 号，2 个 2 号……但

可以通过坐标进行划分，看是哪个半区。探测车状态输出 1 为开始扫描，输出 0 为探测车刚到达探测点并等待。

当探测车数量为 20 台时，每个半区有 10 台探测车，每台车要扫描 32 列探测点，刚好均分完，不会出现一列探测点由 2 辆探测车共同扫描的情况。初始布局如图 10 所示。

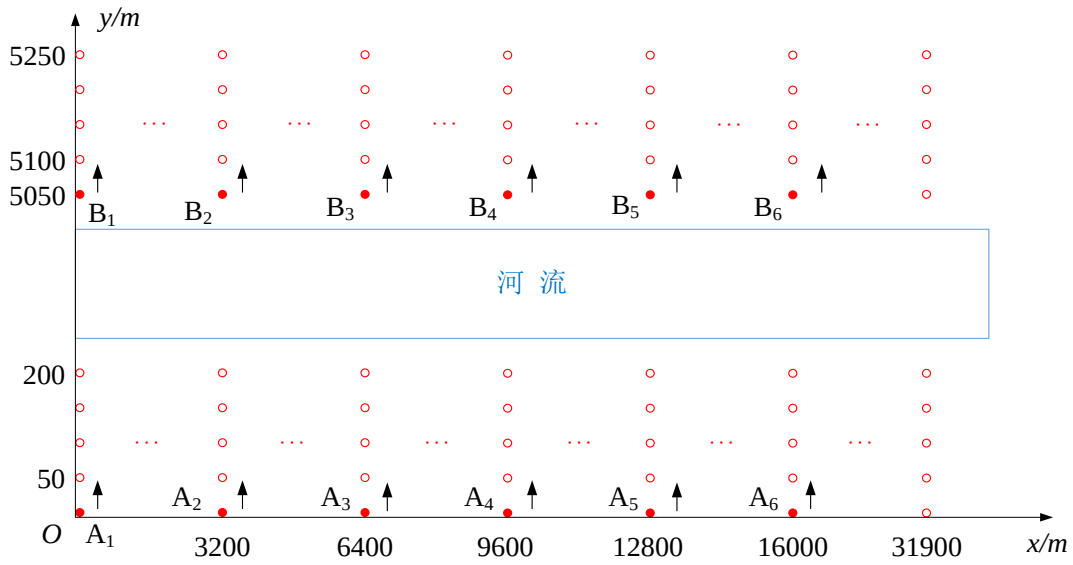

图 10　20 台探测车初始布局示意图

由图 10 可得到 20 台探测车的初始布局的位置坐标，如表 3 所示。

表 3　20 台探测车初始布局

车序号	A1	A2	A3	A4	A5
初始坐标	(0,0)	(3200,0)	(6400,0)	(9600,0)	(12800,0)
车序号	A6	A7	A8	A9	A10
初始坐标	(16000,0)	(19200,0)	(22400,0)	(25600,0)	(28800,0)
车序号	B1	B2	B3	B4	B5
初始坐标	(0,5050)	(3200,5050)	(6400,5050)	(9600,5050)	(12800,5050)
车序号	B6	B7	B8	B9	B10
初始坐标	(16000,5050)	(19200,5050)	(22400,5050)	(25600,5050)	(28800,5050)

由图 3 可知，每台探测车的行走路线均相同，为 S 型。

当探测车数量为 20 台时，所需时长为 5766s。扫描次序如表 4 所示。

表 4　20 台探测车扫描过程

时刻/s	探测车序号	*x* 坐标	*y* 坐标	探测车状态
0	1	0	0	1
0	7	9600	0	1
…	…	…	…	…
5754	12	19100	5050	1
5754	18	28700	5050	1

当探测车数量为 64 台时，所需时长为 4818s。扫描初始布局与 20 台探测车的情况类似，平均分布在上、下半区的底边。每台探测车的行走路线相同，均为 S 型。扫描次序如表 5 所示。

表 5　64 台探测车扫描过程

时刻/s	探测车序号	*x* 坐标	*y* 坐标	探测车状态
0	1	0	0	1
0	18	8000	5050	1
…	…	…	…	…
4806	33	16900	0	1
4806	50	24900	5050	1

