烟幕干扰弹的无人机投放策略研究

队伍编号：5049

2025 年 12 月 26 日

摘要

定点精确投放的烟幕干扰弹，可以在目标前方特定空域形成遮蔽，有效地干扰敌方导弹。本文针对无人机投放烟幕干扰弹以掩护目标的问题，建立了基于运动学和空间几何的数学模型。通过分析导弹与烟幕云团的相对运动与位置关系，获得了不同情境下的最优投放策略，以得到最大化有效遮蔽时间。

针对**问题**一，无人机 $FY1$ 飞行策略和投弹策略已知，我们建立了烟幕弹的运动与扩散模型。通过计算烟幕弹起爆时的空间位置与扩散半径，结合导弹 $M1$ 的运动轨迹，进行**时间步长离散化**仿真，同时利用**遮蔽判定指示函数**，从而得出有效遮蔽时长为 1.4s。

针对**问题**二，我们以无人机飞行方向、速度、投放点和起爆点为**决策变量**，以遮蔽时间为目标函数，建立了**非线性规划模型**。为提高求解效率，我们采用双层优化策略，确定了使得遮蔽时间最长的飞行参数和投放策略。

针对**问题**三，在问题二的基础上引入多枚烟幕弹，使问题演化为具有时间顺序约束的**多变量优化问题**。针对传统的优化算法在多变量情况下容易陷入停滞的情况，我们引入**灾变机制**的改进粒子群算法，求得多枚烟幕干扰弹的最优投放时序与起爆方案, 并将结果存放于 result1.xlsx 文件中。

针对**问题四**，任务变更为三架无人机共同拦截一枚导弹，那么核心问题在于**协调三架处于不同空间位置的无人机**的投弹策略。为此我们选择鲁棒性强的**差分进化算法 (DE)** 对多无人机联合决策变量求解，得到最优策略，并将结果存放于 result2.xlsx 文件中。

针对**问题五**，模型扩展为**多机多弹协同对抗多枚导弹**，同时涉及无人机飞行轨迹规划与烟幕弹投放时序选择，具有显著的**时空耦合特性和高维复杂性**。为此，我们沿用问题二中使用的双层优化模型，将轨迹规划与投弹调度问题进行有效解耦。**外层**采用**遗传算法**对无人机的飞行参数进行优化，以确定合理的空间位姿；**内层**在给定飞行轨迹的条件下，利用**贪心策略**对烟幕弹投放时机进行高效调度，在满足弹药数量及时间间隔约束的前提下最大化对

来袭导弹的遮蔽时间。最终得到了协同的最优策略，并保存在 result3.xlsx 文件中。

关键词：烟幕干扰；动态几何遮蔽；多目标优化；投放策略；多机协同

目录

1 问题重述

1.1 问题背景

现代战争中，利用无人机投放烟幕干扰弹是保护地面重要目标的有效手段，具有成本低、效费比高等优点。烟幕弹在空中起爆后形成云团，形成遮蔽，干扰敌方导弹。本题要求在给定的战场环境下（包含来袭导弹位置、无人机位置、目标位置及物理参数），综合考虑多个物体的运动轨迹，设计无人机的飞行与投放策略，以实现最大化的遮蔽效果。

图 1: 问题背景

1.2 问题提出

- **问题 1**：对于给定单无人机 $FY1$ 的具体飞行参数，根据运动学建立数学模型计算得出导弹 $M1$ 的有效遮蔽时长。
- **问题 2**：从问题一的特殊情况出发拓展到了一般情况，优化单无人机 $FY1$ 投放 1 枚烟幕弹的策略（方向，速度，投放点，起爆点），使遮蔽时间最大。
- **问题 3**：面对单无人机 $FY1$ 投放多枚烟幕弹的情况，规划烟雾弹的投递时序，实现总遮蔽时间的最大化。
- **问题 4**：不同于单机多枚烟雾弹的情形，情况变成了多机协同（$FY1-FY3$）各投 1 枚弹对抗 $M1$ ，需要找到其投放的最优策略。

- **问题 5**：全要素协同场景，需要找到（5 架无人机，每架最多 3 枚弹）对抗 3 枚导弹（$M1-M3$）的最优策略。

2 问题分析

2.1 问题一的分析

问题一是最简单的情景，要求计算得到给定参数下的有效遮蔽时间。首先以烟幕弹爆炸瞬间的空间位置与导弹的空间位置作为初始值，后续考虑扩散半径，结合导弹运动轨迹，求解两者的交点，从而得到有效遮蔽时长。

2.2 问题二的分析

问题二是问题一的推广，需要将无人机飞行方向、速度、投放点、起爆点作为决策变量进行优化，以最大化遮蔽时间作为目标函数，获取在满足一定的约束条件下的最佳解。实现烟幕弹的最佳投放策略。

2.3 问题三的分析

问题三是问题二的优化，需要控制好烟幕弹投放的时序，完成烟幕弹的协同投放，以实现多弹接力式的遮蔽效果，从而实现最大化有效遮蔽时间。

2.4 问题四的分析

问题四是多机协同的情况，需要考虑多架无人机在时空上的协同，实现在空间与时间上的遮蔽效果的最大化。

2.5 问题五的分析

问题五是最复杂的情景，涉及多机多弹对抗多枚导弹的情况。需要综合考虑无人机的任务分配、轨迹规划以及烟幕弹的投放时序等多个因素，建立一个综合性的优化模型，以实现整体遮蔽效益的最大化。但是直接考虑所有的决策变量，时间复杂度与空间复杂度较高，无法直接求解。因而需要对问题进行简化，将问题转化为多智能体协同任务分配与投放策略问题。

2.6 总体分析

根据上述的要求，我们进行了如下图2所示的工作：

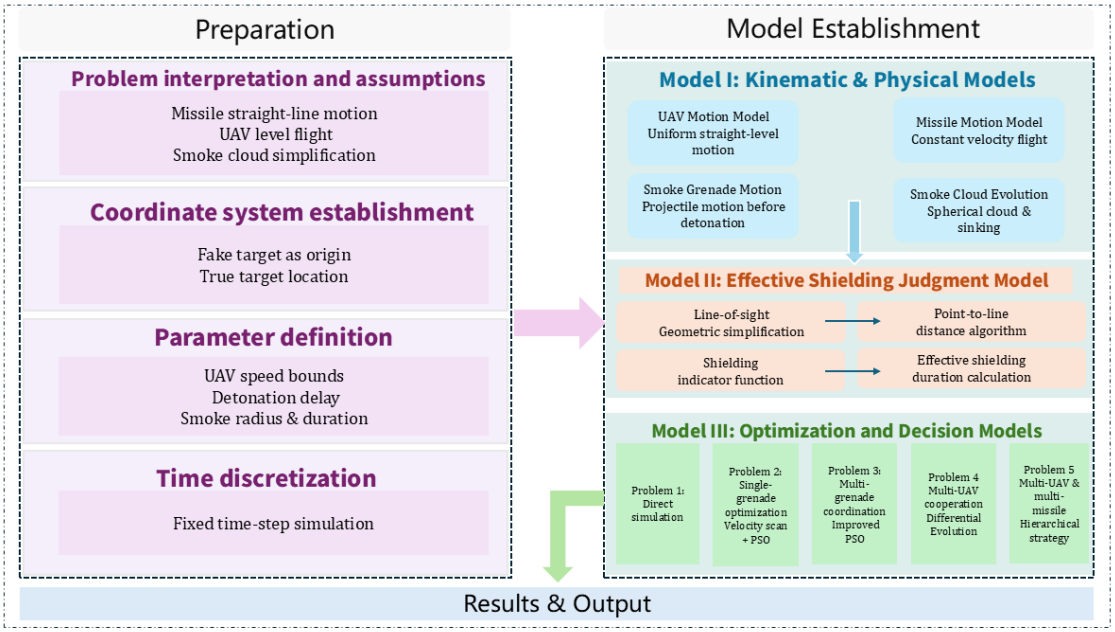


图 2: 总体分析框架

3 模型假设

为简化问题，主要假设如下：

1. 假设烟幕弹在投放后忽略空气阻力，仅受重力作用做平抛运动。
2. 假设烟幕云团形成球状且半径在有效时间内保持稳定（或按题目给定速度扩散/下沉）。
3. 假设来袭导弹做匀速直线运动，不进行机动变轨。
4. 假设雷达发现目标时刻为 $t = 0$。
5. 忽略无人机调整飞行方向时的转弯半径，视为瞬时变向。
6. 假设无人机、导弹、烟幕弹、烟幕云团均为质点。

4 符号说明

本文主要使用的符号定义如下表1所示：

表 1: 符号说明

符号	含义	单位
M_j	第 j 枚来袭导弹 $(j=1,2,3)$	-
FY_i	第 i 架无人机 $(i=1,2,3,4,5)$	-
V_m	导弹飞行速度 (固定 300)	m/s
v_{FY_i}	第 i 架无人机飞行速度	m/s
$\boldsymbol{v}_{FY_i}$	第 i 架无人机速度向量	m/s
$\boldsymbol{n}_{M_j}$	第 j 枚导弹飞行方向单位向量	-
α (α_i)	(第 i 架) 无人机飞行航向角	rad
g	重力加速度 (取 9.8)	m/s^2
$\boldsymbol{P}_O$	假目标位置 $(0,0,0)$	m
$\boldsymbol{P}_T$	真目标位置 $(0,200,5)$	m
$\boldsymbol{P}_{M_j}(t)$	第 j 枚导弹 t 时刻位置	m
$\boldsymbol{P}_{FY_i}(t)$	第 i 架无人机 t 时刻位置	m
$\boldsymbol{P}_{drop,k}$	第 k 枚烟幕弹投放位置	m
$\boldsymbol{P}_{burst,k}$	第 k 枚烟幕弹起爆位置	m
$\boldsymbol{P}_{S,k}(t)$	第 k 枚烟幕弹云团 t 时刻位置	m
t	当前时刻	s
t_{drop} $(t_{drop,k})$	(第 k 枚) 烟幕弹投放时刻	s
t_{burst} $(t_{burst,k})$	(第 k 枚) 烟幕弹起爆时刻	s
t_{delay} $(t_{delay,k})$	(第 k 枚) 烟幕弹起爆延时	s
Δt	离散化时间步长	s
T_{end}	终止时间	s
T_{cover}	有效遮蔽总时间	s
R_{cloud}	烟幕云团有效遮蔽半径 (固定 10)	m
v_{cloud}	烟幕云团下沉速度 (固定 3)	m/s
$D(t)$ $(D_{k,j}(t))$	(第 k 枚) 烟幕云团中心到 (第 j 枚) 导弹-目标连线距离	m
$I_j(t)$	第 j 枚导弹的遮蔽指示函数	-
$I_{i,k,j}(t)$	第 i 机第 k 弹对第 j 枚导弹遮蔽指示函数	-
$I_{total}(t)$	系统总遮蔽状态 (逻辑或)	-
$S_j(t)$	整个系统中第 j 枚导弹的遮蔽状态	-
X $(\mathbf{X})$	决策变量向量	-
$J(X)$	目标函数 (遮蔽时间)	s
F	差分进化缩放因子	-
λ	惩罚函数权重系数	-
N (NP)	种群规模/粒子数	-
G_{max}	最大迭代次数	-

5 模型建立

来袭导弹在飞行过程中始终以真目标为制导对象，其探测与制导过程可近似理解为沿导弹当前位置指向目标的视线进行信息获取。无人机投放的烟幕干扰弹在起爆后形成高浓度烟幕云团，通过遮挡导弹对目标的直接探测视线，从而降低导弹对目标的识别与锁定能力。因此，烟幕干扰效果的本质并不取决于烟幕云团是否“接触”导弹或目标本身，而取决于烟幕云团在空间中是否覆盖导弹–目标之间的关键视线区域。基于上述认识，本文将烟幕干扰问题的核心抽象为一个**随时间变化的空间几何遮挡判定问题**，并在此基础上进一步量化有效遮蔽时间。根据题目描述首先建立统一的三维直角坐标系 $Oxyz$。以假目标中心为原点 x 轴与 y 轴位于水平面内，z 轴竖直向上则假目标位置为 $P_O = (0, 0, 0)$, 真目标位于点 $P_T = (0, 200, h_T)$ 其中 $h_T = 5m$ 为真目标中心高度。

5.1 导弹运动模型构建

根据题目的描述，来袭的导弹始终锁定假目标，将其作为攻击对象，题目给定导弹 M1-M3 的初始位置为，并且他们的飞行速度固定为 $V_m = 300\,\mathrm{m/s}$，同时假目标位于坐标原点 $(0, 0, 0)$。因为在问题一至问题五中，导弹 M1、M2、M3 的运动规律相同，仅初始位置不同。所以为统一描述多枚导弹的运动特性，需建立通用的导弹运动数学模型。首先我们考虑第 j 枚导弹的初始位置为点 $P_{M_j}(0)$，那么该导弹的初始方向向量为:

$$\boldsymbol{n}_{\boldsymbol{M_j}} = \frac{\boldsymbol{P_O} - \boldsymbol{P}_{M_j}(0)}{||\boldsymbol{P_O} - \boldsymbol{P_{M_j}}(0)||} \tag{1}$$

因为导弹以速度 V_m 做匀速直线运动，那么我们就能获取它任意时刻 t 的位置:

$$\boldsymbol{P}_{M_j}(t) = \boldsymbol{P}_{M_j}(0) + V_m \cdot t \cdot \boldsymbol{n}_{M_j} \tag{2}$$

5.2 无人机运动模型构建

根据题目描述，我们可以获取无人机 FY1-FY5 的初始位置，FY1(17800,0,1800)、FY2(12000,1400,1400)、FY3(6000,−3000,700)、FY4(11000,2000,1800)、FY5(13000,−2000,1300)，他们的飞行速度范围为 $[70, 140]\,\mathrm{m/s}$，并且保持等高度匀速直线飞行在问题一中，无人机速度为 $120\,\mathrm{m/s}$，指向假目标飞行；在问题二至问题五中，飞行速度 v_{FY_i} 和航向角 α_i 均为待优化的决策变量。题目强调了” 无人机一旦受领任务，其速度和航向就确定了且不再调整”，这意味着无人机在整个任务过程中保持匀速等高直线飞行。那么，我们可以将第 i 枚无人机的初始位置用 $\boldsymbol{P}_{FY_i}(0)$，飞行参数用飞行速度 v_{FY_i}，航向角 α_i 来表示, 因而我们可以得到他的速度向量，表示为:

$$\boldsymbol{v}_{FY_i} = v_{FY_i}(\cos\alpha_i, \sin\alpha_i, 0) \tag{3}$$

从而，我们可以确定无人机在任意时刻 t 的位置为：

$$\boldsymbol{P}_{FY_i}(t) = \boldsymbol{P}_{FY_i}(0) + \boldsymbol{v}_{FY_i} t \tag{4}$$

5.3 烟幕干扰弹的运动模型

根据题目描述，烟幕干扰弹具有以下特性：

投放延时 t_{drop}：无人机受领任务后多久投放烟幕弹

起爆延时 $t_{delay} \in [1, 8]\,\mathrm{s}$：烟幕弹投放后多久起爆

烟幕半径 $R_{cloud} = 10\,\mathrm{m}$：起爆后形成的烟幕云团有效遮蔽半径

下沉速度 $v_{cloud} = 3\,\mathrm{m/s}$：烟幕云团匀速下沉的速度

有效时间：烟幕云团在起爆后 20 秒内保持有效遮蔽能力

同时烟幕弹的运动也能被分成两个部分，投放阶段（投放至起爆）与爆炸阶段（起爆后）。那么，第 k 枚烟幕弹在 $t_{drop,k}$ 时刻投放，经过延时为 $t_{delay,k}$，重力加速度为 g 的运动后，可以将该运动分解为竖直方向上的自由落体运动与水平方向的匀速直线运动，如下图3所示，同时可以得到起爆时刻为：

$$t_{burst,k} = t_{drop,k} + t_{delay,k} \tag{5}$$

起爆位置为

$$\boldsymbol{P}_{burst,k} = \boldsymbol{P}_{FY_i}(t_{drop,k}) + \boldsymbol{v}_{FY_i} t_{delay,k} - \left(0, 0, \frac{1}{2} g t_{delay,k}^2\right) \tag{6}$$

烟幕弹在起爆后，形成半径为 R_{cloud} 的球形云团，其中心以恒定速度 v_{cloud} 匀速下沉，那么我们可以得到该枚烟雾弹在起爆后任意时刻的位置为：

$$\boldsymbol{P}_{S,k}(t) = \boldsymbol{P}_{burst,k} - (0, 0, v_{cloud}(t - t_{burst,k})), \quad t \geq t_{burst,k} \tag{7}$$

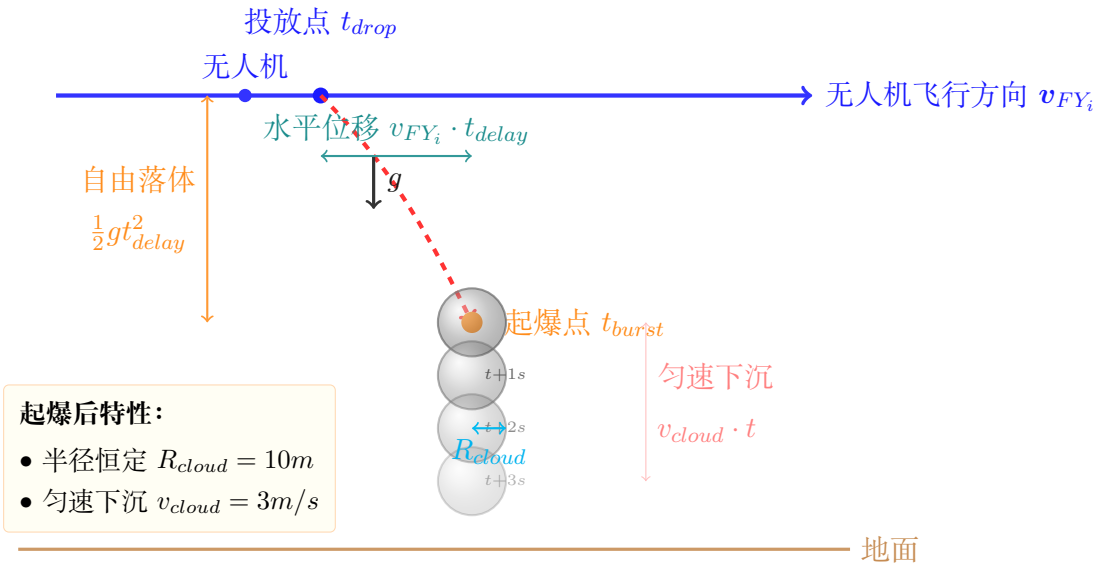


图 3: 烟幕弹运动与云团下沉示意图

5.4 有效遮蔽判定模型

5.4.1 几何遮挡条件

问题背景与简化假设 根据题目给定的初始条件，3 枚导弹 M1、M2、M3 分别位于 (20000,0,2000)、(19000,600,2100)、(18000,−600,1900)，无人机 FY1 位于 (17800,0,1800)、FY2 位于 (12000,1400,1400)、FY3 位于 (6000,−3000,700)、FY4 位于 (11000,2000,1800)、FY5 位于 (13000,−2000,1300)，而真目标（底面圆心位于 $(0, 200, 0)$，底面半径 7 m，高度 10 m 的圆柱体）距离二者均在数万米量级。在如此远距离下，目标的几何尺寸（半径 7 m）相对于视线长度（约 20 km）可忽略不计。基于以下合理假设，本文对遮蔽判定模型进行简化。

1. **目标质点化**：将真目标简化为其质心点 $\boldsymbol{P}_T = (0, 200, 5)$（底面圆心加上半高 5 m），忽略其几何尺寸的影响。这一简化在远距离光学/红外探测中是合理的，符合几何光学的小角度近似。

2. **导弹视野假设**：题目未明确限定导弹的视场角，结合实际惯性制导导弹的特点，可认为导弹探测器的视场足够大，能够覆盖目标方向的合理区域。因此，只需考虑视线方向上的遮蔽，而不必考虑视场边界。

