烟幕干扰弹的无人机投放策略研究

队伍编号：_____

2025 年 12 月 19 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，建立了基于运动学模型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，我们建立了烟幕弹自由落体与扩散模型。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。

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。

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。

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：

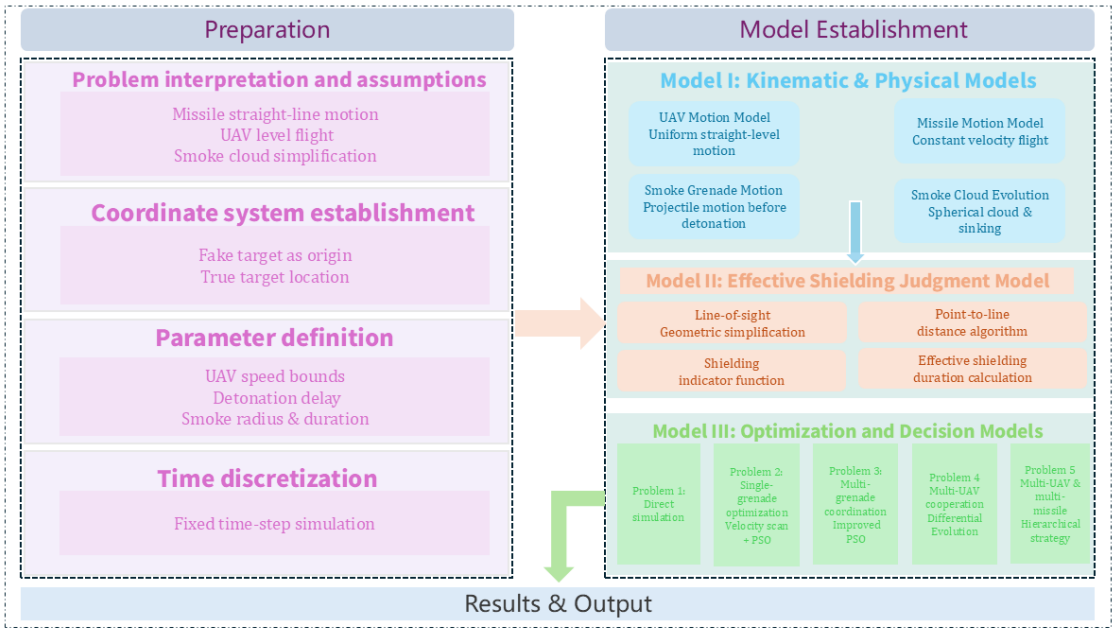


图 2: 总体分析框架

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4所示：

表 1: 符号说明（按物理意义分组排序）

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	无人机飞行航向角	rad
g	重力加速度	m/s^2
t_{drop}	烟幕弹投放时刻	s
t_{burst}	烟幕弹起爆时刻	s
t_{delay}	烟幕弹起爆延时	s
R_{cloud}	烟幕云团有效遮蔽半径	m
v_{cloud}	烟幕云团下沉速度	m/s
$\boldsymbol{P}_{S,k}(t)$	第 k 枚烟幕弹 t 时刻位置	m
$\boldsymbol{P}_{B,k}$	第 k 枚烟幕弹起爆位置	m
$D(t)$	烟幕云团中心到导弹-目标连线距离	m
T_{cover}	有效遮蔽时间	s
$I_j(t)$	第 j 枚导弹的遮蔽指示函数	-

5 模型建立

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。第 j 枚导弹的初始位置位于点 $P_{M_j}(0)$，导弹的飞行方向向量为：

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

导弹在任意时刻 t 的位置为：

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

5.1 无人机运动模型构建

第 i 枚无人机的初始位置为 $\boldsymbol{P}_{FY_i}(0)$, 第 i 架无人机飞行速度为 v_i，航向角为 α_i ，其速度向量表示为：

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

无人机在任意时刻 t 的位置为：

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

5.2 烟幕干扰弹的运动模型

设第 k 枚烟幕弹在 $t_{drop,k}$ 时刻投放，起爆延时为 $t_{delay,k}$，重力加速度为 g 起爆时刻为：

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

起爆位置为：

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

5.3 烟幕弹的下沉模型

烟幕弹起爆后形成半径为 R_{cloud} 的球形云团，其中心以速度 v_{cloud} 匀速下沉：

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

5.4 有效遮蔽判定模型

5.4.1 几何遮挡条件

根据题目，可知无人机及导弹的初始位置离真目标（原点为（0，200，0），半径为 7，高度为 10 的圆柱体）所在位置十分遥远，而题中未提到无人机和导弹的视野，那说明其不在题目考虑范围内。我们可以合理假设导弹视野为全范围，即只要在导弹正前方的事物，都处于导弹视野范围内，那对于一个距离十分远，且相对于距离量级来说大小很小的一个真目标，便于进行计算和后续的优化问题，我们可以将其简化为一个质点，进而将遮蔽条件转化为：当烟幕云团中心到“导弹与真目标质心连线”的垂直距离小于等于烟幕半径时，视为有效遮蔽。

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：

$$r = \frac{\overrightarrow{AP} \cdot \overrightarrow{AB}}{\|\overrightarrow{AB}\|^2} \tag{8}$$

距离分三种情况讨论：

$$D(t) = \begin{cases} \|\overrightarrow{AP}\| & \text{if } r \leq 0 \text{ (导弹后方)} \\ \|P - B\| & \text{if } r \geq 1 \text{ (目标后方)} \\ \frac{\|\overrightarrow{AP} \times \overrightarrow{AB}\|}{\|\overrightarrow{AB}\|} & \text{if } 0 < r < 1 \text{ (线段投影内)} \end{cases} \tag{9}$$

5.5 遮蔽时间定义

遮蔽时间定义为：

$$I_j(t) = \begin{cases} 1, & \text{if } D_{k,i}(t) \leq R_{cloud} \text{ and } t \geq t_{burst,k} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

其中 $D_{k,i}(t)$ 为第 k 枚烟幕弹对第 j 枚导弹在时刻 t 的点到线段距离。
导弹 j 的总有效遮蔽时长为：

$$T_j = \int_0^{T_{end}} I_j\,(t)\,dt \tag{11}$$

在不同问题中，优化目标统一表示为：

$$\max\ T_{cover}\ = \sum_j T_j \tag{12}$$

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。针对本题，飞行策略和投弹策略为题目给定，本题求解核心在于对已知任务的仿真模拟，所以直接使用时间步长离散化仿真即可解决本题。

6.1.1 参数代入与场景设定

根据题目条件，无人机 FY1 的初始位置为 $\mathbf{P}_{FY_1}(0) = (17800,\, 0,\, 1800)$, 飞行速度为 $v_{FY_1} = 120\,\mathrm{m/s}$, 航向指向假目标 $\boldsymbol{P_O} = (0, 0, 0)$。烟幕干扰弹在受领任务后 $t_{drop} = 1.5\,\mathrm{s}$ 时投放，起爆延迟时间为 $t_{delay} = 3.6\,\mathrm{s}$. 导弹 M1 的初始位置为 $\mathbf{P}_M(0) = (20000,\, 0,\, 2000)$, 飞行速度为 $v_M = 300\,\mathrm{m/s}$, 并沿直线飞向假目标。烟幕云团半径取 $R_s = 10\,\mathrm{m}$ 下沉速度取 $v_s = 3\,\mathrm{m/s}$.

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。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

$$t_{burst} = t_{drop} + t_{delay}, \tag{13}$$

则其起爆位置为

$$\mathbf{P}_{burst} = \mathbf{P}_{FY_1}(t_{drop}) + \boldsymbol{v}_{FY_1} t_{delay} - \left(0,\, 0,\, \tfrac{1}{2} g t_{delay}^2\right). \tag{14}$$

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

$$\mathbf{P}_S(t) = \mathbf{P}_{burst} - (0,\, 0,\, v_s(t - t_{burst})), \quad t \geq t_{burst}. \tag{15}$$

导弹 M1 在任意时刻 t 的位置为

$$\mathbf{P}_M(t) = \mathbf{P}_M(0) + v_M \mathbf{n}_M t, \tag{16}$$

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

为计算烟幕干扰弹对导弹 M1 的有效遮蔽时长，将导弹飞行全过程时间区间 $[0, T_{end}]$ 进行离散化处理。设时间步长为 Δt，离散时间点为

$$t_k = k\Delta t, \quad k = 0, 1, \ldots, N. \tag{17}$$

在每一离散时刻 t_k，计算烟幕云团中心到导弹与真目标连线的最短距离 $d(t_k)$。当满足

$$d(t_k) \leq R_{cloud} \quad \text{且} \quad t_k \geq t_{burst} \tag{18}$$

时，认为该时刻烟幕对导弹 M1 实现有效遮蔽。定义遮蔽指示函数

$$I(t_k) = \begin{cases} 1, & d(t_k) \leq R_{cloud},\ t_k \geq t_{burst}, \\ 0, & \text{otherwise}. \end{cases} \tag{19}$$

