

浙江大学第十七届 大学生数学建模竞赛

2019 年 5 月 7 日—5 月 17 日

编号

3923

题目

A ☒

B ☐

(在所选题目上打勾)

	参赛队员 1	参赛队员 2	参赛队员 3
姓名	武子越	马万腾	李梓彰
学号	3170104155	3170104754	3170104159
院(系)	竺可桢学院	竺可桢学院	竺可桢学院
专业	统计学	统计学	自动化
手机	17342023202	18888920643	18998996062
Email	3170104155@zju.edu.cn	3170104754@zju.edu.cn	3170104159@zju.edu.cn

浙江大学本科生院
浙江大学数学建模实践基地

ProblemA

摘 要

目标追踪问题在生活中相当常见，在许多领域中都有实际应用的价值，如军事领域、航空航天领域、生产流水线设计领域等。本文就目标追踪问题进行了详细的讨论，具体内容包括问题分析、模型建立与编程求解，其目的在于在探测器中心可以随时间变化的前提下，确定每部追踪器任意时刻探测中心的位置以及部分目标首次开始追踪的时刻，使得被成功追踪的目标数量尽可能多。对于直线上一个追踪器的问题，本文利用一种包含贪婪思想的算法进行求解；同时，为了加快求解速度，本文定义了绝对劣策略的判别标准并依照一定的原则来剔除这种绝对劣策略来进行搜索空间的缩减，最终在剔除绝对劣策略后剩余的目标上进行快速求解。对于直线上多个追踪器的问题，本文对目标进行扩充，确定了一种个体编码策略，利用遗传算法进行求解，同时采用一些操作减少程序运行的空间复杂度，最终确定根据求解情况确定探测中心的移动策略。对于三维空间中的情况，我们通过利用空间上的几何关系适当选取单位时间的方式将运动离散化，转变成和一维相似的情况，利用相同方式进行处理。最终文章对两种模型的结果从解的优劣以及效率两个角度进行分析，同时思考了该算法及模型的推广方式以及更多改进的可能。

关键词：贪婪算法、遗传算法、离散化思想

1 问题的提出

1.1 问题背景

目标追踪问题在众多领域中都有重要的应用，比如军事设备追击、人造卫星跟踪等。一些特殊过程的信号追踪需要一段时间的初始化，例如卫星 GPS 信号的捕获就需要先进行初始化，以达到内部载波和接收器的吻合。将初始化过程视为一次开始时间不固定，但所需时长确定的追踪，将后续的信号捕获视为开始和结束时间固定的追踪，便得到了一个“固定时长的首次追踪”加上“固定时间的二次追踪”的过程，这便引出了我们要研究的问题。

1.2 问题重述

考虑 N 个目标 $T_j, j = 1, 2, \dots, N$ ，给定每个目标的运动，以及首次追踪所需时间 p_j ，第二次追踪的开始时间 a_j 和结束时间 b_j 。首次追踪的开始时间 s_j 可以自主设定。第一次追踪和第二次追踪全部完成则代表该目标被成功追踪。不同时刻用于追踪某个目标的追踪器可以不同，但同一追踪器在某一时刻只能追踪一个目标。给出 M 个追踪器和追踪器的探测区间，每个追踪器的探测中心可以随时变化。现在要求给出追踪器的追踪策略，包括探测中心每秒所在位置，以及部分目标首次追踪开始时刻 s_j ，使得被成功追踪的目标数量尽可能多。

考虑两种情况：

1. 运动在一维空间内，且所有运动均为匀速。每部追踪器的探测中心也在该直线上，探测范围为一以探测中心为中点、长度固定的闭区间。
2. 运动在三维空间内，且所有运动为三维空间内的抛体运动，不同目标对应的抛掷时刻、高度、速率和方向未必相同。追踪器的探测范围为一以探测中心为球心、半径固定的球面及其内部。

假设 $M = 1$ 或 M 为某个大于 1 的固定常数（通常不超过 5），分别建立数学模型、设计算法，求解此问题。并对附件给出的情形（1）的数据，分别计算 $M = 1$ 和 $M = 5$ 时的相应结果并进行填写。

2 模型假设

假设目标随时间的运动是离散的。即对于每个目标 T_j 的每个单独的的时刻的位置，可由开始运动时刻 ST_j 和目标的速度 v_j 决定，关于“时刻”这个概念的详细假设如下：

由于在此题目设定的目标追踪情景下，所有约束条件的时间都是以整数变化的整数变量，因此我们把每一个整数的时间值定义为“时刻”。这种情况下，时间是离散的，或者说以单位长度时间为连续的，而此时每个目标的位置其实也是离散的，但同时也是以每一个单位时间的时刻而连续的。举例来说，对于第 j 个目标，从 a_j 到 b_j 内以 v_j 运动的意思即为在 a_j 时刻的位置与在 b_j 的位置之间相差 $v_j \times (b_j - a_j)$ ，更重要的是，在 a_j 到 a_{j+1} 前一瞬间这一段时间之内的位置，都与 a_j 这个时刻的位置相同。因此很容易发现，如果一个目标“结束运动于时刻 T ”，而另一个目标“开始运动于时刻 T ”，其实这两个目标的运动时间是不重叠的，前者是运动到 T 这个时刻的前一个瞬间就停止。因此对于每一个目标的 s_j 和 p_j ，其实就指开始运动于 s_j 时刻，而停止运动于 $(s_j + p_j)$ 时刻。而当我们讨论追踪器探测中心在某个时刻 T 的位置时，言下之意也是指这个时刻 T 到下一个时刻前一瞬间的这段时间内探测中心的位置。

此外，由于题目已经说明首次追踪时间为从 s_j 时刻到 $s_j + p_j$ 时刻为首次追踪， a_j 时刻到 b_j 时刻为二次追踪，故我们假设 $s_j + p_j$ 一定小于等于 a_j ，并且显然 s_j 大于等于开始运动时刻 ST_j ，并在此基础上进行后续的问题讨论。

也就是说，我们在最终上交的 Excel 表里，假如要标示时刻 18 到时刻 24 运行时长为 6，我们会标注编号为 18、19、20、21、22、23 这六个方格，这一点需要特别注意。

当物体以秒速 v_j 做匀速直线运动时，我们假设其每秒运动距离 v_j 小于等于追踪器探测区间长度 L ，这样才能以秒为单位具体的确定追踪器探测中心的位置。此时只需追踪器的每秒待在固定的位置，就可以保证目标在这一秒内跑不出它的探测范围。而事实上，“Problem A.xls”中物体匀速直线运动时的数据也符合我们的假设。

而当物体做抛体运动时，由于其速率会随着时间的而不断增加，故在物体运动后期，其每秒的位移一定大于追踪器的探测范围，所以此时以秒为单位研究探测中心的运动是不合理的，此处我们采用另一种方法，将在建模部分说明。

