Practice of Intelligent Algorithms in UAV Path Planning

Jialiang Lu
3230104505@zju.edu.cn
ISEE, ZJU

Qiran Xian
ISEE, ZJU

Chenxi Ji
ISEE, ZJU

Completed Grouply

in partial fulfilment for the course

Algorithm Design and Intelligent Computing

Information Science

College of Information Science & Electronic Engineering
Zhejiang University

2026 年 2 月 23 日

目录

一、 问题背景

路径规划是无人机（UAV）应用中的核心挑战，需要在路径优越性（如距离、能耗）与安全约束之间取得微妙平衡。虽然传统算法（A*、APF、RRT）和群智能技术（PSO、GA）已被广泛采用，但它们往往面临扩展性差的问题，或难以处理多个冲突目标之间的权衡。此外，现有的多目标优化方法通常忽略了无人机的运动学约束，导致生成的路径虽在理论上最优，但在实际中却无法飞行。针对上述局限性，我们尝试引入基于导航变量的多目标粒子群算法（NMOPSO，Duong, Bui, and Phung 2025）以及由灰狼算法（GWO，Mirjalili, Mirjalili, and Lewis 2014）优化而得的多目标灰狼算法（MOGWO）来求解该问题。我们分析了两种算法在该问题上的性能表现并进行了比较。

二、 问题建模

2.1 运动学模型

我们将问题放在 3 维笛卡尔坐标系中考虑，无人机任意时刻的位置记为：

$$\boldsymbol{r} = [x, y, z]^{\mathrm{T}}$$

如果我们记无人机的在某时刻的速率为 V，那么其运动方程为：

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = V\cos\alpha\cos\beta \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = V\cos\alpha\sin\beta \\ \dfrac{\mathrm{d}z}{\mathrm{d}t} = V\sin\alpha \end{cases}$$

式中 α 表示无人机在全局坐标系下的飞行路径角，即速度矢量与与水平面的夹角；β 表示在全局坐标系下无人机的航向角，即速度矢量在水平面内的投影与参考方向（x 轴）之间的夹角。无人机飞行的物理限制给出了上述参数应该满足的约束：

$$\begin{cases} V_{\min} \le V \le V_{\max} \\ |\Delta\alpha| = |\theta| \le \theta_{\max} \\ |\Delta\beta| = |\psi| \le \psi_{\max} \end{cases}$$

其中 $V_{\min}$ 和 $V_{\max}$ 分别表示无人机飞行时的最小与最大速率，$\theta_{\max}$ 和 $\psi_{\max}$ 分别表示两段航线间无人机转向时飞行路径角的最大改变量和航向角的最大改变量，即最大爬升角和最大转向角。

2.2 目标建模

我们引入如下四个目标函数，用于刻画四个目标：

2.2.1 路径长度

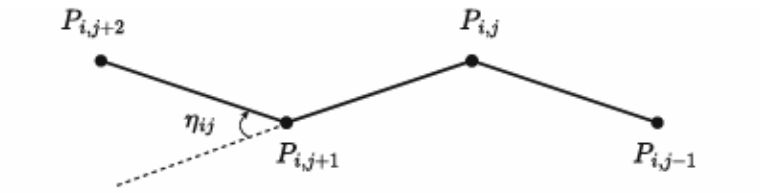

图 1: UAV 飞行路径示意

无人机的飞行路径被设定为从固定的起点 P_{i1} 出发,经过给定的 n 个途径点 $P_{i2},\cdots,P_{i,n-1}\}$ 抵达给定的终点 P_{in} 而形成的一条折线路径 $P_{i1},P_{i2},\cdots,P_{in}$，如图1所示。其中每个航向点的坐标都是给定的 $P_{ij}=(x_{ij},y_{ij},z_{ij})$。为了给出总路径最短这一目标，我们给出如下的刻画路径长度的归一化函数：

$$F_1=\begin{cases}1-\dfrac{\left\|\overrightarrow{P_{i1}P_{in}}\right\|}{\sum_{j=1}^{n-1}\left\|\overrightarrow{P_{ij}P_{i,j+1}}\right\|} & \left\|\overrightarrow{P_{ij}P_{i,j+1}}\right\|\geq R_{\min}\\ \infty & \text{otherwise}\end{cases}$$

其中 $R_{\min}$ 约束了每两个航向点之间的最短距离，由于惯性的存在，过短的距离是无法实现的。

2.2.2 避障

第二个目标是我们希望无人机可以避开障碍，选择足够安全的路径飞行。我们假设所有 K 个障碍都可以被建模为圆柱体，每个障碍由两个参数：中心位置 C_k 以及半径 R_k 定义，如图2所示。对于所有路径，我们判定撞击障碍的路径非法，离障碍过近的飞行由于扰动的存在较为危险，只有保持足够距离的路线才是安全的。据此，在给出无人机到障碍的安全距离 S 以及无人机自身尺寸 D 两个参数的的情况下，对每一个航段（相邻两个航向点之间的飞行路径）定义对障碍 k 的危险系数 $\mathcal{T}$：

$$\mathcal{T}_k\left(\overrightarrow{P_{ij}P_{i,j+1}}\right)=\begin{cases}0 & d_k\geq D+R_k+S\\ 1-\dfrac{d_k-D-R_k}{S} & D+R_k<d_k<D+R_k+S\\ \infty & \text{otherwise}\end{cases}$$

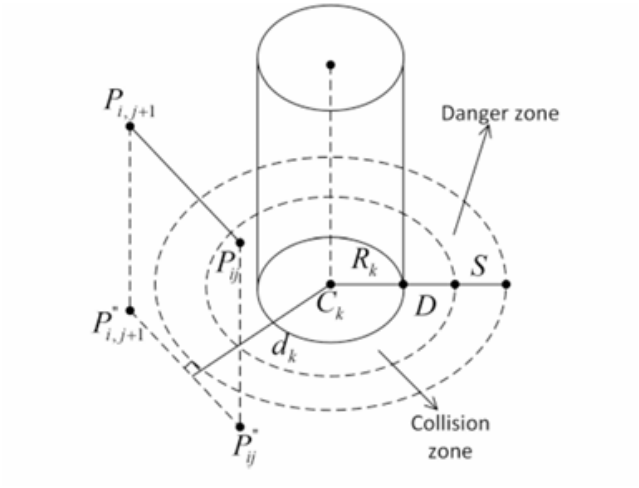

图 2: 障碍物建模示意

其中 d_k 表示障碍中心 C_k 到航段 $P_{ij}P_{i,j+1}$ 的最短距离。其含义就是在危险距离内，危险程度和到障碍物的距离成反比。最后再对这一危险系数做整段航行的平均与归一化：

$$F_2 = \frac{1}{K(n-1)} \sum_{j=1}^{n-1} \sum_{k=1}^{K} \mathcal{T}_k \left(\overrightarrow{P_{ij}P_{i,j+1}} \right)$$