则烟幕干扰弹对导弹 M1 的总有效遮蔽时长可近似表示为

$$T_{cover} \approx \sum_{k=0}^{N} I(t_k)\, \Delta t. \tag{20}$$

6.1.4 模型汇总

$$
\begin{cases}
\mathbf{P}_M(t) = \mathbf{P}_M(0) + V_m \mathbf{n}_M t \\
\mathbf{P}_{FY_1}(t) = \mathbf{P}_{FY_1}(0) + \boldsymbol{v}_{FY_1} t \\
\mathbf{P}_B = \mathbf{P}_{FY_1}(t_{drop}) + \boldsymbol{v}_{FY_1} t_{delay} - \left(0, 0, \frac{1}{2} g t_{delay}^2\right) \\
\mathbf{P}_S(t) = \mathbf{P}_B - (0, 0, v_{cloud}(t - t_{burst})) \\
r = \dfrac{(\mathbf{P}_S(t) - \mathbf{P}_M(t)) \cdot (\mathbf{P}_T - \mathbf{P}_S(t))}{||\mathbf{P}_T - \mathbf{P}_M(t)||^2} \\
D(t) = \begin{cases} \|(\mathbf{P}_S(t) - \mathbf{P}_M(t))\| & \text{if } r \le 0 \\ \|(\mathbf{P}_T - \mathbf{P}_S(t))\| & \text{if } r \ge 1 \\ \frac{(\mathbf{P}_S(t) - \mathbf{P}_M(t)) \times (\mathbf{P}_T - \mathbf{P}_S(t))}{||\mathbf{P}_T - \mathbf{P}_M(t)||^2} & \text{if } 0 < r < 1 \end{cases} \\
I(t) = \begin{cases} 1, & \text{if } D(t) \le R_{cloud} \text{ and } t \ge t_{burst} \\ 0, & \text{otherwise} \end{cases} \\
T = \displaystyle\int_0^{T_{end}} I(t)\, dt
\end{cases}
\tag{21}
$$

6.1.5 求解结果：

经计算，结果如下图4所示，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 $1.4s$。

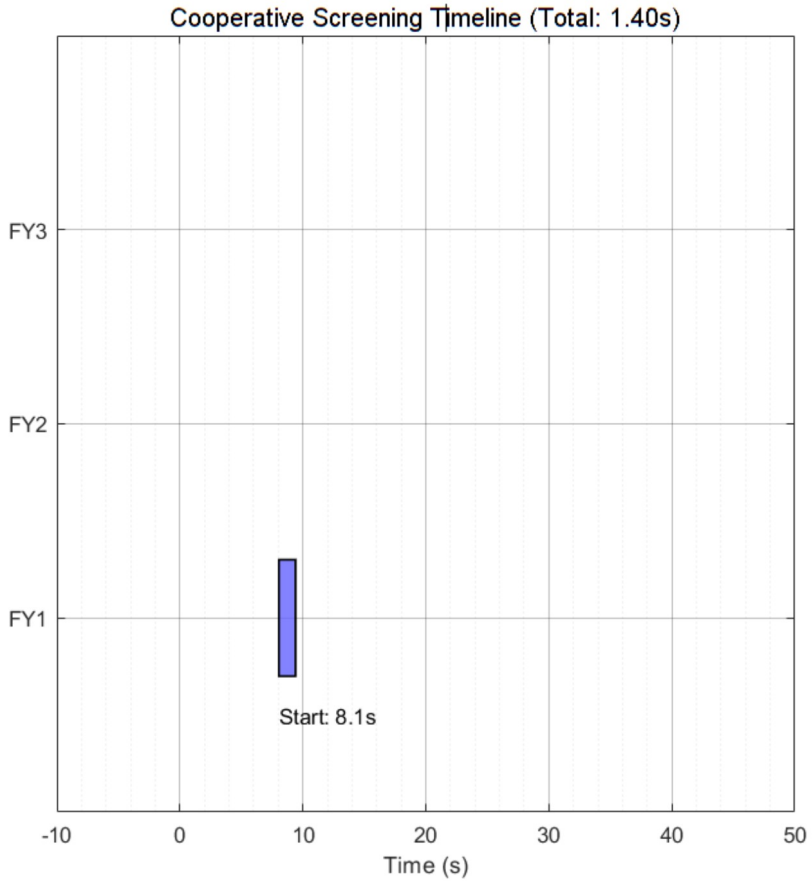


图 3: 问题一结果

6.2 问题二求解：单机单弹优化

在问题二中，无人机 FY1 的任务是通过自身的飞行参数和投弹时机，使得单枚烟雾弹对导弹 M1 的有效遮蔽时间最大化。

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：飞行速度 v_{FY1}, 飞行航向角 α, 投放时刻 t_{drop}, 延时起爆时间 t_{delay}, 由此定义出决策向量 X:

$$X = [v_{FY_1}, \alpha, t_{drop}, t_{delay}]^T \tag{22}$$

6.2.2 运动状态方程

基于问题一的运动学模型，需要将无人机的速度向量参数化。无人机速度向量可表示为：

$$\boldsymbol{v_{FY_1}}\left(v_{FY_1}, \alpha\right) = [v_{FY_1}\cos\alpha, v_{FY_1}\sin\alpha, 0]^T \tag{23}$$

投放点位置和起爆点位置更新为用 X 表示的函数：

$$\boldsymbol{P}_{drop}(X) = \boldsymbol{P}_{FY_1}(0) + \boldsymbol{v}_{\boldsymbol{FY_1}}(v_{FY_1}, \alpha) \cdot t_{drop} \tag{24}$$

$$\boldsymbol{P}_{burst}(X) = \boldsymbol{P}_{drop}(X) + \boldsymbol{v}_{\boldsymbol{FY_1}}(v_{FY_1}, \alpha) \cdot t_{delay} + \left[0, 0, -\frac{1}{2} g t_{delay}^2\right]^T \tag{25}$$

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：

$$maxJ(X) = T_{cover} = \int_0^{T_{end}} I(t; X)\, dt \tag{26}$$

6.2.4 约束条件

根据题目和对于模型速度特征的分析，约束条件如下: 速度约束：

$$70 \leq v_{FY_1} \leq 140 \tag{27}$$

时间约束：

$$0 \leq t_{drop} \leq 12, 1 \leq t_{delay} \leq 8 \tag{28}$$

高度约束：

$$z_{burst}(X) > 0 \tag{29}$$

6.2.5 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。

算法整体架构 外层循环（针对速度，因为速度对优化结果的影响最大）：将速度在 [70，140] 区间内以 0.5m/s 为步长进行离散化扫描。内层循环（针对剩余参数）：针对每一个固定的速度，使用 PSO 算法寻找最优组合。这样分层寻优的好处是将优化问题维度从 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。

粒子群算法设计 在内层针对剩余决策变量进行寻优。设置粒子群规模为 N=30; 粒子速度与位置更新公式：

$$v_i^{k+1} = w v_i^k + c_1 r_1 \left(p_{best,i} - x_i^k\right) + c_2 r_2 \left(g_{best} - x_i^k\right) \tag{30}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{31}$$

适应度函数：

$$F(x) = -T_{cover}(x) \tag{32}$$

6.2.6 求解结果

经过计算，得到最优投放策略如下：
无人机以 196.91° 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5所示。

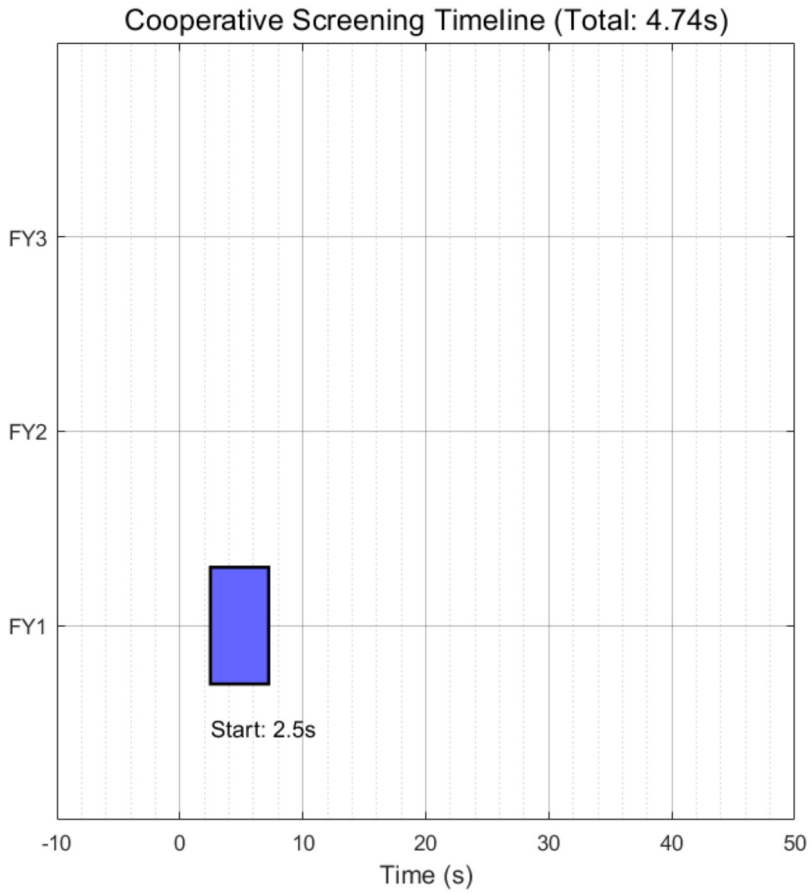


图 4: 问题二结果

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：
飞行速度 v_{FY_1}

飞行航向角 α
投放时刻 $t_{drop,i}$
延时起爆时间 $t_{delay,i}$
由此定义出决策向量 X:

$$\mathbf{X} = [v_{FY_1}, \alpha, t_{drop,1}, t_{drop,2}, t_{drop,3}, t_{delay,1}, t_{delay,2}, t_{delay,3}]^T \tag{33}$$