3 符号说明

ST_j	第 j 个目标开始运动的时刻
ET_j	第 j 个目标结束运动的时刻
v_j	第 j 个目标的运动速率
t_{1i}	扩充后第 i 个目标的首次追踪开始时刻
t_{2i}	扩充后第 i 个目标的首次追踪结束时刻
a_i	扩充后第 i 个目标的二次追踪开始时刻
b_i	扩充后第 i 个目标的二次追踪结束时刻
s_j	对目标 T_j 的首次追踪开始时间
L	一维情况下探测器的探测区间长度
v_{xj}	抛出情景下 x 方向速度
v_{yj}	抛出情景下 y 方向速度
v_{zj}	抛出情景下 z 方向速度
g	重力加速度

表 1: assumptions and notations

4 模型的建立与求解

由于题目中说明“追踪器的探测中心可根据需要随时变化”，故在某种程度上我们可以只以时间来考虑解的存在性与可行性。由于探测中心可以随时变化，所以当追踪器选择在一个时间段追踪某个目标时，只要通过不断改变探测中心就可以保证一定能追踪到该目标。这同样也说明，对追踪器的追踪情况我们可以只通过在每个时刻点上追踪器负责探测的目标来进行研究，而不需要同时考虑其探测中心具体运动情况。这也说明在时间轴上不重合的两个目标之间是互不影响的。

由于可以通过选取追踪目标来刻画追踪情况，所以我们可以把问题分成两部分，即每个时刻追踪目标的选取，和在给定追踪目标下追踪器探测中心移动策略的生成。由于后一个问题相对简单，所以对于 M 的任意一种情况，前一个问题都是我们重点研究的对象。

此外，很容易发现，由于每个目标被追踪的时间必然在时刻 b_j 之前，故其实题目所给数据中的结束运动时刻 ET_j 是无用的数据，我们在后面的建模分析求解中也没有再次提及和使用。

4.1 问题一 ($M = 1$) 的模型建立与求解

4.1.1 NP 难的证明

假如对一个“给定 n 个 $\{P_i, [a_i, b_i]\}$ ，判断是否有可行解”的问题，有一个算法可以在时间 P 内解决，那么对于任意的整数划分问题 $\{c_1, c_2, \dots, c_n\}$, $M = \sum_i^n c_i$ 。在多项式时间内构造本题实例 I:

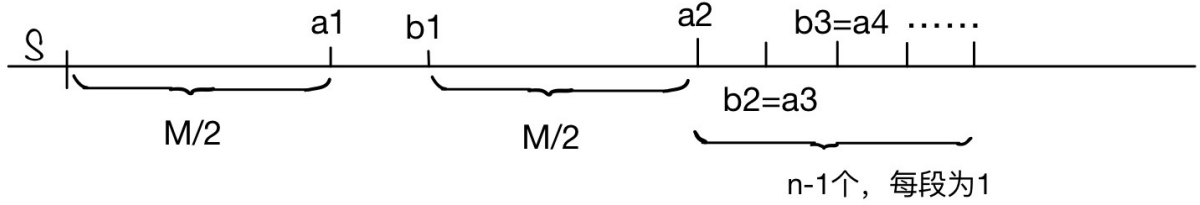


图 1: NP 难的说明示例

图 1 中 $P_i = c_i$, 且 S 为所有运动的开始时刻

$$a_1 - s = \frac{M}{2}$$

$$a_2 - b_1 = \frac{M}{2}$$

则整数划分有解当且仅当实例 I 有可行解。所以此算法可在 P 时间内判定整数划分问题，所以假设不成立，故原问题为 NP 难。

4.1.2 贪婪算法模型的求解

针对追踪器每秒追踪哪个目标的问题，我们通过贪婪算法构建了一个初步的模型，求出了一个这种策略下的最优解，这显然不是一个全局的最优解，因为本题是一个 NP 难的问题（前一部分已给出证明），不能通过这种模型求得全局最优解。但在尽量减少计算复杂度的情况下（后面的第二种解法会面临的问题）通过这种策略不失为一种好的选择，也可以帮助我们进一步加深对这一问题的理解。

以图 2 中的取了一些简单的目标点为例，我们阐述一下这一基于贪婪算法的模型的详细建立准则：以时间轴为准按照开始时间 ST_j 由小到大依顺序排列。之后遵循一定的顺序进行解的寻找。横坐标为时间，纵坐标为目标，黄色为第一段（可变），红色为第二段（固定），黑色方框为目标运动时间。数字为红色的表示追踪成功。寻找解的时候遵照以下原则：

1. 相同时间开始的优先选择 b_j 最小的（如 1、2 选择 2）
2. 在追踪过程中如果出现 b_j 更小的，跳到 b_j 更小的目标（如在 3 秒时从 2 跳到 4）
3. 结束时间 b_j 相同，总时间短的优先（如 13、14 的情况）
4. 不空闲原则（如 11、12 交错），为了防止探测器空闲，可以随时去处理一段黄色区域

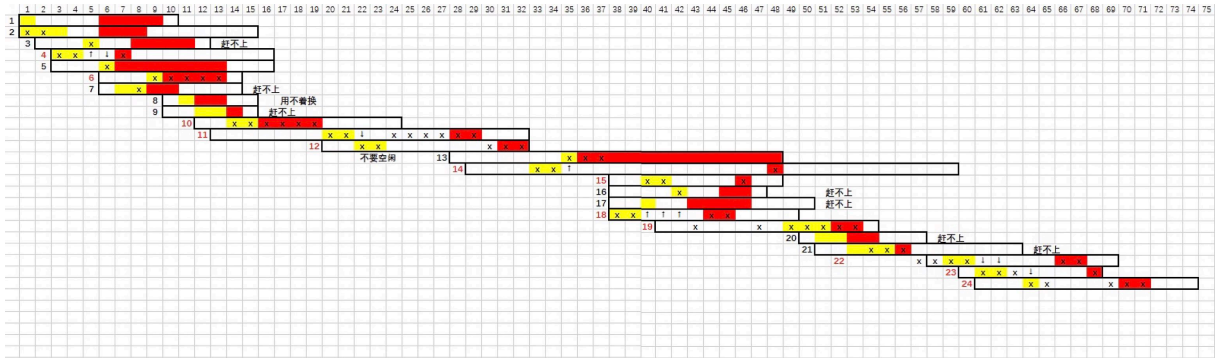


图 2: 贪婪模型简单示例

因此基于图 2 所示我们可以按照这个想法进行判断：

- 按照目标运动开始时间升序排列
- 开始时由于 2 号更早结束，选择 2 号
- 3s 时 4 号出现，发现 4 号更早结束，跳到 4 号
- 7s 时 4 号追踪结束，选择目前出现的目标中红色最早结束的，但是 7 号已经来不及追踪了（黄色区域无法满足），跳过，选择下一个红色最早结束的（6 号）
- 10s 时 8 号出现，结束时间相同，不用更换
- 13s 时 6 号追踪结束，9 号赶不上了，选择 10 号
- 19s 时 10 号追踪结束，追踪 11 号，处理完 11 号黄色区域，不要让探测器空闲，先去追踪 12 号黄色区域，追踪完继续追踪 11 号
- 33s 时对 13、14 号，选择总时间（红 + 黄）短的 14 号追踪
- 35s 时不空闲原则
- 38s 时出现了 4 个结束时间更早的目标，先跳到 18 并立刻追踪 18 号的黄色区域
- 40s 不空闲原则，此时处理 15、16、17 都可以，但同样遵循总时间最短的原则。
- 追踪完后 42s 跳到 19 号（不空闲原则），43s 跳回追踪 14 号。
- 46s 追踪完 18 号，16、17 赶不上，追踪 15 号
- 49s 重新追踪 19 号（之前空闲时候来不及追踪完，现在重来）
- 22、23、24 号遵照不空闲原则

