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）：B

我们的队号为（填写完整的队号）：201718001010

所属学校（请填写完整的全名）：国防科技大学

参赛队员（打印并签名）：

1.

2.

3.

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　　基于特殊多目标粒子群算法的智能考古探测

摘　　要：

本文在基于特殊多目标优化的粒子群算法（Special Particle Swarm Optimization,SPSO），对多台考古探测车协同工作问题进行优化。设定第一个优化目标为总探测时间，第二个优化目标为全部探测车彻底关机时间的方差，且总探测时间的优先级高于关机时间的方差，即当总探测时间相同时，粒子沿着方差小的方向进化，从而更快找到关于第一目标的全局最优解。

通过分析题目，在尽量缩短扫描时间的条件下，确定了探测车的运动规则。根据题中给出的时间限制条件和探测车位置信息，确定了探测车状态更新规则。结合运动规则与状态更新规则，对探测车状态更新过程进行模拟，求得总探测时间及相关物理量。

对于问题一，以探测车的初始布局为优化变量，采用 SPSO 算法，对探测车数目为 12、20、64、80 分别进行优化，求得最短探测时间分别为：7122s、6834s、6756s、6066s，并给出了探测车的初始位置、行驶路线和扫描次序（详见附录）。

对于问题二，定义了关于完成任务效率的衡量指标，以探测车数目为优化变量，通过枚举数目分别用 SPSO 算法优化，求得各自完成任务的效率。效率最高的探测车数目为 10 台。

对于问题三，通过调用问题一和二的结果数据，很容易对探测车实现计算机的无线远程协调控制。

关键字：SPSO 算法　探测车数目　完成任务效率

1. 问题重述

考古探测通常采用现有技术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。

时间限制条件：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 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。

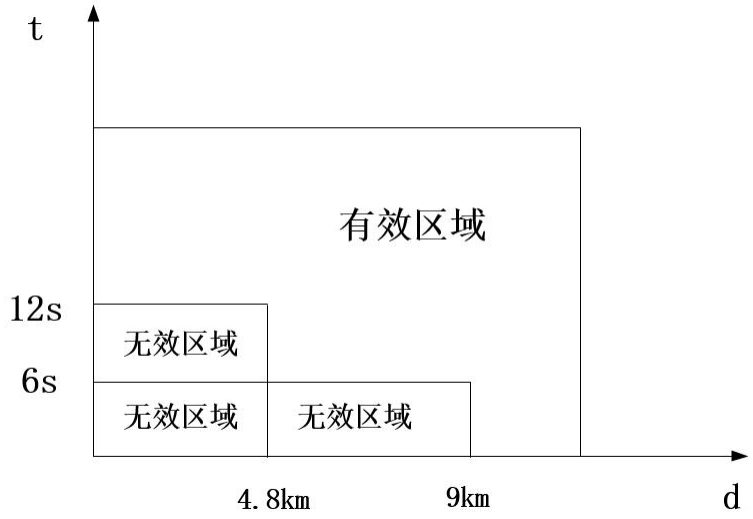


图 1　扫描限制图

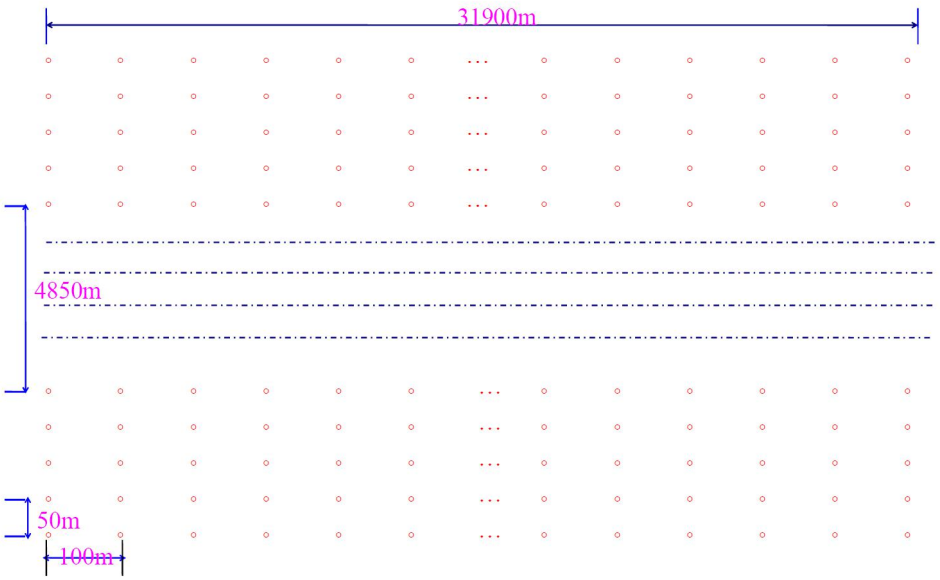


图 2　探测点分布图

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？

2. 问题一

问题一需要分别计算 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。

2.1. 问题一分析

为方便问题分析和建模，我们首先在整个区域上建立直角坐标系：以最左上角的节点作为坐标原点，水平方向为 X 轴，垂直方向为 Y 轴。区域内的每个节点都可以用一组坐标进行编号，如图 3，最左上角的节点编号为(1,1)，其它节点可分别编为(1,2),(2,1),(2,2)……下文中将直接调用编号表示节点的位置，不再赘述。

2.1.1. 探测车行驶路径

由于探测车行只能沿水平或垂直方向达到下一节点，所以探测车在完成对所在位置节点的探测后，有三个方向可以选择行驶（假设探测车不能返回已探测的节点），而在探测区域的边或和上仅有两个和一个方向可以选择行驶，如图 3。

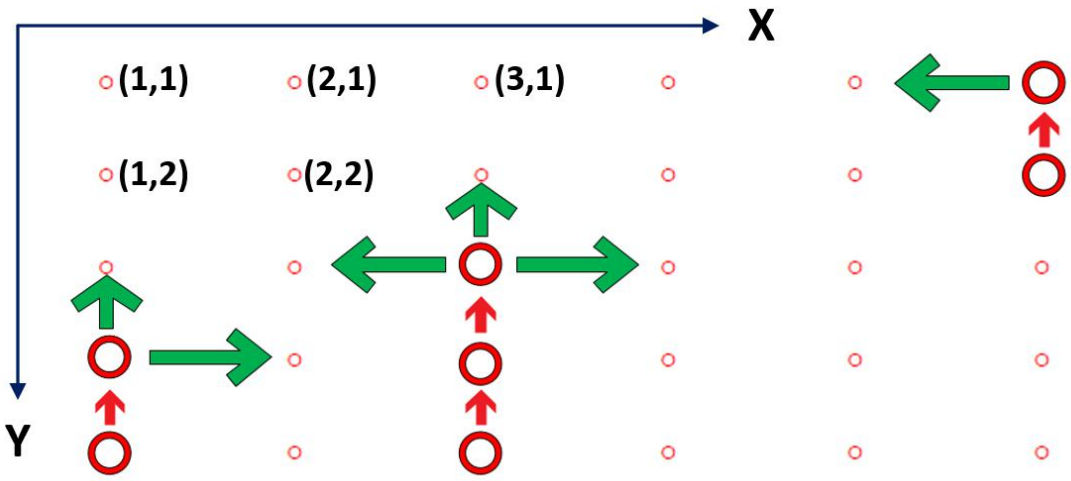


图 3　探测车行驶的方向选择

已知垂直方向上两点的距离为 50m，需要 12s 的行驶时间；水平方向上两点行驶的距离为 100m，需要 24s 的行驶时间。在本题中，我们假设探测车仅按照如图 4 所示的两种方案遍历所需探测的所有节点：方案一优先选择遍历垂直方向