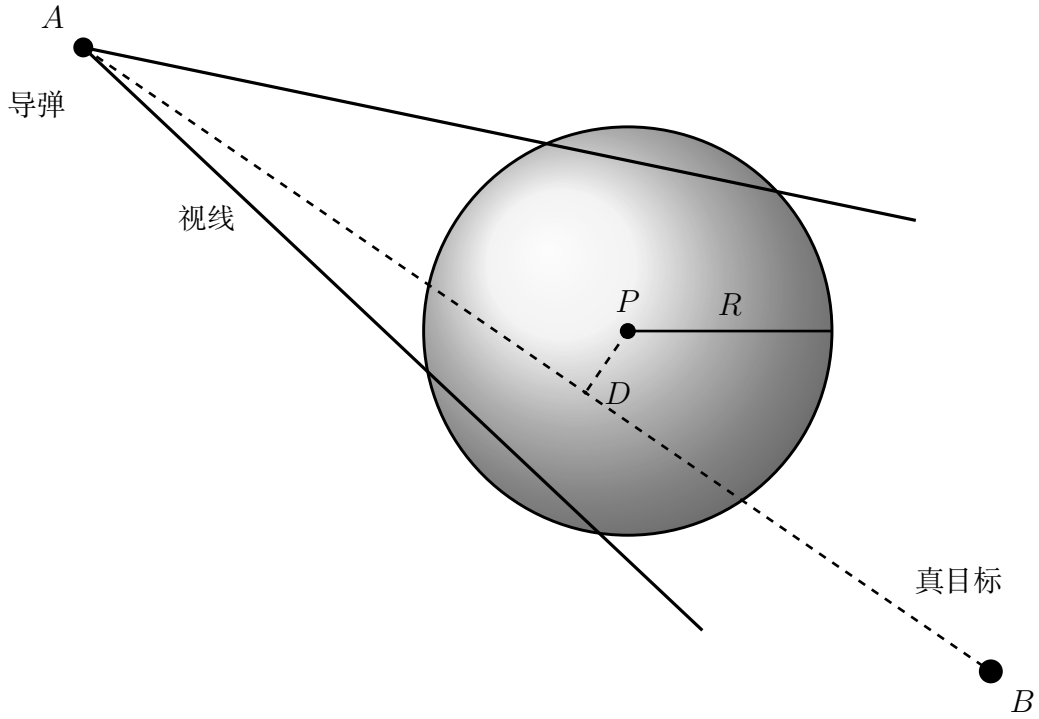
3. **视线线段化**：导弹的光学探测路径可抽象为连接导弹位置 A 与目标位置 B 的直线段 AB。当烟幕云团中心与该线段的空间距离小于烟幕半径时，视线被有效遮挡。

基于上述简化，简化的遮挡示意关系如下图4所示，只需 $D<=R$ 时，导弹视线被遮蔽。那么有效遮蔽的充要条件可表述为：

$$D(t) \leq R_{cloud} \tag{8}$$

其中，$D(t)$ 为时刻 t 时烟幕云团中心 $\boldsymbol{P}_S(t)$ 到视线线段 $P_M(t)P_T$ 向量的**最短距离**（点到线段距离），$R_{cloud} = 10\,\text{m}$ 为烟幕云团的有效遮蔽半径。那么 $D(t)$ 的求解尤为重要，下面我们将介绍点到线段距离的计算方法。

图 4: 导弹-烟幕弹-目标遮蔽模型示意图



5.4.2 点到线段距离算法

设导弹位置为点 A，真目标位置为点 B，烟幕弹中心为点 P。向量 $\overrightarrow{AB} = B - A$，向量 $\overrightarrow{AP} = P - A$。投影系数 r 计算公式为：

$$r = \frac{\overrightarrow{AP} \cdot \overrightarrow{AB}}{\|\overrightarrow{AB}\|^2} \tag{9}$$

距离分三种情况讨论，如下图5：

$$D(t) = \begin{cases} \|\overrightarrow{AP}\| & \text{if } r \le 0 \text{ (导弹后方)} \\ \|P - B\| & \text{if } r \ge 1 \text{ (目标后方)} \\ \frac{\|\overrightarrow{AP} \times \overrightarrow{AB}\|}{\|\overrightarrow{AB}\|} & \text{if } 0 < r < 1 \text{ (线段投影内)} \end{cases} \tag{10}$$

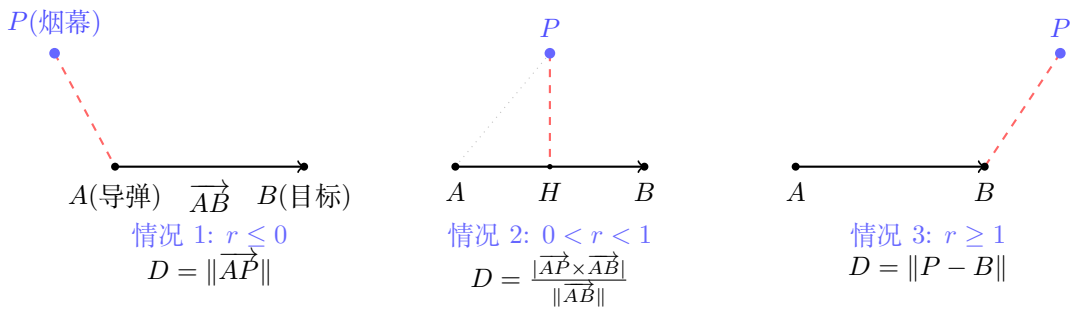


图 5: 点到线段距离的三种计算情况

5.5 遮蔽时间模型

5.5.1 遮蔽指示函数

为定量描述烟幕云团对导弹的遮蔽效果，引入**遮蔽指示函数** $I_j(t)$，用于表征时刻 t 时第 j 枚导弹是否被有效遮蔽。该函数定义为：

$$I_j(t) = \begin{cases} 1, & \text{if } D_{k,j}(t) \le R_{cloud} \text{ and } t \ge t_{burst,k} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

其中 $D_{k,j}(t)$ 为第 k 枚烟幕弹对第 j 枚导弹在时刻 t 的点到线段距离，$R_{cloud} = 10\,\mathrm{m}$：烟幕云团的有效遮蔽半径，$t_{burst,k}$ 为第 k 枚烟幕弹的起爆时刻。

该指示函数具有以下物理含义：

1. 当 $t < t_{burst,k}$ 时，烟幕弹尚未起爆，$I_j(t) = 0$（无遮蔽）
2. 当 $t \ge t_{burst,k}$ 且 $D_{k,j}(t) \le R_{cloud}$ 时，烟幕云团有效遮挡视线，$I_j(t) = 1$（有效遮蔽）
3. 当 $D_{k,j}(t) > R_{cloud}$ 时，烟幕云团与视线距离过远，$I_j(t) = 0$（遮蔽失效）

5.5.2 总有效遮蔽时长

对于单枚导弹 j，在整个飞行过程 $[0, T_{end}]$ 内的总有效遮蔽时长定义为指示函数的时间积分。

$$T_j = \int_0^{T_{end}} I_j(t)\, dt \tag{12}$$

其中 T_{end} 为导弹飞行终止时刻。通过该积分，我们就可以统计出导弹在飞行全程中被烟幕有效遮蔽的累计时间。

在多枚导弹（$M_1, M_2, \ldots, M_m$）协同拦截的场景中，系统的总体优化目标定义为所有导弹的遮蔽时长之和：

$$\max T_{cover} = \sum_{j=1}^{m} T_j \tag{13}$$

该目标函数体现了“最大化整体防护效果”的战术需求。

5.5.3 时间离散化策略

由于系统涉及导弹、烟幕云团等多个实体的连续时变运动，且遮蔽判定条件包含复杂的几何关系与非线性逻辑，无法通过解析方法直接求解遮蔽时间积分 $\int_0^{T_{end}} I_j(t)dt$。因此，本文采用时间离散化方法，如下图6所示，将连续时间轴划分为等间隔的离散时刻：

$$t_k = k\Delta t, \quad k = 0, 1, 2, \ldots, N \tag{14}$$

其中，Δt 为时间步长，$N = \lfloor T_{end}/\Delta t \rfloor$。

在每个离散时刻 t_k，首先根据运动学方程，更新导弹位置 $\boldsymbol{P}_{M_j}(t_k)$ 和烟幕云团中心位置 $\boldsymbol{P}_{S,k}(t_k)$ 然后计算点到线段距离 $D_{k,j}(t_k)$ 接着根据遮蔽判定条件，确定遮蔽状态 $I_j(t_k) \in \{0,1\}$ 最终，有效遮蔽时长可通过**黎曼和**近似为：

$$T_{cover} \approx \sum_{k=0}^{N} I(t_k) \cdot \Delta t \tag{15}$$

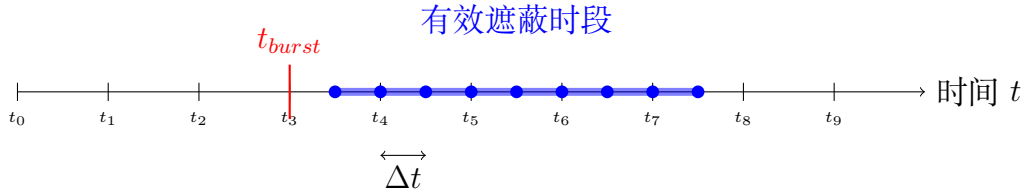
该公式本质上是定积分的数值计算，当 $\Delta t \to 0$ 时，离散求和收敛于连续积分的精确值。

时间步长的选取需权衡计算精度与效率：

- **粗扫描阶段**（初步搜索）：$\Delta t = 0.1s$，快速定位潜在遮蔽区间
- **精细计算阶段**（优化求解）：$\Delta t = 0.01s$ 或 $0.001s$，确保精度

该方法不仅适用于单目标单弹场景，在多机多弹协同问题中也能高效处理时间窗口重叠与遮蔽并集的计算。

图 6: 时间离散化与遮蔽时间累加示意



6 模型求解

6.1 问题一求解：固定参数

问题一针对的是具体场景在给定所有的参数下 (烟幕弹的投放时间，起爆时间)，求解得出单枚烟幕弹对单枚导弹的有效遮蔽时间，需要我们利用各个物体的位置函数，结合有效遮挡判据得到有效遮挡时间。题目给定了无人机 FY1 的飞行策略（速度、航向）和投弹策略（投放时刻、起爆延时），本问的核心任务是**仿真验证**已知策略的有效性。与后续优化问题（问题二至五）不同，问题一无需搜索最优解，而是基于确定性输入进行数值模拟。因此，采用**时间离散化仿真方法**即可直接求解。

求解步骤概览

1. **参数代入**：将题目给定的所有物理参数代入运动学方程
2. **轨迹计算**：计算烟幕弹起爆位置及各实体的时变位置

3. **遮蔽判定**：逐时刻检查烟幕云团是否有效遮蔽导弹视线

4. **时长统计**：累计满足遮蔽条件的时间段，得到总遮蔽时长

6.1.1 参数代入与场景设定

根据题目条件，各实体的初始参数如下：

无人机 FY1 参数

- 初始位置：$\mathbf{P}_{FY_1}(0) = (17800,\ 0,\ 1800)\,\mathrm{m}$
- 飞行速度：$v_{FY_1} = 120\,\mathrm{m/s}$
- 飞行航向：指向假目标 $\boldsymbol{P}_O = (0, 0, 0)$，即速度向量 $\boldsymbol{v}_{FY_1} = (-120, 0, 0)\,\mathrm{m/s}$

导弹 M1 参数

- 初始位置：$\mathbf{P}_{M_1}(0) = (20000,\ 0,\ 2000)\,\mathrm{m}$
- 飞行速度：$V_m = 300\,\mathrm{m/s}$（沿指向假目标的方向）
- 飞行方向：单位向量 $\mathbf{n}_{M_1} = \frac{\boldsymbol{P}_O - \mathbf{P}_{M_1}(0)}{\|\boldsymbol{P}_O - \mathbf{P}_{M_1}(0)\|}$

烟幕弹投放策略

- 投放时刻：$t_{drop} = 1.5\,\mathrm{s}$（受领任务后 1.5 秒投放）
- 起爆延时：$t_{delay} = 3.6\,\mathrm{s}$（投放后 3.6 秒起爆）
- 起爆时刻：$t_{burst} = t_{drop} + t_{delay} = 5.1\,\mathrm{s}$

烟幕云团物理参数

- 有效半径：$R_{cloud} = 10\,\mathrm{m}$
- 下沉速度：$v_{cloud} = 3\,\mathrm{m/s}$
- 有效时间：起爆后 20 秒内保持遮蔽能力

6.1.2 运动轨迹计算

由模型建立中的运动模型，可确定系统中各实体的空间位置随时间的变化情况。无人机在投放时刻的空间位置为 $\mathbf{P}_{FY_1} = \mathbf{P}_{FY_1}(t_{drop})$., 烟幕干扰弹的起爆时刻为

$$t_{burst} = t_{drop} + t_{delay}, \tag{16}$$

则其起爆位置为

$$\mathbf{P}_{burst} = \mathbf{P}_{FY_1}(t_{drop}) + \boldsymbol{v}_{FY_1} t_{delay} - \left(0,\, 0,\, \tfrac{1}{2} g t_{delay}^2\right). \tag{17}$$

起爆后，烟幕云团中心以恒定速度沿竖直方向下沉，其在任意时刻 t 的位置为

$$\mathbf{P}_S(t) = \mathbf{P}_{burst} - (0,\, 0,\, v_{cloud}(t - t_{burst})), \quad t \geq t_{burst}. \tag{18}$$

导弹 M1 在任意时刻 t 的位置为

$$\mathbf{P}_M(t) = \mathbf{P}_M(0) + V_M \mathbf{n}_M t, \tag{19}$$

其中 $\mathbf{n}_M$ 为导弹指向假目标的单位方向向量。

6.1.3 有效遮蔽时长计算

为计算有效遮蔽时长，将时间区间 $[0, T_{end}]$ 离散化。取时间步长 $\Delta t = 0.05\,\mathrm{s}$，终止时间 T_{end} 为导弹到达假目标的时间。离散时刻定义为：

$$t_k = k\Delta t, \quad k = 0, 1, \ldots, N, \quad N = \lfloor T_{end}/\Delta t \rfloor \tag{20}$$

逐时刻遮蔽判定 在每个离散时刻 t_k，执行以下判定流程：

1. **时间窗口检查**：若 $t_k < t_{burst}$ 或 $t_k > t_{burst} + 20$，则烟幕弹尚未起爆或已失效，$I(t_k) = 0$
2. **空间位置更新**：计算 $\mathbf{P}_{M_1}(t_k)$、$\mathbf{P}_S(t_k)$ 和真目标位置 $\mathbf{P}_T = (0, 200, 5)$
3. **距离计算**：计算烟幕云团中心到” 导弹-目标” 连线的最短距离 $D(t_k)$（点到线段距离算法）
4. **遮蔽判断**：若 $D(t_k) \leq R_{cloud} = 10\,\mathrm{m}$，则 $I(t_k) = 1$；否则 $I(t_k) = 0$

遮蔽指示函数定义为：

$$I(t_k) = \begin{cases} 1, & D(t_k) \leq R_{cloud} \text{ 且 } t_k \in [t_{burst}, t_{burst} + 20], \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

总时长统计 累计所有满足遮蔽条件的时间点，得到总有效遮蔽时长：

$$T_{cover} = \sum_{k=0}^{N} I(t_k) \cdot \Delta t \tag{22}$$

6.1.4 模型汇总

综合上述分析，问题一的求解模型可归纳为：

$$\begin{cases}
\textbf{输入参数}: & \mathbf{P}_{FY_1}(0),\ v_{FY_1},\ t_{drop},\ t_{delay},\ \mathbf{P}_{M_1}(0),\ V_m \\
\textbf{导弹运动}: & \mathbf{P}_{M_1}(t) = \mathbf{P}_{M_1}(0) + V_m \mathbf{n}_{M_1} \cdot t \\
\textbf{无人机运动}: & \mathbf{P}_{FY_1}(t) = \mathbf{P}_{FY_1}(0) + \boldsymbol{v}_{FY_1} \cdot t \\
\textbf{起爆位置}: & \mathbf{P}_{burst} = \mathbf{P}_{FY_1}(t_{drop}) + \boldsymbol{v}_{FY_1} \cdot t_{delay} - (0, 0, \frac{1}{2} g t_{delay}^2) \\
\textbf{云团运动}: & \mathbf{P}_S(t) = \mathbf{P}_{burst} - (0, 0, v_{cloud}(t - t_{burst})), \quad t \geq t_{burst} \\
\textbf{距离计算}: & D(t) = \begin{cases} \|\mathbf{P}_S(t) - \mathbf{P}_{M_1}(t)\| & \text{if } r \leq 0 \\ \|\mathbf{P}_T - \mathbf{P}_S(t)\| & \text{if } r \geq 1 \\ \frac{\|(\mathbf{P}_S(t) - \mathbf{P}_{M_1}(t)) \times (\mathbf{P}_T - \mathbf{P}_{M_1}(t))\|}{\|\mathbf{P}_T - \mathbf{P}_{M_1}(t)\|} & \text{if } 0 < r < 1 \end{cases} \\
\textbf{遮蔽判定}: & I(t) = \begin{cases} 1, & D(t) \leq R_{cloud} \text{ 且 } t_{burst} \leq t \leq t_{burst} + 20 \\ 0, & \text{otherwise} \end{cases} \\
\textbf{输出结果}: & T_{cover} = \int_0^{T_{end}} I(t)\, dt \approx \sum_{k=0}^{N} I(t_k) \cdot \Delta t
\end{cases} \tag{23}$$

其中，投影系数 $r = \frac{(\mathbf{P}_S(t) - \mathbf{P}_{M_1}(t)) \cdot (\mathbf{P}_T - \mathbf{P}_{M_1}(t))}{\|\mathbf{P}_T - \mathbf{P}_{M_1}(t)\|^2}$。

6.1.5 求解结果：

经计算，结果如下图7所示，清晰展示了烟幕云团的生效时间段。其中烟幕弹起爆时刻：$t = 5.10s$, 烟幕起爆初始坐标：$(17188.00, 0.00, 1736.50)$ 有效遮蔽时间段为 $8.1 - 9.5s$，总时长为 $1.4s$。

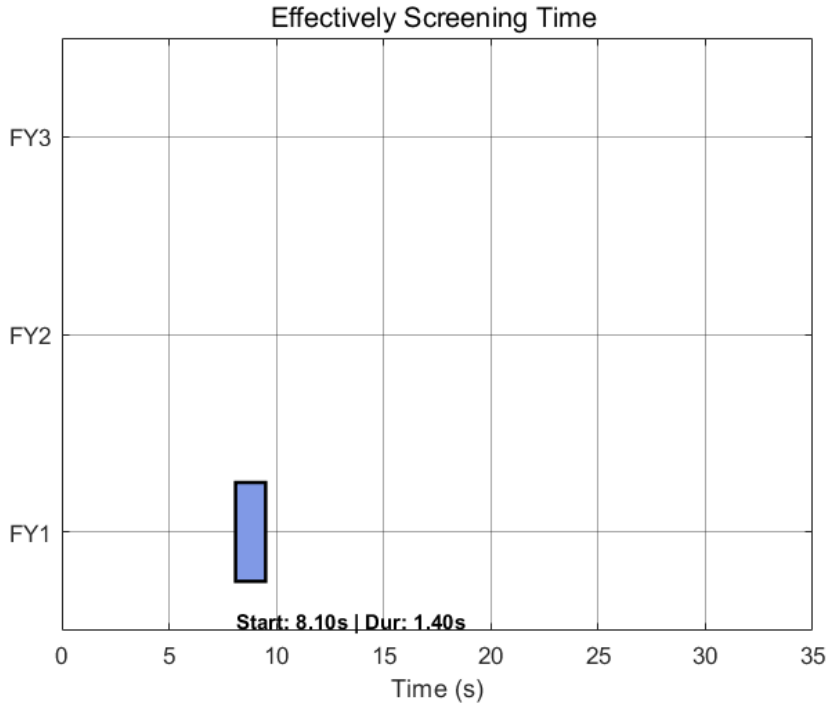


图 7: 问题一结果

6.2 问题二求解：单机单弹优化

问题二是问题一的泛化与扩展，与问题一给定参数仿真验证不同，问题二要求在**参数可调**的条件下，优化无人机 FY1 的飞行策略与投弹策略，使得单枚烟幕弹对导弹 M1 的有效遮蔽时间达到最大。本问的核心挑战是：决策变量由 0 个（问题一全部给定）增加到 4 个，并且目标函数 $T_{cover}(X)$ 高度非线性、非凸，无法通过解析方法求解。因此，需建立**非线性规划模型**，并采用智能优化算法进行数值求解。

6.2.1 决策变量

根据题目要求，无人机一旦受领任务，其速度和航向就确定了且不再调整。因此在进行任务决策时我们需要决策以下四个变量：如下表2所示

表 2: 问题二决策变量说明

变量	物理意义	取值范围
v_{FY_1}	无人机飞行速度	$[70, 140]$ m/s
α	无人机航向角（相对 x 轴）	$[0, 2\pi)$ rad
t_{drop}	烟幕弹投放时刻	$[0, 12]$ s
t_{delay}	烟幕弹起爆延时	$[1, 8]$ s