与这种简单的情况相似，我们将题目中的 100 个目标也根据开始时间 ST_j 进行了排序并通过贪婪算法进行了解的寻找，如图 3 所示，100 个目标的排列和求解的过程我们放在了“贪婪模型数据探索.xlsx”文件中。该贪婪算法在求解此实例的时候求得局部最优解为 13。

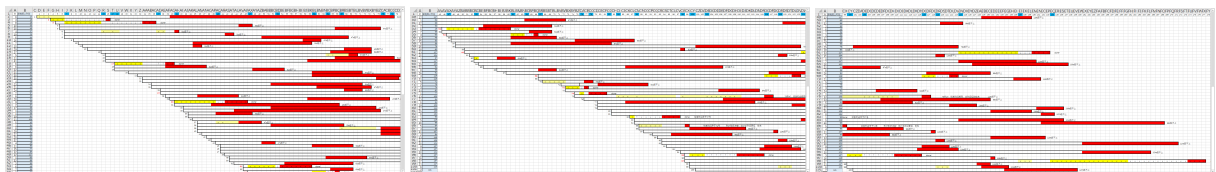


图 3: 各目标在时间轴上的排序

4.1.3 通过扩充目标进行求解

显然，通过贪婪算法求解有时间上比较快的优势，但求得的解并不是最优解。考虑到题目中主要的未知量来自于每个目标 j 的 s_j 都是不确定的，直接通过动态规划或者启发式算法都会面临难以确定策略的问题，因此我们在思考后决定，**对题目的 100 个目标进行扩充**。

由于每个目标的可探测时间可以由四个参数 a_j 、 b_j 、 s_j 和 p_j 确定的两个时间段 $[s_j, s_j + p_j]$ 和 $[a_j, b_j]$ 来描述，而由于 s_j 是一直在变化的因此很多做法难以开展。但是同时考虑到 s_j 的变化是离散角度变化的，它可能的取值只有从题目给定的 ST_j 到 $a_j - p_j$ 这两个时刻之间的有限个值可选，也就是 $a_j - p_j - ST_j + 1$ 个值可选，因此若我们将某一个特定的目标 j 扩充为这样的 $a_j - p_j - ST_j + 1$ 个目标，它们各自有确定的首次追踪开始时间、首次追踪结束时间、二次追踪开始时间和二次追踪结束时间。对于扩充后的每个目标我们将其下标记为 i ，因此有上述四个值对应的记号为 t_{1i} ， t_{2i} ， a_i ， b_i ，经过实际扩充后发现可将目标数量由 100 增大至 2255，故 $i \in [1, 2255]$ 。

这样，我们就将题目中的变化因素抹除了。同时由于从同一个目标扩充而来的所有目标的 a_i 与 b_i 是相同的，所以对这 2255 个目标寻求最优解的过程中一定不会将原本来自一个相同目标的两个扩充后目标同时考虑进来，保证了不会取到“伪最优解”，故我们只要把这 2255 个目标一视同仁进行后面的分析求解即可，不用做额外的操作。

4.1.4 问题复杂性分析

扩充后的问题可以和图的最大团问题相联系。扩充后两个目标如果没有重合的时段，则意味着这两个目标可以位于同一个可行解之中。如果有重合，则至多只能选择其中一个目标。每个扩充后的目标可以视作一个顶点，若两目标之间没有重合时段，则将其用一条边相连，因此， $M=1$ 时跟踪目标最大数量问题可以看做求图中的最大团问题。

由于扩充之后图的顶点数目为 2255，因此，求解该问题将会变得异常复杂，会耗费大量算力与时间，因此，我们希望能够找到一种方法，在目标扩充之前对于目标进行预处理，剔除一些绝对劣的目标来缩小搜索范围。

4.1.5 通过剔除绝对劣目标缩小搜索空间

绝对劣目标的定义为，对于任意包含该目标的可行解，若存在某一目标，使得剔除该目标后可行解不会变得更坏，则该目标为绝对劣目标，可以进行剔除使得搜索空间得以缩小。

对于 $M=1$ 的情况，对于绝对劣目标的判断原则如下：若某目标 j 的第二段追踪时段 $[a_j, b_j]$ 能够包含另外一目标两段追踪时间，则该目标为绝对劣目标，可以进行剔除。如图，对于 54 号目标而言，55、57 号目标的第二段追踪时间能够完全覆盖 54 号目标的两段追踪时间，则 55、57 号目标为绝对劣目标，可以进行剔除。容易证明，一定存在最优解可以不包含 55、57 号目标（否则将其替换为 54 目标仍然不会改变其最优性）。

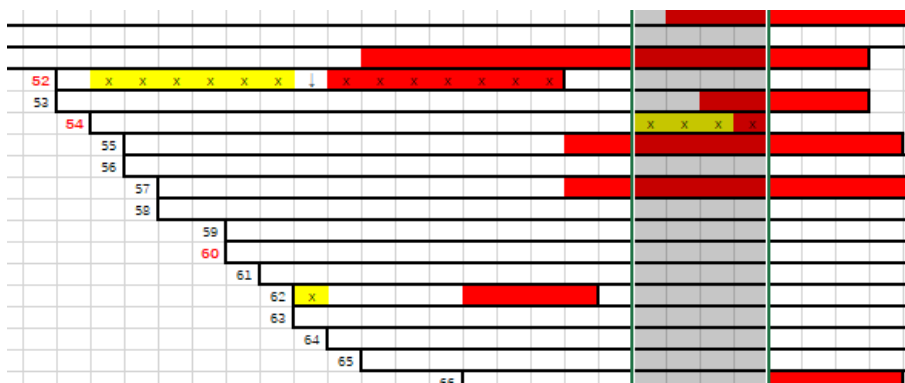


图 4: 绝对劣目标的判断 (根据第二段追踪时间)

另外一种绝对劣的情况如下：在某目标 j 的运动时间中，无论从何时开始追踪，都可以找到另一目标，使其两段追踪时间都能够被目标 j 的第一段追踪时间所覆盖，则目标 j 为绝对劣目标，可以进行剔除。如图，对于 12 号目标而言，其第一段追踪时间为 29 秒，可以从 12 秒到 39 秒间任意时刻开始追踪。若从 12-17 秒间任意时刻开始第一段追踪，则第一段追踪时间可以覆盖 17 号目标的两段追踪时间，若从 18-30 秒间任意时刻开始追踪，则第一段追踪时间可以覆盖 30 号目标的两段追踪时间，若从 31-39 秒间任意时刻开始追踪，则第一段追踪时间可以覆盖 44 号目标的两段追踪时间。