2.2.3 平稳高度

测量，隐蔽等实际需求提出了第三个目标，无人机飞行时离地高度要足够平稳。这一目标用每个点的相对高差归一化系数的平均值来衡量。对航向点 P_{ij}，它的相对高差归一化系数为：

$$\mathcal{H} = \begin{cases} \dfrac{2\left|h_{ij} - h_{\text{mean}}\right|}{h_{\max} - h_{\min}} & h_{\min} \le h \le h_{\max} \\ \infty & \text{otherwise} \end{cases}$$

其中平均高度取为路径最大相对高度和最小相对高度的中值：

$$h_{\text{mean}} = \frac{h_{\max} + h_{\min}}{2}$$

于是对应刻画整段航行相对高度波动的目标函数为：

$$F_3 = \frac{1}{n} \sum_{j=1}^{n} \mathcal{H}_{ij}$$

2.2.4 转动角代价

我们希望总的飞行路线不要出现过多的转动，因为有过多转动的路线不适合发挥足够的速度，而且消耗更多的能量，因此第四个目标为路径的转向角代价。为了区分左转和右转两

个方向的转动，相邻两个航段的转向角以下式求解：

$$\eta_{ij} = \arctan\left(\frac{\left\|\overrightarrow{P_{ij}P_{i,j+1}} \times \overrightarrow{P_{i,j+1}P_{i,j+2}}\right\|}{\overrightarrow{P_{ij}P_{i,j+1}} \cdot \overrightarrow{P_{i,j+1}P_{i,j+2}}}\right)$$

转动角只能对起终点之外的点定义，对应目标函数即整段航行的归一化平均转向角：

$$F_4 = \frac{1}{n-2}\sum_{j=1}^{n-2}\frac{|\eta_{ij}|}{\pi}$$

2.3 多目标优化

对于 4 个目标函数，我们不能保证一定存在一条路径：

$$\hat{X} = (x_1, y_1, z_1, x_2, y_2, z_2, \cdots, x_n, y_n, z_n)$$

使得它在起终点固定且合法，相对高度恒正的约束下对于每一个目标函数都是最优的：

$$F_i(\hat{X}) \leq F_i(X) \quad \forall i \in \{1,2,3,4\}, X \in \mathbb{R}^{3n}$$

我们将求解这个多目标优化问题建模为求解这四个目标函数在可行集上的 Pareto-Front，即非支配解集 $\mathcal{P}$。它要求对任意一个 $X \in \mathcal{P}$，不存在任意一个 $X' \in \mathbb{R}^{3n}$ 满足:

$$F_i(X') \leq F_i(X) \quad i = 1,2,3,4$$

并且至少有一个 $j \in \{1,2,3,4\}$ 使得：

$$F_j(X') < F_j(X)$$

换言之，非支配解集中的任意一个解在决策空间中不被任意的其它的解支配。

三、 NMOPSO 算法原理

NMOPSO（Duong, Bui, and Phung 2025），即基于导航变量的 MOPSO 算法，作为粒子群算法 PSO 在求解上述建模的无人机路径规划问题时的一个优化变体，相比传统方法有明显的性能提升。我们首先介绍该算法的设计原理，并复现算法在该问题上的表现。

3.1 MOPSO

MOPSO 算法（Coello, Pulido, and Lechuga 2004）是一种将单目标 PSO 算法扩展到多目标领域的机制，主要是通过外部存档，网格机制和领导者选择策略来实现对于非支配解的搜索。

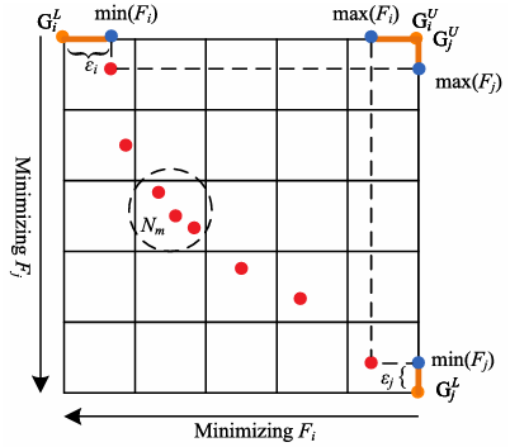

图 3: $[F_i, F_j]$ 下的二维分割示意

基本的粒子群算法如下：

$$v_i^{t+1} = wv_i^t + c_1r_1(x_{pb,i}^t - x_i^t) + c_2r_2(x_{gb,i}^t - x_i^t)$$
$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

其中 w 为惯性系数，c_1 和 c_2 分别是单粒子的历史最优系数和集体的历史最优系数，r_1 和 r_2 是 $[0,1]$ 分布中生成的随机数，用于引入一定的随机性。

PSO 容易出现粒子聚集在某个非支配解附近的情况，为了解决这个问题，MOPSO 引入了一个仓库 $\mathcal{P}$ 存储迄今为之发现的所有非支配解，并设计了一种方法，让每个粒子按照一定的概率选取某个已知的非支配解 x_{lb} 作为 x_{gb}，使得它们尽量去探索非支配解稀疏的未探索区域。具体来说，它将 $[F_1, F_2, F_3, F_4]$ 张成的超空间均匀的分割成超立方体，对于每个维度 F_i 在上下界内分割出均匀的 M 个部分，每个超立方体的边长为：

$$2\epsilon_i = \frac{1}{(M-1)}\left(\max_{x\in\mathcal{P}} F_i(x) - \min_{x\in\mathcal{P}} F_i(x)\right)$$

此时所有网格的上下界给定：

$$G_i^L = \min_{x\in\mathcal{P}} F_i(x) - \epsilon_i$$
$$G_i^U = \max_{x\in\mathcal{P}} F_i(x) + \epsilon_i$$

划分方式也给定。可以给出任意一个非支配解所处的网格编号：

$$c_i = \left\lfloor M\frac{F_i(x) - G_i^L}{G_i^U - G_i^L}\right\rfloor$$

分割方式的示意如图3。定义每一格的拥挤程度：

$$\gamma_m = e^{-\kappa N_m}$$

其中 N_m 表示该格内的非支配解数量。我们希望粒子尽量往稀疏的区域探索，因此它应该有更高的概率去 N_m 较少的区域，所以我们让粒子按照如下概率选择非支配解：

$$p_m = \frac{\gamma_m}{\sum_{l=1}^{\mathcal{L}} \gamma_l}$$

其中 $\mathcal{L}$ 为所有超立方体的数量。在这种方法下，MOPSO 的更新过程可以记为：

$$\begin{aligned} v_i^{t+1} &= w v_i^t + c_1 r_1 (x_{pb,i}^t - x_i^t) + c_2 r_2 (x_{lb,i}^t - x_i^t) \\ x_i^{t+1} &= x_i^t + v_i^{t+1} \end{aligned}$$