定义决策向量为：

$$X = [v_{FY_1},\, \alpha,\, t_{drop},\, t_{delay}]^T \in \mathbb{R}^4 \tag{24}$$

这四个变量并非相互独立，而是通过物理约束相互耦合：

- **速度-航向耦合**：v_{FY_1} 和 α 共同决定无人机的飞行轨迹，进而影响烟幕弹的投放位置 $\boldsymbol{P}_{drop}$
- **时间-空间耦合**：t_{drop} 决定投放位置，t_{delay} 决定起爆位置，两者共同影响烟幕云团与导弹视线的相对位置关系
- **高度约束耦合**：t_{delay} 过大会导致烟幕弹落地后起爆（违反物理约束），因此 t_{delay} 的可行域受 v_{FY_1}、α、t_{drop} 的间接约束

6.2.2 状态方程

基于问题一建立的运动学模型，将决策变量 X 参数化地嵌入状态方程
将标量速度 v_{FY_1} 和航向角 α 转换为三维速度向量：

$$\boldsymbol{v}_{FY_1}(v_{FY_1}, \alpha) = [v_{FY_1}\cos\alpha,\, v_{FY_1}\sin\alpha,\, 0]^T \tag{25}$$

投放点位置（依赖于 v_{FY_1}、α、t_{drop}）

$$\boldsymbol{P}_{drop}(X) = \boldsymbol{P}_{FY_1}(0) + \boldsymbol{v}_{FY_1}(v_{FY_1}, \alpha) \cdot t_{drop} \tag{26}$$

起爆点位置（依赖于全部四个决策变量）：

$$\boldsymbol{P}_{burst}(X) = \boldsymbol{P}_{drop}(X) + \boldsymbol{v}_{FY_1}(v_{FY_1}, \alpha) \cdot t_{delay} + [0, 0, -\tfrac{1}{2}gt_{delay}^2]^T \tag{27}$$

烟幕云团中心轨迹（$t \geq t_{burst}$）：

$$\boldsymbol{P}_S(t; X) = \boldsymbol{P}_{burst}(X) - [0, 0, v_{cloud}(t - t_{burst})]^T \tag{28}$$

6.2.3 目标函数

目标是最大化有效遮蔽时长，利用问题一中定义的遮蔽判定指示函数得出目标函数：

$$max J\,(X) = T_{cover} = \int_0^{T_{end}} I\,(t; X)\, dt \tag{29}$$

其中，遮蔽指示函数依赖于决策变量 X：

$$I(t; X) = \begin{cases} 1, & D(t; X) \leq R_{cloud} \text{ 且} t \geq t_{burst}(X) \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

$D(t; X)$ 为烟幕云团中心 $\boldsymbol{P}_S(t; X)$ 到导弹视线的距离（点到线段距离）。

6.2.4 约束条件

根据题目要求和对于模型速度特征的分析，约束条件如下:

速度约束：无人机受领任务后，瞬时调整飞行方向，然后以 70 140 m/s 的速度等高度匀速直线飞行

$$70 \leq v_{FY_1} \leq 140 \tag{31}$$

时间约束：投放时刻上限 12 秒考虑到导弹飞行速度，过晚投放可能错过拦截时机。

$$0 \leq t_{drop} \leq 12, 1 \leq t_{delay} \leq 8 \tag{32}$$

高度约束：烟幕弹必须在空中起爆，因此起爆点的高度必须大于零

$$z_{burst}\left(X\right) > 0 \tag{33}$$

6.2.5 模型汇总

综合上述分析，问题二的完整优化模型可表示为:

$$\begin{cases} \text{决策变量:} \quad X = [v_{FY_1}, \alpha, t_{drop}, t_{delay}]^T \\ \text{目标函数:} \quad \max\ T_{cover}(X) = \displaystyle\int_0^{T_{end}} I(t;X)dt \\ \text{状态方程:} \quad \begin{cases} \boldsymbol{v}_{FY_1} = v_{FY_1}[\cos\alpha, \sin\alpha, 0]^T \\ \boldsymbol{P}_{drop} = \boldsymbol{P}_{FY_1}(0) + \boldsymbol{v}_{FY_1} \cdot t_{drop} \\ \boldsymbol{P}_{burst} = \boldsymbol{P}_{drop} + \boldsymbol{v}_{FY_1} \cdot t_{delay} + [0, 0, -\frac{1}{2}gt_{delay}^2]^T \\ \boldsymbol{P}_S(t) = \boldsymbol{P}_{burst} - [0, 0, v_{cloud}(t - t_{burst})]^T \end{cases} \\ \text{遮蔽判定:} \quad I(t;X) = \begin{cases} 1, & D(t) \leq R_{cloud} \text{ 且} t_{burst} \leq t \leq t_{burst} + 20 \\ 0, & \text{otherwise} \end{cases} \\ \text{约束条件:} \quad \begin{cases} 70 \leq v_{FY_1} \leq 140 \\ 0 \leq t_{drop} \leq 12, \quad 1 \leq t_{delay} \leq 8 \\ z_{burst}(X) > 0 \end{cases} \end{cases} \tag{34}$$

6.2.6 求解算法

由于目标函数涉及复杂的几何运动及分段逻辑，具有高度的非线性和非凸性，且决策变量相对于一般的单变量优化问题具有高自由度。所以为了解决此类问题，本问题采用双层优化策略，外层使用网格扫描，内层使用粒子群优化算法。这样使得我们将优化问题维度从 4 维降至 3 维，降低了搜索难度降低问题陷入局部最优的概率。

外层循环:

我们采用了**网格扫描法**，首先将速度 v_{FY_1} 在区间 $[70, 140]$ 上以步长 $\Delta v = 0.5\,\mathrm{m/s}$ 离散化

$$v_{FY_1} \in \{70.0,\ 70.5,\ 71.0,\ \ldots,\ 139.5,\ 140.0\} \tag{35}$$

然后对每个离散速度值，调用内层 PSO 算法求解子问题

$$\max_{(\alpha, t_{drop}, t_{delay})} T_{cover}(v_{FY_1}, \alpha, t_{drop}, t_{delay}) \tag{36}$$

记录每个 v_{FY_1} 对应的最优解 $(\alpha^*, t^*_{drop}, t^*_{delay})$ 及其目标函数值，最终选择全局最优。

内层循环:

我们采用了粒子群优化算法（PSO），我们对首先对粒子进行 **粒子编码**，将第 i 个粒子的位置用 $x_i = [\alpha_i, t_{drop,i}, t_{delay,i}]$, 速度用 v_i 来表示；那么每个粒子表示一个候选解 x_i，然后我们进行**参数设置**, 将种群规模设置为 N 置为 30，最大迭代次数 G_{max} 置为 50 次，惯性权重 $w =$ 设为 0.6，从而平衡全局与局部搜索，学习因子 $c_1 = c_2$ 设置为 1.5，从而平衡个体认知与社会学习。同时粒子的**速度与位置更新公式**为

$$v_i^{k+1} = w \cdot v_i^k + c_1 r_1 \odot (p_{best,i} - x_i^k) + c_2 r_2 \odot (g_{best} - x_i^k) \tag{37}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{38}$$

其中，$r_1, r_2 \sim \mathrm{Uniform}(0, 1)$ 为随机数，$\odot$ 表示逐元素乘法，$p_{best,i}$ 为粒子 i 的历史最优位置，g_{best} 为全局最优位置。

在每次迭代中，算法通过上述更新公式调整粒子位置，并通过适应度函数评估每个粒子解的优劣，从而更新个体历史最优 $p_{best,i}$ 和全局最优 g_{best}。**适应度函数** $F(x)$ 定义为遮蔽时间的负值（转化为最小化问题）:

$$F(x) = -T_{cover}(x) \tag{39}$$

算法持续迭代直至达到最大迭代次数或收敛条件，最终输出全局最优解 g_{best} 及对应的最大遮蔽时间。

6.2.7 求解结果

基于 MATLAB 实现的双层优化算法（代码见附录）, 得到最优投放策略如下:
无人机以 176.91° 的航向角度，72m/s 的速度，0s 时刻投放，2.4991s 后起爆，有效遮蔽时长为 4.738s, 如下图8所示。相比问题一的 1.4s 提升 238%

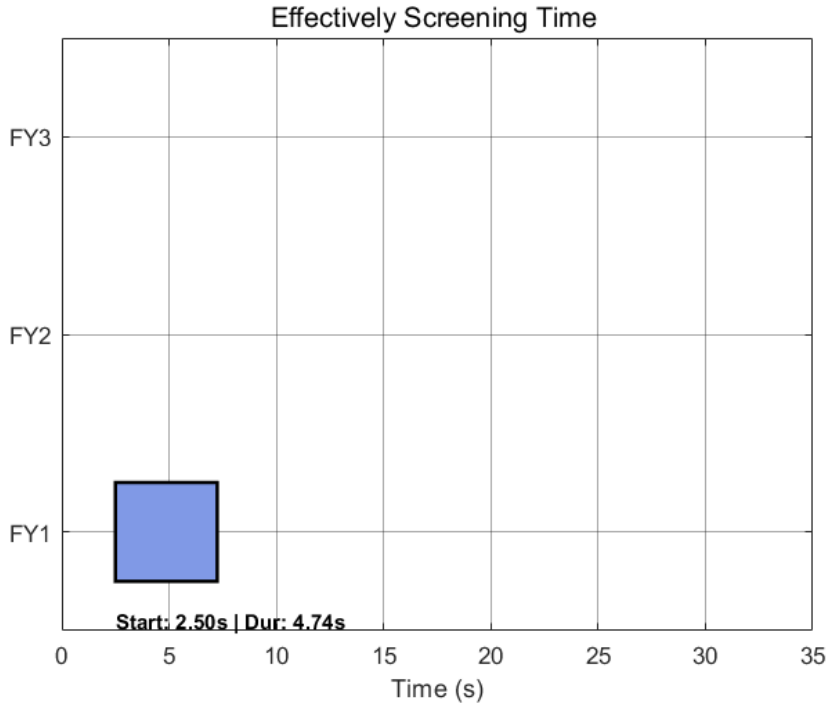


图 8: 问题二结果

6.3 问题三求解：多弹协同策略

在本题中，无人机 FY1 需要连续投放 3 枚烟幕弹以对导弹 M1 实施干扰。与投放一枚烟幕弹不同，三枚烟幕弹的投弹策略涉及时序上的配合，核心问题在对多参数的优化使得烟幕弹的遮蔽时间最大

6.3.1 决策变量

根据题目要求，无人机一旦受领任务，其速度和航向就确定了且不再调整。因此在进行任务决策时我们需要决策以下八个变量3: 飞行速度 v_{FY_1} , 飞行航向角 α , 投放时刻 $t_{drop,i}\quad (i \in \{1,2,3\})$, 延时起爆时间 $\quad t_{delay,i}\quad ,(i \in \{1,2,3\})$, 由此定义出决策向量 $\mathbf{X}$:

$$\mathbf{X} = [v_{FY_1},\, \alpha,\, t_{drop,1},\, t_{drop,2},\, t_{drop,3},\, t_{delay,1},\, t_{delay,2},\, t_{delay,3}]^T \in \mathbb{R}^8 \tag{40}$$

表 3: 问题三决策变量及约束

变量类型	具体变量	约束范围
全局飞行参数	v_{FY_1}	$[70, 140]\,\mathrm{m/s}$
	α	$[0, 2\pi)\,\mathrm{rad}$
投放时刻	$t_{drop,1}$	$[0, 12]\,\mathrm{s}$
	$t_{drop,2}$	$[t_{drop,1}+1, 12]\,\mathrm{s}$
	$t_{drop,3}$	$[t_{drop,2}+1, 12]\,\mathrm{s}$
起爆延时	$t_{delay,1}$	$[1, 8]\,\mathrm{s}$
	$t_{delay,2}$	$[1, 8]\,\mathrm{s}$
	$t_{delay,3}$	$[1, 8]\,\mathrm{s}$

6.3.2 多弹运动状态方程

基于问题一建立的单弹运动模型，扩展到多弹场景。第 k 枚烟幕弹的运动状态由下式描述:

1. 投放点坐标:

$$P_{drop,k}(\mathbf{X}) = B_0 + \boldsymbol{v}_{FY_1}(v_{FY_1}, \alpha) \cdot t_{drop,k} \tag{41}$$

2. 起爆点坐标

$$P_{burst,k}(\mathbf{X}) = P_{drop,k}(\mathbf{X}) + \boldsymbol{v}_{FY_1}(v_{FY_1}, \alpha) \cdot t_{delay,k} + \left[0, 0, -\frac{1}{2} g t_{delay,k}^2\right]^T \tag{42}$$

3. 烟幕云团中心轨迹：起爆时刻 $t_{burst,k} = t_{drop,k} + t_{delay,k}$ 后，第 k 个云团中心位置为

$$P_{S,k}(t) = P_{burst,k} - [0, 0, v_{cloud} \cdot (t - t_{burst,k})]^T \quad , \quad t \in [t_{burst,k}, t_{burst,k} + 20] \tag{43}$$

该时间窗口 $[t_{burst,k}, t_{burst,k} + 20]$ 表示第 k 枚烟幕弹的有效作用时间为 20 秒。

6.3.3 目标函数

定义第 k 枚烟幕弹在时刻 t 的遮蔽状态为:

$$I_k(t; \mathbf{X}) = \begin{cases} 1, & D_k(t) \le R_{cloud} \text{ 且} t \in [t_{burst,k}, t_{burst,k} + 20] \\ 0, & \text{otherwise} \end{cases} \tag{44}$$

其中，$D_k(t)$ 为第 k 个烟幕云团中心到” 导弹 M_1-目标 P_T” 视线的最短距离。**系统总遮蔽状态（逻辑 OR）**由于三个烟幕云团可能同时存在，系统在时刻 t 的遮蔽状态定义为**逻辑或**:

$$I_{total}(t; \mathbf{X}) = \bigvee_{k=1}^{3} I_k(t; \mathbf{X}) \tag{45}$$

其中 $\bigvee_{k=1}^{3}$ 为逻辑或运算符。该定义的物理意义为：只要**至少有一枚**烟幕弹正在有效遮蔽，导弹即被视为处于遮蔽状态（$I_{total}=1$）。最终优化目标为最大化总有效遮蔽时长 T_{cover} 定义为指示函数的时间积分：

$$\max\ J(\mathbf{X})=T_{cover}(\mathbf{X})=\int_{0}^{T_{end}} I_{total}(t;\mathbf{X})\,dt \tag{46}$$

该积分实际上计算的是遮蔽时间段的**并集长度**。例如，若烟幕弹 1 在 $[5,8]$ 秒生效、烟幕弹 2 在 $[7,10]$ 秒生效，则并集为 $[5,10]$，总时长为 5 秒，而非 $3+3=6$ 秒。：

6.3.4 约束条件

基本边界约束：

$$\begin{cases} 70\le v_{FY_1}\le 140 & \text{（速度范围）}\\ 0\le\alpha<2\pi & \text{（航向角）}\\ 0<t_{drop,1}<t_{drop,2}<t_{drop,3}\le 12 & \text{（投放时刻有序性）}\\ 1\le t_{delay,k}\le 8,\quad k=1,2,3 & \text{（起爆延时范围）}\end{cases} \tag{47}$$

高度约束：每枚烟幕弹必须在空中起爆

$$z_{burst,k}(\mathbf{X})>0,\quad k=1,2,3 \tag{48}$$

除了满足飞行与高度约束外，本问必须严格满足多枚弹之间的操作间隔约束，题目要求每架无人机投放两枚烟幕弹至少间隔 1s, 为了防止连续投放导致的机械干涉，确保无人机有足够的时间调整姿态，避免烟幕弹在空中发生碰撞。

$$t_{drop,k+1}-t_{drop,k}\ge 1,\quad k=1,2 \tag{49}$$

6.3.5 模型汇总

问题三的多弹协同优化模型可归纳为：

$$
\begin{cases}
\text{决策变量:} & \mathbf{X} = [v_{FY_1}, \alpha, t_{drop,1}, t_{drop,2}, t_{drop,3}, t_{delay,1}, t_{delay,2}, t_{delay,3}]^T \\
\text{目标函数:} & \max\ T_{cover}(\mathbf{X}) = \displaystyle\int_0^{T_{end}} I_{total}(t;\mathbf{X})dt \\
\text{状态方程:} & \begin{cases}
P_{drop,k} = P_{FY_1}(0) + \boldsymbol{v}_{FY_1} \cdot t_{drop,k} \\
P_{burst,k} = P_{drop,k} + \boldsymbol{v}_{FY_1} \cdot t_{delay,k} + [0,0,-\frac{1}{2}gt_{delay,k}^2]^T \\
P_{S,k}(t) = P_{burst,k} - [0,0,v_{cloud}(t - t_{burst,k})]^T, \quad k = 1,2,3
\end{cases} \\
\text{遮蔽并集:} & I_{total}(t;\mathbf{X}) = \displaystyle\bigvee_{k=1}^{3} I_k(t;\mathbf{X}) \\
\text{约束条件:} & \begin{cases}
70 \le v_{FY_1} \le 140 \\
0 < t_{drop,1} < t_{drop,2} < t_{drop,3} \le 12 \\
t_{drop,k+1} - t_{drop,k} \ge 1, \quad k = 1,2 \\
1 \le t_{delay,k} \le 8, \quad k = 1,2,3 \\
z_{burst,k}(\mathbf{X}) > 0, \quad k = 1,2,3
\end{cases}
\end{cases}
\tag{50}
$$

6.3.6 求解算法

针对决策变量维数增加（由 4 维增至 8 维）且包含时序逻辑约束的问题，我们沿用问题二中的网格扫描 + 粒子群算法（PSO）双层优化架构进行求解。

外层：网格扫描速度 $v_{FY_1} \in [70, 140]$，步长 0.5 m/s

内层：改进粒子群算法（PSO with Catastrophe）优化剩余 7 个变量

内层算法改进 由于 8 维问题远比 4 维问题复杂，传统 PSO 易陷入局部最优或收敛停滞。本文引入**灾变机制**(Catastrophe Mechanism）增强全局探索能力

停滞检测：若全局最优解连续 15 代未改进，判定为停滞

$$
\text{stagnation\_counter} \ge 15 \Rightarrow \text{触发灾变} \tag{51}
$$

灾变操作：

1. 保留前 50% 的精英粒子（适应度最优）
2. 将剩余 50% 粒子随机重新初始化于搜索空间
3. 重置历史最优 p_{best}，但保留全局最优 g_{best}

该机制类似重启搜索，有效避免早熟收敛。为自动满足投弹间隔约束 $t_{drop,k+1}-t_{drop,k}\geq 1$，采用**增量编码**而非直接编码：

粒子编码：

$$X_{particle}=[\alpha,\, t_{drop,1},\, \Delta t_1,\, \Delta t_2,\, t_{delay,1},\, t_{delay,2},\, t_{delay,3}]^T \tag{52}$$

其中，$\Delta t_1, \Delta t_2 \geq 0$ 为时间增量。

解码方式：

$$t_{drop,2}=t_{drop,1}+1.0+\Delta t_1 \quad ,\Delta t_1\geq 0 \tag{53}$$

$$t_{drop,3}=t_{drop,2}+1.0+\Delta t_2=(t_{drop,1}+2.0+\Delta t_1+\Delta t_2) \quad ,\Delta t_2\geq 0 \tag{54}$$

该编码方式本质上嵌入了约束，任何粒子自动满足间隔 $\geq 1\,\mathrm{s}$。

适应度函数

$$F(\mathbf{X})=-T_{cover}(\mathbf{X})+\lambda\sum_{k=1}^{3}\max(0,-z_{burst,k})^2 \tag{55}$$

第一项取负值转为最小化，第二项为高度约束的二次惩罚（$\lambda=10^6$）。

算法参数

外层扫描：140 个速度点

粒子群规模：$N=50$

最大迭代次数：$G_{max}=300$

惯性权重：$w=0.8$

学习因子：$c_1=c_2=1.5$

时间步长：$\Delta t=0.05\,\mathrm{s}$

6.3.7 求解结果

根据上述的流程，我们采用引入了灾变机制的改进粒子群算法得到无人机 FY1 航向方向角为 $0.1543rad(8.84°)$, 速度大小为 $101.88m/s$ 时, 对应的最大有效遮挡时间是 6.45s。投放 3 枚烟幕弹干扰导弹 M1 的最优策略见下图9，三枚烟幕弹的具体投放策略见下表4，分析该结果，我们发现烟幕弹 2 在 $[1.0, 4.16]$ 秒生效，烟幕弹 1 在 $[4.77, 6.93]$ 秒接力，实现了部分时间覆盖的延续，但是烟幕弹 3 未能发挥作用。