容易证明，一定存在最优解可以不包含 12 号目标（否则根据其第一段追踪开始时间将其替换为 17、30 或 44 号目标仍然不会改变其最优性）。

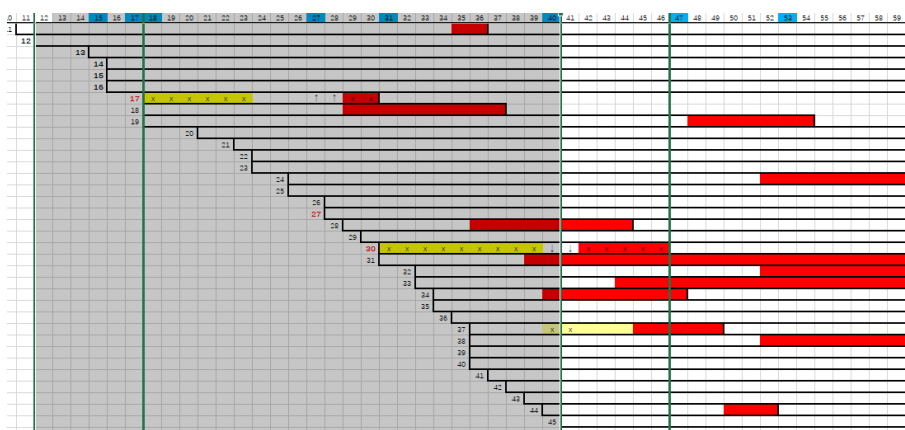


图 5: 绝对劣目标的判断 (第一段追踪时间覆盖 17 号目标)



图 6: 绝对劣目标的判断 (第一段追踪时间覆盖 30 号目标)



图 7: 绝对劣目标的判断 (第一段追踪时间覆盖 44 号目标)

通过以上两种剔除目标的策略，我们将搜索范围进行了缩小，最终剩余 43 个目标，通过以上的分析可以知道，在 $M=1$ 的情况下，最优解一定由这 43 个目标中的某些目标构成，因此，扩充的方法只需要在这 43 个目标上进行，对于此扩充目标进行求解，此步我们最终求出的局部最优解为 18，在此最优解中，探测器移动策略如下图所示（为了方便起见，根据开始时间升序的顺序为目标重新编号）：

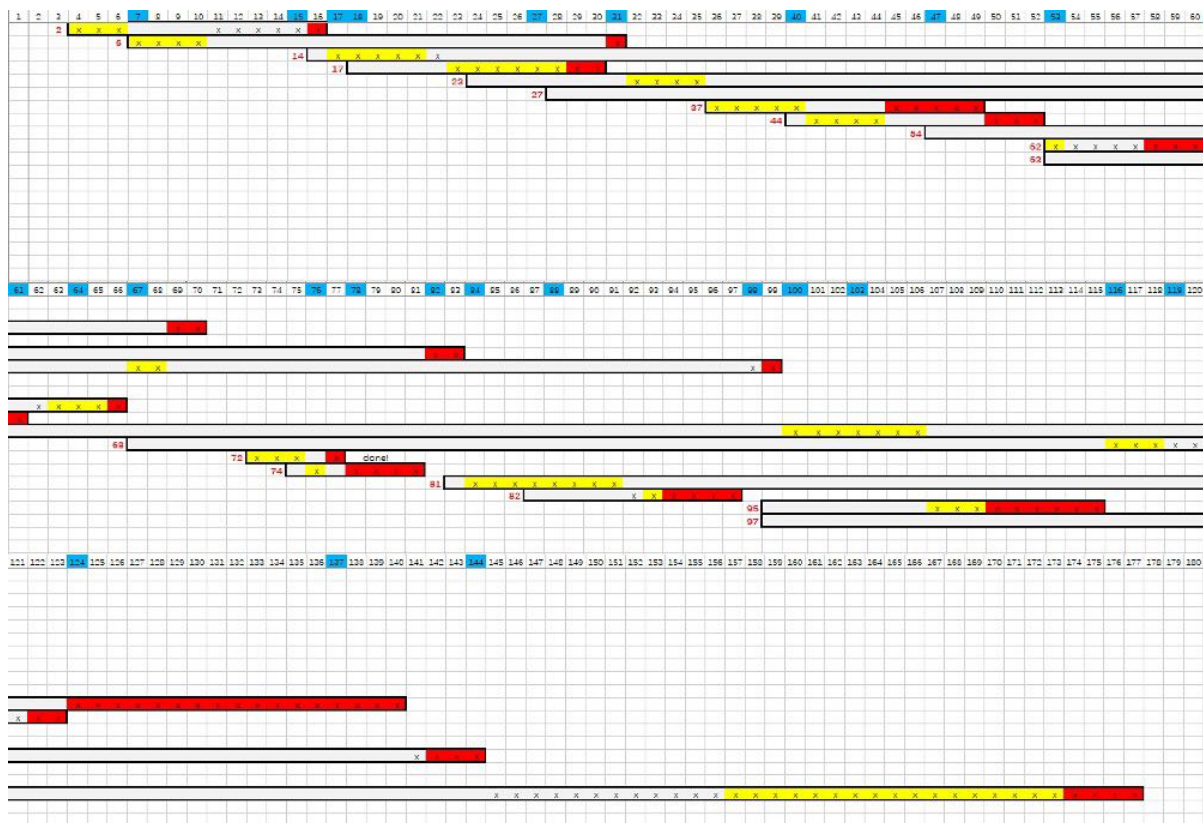


图 8: M=1 时解得 18 的探测器移动策略

从直观上来看，最优解的时间安排是非常巧妙的，可以说，探测数目为 18 的可行解几乎将探测器的所有时间占用，其中第一段观测时间由于较为灵活，因此往往被安排在一些时间恰好相同的空闲时段中。

最优解中探测器追踪目标的具体信息如下表：

表 2: 选出的 18 个目标的信息

目标	开始时间升序编号	首次追踪时长	首次追踪时段	二次追踪时段
64	2	3	4-6	16
85	6	4	7-10	31
70	14	5	17-21	69-70
49	17	6	23-28	29-30
43	23	4	32-35	82-83
98	27	2	67-68	99
86	37	5	36-40	45-49
57	44	4	41-44	50-52
72	54	3	63-65	66
50	62	1	53	58-61
10	63	7	100-106	124-140
94	68	3	116-118	122-123
28	72	3	73-75	77
88	74	1	76	78-81
13	81	8	84-91	142-144
66	82	1	93	94-97
8	95	3	107-109	110-115
54	97	17	157-173	174-177

4.2 问题一 ($M = 5$) 的模型建立与求解

针对 $M = 5$ 的情况, 仍然使用扩充后的 2255 个目标来作为模型求解时的数据。但是此时因为追踪器的数量过多, 人为的去寻找最优解不现实, 因此我们决定选择使用遗传算法来进行最优解的寻找。针对这一特定问题将其模型化为适合遗传算法的情景并进行迭代求解。