3.2 基于导航向量的表示

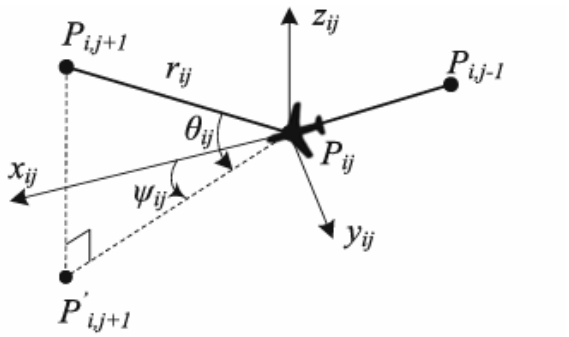

图 4: 相邻航向点之间的导航向量示意

对于 PSO 和 MOPSO，它们求解无人机路径问题都是将所有导航点的坐标拼接：

$$X_i = (x_{i1}, y_{i1}, z_{i1}, \cdots, x_{in}, y_{in}, z_{in})$$

即将每个路径表示为 $\mathbb{R}^{3n}$ 中的一个粒子执行算法。但是这个方法难以施加运动学模型的约束，NMOPSO 考虑使用导航向量代替途径点的坐标，即给出每一个航段的航行距离与方向：

$$\Gamma_i = (r_{i1}, \theta_{i1}, \psi_{i1}, \cdots, r_{in}, \theta_{in}, \psi_{in})$$

此时 r 表示航行距离，θ 为转向时的爬升角，ψ 为转向时的转向角，如图4所示。这时运动学约束就便于施加：

$$\begin{cases} |\theta_{ij}| \le \theta_{\max} \\ |\psi_{ij}| \le \psi_{\max} \end{cases}$$

而在 Γ_i 下，两个相邻航向点的其次坐标变换关系可以表示为：

$$P_{i,j+1} = \begin{bmatrix} x_{i,j+1} \\ y_{i,j+1} \\ z_{i,j+1} \\ 1 \end{bmatrix} = T_{i,j+1} \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \\ 1 \end{bmatrix} = T^j_{i,j+1} P_{ij}$$

其中 $T^j_{i,j+1}$ 为对应于 Γ_i 的齐次变换矩阵：

$$T^j_{i,j+1} = R_z(\psi_{ij}) R_y(\theta_{ij}) M_x(r_{ij}) = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi & \cos\psi\sin\theta & r\cos\psi\cos\theta \\ \sin\psi\cos\theta & \cos\psi & \sin\psi\sin\theta & r\sin\psi\cos\theta \\ -\sin\theta & 0 & \cos\theta & -r\sin\theta \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

任意一个航向点的坐标都可以通过经过的导航向量对应的齐次变换矩阵的连续作用计算。

3.3 自适应变异

为了进一步防止粒子聚集在局部最优，还需要给每个粒子的施加局部扰动，第 t 次迭代时对每个粒子的扰动规则如下：

$$x^t_{ij} = x^t_{ij} + \mathcal{N}_{ij} G^t x^t_{pb,i}$$

其中 x_{ij} 表示粒子 x_i 的随机选择得到的第 j 个分量。$\mathcal{N}_{ij}$ 为高斯分布 $N(0,1)$ 中采样的一个随机变量，变异增益 G_t 定义为：

$$G_t = \tanh\left(\frac{\Delta}{N^t_r}\right)$$

其中 N^t_r 表示第 t 次迭代时已经有非支配解存在的超立方体的数量，Δ 为一预先定义的超参数。这一增益形式决定了这一扰动是自适应的，当非支配解的分布过于密集，尚未探索的区域较大时随机扰动较强，而当解足够分散时扰动变弱。

NMOPSO 的伪代码如 **Algorithm1**（Duong, Bui, and Phung 2025）所示。

四、 NMOPSO 算法分析

4.1 运行结果

我们选取了一个数字高程模型，并人为定义了障碍参数。NMOPSO 算法运行时选择参数如下：粒子数量 $n = 12$，惯性权重 $w = 1$，学习因子 $c_1 = c_2 = 1.5$，网格分割数量 $M = 7$，拥挤系数 $\kappa = 2$，变异增益参数 $\Delta = 5$；运动学约束 $\phi_{\max} = \psi_{\max} = \frac{\pi}{4}$。结果如图5所示，其中地形被隐去了。该结果表明算法能够搜索到合法的路径，既没有撞击地形，也没有撞击障碍，并成功抵达了终点。此外，该路径较为平滑，结合 J_4 曲线的下降，说明 NMOPSO 很

Algorithm 1: Pseudocode for the NMOPSO algorithm

Get the search map and initial information;
Initialize swarm parameters w, c_1, c_2;
Initialize grid dimension M, scaling coefficient κ, mutation coefficient Δ;
foreach *particle i in swarm* **do**
    Generate random path Γ_i^0;
    Get particle's position X_i^0 from Γ_i^0;
    Evaluate fitness $F_j(X_i^0)$ of particle i;
    Set $pbest_i$ for particle i based on the fitness;
Initialize repository $\mathcal{P}$;
Create a hypergrid and find grid location c_i for solutions in $\mathcal{P}$;
while *not max_iteration* **do**
    foreach *particle i in swarm* **do**
        Select a leader from $\mathcal{P}$;
        Calculate velocity v_i^t and update new position Γ_i^t ;
        Map Γ_i^t to X_i^t;
        Update fitness $F_j(X_i^t)$;
        Update $pbest_i$;
        Apply mutation;
    Update $\mathcal{P}$;
    Update the hypergrid;
Set Pareto optimal set $\mathcal{P}^* = \mathcal{P}$;
Generate paths from non-dominated solutions in $\mathcal{P}^*$;

好地优化了无人机的动力学约束；路径也与障碍物保持了安全距离，将 J_2 控制在零值。就整体代价演化而言，除了在零值附近波动的 J_2，J_1, J_3, J_4 均表现出明显的下降趋势，说明了 NMOPSO 有效地引导了粒子向低代价区域移动。且在大约 550 次迭代后，各曲线都区域平稳，说明了算法已经收敛，找到了当前状态下的最优解集。

4.2 性能比较

我们将 NMOPSO 与 PSO，QPSO（量子行为粒子群优化算法）以及 DE（差分进化算法）的算法性能比较结果记录在图6中。由于这些算法都是为多目标函数设计的，我们在使