表 4: Problem 3 策略表

烟幕弹序号	无人机航向 (rad)	无人机速度 (m/s)	投放坐标 (m)			起爆坐标 (m)		
			X	Y	Z	X	Y	Z
1	0.1543 (8.84°)	101.88	17800	0	1800	17800	0	1800
2	0.1543 (8.84°)	101.88	17900.7	15.7	1800	17900.7	15.7	1800
3	0.1543 (8.84°)	101.88	18792.6	154.4	1800	20302.6	389.2	697.5

烟幕弹序号	有效干扰时长 (s)	生效时间 (s)	失效时间 (s)
1	2.63	4.77	7.4
2	4.16	1.00000	5.16
3	0.00000	0.00000	0.00000

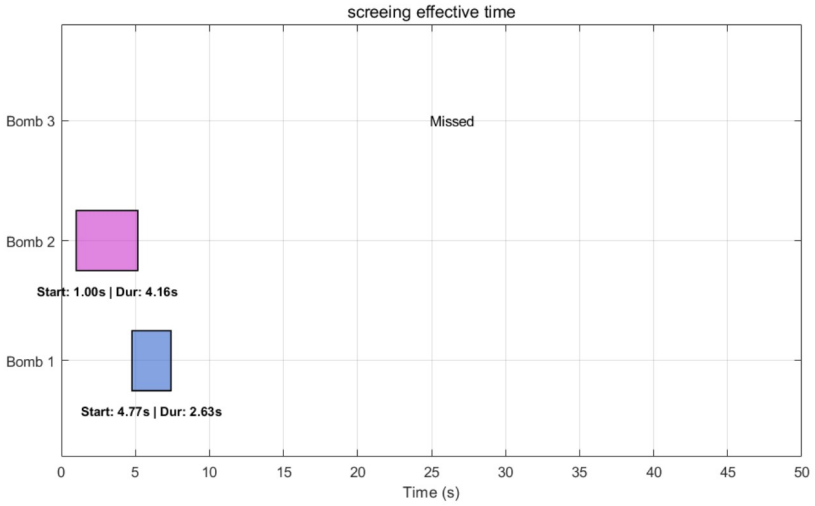


图 9: Problem 3 结果

6.4 问题四求解：多机协同优化

在问题一至问题三中，我们研究的是单架无人机通过调整自身航迹和投弹策略对单枚来袭导弹进行遮蔽拦截的问题。在问题四中，任务变更为 FY_1 FY_2 FY_3 三架无人机，每架投放一枚烟幕弹共同拦截导弹 $M1$，由于三架无人机的初始位置各不相同，分别位于战场的不同区域。这种空间布局导致各无人机到导弹初始位置、真目标位置、假目标位置的距离和方位角均不同，需要针对性地设计各自的飞行策略。为了最大化总遮蔽时间，理想状态是三枚烟幕弹在时间轴上形成**无缝接力**，即第一枚烟幕弹失效时刻恰好衔接第二枚的生效时刻。这要求对各无人机的投放时间 $t_{drop,i}$ 和起爆延时 $t_{delay,i}$ 进行精细化协调，使得烟幕云团的空间位置和时间窗口能够实现完美配合。同时，相比问题二的 4 维决策空间和问题三的 8 维决策空间，问题四的决策维数上升至 12 维，搜索空间呈

指数级增长并且各维度之间相互作用，相互影响。因此，核心问题在于协调三架处于不同空间位置的无人机的投弹策略。

6.4.1 决策变量

根据题目要求，无人机一旦受领任务，其速度和航向就确定了且不再调整。因此在进行任务决策时我们需要决策 12 维变量：系统包含 3 个独立的运动实体（无人机）。定义 $i \in 1,2,3$ 分别对应 FY_1, FY_2, FY_3。每个实体的决策向量 $\mathbf{u}_i$ 包含 4 个参数：

$$\mathbf{u}_i = [v_{FY_i}, \alpha_i, t_{drop,i}, t_{delay,i}] \tag{56}$$

各参数的物理意义如下表所示：

表 5: 问题四决策变量说明

参数	含义	说明
v_{FY_i}	第 i 架无人机飞行速度	影响无人机的机动能力和到达投放点的时间，取值范围 $[70, 140]$ m/s
α_i	第 i 架无人机航向角	决定无人机的飞行方向，直接影响投放点的空间位置，取值范围 $[0, 2\pi]$ rad
$t_{drop,i}$	第 i 架无人机投放烟幕弹时刻	决定烟幕弹的释放时机，影响整体时序协同，不同位置的无人机需要不同的投放时间窗口
$t_{delay,i}$	第 i 枚烟幕弹起爆延时	控制烟幕云团的生成高度和空间位置，取值范围 $[1, 8]$ s

全局决策向量 X 包含 12 维向量：

$$\mathbf{X} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] = [v_1, \alpha_1, t_{drop,1}, t_{delay,1}, \ldots, v_3, \alpha_3, t_{drop,3}, t_{delay,3}]^T \tag{57}$$

6.4.2 状态方程

各无人机的初始位置 $\mathbf{Pos}_{UAV,i}$ 不同，第 i 架无人机以速度 v_{FY_i}、航向 α_i 飞行 $t_{drop,i}$ 秒后到达投放点 $P_{drop,i}$

$$P_{drop,i} = \mathbf{Pos}_{UAV,i} + \boldsymbol{v}_{FY_i} \cdot t_{drop,i} \tag{58}$$

烟幕弹从投放点开始，随无人机惯性飞行 $t_{delay,i}$ 秒后起爆，同时受重力作用下降，则起爆处的位置 $P_{burst,i}$ 为：

$$P_{burst,i} = P_{drop,i} + \boldsymbol{v}_{FY_i} \cdot t_{delay,i} + [0, 0, -\frac{1}{2} g t_{delay,i}^2]^T \tag{59}$$

起爆后，烟幕云团以下沉速度 $v_{cloud} = 3$ m/s 垂直下降，其中心轨迹 $P_{S,i}(t)$ 表示为

$$P_{S,i}(t) = P_{burst,i} - [0, 0, v_{cloud} \cdot (t - t_{burst,i})]^T \tag{60}$$

6.4.3 目标函数

由于 3 个烟幕团可能同时存在并产生遮挡效果，总遮蔽时间不能简单求和，而应计算时间轴上的并集。定义第 i 架飞机第 k 个烟幕云团对导弹 j 的遮蔽指示函数 $I_{i,k,j}(t)$。则该题中系统总遮蔽状态 $S_1(t)$ 为各子状态的逻辑或：

$$S_1(t) = \bigvee_{i=1}^{3} I_{i,1,1}(t) \tag{61}$$

最终优化目标函数定义为最小化代价 $J(\mathbf{X})$：

$$minJ(\mathbf{X}) = -\left(\int_0^{T_{end}} S_1(t)dt\right) + \lambda \cdot \sum_{i=1}^{3} \max(0, d_{min,i} - R_{cloud}) \tag{62}$$

其中：第一项为总遮蔽时长（取负求最小），第二项为惩罚项，$d_{min,i}$ 为第 i 枚烟幕弹在其生命周期内距离视线的最近距离，$\lambda = 0.1$ 为权重系数。当烟幕完全脱靶时，该项通过惩罚距离迫使解向视线靠拢。

6.4.4 约束条件

速度约束：该约束由无人机的物理性能决定, 由题目给出

$$70 \le v_{FY_i} \le 140, \quad \forall i \tag{63}$$

时间窗约束：考虑到导弹 M_1 以 300 m/s 的速度从 $(20000, 0, 2000)$ 飞向假目标 $(0, 0, 0)$，全程飞行时间约为 66.7 s。在此期间，第一枚烟幕弹应在导弹飞行前期（$0 \sim 15$ s）生效，第二枚在中期（$15 \sim 35$ s）接力，第三枚在后期（$35 \sim 55$ s）继续遮蔽。

$$t_{drop,1} \le 15, \quad t_{drop,2} \le 35, \quad t_{drop,3} \le 55 \tag{64}$$

起爆延时约束：该约束确保烟幕弹有足够的惯性飞行距离（至少 $70 \times 1 = 70$ m），同时避免起爆高度过低（延时过长会导致烟幕弹撞地）。

$$1 \le t_{delay,i} \le 8 \tag{65}$$

高度约束：保证所有烟幕弹的起爆点高度非负，避免地下起爆的物理不合理解。

$$z_{burst,i}(\mathbf{X}) \ge 0, \quad i = 1, 2, 3 \tag{66}$$

6.4.5 模型汇总

问题四的多机协同优化模型可系统表示为：

$$
\begin{cases}
\text{决策变量:} & \mathbf{X} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3]^T, \quad \mathbf{u}_i = [v_{FY_i}, \alpha_i, t_{drop,i}, t_{delay,i}] \\
\text{目标函数:} & \min \ J(\mathbf{X}) = -\displaystyle\int_0^{T_{end}} S_1(t)dt + \lambda \sum_{i=1}^{3} \max(d_{min,i} - R_{cloud}, 0) \\
\text{状态方程:} & \begin{cases}
\boldsymbol{v}_{FY_i} = [v_{FY_i}\cos\alpha_i, v_{FY_i}\sin\alpha_i, 0]^T \\
P_{drop,i} = \boldsymbol{Pos}_{UAV,i} + \boldsymbol{v}_{FY_i} \cdot t_{drop,i} \\
P_{burst,i} = P_{drop,i} + \boldsymbol{v}_{FY_i} \cdot t_{delay,i} + [0, 0, -\frac{1}{2}gt_{delay,i}^2]^T \\
P_{S,i}(t) = P_{burst,i} - [0, 0, v_{cloud}(t - t_{burst,i})]^T, \quad i = 1, 2, 3
\end{cases} \\
\text{遮蔽并集:} & S_1(t) = \displaystyle\bigvee_{i=1}^{3} I_{i,1,1}(t) \\
\text{约束条件:} & \begin{cases}
70 \le v_{FY_i} \le 140, \quad i = 1, 2, 3 \\
t_{drop,1} \le 15, \quad t_{drop,2} \le 35, \quad t_{drop,3} \le 55 \\
1 \le t_{delay,i} \le 8, \quad i = 1, 2, 3 \\
z_{burst,i}(\mathbf{X}) \ge 0, \quad i = 1, 2, 3
\end{cases}
\end{cases}
\tag{67}
$$

6.4.6 求解算法

由于决策变量维数增加为 12 维，而且各维度之间差异较大，同时差分进化算法 (DE) 对目标函数的连续性、可微性无要求，能够处理非凸、多峰、高维的复杂优化问题。即使存在局部最优，差分变异机制也能保持种群多样性，跳出局部陷阱。DE 算法基于差分向量进行变异，自适应地调整搜索步长，对变量尺度差异不敏感。无需对决策变量进行归一化预处理。DE 算法通过种群并行评估多个候选解，能够同时探索决策空间的多个区域，适合处理多无人机协同的组合优化问题。相比遗传算法（GA），DE 算法的变异策略更加高效，通常能在更少的迭代次数内找到高质量解。

算法流程设计

- **种群初始化**：

  生成 $NP = 200$ 个个体组成初始种群。针对航向角 α_i，基于各无人机到假目标的初始视线角 $\theta_{base,i}$ 进行**正态分布初始化**，范围控制在 $[\theta_{base,i} - 45°, \theta_{base,i} + 45°]$ 内，提高搜索效率。

  初始化策略说明：

  – 对于第 i 架无人机，计算其初始位置 $\mathbf{Pos}_{UAV,i}$ 到假目标 $(0, 0, 0)$ 的方位角：

$$
\theta_{base,i} = \arctan 2(0 - y_{UAV,i}, 0 - x_{UAV,i}) \tag{68}
$$

  - 在 $\theta_{base,i}$ 附近采样航向角 α_i，使得无人机初始航向大致指向假目标方向，缩小搜索范围。
  - 其他变量（速度、投放时间、延时）在约束范围内均匀随机采样，保证初始种群的多样性。

- **变异**：

  采用 **DE/rand/1** 策略生成变异向量 $\mathbf{V}_g$：

$$\mathbf{V}_{i,g} = \mathbf{X}_{r_1,g} + F \cdot (\mathbf{X}_{r_2,g} - \mathbf{X}_{r_3,g}) \tag{69}$$

  其中 r_1, r_2, r_3 为从当前种群中随机选择的三个互不相同的个体索引（且均不等于 i），F 为缩放因子。

  自适应缩放因子：为了平衡全局探索与局部开发，缩放因子 F 采用**线性衰减策略**

$$F(g) = F_{max} - \frac{(F_{max} - F_{min}) \cdot g}{G_{max}} \tag{70}$$

  其中 $F_{max} = 0.5$，$F_{min} = 0$，g 为当前迭代次数，$G_{max} = 800$ 为最大迭代次数。初期较大的 F 值有助于全局探索，后期较小的 F 值有助于局部精细搜索。

- **交叉**：

  对变异向量 $\mathbf{V}_{i,g}$ 与目标向量 $\mathbf{X}_{i,g}$ 进行**二项式交叉**，生成试验向量 $\mathbf{U}_{i,g}$：

$$U_{i,g}^{(j)} = \begin{cases} V_{i,g}^{(j)}, & \text{若 } \mathrm{rand}(0,1) \le CR \text{ 或 } j = j_{rand} \\ X_{i,g}^{(j)}, & \text{否则} \end{cases} \tag{71}$$

  其中 $j \in \{1, 2, \ldots, 12\}$ 为维度索引，$CR = 0.9$ 为交叉概率，j_{rand} 为随机选择的一个维度（确保至少有一个维度来自变异向量）。

  高交叉率的选择：$CR = 0.9$ 意味着约 90% 的维度来自变异向量，能够充分利用种群的差分信息，加速收敛。

- **选择**：

  计算试验向量与目标向量的适应度，采用**贪婪策略**选择适应度更优的个体进入下一代：

$$\mathbf{X}_{i,g+1} = \begin{cases} \mathbf{U}_{i,g}, & \text{若 } J(\mathbf{U}_{i,g}) < J(\mathbf{X}_{i,g}) \\ \mathbf{X}_{i,g}, & \text{否则} \end{cases} \tag{72}$$

  这种选择机制确保种群的平均适应度单调递减（最小化问题），不会出现退化现象。

- **终止条件**：

  达到最大迭代次数 $G_{max} = 800$

6.4.7 求解结果

我们通过差分进化算法（DE）用 matlab 求解，对应的三机协同投放策略如下表6所示，效果如下图10所示，得到了最大的总遮蔽有效时间 14.9600s，同时 FY_1 和 FY_2 的遮蔽时间段存在约 6.45 s 的间隙（$12.10 - 9.42 = 2.68$ s），FY_2 和 FY_3 之间存在约 5.28 s 的间隙。这表明三机协同并未实现完美的无缝接力，存在两个遮蔽空窗期。但是从全局来看总的遮蔽时间能够达到较大，能够提供较好的遮蔽效果。通过观察投放点和起爆点的坐标，可以看出 FY_1 的烟幕弹在导弹飞行前期形成遮蔽，此时导弹距离假目标较远，视线较长。FY_2 的烟幕弹在导弹飞行中期接力。FY_3 的烟幕弹在导弹飞行后期继续遮蔽，此时导弹接近假目标，视线较短。这种前、中、后三段分布的策略充分利用了三架无人机的空间位置优势，形成” 接力式立体封锁”。

表 6: 问题 4 三机协同立体封锁最终策略

			投放点 (m)	起爆点 (m)
无人机	航向 (°)	T_{cover}	(X,Y,Z)	(X,Y,Z)
FY1	178.42	4.65	(17747.32, 1.45, 1800)	(17521.68, 7.68, 1760)
FY2	-129.29	4.82	(11492.56, 779.81, 1400)	(10897.90, 53.01, 1099)
FY3	106.94	5.46	(5291.39, −673.54, 700)	(5051.65, 113.58, 518)

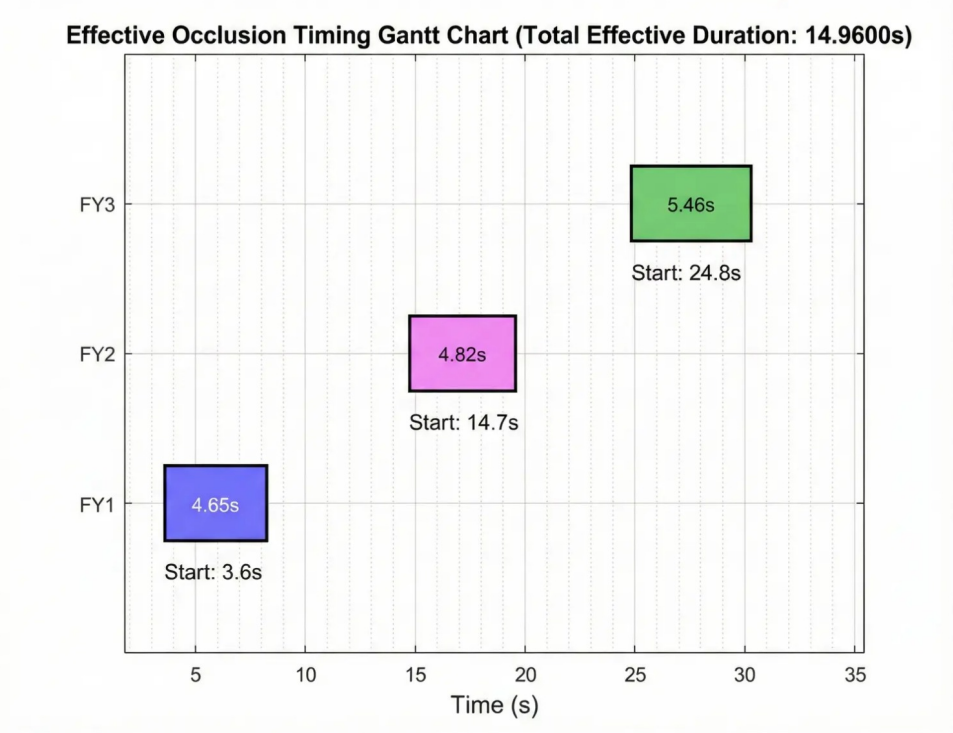


图 10: 问题 4 三机协同立体封锁结果

6.5 问题五求解：多机多目标协同拦截

问题五将作战场景扩展至**多对多协同拦截**：5 架无人机 $FY_1 - FY_5$ 协同拦截 3 枚来袭导弹 $M_1 - M_3$，每架无人机最多可携带 3 枚烟幕弹。相比问题四，问题五的复杂度呈现**指数级增长**：

1. **决策向量过多**：

   若采用问题四的建模方式，需要优化的决策变量维数为：

   $$n_{vars} = 5 \times (2 + 3 \times 2) = 5 \times 8 = 40\text{维} \tag{73}$$

   其中每架无人机需要决策：速度 v_{FY_i}、航向 α_i，以及 3 枚烟幕弹的投放时间 $t_{drop}^{(i,k)}$ 和起爆延时 $t_{delay}^{(i,k)}$ $(k = 1, 2, 3)$。在 40 维连续空间中进行全局优化，即使采用高效的进化算法，单次适应度评估也需要模拟 5 架无人机、15 枚潜在烟幕弹、3 枚导弹在 60 秒内的运动轨迹，计算复杂度约为 $O(5 \times 15 \times 3 \times 1200) = O(2.7 \times 10^5)$ 次基本运算。若种群规模 200、迭代 1000 次，总计算量将达到 $O(5.4 \times 10^{10})$

2. **多目标耦合**：与问题四的单导弹拦截不同，问题五需要同时遮蔽 3 枚导弹。这引入了资源分配问题：5 架无人机的 15 枚烟幕弹应如何在 3 枚导弹之间分配，某架无人机应优先拦截哪枚导弹？

3. **时空协同复杂性**：3 枚导弹的初始位置、飞行方向各不相同，5 架无人机分布在战场的不同区域。某个空间位置的无人机可能对导弹 M_1 有利，但对 M_2、M_3 不利。因此需要在**空间**、**时间**、**目标分配**三个维度进行联合优化。

为应对上述复杂度挑战，本文采用**分层解耦策略**——将 40 维高耦合优化问题分解为：

- **外层（战略层）**：优化 5 架无人机的飞行参数（速度、航向），决策维数降至 10 维。
- **内层（战术层）**：在给定飞行轨迹下，通过启发式算法自动生成每架无人机的最优投弹方案（包括目标分配、投放时间、起爆延时）。