6.3.2 状态方程更新

基于问题一的运动学模型，第 k 枚烟幕弹的运动状态由下式描述:
1. 投放点坐标：

$$P_{drop,k}(\mathbf{X}) = B_0 + \mathbf{v}_{FY1}(v_{FY1}, \alpha) \cdot t_{drop,k} \tag{34}$$

2. 起爆点坐标

$$P_{burst,k}(\mathbf{X}) = P_{drop,k}(\mathbf{X}) + \mathbf{v}_{FY1}(v_{FY1}, \alpha) \cdot t_{delay,k} + \left[0, 0, -\frac{1}{2} g t_{delay,k}^2\right]^T \tag{35}$$

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

$$P_{S,k}(t) = P_{burst,k} - [0, 0, v_{sink} \cdot (t - t_{burst,k})]^T \quad , \quad t \in [t_{burst,k}, t_{burst,k} + 20] \tag{36}$$

6.3.3 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 k 枚烟幕弹在 t 时刻的遮蔽指示函数 $I_k(t; \mathbf{X})$:

$$I_k(t; \mathbf{X}) = \begin{cases} 1, & \text{if } D_k(t) \le R_{cloud} \text{ and } t \in [t_{burst,k}, t_{burst,k} + 20] \\ 0, & \text{otherwise} \end{cases} \tag{37}$$

其中 $D_k(t)$ 为第 k 个烟幕中心到“导弹-目标”视线的距离（计算方法同问题一）。系统在 t 时刻的遮蔽状态是所有单个烟幕弹的逻辑或

$$I_{total}(t; \mathbf{X}) = \max\{I_1(t), I_2(t), I_3(t)\} \tag{38}$$

最终优化目标为最大化总有效遮蔽时长 T_{cover}：

$$maxJ(\mathbf{X}) = T_{cover} = \int_0^{T_{end}} I_{total}(t; \mathbf{X}) dt \tag{39}$$

6.3.4 约束条件

根据题目和对于模型速度特征的分析，约束条件如下: 除了满足问题二中的飞行与高度约束外，本问必须严格满足多枚弹之间的操作间隔约束：
投弹间隔约束：题目要求每架无人机投放两枚烟幕弹至少间隔 1s。

$$t_{drop,k+1} - t_{drop,k} \ge 1, \quad k = 1, 2 \tag{40}$$

基本边界约束：

$$\begin{cases} 70 \le v_{FY_1} \le 140 \\ 0 < t_{drop,1} < t_{drop,2} < t_{drop,3} \le 12 \\ 1 \le t_{delay,k} \le 8, \quad k = 1,2,3 \end{cases} \tag{41}$$

高度约束：

$$z_{burst,k}(\mathbf{X}) > 0, \quad k = 1,2,3 \tag{42}$$

6.3.5 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，传统的优化算法容易陷入停滞，因此我们采用引入灾变机制的改进粒子群算法求解。

算法整体架构 不再采用网格扫描，而是直接在 8 维解空间内进行全局寻优。算法通过监测种群的进化状态，动态调整搜索策略，平衡全局探索与局部开发能力.

改进粒子群算法设计

- 粒子编码与约束处理：

  虽然决策变量定义为绝对时刻，但在算法实现中，为了自动满足间隔约束，对时间参数采用增量编码 $(\Delta t_{12}, \Delta t_{23})$ 进行搜索，解码时再转换为绝对时间 t_{drop}。

- 灾变重启机制：
  针对多波次协同容易陷入局部最优的问题，引入" 灾变" 策略
  停滞检测：设置停滞计数器 stagnation_counter，若全局最优解 G_{best} 连续一定代数未更新，则判定算法陷入停滞。
  灾变操作：触发灾变，保留种群中前 10% 的精英粒子（历史最优个体），对其余 90% 的粒子进行完全重置（位置与速度重新随机化）。该机制能有效打破种群的聚集状态，跳出局部极值陷阱，寻找更优的多弹协同策略。

- 适应度函数：对时间轴进行离散化扫描，计算每一时刻系统总遮蔽状态 $I_{sys}(t)$，积分得到总有效时长，取负值作为适应度函数 $F(\mathbf{X}) = -T_{cover}$ 进行极小化求解.

6.3.6 求解结果

根据上述的流程，我们采用灾变机制的改进粒子群算法得到无人机 FY1 航向方向角为 180.67°, 速度大小为 95m/s 时, 对应的最大有效遮挡时间是 12.54241s。投放 3 枚烟幕弹干扰导弹 M1 的最优策略见下图6，三枚烟幕弹的具体投放策略见下表**??**。

表 2: Problem 3 策略表

烟幕弹序号	投放坐标 (m)			起爆坐标 (m)		
	X	Y	Z	X	Y	Z
1	17800	0	1800	17800	0	1800
2	17915.99463	13.80460	1800	17915.99463	13.80460	1800
3	22048.21768	505.58318	1800	23913.0268	727.51534	533.54687

烟幕弹序号	有效干扰时长 (s)	生效时间 (s)	失效时间 (s)
1	2.55600	0.79116	5.41117
2	4.37985	1.00000	5.07793
3	0.00000	0.00000	0.00000

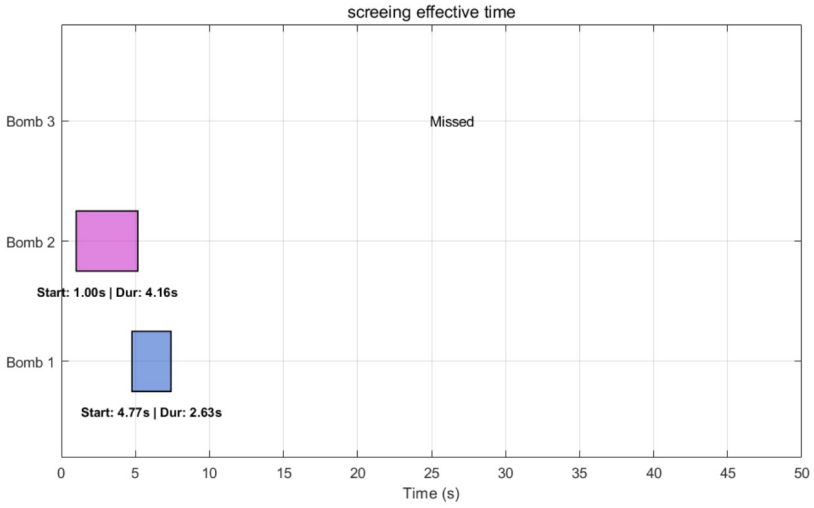


图 5: Problem 3 结果

6.4 问题四求解：多机协同优化

在问题四中，任务变更为 FY_1 FY_2 FY_3 三架无人机共同拦截导弹 $M1$，每架无人机投放一枚烟幕弹，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，使得烟幕弹的遮蔽时间最大.