当探测车数量为 80 台时，所需时长为 4830s。扫描初始布局与 20 台、64 台探测车的情况类似，平均分布在上、下半区的底边。每台探测车的行走路线相同，均为 S 型。扫描次序如表 6 所示。

表 6　80 台探测车扫描过程

时刻/s	探测车序号	*x* 坐标	*y* 坐标	探测车状态
0	1	0	0	1
0	22	8000	5050	1
…	…	…	…	…
4818	43	17500	0	1
4818	64	25500	5050	1

4.2 问题 2

探测车完成任务的效率最高时，需要有良好的对称性和均分性，适用整除模型。故我们考虑半区探测车数量为 320 的约数的情况。此时计算量较小，编程较容易实现。程序在附件一中给出。

通过仿真计算得到，所需时长随着探测车数量的增加而改变，但并非车数越多，时长越短。当车数较少时，车数增加，用时减少；当车数过多时，车数增加，用时增加。所以存在车数的最优值。

根据问题 1 的求解过程，我们改变了整除模型中探测车的数量，通过程序仿真计算出不同探测车数量对应的所需时长，如表 7 所示，并画出了所需时长随探测车数量变化的曲线图，如图 11 和图 12 所示。图 12 是图 11 的局部细节图。

表 7　所需时长与探测车数量的关系

探测车数量	2	4	8	10	16	20	32
所需时长	42222	21102	10542	8442	6258	5766	5412
探测车数量	40	64	80	128	160	320	640
所需时长	4818	4818	4830	4542	4614	4650	4686