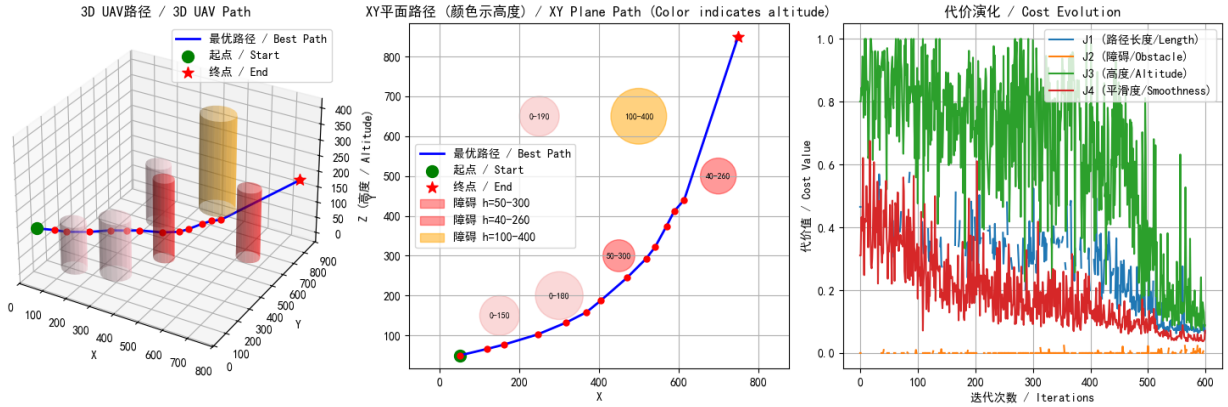

图 5: NMOPSO 算法运行结果

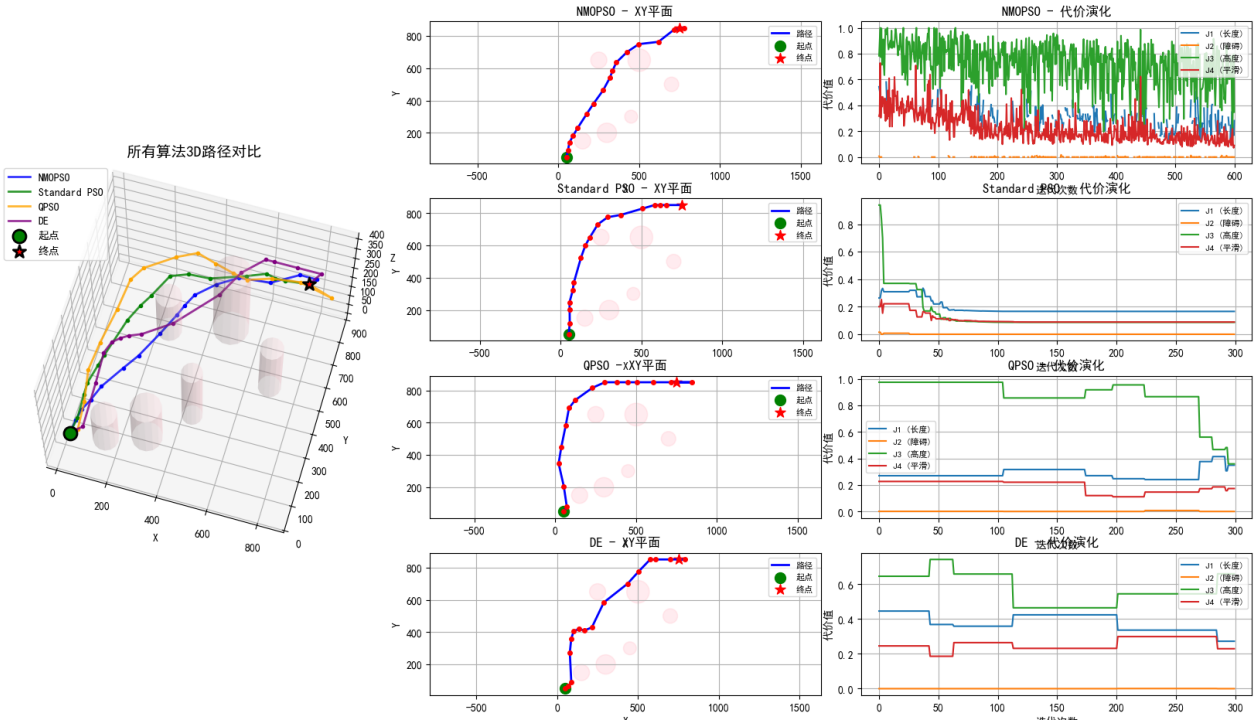

图 6: 启发式算法在无人机规划问题上的性能比较

用这些算法求解该问题时将问题转化为求解几个目标的加权和：

$$\hat{X} = \arg\min_{X} \hat{F}(X) \quad \hat{F}(X) = \sum_{i} w_i F_i(X)$$

由于我们在目标函数建模中已经将所有目标函数都归一化了，所以此处的权重取为 $w_i \equiv 1$。可以看出 NMOPSO 规划的路线具有如下的优势：

1. 从 3D 路径对比图和 XY 平面投影图可以观察出，相比于标准 PSO、QPSO 和 DE 算法，NMOPSO 生成的路径最为平滑且连续，说明了基于导航变量的路径表示法的有效性，运动学约束更稳健。

2. 在障碍物密集的 XY 平面实验中，NMOPSO 的路径在穿过障碍群时具有更均匀的安全距离，进而帮助它获得最短的总路径长度。这表明对多目标函数的非支配解集的求解使得它能够在保持避障的最优性的同时获得更好的在其它目标函数上的表现，而不会将多个目标模糊化。

3. 右侧代价演化曲线显示，PSO，QPSO 和 DE 算法在迭代初期容易陷入阶梯式停滞，表现出过早收敛，进一步优化困难的迹象。而 NMOPSO 的代价曲线波动反映了其自适应变异机制正在积极工作，通过维持种群多样性来持续探索非支配解空间，最终在全局搜索能力和多目标收敛精度上均表现出明显优势。