4.2.1 遗传算法简述

遗传算法是人工智能中用于解决最优化问题的一种准启发式搜索算法, 我们一般把问题的每一个解构造为一个个体, 将所有的个体组成的集合称为一个种群。之后对这个由问题解组成的种群施加一些生物学上的现象, 如遗传、突变、杂交、自然选择等, 并保证生成适应值更高的个体, 所谓适应值更高即是符合问题的更优的解。遗传算法的关键是如何将特定的问题按照一定的编码策略转成适合描述和进行进一步操作的“个体”, 以及如何设定每一次迭代中应该有的操作。

4.2.2 目标选取模型的建立

首先, 我们的所有 2255 个扩充后的目标数据都储存在了文件“expandA.mat”中。该文件载入 Matlab 后为一个 2255×4 的矩阵, 每一行对应一个目标 i , 其四个元素分别为该目标 i 的首次追踪开始时刻 t_{1i} 、首次追踪结束时刻 t_{2i} 、二次追踪开始时刻 a_i 和二次追踪结束时刻 b_i 。

基于此, 我们确定这样的个体编码策略: 将一个个体 (即一个追踪策略) 设置为一个 1×2255 的行向量 A , A 中的第 k 个元素 (即基因) 即对应“expandA.mat”中的 2255 个目标里的第 k 个目标, 在 A 中, 若该元素为 1 则代表追踪对应的目标, 0 则代表不追踪, 因此我们确定了种群中个体的表达方式和编码及解码策略。

对于整个种群, 为了便于在 Matlab 中计算求解, 我们用一整个大的矩阵 M 来表示整个种群: M 的每一行就是一个个体, 因此 M 是一个 $n \times 2255$ 的矩阵, 而这里为了初值预设方便, 我们将 n 直接设置为 2255, 并把整个种群的初值设置为 2255×2255 的单位阵, 即每个元素都是一种最简单的追踪策略 (只追踪一个目标) 且互相之间不重复。整个初值的数据放在了“first_group.mat”文件中。

之后在每次迭代里, 我们加入了有限次以下操作: 变异和交配。为了减少迭代中在无意义的进化方向上可能花费的时间, 我们保证变异仅仅是某一个个体的某一位基因可能变为 1, 而不考虑变为 0 的变异方向, 并且在变异后检测变异后的个体所对应的策略是否是可行的 (检测是否是可行的策略通过调用“individualevaluation.m”文件对应的 `individualevaluation()` 函数来实现), 因此很明显, 我们的变异一定是向解更优的方向进行变异的, **这一操作也保证了整个种群在向更优的方向进化**。而对于交配, 同样为随机选取两个个体进行交配, 同时**为了减少计算量和计算时间, 我们采取对两个个体取或的操作**, 这样一来, 交配产生的个体中含 1 的数量肯定多于其父与母。之后对于新产生的个体有两种后续操作: 如果新个体不是可行的策略, 那么保留其父母; 如果新个体可行, **那么替换掉其父母中含 1 数量较多的那一个**。

可以发现, 为了减少程序运行的空间复杂度, 我们的一切操作都保证整个种群的个体永远维持在 2255 个, 不增不减。此外为了不让种群往坏的方向迭代而定义了取或的交配法则来保证新个体一定是优于 (含 1 更多) 其父母的。关于**新个体替换掉父母中含 1 较多的那个**的原因现说明如下: 如果替换掉较少的那个, 经过实验发现, 种群的最优解中含 1 的个数可能限制在一个离真正的最优解还有一定距离的值而无法继续“向前迭代”。这其实是很好理解的, 假若已经有了一个可以追踪多个目标的方案, 在不改变追踪这些已有目标 (显然取或是不会改变父母个体中原有的 1 的存在性的) 的情况下, 只再增加 1 个追踪目标后新的追踪方案才是大概率可行的。而如果每次产生的新个体都替换掉那个含 1 少的父母中的个体, 则很快种群会被含 1 数量“比较多”的个体占满, 但他们互相再交配一次产生的个体很可能因为含 1 数目远高于最优解应该有的含 1 的数目而被放弃, 因此整个种群陷入一种“停滞不前”的死循环。

在实际运行后我们发现, 第一次运行完遗传算法的 30000 次迭代后, 得到的结果为 42 左右 (可以追踪的目标为 42 个), 这之后我们从迭代结束后的种群中**选出追踪目标最多的 66 个方案, 即个体**, 之后补充到原本的 2255×2255 的矩阵中, 使之成为一个 $t \times 2255$ 的矩阵 ($t \geq 2255$), 之后再对这个新矩阵做一次

遗传算法的迭代 30000 次操作，但是这 30000 次迭代与之前不同的地方在于：**每次交配都是前 2255 个只探测一个目标的个体中的随机一个来和这新增加的 66 个个体中的随机一个进行交配，并用可行的交配方案更新这 66 个个体中的交配者。**

这第二次迭代 30000 次的操作极为关键，因为这可以有效解决整个种群都具有一定数量的 count 值而导致交配无法使整个种群进化的情景，将最优的 66 个个体加入一个全新的种群并且每次交配后都用更好的个体来替换父母中更优的那个就是为了防止上述情况的产生。并且实践也证明，在前 30000 次迭代中似乎已经无法增加的最优值在后 30000 次迭代中可以得到有效的增加。

事实上，这种算法适用于所有 $M \geq 2$ 的情况。

4.2.3 探测中心移动策略生成

在追踪的物体确定后，追踪器每秒要追踪的物体也就确定了。对于策略生成部分，我们选择统计每秒需要追踪的物体，然后将每秒的任务分给 5 个追踪器，同时保证：

- 每个追踪器的探测区间在一秒内不需移动。

所以在此我们设计一个将追踪器分配给目标的方案，并确定每秒探测中心的位置，方案如下：

1. 按照遗传算法得到的目标结果，按照目标编号从小到大顺序统计出每秒所需探测的目标。
2. 对于每秒内需要被追踪的个体，按照追踪器编号从小到大的顺序将追踪器分配给目标。
3. 对于 t 秒内要追踪的目标，计算出其 t 秒开始时所在位置 $x(t)$ ，将追踪器的探测中心置于 $x(t) + \frac{L}{2}$ 处，即可保证其探测中心在一秒内不需移动就能探测到目标。

4.2.4 代码实现

整个迭代的过程在“GA.m”程序文件中实现，其中需要调用“individualevaluation.m”中的 individualevaluation 函数来判断单个个体是否为可行策略。程序的源代码会在附录中给出，我们在每一次迭代中都加入了记录并显示该次迭代后含 1 最多个体的 1 的数目和该个体在整个种群中的下标的操作，之后最后一次迭代结束后将含 1 最多的最优个体单独保存为一个“save.mat”文件，之后通过比对该最优个体和“expandA.mat”中的数据得到该最优策略分别探测的目标以及各目标的各项时间指标。同时，第一个 30000 次迭代后将整个种群数组也保存一个文件方便取出 count 最大的 25 个个体。而在第二个 30000 次迭代后也会保存 count 最大（追踪目标最大）的个体（策略），并保存整个种群。两次运行“GA.m”后保存和使用的的数据文件如下：