上的一列节点，当到达探测区域边缘时转到下一列，然后再沿垂直方向遍历下一列所有节点，直到遍历二十个节点；方案二优先选择遍历水平方向上的一排节点，当到达探测区域边缘时转到下一排，然后再沿水平方向遍历下一排所有节点，直到遍历十二个节点。两种方案所消耗的时间分别为：

$$\text{方案一：}\ T_1 = 4\times4\times12+3\times24 = 264s$$

$$\text{方案二：}\ T_2 = 5\times3\times24+4\times12 = 408s$$

由此可以看出，方案一可以大大节省时间，所以在下文的建模中，我们假设所有探测车均按照方案一的路径方案遍历各个节点。

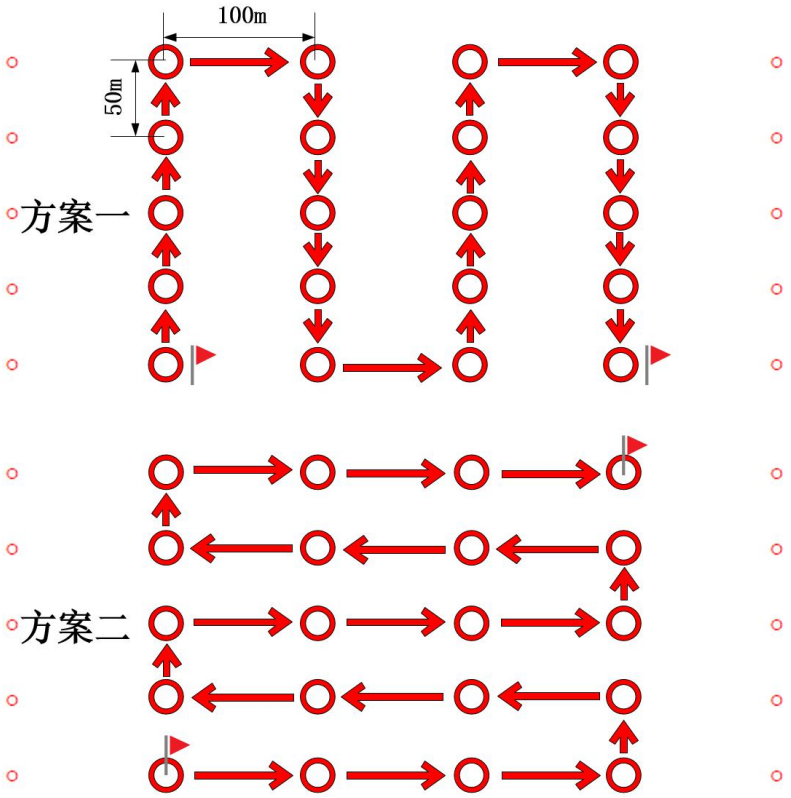


图 4　探测车遍历各节点的两种行驶方案

2.1.2. 探测车状态分析

假设探测车在遍历所有节点的过程中，总体上可以分为五个状态，见表 1。

表 1 探测车状态

状态	表示符号
探测车完成对规划区域的探测任务，彻底关机	0
探测车正在探测某个节点，处于开机工作状态，时间为 12s	1
探测车在某个未探测的节点处，但等待开机状态，时间不定	2
探测车位处于横向运动状态，时间为 24s	3
探测车位处于纵向运动状态，时间为 12s	4

由于探测车之间在工作时存在电磁干扰现象，所以本题还给出了关于探测时间的限制条件，可以概括为：如果两辆探测车之间距离间隔 4.8km 以内（本文模型中假设包括 4.8km），则第一辆车完成探测工作之后，第二辆车才可以开机进行探测，即两辆车不能在同一时刻处于 1 状态，而可以共同处于 1 与 2 或者 1 与 5 状态（时间限制条件可不考虑 3、4 状态，因为车在运动时一定处于关机状态，不会影响到其它工作状态的车）。如果两辆探测车之间距离间隔 9km 以内，则第一辆车开始开机探测的 6s 之后，另一辆车可以开机进行探测，即两辆车可以同时处于 1 状态，但开机时刻需要间隔 6s 以上。

下面结合一种简单的探测车行驶情况，说明时间限制条件对于探测车行驶和开机工作的影响，如图 5 所示。

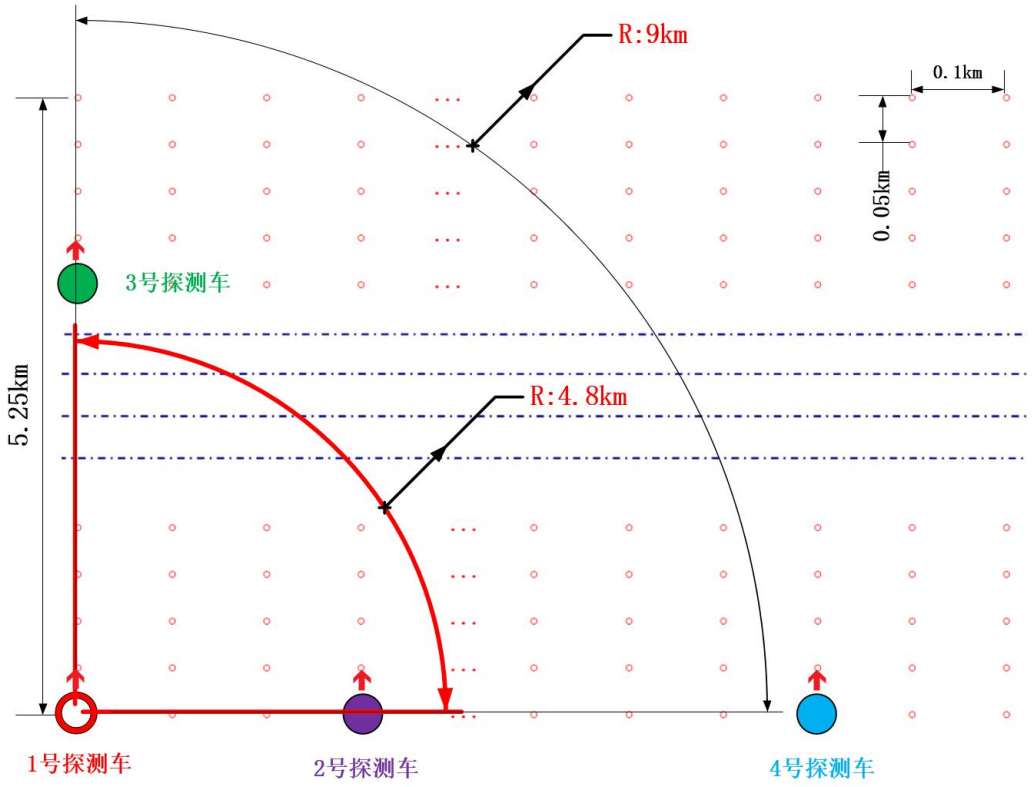


图 5 探测车工作状态分析示意图

假设 1 号探测车（位置：（1,10））在 t=0s 时开始工作（处于 1 状态），2、3、4 三辆车均处于未探测点上。以一号车位置为圆心，4.8km 和 9km 为半径画弧，2 号车位于 4.8km 覆盖范围内，3 号车位于 4.8km~9km 覆盖范围内，4 号车位于 9km 覆盖范围外。知道了四辆车的位置关系，结合时间限制条件我们可知：2 号车在 t=0s 时刻必须处于 2 状态，至少 t=12s 后才可开始工作（即至少 12s 后可由 2 状态转化为 1 状态）；3 号车在 t=0s 时刻必须处于 2 状态，至少 t=6s 后才可开始工作（即至少 6s 后可由 2 状态转化为 1 状态）；4 号车则完全不需要考虑 1 号车的工作状态，两者间距足够远，所以在 t=0s 时刻既可以处于 1 状态

又可以处于 2 状态。以上就是基于时间限制条件，对探测车工作状态的简单分析。

2.2. 问题一建模

在 2.1.中我们主要简单分析和讨论了问题一中探测车行驶路径问题，以及时间限制条件对于探测车工作状态的影响，下面给出关于问题一的具体数学建模方法。

2.2.1. 探测车初始位置与工作区域分配

每辆探测车初始位置对于我们的整个建模过程十分重要，所以我们首先需要确定探测车的初始位置。在 2.1.1.的行驶路径探究中，我们确定了探测车的行驶方案（图 4，方案一）：即优先选择遍历垂直方向上的一列节点，当到达探测区域边缘节点时向右转到下一列，然后再沿垂直方向遍历下一列所有节点，如此循环直到遍历完所有目标节点。在我们的模型中，规定所有探测车均按照此统一路径进行运动。

为了化简模型，方便计算，我们首先根据探测车的总体数量，将整个待探测区域（共 3200 个节点）较均匀的划分成若干个长方形子区域，并假设每个子区域内的探测车都是从最右上角的节点出发，例如（1,1）、（6,1）等。当有 160 台探测车时，整个待测区域可均匀划分成 160 个子区域，每个区域包含 20 个节点，根据假设，每台探测车的初始位置与行驶路径如图 6 所示。

160 台车时 3200 个点可以平均分配到 160 个子区域内，但当探测车数量不能被 3200 整除时，我们仍然需要考虑将所有节点较为均匀的分配，这样可以使得每台车的工作量大致相同，提高效率（后文将通过仿真结果，证明均匀分配原则的合理性）。