实验结果充分证明，NMOPSO 算法能够生成符合实际无人机物理飞行特性的最优路径，且在复杂多约束环境下的鲁棒性优于其他对比算法。

4.3 算法缺陷

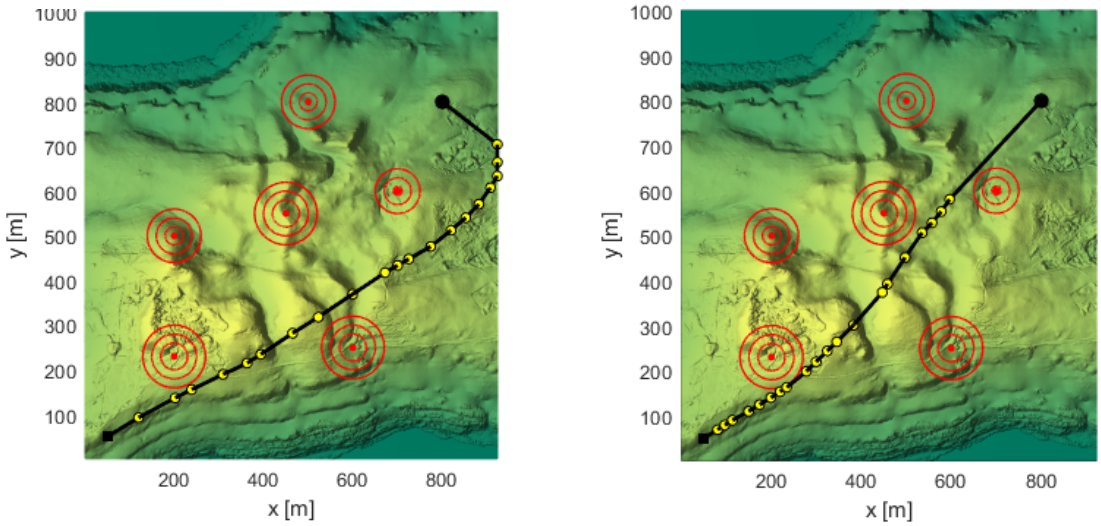

图 7: NMOPSO 运行结果，迭代次数分别为 800 与 2000 时

根据仿真对比实验结果，NMOPSO 在多目标路径规划任务中展现出以下显著缺陷：

1. **收敛速度缓慢：**在代价演化曲线中，PSO、QPSO 和 DE 均在 50 ∼ 100 代内迅速收敛，而 NMOPSO 的高度代价 J_3 和平滑度代价 J_4 在 400 代前仍剧烈震荡，直至 500 代后才平稳。这主要是由于对非支配解的搜寻以及拥挤度机制的存在导致演化初期 $\mathcal{P}$ 中存在大量互不支配的解，使粒子群失去强有力的全局最优引导，搜索方向过于发散。尤其是当我们希望路线能够细化，进而增大设置的导航向量的数量时，容易出现图7中左侧未收敛的结果，所需的最大迭代次数将急剧增长。

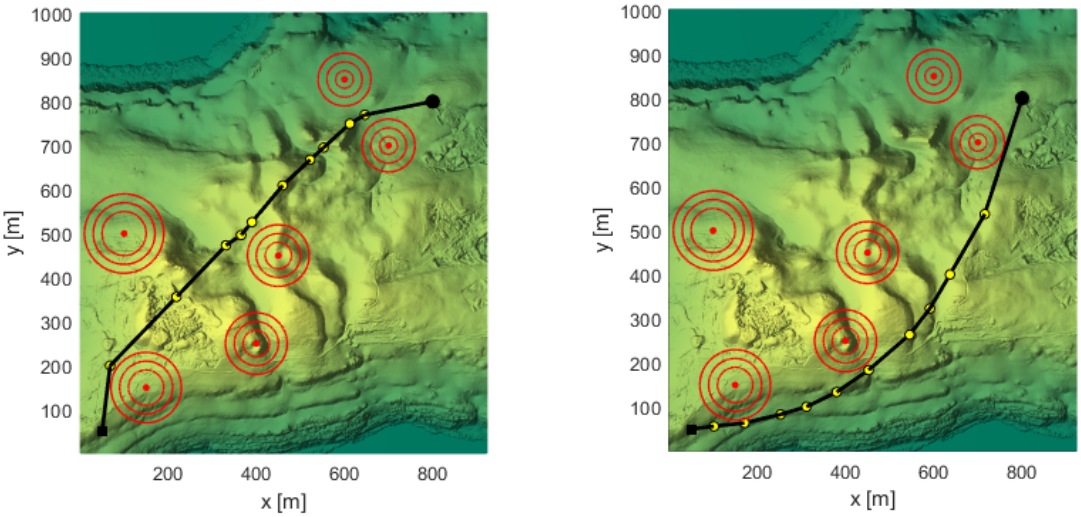

图 8: NMOPSO 运行结果，迭代次数相同时

2. **搜索稳定性差：** NMOPSO 的 J_3 和 J_4 曲线在迭代后期仍伴随明显的锯齿状波动，远不如 QPSO 与 DE 的阶梯式稳步下降。这反映了 NMOPSO 的选择偏好不足，缺少对于非支配解集中的解进行进一步选择的约束，因此算法在维护多个冲突目标平衡时，粒子易在不同非支配解之间摆动。

3. **初始位置依赖：**如果在起点较近处存在一个障碍，那么初始导航向对于路径规划起到决定性的作用，因为从左侧绕行和右侧绕行可能具有完全不同的结果与可通行性。在这种情况下，如果初始导航向量没能够很好地均匀分配到两个方向上，就容易陷入局部最优，并且由于两侧绕行意味着粒子的第一个导航向量需要有一个大角度的变化，在受到群体行为制约以及随机性不足的情况下是难以时限的。如图8所示，在迭代次数相同并且已收敛时算法会给出不同的两种路径，尽管它们可能是互不支配的，但是在偏好下两种结果显然是有区别的。

4.4 实用性改进

在 NMOPSO 中障碍被建模为无限高的圆柱体，这决定了路径只能从障碍的两侧绕过。但是在实际应用中，障碍往往是限高的，无论是武器的打击范围还是城市中的建筑物，使用限高的障碍都能够更好地建模问题并且找到上方通行路径。此外，对于一些限高空域，会存在下方通过的路径。为了提升算法求解的灵活性与泛化性，增强算法在实际场景中的应用，我们考虑对障碍区域增设高度范围。在这种情形下，我们修改障碍物的建模为中心位置 C_k，半径 R_k，高度范围 $[H_{kl}, H_{ku}]$ 的圆柱体，每一条边 $\overrightarrow{P_{ij}P_{i,j+1}}$ 的危险系数定义为：

$$\mathcal{T}_k(\overrightarrow{P_{ij}P_{i,j+1}}) = I\left[\mathcal{H}_k(\overrightarrow{P_{ij}P_{i,j+1}})\right] \cdot f(d_k)$$

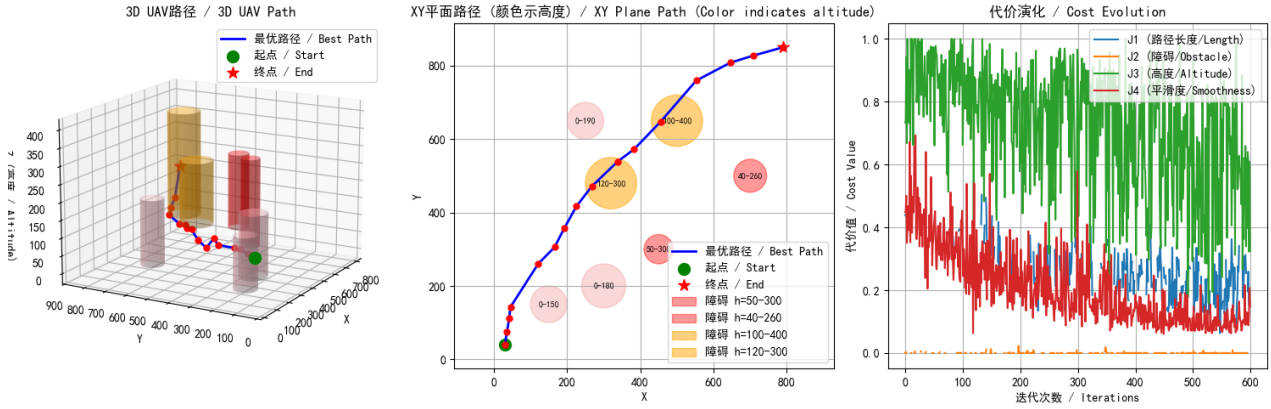

图 9: 增加障碍物高度限制后 NMOPSO 运行结果

其中，$f(d_k)$ 为原有的水平距离映射函数：

$$f\left(d_k\right)=\begin{cases}0 & d_k \geq D+R_k+S_1 \\ 1-\dfrac{d_k-D-R_k}{S_1} & D+R_k<d_k<D+R_k+S_1 \\ \infty & \text{otherwise}\end{cases}$$