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：

$$\mathbf{u}_i = [v_i, \alpha_i, t_{drop,i}, t_{delay,i}] \tag{43}$$

全局决策向量 X 维 12 维向量：

$$\mathbf{X} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] = [v1, 1, tdrop, 1, tdelay, 1, ..., v3, 3, tdrop, 3, tdelay, 3]^T \tag{44}$$

6.4.2 状态方程

各无人机的初始位置 $\mathbf{Pos}_{UAV,i}$ 不同，第 i 架无人机投放的烟幕弹在 t 刻的中心位置 $P_{S,i}(t)$ 更新公式如下投放点坐标：

$$P_{drop,i} = \mathbf{Pos}_{UAV,i} + \mathbf{v}_i(v_i, \alpha_i) \cdot t_{drop,i} \tag{45}$$

起爆点坐标

$$P_{burst,i} = P_{drop,i} + \mathbf{v}_i \cdot t_{delay,i} + \mathbf{G}(t_{delay,i}) \tag{46}$$

其中 $\mathbf{G}(t) = [0, 0, -\frac{1}{2}gt^2]^T$ 为重力项。
烟幕云团中心轨迹：

$$P_{S,i}(t) = P_{burst,i} - [0, 0, v_{sink} \cdot (t - t_{burst,i})]^T \tag{47}$$

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 个烟幕云团对目标的遮蔽指示函数 $I_i(t)$:

$$I_i(t; \mathbf{X}) = \begin{cases} 1, & \text{if } \operatorname{dist}(P_{S,i}(t), \operatorname{Line}_{M1-Target}) \leq R_{cloud} \\ 0, & otherwise \end{cases} \tag{48}$$

系统总遮蔽状态为各子状态的逻辑或：

$$Isys(t) = \max_{i=1,2,3} I_i(t) \tag{49}$$

最终优化目标函数定义为最小化代价 $J(\mathbf{X})$：

$$\min J(\mathbf{X}) = -\left(\int_0^{T_{end}} I_{sys}(t) dt \right) + \lambda \cdot \sum_{i=1}^{3} \max(d_{min,i}, R_{cloud}) \tag{50}$$

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：

$$70 \leq v_i \leq 140, \quad \forall i \tag{51}$$

时间窗约束：我们通过物理知识可知，遮蔽的理想状态是烟幕弹之间无缝接力，所以设置时间窗口约束，减小搜索空间：

$$t_{drop,1} \leq 15, \quad t_{drop,2} \leq 35, \quad t_{drop,3} \leq 55 \tag{52}$$

起爆延时约束：

$$1 \leq t_{delay,i} \leq 8 \tag{53}$$

高度约束：

$$z_{burst,i}(\mathbf{X}) \geq 0, \quad i = 1, 2, 3 \tag{54}$$

6.4.5 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，所以第四问我们选择鲁棒性强的差分进化算法（DE）求解

算法流程设计

- 种群初始化：生成 NP=200 个个体。针对航向角 α_i，基于各无人机到目标的初始视线角 $\theta_{base,i}$ 进行正态分布初始化，范围控制在 $\theta_{base,i} \pm 45°$ 内，提高搜索效率
- 变异: 采用 DE/rand/1 策略生成变异向量 $\mathbf{V}_g$：

  $$\mathbf{V}_{i,g} = \mathbf{X}_{r1,g} + F \cdot (\mathbf{X}_{r2,g} - \mathbf{X}_{r3,g}) \tag{55}$$

  其中缩放因子 F 采用自适应策略，随迭代次数从 0.5 线性衰减，平衡全局探索与局部开发。
- 交叉: 对变异向量与目标向量进行二项式交叉，生成试验向量 $\mathbf{U}_{i,g}$。交叉概率 $CR = 0.9$。
- 选择: 计算试验向量与目标向量的适应度，选择适应度更优的个体进入下一代。采用贪婪策略，若试验向量的适应度优于目标向量，则在下一代中保留试验向量：

  $$\mathbf{X}_{i,g+1} = \begin{cases} \mathbf{U}_{i,g}, & \text{if } J(\mathbf{U}_{i,g}) < J(\mathbf{X}_{i,g}) \\ \mathbf{X}_{i,g}, & \text{otherwise} \end{cases} \tag{56}$$

- 终止条件: 最大迭代次数 $G_{max} = 800$ 或适应度不再显著提升

6.4.6 求解结果

我们通过差分进化算法（DE）用 matlab 求解，效果如下图7所示，得到了最大的总遮蔽有效时间 15s，并且烟幕弹的遮蔽时间衔接良好，能够提供较好的遮蔽效果。对应的三机协同投放策略如下表3所示。

表 3: 问题 4 三机协同立体封锁最终策略（坐标精简版）

无人机	航向 (°)	T_{cover}	投放点 (m) (X,Y,Z)	起爆点 (m) (X,Y,Z)
FY1	78.65	4.65	(17747.32, 1.45, 1800)	(17521.68, 7.68, 1760)
FY2	119.78	4.82	(11492.56, 779.81, 1400)	(10897.90, 53.01, 1099)
FY3	135.11	5.46	(5291.39, −673.54, 700)	(5051.65, 113.58, 518)

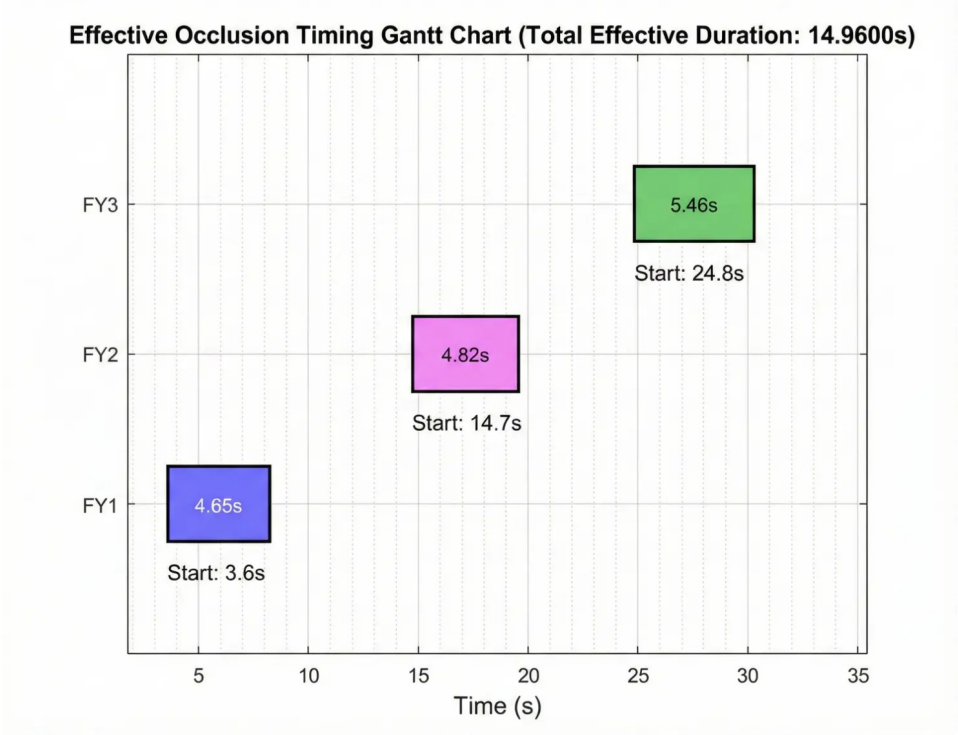


图 6: 问题 4 三机协同立体封锁结果

6.5 问题五求解：多机多目标协同拦截

在问题五中，作战场景扩展为 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。问题的核心在于如何在复杂的空间分布下，合理分配无人机的飞行策略和弹药资源，使得对所有导弹的总有效遮蔽时间最大化。

6.5.1 决策变量体系

考虑到问题规模显著扩大，且受到前几问优化算法的启发，我们将决策变量分为战略层（飞行参数）和战术层（投弹参数）。战略决策变量：定义全局飞行决策向量，包含

5 架无人机的速度和航向：

$$X_{fly} = [v_1, \alpha_1, v_2, \alpha_2, \ldots, v_5, \alpha_5]^T \tag{57}$$

战术决策变量：对于第 i 架无人机，其携带的第 k 枚烟幕弹 $k = 1, 2, 3$ 的投放参数由算法在内层生成，不直接作为外层优化变量，但需满足物理约束：

$$u_{i,k} = \left[t_{drop}^{(i,k)}, t_{delay}^{(i,k)} \right] \tag{58}$$

6.5.2 状态空间

多导弹运动模型的建立参考第一问，因为每个导弹的飞行特性为一致，只有初始位置不同。多无人机运动模型亦可由之前类比得到。

6.5.3 目标函数

由于存在 3 个独立的来袭目标，系统总效益定义为所有导弹被有效遮蔽时长的总和：

$$S_j(t) = \bigvee_{i=1}^{5} \bigvee_{k=1}^{3} I_{i,k,j}(t) \tag{59}$$

最终优化目标为最大化总效益 J：

$$\max J(X_{fly}) = \sum_{j=1}^{3} \left(\int_0^{T_{end}} S_j(t) dt \right) \tag{60}$$