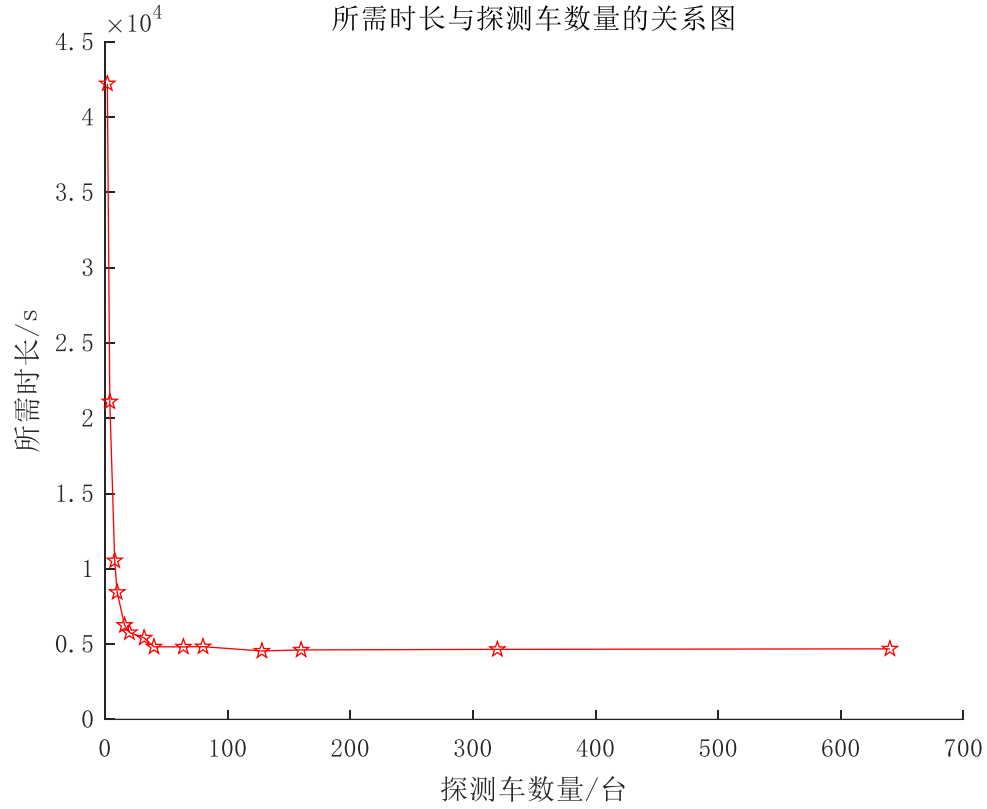

图 11　所需时长与探测车数量关系图

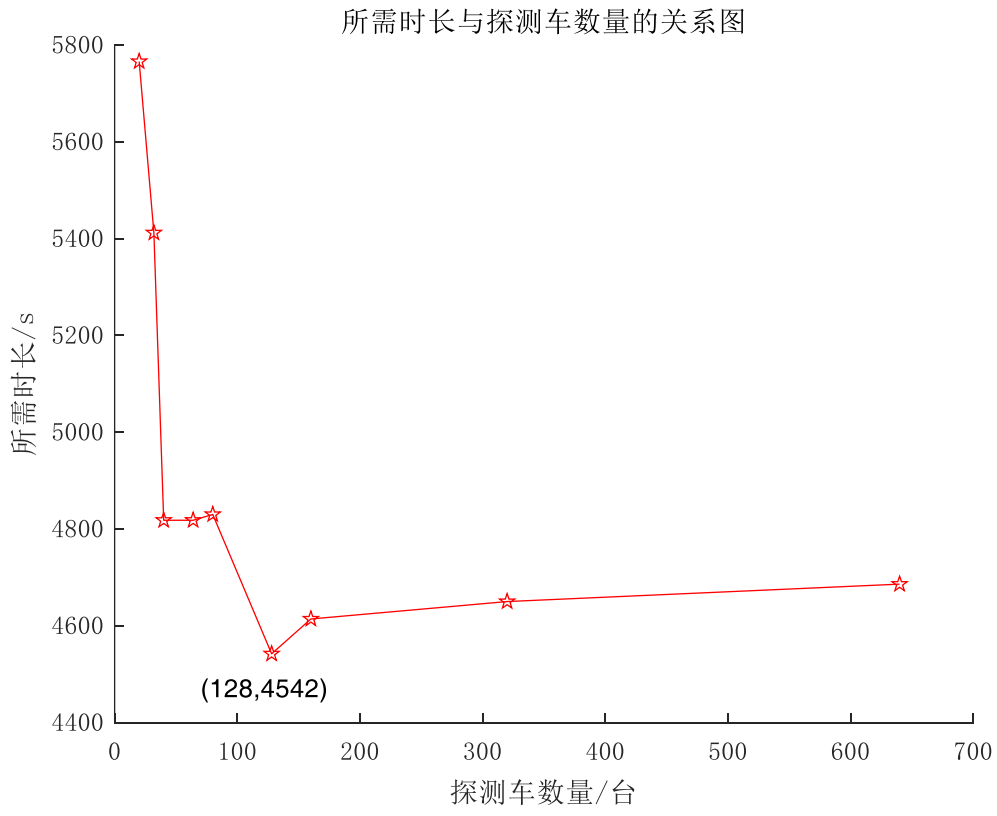

图 12 所需时长与探测车数量关系局部细节图

由图 12 可得用时最短对应的探测车数量。当探测车数量为 128 台时，探测效率最高，完成探测所需的总时间最短，为 4542s。扫描初始布局与 20 台、64 台、80 台探测车的情况类似，平均分布在上、下半区的底边。每台探测车的行走路线相同，均为 S 型。扫描次序如表 8 所示。

表 8　128 台探测车扫描过程

时刻/s	探测车序号	*x* 坐标	*y* 坐标	探测车状态
0	1	0	0	1
0	32	7500	5050	1
…	…	…	…	…
4530	66	16400	5250	1
4530	95	23900	200	1

4.3 问题 3

问题 3 要求求解出探测车的位置随时间的变化，以及何时向何车发布何指令。

在运行仿真程序时，分别将问题 1 和问题 2 的求解过程中的数据输出，即可得到结果。输出的结果有时刻（s），探测车序号，坐标，探测车状态。为了简化编程，探测车的每个序号代表两台车，分别代表上下半区，结合坐标即可判定车在哪个半区。当探测车开始扫描某探测点时，状态记为 1。当探测车刚到达某探测点并等待时，状态记为 0。程序见附件二。附件中的 txt 文件为程序的输出结果。输出结果的格式在问题 1 和问题 2 中已经给出。附件中的 xlsx 文件为添加了表头信息的结果。程序在附录二中。