1. save.mat 为保存第一个迭代 30000 次后的追踪目标最多的那个方案，为一个 1×2255 的向量
2. groupdata.mat 为保存第一个迭代 30000 次后整个种群的值的矩阵，尺寸为 2255×2255
3. second_group.mat 是从 groupdata.mat 中选出 66 个 $count \geq 39$ 的个体后补到最一开始的初值种群矩阵下面后生成的 2321×2255 矩阵
4. secondsave.mat 为保存第二个迭代 30000 次后的追踪目标最多的那个方案，为一个 1×2255 的向量
5. secondgroupdata.m 为保存第二个迭代 30000 次后整个种群的值的矩阵，尺寸为 2321×2255

由于遗传算法针对的是 $M \geq 2$ 的情况，故 individualevaluation 函数中有一个需要注意的细节：不光判断该策略是否可行，还要避免发生因为一个目标扩充出来的多个目标都被追踪的策略被判断为可行的情况（这种情况之所以会发生就是因为只判断时间轴上某个时刻重合数大于追踪器数目会导致一个目标扩充出来的多个“分身”都被纳入其中）。因此我们在 individualevaluation 函数一开始就判断该策略有没有包含一个目标扩充出的“分身”，如果有的话，直接记这种情况为该策略不可行。这样当然会有点增加达到最优解的时间，但是代码实现更加简单且计算更快。

4.3 问题二的模型建立与求解

4.3.1 问题分析

按照题目给定的情景,所有目标为某个时间开始从某一高度抛出,则假设各目标的抛出点在地面上投影是重合的,只是高度不同(其实从后面的分析不难看出地面上投影重合也不是很重要的条件),对于每个目标点抛出时不同速度和角度,可以将其归结为三个未知量来表示,即沿某固定世界坐标系的 x 轴、 y 轴和 z 轴方向的 v_{xj} , v_{yj} 和 v_{zj} ,各速度值可以为负数表示相反方向。之后,通过结合 v_{xj} , v_{yj} 和 v_{zj} 三个速度值以及抛出高度 h_j 和开始运动时刻 ST_j 就可以得出结束运动时刻 ET_j ,随后再自行定义首次追踪时长和二次追踪的开始时刻和结束时刻即可,与一维情况给的数据类型相同。

不难发现,由于问题的关键点在于题目设定的一个追踪器在某时刻只可以追踪一个目标,且探测中心可以随意运动,其实在这个条件下一切位置的概念都不需要纳入考虑,只需要在时间轴上划分即可,这样当抽象出问题的数学表述时,其实与一维情况无异。所以我们还是将问题分为追踪目标选取和给定目标下追踪器探测中心移动策略生成。

4.3.2 问题离散化

基于第一题中“以秒为单位”的启发,我们可以人为设置单位时间,使得问题转化为与问题一类似的情况。我们的离散化遵循的原则是:使得所有物体在任一个单位时间内的运动轨迹都可以被追踪器探测范围覆盖到。为了达到这个目的,我们对目标的轨迹曲线进行了物理和几何上的分析,即:任何一个目标的运动都可被分解为 x 轴和 y 轴上速度为 v_{xj} , v_{yj} 的匀速直线运动和 z 轴上初速度为 v_{zj} 的落体运动。以下推导在 t 秒时目标 T_j 在空间内的坐标,其中 $ST_j \leq t \leq ET_j$:

记 $q = t - ST_j$ 为物体运动的时间,则:

x 轴上的位移为 $v_{xj}q$

y 轴上的位移为 $v_{yj}q$

z 轴上的位移为 $v_{zj}q - \frac{1}{2}gq^2$

故在 t 秒时目标 T_j 在空间内的坐标为:

$$\left(v_{xj}(t - ST_j), v_{yj}(t - ST_j), h_j + v_{zj}(t - ST_j) - \frac{1}{2}g(t - ST_j)^2 \right)$$

记为 $\vec{X}_j(t) = (x_j(t), y_j(t), z_j(t))$ 。

由于目标运动轨迹可以视为一个平面上的抛物线,我们可以寻找一个 Δt ,使得:

$$\max_j \|\vec{X}_j(ET_j) - \vec{X}_j(ET_j - \Delta t)\| = L$$

由于在 ET_j 时,目标在 z 轴速度分量达到最大,此时以 $\frac{\vec{X}(ET_j) + \vec{X}(ET_j - \Delta t)}{2}$ 为追踪器探测中心恰好可以使得追踪器在最后 Δt 的时间内可以完全覆盖到目标,如图所示。