6.5.4 约束条件

最大载弹量约束：每架无人机投放数量 N<=3；
投弹间隔约束：

$$|t_{drop}^{(i,k)} -_{drop}^{(i,k-1)}| \geq 1, \forall k > 1 \tag{61}$$

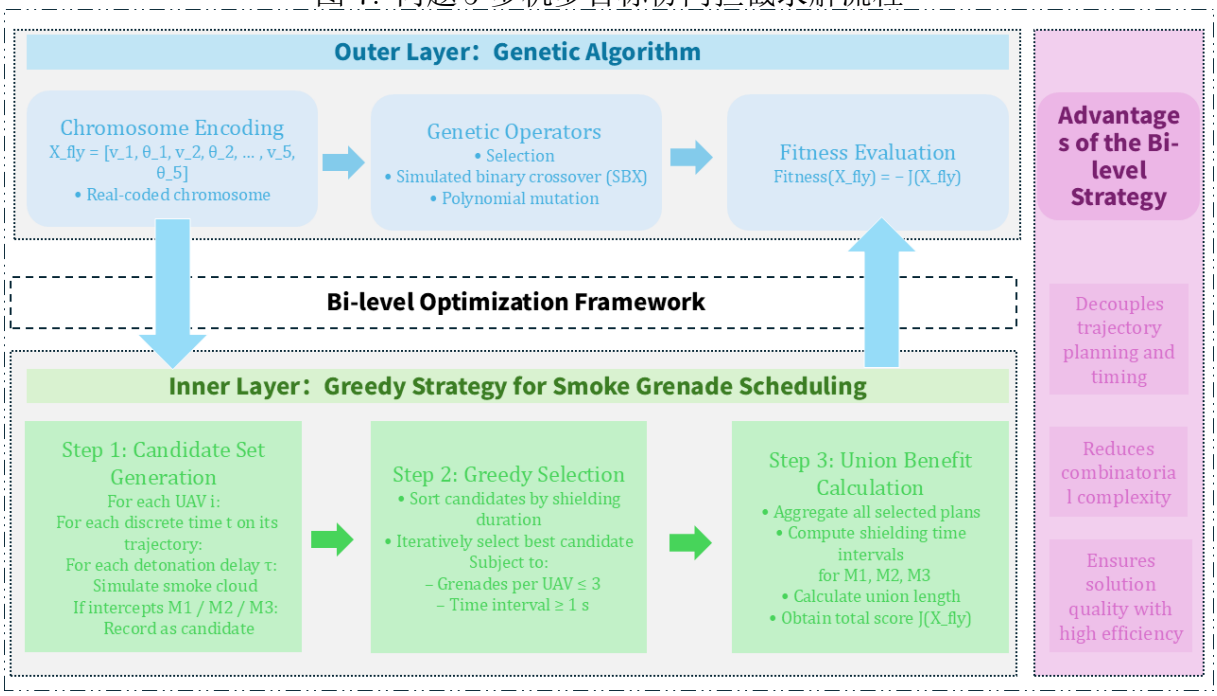
飞行边界约束：

$$70 \leq v_i \leq 140, 0 \leq \alpha_i \leq 2\pi \tag{62}$$

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8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 7: 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。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。实现编码 X_{fly}，则适应度函数为：

$$Fitness\left(X_{fly}\right) = -J\left(X_{fly}\right) \tag{63}$$

- 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

  遍历第 i 架无人机飞行路径上的离散时间点 t，对于每一个 t，遍历可能的延时起爆时间，检测生成的烟幕云团能否拦截 M1, M2, M3 中的任意一枚。若能拦截，记录该候选方案。

  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

  汇总所有无人机的投弹方案，分别计算对 M1, M2, M3 的遮蔽时间区间并集，求和得到总分。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

通过该算法求解，得到 5 机 3 弹协同拦截 3 枚导弹的最优策略如下表6.5.6所示，总有效遮蔽时间为 28.763s，效果如图9所示。

表 4: 五机协同的详细方案

Drone	BombID	DropTime (s)	ExpTime (s)	Target	CoverDuration (s)	Interval
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	M	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]

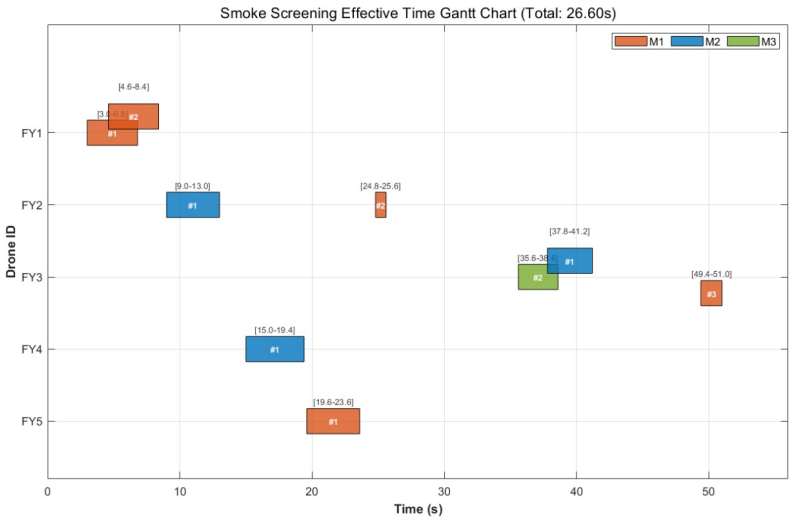


图 8: 五机协同的最优方案

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导弹运动影响，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] 作者. 题目. 期刊名, 年份, 卷 (期): 页码.

[2] 张三. 无人机协同作战理论. 北京: 科学出版社, 2020.

[3] CUMCM 组委会. 202x 年全国大学生数学建模竞赛赛题.

附录：主要代码

MATLAB 轨迹仿真代码

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;
```

```
115
116 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
```

```
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
```

```
36
37      %% 3. 并行扫描循环
38      parfor i = 1 : num_scan
39          v_curr = v_scan_list(i);
40
41          % --- PSO 配置 ---
42          Vec_Base = Env.Pos_TrueTarget - Env.Pos_FY1_Init;
43          Base_Angle = rad2deg(atan2(Vec_Base(2), Vec_Base(1)));
44
45          LB = [Base_Angle-30, 0, 1.0];
46          UB = [Base_Angle+30, 12, 8.0];
47
48          [best_x, best_val] = Run_Micro_PSO(v_curr, LB, UB, Env);
49
50          results_score(i) = best_val;
51          results_params(i, :) = best_x;
52      end
53
54      %% 4. 提取最优结果
55      [max_score, idx_best] = max(results_score);
56      best_v = v_scan_list(idx_best);
57      best_p = results_params(idx_best, :);
58
59      fprintf('>>> 计算完成\n');
60      fprintf('>>> 冠军速度: %.1f m/s\n', best_v);
61      fprintf('>>> 极限遮蔽: %.5f 秒\n', max_score);
62
63      %% 5. 绘图
64      fprintf('\n>>> 正在生成时序甘特图...\n');
65
66      [t_start_opt, t_end_opt, duration_opt] = Recompute_Time_Series(
            best_v, best_p, Env);
67
68      if duration_opt > 0
69          figure('Color', 'w', 'Position', [100, 100, 600, 600], 'Name'
                , 'Simulated Gantt Chart');
70          hold on;
71
72          bar_height = 0.6;
73          fy_index = 1;
```

```
74          color_fill = [100, 100, 255]/255;
75
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
110     ang_rad = deg2rad(a);
111     Vel_Vec = [cos(ang_rad), sin(ang_rad), 0] * v;
112     P_Drop = Env.Pos_FY1_Init + Vel_Vec * td;
113     P_Pop = P_Drop + Vel_Vec * ty;
114     P_Pop(3) = P_Pop(3) - 0.5 * Env.g * ty^2;
115
116     t_pop = td + ty;
117     t_max_sim = Env.Dist_Total_M1 / Env.V_M1;
118     t_vec = 0 : dt : t_max_sim;
119
120     valid_mask = false(size(t_vec));
121
122     % 向量化检查每一时刻
123     for k = 1:length(t_vec)
124         t_curr = t_vec(k);
125
126         % 1. 导弹位置
127         if t_curr * Env.V_M1 >= Env.Dist_Total_M1
128             P_M = Env.Pos_FakeTarget;
129         else
130             P_M = Env.Pos_M1_Init + Env.Dir_M1 * (Env.V_M1 * t_curr);
131         end
132
133         % 2. 烟雾位置
134         if t_curr >= t_pop && (t_curr - t_pop) <= Env.Time_smoke_last
135             t_rel = t_curr - t_pop;
136             P_Smk = P_Pop - [0, 0, Env.V_smoke_sink] * t_rel;
137
138             % 3. 遮挡判定
139             P1 = P_M;
140             P2 = Env.Pos_TrueTarget;
141             Q = P_Smk;
142
143             v_vec = P2 - P1;
144             w_vec = Q - P1;
145
146             c1 = dot(w_vec, v_vec);
147             c2 = dot(v_vec, v_vec);
```