这种分层架构将航迹规划与时序调度解耦，显著降低搜索空间维度，同时保持解的质量。

战略决策变量（外层） 定义全局飞行决策向量 X_{fly}，包含 5 架无人机的速度和航向：

$$X_{fly} = [v_1, \alpha_1, v_2, \alpha_2, \ldots, v_5, \alpha_5]^T \in \mathbb{R}^{10} \tag{74}$$

各变量的物理意义如下：

表 7: 问题五战略决策变量

参数	含义	说明
v_{FY_i}	第 i 架无人机速度	决定无人机的机动范围和到达关键位置的时间，取值 $[70, 140]$ m/s
α_i	第 i 架无人机航向	决定无人机的飞行方向，直接影响其能够覆盖的空间区域和可拦截的导弹集合，取值 $[0, 2\pi]$ rad

这样优化飞行参数（10 维），而非直接优化所有投弹参数（40 维），大幅降低了搜索空间的复杂度。

战术决策变量（内层生成） 对于第 i 架无人机，其携带的第 k 枚烟幕弹（$k = 1, 2, 3$）的投放参数**不作为外层优化变量**，而是在内层通过启发式算法自动生成：

$$u_{i,k} = \left[t_{drop}^{(i,k)}, t_{delay}^{(i,k)}, m_{target}^{(i,k)} \right] \tag{75}$$

其中 $m_{target}^{(i,k)} \in \{1, 2, 3\}$ 为该烟幕弹拦截的目标导弹编号。给定外层确定的飞行轨迹 X_{fly}，内层算法遍历所有可能的投弹时间和起爆延时组合，评估每种组合对 3 枚导弹的遮蔽效果，通过贪心策略选择总遮蔽时间最大的 3 次投弹（详见后续" 求解算法" 部分）。

6.5.1 状态空间

多导弹运动模型 3 枚导弹 M_1, M_2, M_3 的初始位置和飞行特性由题目给定：

$$\begin{aligned}
M_1 &: \mathbf{P}_{M_1}(0) = (20000, 0, 2000), \quad \mathbf{n}_{M_1} = \frac{(0,0,0) - \mathbf{P}_{M_1}(0)}{\|\cdots\|} \\
M_2 &: \mathbf{P}_{M_2}(0) = (19000, 600, 2100), \quad \mathbf{n}_{M_2} = \frac{(0,0,0) - \mathbf{P}_{M_2}(0)}{\|\cdots\|} \\
M_3 &: \mathbf{P}_{M_3}(0) = (18000, -600, 1900), \quad \mathbf{n}_{M_3} = \frac{(0,0,0) - \mathbf{P}_{M_3}(0)}{\|\cdots\|}
\end{aligned} \tag{76}$$

所有导弹均以速度 $V_m = 300$ m/s 匀速直线飞向假目标 $(0, 0, 0)$，运动方程为：

$$\mathbf{P}_{M_j}(t) = \mathbf{P}_{M_j}(0) + V_m \mathbf{n}_{M_j} t, \quad j = 1, 2, 3 \tag{77}$$

多无人机运动模型 5 架无人机的初始位置由题目给定，运动方程与问题四类似：

$$\mathbf{P}_{FY_i}(t) = \mathbf{P}_{FY_i}(0) + \boldsymbol{v}_{FY_i} t, \quad i = 1, \ldots, 5 \tag{78}$$

其中 $\boldsymbol{v}_{FY_i} = [v_{FY_i} \cos\alpha_i, v_{FY_i} \sin\alpha_i, 0]^T$ 为第 i 架无人机的速度向量。

烟幕弹运动模型 第 i 架无人机投放的第 k 枚烟幕弹的运动过程为：

$$\begin{aligned}
&\text{投放点：} \quad P_{drop}^{(i,k)} = \mathbf{P}_{FY_i}(t_{drop}^{(i,k)}) \\
&\text{起爆点：} \quad P_{burst}^{(i,k)} = P_{drop}^{(i,k)} + \mathbf{v}_i t_{delay}^{(i,k)} + [0,0,-\tfrac{1}{2}g(t_{delay}^{(i,k)})^2]^T \\
&\text{云团轨迹：} \quad P_S^{(i,k)}(t) = P_{burst}^{(i,k)} - [0,0,v_{cloud}(t - t_{burst}^{(i,k)})]^T
\end{aligned} \tag{79}$$

其中 $t_{burst}^{(i,k)} = t_{drop}^{(i,k)} + t_{delay}^{(i,k)}$ 为起爆时刻，$v_{cloud} = 3$ m/s 为下沉速度。
定义第 i 架无人机第 k 枚烟幕弹对第 j 枚导弹的遮蔽指示函数：

$$I_{i,k,j}(t) = \begin{cases} 1, & \text{若 } d(P_S^{(i,k)}(t), L_{M_j}(t)) \le R_{cloud} \text{ 且 } t \in [t_{burst}^{(i,k)}, t_{burst}^{(i,k)} + T_{life}] \\ 0, & \text{否则} \end{cases} \tag{80}$$

其中 $L_{M_j}(t)$ 为导弹 M_j 指向真目标的视线,$d(P_S^{(i,k)}(t), L_{M_j}(t))$ 表示第 i 架无人机第 k 枚烟幕弹当前位置与导弹视线的距离，$R_{cloud} = 10$ m，$T_{life} = 20$ s。

6.5.2 目标函数

由于存在 3 个独立的拦截目标，系统总效益定义为**所有导弹被有效遮蔽时长的总和**。对于第 j 枚导弹（$j = 1,2,3$），其被遮蔽状态 $S_j(t)$ 为所有可能遮蔽源的**逻辑或**：

$$S_j(t) = \bigvee_{i=1}^{5} \bigvee_{k=1}^{3} I_{i,k,j}(t) = \max_{i,k}\{I_{i,k,j}(t)\} \tag{81}$$

第 j 枚导弹的总遮蔽时间 $T_{cover,j}$ 为：

$$T_{cover,j} = \int_0^{T_{end}} S_j(t)\, dt \tag{82}$$

最终优化目标为**最大化总效益**：

$$\max J(X_{fly}) = \sum_{j=1}^{3} T_{cover,j} = \sum_{j=1}^{3} \left(\int_0^{T_{end}} S_j(t)\, dt \right) \tag{83}$$

并集计算说明：

- 对于单枚导弹 M_j，可能被多架无人机、多枚烟幕弹同时遮蔽。例如，在 $t = 10$ s 时刻，若 FY_1 的第 1 枚烟幕弹和 FY_2 的第 2 枚烟幕弹同时遮蔽 M_1，仍然只计为 1 秒的遮蔽时间（而非 2 秒）。
- 因此需要对每枚导弹单独计算时间轴上的遮蔽并集，然后求和。这与问题三、问题四的并集计算原理一致，但扩展至多目标场景。
- 实际计算中，采用离散时间（步长 $\Delta t = 0.05$ s），统计每个时间步长内 $S_j(t) = 1$ 的总数，再乘以 Δt 得到总遮蔽时间。

6.5.3 约束条件

最大载弹量约束：每架无人机投放数量 $N \leq 3$, 那么定义指示函数 $\mathbb{1}_{drop}^{(i,k)}$，表示第 i 架无人机是否投放第 k 枚烟幕弹, 满足

$$\sum_{k=1}^{3} \mathbb{1}_{drop}^{(i,k)} \leq 3, \quad i = 1, \ldots, 5 \tag{84}$$

投弹间隔约束：

$$|t_{drop}^{(i,k)} - {}_{drop}^{(i,k-1)}| \geq 1, \forall k > 1 \tag{85}$$

飞行边界约束：

$$70 \leq v_{FY_i} \leq 140, 0 \leq \alpha_i \leq 2\pi \tag{86}$$

高度约束: 所有烟幕弹的起爆点高度必须非负

$$z_{burst}^{(i,k)} \geq 0, \quad \forall i, k \tag{87}$$

6.5.4 模型汇总

问题五的多机多目标协同拦截优化模型的完整数学表达为：

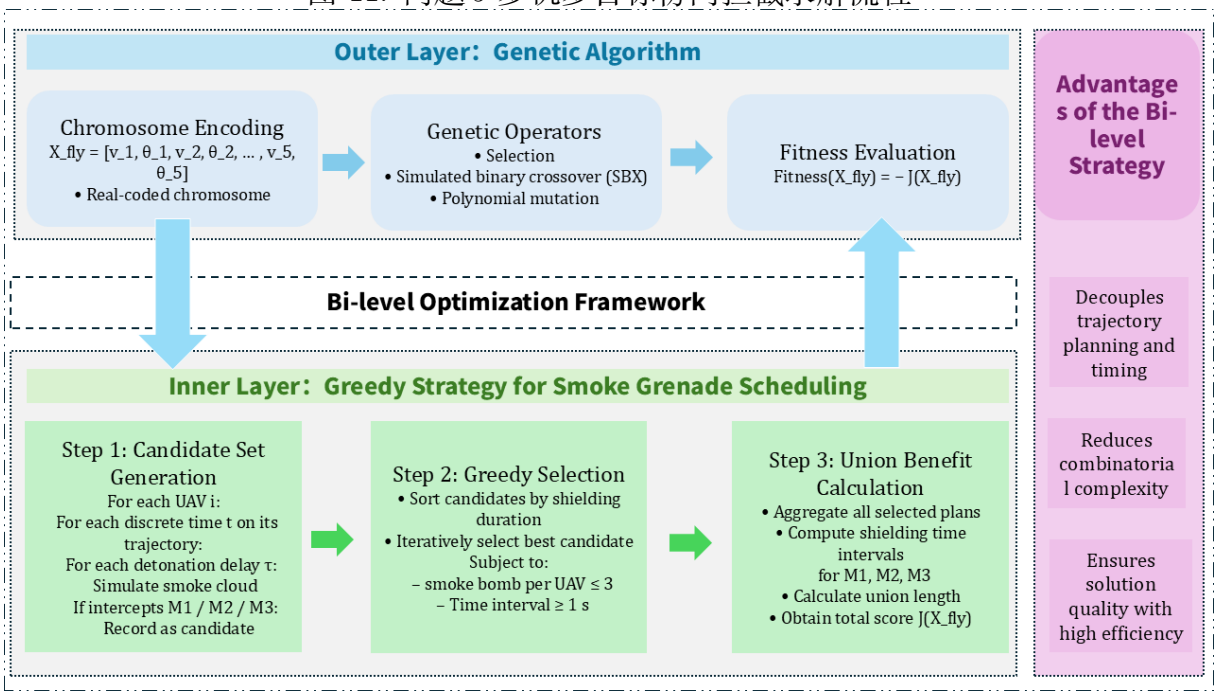
$$\begin{cases} \text{决策变量:} \quad X_{fly} = [v_1, \alpha_1, v_2, \alpha_2, \ldots, v_5, \alpha_5]^T \\ \text{目标函数:} \quad \max \ J(X_{fly}) = \sum_{j=1}^{3} \int_0^{T_{end}} S_j(t) dt \\ \text{导弹运动:} \quad \boldsymbol{P}_{M_j}(t) = \boldsymbol{P}_{M_j}(0) + V_m \boldsymbol{n}_{M_j} t, \quad j = 1, 2, 3 \\ \text{无人机运动:} \quad \begin{cases} \boldsymbol{v}_i = [v_{FY_i} \cos \alpha_i, v_{FY_i} \sin \alpha_i, 0]^T \\ \boldsymbol{P}_{FY_i}(t) = \boldsymbol{P}_{FY_i}(0) + \boldsymbol{v}_i t, \quad i = 1, \ldots, 5 \end{cases} \\ \text{烟幕弹运动:} \quad \begin{cases} P_{drop}^{(i,k)} = \boldsymbol{P}_{FY_i}(t_{drop}^{(i,k)}) \\ P_{burst}^{(i,k)} = P_{drop}^{(i,k)} + \boldsymbol{v}_i t_{delay}^{(i,k)} + [0, 0, -\frac{1}{2} g (t_{delay}^{(i,k)})^2]^T \\ P_S^{(i,k)}(t) = P_{burst}^{(i,k)} - [0, 0, v_{cloud}(t - t_{burst}^{(i,k)})]^T \end{cases} \\ \text{遮蔽状态:} \quad S_j(t) = \bigvee_{i=1}^{5} \bigvee_{k=1}^{3} I_{i,k,j}(t), \quad j = 1, 2, 3 \\ \text{约束条件:} \quad \begin{cases} 70 \leq v_{FY_i} \leq 140, \quad 0 \leq \alpha_i \leq 2\pi, \quad i = 1, \ldots, 5 \\ \sum_{k=1}^{3} \mathbb{1}_{drop}^{(i,k)} \leq 3, \quad i = 1, \ldots, 5 \\ |t_{drop}^{(i,k)} - t_{drop}^{(i,k-1)}| \geq 1, \quad \forall i, k > 1 \\ z_{burst}^{(i,k)} \geq 0, \quad \forall i, k \end{cases} \end{cases} \tag{88}$$

其中，$\mathbb{1}_{drop}^{(i,k)}$ 为指示函数，表示第 i 架无人机是否投放第 k 枚烟幕弹；$\bigvee$ 表示逻辑或运算。

6.5.5 求解算法

针对 5 机 3 弹的高维组合优化问题，依旧采用双层优化结构。外层使用遗传算法搜索无人机的最优飞行参数，内层使用贪心启发式算法快速计算给定航迹下的最优投弹方案。整体流程设计如下图11所示，同时我们所提出的双层框架将航迹规划问题与随时间变化的烟雾弹调度问题分离开来。外层的遗传算法专注于无人机的全局定位，而内层的贪心策略则能高效地确定最佳引爆时间，从而显著降低了计算复杂度，同时又不牺牲解决方案的质量。

图 11: 问题5 多机多目标协同拦截求解流程



- 外层：遗传算法全局寻优将 5 架无人机的速度和航向编码为一条染色体。

$$X_{fly} = [v_1, \alpha_1, v_2, \alpha_2, \ldots, v_5, \alpha_5] \in \mathbb{R}^{10} \tag{89}$$

采用实数编码，操作算子采用迷你二进制交叉和多项式变异，便于利用连续空间的局部信息，模拟二进制交叉（SBX）和多项式变异能够在保持种群多样性的同时加速收敛实现编码 X_{fly}，对于种群中的每个个体 X_{fly}，调用内层算法计算总遮蔽时间 $J(X_{fly})$，适应度定义为：

$$Fitness(X_{fly}) = -J(X_{fly}) \tag{90}$$

取负号是因为遗传算法标准框架为**最小化**问题，而我们的目标是最大化遮蔽时间。遗传算子

- **选择**：采用**锦标赛选择**（Tournament Selection），锦标赛规模为 3。从种群中随机抽取 3 个个体，选择适应度最优的进入交配池。这种选择压力适中，既能保持种群多样性，又能加速优良基因的传播。
- **交叉**：采用**模拟二进制交叉**（Simulated Binary Crossover, SBX），分布指数 $\eta_c = 15$。SBX 算子模拟二进制编码的单点交叉效果，能够在父代附近生成子代，保持搜索的局部性。
- **变异**：采用**多项式变异**（Polynomial Mutation），分布指数 $\eta_m = 20$，变异概率 $p_m = 1/10 = 0.1$（每个变量）。多项式变异在变量当前值附近进行扰动，扰动幅度随分布指数增大而减小，适合精细搜索。

算法参数

- 种群规模：$NP = 150$
- 最大迭代次数：$G_{max} = 80$
- 精英保留：每代保留适应度最优的前 5 个个体直接进入下一代

- 内层：贪心策略解算投弹方案对于种群中每一个体确定的飞行轨迹，内层算法通过以下步骤计算适应度：

Step 1: 生成候选打击集

对于第 i 架无人机，遍历其飞行路径上的离散时间点 $t_{drop} \in [0, 70]$（步长 1 秒），对于每个投放时刻，再遍历可能的起爆延时 $t_{delay} \in [1, 15]$（步长 2 秒）。

对于每个 (t_{drop}, t_{delay}) 组合，计算对应的烟幕弹起爆点位置 P_{burst}，然后检测该烟幕云团能否有效遮蔽 M_1, M_2, M_3 中的任意一枚。具体判定方法：

- 模拟烟幕云团从起爆时刻 t_{burst} 到 $t_{burst} + 20$ s 的下沉轨迹
- 对于每枚导弹 M_j，计算烟幕云团中心到导弹视线 $L_{M_j}(t)$ 的距离 $d(t)$
- 若存在时间区间 $[t_{start}, t_{end}]$ 使得 $d(t) \leq 10$ m，则记录该候选方案：

$$\text{Candidate} = [t_{drop}, t_{delay}, j, t_{start}, t_{end}, \Delta t_{cover}] \tag{91}$$

其中 $\Delta t_{cover} = t_{end} - t_{start}$ 为遮蔽时长。

遍历完成后，第 i 架无人机的候选集 $\mathcal{C}_i$ 包含所有可能的有效打击方案。

Step 2: 贪心选择

对候选方案集按遮蔽时长降序排列。依次选择遮蔽贡献最大的方案加入最终策略，需满足：该无人机已选弹药数 < 3；与该无人机已选方案的时间间隔 $>= 1s$。

Step 3: 计算并集效益

汇总所有 5 架无人机的投弹方案 $\{\mathcal{S}_1, \ldots, \mathcal{S}_5\}$，分别计算对 M_1, M_2, M_3 的遮蔽时间区间并集。

对于第 j 枚导弹，提取所有针对它的遮蔽区间 $\{[t_{start}^{(1)}, t_{end}^{(1)}], [t_{start}^{(2)}, t_{end}^{(2)}], \ldots\}$，计算并集总长度：

1. 按起始时间排序
2. 合并重叠区间（若 $t_{start}^{(i+1)} < t_{end}^{(i)}$，则合并为 $[t_{start}^{(i)}, \max(t_{end}^{(i)}, t_{end}^{(i+1)})]$）
3. 求和所有合并后区间的长度

最终返回总遮蔽时间：

$$J(X_{fly}) = \sum_{j=1}^{3} T_{cover,j} \tag{92}$$

此方法的优势在于将复杂的时序配合问题解耦：外层负责“站位”（规划航线），内层负责“输出”（寻找最佳开火时机），在保证解的质量的同时大幅降低了计算复杂度。

6.5.6 求解结果

通过双层优化算法，用 MATLAB 求解，得到 5 机 3 弹协同拦截 3 枚导弹的最优策略如下表8所示，可以统计出 5 架无人机共投放了 **9 枚烟幕弹**（未达到最大 15 枚），总有效遮蔽时间为 26.6s，效果如图12所示。 分析该结果我们可以发现 M_1 和 M_2 获得了

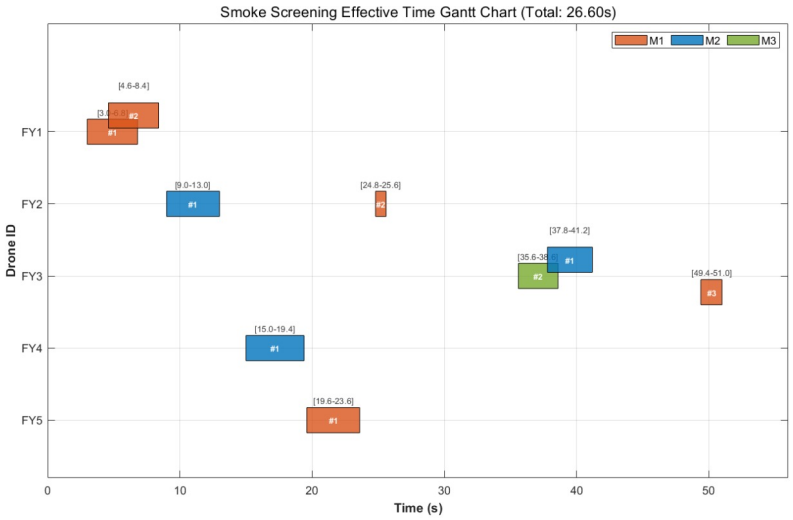


图 12: 五机协同的最优方案

大部分资源（5 枚 +3 枚），而 M_3 仅获得 1 枚。这种**非均衡分配**是优化算法基于总遮蔽时间最大化目标做出的理性选择——将有限资源集中于”性价比”最高的目标，而非追求平均分配。同时对 M_1 的遮蔽时间线进行分析我们发现 [3.0, 8.4] s：FY1 的两枚烟幕弹形成持续遮蔽（有 0.6s 重叠，并集为 5.4s）[19.6,23.6] 并上 [24.8 ,25.6] s：FY5(#1) 和 FY2(#2) 接力遮蔽 [49.4, 51.0] s：FY3(#3) 在导弹飞行末期补充遮蔽（1.6s）可以看