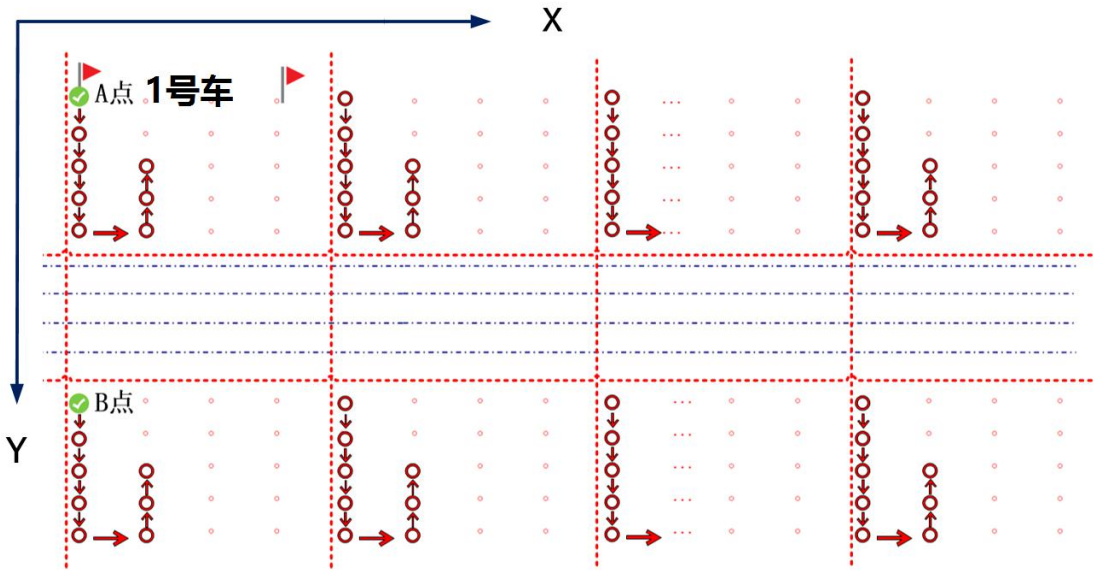


图 6　160 台探测车时子区域分配（每个子区域平均 20 个点）与探测车行驶路径

值得注意的是，在后文的建模中，我们仅假设工作区域在最左侧的两台车（分别位于河流的左右两岸）从固定的节点（图 6，AB 点，坐标位置：（1,1）、（1,6））出发，而其它车出发时 Y 坐标位置固定，但 X 坐标位置设为可调参数，形式如下：

$$(1,1)--(X_1,1)--(X_3,1)--(X_5,1)\cdots\cdots$$
$$(1,6)--(X_2,6)--(X_4,6)--(X_6,6)\cdots\cdots$$

通过建模和编程求解得到最优的位置，使整个工作时间最短。换言之，每台探测车工作的子区域原则上是尽可能平均分配，但优化后的结果会出现有些子区域内的节点较多、有些子区域的节点较少的情况。

2.2.2. 探测车运动规则

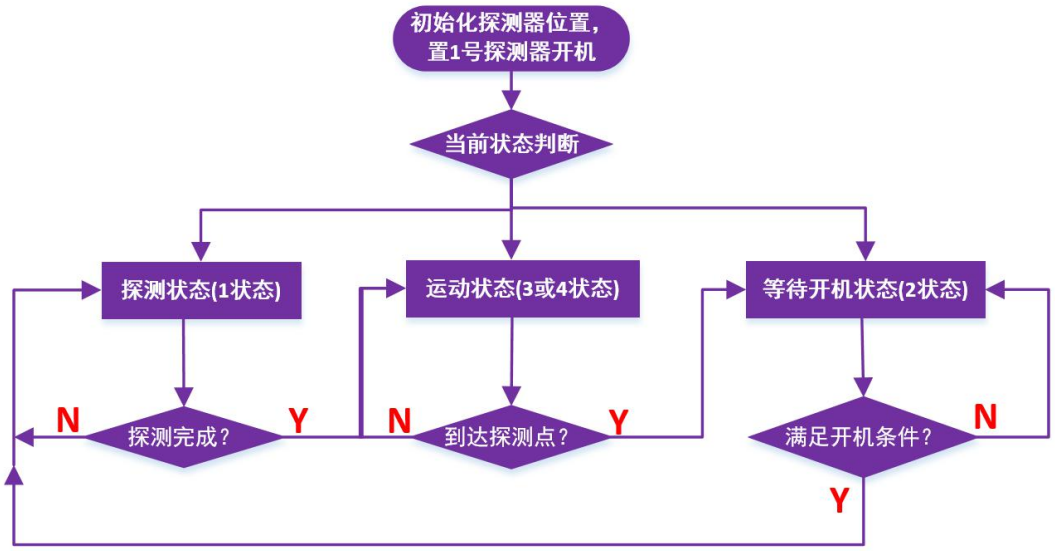
在 2.1.2.中，我们讨论了探测车在遍历所有节点时的五种状态（表一），下面详细规定探测车的运动规则。

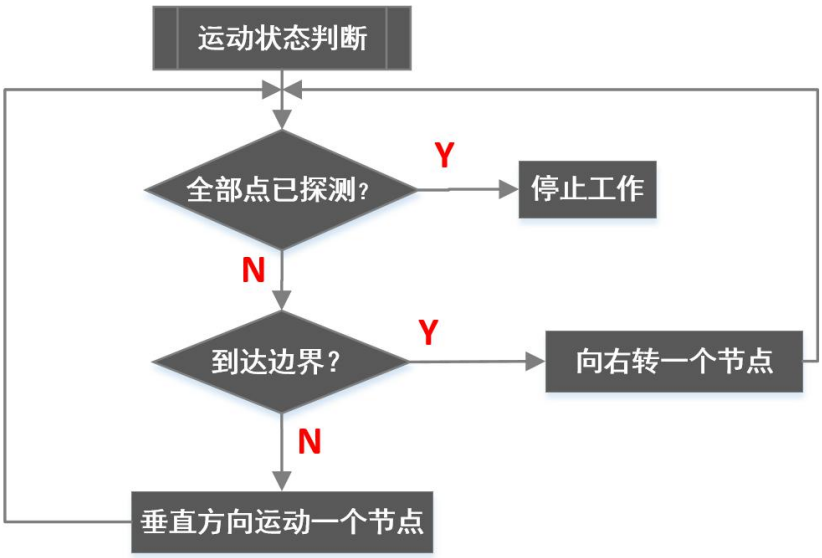
- 时间单位：题目中给出的时间度量主要包括一个节点扫描时间 12s、车在两个节点间的运动时间 12s 或 24s、时间限制条件 6s 或 12s。所以在建模中我们规定，6s 作为一个时间单位，即每隔 6s 对所有探测车的状态进行一次判断。
- 探测车的初始位置在 2.2.1.中已给出，已有两台车位置固定（图 6，AB 点），其它车辆沿 Y 方向初始位置固定，沿 X 方向的初始位置（列数）将通过**粒子群算法**进行优化，找到最短时间完成扫描任务的方案。
- 探测车初始状态判断：由于工作区域最左上侧子区域内的探测车初始位置固定（图 6，A 点），所以设定该车为**1 号探测车**，初始时刻即开始工作（1 状态），根据时间限制条件，与 1 号车的距离相距大于 9km 的、且彼此之间距离同样大于 9km 的其它探测车同样在初始时刻开始工作（1 状态）。而其它探测车初始时刻保持关机状态（2 状态），当一段时间后若满足时间限制条件则可开机工作（由 2 状态转为 1 状态）。
- 时间限制条件对于探测车的状态起着决定性因素，当某台探测车达到一个未探测节点时，首先需要判断与其它全部处于工作状态（1 状态）的探测车之间的距离，只有当该车完全满足时间限制条件时，下一时刻才可以开机工作，否则须保持 2 状态。
- 规定当探测车完成对某一节点的探测任务后，下一刻立即前往下一个未探测节点，运动过程中保持关机。且处于 3 状态和 4 状态的探测车对处于 1 状态的探测车不会产生影响。
- 规定当一台探测车完成其划分的子区域的探测任务后，将自动停止工作并关

机，处于 0 状态。

- 探测车完成整片区域（3200 个节点）扫描任务所用的时间是指从第一台探测车开始工作到最后一台探测车完成扫描转为 0 状态所消耗的时间。
- 在建模过程中，考虑当探测车数目改变时，每台探测车行驶路径方案是固定的，而每台探测车的初始位置和初始状态将会随着车数目的变化而变化。
- 若某一节点已被探测，则该点不会被任何探测车再次扫描。