```
148
149             if c1 <= 0
150                 dist = norm(Q - P1);
151             elseif c2 <= c1
152                 dist = norm(Q - P2);
153             else
154                 b = c1 / c2;
155                 Pb = P1 + b * v_vec;
156                 dist = norm(Q - Pb);
157             end
158
159             if dist <= Env.R_smoke
160                 valid_mask(k) = true;
161             end
162         end
163     end
164
165     % 提取结果
166     if any(valid_mask)
167         idx = find(valid_mask);
168         t_start = t_vec(idx(1));
169         t_end = t_vec(idx(end));
170         valid_dur = sum(valid_mask) * dt;
171     else
172         t_start = NaN; t_end = NaN; valid_dur = 0;
173     end
174 end
175
176 %% --- 内部微型 PSO 求解器 ---
177 function [best_pos, best_val] = Run_Micro_PSO(v, lb, ub, Env)
178     % 粒子群参数
179     n_part = 30;
180     n_iter = 50;
181     w = 0.6; c1 = 1.5; c2 = 1.5;
182     n_vars = 3;
183
184     % 初始化
185     pos = repmat(lb, n_part, 1) + rand(n_part, n_vars) .* repmat(ub-
            lb, n_part, 1);
186
```

```
    % [种子注入]
    if v < 90
        pos(1,:) = [176.88, 0.01, 2.5];
    else
        pos(1,:) = [178.46, 0.01, 3.3];
    end

    vel = zeros(n_part, n_vars);
    pbest_pos = pos;
    pbest_val = zeros(n_part, 1);
    gbest_pos = zeros(1, n_vars);
    gbest_val = -1e9;

    % 评估初始种群
    for i = 1:n_part
        val = -Tactical_Sim_Engine(v, pos(i,1), pos(i,2), pos(i,3),
           Env.dt, Env);
        pbest_val(i) = val;
        if val > gbest_val
            gbest_val = val;
            gbest_pos = pos(i,:);
        end
    end

    % 迭代
    for t = 1:n_iter
        for i = 1:n_part
            r1 = rand(1, n_vars); r2 = rand(1, n_vars);
            vel(i,:) = w*vel(i,:) + c1*r1.*(pbest_pos(i,:)-pos(i,:))
               + c2*r2.*(gbest_pos-pos(i,:));
            pos(i,:) = pos(i,:) + vel(i,:);
            pos(i,:) = max(pos(i,:), lb);
            pos(i,:) = min(pos(i,:), ub);

            val = -Tactical_Sim_Engine(v, pos(i,1), pos(i,2), pos(i
               ,3), Env.dt, Env);

            if val > pbest_val(i)
                pbest_val(i) = val;
                pbest_pos(i,:) = pos(i,:);
```

```
            end
            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
    idx_more = b > 1; if any(idx_more), Pb(:, idx_more) = repmat(Env.
        Pos_TrueTarget', 1, sum(idx_more)); end

    dists_sq = sum((P_Smk_mat - Pb).^2, 1);
    count = sum(dists_sq <= Env.R_smoke^2);
    score = -(count * dt);
end
```

Listing 3: Problem 3

```
clc; clear; close all;

%% 1. 参数初始化
n_particles = 200;       % 增加粒子数以覆盖更大的搜索空间(8维)
n_iterations = 300;      % 增加迭代次数
c1 = 2.0; c2 = 2.0;
w_max = 0.9; w_min = 0.4;

lb = [70,  0,    0,  1,  1,  0,  0,  0];
ub = [140, 2*pi, 60, 20, 20, 15, 15, 15];

% 速度限制
v_max = 0.15 * (ub - lb);
v_min = -v_max;

%% 2. 种群初始化
particles = repmat(lb, n_particles, 1) + rand(n_particles, 8) .* (ub
    - lb);
velocities = zeros(n_particles, 8);

pbest_pos = particles;
pbest_val = inf(n_particles, 1);
gbest_pos = zeros(1, 8);
gbest_val = inf;

stagnation_counter = 0; % 停滞计数器，用于判断是否陷入局部最优

%% 3. PSO 主循环
```

```
disp('开始多烟幕弹协同优化␣(引入灾变机制防止局部最优)...');
tic;

for iter = 1:n_iterations
    w = w_max - (w_max - w_min) * iter / n_iterations;

    current_gbest_improved = false;

    for i = 1:n_particles
        current_x = particles(i, :);
        current_x(1) = round(current_x(1)); % 速度取整

        % 边界处理
        current_x = max(current_x, lb);
        current_x = min(current_x, ub);

        % 计算适应度
        fitness = calculate_fitness_multi(current_x);

        % 更新个体最优
        if fitness < pbest_val(i)
            pbest_val(i) = fitness;
            pbest_pos(i, :) = current_x;
        end

        % 更新全局最优
        if fitness < gbest_val
            gbest_val = fitness;
            gbest_pos = current_x;
            current_gbest_improved = true;
        end
    end

    % --- 灾变机制 ---
    if current_gbest_improved
        stagnation_counter = 0;
    else
        stagnation_counter = stagnation_counter + 1;
    end

```

```
    % 如果连续20代没有进步，重置部分粒子
    if stagnation_counter > 20 && iter < n_iterations - 50
        disp(['␣␣␣␣>>␣检测到局部最优停滞，触发灾变重置␣(Iter␣'
           num2str(iter) ')']);
        % 保留前 10% 的精英
        n_keep = round(n_particles * 0.1);
        [~, sorted_idx] = sort(pbest_val);
        for k = n_keep+1 : n_particles
            idx = sorted_idx(k);
            particles(idx, :) = lb + rand(1, 8) .* (ub - lb);
            velocities(idx, :) = zeros(1, 8);
            pbest_val(idx) = inf; % 重置历史记忆
        end
        stagnation_counter = 0;
    end

    % 粒子更新公式
    for i = 1:n_particles
        r1 = rand(1, 8);
        r2 = rand(1, 8);

        velocities(i,:) = w * velocities(i,:) ...
            + c1 * r1 .* (pbest_pos(i,:) - particles(i,:)) ...
            + c2 * r2 .* (gbest_pos - particles(i,:));

        velocities(i,:) = max(min(velocities(i,:), v_max), v_min);
        particles(i,:) = particles(i,:) + velocities(i,:);

        % 变异操作 (5%概率)
        if rand < 0.05
            dim = randi(8);
            particles(i, dim) = lb(dim) + rand * (ub(dim) - lb(dim));
        end
        particles(i,:) = max(min(particles(i,:), ub), lb);
    end

    % 进度显示
    if mod(iter, 10) == 0 || iter == 1
        real_time = 0;
        if gbest_val < -1000, real_time = -(gbest_val + 10000); end
```

```
107         fprintf('迭代␣%d/%d,␣当前最佳遮蔽时长:␣%.5f␣秒␣(停滞:␣%d)\n',
                ...
108             iter, n_iterations, real_time, stagnation_counter);
109     end
110 end
111 toc;
112
113 %% 4. 结果解析与输出
114 best_params = gbest_pos;
115 best_params(1) = round(best_params(1));
116
117 % 解码时间参数
118 t_drop1 = best_params(3);
119 t_drop2 = t_drop1 + best_params(4);
120 t_drop3 = t_drop2 + best_params(5);
121 t_delay1 = best_params(6);
122 t_delay2 = best_params(7);
123 t_delay3 = best_params(8);
124
125 % 最终高精度验证
126 final_time = calculate_masking_pure_multi(best_params, 0.00001);
127
128 fprintf('\n----------------␣最终优化结果␣(Problem␣3)␣
        ----------------\n');
129 fprintf('最大有效遮蔽时长:␣%.5f␣秒\n', final_time);
130 fprintf('\n策略详情: \n');
131 fprintf('无人机速度:␣%d␣m/s\n', best_params(1));
132 fprintf('无人机航向:␣%.4f␣rad␣(%.2f␣度)\n', best_params(2), rad2deg(
        best_params(2)));
133 fprintf('\n[烟弹1]␣投放时刻:␣%6.4f␣s,␣起爆延时:␣%6.4f␣s,␣起爆时刻:␣
        %6.4f␣s\n', t_drop1, t_delay1, t_drop1+t_delay1);
134 fprintf('[烟弹2]␣投放时刻:␣%6.4f␣s,␣起爆延时:␣%6.4f␣s,␣起爆时刻:␣%6.4
        f␣s\n', t_drop2, t_delay2, t_drop2+t_delay2);
135 fprintf('[烟弹3]␣投放时刻:␣%6.4f␣s,␣起爆延时:␣%6.4f␣s,␣起爆时刻:␣%6.4
        f␣s\n', t_drop3, t_delay3, t_drop3+t_delay3);
136 fprintf('注: 投放间隔分别为␣%.4f␣s␣和␣%.4f␣s␣(均满足>=1s约束)\n',
        best_params(4), best_params(5));
137
138
139 %% ================= 子函数定义区域 =================
```