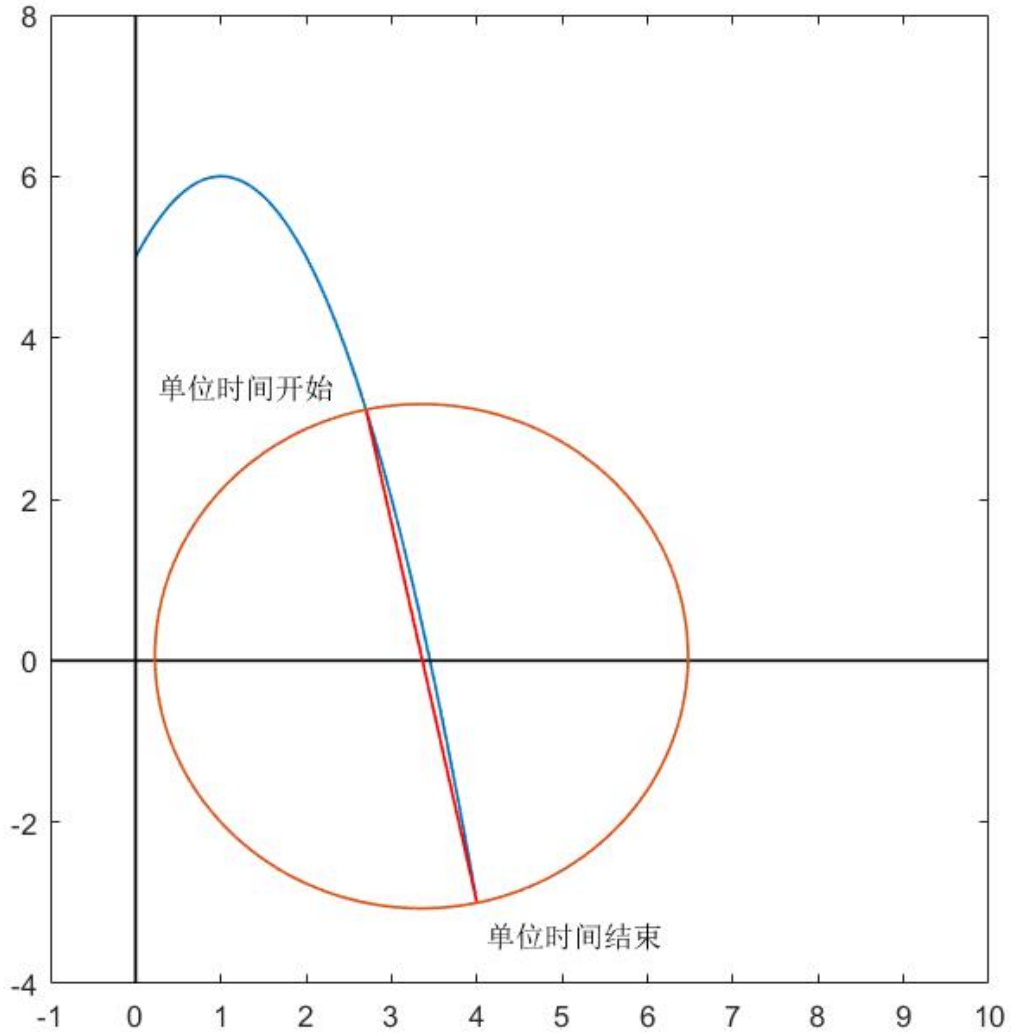


图 9: 抛体示意图

根据抛物线性质可知，对任意一个目标运动轨迹之中的任意一段 Δt ，以 $\frac{\vec{X}_j(t) + \vec{X}_j(t + \Delta t)}{2}$ 为探测中心的追踪器可以完全探测这段时间内目标的运动。

此处我们以 Δt 为单位时间分割时间轴，将运动进行离散化，对于物体运动的第二次追踪时间 a_j, b_j ，只需将时间轴分割后，对 a_j 取小于等于 a_j 的单位时刻，对 b_j 取大于等于 b_j 的单位时刻。此时问题就转化为和一维状态离散时间的情况相似了。

4.3.3 缩小搜索空间的贪婪算法模型 ($M = 1$)

与一维情况类似，在选取目标时，我们可以通过剔除绝对劣的目标来删除部分绝对不会被选中的目标，以此提升贪婪算法的效率。此时

而在算法选择目标阶段，我们仍然按照问题一中的描述来进行贪婪的选择。这样可以得到一组在三维情况下的坐标选择。

4.3.4 遗传算法模型 ($M \geq 2$)

当 $M \geq 2$ 时, 与问题一相似的, 我们使用遗传算法进行坐标的选取。但是此处由于 Δt 较小, 对于大部分实例而言分割区间较多, 采用遗传算法会导致计算量较大, 迭代运行时间变长。

4.3.5 探测中心移动策略生成

根据我们的离散化策略, 对于任意一个追踪器追踪 t 时刻的目标 T_j , 可以将其设置在 $\frac{\bar{X}_j(t) + \bar{X}_j(t + \Delta t)}{2}$, 即保证在后一个单位时间内其运动一直在我们的追踪器探测范围内。此时的策略生成方式就和问题一中相似了, 此处不再重复。

5 结果分析

5.1 问题一模型结果分析

5.1.1 贪婪算法和遗传算法模型结果分析 ($M = 1$)

在问题一中, 通过贪婪算法我们对于该实例求得一个局部最优解为 13, 然而该算法还有较大的改进空间, 不空闲原则虽然能够使得探测器随时在工作, 但是很多时候由于问题本身的原因, 在空闲时间处理追踪其他目标的时间不够导致追踪无法完成, 使得这一部分空闲时间并没有发挥应有的价值, 在题目实例之中, 尤其是 100 秒之后的时间中有很多尚未利用的分散的空闲时间。

通过剔除绝对劣目标来减小搜索空间后, 利用遗传算法收敛得到一个最优解 18, 相比于贪婪算法的 13 又有了一些改进, 在 18 的结果中可以看到, 对于目标的选取, 算法综合考量了第二段追踪时长、两次追踪总时长、物体运动总时长等因素。总体来说, 运动时长较长, 同时两次追踪时长较短的目标更容易被选中成为追踪器追踪的备选。

5.1.2 遗传算法模型结果分析 ($M = 5$)

经过多次数据测试和迭代, 我们求得的最优结果为 47, 储存在 “secondsave.mat” 中并将数据记录在要求上交的 Excel 表格中。

经过分析发现, 其实 47 应该不是最优解, 这是因为遗传算法所需要的是漫长时间的迭代, 和完美符合特定问题的编码及迭代中生物操作的呈现, 这一方面我们还有待提高, 可以使求出的解更接近于最终最优的答案。但是本次由于时间受限, 只求到 47 这个最优解为止。

5.2 问题二模型结果分析

问题二的分析同问题一的分析方式相似。此处我们没有生成问题实例, 故没有得出具体的结果。但是从定性的角度分析, 对于数量一样多的目标, 问题二的求解计算量要显著大于问题一的求解计算量。

6 模型的评价与推广

6.1 模型的评价

6.1.1 贪婪算法模型

贪婪算法模型的核心思想是通过一个固定的规则在线性时间内排出可行解。从解的最优性角度考虑, 这种贪婪算法每一步都在进行 “非劣” 选择, 即在现有选择基础上做短时间内的非劣决策。这种选择策略可以保证每一步的解在较短时间内最优, 但通常只能找到局部最优解而无法保证整体最优。比如该题目 $M=1$ 时使用的算法就使用了结束时间早优先、总时间短优先且探测器空闲时间尽量少的贪婪思想。虽然

能够在每一步做出当前最利于提升效果的决策，但是求出的解显然没有优于遗传算法所得到的解，仍然有较大的改进空间。

贪婪算法的优势在于时间复杂度低，能在多项式时间内找到一个相当不错的可行解，但是缺点在于近似程度较差，对于个别实例，该贪婪算法最坏情况界为无穷大。

6.1.2 遗传算法模型

遗传算法的核心是模仿种群的交配和淘汰，即通过我们的筛选机制将优秀的个体（即可能交配生成最优解的决策方案）进行富集，从而将下次交配生成更优策略的可能性增加。

同时为了避免随机交配导致种群内所有个体都缺乏某个因子，从而种群不断交配只能陷入局部最优解，我们设置了基因的变异频率。每个个体身上的基因有一定几率进行变异，这种变异有可能给整体带来更优秀的基因，这就使得遗传算法的迭代过程有机会跳出局部最优解。同时由于淘汰机制的存在，不合适的个体将被删去，这保障了算法解向最优解靠拢。

遗传算法的另一个优势在于，它适用于 $M \geq 2$ 的所有情况。

6.2 模型的推广

6.2.1 贪婪算法模型

该贪婪算法的思想在很多现实的问题中可以有应用。比如说，教室的课程排班问题，如何使得 M 个教室排下最多的课程，可以优先排时间短的课程或者结束时间早的课程。当然，贪婪思想在某些情况下将会比求出最优解的思想更为实用，比如军事上的实时追踪问题中，敌机或敌方导弹的位置变化非常重要，而良好军事决策需要尽可能使得追踪数量最大化；然而在实战的过程中往往需要在短时间内进行决策，此时贪婪算法的速度优势可以发挥其价值，能够确保在每个时间点做出不会让形势变得更坏的决策，同时使得最终的结果从平均意义来说相对较好。