以上是关于探测车运动规则的详细规定，在运用 MATLAB 进行软件编程时，具体程序逻辑框图如下：





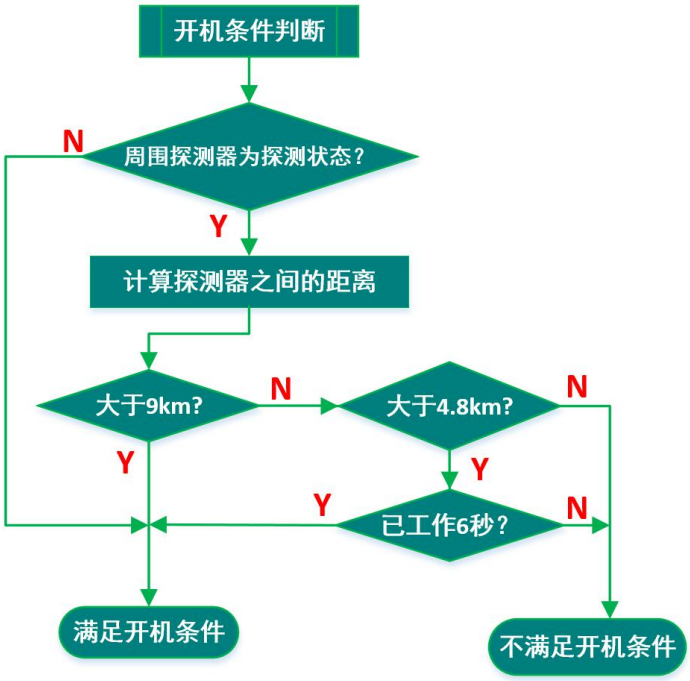


图 7　探测车运动情况判断逻辑框图

2.3.　12 台探测车的求解

2.3.1.　固定初始位置求解模型

通过 2.2.中建立的模型，我们首先对 12 台探测车的情况进行求解。为了证明模型的正确性，首先将每台车的初始位置（包括 X、Y 方向坐标）全部固定，求解每台探测车的运动情况和完成全部 3200 个节点扫描所需要的时间。假设 12 台车的初始坐标位置如下：

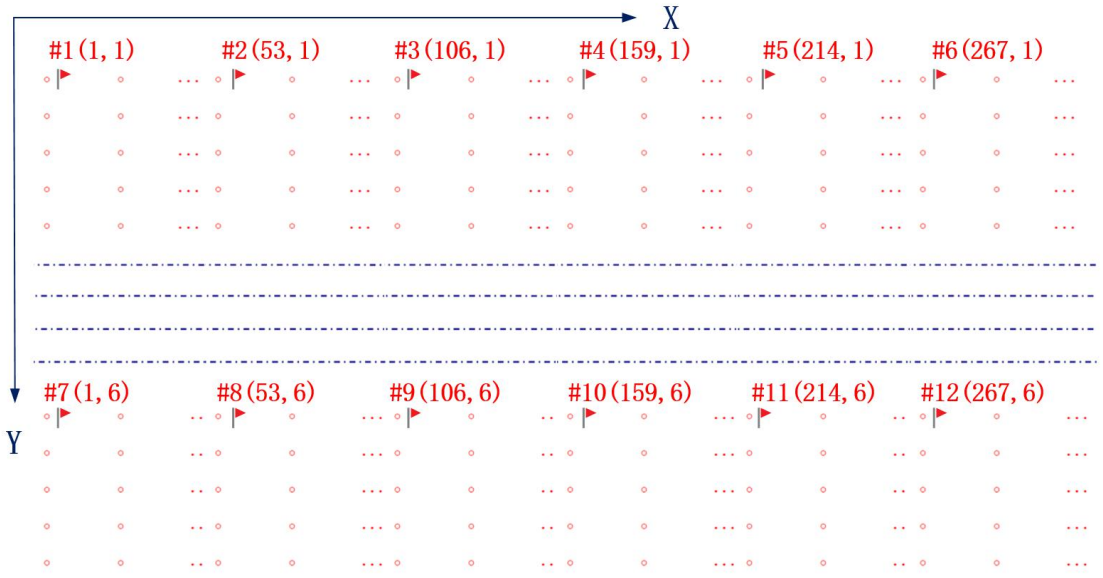


图 8　探测车初始时刻位置示意图

初始时刻，1 号探测车设置为 1 状态，其它车的状态通过时间限制条件判定，运行程序得到 12 台车的运动情况，如表 2：

表 2：探测车运动状态反映表

时刻（6s）	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
1	2	2	2	2	2	2	2	2	2	2	2	2
2	1	2	1	2	1	2	2	2	2	2	2	2
3	1	1	1	1	1	1	2	2	2	2	2	2
4	4	1	4	1	4	1	1	2	1	2	1	2
5	4	4	4	4	4	4	1	1	1	1	1	1
6	1	4	1	4	1	4	4	1	4	1	4	1
7	1	1	1	1	1	1	4	4	4	4	4	4
…						……						
1140	1	4	1	4	1	4	4	1	4	1	4	1
1141	1	1	1	1	1	1	4	4	4	4	4	4
1142	0	1	3	1	3	1	1	4	1	4	1	4
1143	0	3	3	3	3	3	1	1	1	1	1	1
1144	0	3	3	3	3	3	0	1	3	1	3	1
1145	0	3	3	3	3	3	0	3	3	3	3	3
…						……						
1206	0	0	0	4	0	0	0	0	0	1	0	0
1207	0	0	0	1	0	0	0	0	0	4	0	0
1208	0	0	0	1	0	0	0	0	0	4	0	0
1209	0	0	0	0	0	0	0	0	0	1	0	0
1210	0	0	0	0	0	0	0	0	0	1	0	0
1211	0	0	0	0	0	0	0	0	0	0	0	0

探测车状态 1 到 4 的具体含义，已由表 1 给出说明。启动前，所有车都处于 2 状态等待开机工作，表 2 第二行可看出，初始时刻规定 1 号探测车开始工作，可得 3 号车和 5 号车也同时开始工作。第 $(1142-1)\times 6=6852\text{s}$ 时，1 号车完成对分配的子区域内所有节点的探测任务，回归 0 状态。随后其它号车陆续完成探测任务，回归 0 状态。第 $(1211-1)\times 6=7260\text{s}$ 时，10 号车完成探测任务，此刻所有探测车均处于 0 状态，整个区域（3200 个节点）扫描完毕。

由此可得，当 12 台车初始时刻位置固定（图 8）、行驶路径确定后，探测完全部节点所用的时间为 7260s。

2.3.2. 基于特殊多目标粒子群算法的模型求解

粒子群算法基于群体智能，采用全局搜索策略寻优。每个粒子根据全局最优位置和自身最优位置更新速度。粒子群算法相对遗传算法具有更加高效的信息共享机制，即启发性更强。因此，理论上粒子群算法收敛速度更快、全局搜索能力更强。

设目标搜索空间为 D 维，粒子总数为 N，总迭代次数为 T。第 i 个粒子在 D 维空间的位置矢量为 $X_i=(x_{i1},x_{i2},\ldots,x_{iD})$，飞行速度矢量为 $V_i=(v_{i1},v_{i2},\ldots,v_{iD})$，第 i 个粒子的历史最优位置为 $P_i=(p_{i1},p_{i2},\ldots,p_{iD})$，全局历史最优位置为 $P_g=(p_{g1},x_{g2},\ldots,x_{gD})$。根据如下公式更新粒子速度和位置：

$$\begin{cases} v_{id}(t+1)=w(t)\cdot v_{id}(t)+c_1\cdot R_1[p_{id}-x_{id}(t)]+c_2\cdot R_2[p_{gd}-x_{gd}(t)] \\ x_{id}(t+1)=x_{id}(t)+v_{id}(t+1) \\ x_{id}(0)=x_d^{low}+R_3(x_d^{up}-x_d^{low}) \\ d=1,2,\ldots,D;i=1,2,\ldots,I;t=1,2,\ldots,T \end{cases}$$

式中，w 为惯性因子，c_1 和 c_2 为加速因子（常数），$R_1R_2R_3$ 为[0,1]上的随机数。迭代中若 x_iv_i 超出边界，则取边界值。为保证算法有较好的收敛性和搜索能力，可令 w 为随迭代次数变化的线性减小函数。对于该多脉冲转移优化问题，适应度函数 F 即目标函数 Δv，粒子群算法流程图见图 9 所示。

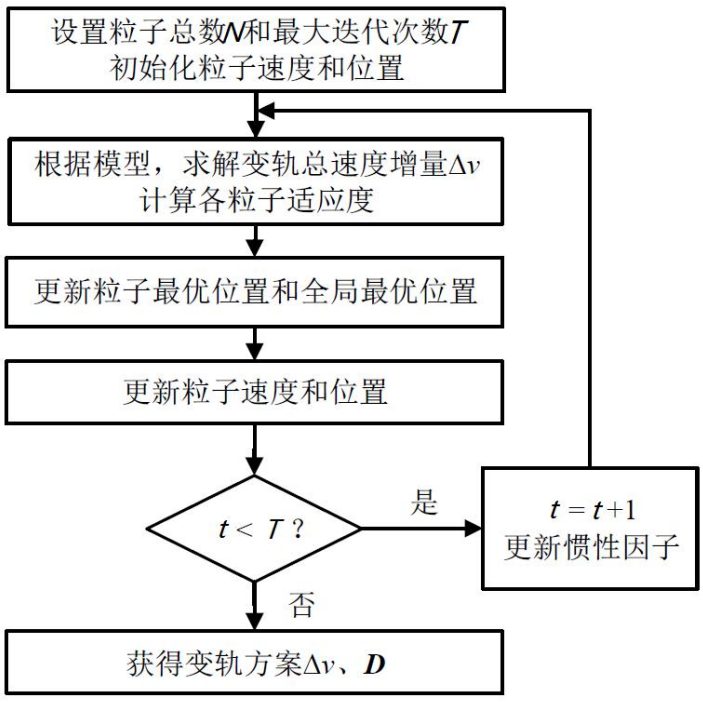


图 9　粒子群算法流程图

为了实现对扫描时间的优化，设定探测车初始时刻位置为自变量，采用粒子群算法，优化目标设置如下表 3：

表 3 优化目标设置

	第一优先级	第二优先级
优化目标：	总探测时间	探测车关机时间方差

算法思想：在运算过程中首先判断粒子的总探测时间是否减小，若减小则更新粒子最优状态，若总探测时间相同，则继续判断关机时间方差，若方差减小则继续更新粒子最优状态，经过多次迭代得到最终可以得到最短的总探测时间，并得到相应每台探测车初始时刻位置。迭代结果见图 10，探测车初始位置见表 4，探测车运动状态见表 5。