```
140
141 function fitness = calculate_fitness_multi(x)
142     % 1. 解码所有参数
143     v = x(1); theta = x(2);
144     t_d1 = x(3);
145     t_d2 = t_d1 + x(4); % 确保间隔
146     t_d3 = t_d2 + x(5); % 确保间隔
147     del1 = x(6); del2 = x(7); del3 = x(8);
148
149     % 2. 计算3枚弹的运动学参数
150     [init_pos_m1, v_vec_m1, target_pos] = get_env_params(v, theta);
151
152     [exp1, pos1] = get_bomb_kinematics(v, theta, t_d1, del1);
153     [exp2, pos2] = get_bomb_kinematics(v, theta, t_d2, del2);
154     [exp3, pos3] = get_bomb_kinematics(v, theta, t_d3, del3);
155
156     % 物理约束：任何一枚在地下爆炸都给予惩罚
157     if pos1(3)<0 || pos2(3)<0 || pos3(3)<0
158         fitness = 1e6; return;
159     end
160
161     % 3. 确定仿真时间范围
162     smoke_dur = 20;
163     t_hit = norm(init_pos_m1) / 300;
164
165     % 整个遮蔽过程是3个区间的并集，我们取最早起爆到最晚结束
166     t_start_global = min([exp1, exp2, exp3]);
167     t_end_global = min(max([exp1, exp2, exp3]) + smoke_dur, t_hit);
168
169     if t_start_global >= t_end_global
170         fitness = 1e5; return;
171     end
172
173     % --- 智能扫描策略 ---
174
175     % A. 粗扫描 (0.1s)
176     dt_coarse = 0.1;
177     T_coarse = t_start_global : dt_coarse : t_end_global;
178     if isempty(T_coarse), fitness = 1e5; return; end
179
```

```
    min_dist_global = inf;
    potential_times = [];

    % 预计算导弹位置
    Pm_c = init_pos_m1 + v_vec_m1 .* T_coarse(:);
    Pt = target_pos;

    for k = 1:length(T_coarse)
        t = T_coarse(k);
        pm = Pm_c(k,:);
        vec_line = Pt - pm;
        norm_line = norm(vec_line);

        dist_min_k = inf;

        % 检查3个烟雾团
        clouds = [exp1, pos1; exp2, pos2; exp3, pos3]; % 3x4 matrix
        for m = 1:3
            t_exp = clouds(m, 1);
            p_exp = clouds(m, 2:4);

            % 只有在烟雾存续期内才计算
            if t >= t_exp && t <= t_exp + smoke_dur
                ps = p_exp;
                ps(3) = ps(3) - 3 * (t - t_exp); % 下沉

                vec_point = ps - pm;
                d = norm(cross(vec_point, vec_line)) / norm_line;
                if d < dist_min_k, dist_min_k = d; end
            end
        end

        if dist_min_k < min_dist_global, min_dist_global = dist_min_k
            ; end
        if dist_min_k < 18, potential_times = [potential_times; t];
            end
    end

    % B. 精细计算
    if isempty(potential_times)
```

```
        fitness = min_dist_global;
    else
        dt_fine = 0.0001;
        window = 0.06;
        all_fine_times = [];

        for t_center = potential_times'
            t_sub = (max(t_start_global, t_center - window) : dt_fine
                : min(t_end_global, t_center + window))';
            t_sub = round(t_sub / dt_fine) * dt_fine; % 网格对齐
            all_fine_times = [all_fine_times; t_sub];
        end

        if isempty(all_fine_times), fitness = min_dist_global; return
           ; end
        T_fine = unique(all_fine_times);

        Pm_f = init_pos_m1 + v_vec_m1 .* T_fine;

        mask_count = 0;

        % 向量化比较麻烦，这里用循环处理这3个球
        % 为了加速，把3个球的信息展开
        exp_times = [exp1, exp2, exp3];
        pos_starts = [pos1; pos2; pos3];

        for j = 1:length(T_fine)
            t = T_fine(j);
            pm = Pm_f(j,:);
            vec_line = Pt - pm;
            norm_line = norm(vec_line);

            is_masked = false;

            % 遍历3个球
            for m = 1:3
                if t >= exp_times(m) && t <= exp_times(m) + smoke_dur
                    ps = pos_starts(m, :);
                    ps(3) = ps(3) - 3 * (t - exp_times(m));

```

```
                    vec_point = ps - pm;
                    % 快速判断距离平方
                    cp = cross(vec_point, vec_line);
                    d2 = sum(cp.^2) / (norm_line^2);

                    if d2 <= 100 % 10^2
                        is_masked = true;
                        break; % 只要被任意一个遮住就算遮住
                    end
                end
            end

            if is_masked
                mask_count = mask_count + 1;
            end
        end

        fitness = -(mask_count * dt_fine) - 10000;
    end
end

function mask_time = calculate_masking_pure_multi(x, dt)
    % 解码
    v = x(1); theta = x(2);
    t_d1 = x(3); t_d2 = t_d1 + x(4); t_d3 = t_d2 + x(5);
    del1 = x(6); del2 = x(7); del3 = x(8);

    [init_pos_m1, v_vec_m1, target_pos] = get_env_params(v, theta);
    [exp1, pos1] = get_bomb_kinematics(v, theta, t_d1, del1);
    [exp2, pos2] = get_bomb_kinematics(v, theta, t_d2, del2);
    [exp3, pos3] = get_bomb_kinematics(v, theta, t_d3, del3);

    smoke_dur = 20;
    t_hit = norm(init_pos_m1) / 300;
    t_start = min([exp1, exp2, exp3]);
    t_end = min(max([exp1, exp2, exp3]) + smoke_dur, t_hit);

    if t_start >= t_end, mask_time=0; return; end

    T = t_start : dt : t_end;
```

```
    count = 0;

    exp_times = [exp1, exp2, exp3];
    pos_starts = [pos1; pos2; pos3];

    for k = 1:length(T)
        t = T(k);
        pm = init_pos_m1 + v_vec_m1 * t;
        vec_line = target_pos - pm;
        norm_line = norm(vec_line);

        masked = false;
        for m = 1:3
            if t >= exp_times(m) && t <= exp_times(m) + smoke_dur
                ps = pos_starts(m, :);
                ps(3) = ps(3) - 3 * (t - exp_times(m));

                vec_point = ps - pm;
                d = norm(cross(vec_point, vec_line)) / norm_line;
                if d <= 10
                    masked = true; break;
                end
            end
        end
        if masked, count = count + 1; end
    end
    mask_time = count * dt;
end

function [init_pos_m1, v_vec_m1, target_pos] = get_env_params(v,
   theta)
    init_pos_m1 = [20000, 0, 2000];
    fake_target = [0, 0, 0];
    dir_m1 = (fake_target - init_pos_m1) / norm(fake_target -
       init_pos_m1);
    v_vec_m1 = dir_m1 * 300;
    target_pos = [0, 200, 0];
end

function [t_explode, explode_pos] = get_bomb_kinematics(v_mag, theta,
```

```
    t_drop, t_delay)
    init_pos_uav = [17800, 0, 1800];
    v_vec_uav = [v_mag * cos(theta), v_mag * sin(theta), 0];
    drop_pos = init_pos_uav + v_vec_uav * t_drop;

    g = 9.8;
    explode_pos = drop_pos;
    explode_pos(1) = drop_pos(1) + v_vec_uav(1) * t_delay;
    explode_pos(2) = drop_pos(2) + v_vec_uav(2) * t_delay;
    explode_pos(3) = drop_pos(3) - 0.5 * g * t_delay^2;

    t_explode = t_drop + t_delay;
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));
```

```
26
27      % 变量顺序: [V, Ang, T_drop, T_delay] * 3架
28      % 速度约束 70-140
29      LB = [70, Base_Ang1-45, 0, 1,   70, Base_Ang2-45, 0, 1,   70,
            Base_Ang3-45, 0, 1];
30      UB = [140, Base_Ang1+45, 15, 8, 140, Base_Ang2+45, 35, 8, 140,
            Base_Ang3+45, 55, 8];
31
32      de_opts.NP = 200;
33      de_opts.MaxIter = 800;
34      de_opts.F = 0.5;
35      de_opts.CR = 0.9;
36
37      fprintf('=========================================================\n
            ');
38      fprintf('␣␣␣␣␣␣问题4: 三机协同立体封锁\n');
39      fprintf('=========================================================\n
            ');
40
41      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
74  function Intervals = Calculate_Exact_Intervals(Res, Env)
75      % 初始化
76      Intervals = repmat(struct('is_effective', false, 'start_time',
            NaN, 'end_time', NaN), 3, 1);
77
78      dt = 0.01;
79      T_max = Env.Dist_M1 / Env.V_M1 + 5;
80      t_vec = 0 : dt : T_max;
81
82      % 预计算每架飞机的烟雾位置参数
83      SmokeData = [];
84      for i=1:3
85          ang = deg2rad(Res(i).a);
86          V = [cos(ang), sin(ang), 0] * Res(i).v;
87          P_Drop = Env.Pos_UAVs(i,:) + V * Res(i).td;
88          P_Pop_Init = P_Drop + V * Res(i).ty;
89          P_Pop_Init(3) = P_Pop_Init(3) - 0.5 * 9.8 * Res(i).ty^2;
90          SmokeData(i).P_Pop_Init = P_Pop_Init;
91          SmokeData(i).tp = Res(i).tp;
92      end
```