6 模型的优缺点

我们根据探测车数量的数字特点，建立了一般模型和整除模型。

整除模型利用探测区域的对称性，极大简化了计算，减少了计算量，并最终得到效率最高的探测车数量和扫描方案。

缺点是整除模型不适用与每个半区的探测车数量不是 320 的约数的情况。本文对问题的分析和求解过于理想化，分析得不够透彻。

附 录

```
//附录一
//计算不同的探测车数量对应的所需时长
    #include<cstdio>
    #include<cstdlib>
    #include<algorithm>
    #include<iostream>
    #include<string.h>
    #include<vector>
    #include<string>

    using namespace std;
    const int N=640;
```

```
int x[6400+10],y[6400+10],t[N+10];
bool vi[N+10];
class issue
{
    public:
    int id,x,y,t;
    bool turn_on;
    issue(int idd,int xx,int yy,int tt,bool ok)
    {
        id=idd;
        x=xx;
        y=yy;
        t=tt;
        turn_on=ok;
    }
};
long long dis2(int x1,int y1,int x2,int y2)
{
    return 1ll*(x1-x2)*(x1-x2)+1ll*(y1-y2)*(y1-y2);
}
bool ok(int i,int tt,int k,int n,int m)
{
    if(tt<t[i]+((k%5==1)+2)*12) return 0;
    for(int j=1;j<=n;++j)
    {
        if(j==i) continue;

if(dis2(x[(i-1)*5*m+k],y[(i-1)*5*m+k],x[(j-1)*5*m+k],y[(j-1)*5*m+k])<1ll*4800*4
800&&abs(tt-t[j])<12) return 0;

if(dis2(x[(i-1)*5*m+k],y[(i-1)*5*m+k],x[(j-1)*5*m+k],y[(j-1)*5*m+k])<1ll*9000*9
000&&abs(tt-t[j])<6) return 0;
    }
    return 1;
}
string Tostring(int n)
{
    string s;
    s.clear();
    while(n)
    {
        s=s+(char)('0'+n%10);
```

```
            n/=10;
        }
        return "result"+s+".txt";
    }
    int work(int n)
    {
        int m=640/n;
        memset(x,0,sizeof(x));
        memset(y,0,sizeof(y));
        vector<issue> data;
        data.clear();
        for(int i=1;i<=5;++i) y[i]=(i-1)*50,y[5*m+i]=(i-1)*50+5050;
        for(int i=2;i<=m;++i)
            for(int j=1;j<=5;++j)
            {
                x[(i-1)*5+j]=(i-1)*100; y[(i-1)*5+j]=y[(i-2)*5+6-j];
                x[5*m+(i-1)*5+j]=(i-1)*100;
y[5*m+(i-1)*5+j]=y[5*m+(i-2)*5+6-j];
            }
        for(int i=m*10+1;i<=3200;++i)
            x[i]=x[i-m*10]+m*100,y[i]=y[i-m*10];
        memset(t,210,sizeof(t));
        int ans=0;
        for(int k=1;k<=5*m;++k)
        {
            memset(vi,1,sizeof(vi));
            for(int rest=n,tt=ans;rest;++tt)
            {
                for(int i=1;i<=n;++i)
                    if(vi[i]&&ok(i,tt,k,n,m)) t[i]=tt,vi[i]=0,--rest;
            }
            ans=t[1];
            for(int i=1;i<=n;++i)
            {
                data.push_back(issue(i,x[(i-1)*5*m+k],y[(i-1)*5*m+k],t[i],1));
                if(k<5*m)
data.push_back(issue(i,x[(i-1)*5*m+k+1],y[(i-1)*5*m+k+1],t[i]+((k%5==0)+2)*12,0
));
            }
            for(int i=1;i<=n;++i) ans=min(ans,t[i]);
        }
        for(int i=1;i<=n;++i) ans=max(ans,t[i]);
```

```
	//		freopen(Tostring(n).c_str(),"w",stdout);
	//		for(vector<issue>::iterator it=data.begin();it!=data.end();++it)