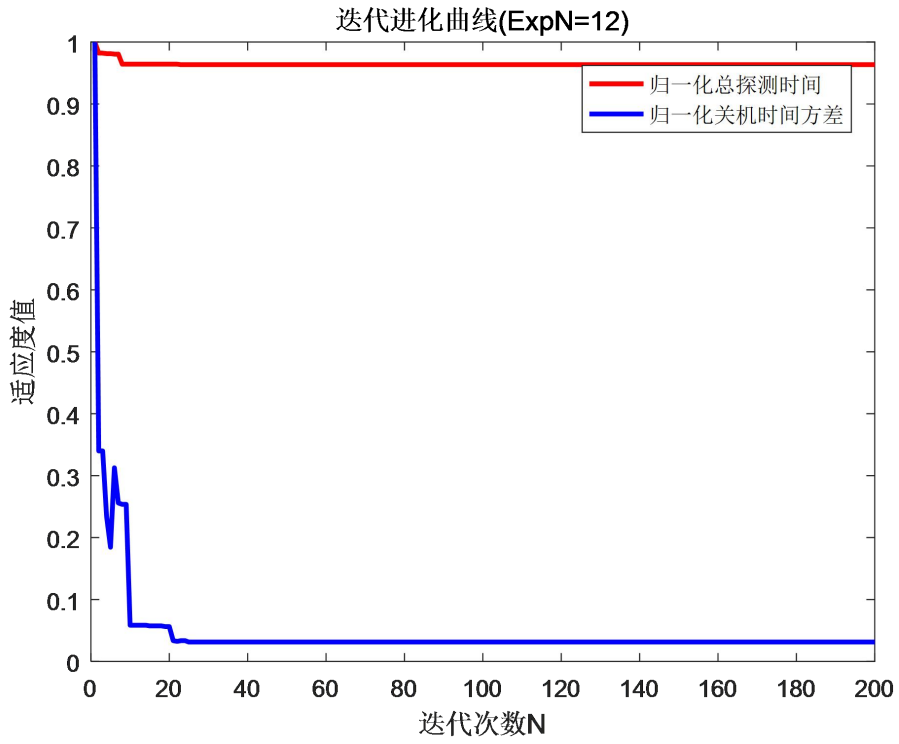


图 10　粒子群算法迭代结果

为统一单位我们对目标函数进行归一化处理，图 10 显示目标函数逐渐收敛，当迭代到一定次数后，目标函数达到最优值。根据表 4、表 5 中显示的优化结果，我们最终得到了 12 台探测车的初始位置，以及完成任务总时间的最小值：$(1188-1)\times 6=7122\text{s}$。

表 4 优化后的初始时刻位置表

探测车	初始位置	探测车	初始位置
#1	（1,1）	#7	（1,6）
#2	（55,1）	#8	（54,6）
#3	（108,1）	#9	（107,6）
#4	（161,1）	#10	（161,6）
#5	（214,1）	#11	（214,6）
#6	（268,1）	#12	（268,6）

表 5 优化后的运动状态反映表

时刻（6s）	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
1	2	2	2	2	2	2	2	2	2	2	2	2
2	1	2	1	2	1	2	2	2	2	2	2	2
3	1	1	1	1	1	1	2	2	2	2	2	2
4	4	1	4	1	4	1	1	2	1	2	1	2
5	4	4	4	4	4	4	1	1	1	1	1	1
6	1	4	1	4	1	4	4	1	4	1	4	1
7	1	1	1	1	1	1	4	4	4	4	4	4
…	……											
1181	4	0	0	0	4	0	0	4	1	0	0	0
1182	4	0	0	0	4	0	0	1	1	0	0	0
1183	1	0	0	0	1	0	0	1	4	0	0	0
1184	1	0	0	0	1	0	0	4	4	0	0	0
1185	0	0	0	0	0	0	0	4	1	0	0	0
1186	0	0	0	0	0	0	0	1	1	0	0	0
1187	0	0	0	0	0	0	0	1	0	0	0	0
1188	0	0	0	0	0	0	0	0	0	0	0	0

2.4. 其它三种情况（20、64、80 台探测车）的求解

在上一小节，我们对于 12 台探测车的情况进行求解，并对每台车初始时刻位置进行优化，最终得到完成扫描任务最快需 7122 秒。下面求解其它三种情况，我们假设探测车的行驶路径仍采用 2.1.1.中的方案一，最左侧的两台车初始时刻位置固定，仍为点（1,1）和点（1,6），且这两台车从初始时刻即开始工作（1

状态）。运用与 2.3.中相同的建模和求解方法，最终得到 20、64、80 台探测车的优化初始位置与最短总时间。

图 11 为三种情况下，运用粒子群算法得到迭代结果图。

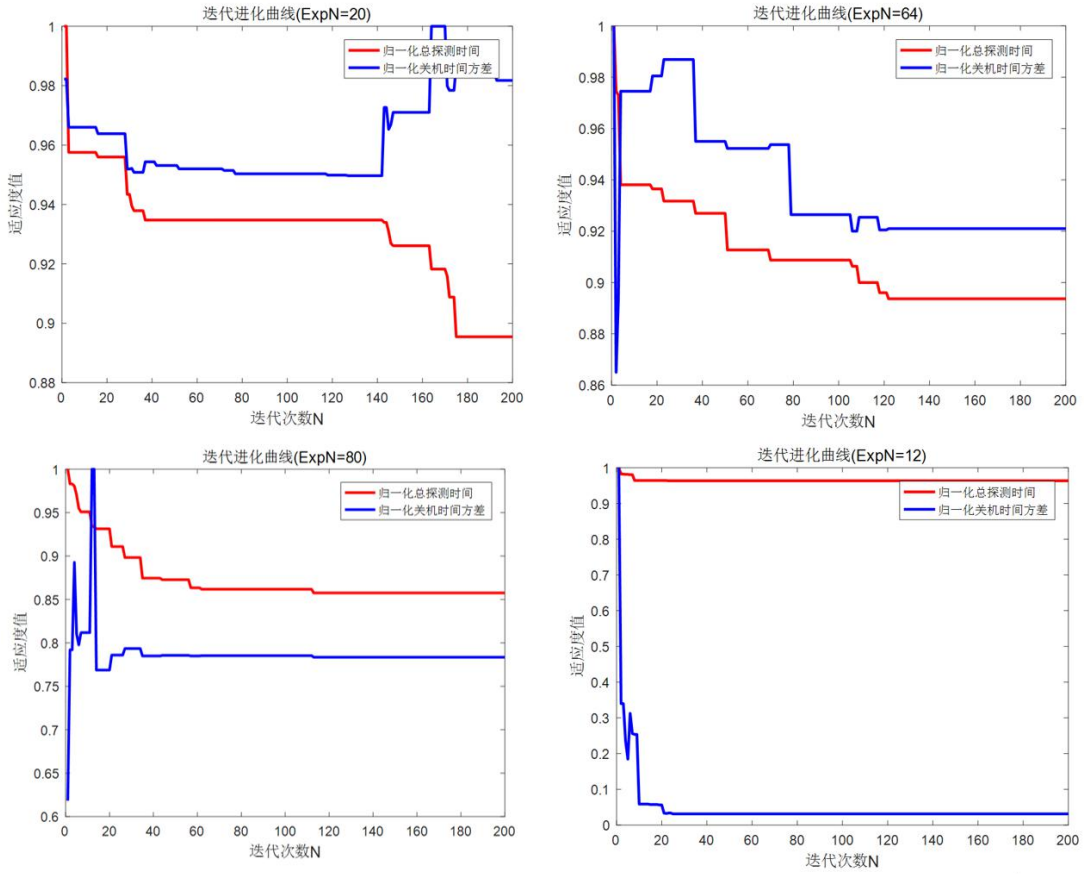


图 11　迭代结果（20、64、80、12 台探测车）

由此得到：20 台探测车时完成任务的最短时间为 **6834s**，64 台探测车时完成任务的最短时间为 **6756s**，80 台探测车时完成任务的最短时间为 **6066 s**。具体的每台探测车的初始时刻位置在附录中给出，这里不再给出表格。