表 8: 五机协同的详细方案

无人机	烟幕弹	投放时刻 (s)	起爆时刻 (s)	目标	遮蔽时长 (s)	有效区间
FY1: 88.19 m/s, 178.35°(3.11 rad)						
FY1	#1	0.00	3.00	M1	3.80	[3.0–6.8]
FY1	#2	1.00	4.00	M1	3.80	[4.6–8.4]
FY2: 110.93 m/s, 283.65°(4.95 rad)						
FY2	#1	6.00	9.00	M2	4.00	[9.0–13.0]
FY2	#2	8.00	13.00	M1	0.80	[24.8–25.6]
FY3: 108.05 m/s, 121.72°(2.12 rad)						
FY3	#1	28.00	35.00	M2	3.40	[37.8–41.2]
FY3	#2	25.00	32.00	M3	3.00	[35.6–38.6]
FY3	#3	26.00	33.00	M1	1.60	[49.4–51.0]
FY4: 135.80 m/s, 262.62°(4.58 rad)						
FY4	#1	1.00	12.00	M2	4.40	[15.0–19.4]
FY5: 134.90 m/s, 123.38°(2.15 rad)						
FY5	#1	13.00	18.00	M1	4.00	[19.6–23.6]

表 9: 问题五目标分配与遮蔽时间统计

导弹	分配烟幕弹数	来源无人机	总遮蔽时间 (s)
M_1	5 枚	FY1(#1,#2), FY2(#2), FY3(#3), FY5(#1)	11.8
M_2	3 枚	FY2(#1), FY3(#1), FY4(#1)	11.8
M_3	1 枚	FY3(#2)	3.0
		总计：	**26.6 s**

出，对 M_1 形成了前期-中期-末期三段式遮蔽，但中间存在较大空窗期（$8.4 \sim 19.6$ s，长达 11.2s）。同时通过对 5 架无人机 60% 的烟幕弹利用率分析我们可以推测并非所有投弹都能产生正收益，某些空间位置、时间窗口下，即使投放烟幕弹也无法有效遮蔽任何导弹（例如烟幕弹下沉至地面时导弹尚未到达），贪心算法在候选集中未找到足够多的高质量方案（遮蔽时长 >0.5s），因此部分无人机提前终止投弹，若强制要求每架无人机投满 3 枚，可能导致总遮蔽时间反而下降（低质量烟幕弹占用了时间窗口，影响其他无人机的协同）。

7 模型评价

本文建立的模型基于问题的物理背景与作战约束，对导弹、无人机及烟幕干扰弹的运动过程进行了合理简化。针对问题中高度耦合的多无人机、多烟幕弹、多时间决策问题，本文提出了一种外层遗传算法 + 内层贪心策略的双层优化框架。外层遗传算法负责无人机飞行速度与航向的全局搜索，避免了传统局部搜索方法易陷入局部最优的问题；内层贪心策略在给定飞行轨迹的条件下，对烟幕弹投放时机进行高效调度，通过候选集生成与约束筛选，快速获得近似最优的投弹方案。

7.1 模型的优点

- 考虑了烟幕弹的物理沉降特性，符合实际战场环境。
- 算法具有较强的通用性，可扩展至更多数量的无人机集群。
- 采用分层优化结构，降低了高维组合优化问题的计算复杂度。
- 将真目标简化为关键视线点，避免了不必要的三维几何计算，使得遮蔽判定在保持精度的前提下具备较高的计算效率

7.2 模型的缺点

尽管本文模型在计算效率与策略效果方面表现良好，但仍存在一定局限性。例如，模型中未考虑风场对无人机、烟幕云团扩散的影响，且假设导弹飞行路径不受干扰保持直线飞行，这在复杂战场环境中可能存在偏差。

7.3 模型的改进方向

- 当前模型假设导弹不机动，未来可增加导弹末端机动的对抗策略。
- 模型中未考虑风场对物体运动影响，可引入风场模型，对运动进行建模。
- 未来可结合强化学习方法，进一步提升模型的适应性与智能化水平。

参考文献

[1] Anthropic. Claude 4.5 sonnet. anthropic-ai, 2024. Accessed: 2025-12-19.

[2] Google. Gemini. google-ai, 2024. Accessed: 2025-12-19.

[3] Ruiyao Luo, Delin Wang, Wei Luo, Wugang Liu, Feng Ding, Guoxin Wan, and Yifeng Shen. Research on deployment strategy of smoke grenades against integrated reconnaissance and strike uavs. *Electro-Optic Technology Application*, 37(06):90–98, 2022.

[4] OpenAI. Chatgpt (gpt-4o). openai, 2024. Accessed: 2025-12-19.

[5] Zhuo Zhu, Qiyan Liu, and Ce Zhang. Autonomous jamming decision method based on genetic algorithm. *Journal of Civil Aviation University of China*, 41(04):58–64, 2023.

附录：

支撑材料

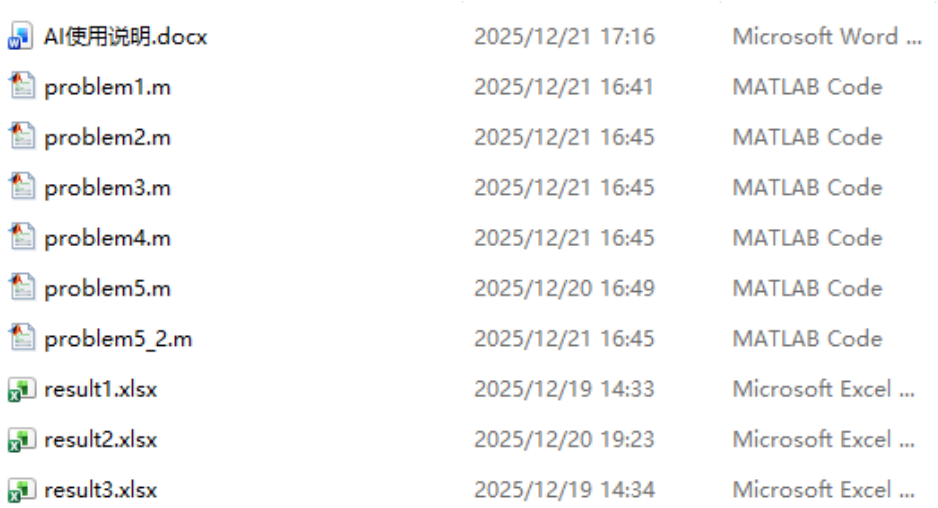


图 13: 支撑材料目录

代码

Listing 1: Problem 1

```
clc; clear; close all;

%% 1. 参数设置
% 烟幕弹参数
R_smoke = 10;            % 有效遮蔽半径（阈值）
Time_smoke_last = 25;    % 延长一点仿真时间以便看全曲线
V_smoke_sink = 3;        % 烟雾团下沉速度

g = 9.8;

% 目标位置
Pos_FakeTarget = [0, 0, 0];           % 假目标
Pos_TrueTarget_Center = [0, 200, 5]; % 真目标

%% 2. 计算烟幕弹起爆初始位置
% FY1 初始状态
Pos_FY1_0 = [17800, 0, 1800];
V_FY1 = 120;

% 时间节点
t_drop = 1.5;            % 投放时刻
t_delay = 3.6;           % 延时时长
t_pop = t_drop + t_delay; % 起爆时刻 (5.1s)

% 1. 投放点位置
Pos_Drop = Pos_FY1_0 + [-1, 0, 0] * V_FY1 * t_drop;
% 2. 起爆点初始位置
Delta_X = -1 * V_FY1 * t_delay;
Delta_Z = -0.5 * g * t_delay^2;
Pos_Smoke_Init = Pos_Drop + [Delta_X, 0, Delta_Z];

fprintf('Smoke grenade detonation time: t = %.2f s\n', t_pop);

%% 3. 计算有效遮蔽时长（并记录数据）
% M1 初始状态
Pos_M1_0 = [20000, 0, 2000];
V_M1 = 300;
```

```
Vec_M1 = Pos_FakeTarget - Pos_M1_0;
Dist_M1_Total = norm(Vec_M1);
Dir_M1 = Vec_M1 / Dist_M1_Total;

% 时间积分设置
dt = 0.05;
valid_time = 0;
t_start_effective = NaN; % 记录开始的时刻
t_end_effective = NaN;   % 记录结束的时刻

fprintf('Simulating the dynamic occlusion process...\n');

Total_Sim_Time = 30;

for t_current = 0 : dt : Total_Sim_Time

    % A. 更新导弹位置
    dist_flown = V_M1 * t_current;
    if dist_flown >= Dist_M1_Total
        Current_M1_Pos = Pos_FakeTarget;
    else
        Current_M1_Pos = Pos_M1_0 + Dir_M1 * dist_flown;
    end

    % B. 更新烟雾位置 & 计算距离
    if t_current >= t_pop
        t_relative = t_current - t_pop;

        if t_relative <= Time_smoke_last
            Sink_Dist = V_smoke_sink * t_relative;
            Current_Smoke_Pos = Pos_Smoke_Init - [0, 0, Sink_Dist];

            % 计算遮挡距离
            P1 = Current_M1_Pos;
            P2 = Pos_TrueTarget_Center;
            Q = Current_Smoke_Pos;

            v = P2 - P1;
            w = Q - P1;
            c1 = dot(w, v);
```

```
            c2 = dot(v, v);

            if c1 <= 0
                dist = norm(Q - P1);
            elseif c2 <= c1
                dist = norm(Q - P2);
            else
                b = c1 / c2;
                Pb = P1 + b * v;
                dist = norm(Q - Pb);
            end

            % --- 捕捉开始和结束时间 ---
            if dist <= R_smoke
                if isnan(t_start_effective)
                    t_start_effective = t_current;
                end
                t_end_effective = t_current;
                valid_time = valid_time + dt;
            end

        else
            dist = NaN;
        end
    else
        dist = norm(Pos_Smoke_Init - Current_M1_Pos);
    end
end
fprintf('>>> Effective shielding duration: %.4f seconds <<<\n',
    valid_time);

%% 4. 绘图
figure('Color', 'w', 'Position', [100, 100, 600, 600]);
hold on;

bar_height = 0.6;
fy_index = 1;
color_fy1 = [100, 100, 255]/255;

if valid_time > 0 && ~isnan(t_start_effective)
```

```
117     x_patch = [t_start_effective, t_end_effective, t_end_effective,
            t_start_effective];
118     y_patch = [fy_index - bar_height/2, fy_index - bar_height/2,
            fy_index + bar_height/2, fy_index + bar_height/2];
119
120     patch(x_patch, y_patch, color_fy1, 'EdgeColor', 'k', 'LineWidth',
             1, 'FaceAlpha', 0.8);
121
122     text_str = sprintf('Start: %.1fs', t_start_effective);
123     text_y_pos = fy_index - bar_height/2 - 0.2;
124     text(t_start_effective, text_y_pos, text_str, ...
125         'FontSize', 10, 'Color', 'k', 'FontWeight', 'normal', '
                HorizontalAlignment', 'left');
126     % ----------------
127 else
128     fprintf('Warning: No effective shielding generated with current
            parameters.\n');
129 end
130
131 ax = gca;
132 ax.YDir = 'normal';
133 ylim([0, 4]);
134 yticks([1, 2, 3]);
135 yticklabels({'FY1', 'FY2', 'FY3'});
136
137 xlim([-10, 50]);
138 xlabel('时间 (s)', 'FontSize', 11);
139
140 grid on;
141 ax.GridAlpha = 0.3;
142 ax.MinorGridAlpha = 0.1;
143 ax.XMinorGrid = 'on';
144
145 title_str = sprintf('协同遮蔽时序 (总长: %.2fs)', valid_time);
146 title(title_str, 'FontSize', 12, 'FontWeight', 'normal');
147
148 box on;
149
150 hold off;
```

Listing 2: Problem 2

```
function Full_Range_PSO_Sweep_GanttOnly()
    clc; close all;
    %% 1. 场景基础参数
    Env.g = 9.8;
    Env.Pos_FakeTarget = [0, 0, 0];
    Env.Pos_TrueTarget = [0, 200, 5];
    Env.Pos_M1_Init = [20000, 0, 2000];
    Env.V_M1 = 300;
    Env.Vec_M = Env.Pos_FakeTarget - Env.Pos_M1_Init;
    Env.Dir_M1 = Env.Vec_M / norm(Env.Vec_M);
    Env.Dist_Total_M1 = norm(Env.Vec_M);
    Env.Pos_FY1_Init = [17800, 0, 1800];
    Env.V_smoke_sink = 3;
    Env.R_smoke = 10;
    Env.Time_smoke_last = 25;

    % === 仿真精度设置 ===
    Env.dt = 0.001;

    %% 2. 扫描设置
    v_scan_list = 70 : 0.5 : 140;
    num_scan = length(v_scan_list);

    % 结果存储
    results_score = zeros(num_scan, 1);
    results_params = zeros(num_scan, 3); % [航向, 投放, 延时]

    fprintf('=========================================================\n
        ');
    fprintf('␣␣␣␣␣␣␣正在计算最优解...\n');
    fprintf('=========================================================\n
        ');

    try
        if isempty(gcp('nocreate')), parpool; end
    catch
    end

    %% 3. 并行扫描循环
```

```
    parfor i = 1 : num_scan
        v_curr = v_scan_list(i);

        % --- PSO 配置 ---
        Vec_Base = Env.Pos_TrueTarget - Env.Pos_FY1_Init;
        Base_Angle = rad2deg(atan2(Vec_Base(2), Vec_Base(1)));

        LB = [Base_Angle-30, 0, 1.0];
        UB = [Base_Angle+30, 12, 8.0];

        [best_x, best_val] = Run_Micro_PSO(v_curr, LB, UB, Env);

        results_score(i) = best_val;
        results_params(i, :) = best_x;
    end

    %% 4. 提取最优结果
    [max_score, idx_best] = max(results_score);
    best_v = v_scan_list(idx_best);
    best_p = results_params(idx_best, :);

    fprintf('>>> 计算完成\n');
    fprintf('>>> 冠军速度: %.1f m/s\n', best_v);
    fprintf('>>> 极限遮蔽: %.5f 秒\n', max_score);

    %% 5. 绘图
    fprintf('\n>>> 正在生成时序甘特图...\n');

    [t_start_opt, t_end_opt, duration_opt] = Recompute_Time_Series(
       best_v, best_p, Env);

    if duration_opt > 0
        figure('Color', 'w', 'Position', [100, 100, 600, 600], 'Name'
           , 'Simulated Gantt Chart');
        hold on;

        bar_height = 0.6;
        fy_index = 1;
        color_fill = [100, 100, 255]/255;

```

```
76          x_patch = [t_start_opt, t_end_opt, t_end_opt, t_start_opt];
77          y_patch = [fy_index - bar_height/2, fy_index - bar_height/2,
                fy_index + bar_height/2, fy_index + bar_height/2];
78          patch(x_patch, y_patch, color_fill, 'EdgeColor', 'k', '
                LineWidth', 1.5);
79
80          text_str = sprintf('Start: %.1fs', t_start_opt);
81          text(t_start_opt, fy_index - bar_height/2 - 0.15, text_str,
                ...
82              'FontSize', 11, 'Color', 'k', 'HorizontalAlignment', '
                    left', 'VerticalAlignment', 'top');
83
84          ylim([0, 4]);
85          yticks([1, 2, 3]);
86          yticklabels({'FY1', 'FY2', 'FY3'});
87          xlim([-10, 50]);
88          xlabel('时间 (s)', 'FontSize', 12);
89
90          title(sprintf('协同遮蔽时序 (总长: %.2fs)', duration_opt), '
                FontSize', 14);
91
92          grid on;
93          ax = gca;
94          ax.GridAlpha = 0.3;
95          ax.XMinorGrid = 'on';
96          box on;
97          hold off;
98      else
99          fprintf('警告: 最优解未能形成有效遮蔽, 无法绘制甘特图。\n');
100     end
101 end
102
103 %% --- 辅助函数: 重新计算时序以用于绘图 ---
104 function [t_start, t_end, valid_dur] = Recompute_Time_Series(v,
        params, Env)
105     % 解包参数
106     a = params(1); td = params(2); ty = params(3);
107     dt = 0.001; % 高精度
108
109     % 物理运动计算
```

```
    ang_rad = deg2rad(a);
    Vel_Vec = [cos(ang_rad), sin(ang_rad), 0] * v;
    P_Drop = Env.Pos_FY1_Init + Vel_Vec * td;
    P_Pop = P_Drop + Vel_Vec * ty;
    P_Pop(3) = P_Pop(3) - 0.5 * Env.g * ty^2;

    t_pop = td + ty;
    t_max_sim = Env.Dist_Total_M1 / Env.V_M1;
    t_vec = 0 : dt : t_max_sim;

    valid_mask = false(size(t_vec));

    % 向量化检查每一时刻
    for k = 1:length(t_vec)
        t_curr = t_vec(k);

        % 1. 导弹位置
        if t_curr * Env.V_M1 >= Env.Dist_Total_M1
            P_M = Env.Pos_FakeTarget;
        else
            P_M = Env.Pos_M1_Init + Env.Dir_M1 * (Env.V_M1 * t_curr);
        end

        % 2. 烟雾位置
        if t_curr >= t_pop && (t_curr - t_pop) <= Env.Time_smoke_last
            t_rel = t_curr - t_pop;
            P_Smk = P_Pop - [0, 0, Env.V_smoke_sink] * t_rel;

            % 3. 遮挡判定
            P1 = P_M;
            P2 = Env.Pos_TrueTarget;
            Q = P_Smk;

            v_vec = P2 - P1;
            w_vec = Q - P1;

            c1 = dot(w_vec, v_vec);
            c2 = dot(v_vec, v_vec);

            if c1 <= 0
```

```
                dist = norm(Q - P1);
            elseif c2 <= c1
                dist = norm(Q - P2);
            else
                b = c1 / c2;
                Pb = P1 + b * v_vec;
                dist = norm(Q - Pb);
            end

            if dist <= Env.R_smoke
                valid_mask(k) = true;
            end
        end
    end

    % 提取结果
    if any(valid_mask)
        idx = find(valid_mask);
        t_start = t_vec(idx(1));
        t_end = t_vec(idx(end));
        valid_dur = sum(valid_mask) * dt;
    else
        t_start = NaN; t_end = NaN; valid_dur = 0;
    end
end

%% --- 内部微型 PSO 求解器 ---
function [best_pos, best_val] = Run_Micro_PSO(v, lb, ub, Env)
    % 粒子群参数
    n_part = 30;
    n_iter = 50;
    w = 0.6; c1 = 1.5; c2 = 1.5;
    n_vars = 3;

    % 初始化
    pos = repmat(lb, n_part, 1) + rand(n_part, n_vars) .* repmat(ub-
        lb, n_part, 1);

    % [种子注入]
    if v < 90
```

```
189         pos(1,:) = [176.88, 0.01, 2.5];
190     else
191         pos(1,:) = [178.46, 0.01, 3.3];
192     end
193
194     vel = zeros(n_part, n_vars);
195     pbest_pos = pos;
196     pbest_val = zeros(n_part, 1);
197     gbest_pos = zeros(1, n_vars);
198     gbest_val = -1e9;
199
200     % 评估初始种群
201     for i = 1:n_part
202         val = -Tactical_Sim_Engine(v, pos(i,1), pos(i,2), pos(i,3),
               Env.dt, Env);
203         pbest_val(i) = val;
204         if val > gbest_val
205             gbest_val = val;
206             gbest_pos = pos(i,:);
207         end
208     end
209
210     % 迭代
211     for t = 1:n_iter
212         for i = 1:n_part
213             r1 = rand(1, n_vars); r2 = rand(1, n_vars);
214             vel(i,:) = w*vel(i,:) + c1*r1.*(pbest_pos(i,:)-pos(i,:))
                   + c2*r2.*(gbest_pos-pos(i,:));
215             pos(i,:) = pos(i,:) + vel(i,:);
216             pos(i,:) = max(pos(i,:), lb);
217             pos(i,:) = min(pos(i,:), ub);
218
219             val = -Tactical_Sim_Engine(v, pos(i,1), pos(i,2), pos(i
                   ,3), Env.dt, Env);
220
221             if val > pbest_val(i)
222                 pbest_val(i) = val;
223                 pbest_pos(i,:) = pos(i,:);
224             end
225             if val > gbest_val
```

```
                gbest_val = val;
                gbest_pos = pos(i,:);
            end
        end
    end
    best_pos = gbest_pos;
    best_val = gbest_val;
end

%% --- 仿真核函数 ---
function score = Tactical_Sim_Engine(v, a, td, ty, dt, Env)
    ang_rad = deg2rad(a);
    Dir_Vec = [cos(ang_rad), sin(ang_rad), 0];
    Vel_Vec = Dir_Vec * v;
    P_Drop = Env.Pos_FY1_Init + Vel_Vec * td;
    P_Pop = P_Drop + Vel_Vec * ty;
    P_Pop(3) = P_Pop(3) - 0.5 * Env.g * ty^2;
    if P_Pop(3) < 0, score = 0; return; end

    t_pop = td + ty;
    t_start = max(0, t_pop);
    t_end = min(Env.Dist_Total_M1/Env.V_M1, t_pop + Env.
       Time_smoke_last);
    if t_start >= t_end, score = 0; return; end

    t_vec = t_start : dt : t_end;
    if isempty(t_vec), score=0; return; end

    d_m_vec = Env.V_M1 * t_vec;
    P_M_mat = Env.Pos_M1_Init' + Env.Dir_M1' * d_m_vec;
    P_Smk_mat = P_Pop' - [0;0;Env.V_smoke_sink] * (t_vec - t_pop);

    V_LOS_mat = Env.Pos_TrueTarget' - P_M_mat;
    W_mat = P_Smk_mat - P_M_mat;
    c1 = sum(W_mat .* V_LOS_mat, 1);
    c2 = sum(V_LOS_mat .* V_LOS_mat, 1);
    b = c1 ./ c2;

    Pb = P_M_mat + V_LOS_mat .* b;
    idx_less = b < 0; if any(idx_less), Pb(:, idx_less) = P_M_mat(:,
```

```
        idx_less); end
265     idx_more = b > 1; if any(idx_more), Pb(:, idx_more) = repmat(Env.
        Pos_TrueTarget', 1, sum(idx_more)); end
266
267     dists_sq = sum((P_Smk_mat - Pb).^2, 1);
268     count = sum(dists_sq <= Env.R_smoke^2);
269     score = -(count * dt);
270 end
```