	//			cout<<(*it).id<<'	'<<(*it).x<<'	'<<(*it).y<<'	'<<(*it).t<<'
'<<(*it).turn_on<<endl;
	//		fclose(stdout);
		return ans+12;
	}
	int main()
	{
		freopen("result.txt","w",stdout);
		for(int i=1;i<=320;++i)
			if(320%i==0) printf("%d %d\n",2*i,work(2*i));
		return 0;
	}

//附录二
//计算不同的探测车数量，对应的扫描过程，并输出 txt 文件
#include<cstdio>
#include<cstdlib>
#include<algorithm>
#include<iostream>
#include<string.h>
#include<vector>
#include<string>

using namespace std;
const int N=640;
int x[6400+10],y[6400+10],t[N+10];
bool vi[N+10];
class issue
{
	public:
	int id,x,y,t;
	bool turn_on;
	issue(int idd,int xx,int yy,int tt,bool ok)
	{
		id=idd;
		x=xx;
		y=yy;
		t=tt;
		turn_on=ok;
	}
```

```
};
long long dis2(int x1,int y1,int x2,int y2)
{
	return 1ll*(x1-x2)*(x1-x2)+1ll*(y1-y2)*(y1-y2);
}
bool ok(int i,int tt,int k,int n,int m)
{
	if(tt<t[i]+((k%5==1)+2)*12) return 0;
	for(int j=1;j<=n;++j)
	{
		if(j==i) continue;

if(dis2(x[(i-1)*5*m+k],y[(i-1)*5*m+k],x[(j-1)*5*m+k],y[(j-1)*5*m+k])<1ll*4800*4
800&&abs(tt-t[j])<12) return 0;

if(dis2(x[(i-1)*5*m+k],y[(i-1)*5*m+k],x[(j-1)*5*m+k],y[(j-1)*5*m+k])<1ll*9000*9
000&&abs(tt-t[j])<6) return 0;
	}
	return 1;
}
bool cmp(issue a,issue b)
{
	if(a.t==b.t) return a.id<b.id;
	return a.t<b.t;
}
string Tostring(int n)
{
	string s;
	s.clear();
	while(n)
	{
		s=(char)('0'+n%10)+s;
		n/=10;
	}
	return "result"+s+".txt";
}
int work(int n)
{
	int m=640/n;
	memset(x,0,sizeof(x));
	memset(y,0,sizeof(y));
	vector<issue> data;
```

```
	data.clear();
	for(int i=1;i<=5;++i) y[i]=(i-1)*50,y[5*m+i]=(i-1)*50+5050;
	for(int i=2;i<=m;++i)
		for(int j=1;j<=5;++j)
		{
			x[(i-1)*5+j]=(i-1)*100; y[(i-1)*5+j]=y[(i-2)*5+6-j];
			x[5*m+(i-1)*5+j]=(i-1)*100; y[5*m+(i-1)*5+j]=y[5*m+(i-2)*5+6-j];
		}
	for(int i=m*10+1;i<=3200;++i)
		x[i]=x[i-m*10]+m*100,y[i]=y[i-m*10];
	memset(t,210,sizeof(t));
	int ans=0;
	for(int k=1;k<=5*m;++k)
	{
		memset(vi,1,sizeof(vi));
		for(int rest=n,tt=ans;rest;++tt)
		{
			for(int i=1;i<=n;++i)
				if(vi[i]&&ok(i,tt,k,n,m)) t[i]=tt,vi[i]=0,--rest;
		}
		ans=t[1];
		for(int i=1;i<=n;++i)
		{
			data.push_back(issue(i,x[(i-1)*5*m+k],y[(i-1)*5*m+k],t[i],1));
			if(k<5*m)
data.push_back(issue(i,x[(i-1)*5*m+k+1],y[(i-1)*5*m+k+1],t[i]+((k%5==0)+2)*12,0
));
		}
		for(int i=1;i<=n;++i) ans=min(ans,t[i]);
	}
	for(int i=1;i<=n;++i) ans=max(ans,t[i]);
	freopen(Tostring(n).c_str(),"w",stdout);
	sort(data.begin(),data.end(),cmp);
	for(vector<issue>::iterator it=data.begin();it!=data.end();++it)
		cout<<(*it).id<<' '<<(*it).x<<' '<<(*it).y<<' '<<(*it).t<<'
'<<(*it).turn_on<<endl;
	//printf("%d\n",ans);
	fclose(stdout);
	return ans+12;
}
int main()
{
```

```
    for(int i=1;i<=320;++i)
        if(320%i==0) work(2*i);
    return 0;
}
```