从中模型一的求解结果，我们可分析得出以下结论：

- 随着探测车数量的增加，完成探测任务所需要的最短时间逐渐减少，但不服从成倍减少的趋势。这是因为探测车数量的增多，会使得它们两两之间距离减小，受时间限制条件的制约，导致许多台探测车长期处于待机状态（2 状态），所以大量增加探测车数量并不能有效缩短扫描时间。
- 探测车的初始时刻位置经过粒子群算法优化，可以看出初始时刻位置大致上是均匀分布的（例如表 3），即每台探测车分配的任务大致上是均匀的，这也与我们在 2.2.1.中提出的工作区域均匀分配原则相符合。

3. 问题二

3.1. 问题二分析

问题二要求我们确定探测车数量，使得完成扫描任务的效率最高。所以我们需要建立数学模型，以探测车数量作为自变量、完成任务效率作为目标函数，并找到目标函数的最优解。为化简计算，首先做出以下假设：

- 探测车行驶路径仍然采用 2.1.1.中的方案一，即优先选择遍历垂直方向上的节点。
- 最左侧的两台车初始时刻位置固定，仍为点（1,1）和点（1,6），且这两台车从初始时刻即开始工作（1 状态）。
- 其它假设与 2.2.2.探测车运动规则中的假设完全一致。
- 当探测车数量确定后，每台车的初始时刻位置由粒子群算法迭代得到的最优解确定，确保完成任务的时间最短。
- 假设完成任务的效率仅与探测车数目有关，不考虑其它变量对于效率的影响。

3.2. 确定目标函数

直观上考虑，如果若干台探测车完成扫描任务的总时间相同，则探测车数量越少，完成任务的效率越高。所以规定工作效率的目标函数为：

$$f = 1/\left(t^* \cdot ExpN^*\right)$$

式中 f 表示目标函数，t^* 表示归一化的探测总时间，$ExpN^*$ 表示归一化的探测车数目。f 越大，完成任务的效率越高。

3.3. 问题二求解

为缩短 MATLAB 程序的运算时间，我们在此仅选取探测车数量 $ExpN$ 分别为 10、12、20、30、40、50、64、70、80、90、100 台时的情况，通过粒子群算法可以得到不同数量探测车对应的 t（迭代结果如图 12 所示），接着再计算目标函数 f，表 6 给出了程序运算后的目标函数结果。

从表 6 中可看出，当探测车数量为 10 时，目标函数 f 值达到最大，所以安排 10 台探测车工作可以使得完成探测任务的效率最高。10 台探测车所对应的初始时刻位置和运动状态见表 7 和表 8，完成任务所用时间为 8448 秒。

表 6 目标函数结果

ExpN	$ExpN^*$	*t*	t^*	*f*
10	0.10	8448	1.331	7.514
12	0.12	7122	1.122	7.428
20	0.20	6834	1.077	4.644
30	0.30	6852	1.080	3.088
40	0.40	6378	1.005	2.489
50	0.50	5904	0.930	2.150
64	0.64	6756	1.064	1.468
70	0.70	6090	0.959	1.489
80	0.80	6066	0.956	1.308
90	0.90	6132	0.966	1.150
100	1.00	6348	1.000	1.000

表 7 初始时刻位置表

探测车	初始位置	探测车	初始位置
#1	（1,1）	#6	（1,6）
#2	（65,1）	#7	（65,6）
#3	（129,1）	#8	（129,6）
#4	（193,1）	#9	（193,6）
#5	（257,1）	#10	（257,6）

表 8 运动状态反映表

时刻（6s）	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	2	2	2	2	2	2	2	2	2	2
2	1	2	1	2	1	2	2	2	2	2
3	1	1	1	1	1	2	2	2	2	2
4	4	1	4	1	4	1	2	1	2	1
5	4	4	4	4	4	1	1	1	1	1
…					……					
1407	0	0	0	0	0	1	1	1	1	1
1408	0	0	0	0	0	0	1	0	1	0
1409	0	0	0	0	0	0	0	0	0	0

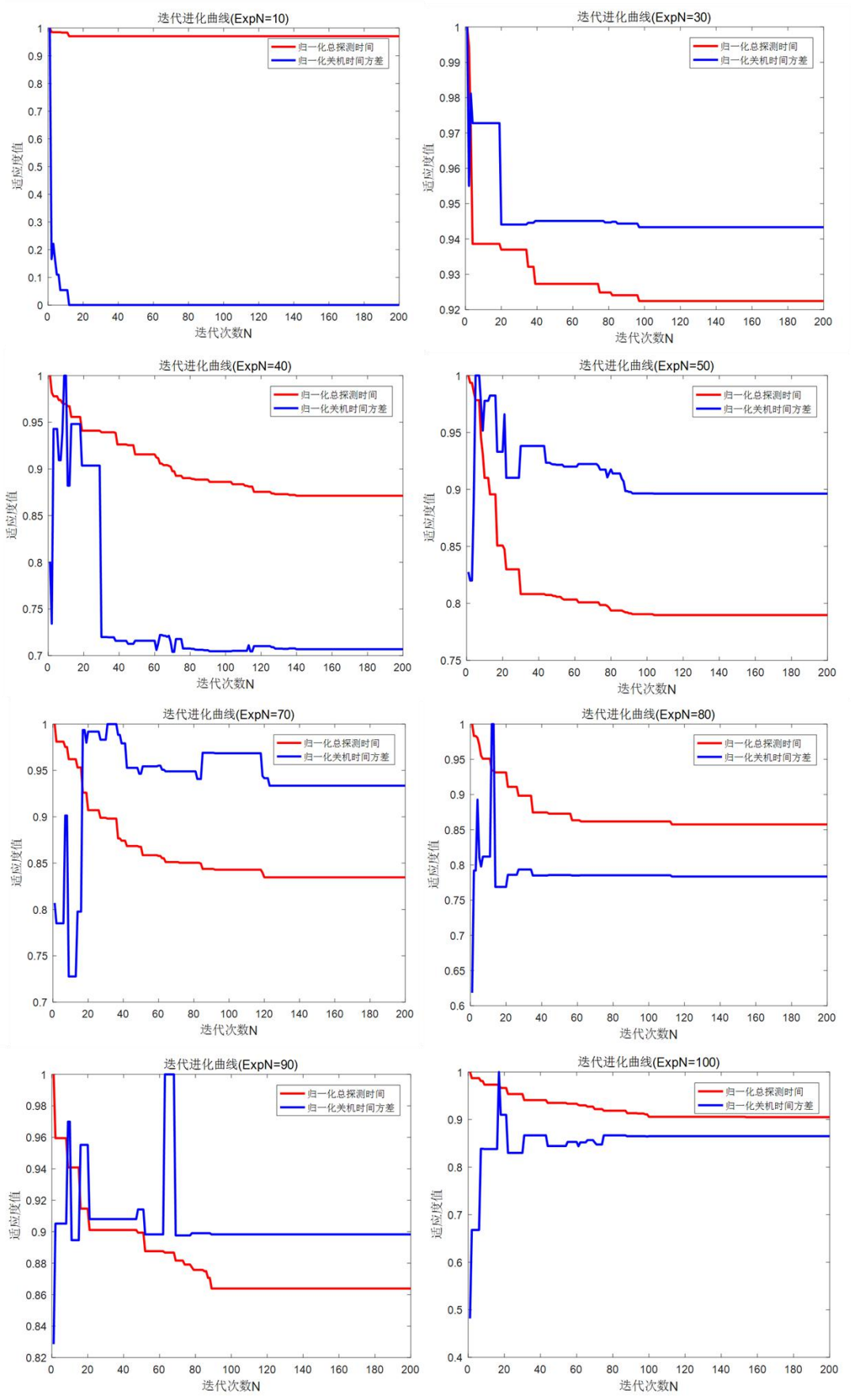


图 12　迭代结果（10、30、40、50、70、80、90、100 台探测车）

问题三

问题三需要将问题1、问题2 的结果表达成计算机能够处理的数据组织方式，以便于计算机进行无线远程协调控制。我们以问题 1 最后得到的优化结果为例，进行说明。

首先，12 台探测车在每个时刻的位置，由附件 Final Result 3 文件夹下的 ExpPosition_2.xlsx 确定 X、Y 方向坐标，对应每个时刻的工作状态则由附件 Final Result 3 文件夹下的 ExpState_2.xlsx 给出，文件说明详见附件。计算机通过调用数据库里的数据即可获得某一时刻任意一台探测车所处的坐标位置和工作状态。