Listing 3: Problem 3

```
1  function smoke_strategy_optimization()
2      clear; clc;
3      % 1. 环境与物理参数定义
4      params.Vm = 300;                        % 导弹速度
5      params.Pm0 = [20000, 0, 2000];          % 导弹初位置
6      params.Target_missile = [0, 0, 0];      % 导弹目标点(假目标)
7      params.P_true = [0, 200, 5];            % 真目标遮蔽判定点
8      params.P_uav0 = [17800, 0, 1800];       % 无人机初位置
9      params.g = 9.8;                         % 重力加速度
10     params.Vsink = 3;                       % 烟幕下沉速度
11     params.R = 10;                          % 有效半径
12     params.Duration = 20;                   % 烟幕持续时间
13
14     % 计算导弹运动单位向量
15     params.dir_m = (params.Target_missile - params.Pm0) / norm(params
           .Target_missile - params.Pm0);
16
17     % 2. PSO参数设置
18     nVar = 8;                               % 变量: [v, theta, t1, dt12,
           dt23, tau1, tau2, tau3]
19     lb = [70,  0,    0,  1.0, 1.0, 0, 0, 0];   % 下界
20     ub = [140, 2*pi, 60, 10,  10,  15, 15, 15]; % 上界 (t1和dt上限根
           据战场时空估计)
21
22     nPop = 50;                              % 种群规模
23     maxIter = 300;                          % 最大迭代次数
24     w = 0.8;                                % 惯性权重
25     c1 = 1.5;                               % 个体学习因子
26     c2 = 1.5;                               % 社会学习因子
27     stallLimit = 15;                        % 触发灾变的停滞步数
```

```
28
29      % 初始化种群
30      particles = repmat(struct('pos',[],'vel',[],'cost',0,'bestPos'
            ,[],'bestCost',-inf), nPop, 1);
31      globalBest.cost = -inf;
32      stallCounter = 0;
33
34      for i = 1:nPop
35          particles(i).pos = lb + (ub - lb) .* rand(1, nVar);
36          particles(i).vel = zeros(1, nVar);
37          particles(i).cost = fitness_func(particles(i).pos, params);
38          particles(i).bestPos = particles(i).pos;
39          particles(i).bestCost = particles(i).cost;
40          if particles(i).cost > globalBest.cost
41              globalBest = particles(i);
42          end
43      end
44
45      % 3. 迭代优化
46      for it = 1:maxIter
47          prevBestCost = globalBest.cost;
48
49          for i = 1:nPop
50              % 更新速度与位置
51              particles(i).vel = w*particles(i).vel + c1*rand(1,nVar)
                    .*(particles(i).bestPos - particles(i).pos) ...
52                                  + c2*rand(1,nVar).*(globalBest.pos -
                                        particles(i).pos);
53              particles(i).pos = particles(i).pos + particles(i).vel;
54              % 边界检查
55              particles(i).pos = max(min(particles(i).pos, ub), lb);
56
57              % 计算适应度
58              particles(i).cost = fitness_func(particles(i).pos, params
                    );
59
60              if particles(i).cost > particles(i).bestCost
61                  particles(i).bestCost = particles(i).cost;
62                  particles(i).bestPos = particles(i).pos;
63              end
```

```
            if particles(i).cost > globalBest.cost
                globalBest = particles(i);
            end
        end

        % --- 灾变机制 ---
        if abs(globalBest.cost - prevBestCost) < 1e-4
            stallCounter = stallCounter + 1;
        else
            stallCounter = 0;
        end

        if stallCounter >= stallLimit
            % 对除了全局最优外的50%粒子进行变异/重置
            for j = 1:floor(nPop/2)
                idx = randi(nPop);
                if idx ~= 1 % 假设1号不一定是最好，这里简单处理
                    particles(idx).pos = lb + (ub - lb) .* rand(1,
                        nVar);
                    particles(idx).vel = (rand(1,nVar)-0.5).*(ub-lb)
                        *0.2;
                end
            end
            stallCounter = 0;
            fprintf('Iter %d: Catastrophe triggered!\n', it);
        end

        fprintf('Iter %d: Max Duration = %.4f s\n', it, globalBest.
            cost);
    end

    % 4. 输出结果
    display_results(globalBest.pos, params);
end

%% 适应度函数：计算三枚弹的总有效遮蔽时长（并集）
function totalTime = fitness_func(x, params)
    v = x(1); theta = x(2);
    t_drops = [x(3), x(3)+x(4), x(3)+x(4)+x(5)]; % 投放时间
    taus = [x(6), x(7), x(8)];                    % 延时
```

```
101
102     dt = 0.05; % 时间步长
103     t_sim = 0:dt:100;
104     is_shielded = false(size(t_sim));
105
106     % 无人机速度向量
107     Vu = [v*cos(theta), v*sin(theta), 0];
108
109     % 计算每枚弹的爆炸点和状态
110     for i = 1:3
111         P_drop = params.P_uav0 + Vu * t_drops(i);
112         % 起爆点坐标（平抛运动）
113         P_exp = P_drop + [Vu(1)*taus(i), Vu(2)*taus(i), -0.5*params.g
                *taus(i)^2];
114         t_exp = t_drops(i) + taus(i);
115
116         % 检查每一秒是否遮蔽
117         for k = 1:length(t_sim)
118             tk = t_sim(k);
119             if tk >= t_exp && tk <= t_exp + params.Duration
120                 % 烟幕中心在 tk 时刻的位置
121                 P_cloud = P_exp - [0, 0, params.Vsink * (tk - t_exp)
                        ];
122                 % 导弹在 tk 时刻的位置
123                 Pm = params.Pm0 + params.dir_m * params.Vm * tk;
124                 % 计算点 P_cloud 到线段 (Pm -- P_true) 的距离
125                 dist = point_to_line_dist(P_cloud, Pm, params.P_true)
                        ;
126                 if dist <= params.R
127                     is_shielded(k) = true;
128                 end
129             end
130         end
131     end
132     totalTime = sum(is_shielded) * dt;
133 end
134
135 %% 工具函数：点到线段的距离
136 function d = point_to_line_dist(P, A, B)
137     v = B - A;
```

```
    w = P - A;
    c1 = dot(w, v);
    if c1 <= 0, d = norm(P - A); return; end
    c2 = dot(v, v);
    if c2 <= c1, d = norm(P - B); return; end
    b = c1 / c2;
    Pb = A + b * v;
    d = norm(P - Pb);
end

function display_results(x, params)
    v = x(1); theta = x(2);
    t_drops = [x(3), x(3)+x(4), x(3)+x(4)+x(5)];
    taus = [x(6), x(7), x(8)];
    Vu = [v*cos(theta), v*sin(theta), 0];

    fprintf('\n---␣优化结果␣---\n');
    fprintf('无人机航向␣(rad):␣%.4f␣(deg:␣%.2f)\n', theta, rad2deg(
        theta));
    fprintf('无人机速度␣(m/s):␣%.2f\n', v);

    for i = 1:3
    P_drop = params.P_uav0 + Vu * t_drops(i);
    P_exp = P_drop + [Vu(1)*taus(i), Vu(2)*taus(i), -0.5*params.g*
        taus(i)^2];
    fprintf('第%d枚烟幕弹:\n', i);
    fprintf('␣投放点:␣(%.2f,␣%.2f,␣%.2f)\n', P_drop(1), P_drop(2),
        P_drop(3))
    fprintf('␣爆炸点:␣(%.2f,␣%.2f,␣%.2f)\n', P_exp(1), P_exp(2),
        P_exp(3));
    end
    fprintf('最终有效干扰总时长:␣%.4f␣s\n', fitness_func(x, params));
end
```

Listing 4: Problem 4

```
function Problem4_MultiUAV_Final_Strategy()
    clc; close all;
    %% 1. 全局环境参数
    Env.g = 9.8;
    Env.Pos_True = [0, 200, 0];
```

```
    Env.Pos_M1   = [20000, 0, 2000];
    Env.V_M1     = 300;
    Env.Vec_M    = [0, 0, 0] - Env.Pos_M1;
    Env.Dist_M1  = norm(Env.Vec_M);
    Env.Dir_M1   = Env.Vec_M / Env.Dist_M1;
    Env.V_sink   = 3;
    Env.R_smk    = 10;
    Env.T_last   = 20;

    Env.Pos_UAVs = [
        17800, 0,    1800;  % FY1
        12000, 1400, 1400;  % FY2
        6000, -3000, 700    % FY3
    ];

    %% 2. 优化参数设置
    % 估算基准航向
    Base_Ang1 = rad2deg(atan2(200 - 0,    0 - 17800));
    Base_Ang2 = rad2deg(atan2(200 - 1400, 0 - 12000));
    Base_Ang3 = rad2deg(atan2(200 + 3000, 0 - 6000));

    % 变量顺序: [V, Ang, T_drop, T_delay] * 3架
    % 速度约束 70-140
    LB = [70, Base_Ang1-45, 0, 1,   70, Base_Ang2-45, 0, 1,   70,
       Base_Ang3-45, 0, 1];
    UB = [140, Base_Ang1+45, 15, 8, 140, Base_Ang2+45, 35, 8, 140,
       Base_Ang3+45, 55, 8];

    de_opts.NP = 200;
    de_opts.MaxIter = 800;
    de_opts.F = 0.5;
    de_opts.CR = 0.9;

    fprintf('=========================================================\n
       ');
    fprintf('␣␣␣␣␣␣问题4：三机协同立体封锁\n');
    fprintf('=========================================================\n
       ');

    try, if isempty(gcp('nocreate')), parpool; end; end
```

```
42      CostFunc = @(x) CostFunc_3UAV(x, Env);
43
44      tic;
45      % 运行优化（无初解）
46      [best_x, best_val] = Run_DE_3UAV(CostFunc, LB, UB, de_opts);
47      total_time = toc;
48
49      %% 3. 结果解析与策略输出
50      [~, real_shield_time] = CostFunc_3UAV(best_x, Env);
51      Res = Parse_Params(best_x);
52
53      fprintf('\n>>> 优化完成！耗时：%.2f 秒\n', total_time);
54      fprintf('>>> 最终最大遮蔽时长：%.4f 秒\n', real_shield_time);
55
56      fprintf('\n================== [最终投弹策略表]
            ==================\n');
57      fprintf('| 机号 | 飞行速度 | 飞行航向 | 投放时刻(s) | 延时时长(
            s) | 起爆时刻(s) |\n');
58      fprintf('
            |------|-----------|-----------|-------------|-------------|-----------
            n');
59      fprintf('| FY1  | %9.2f | %9.2f | %11.4f | %11.4f | %11.4f |\n',
            ...
60          Res(1).v, Res(1).a, Res(1).td, Res(1).ty, Res(1).tp);
61      fprintf('| FY2  | %9.2f | %9.2f | %11.4f | %11.4f | %11.4f |\n',
            ...
62          Res(2).v, Res(2).a, Res(2).td, Res(2).ty, Res(2).tp);
63      fprintf('| FY3  | %9.2f | %9.2f | %11.4f | %11.4f | %11.4f |\n',
            ...
64          Res(3).v, Res(3).a, Res(3).td, Res(3).ty, Res(3).tp);
65      fprintf('=========================================================\n
            ');
66
67      %% 4. 绘图
68      Plot_Effective_Gantt(Res, real_shield_time, Env);
69      Save_To_Excel(Res, 'result2.xlsx');
70  end
71
72
73  %  辅助函数：精确计算有效时间区间
```

```
function Intervals = Calculate_Exact_Intervals(Res, Env)
    % 初始化
    Intervals = repmat(struct('is_effective', false, 'start_time',
        NaN, 'end_time', NaN), 3, 1);

    dt = 0.01;
    T_max = Env.Dist_M1 / Env.V_M1 + 5;
    t_vec = 0 : dt : T_max;

    % 预计算每架飞机的烟雾位置参数
    SmokeData = [];
    for i=1:3
        ang = deg2rad(Res(i).a);
        V = [cos(ang), sin(ang), 0] * Res(i).v;
        P_Drop = Env.Pos_UAVs(i,:) + V * Res(i).td;
        P_Pop_Init = P_Drop + V * Res(i).ty;
        P_Pop_Init(3) = P_Pop_Init(3) - 0.5 * 9.8 * Res(i).ty^2;
        SmokeData(i).P_Pop_Init = P_Pop_Init;
        SmokeData(i).tp = Res(i).tp;
    end

    % 逐帧检测遮挡情况
    for i = 1:3
        mask = false(size(t_vec));
        for k = 1:length(t_vec)
            t = t_vec(k);
            % 1. 导弹当前位置
            if t * Env.V_M1 > Env.Dist_M1
                P_M = Env.Pos_True; % 已到达
            else
                P_M = Env.Pos_M1 + Env.Dir_M1 * (Env.V_M1 * t);
            end

            % 2. 判断该飞机的烟雾是否存在
            if t >= SmokeData(i).tp && t <= SmokeData(i).tp + Env.
               T_last
                t_rel = t - SmokeData(i).tp;
                P_Smk = SmokeData(i).P_Pop_Init - [0, 0, Env.V_sink *
                    t_rel];

```

```
                % 3. 判断是否遮挡
                v_los = Env.Pos_True - P_M;
                w_vec = P_Smk - P_M;
                c1 = dot(w_vec, v_los);
                c2 = dot(v_los, v_los);
                if c1 > 0
                    b = c1 / c2;
                    Pb = P_M + v_los * b;
                    dist_sq = sum((P_Smk - Pb).^2);
                    if dist_sq <= Env.R_smk^2
                        mask(k) = true;
                    end
                end
            end
        end

        % 提取起止时间
        if any(mask)
            idx = find(mask);
            Intervals(i).is_effective = true;
            Intervals(i).start_time = t_vec(idx(1));
            Intervals(i).end_time   = t_vec(idx(end));
        end
    end
end


function Res = Parse_Params(x)
    for i = 1:3
        idx = (i-1)*4;
        Res(i).v  = x(idx+1); Res(i).a  = x(idx+2);
        Res(i).td = x(idx+3); Res(i).ty = x(idx+4);
        Res(i).tp = Res(i).td + Res(i).ty;
    end
end

function [score, pure_time] = CostFunc_3UAV(x, Env)
    Res = Parse_Params(x);
    for i=1:3
        if Res(i).v < 70 || Res(i).v > 140, score = 1e6; pure_time =
```

```
            0; return; end
151     end
152     P_Pop = zeros(3, 3); Times_Pop = zeros(3, 1);
153     for i = 1:3
154         ang_rad = deg2rad(Res(i).a);
155         Vel_Vec = [cos(ang_rad), sin(ang_rad), 0] * Res(i).v;
156         P_Drop = Env.Pos_UAVs(i,:) + Vel_Vec * Res(i).td;
157         Disp = Vel_Vec * Res(i).ty; Disp(3) = Disp(3) - 0.5 * 9.8 *
               Res(i).ty^2;
158         P_Pop(i,:) = P_Drop + Disp; Times_Pop(i) = Res(i).tp;
159         if P_Pop(i,3) < 0, score = 1e5; pure_time = 0; return; end
160     end
161     t_start = min(Times_Pop); t_end = min(Env.Dist_M1/Env.V_M1, max(
           Times_Pop) + Env.T_last);
162     if t_start >= t_end, score = 1e5; pure_time = 0; return; end
163     Target_Points = [0, 200, 5; 0, 200, 10; 0, 200, 0; -7, 200, 5; 7,
            200, 5];
164     num_pts = 5; dt = 0.05; t_vec = t_start : dt : t_end;
165     total_coverage = 0; penalty_dist = 0; min_dists = [1e9, 1e9, 1e9
           ];
166     for t = t_vec
167         P_M = Env.Pos_M1 + Env.Dir_M1 * (Env.V_M1 * t);
               blocked_pts_count = 0;
168         for p = 1:num_pts
169             TP = Target_Points(p,:); v_los = TP - P_M; len_los_sq =
                   sum(v_los.^2); is_pt_blocked = false;
170             for k = 1:3
171                 if t >= Times_Pop(k) && t <= Times_Pop(k) + Env.
                       T_last
172                     P_Smk = P_Pop(k,:) - [0, 0, Env.V_sink * (t -
                           Times_Pop(k))];
173                     w_vec = P_Smk - P_M; c1 = dot(w_vec, v_los);
174                     if c1 > 0
175                         b = max(0, min(1, c1 / len_los_sq)); Pb = P_M
                               + v_los * b;
176                         d_sq = sum((P_Smk - Pb).^2); dist = sqrt(d_sq
                               );
177                         if p == 1 && dist < min_dists(k), min_dists(k
                               ) = dist; end
178                         if d_sq <= Env.R_smk^2, is_pt_blocked = true;
```

```
                            end
                        end
                    end
                end
                if is_pt_blocked, blocked_pts_count = blocked_pts_count + 1; end
            end
            total_coverage = total_coverage + (blocked_pts_count / num_pts) * dt;
        end
        pure_time = total_coverage;
        for k=1:3, penalty_dist = penalty_dist + max(0, min_dists(k) - Env.R_smk); end
        score = -total_coverage + 0.1 * penalty_dist;
end

function [best_mem, best_val] = Run_DE_3UAV(cost_func, lb, ub, opts)
    NP = opts.NP; D = length(lb);
    pop = repmat(lb, NP, 1) + rand(NP, D) .* repmat(ub-lb, NP, 1);

    val = zeros(NP, 1);
    parfor i=1:NP, val(i) = cost_func(pop(i,:)); end
    [best_val, idx] = min(val);
    best_mem = pop(idx, :);

    h = waitbar(0, '正在寻找最优策略...');
    for gen = 1 : opts.MaxIter
        F = opts.F * (1 - 0.2 * gen/opts.MaxIter);
        pop_new = pop; val_new = val;
        parfor i = 1 : NP
            r = randperm(NP, 3);
            mutant = best_mem + F * (pop(r(1),:) - pop(r(2),:));
            trial = pop(i, :);
            j_rand = randi(D);
            for j = 1 : D
                if rand < opts.CR || j == j_rand, trial(j) = mutant(j); end
            end
            trial = max(trial, lb); trial = min(trial, ub);
            t_v = feval(cost_func, trial);
```

```
            if t_v < val(i), pop_new(i,:) = trial; val_new(i) = t_v;
               end
        end
        pop = pop_new; val = val_new;
        [c_best, idx] = min(val);
        if c_best < best_val, best_val = c_best; best_mem = pop(idx,
           :); end

        if mod(gen, 50) == 0
            [~, t_real] = feval(cost_func, best_mem);
            waitbar(gen/opts.MaxIter, h, sprintf('Iter␣%d:␣%.2fs',
               gen, t_real));
        end
    end
    close(h);
end

%  绘图函数
function Plot_Effective_Gantt(Res, score, Env)
    figure('Color', 'w', 'Position', [100, 100, 700, 500], 'Name', '
       Final␣Strategy␣&␣Schedule');
    hold on;

    colors = {
        [100, 100, 255]/255,    % FY1 蓝
        [238, 130, 238]/255,    % FY2 紫
        [100, 200, 100]/255     % FY3 绿
    };

    % 计算精确区间
    Intervals = Calculate_Exact_Intervals(Res, Env);
    bar_height = 0.5;

    for i = 1:3
        % 只画有效遮蔽区间
        if Intervals(i).is_effective
            t_s = Intervals(i).start_time;
            t_e = Intervals(i).end_time;

            x_patch = [t_s, t_e, t_e, t_s];
```

```
250             y_patch = [i - bar_height/2, i - bar_height/2, i +
                    bar_height/2, i + bar_height/2];
251             patch(x_patch, y_patch, colors{i}, 'EdgeColor', 'k', '
                    LineWidth', 1.2, 'FaceAlpha', 0.9);
252
253             text(t_s, i - bar_height/2 - 0.2, sprintf('Start:␣%.1fs',
                     t_s), ...
254                  'FontSize', 10, 'Color', 'k', 'HorizontalAlignment',
                        'left');
255
256             text((t_s+t_e)/2, i, sprintf('Dur:␣%.2fs', t_e - t_s),
                    ...
257                   'FontSize', 9, 'Color', 'w', 'HorizontalAlignment',
                         'center', 'FontWeight', 'bold');
258         else
259             t_tp = Res(i).tp;
260             rectangle('Position', [t_tp, i-0.1, 1, 0.2], 'EdgeColor',
                     [0.8 0.8 0.8], 'LineStyle', '--');
261             text(t_tp, i-0.3, 'Ineffective', 'FontSize', 8, 'Color',
                    'r');
262         end
263     end
264
265     ylim([0, 4]);
266     yticks([1, 2, 3]);
267     yticklabels({'FY1', 'FY2', 'FY3'});
268
269     valid_times = [Intervals.start_time, Intervals.end_time];
270     valid_times = valid_times(~isnan(valid_times));
271     if isempty(valid_times), xlim([0, 30]); else, xlim([min(
            valid_times)-2, max(valid_times)+5]); end
272
273     xlabel('时间␣(s)', 'FontSize', 12);
274     title(sprintf('三机协同有效遮蔽时序␣(Total:␣%.4fs)', score), '
            FontSize', 14);
275     grid on; ax=gca; ax.GridAlpha=0.3; ax.XMinorGrid='on'; box on;
276     hold off;
277 end
278
279 function Save_To_Excel(Res, filename)
```

```
    data = zeros(3, 5);
    for i=1:3
        data(i,1) = Res(i).v;
        data(i,2) = Res(i).a;
        data(i,3) = Res(i).td;
        data(i,4) = Res(i).ty;
        data(i,5) = Res(i).tp;
    end
    try, writematrix(data, filename); catch, end
end
```