因此，贪婪算法往往可以运用在时间上要求较高的情况，如实时监测、灾情预警、及时反馈系统等。这一类实际问题往往不需要最优解，只需要一个相对较好的解，这一类贪婪的思想在这种情况下将会有更大的实用性。

6.2.2 遗传算法与数据扩充思想

对于扩充数据的思想，当具体问题中有很多这样的未知数需要确定其分布，但其分布却不是解决问题时所解决的主要问题，可以借鉴这种思想。即是在存储空间允许且本身其不确定是在有限空间内的有限种可能时，可以把每一种可能作为一个新的研究对象来扩充研究对象集合，同时使用一些后续措施来确保不会重复采集原本由一个研究对象扩充出的各个“分身”即可。这种方法可以有效地减少次要的未知因素对求解过程的影响。而至于遗传算法，作为进化算法中的一种代表算法，其应用的领域更多。只要可以将具体的问题编码化成为算法针对的种群就有很好的推广性。针对解的寻找与优化问题，如函数优化、组合优化、路径规划、车间调度、装箱问题等等情景都可以使用。并且只要时间复杂度允许，在针对具体问题确定好正确的每次迭代进化内的操作后，理论上遗传算法不会错过最优值，因此有很强的应用价值和推广性。

7 参考文献

- 1 武广号, 文毅, 乐美峰. 遗传算法及其应用 [J]. 应用力学学报, 23(6):9-10, 2000.
- 2 杨启文, 蒋静坪, 张国宏. 遗传算法优化速度的改进 [J]. 软件学报, 12(2):270-275, 2001.

- 3 Ruiz, Rubén, and Thomas Stütze. "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem." *European Journal of Operational Research* 177.3: 2033-2049, 2007.

8 论文附录

8.1 代码汇总

GA.m

```
1 function count = GA()
2 clear all;
3 load('first_group.mat'); %ok<*LOAD>
4 X=first_group;
5 scale=2255;
6 % load('second_group.mat');
7 % X = second_group;
8 % scale = 2269;
9 iter=0;
10 Lim1=30000;
11 N=300;
12 while(iter<Lim1)
13     iter=iter+1;
14     for k = 1:100%往好的方向变异
15         a = unidrnd(scale);
16         b = unidrnd(2255);
17         X(a,b) = 1;
18         if(individuevaluation(X(a,1:2255))==1)
19             continue
20         else
21             X(a,b) = 0;
22         end
23     end
24     for i=1:N %交配
25         x=unidrnd(scale,1,2);
26         % x = [unidrnd(2255) ...
27             ceil(2255+rand*14)];%针对第二次迭代，此时只让加进来的14个个体和初始个体交配
28         y=X(x(1),1:2255)|X(x(2),1:2255);
29         if(individuevaluation(y)==0)
30             continue
31         else
32             if (sum(X(x(1),:)) >= sum(X(x(2),:)))
33                 X(x(1),1:2255)=y;
34             else
35                 X(x(2),1:2255)=y;
36             end
37         end
38     end
39     count=[sum(X(1,:)) 1];
40     for i=1:scale
41         temp=sum(X(i,:));
42         if temp>=count(1)
43             count=[temp i];
44         end
45     end
46     disp([count,iter]);
47     if(iter==Lim1)
48         rightdata = X(count(2),:);
49     end
```

```

50     save save.mat rightdata;
51     save groupdata.mat X;
52 %     save secondsave.mat rightdata;
53 %     save second_group.mat X;
54 end
55 end
56 end

```

expanddata.m

```

1  load('A.mat');
2  load('expandA.mat');
3  col = 101;
4  for i = 1:100
5      for j = (expandA(i,1)+1):(expandA(i,3)-expandA(i,2)+expandA(i,1))
6          expandA(col,1) = j;
7          expandA(col,2) = j+expandA(i,2)-expandA(i,1);
8          expandA(col,3) = expandA(i,3);
9          expandA(col,4) = expandA(i,4);
10         col = col+1;
11     end
12 end

```

individualevaluation.m

```

1  function eval = individualevaluation(A)
2  %A为种群中一个个体，是一个1x2255的行向量，每个值为0或1
3  eval = 1;
4  scale = 2255;
5  %scale = 2269;
6  load('expandA.mat');
7  evalarray = zeros(5,2255);%用于储存判断数据的矩阵
8  evalarray(1,:) = A;
9
10 for i = 1:2255
11     if(A(i)==1)
12         evalarray(2:5,i) = expandA(i,:);
13     end
14 end
15
16 %先判断是否有分身
17 temp = 0;
18 for i = 1:2255
19     if(A(i)==1)
20         if(temp==evalarray(4,i))
21             eval = 0;
22             return
23         else
24             temp = evalarray(4,i);
25         end
26     end
27 end
28
29
30 timeaxis = zeros(1,178);%时间最大为178

```

```

31 for i = 1:2255
32     if(evalarray(1,i)==1)
33         for t = evalarray(2,i):(evalarray(3,i)-1)
34             timeaxis(t) = timeaxis(t)+1;
35         end
36         for u = evalarray(4,i):(evalarray(5,i)-1)
37             timeaxis(u) = timeaxis(u)+1;
38         end
39     end
40 end
41 for t = 1:178
42     if(timeaxis(t)>5)%将这里的1换成5则可以判断5个追踪器是否可行
43         eval = 0;
44         break;
45     end
46 end
47 %若A可行，则返回1，不行则返回0
48 end

```

8.2 数据汇总

我们的所有运行得到数据以.mat 文件的形式保存在支撑材料中，其中 expandA 表示扩充目标，first-group 表示初始迭代的种群，groupdata 表示第一次迭代终止时种群内所有个体情况。

具体得到的追踪器探测中心移动策略存放在“Problem A.xls”当中

8.3 文件清单

如图 10和图 11所示，所有需要的数据文件及源代码如显示排列，Excel 表单列。








名称	修改日期	类型	大小
 expandA	2019/5/17 4:24	Microsoft Acces...	2 KB
 first_group	2019/5/16 17:58	Microsoft Acces...	28 KB
 groupdata	2019/5/17 6:35	Microsoft Acces...	146 KB
 save	2019/5/17 6:35	Microsoft Acces...	1 KB
 second_group	2019/5/17 3:22	Microsoft Acces...	29 KB
 secondgroupdata	2019/5/17 2:57	Microsoft Acces...	151 KB
 secondsave	2019/5/17 7:09	Microsoft Acces...	1 KB

图 10: 数据集合

名称	修改日期	类型	大小
 expanddata.m	2019/5/13 21:47	M 文件	1 KB
 GA.m	2019/5/17 6:42	M 文件	2 KB
 individualevaluation.m	2019/5/16 21:01	M 文件	1 KB

图 11: 源代码集合