接下来，计算机根据探测车当前时刻的位置和状态，下达指令控制探测车下一时刻的运动状态，如下表：

探测车状态	指令
0	彻底关机
1（上一时刻状态不为 1）	下达开机指令
1（上一时刻状态为 1）	维持开机状态
2	保持待机状态
3	横向移动
4	纵向移动

根据以上指令下达规则，即可实现计算机无线远程协调控制。

附录 A：探测车状态更新过程模拟

```
function [time,ExpCloseTime] = CalculateTime(x)
% 假设一共有 ExpN=12 个探测器
% x = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12]
% 分别表示初始探测器探测位置横坐标
% x1-x6 的 y 变化范围为 1-5，x7-x12 的 y 变化范围为 6-10
% time :探测器探测完全部区域的总时间

% 输入：x
% x = [1 53 106 159 214 267 1 53 106 159 214 267];
% x = g_full;

% 探测器数目 ExpN
ExpN = size(x,2);

% 探测区域矩阵：0 表示已经探测，1 表示未被探测
SearchPoint = ones(10,320);

% 探测器初始布局（摆放位置）
ExpPosX = x;
ExpPosY = [1*ones(1,ExpN/2) 6*ones(1,ExpN/2)];

% 计算探测器之间的距离 Distance（ExpN x ExpN 维）
Distance = CalculateDis(ExpPosX,ExpPosY); % 单位：km

% 探测器状态矩阵：
% 0 表示关机状态，所有待探测点均以探测完
% 1 表示开机工作状态，时间为 12s
% 2 表示等待开机状态，时间不定，由上一个状态和当前的其他探测器之间的距
离决定
% 3 表示处于运动状态，横向移动需 24s
% 4 表示处于运动状态，纵向移动需 12s
ExpState = 2*ones(1,12);       % 初始化为等待开机状态，赋值 2
k = 1; % 用来计算时间, k 更新一次时间为 6s
```

```
% 探测器关机时次数矩阵 ExpCloseK
% 探测器关机时间矩阵 ExpCloseTime
ExpCloseK = zeros(1,ExpN);
ExpCloseTime = zeros(1,ExpN);

% 探测器状态第 1 次更新
ExpState(k+1,1) = 1;       % 假设第一个探测器开始工作
for i = 2:ExpN
    S1 = find(ExpState(k+1,:) == 1);
    flag = 0;     % 用来判断是否存在干扰需要等待       flag=0 表示不需要等待
    for j = 1:length(S1)
        if Distance(i,S1(j))<9
            flag = 1;
        end
    end

    if flag == 1 % 需要等待
        ExpState(k+1,i) = 2;
    else
        ExpState(k+1,i) = 1;
    end
end

while 1
    % 时间更新一步
    k = k + 1;

    % 探测器状态更新 （探测器状态矩阵第 k 行为上一时刻状态，第 k+1 行为需要更新的状态）
    for i = 1:ExpN
        % 关机状态下一直关机
        if ExpState(k,i)==0
            ExpState(k+1,i) = 0;
        end
```

```
        % 处于探测状态下下一时刻如何改变
        if ExpState(k,i)==1
            if ExpState(k-1,i) == 1 % 表示该点已经探测完，需要向下一个点移动
                SearchPoint(ExpPosY(i),ExpPosX(i)) = 0; % 探测区域更新
                % 判断运动方向
                if ExpPosY(i)==1 || ExpPosY(i)==6 % 处在上边界时
                    if SearchPoint(ExpPosY(i)+1,ExpPosX(i))==1 %下方的点还未被探测
                        ExpState(k+1,i) = 4;
                    elseif ExpPosX(i) == 320
                        ExpState(k+1,i) = 0;      % 该探测器到达右边界，故关机
                    elseif SearchPoint(ExpPosY(i),ExpPosX(i)+1)==1
                        ExpState(k+1,i) = 3;      % 下方的点已经被探测,右边的点未被探测，需要横向移动
                    else
                        ExpState(k+1,i) = 0;      % 该探测器再找不到未探测点，故关机
                    end
                elseif ExpPosY(i) == 5 || ExpPosY(i) == 10 % 处在下边界的时
                    if SearchPoint(ExpPosY(i)-1,ExpPosX(i))==1 %上方的点还未被探测
                        ExpState(k+1,i) = 4;
                    elseif ExpPosX(i) == 320
                        ExpState(k+1,i) = 0;      % 该探测器到达右边界，故关机
                    elseif SearchPoint(ExpPosY(i),ExpPosX(i)+1)==1
                        ExpState(k+1,i) = 3;      % 上方的点已经被探测,右边的点未被探测，需要横向移动
                    else
                        ExpState(k+1,i) = 0;      % 该探测器再找不到未探测点，故关机
                    end
                else % 处在竖直方向的中间点时
```

```
                ExpState(k+1,i) = 4;
            end
        else
            % 表示该点刚刚探测一半，未探测完，故需要继续探测
            ExpState(k+1,i) = 1;
        end
    end

    % 处于等待状态下下一时刻如何改变-----需要单独判断

    % 处于运动状态 3 下下一时刻如何改变
    if ExpState(k,i) == 3
        if k<=5 % 横向移动需要 24s，必须需要出现 4 次运动状态为 3
            ExpState(k+1,i) = 3;
        else
            S3 = find(ExpState(k-3:k,i)==3);
            if length(S3)<4
                ExpState(k+1,i) = 3;
            else
                ExpState(k+1,i) = -1;% 到达探测点处于待定状态，得判断
是开机还是等待。
            end
        end
    end

    % 处于运动状态 4 下下一时刻如何改变
    if ExpState(k,i) == 4
        if k<=3 % 纵向移动需要 12s，必须需要出现 2 次运动状态为 4
            ExpState(k+1,i) = 4;
        else
            S4 = find(ExpState(k-1:k,i)==4);
            if length(S4)<2
                ExpState(k+1,i) = 4;
            else
                ExpState(k+1,i) = -1;% 到达探测点处于待定状态，得判断
```

是开机还是等待。

```
                end
            end
        end
    end

    % 探测器状态更新----处于等待状态的探测器下一时刻如何改变
    for i = 1:ExpN
        if ExpState(k,i) == 2
            S1 = find(ExpState(k+1,:)==1);
            flag = 0;     % 用来判断是否存在干扰需要等待       flag=0 表示不
需要等待
            for j = 1:length(S1)
                if     (ExpState(k,S1(j))==1     &&     Distance(i,S1(j))<4.8)     ||
(ExpState(k,S1(j))~=1 && Distance(i,S1(j))<9)
                    flag = 1;
                end
            end

            if flag == 1 % 需要等待
                ExpState(k+1,i) = 2;
            else
                ExpState(k+1,i) = 1;% 立马开机
            end
        end
    end

    % 探测器状态更新----判断刚刚结束运动的探测器是等待还是开机
    for i = 1:ExpN
        if ExpState(k+1,i) == -1
            S1 = find(ExpState(k+1,:)==1);
            flag = 0;     % 用来判断是否存在干扰需要等待       flag=0 表示不
需要等待
            for j = 1:length(S1)
                if     (ExpState(k,S1(j))==1     &&     Distance(i,S1(j))<4.8)     ||
```

```
(ExpState(k,S1(j))~=1 && Distance(i,S1(j))<9)
                        flag = 1;
                    end
                end

                if flag == 1 %  需要等待
                    ExpState(k+1,i) = 2;
                else
                    ExpState(k+1,i) = 1;%  立马开机
                end
            end
        end

        %  探测器位置更新（ExpPosX 和 ExpPosY 的更新）
        for i = 1:ExpN
            if ExpState(k,i) == 0 || ExpState(k,i) == 1 || ExpState(k,i) == 2
                ExpPosX(i) = ExpPosX(i);
                ExpPosY(i) = ExpPosY(i);
            end

            if ExpState(k,i) == 3
                ExpPosX(i) = ExpPosX(i) + 0.25;
            end

            if ExpState(k,i) == 4            %  判断向上还是向下运动
                if ExpState(k-1,i) == 4
                    if SearchPoint(floor(ExpPosY(i)),ExpPosX(i)) == 0
                        ExpPosY(i) = ExpPosY(i) + 0.5;
                    else
                        ExpPosY(i) = ExpPosY(i) - 0.5;
                    end
                else
                    if ExpPosY(i)==1 || ExpPosY(i) == 6
                        ExpPosY(i) = ExpPosY(i) + 0.5;
                    elseif ExpPosY(i)==5 || ExpPosY(i) == 10
```

```
                    ExpPosY(i) = ExpPosY(i) - 0.5;
                elseif SearchPoint(ExpPosY(i)-1,ExpPosX(i))==0
                    ExpPosY(i) = ExpPosY(i) + 0.5;
                elseif SearchPoint(ExpPosY(i)-1,ExpPosX(i))==1
                    ExpPosY(i) = ExpPosY(i) - 0.5;
                end
            end
        end
    end

    % 探测区域矩阵更新
    for i = 1:ExpN
        if ExpState(k,i) == 1 && ExpState(k-1,i) ==1
            SearchPoint(ExpPosY(i),ExpPosX(i)) = 0;
        end
    end

    % 探测器之间的距离矩阵更新
    Distance = CalculateDis(ExpPosX,ExpPosY); % 单位：km

    % 探测区域搜索完毕，终止循环判断
    if isempty(find(ExpState(k+1,:)>0, 1))
        if isempty(find(SearchPoint>0,1))
            break;
        end
    end
end
% 记录探测器关机时间
for i = 1:ExpN
    S0 = find(ExpState(:,i) == 0,1,'first');
    ExpCloseK(1,i) = S0 - 1;
    ExpCloseTime(1,i) = (S0-1)*6;       % 单位：s
end
time = k*6;
```