Listing 5: Problem 5

```
%% 2025 CUMCM Problem A - Question 5 Solver
% 5 Drones vs 3 Missiles, Max 3 Bombs each, Constant Altitude Flight
% Strategy: Double-Layer Optimization (GA + Heuristic Scan)
clear; clc; format shortG;

%% 1. 全局参数定义
global Mis Drones RealTarget g
% 真实目标（保护对象）
RealTarget = [0, 200, 0];
% 重力加速度
g = 9.8;

% --- 导弹参数 M1, M2, M3 ---
% 初始位置
M_pos = [20000, 0, 2000;
         19000, 600, 2100;
         18000, -600, 1900];
% 假目标位置（导弹瞄准点）
FakeTarget = [0, 0, 0];
% 速度标量
Vm = 300;

% 构建导弹结构体
Mis = struct();
for k = 1:3
    Mis(k).P0 = M_pos(k, :);
    dir_vec = FakeTarget - Mis(k).P0;
    dist = norm(dir_vec);
```

```
    Mis(k).dir = dir_vec / dist; % 单位方向向量
    Mis(k).flight_time = dist / Vm; % 飞行总时间
    Mis(k).V = Mis(k).dir * Vm;     % 速度向量
end

% --- 无人机初始参数 FY1 - FY5 ---
D_pos = [17800, 0, 1800;
         12000, 1400, 1400;
         6000, -3000, 700;
         11000, 2000, 1800;
         13000, -2000, 1300];
Drones = struct();
for i = 1:5
    Drones(i).P0 = D_pos(i, :);
    Drones(i).h = D_pos(i, 3); % 保持固定高度
end

%% 2. 遗传算法配置（外层优化）
% 决策变量：10个 [v1, ang1, v2, ang2, v3, ang3, v4, ang4, v5, ang5]
% v范围：[70, 140]，ang范围：[0, 2*pi]
nVars = 10;
lb = repmat([70, 0], 1, 5);
ub = repmat([140, 2*pi], 1, 5);

% GA 选项（为演示速度，种群和代数设得较小，比赛时建议加大）
options = optimoptions('ga', ...
    'PopulationSize', 150, ...     % 种群大小（建议 100+）
    'MaxGenerations', 80, ...      % 最大迭代次数（建议 100+）
    'Display', 'iter', ...         % 显示迭代过程
    'UseParallel', false);         % 如有并行工具箱可设为 true

fprintf('开始遗传算法优化...\n');
tic;
[best_vars, best_score] = ga(@fitness_func, nVars, [], [], [], [], lb
    , ub, [], options);
solve_time = toc;

%% 3. 结果解析与输出
fprintf('\n================优化结果================\n');
fprintf('求解耗时：%.2f秒\n', solve_time);
```

```
fprintf('最大总有效遮蔽时长:␣%.2f␣秒\n', -best_score); % 负负得正

% 解析最优解的详细投放策略
[~, final_plan] = fitness_func(best_vars);

fprintf('\n>>>␣无人机飞行策略␣<<<\n');
for i = 1:5
    v = best_vars(2*i-1);
    ang = best_vars(2*i);
    fprintf('FY%d:␣速度␣%.2f␣m/s,␣航向␣%.2f␣度␣(弧度␣%.2f)\n', ...
        i, v, rad2deg(ang), ang);
end

fprintf('\n>>>␣烟幕弹投放详细方案␣(Result3)␣<<<\n');
fprintf('%-6s␣|␣%-6s␣|␣%-8s␣|␣%-8s␣|␣%-8s␣|␣%-15s\n', ...
    'Drone', 'BombID', 'DropTime', 'ExpTime', 'Target', '
       CoverDuration');
fprintf('
   ----------------------------------------------------------------\
   n');

total_M1 = 0; total_M2 = 0; total_M3 = 0;

for i = 1:5
    drone_plan = final_plan{i};
    if isempty(drone_plan)
        continue;
    end
    for j = 1:size(drone_plan, 1)
        % plan row: [t_drop, t_exp, missile_idx, cover_start,
           cover_end, score]
        d_time = drone_plan(j, 1);
        e_time = drone_plan(j, 2);
        m_idx = drone_plan(j, 3);
        c_dur = drone_plan(j, 5) - drone_plan(j, 4);

        fprintf('FY%-4d␣|␣#%-5d␣|␣%-8.2f␣|␣%-8.2f␣|␣M%-7d␣|␣%.2f␣s␣
           [%.1f-%.1f]\n', ...
            i, j, d_time, e_time, m_idx, c_dur, drone_plan(j,4),
               drone_plan(j,5));
```

```
    end
end
fprintf('========================================\n');


%% ------------------------------------------------------------
%  Local Functions (无需单独文件, 直接包含在脚本中)
%  ------------------------------------------------------------

function [score, all_plans] = fitness_func(vars)
    % 适应度函数: 输入 10 个变量, 输出总遮蔽时间的负值 (求最小化)
    % all_plans 用于最后输出详细信息

    global Mis RealTarget

    % 存储每枚导弹被遮蔽的时间段集合
    % 结构: cell array {M1_intervals; M2_intervals; M3_intervals}
    missile_covers = cell(3, 1);
    all_plans = cell(5, 1);

    % 遍历 5 架无人机
    for i = 1:5
        % 提取当前无人机的决策变量
        v = vars(2*i-1);
        ang = vars(2*i);

        % 计算该无人机的最佳投弹策略 (内层贪心/遍历)
        [drone_intervals, drone_plan] = calculate_drone_strategy(i, v
            , ang);

        % 收集结果
        all_plans{i} = drone_plan;
        for k = 1:3
            if ~isempty(drone_intervals{k})
                missile_covers{k} = [missile_covers{k};
                    drone_intervals{k}];
            end
        end
    end

```

```
    % 计算总有效遮蔽时间（处理并集）
    total_time = 0;
    for k = 1:3
        intervals = missile_covers{k};
        if isempty(intervals)
            continue;
        end
        % 计算区间并集总长度
        union_len = calc_union_length(intervals);
        total_time = total_time + union_len;
    end

    % GA 是求最小值，所以取负
    score = -total_time;
end

function [intervals_by_missile, selected_plan] =
    calculate_drone_strategy(drone_idx, v, ang)
    % 内层核心函数：给定无人机轨迹，找出最优的3次投弹
    % 输入：无人机ID，速度，航向
    % 输出：针对M1-M3的遮蔽区间，以及具体的投弹计划

    global Drones Mis g

    P0 = Drones(drone_idx).P0;
    intervals_by_missile = cell(3, 1);

    % 1. 生成候选投弹列表
    % 离散化遍历投弹时间 t_drop
    % 无人机最长飞行时间估计：导弹最多飞 70秒，取 0-70s
    t_drops = 0 : 1.0 : 70;
    candidates = []; % 格式：[t_drop, t_exp, missile_idx, start_t,
        end_t, duration]

    % 无人机速度向量
    V_drone = [v * cos(ang), v * sin(ang), 0];

    for t_d = t_drops
        % 此刻无人机位置
        P_drop = P0 + V_drone * t_d;
```

```
178
179         % 遍历每枚导弹，看能否拦截
180         for m_id = 1:3
181             % 几何逆推：为了拦截导弹，烟幕弹需要在空中爆炸并形成云团
182             % 烟幕弹水平位置随时间变化：P(t)xy = P_drop_xy +
                    V_drone_xy * (t - t_d)
183             % 烟幕弹垂直位置：Z(t) = P_drop_z - 0.5*g*(t - t_d)^2
184
185             % 简化逻辑：扫描可能的起爆延迟 delta_t
186             % 延迟范围：1s 到 15s（根据高度差估算，1800m自由落体约19s
                    ）
187             for delta_t = 1 : 2 : 15
188                 t_e = t_d + delta_t;
189
190                 % 爆炸点位置
191                 P_boom = P_drop + V_drone * delta_t;
192                 P_boom(3) = P_drop(3) - 0.5 * g * delta_t^2;
193
194                 % 如果爆炸点已经在地下，跳过
195                 if P_boom(3) < 0
196                     continue;
197                 end
198
199                 % 检查该爆炸点产生的云团能否遮蔽导弹 m_id
200                 [is_block, t_start, t_end] = check_interception(
                        P_boom, t_e, m_id);
201
202                 if is_block
203                     dur = t_end - t_start;
204                     if dur > 0.1
205                         % 记录候选策略
206                         candidates = [candidates; t_d, t_e, m_id,
                                t_start, t_end, dur];
207                     end
208                 end
209             end
210         end
211     end
212
213     % 2. 贪心选择最优的 3 个不冲突的投弹时机
```

```
    selected_plan = []; % [t_drop, t_exp, m_idx, t_start, t_end, dur]

    if isempty(candidates)
        return;
    end

    % 按遮蔽时长降序排列
    [~, sort_idx] = sort(candidates(:, 6), 'descend');
    sorted_cands = candidates(sort_idx, :);

    count = 0;
    for i = 1:size(sorted_cands, 1)
        if count >= 3
            break;
        end

        cand = sorted_cands(i, :);
        t_d_curr = cand(1);

        % 检查与已选方案的时间间隔约束 (|t_d1 - t_d2| >= 1)
        conflict = false;
        for j = 1:size(selected_plan, 1)
            if abs(t_d_curr - selected_plan(j, 1)) < 1.0 - 1e-5
                conflict = true;
                break;
            end
        end

        if ~conflict
            selected_plan = [selected_plan; cand];
            % 将结果加入输出格式
            m_idx = cand(3);
            intervals_by_missile{m_idx} = [intervals_by_missile{m_idx
                }; cand(4), cand(5)];
            count = count + 1;
        end
    end
end

function [blocked, t_start, t_end] = check_interception(P_boom, t_exp
```

```
    , m_idx)
253     % 检查单个云团对特定导弹的遮蔽情况
254     global Mis RealTarget
255 
256     blocked = false; t_start = 0; t_end = 0;
257 
258     M_struct = Mis(m_idx);
259 
260     % 烟幕有效时间窗口
261     cloud_life_start = t_exp;
262     cloud_life_end = t_exp + 20;
263 
264     % 导弹飞行结束时间
265     mis_end = M_struct.flight_time;
266 
267     % 实际有效检测时间段（取交集）
268     check_start = max(0, cloud_life_start);
269     check_end = min(mis_end, cloud_life_end);
270 
271     if check_start >= check_end
272         return;
273     end
274 
275     % 离散化检测遮蔽（步长 0.2s）
276     ts = check_start : 0.2 : check_end;
277     is_cov = false(size(ts));
278 
279     % 云团下沉速度
280     V_sink = [0, 0, -3];
281 
282     for k = 1:length(ts)
283         t = ts(k);
284         % 此刻云团中心
285         P_c = P_boom + V_sink * (t - t_exp);
286         % 此刻导弹位置
287         P_m = M_struct.P0 + M_struct.V * t;
288 
289         % 遮蔽判断核心：
290         % 1. 计算 P_c 到线段 P_m -> RealTarget 的距离
291         dist = point_to_segment_dist(P_c, P_m, RealTarget);
```

```
292
293         % 2. 判断是否在半径 10m 内
294         if dist <= 10
295             % 3. 补充判断：云团是否在导弹前方（简单判断距离关系）
296             d_m_target = norm(P_m - RealTarget);
297             d_c_target = norm(P_c - RealTarget);
298             if d_c_target < d_m_target % 云团在导弹和目标之间
299                 is_cov(k) = true;
300             end
301         end
302     end
303
304     if any(is_cov)
305         blocked = true;
306         valid_indices = find(is_cov);
307         t_start = ts(valid_indices(1));
308         t_end = ts(valid_indices(end));
309     end
310 end
311
312 function d = point_to_segment_dist(P, A, B)
313     % 计算点 P 到线段 AB 的最短距离
314     AB = B - A;
315     AP = P - A;
316
317     % 投影系数
318     t = dot(AP, AB) / dot(AB, AB);
319
320     if t < 0
321         closest = A; % 投影在 A 外侧
322     elseif t > 1
323         closest = B; % 投影在 B 外侧
324     else
325         closest = A + t * AB; % 投影在线段上
326     end
327
328     d = norm(P - closest);
329 end
330
331 function len = calc_union_length(intervals)
```

```
    % 计算多个时间区间的并集总长度
    % intervals: Nx2 matrix [start, end]
    if isempty(intervals)
        len = 0; return;
    end

    % 按开始时间排序
    intervals = sortrows(intervals, 1);

    merged = intervals(1, :);
    idx = 1;

    for i = 2:size(intervals, 1)
        curr = intervals(i, :);
        if curr(1) < merged(idx, 2) % 有重叠
            merged(idx, 2) = max(merged(idx, 2), curr(2));
        else
            idx = idx + 1;
            merged(idx, :) = curr;
        end
    end

    len = sum(merged(:, 2) - merged(:, 1));
end
```

Listing 6: Problem 5 Gantt Chart Plotter

```
    clc; clear; close all;

%% 1. Data Input
% Format: {DroneID, BombID, TargetID, StartTime, EndTime}
taskDataRaw = {
    'FY1', '#1', 'M1',  3.00,  6.80;
    'FY1', '#2', 'M1',  4.60,  8.40;
    'FY2', '#1', 'M2',  9.00, 13.00;
    'FY2', '#2', 'M1', 24.80, 25.60;
    'FY3', '#2', 'M3', 35.60, 38.60;
    'FY3', '#1', 'M2', 37.80, 41.20;
    'FY3', '#3', 'M1', 49.40, 51.00;
    'FY4', '#1', 'M2', 15.00, 19.40;
    'FY5', '#1', 'M1', 19.60, 23.60;
```

```
};

% Sort by Drone ID and Start Time to ensure correct stacking logic
[~, idx] = sortrows(taskDataRaw, [1, 4]);
taskData = taskDataRaw(idx, :);

%% 2. Plot Settings
droneNames = {'FY5', 'FY4', 'FY3', 'FY2', 'FY1'}; % Y-axis from
    bottom to top
n_drones = length(droneNames);

% Define Colors for Targets
targetColors = containers.Map();
targetColors('M1') = [0.85, 0.33, 0.10]; % Orange-Red
targetColors('M2') = [0.00, 0.45, 0.74]; % Blue
targetColors('M3') = [0.47, 0.67, 0.19]; % Green

barHeight = 0.35;    % Height of a single bar
levelSpacing = 0.45; % Vertical spacing between overlapping tasks

figure('Color', 'w', 'Position', [100, 100, 1200, 700]);
hold on; grid on; box on;

h_legend = [];
legend_labels = {};

%% 3. Plotting with Overlap Handling
% Map to track the end time of each level for each drone
droneLevels = containers.Map();
for i = 1:n_drones
    droneLevels(droneNames{i}) = [];
end

for i = 1:size(taskData, 1)
    droneID = taskData{i, 1};
    bombID = taskData{i, 2};
    targetID = taskData{i, 3};
    t_start = taskData{i, 4};
    t_end = taskData{i, 5};

```

```
    base_y_idx = find(strcmp(droneNames, droneID));

    % --- Level Calculation Logic ---
    levels = droneLevels(droneID);
    task_level = 0;
    foundLevel = false;

    % Try to find a level where this task fits without overlapping
    for lvl = 1:length(levels)
        if t_start >= levels(lvl)
            task_level = lvl - 1;
            levels(lvl) = t_end;
            foundLevel = true;
            break;
        end
    end

    % Create a new level if needed
    if ~foundLevel
        task_level = length(levels);
        levels = [levels, t_end];
    end
    droneLevels(droneID) = levels;
    % -------------------------------

    % Calculate Y position
    y_center = base_y_idx + task_level * levelSpacing - (length(
       levels)-1)*levelSpacing/2;

    % Get Color
    if isKey(targetColors, targetID)
        c = targetColors(targetID);
    else
        c = [0.5 0.5 0.5];
    end

    % Draw Patch
    x_patch = [t_start, t_end, t_end, t_start];
    y_low = y_center - barHeight/2;
    y_high = y_center + barHeight/2;
```

```
    y_patch = [y_low, y_low, y_high, y_high];

    h = patch(x_patch, y_patch, c, 'EdgeColor', 'k', 'FaceAlpha', 0.8, 'LineWidth', 0.5);

    % Handle Legend (Unique entries only)
    if ~ismember(targetID, legend_labels)
        h_legend = [h_legend, h]; %#ok<AGROW>
        legend_labels = [legend_labels, targetID]; %#ok<AGROW>
    end

    % Text: Bomb ID (Inside bar)
    mid_x = (t_start + t_end) / 2;
    text(mid_x, y_center, bombID, ...
        'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle', ...
        'FontSize', 8, 'FontWeight', 'bold', 'Color', 'w');

    % Text: Time Interval (Outside bar, alternating position)
    label_y_pos = y_high + 0.1;
    if mod(task_level, 2) ~= 0
        label_y_pos = label_y_pos + 0.15;
    end

    text(mid_x, label_y_pos, sprintf('[%.1f-%.1f]', t_start, t_end), ...
        'HorizontalAlignment', 'center', 'FontSize', 7.5, 'Color', [0.2 0.2 0.2], 'Interpreter', 'none');
end

%% 4. Axes and Labels (English)
set(gca, 'YTick', 1:n_drones, 'YTickLabel', droneNames, 'FontSize', 11);
xlabel('Time (s)', 'FontSize', 12, 'FontWeight', 'bold');
ylabel('Drone ID', 'FontSize', 12, 'FontWeight', 'bold');

% Title in English
title('Smoke Screening Effective Time Gantt Chart (Total: 26.60s)', 'FontSize', 14);

```

```
max_time = max([taskData{:, 5}]);
xlim([0, max_time + 5]);
ylim([0.2, n_drones + 1.5]);

% Legend in English (Sorted)
[~, sort_idx] = sort(legend_labels);
% Assuming targets are named M1, M2, etc. If you want "Target M1", modify below:
legend(h_legend(sort_idx), legend_labels(sort_idx), ...
    'Location', 'NorthEast', 'Orientation', 'horizontal', 'FontSize', 10, 'Box', 'on');

hold off;
```