而 $I\left[\mathcal{H}_k(\overrightarrow{P_{ij}P_{i,j+1}})\right]$ 为高度重叠示性函数，定义为：

$$I[\cdot]=\begin{cases}1, & [z_{\min},z_{\max}]\cap[H_{kl},H_{ku}]\neq\emptyset \\ 1-\dfrac{\max\{H_{kl}-z_{\min},z_{\max}-H_{ku},0\}}{S_2}, & [z_{\min},z_{\max}]\cap[H_{kl},H_{ku}]=\emptyset \ \ \& \\ & [z_{\min},z_{\max}]\cap[H_{kl}-S_2,H_{ku}+S_2]\neq\emptyset \\ 0, & [z_{\min},z_{\max}]\cup[H_{kl}-S_2,H_{ku}+S_2]=\emptyset\end{cases}$$

其中 $z_{\min}=\min(z_{ij},z_{i,j+1})$，$z_{\max}=\max(z_{ij},z_{i,j+1})$，$S_2$ 是垂直方向的安全距离。实际使用时，XY 平面内的安全距离 S_1 可以取得比较大，S_2 的取值可以比较小，可以按照下式来确定：

$$S_2=\arctan(\theta_{\max})\cdot S_1$$

或者更小，即上式结果的一半。

添加这一功能后算法运行结果如图9所示。引入限高障碍物后，仿真结果显示路径规划逻辑发生了质变。在限高模型下，无人机不再局限于传统的二维水平绕行，而是实现了真正的三维空间避障。无人机通过上方绕行越过了两个障碍，显著降低了在坐标 $(300,500)$ 附近利用修正后的代价函数 $\mathcal{T}_k$ 识别出的高度窗口，从限高障碍物的下方或缝隙中直接穿过。这种“跨越”策略显著缩短了路径总长度 J_1，证明了三维空间自由度释放对路径择优的积极影响。但是从代价曲线可以看出，限高条件下目标冲突更为剧烈，高度代价曲线 J_3 的震荡频率与幅值较无限高模型明显上升，这是由于算法在维持低高度平飞与爬升或俯冲以跨越限高边界之间进行着交替选择。

五、 GWO 算法原理及其分析

正如第 4.3 节所述，NMOPSO 虽然通过导航变量有效处理了运动学约束，但在面对复杂限高障碍物群时，其基于单一 G_{best} 的更新机制导致了收敛迟滞和路径震荡。为了克服粒子群算法在多模态搜索空间中容易陷入局部最优以及“早熟收敛”的固有缺陷，我们引入了灰狼优化算法（GWO，Mirjalili, Mirjalili, and Lewis 2014）。在本章节中，我们将多目标问题简化为单目标优化问题进行初步探究。通过构建加权代价函数，重点考察算法在复杂障碍环境下的收敛速度与寻优精度，并与前述改进前的算法进行对比，为后续的多目标扩展奠定基础。

5.1 算法介绍

GWO 算法模拟了灰狼的社会等级与群体狩猎行为。与 PSO 相比，其核心优势在于：

1. **多源引导机制**：利用群体中适应度最好的三匹狼（α, β, δ）共同指导位置更新，而非单一最优解，增强了跳出局部极值的能力。
2. **自适应收敛因子**：收敛因子 a 随迭代线性递减，使得算法实现了从早期的广域搜索到后期局部开发的平滑过渡。

5.2 算法实现

我们保持原本 NMOPSO 的导航变量 Γ 的编码方式不变，即将每个个体的解空间依然定义为一系列导航动作的集合 $\Gamma_i = (r_{i1}, \theta_{i1}, \psi_{i1}, \dots)$。然而，我们摒弃了 PSO 中基于速度矢量 v_i 和历史极值 $pbest$ 的更新规则，转而引入 GWO 的分级包围机制。

在每一次迭代 t 中，我们将种群中适应度值最优的三个解分别标记为 α（最优）、β（次优）和 δ（第三优），其余解标记为 ω。个体的位置更新不再依赖惯性飞行，而是由这三只头狼协同引导。对于第 i 个灰狼个体的第 j 维导航变量，其位置更新遵循以下数学模型：

首先，计算个体与三只头狼的距离向量：

$$
\begin{aligned}
D_\alpha &= |C_1 \cdot X_\alpha(t) - X_i(t)| \\
D_\beta &= |C_2 \cdot X_\beta(t) - X_i(t)| \\
D_\delta &= |C_3 \cdot X_\delta(t) - X_i(t)|
\end{aligned}
$$

其中，$X_\alpha, X_\beta, X_\delta$ 分别代表 α, β, δ 狼在当前维度的位置坐标；$C = 2 \cdot r_2$ 是随机扰动向量，$r_2 \in [0, 1]$。

随后，根据改进灰狼算法的策略，我们计算受三只头狼引导的潜在更新位置：

$$X_1 = X_\alpha - A_1 \cdot D_\alpha$$
$$X_2 = X_\beta - A_2 \cdot D_\beta$$
$$X_3 = X_\delta - A_3 \cdot D_\delta$$

此处 $A = 2a \cdot r_1 - a$ 为收敛系数向量，其中 a 随迭代次数从 2 线性递减至 0。最终，下一时刻的位置由三者的加权平均决定：

$$X_i(t+1) = w_1 X_1 + w_2 X_2 + w_3 X_3$$

在本研究中，为了强化最优解的主导作用，权重系数取为 $w_1 = 0.5, w_2 = 0.3, w_3 = 0.2$。这种机制将无人机路径规划问题从粒子飞行转化为狼群围捕，有效提升了在复杂障碍环境下的搜索效率。

为了处理多目标向单目标的转化，我们构建加权适应度函数 $Fitness$，并对违反避障约束（$F_2 > 0$）的个体施加惩罚：

$$Fitness(X) = \begin{cases} \infty, & F_2(X) > 0 \\ \lambda_1 F_1(X) + \lambda_3 F_3(X) + \lambda_4 F_4(X), & \text{otherwise} \end{cases}$$