附录 B：SPOS 算法优化程序

```
% 粒子群算法求解离散变量的优化问题
% 针对之前的程序，更改了优化的目标----同时考虑总探测时间和关机时间的方差
%%
%初始化
tic;
clear all;
clc;
close all;

ExpN = 12;          %探测器数目

N = 100;              %群体粒子个数
D = ExpN - 2;     %粒子维数
T = 200;              %最大迭代次数
c1 = 1.5;           %学习因子 1
c2 = 1.5;           %学习因子 2
wmax = 0.8;         %惯性权重
wmin = 0.4;
disp(['探测器数目：',num2str(ExpN),';     ','最大迭代次数: ',num2str(T),';     ','种群粒子数目：',num2str(N),';']);
disp(['目标 1：总探测时间最小   目标 2：探测器关机时间的方差最小']);
%%
% 定义自变量的变化范围，粒子群位置最大值最小值
% x = [x2 x3 x4 x5 x6 x8 x9 x10 x11 x12];
% x_full = [1 x2 x3 x4 x5 x6 1 x8 x9 x10 x11 x12];
x_half_max = zeros(N,D/2);
x_half_min = zeros(N,D/2);
pa = round(320/(ExpN/2));     % 计算平均每个探测器探测的列数
for i = 1:D/2
    x_half_max(:,i) = (1 + pa*i + 4)*ones(N,1);
    x_half_min(:,i) = (1 + pa*i - 4)*ones(N,1);
end
xmax = [x_half_max x_half_max];
```

```
xmin = [x_half_min x_half_min];

% 定义粒子飞行速度
vmax = 4.*ones(N,D);
vmin = - vmax;

%%
%初始化种群个体
x = zeros(N,D);
for i = 1:D
    x(:,i) = randi([xmin(1,i) xmax(1,i)],[N 1]);
end

% x_part1 = x(:,1:D/2);          %判断是否产生重复个体，若是，重新初始化，同时进行从小到大排序
% for i = 1:N
%     x_temp = unique(x_part1(i,:),'sorted');
%     count = D/2 - length(x_temp);
%     while count>0
%         for j = 1:D/2
%             x_temp(j) = randi([xmin(i,j) xmax(i,j)]);
%         end
%         x_temp = unique(x_part1(i,:),'sorted');
%         count = D/2 - length(x_temp);
%     end
%     x_part1(i,:) = unique(x_temp,'sorted');
% end
%
% x_part2 = x(:,(D/2+1):D);    %判断是否产生重复个体，若是，重新初始化，同时进行从小到大排序
% for i = 1:N
%     x_temp = unique(x_part2(i,:),'sorted');
%     count = D/2 - length(x_temp);
%     for j = 1:count
%         p1 = randi([2 320]);
```

```
%             while ~isempty(find(x_temp==p1, 1))
%                 p1 = randi([2 320]);
%             end
%             x_temp(length(x_temp)+1) = p1;
%         end
%         x_part2(i,:) = unique(x_temp,'sorted');
% end
% x = [x_part1 x_part2];

% 初始化种群速度
v = rand(N,D).*(vmax - vmin) + vmin;
%%
%初始化个体位置和最优值
p = x;
pbest = ones(N,1);
ppbest = ones(N,1);
x_full = [1*ones(N,1) x(:,1:D/2) 1*ones(N,1) x(:,(D/2+1):D)];     % 生产完整的次序
for i = 1:N
    [time,ExpCloseTime] = CalculateTime(x_full(i,:));
    pbest(i) = time;                        % 设第一个目标函数为总的探测时间
    ppbest(i) = var(ExpCloseTime);   % 设第二个目标函数为关机时间的方差
end
% pbest

%初始化全局最优位置和最优值
g = ones(1,D);
gbest = inf;       % 总探测时间最小
ggbest = inf;     % 关机时间方差最小
for i = 1:N
    if pbest(i)<gbest    % 按照总探测时间最小更新
        g = p(i,:);
        gbest = pbest(i);
        ggbest = ppbest(i);
    elseif pbest(i) == gbest       % 总探测时间相同的情况下，按照方差最小更新
```

```
            if ppbest(i)<ggbest
                g = p(i,:);
                gbest = pbest(i);
                ggbest = ppbest(i);
            end
        end
    end
    gb = ones(1,T);         %用来记录迭代过程中总探测时间的变化过程
    ggb = ones(1,T);        %用来记录迭代过程中关机时间的方差的变化过程
    %%
    disp('--------------------开始迭代------------------')
    %按照公式依次迭代直到满足精度或迭代次数
    for  i = 1:T
        %  生产完整的次序
        x_full = [1*ones(N,1) x(:,1:D/2) 1*ones(N,1) x(:,(D/2+1):D)];

        for j = 1:N
            %更新个体最优位置和最优值
            [time,ExpCloseTime] = CalculateTime(x_full(j,:));
            FitValue1 = time;
            FitValue2 = var(ExpCloseTime);
            if FitValue1<pbest(j)
                p(j,:) = x(j,:);
                pbest(j) = FitValue1;
                ppbest(j) = FitValue2;
            elseif FitValue1 == pbest(j)
                if FitValue2 < ppbest(j)
                    p(j,:) = x(j,:);
                    pbest(j) = FitValue1; %  这句话可有可无
                    ppbest(j) = FitValue2;
                end
            end
            %更新全局最优位置和速度
            if pbest(j)<gbest
                g = p(j,:);
```

```
            gbest = pbest(j);
            ggbest = ppbest(j);
        elseif pbest(j) == gbest
            if ppbest(j)<ggbest
                g = p(j,:);
                gbest = pbest(j);
                ggbest = ppbest(j);
            end
        end

        %更新位置和速度值
        w = wmax - (wmax - wmin)*i/T;              %计算动态权重
        v(j,:) = w*v(j,:) + c1*rand*(p(j,:) - x(j,:))...
            +c2*rand*(g - x(j,:));
        x(j,:) = x(j,:) + round(v(j,:));

        %边界条件处理
        for ii = 1:D
            if (v(j,ii)>vmax(j,ii)) || (v(j,ii)<vmin(j,ii))
                v(j,ii) = rand*(vmax(j,ii) - vmin(j,ii)) + vmin(j,ii);
            end
            if (x(j,ii)>xmax(j,ii)) || (x(j,ii)<xmin(j,ii))
                x(j,ii) = randi([xmin(j,ii) xmax(j,ii)]);
            end
%                 % 判断是否有重复的个体，去重复数字处理
%                                     while   (length(find(x(j,1:(ii-1))==x(j,ii)))>0)   ||
(length(find(x(j,(ii+1):D))==x(j,ii))>0)
%                         x(j,ii) = randi([xmin(j,ii) xmax(j,ii)]);
%                 end
        end

        %对更新后的位置进行排序
        % 分为前一半探测器和后一半探测器的探测位置
        for k = 1:2
            k1 = D/2*(k - 1) + 1;
```

```
                k2 = D/2*k;
                xpart = x(j,k1:k2);
                xtemp = unique(xpart,'sorted');
                x(j,k1:k2) = xtemp;
            end
        end
        %记录历代全局最优值
        gb(i) = gbest;
        ggb(i) = ggbest;
        disp(['i=',num2str(i),';                    ','BestTime=',num2str(gbest),'s;
','BestVar=',num2str(ggbest),';']);
end
g;              %最优个体
gb(end);        %最优值
toc;
%  显示结果
g_full = [1 g(1:D/2) 1 g(D/2+1:D)];
[time,ExpCloseTime] = CalculateTime(g_full);
disp('-------------------------------------------------------------------');
disp('最优探测初始位置');
disp(g_full);
disp('-------------------------------------------------------------------');
disp('探测所需要的最短时间/s');
disp(time);

plot_gb = gb./gb(1);
plot_ggb = ggb./max(ggb);
figure(1);
plot(plot_gb,'r-','LineWidth',2);
hold on;
plot(plot_ggb,'b-','LineWidth',2);
xlabel('迭代次数 N');
ylabel('适应度值');
legend('归一化总探测时间','归一化关机时间方差');
title(['迭代进化曲线(','ExpN=',num2str(ExpN),')']);
```