```
93
94      % 逐帧检测遮挡情况
95      for i = 1:3
96          mask = false(size(t_vec));
97          for k = 1:length(t_vec)
98              t = t_vec(k);
99              % 1. 导弹当前位置
100             if t * Env.V_M1 > Env.Dist_M1
101                 P_M = Env.Pos_True; % 已到达
102             else
103                 P_M = Env.Pos_M1 + Env.Dir_M1 * (Env.V_M1 * t);
104             end
105
106             % 2. 判断该飞机的烟雾是否存在
107             if t >= SmokeData(i).tp && t <= SmokeData(i).tp + Env.
                    T_last
108                 t_rel = t - SmokeData(i).tp;
109                 P_Smk = SmokeData(i).P_Pop_Init - [0, 0, Env.V_sink *
                        t_rel];
110
111                 % 3. 判断是否遮挡
112                 v_los = Env.Pos_True - P_M;
113                 w_vec = P_Smk - P_M;
114                 c1 = dot(w_vec, v_los);
115                 c2 = dot(v_los, v_los);
116                 if c1 > 0
117                     b = c1 / c2;
118                     Pb = P_M + v_los * b;
119                     dist_sq = sum((P_Smk - Pb).^2);
120                     if dist_sq <= Env.R_smk^2
121                         mask(k) = true;
122                     end
123                 end
124             end
125         end
126
127         % 提取起止时间
128         if any(mask)
129             idx = find(mask);
130             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
    end
    P_Pop = zeros(3, 3); Times_Pop = zeros(3, 1);
    for i = 1:3
        ang_rad = deg2rad(Res(i).a);
        Vel_Vec = [cos(ang_rad), sin(ang_rad), 0] * Res(i).v;
        P_Drop = Env.Pos_UAVs(i,:) + Vel_Vec * Res(i).td;
        Disp = Vel_Vec * Res(i).ty; Disp(3) = Disp(3) - 0.5 * 9.8 *
           Res(i).ty^2;
        P_Pop(i,:) = P_Drop + Disp; Times_Pop(i) = Res(i).tp;
        if P_Pop(i,3) < 0, score = 1e5; pure_time = 0; return; end
    end
    t_start = min(Times_Pop); t_end = min(Env.Dist_M1/Env.V_M1, max(
       Times_Pop) + Env.T_last);
    if t_start >= t_end, score = 1e5; pure_time = 0; return; end
    Target_Points = [0, 200, 5; 0, 200, 10; 0, 200, 0; -7, 200, 5; 7,
        200, 5];
    num_pts = 5; dt = 0.05; t_vec = t_start : dt : t_end;
    total_coverage = 0; penalty_dist = 0; min_dists = [1e9, 1e9, 1e9
       ];
```

```
    for t = t_vec
        P_M = Env.Pos_M1 + Env.Dir_M1 * (Env.V_M1 * t);
           blocked_pts_count = 0;
        for p = 1:num_pts
            TP = Target_Points(p,:); v_los = TP - P_M; len_los_sq =
               sum(v_los.^2); is_pt_blocked = false;
            for k = 1:3
                if t >= Times_Pop(k) && t <= Times_Pop(k) + Env.
                   T_last
                    P_Smk = P_Pop(k,:) - [0, 0, Env.V_sink * (t -
                       Times_Pop(k))];
                    w_vec = P_Smk - P_M; c1 = dot(w_vec, v_los);
                    if c1 > 0
                        b = max(0, min(1, c1 / len_los_sq)); Pb = P_M
                            + v_los * b;
                        d_sq = sum((P_Smk - Pb).^2); dist = sqrt(d_sq
                           );
                        if p == 1 && dist < min_dists(k), min_dists(k
                           ) = dist; end
                        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');
231     hold on;
232
233     colors = {
234         [100, 100, 255]/255,    % FY1 蓝
235         [238, 130, 238]/255,    % FY2 紫
236         [100, 200, 100]/255     % FY3 绿
237     };
238
239     % 计算精确区间
240     Intervals = Calculate_Exact_Intervals(Res, Env);
241     bar_height = 0.5;
242
243     for i = 1:3
244         % 只画有效遮蔽区间
245         if Intervals(i).is_effective
246             t_s = Intervals(i).start_time;
247             t_e = Intervals(i).end_time;
248
249             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');
```

```
        end
    end

    ylim([0, 4]);
    yticks([1, 2, 3]);
    yticklabels({'FY1', 'FY2', 'FY3'});

    valid_times = [Intervals.start_time, Intervals.end_time];
    valid_times = valid_times(~isnan(valid_times));
    if isempty(valid_times), xlim([0, 30]); else, xlim([min(
        valid_times)-2, max(valid_times)+5]); end

    xlabel('时间␣(s)', 'FontSize', 12);
    title(sprintf('三机协同有效遮蔽时序␣(Total:␣%.4fs)', score), '
        FontSize', 14);
    grid on; ax=gca; ax.GridAlpha=0.3; ax.XMinorGrid='on'; box on;
    hold off;
end

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');
```

```
85
86  total_M1 = 0; total_M2 = 0; total_M3 = 0;
87
88  for i = 1:5
89      drone_plan = final_plan{i};
90      if isempty(drone_plan)
91          continue;
92      end
93      for j = 1:size(drone_plan, 1)
94          % plan row: [t_drop, t_exp, missile_idx, cover_start,
                cover_end, score]
95          d_time = drone_plan(j, 1);
96          e_time = drone_plan(j, 2);
97          m_idx = drone_plan(j, 3);
98          c_dur = drone_plan(j, 5) - drone_plan(j, 4);
99
100         fprintf('FY%-4d | #%-5d | %-8.2f | %-8.2f | M%-7d | %.2f s
                [%.1f-%.1f]\n', ...
101              i, j, d_time, e_time, m_idx, c_dur, drone_plan(j,4),
                    drone_plan(j,5));
102     end
103 end
104 fprintf('=========================================\n');
105
106
107 %% ---------------------------------------------------------
108 %  Local Functions (无需单独文件, 直接包含在脚本中)
109 %  ---------------------------------------------------------
110
111 function [score, all_plans] = fitness_func(vars)
112     % 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)
113     % all_plans 用于最后输出详细信息
114
115     global Mis RealTarget
116
117     % 存储每枚导弹被遮蔽的时间段集合
118     % 结构: cell array {M1_intervals; M2_intervals; M3_intervals}
119     missile_covers = cell(3, 1);
120     all_plans = cell(5, 1);
121
```

```
    % 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
```

```
159     % 输出：针对M1-M3的遮蔽区间，以及具体的投弹计划
160
161     global Drones Mis g
162
163     P0 = Drones(drone_idx).P0;
164     intervals_by_missile = cell(3, 1);
165
166     % 1. 生成候选投弹列表
167     % 离散化遍历投弹时间 t_drop
168     % 无人机最长飞行时间估计：导弹最多飞 70秒，取 0-70s
169     t_drops = 0 : 1.0 : 70;
170     candidates = []; % 格式：[t_drop, t_exp, missile_idx, start_t,
            end_t, duration]
171
172     % 无人机速度向量
173     V_drone = [v * cos(ang), v * sin(ang), 0];
174
175     for t_d = t_drops
176         % 此刻无人机位置
177         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
```

```
                    continue;
                end

                % 检查该爆炸点产生的云团能否遮蔽导弹 m_id
                [is_block, t_start, t_end] = check_interception(
                    P_boom, t_e, m_id);

                if is_block
                    dur = t_end - t_start;
                    if dur > 0.1
                        % 记录候选策略
                        candidates = [candidates; t_d, t_e, m_id,
                            t_start, t_end, dur];
                    end
                end
            end
        end
    end

    % 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)
    % 检查单个云团对特定导弹的遮蔽情况
    global Mis RealTarget

    blocked = false; t_start = 0; t_end = 0;

    M_struct = Mis(m_idx);

    % 烟幕有效时间窗口
    cloud_life_start = t_exp;
    cloud_life_end = t_exp + 20;

    % 导弹飞行结束时间
    mis_end = M_struct.flight_time;

    % 实际有效检测时间段（取交集）
    check_start = max(0, cloud_life_start);
    check_end = min(mis_end, cloud_life_end);

    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 
```

```
function d = point_to_segment_dist(P, A, B)
    % 计算点 P 到线段 AB 的最短距离
    AB = B - A;
    AP = P - A;

    % 投影系数
    t = dot(AP, AB) / dot(AB, AB);

    if t < 0
        closest = A; % 投影在 A 外侧
    elseif t > 1
        closest = B; % 投影在 B 外侧
    else
        closest = A + t * AB; % 投影在线段上
    end

    d = norm(P - closest);
end

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
```

```
352     end
353
354     len = sum(merged(:, 2) - merged(:, 1));
355 end
```