具体算法流程如 **Algorithm 2** 所示。

5.3 结果分析

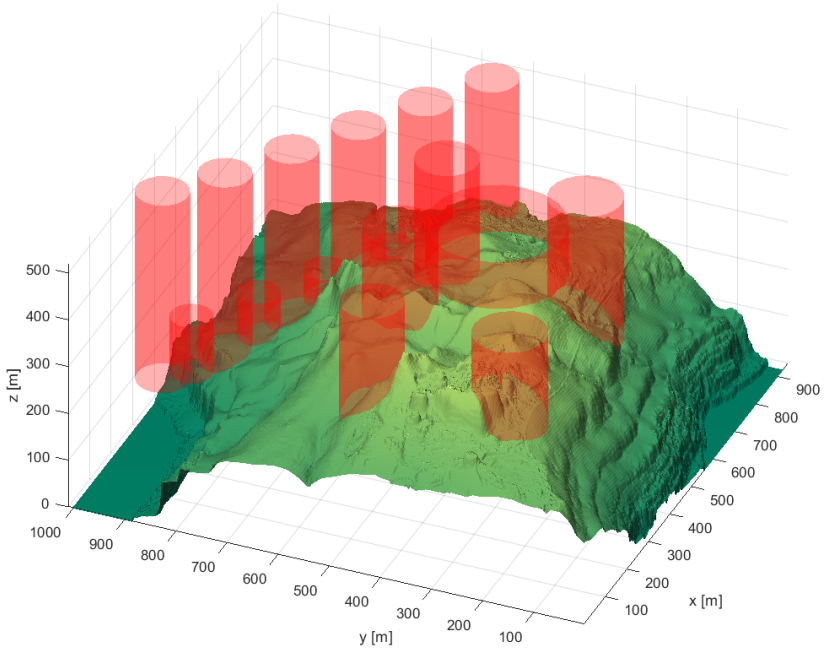

图 10: 复杂场景示意图

Algorithm 2: Pseudocode for the Weighted GWO Path Planner

Input : Obstacle Map, Limits, PopSize N, MaxIt T_{max}

Output: Optimal Path Γ_{best}

Initialize population Γ_i $(i = 1, \ldots, N)$ randomly within bounds;
Initialize parameters a, A, C and weights w_1, w_2, w_3;
foreach *wolf i in population* **do**
    Calculate Fitness f_i using weighted cost function with penalty;
Sort population and Select top 3 as $X_\alpha, X_\beta, X_\delta$;
while $t < T_{max}$ **do**
    Update convergence factor $a = 2 - 2\frac{t}{T_{max}}$;
    foreach *wolf i in population* **do**
        foreach *dimension j of* Γ_i **do**
            Update A, C vectors using random r_1, r_2;
            Calculate distances $D_\alpha, D_\beta, D_\delta$;
            Calculate candidates X_1, X_2, X_3;
            Update position: $X_i^j(t+1) = w_1X_1 + w_2X_2 + w_3X_3$;
        Boundary Check for $\Gamma_i(t+1)$;
    foreach *wolf i in population* **do**
        Evaluate Fitness $f_i(t+1)$;
        Update Global Best if $f_i < f_{best}$;
    Update $X_\alpha, X_\beta, X_\delta$ based on new fitness;
    $t = t + 1$;
Return X_α;

为了评估算法在极端环境下的鲁棒性，本研究构建了一个具有更高拓扑复杂度的仿真场景。如图10所示，该场景集成了垂直维度的机动规避与狭窄空间的穿梭飞行等高难度约束。

NMOPSO 和 GWO 进行规划后的路线结果和目标函数曲线如图13和图11中所示。从结果可以看出，在 3D 地形图中，GWO 规划的绿色路径展现出更强的穿透性，能够以更接近直线的方式通过障碍密集区，这表明 GWO 的层级领导机制在全局搜索和局部开发之间取得了更好的平衡。相比之下，NMOPSO 的路径在遇到中部障碍物群时出现了明显的迂回。这种现象说明 NMOPSO 极易被障碍物边缘的局部可行区域吸引，因缺乏足够的扰动能力而陷入了局部次优的绕行方案。

对四项目标函数的曲线分析可以得出，GWO 的曲线呈现规律的的阶梯式下降，代表其具有更强的、稳定的突破局部最优的能力；相较之下，NMOPSO 经过了一个较明显的平台

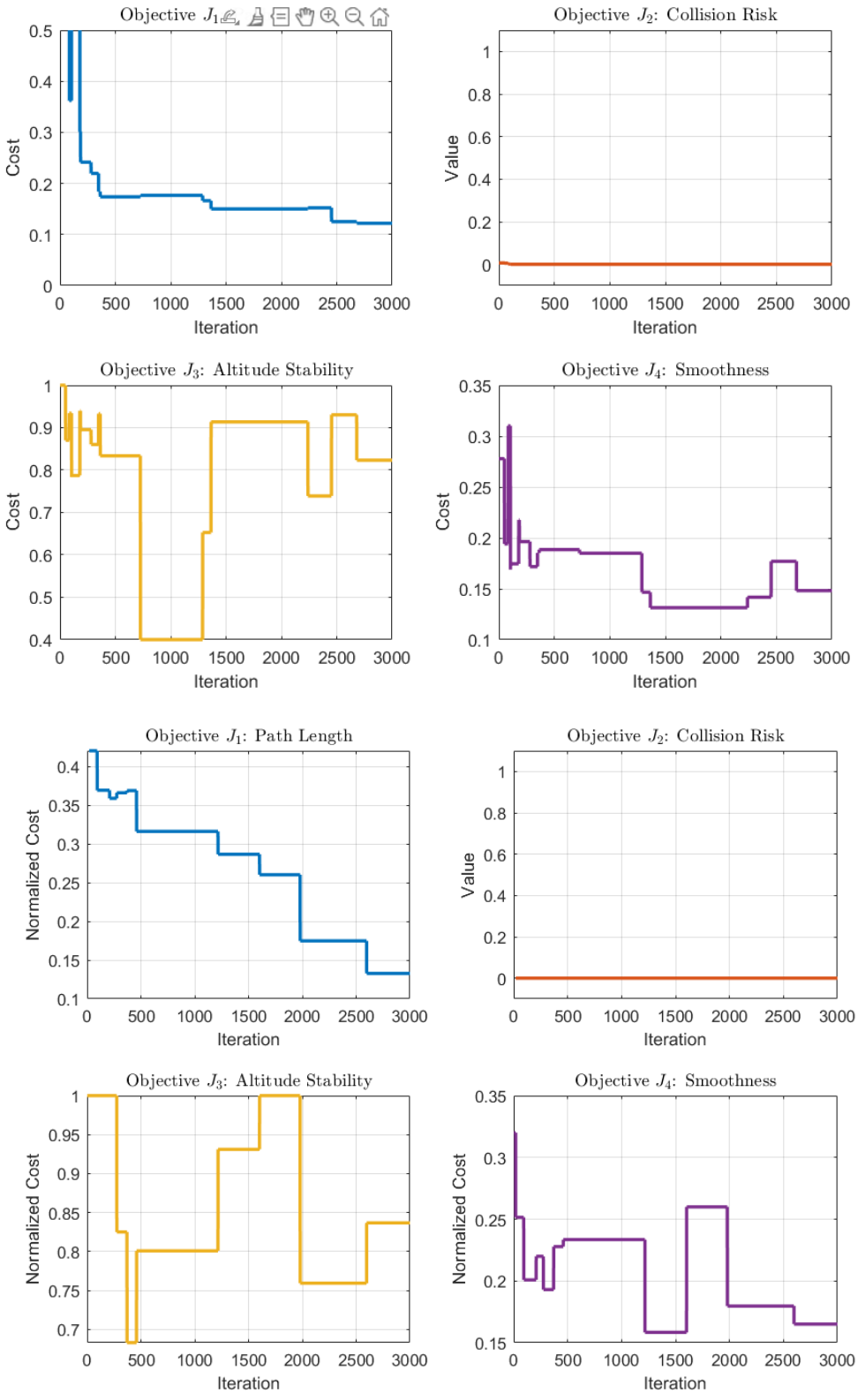

图 11: NMOPSO 与 GWO 在复杂场景下的路径规划目标函数值比较

期，在平台期内算法持续在局部最优之间徘徊，无法有效摆脱，因此对于局部最优的摆脱能力仍然较弱。

就最终结果而言，尽管 GWO 在突破局部最优方面表现卓越，但其收敛速度明显不如 NMOPSO，最终结果也不如 NMOPSO 优秀。从曲线对比可见，NMOPSO 在演化初期能迅速压低代价值，而 GWO 需要历经较多迭代次数才能完成一次阶梯式突破。

六、 MOGWO 算法原理及其分析

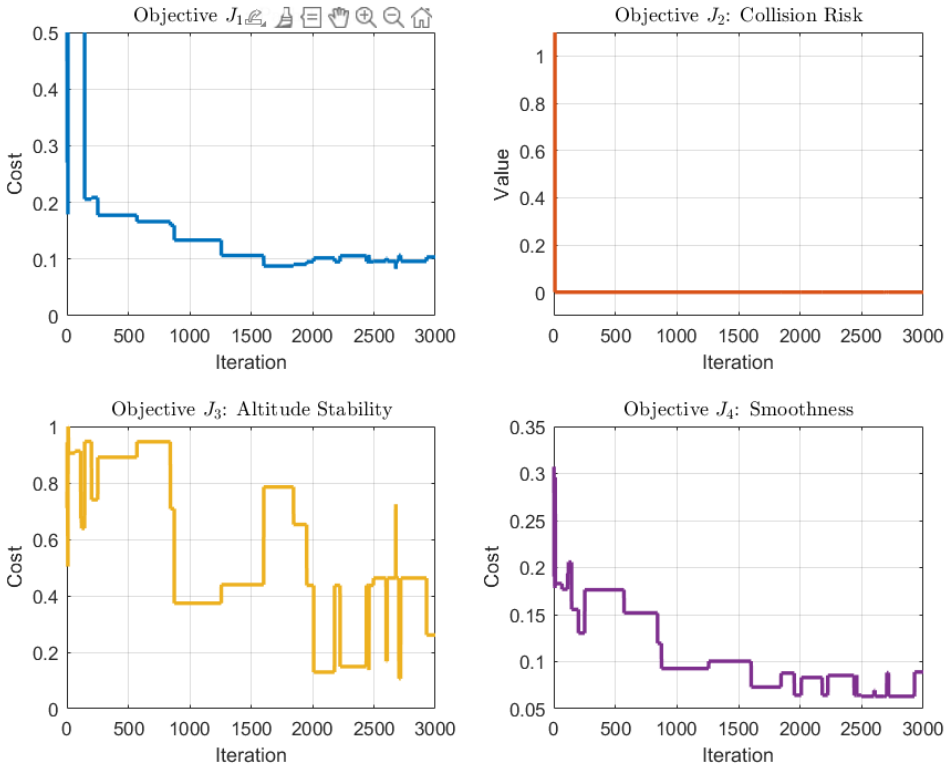

图 12: MOGWO 在复杂场景下的目标函数演化

6.1 算法介绍

在第 5 章中，我们验证了引入灰狼机制后，算法在单目标加权场景下具有显著的收敛速度优势。然而，加权求和法存在本质局限性：它要求决策者预先设定权重，这种设定不仅具有主观性，且无法揭示路径长度与平滑度、安全性之间复杂的非线性权衡关系。

为了弥补这一缺陷，本章提出的 MOGWO 算法不再追求单一的最优解，而是旨在获取一个分布均匀、覆盖面广的 Pareto 非支配解集。该算法保留了导航变量 Γ 的编码方式，利用外部档案和网格机制来平衡解集的收敛性与多样性。

6.2 算法原理

为了解决上述多目标优化模型，我们设计了基于导航变量的多目标灰狼优化算法。该算法的程序逻辑主要由三个核心模块构成：基于网格的档案维护、多头狼引导策略以及位置更新机制。具体的算法流程如 **Algorithm 3** 所示。

在代码实现中，我们采用外部档案 $\mathcal{A}$ 实时存储 Pareto 非支配解。为了在有限的计算资源下保持解集分布的均匀性，引入了自适应网格机制。当档案数量超过预设上限 N_{arc} 时，算法调用删除函数，依据网格拥挤度移除多余解。

同时，α, β, δ 三只头狼的选择不再依赖单一适应度，而是通过 `SelectLeader` 函数从档案中基于轮盘赌策略独立采样。这种随机性采样保证了狼群能够同时兼顾来自 Pareto 前沿不同区域的引导信息。

6.3 结果分析

同样的，我们在之前建立的复杂环境中运行 MOGWO 进行规划求解，求解结果如图12所示，路径如图13所示。观察 MOGWO 的路径长度曲线 J_1，其呈现出非常典型的阶梯状下降特征（如在 500、1500 及 2500 代附近均有明显的代价跳变）。这证明了 MOGWO 依靠其独特的三狼（α, β, δ）协同引导机制，能够有效产生足够的扰动，从而多次跳出局部陷阱寻找全局最优路径。相比之下，NMOPSO 的 J_1 曲线在约 1000 代后进入平台期，表现出明显的“早熟收敛”迹象，难以进一步优化。

同时，就收敛结果而言，MOGWO 的结果完全支配了 NMOPSO 以及 GWO 的结果，除了 J_2 都被控制在 0 之外，MOGWO 最终的 J_1, J_3, J_4 都比前面两种算法得到的的结果更好，即 MOGWO 找到了一个在全局逻辑上更合理的解。就收敛速度而言，MOGWO 也具有巨大优。从 J_1 上看，MOGWO 仅仅使用了 1500 次迭代就成功收敛到了 0.1 以下，J_3 使用了约 2000 次迭代来到 0.4 以下（另外两种算法在 3000 次迭代内均为达到此表现），J_4 更是仅仅使用了不到 1000 次迭代就达到了从未出现的 0.1 的表现。

综上所述，MOGWO 算法在处理复杂多约束环境下的 UAV 路径规划时，展现了比 NMOPSO 以及 GWO 更卓越的表现，是解决高维非凸优化问题的更优选择。

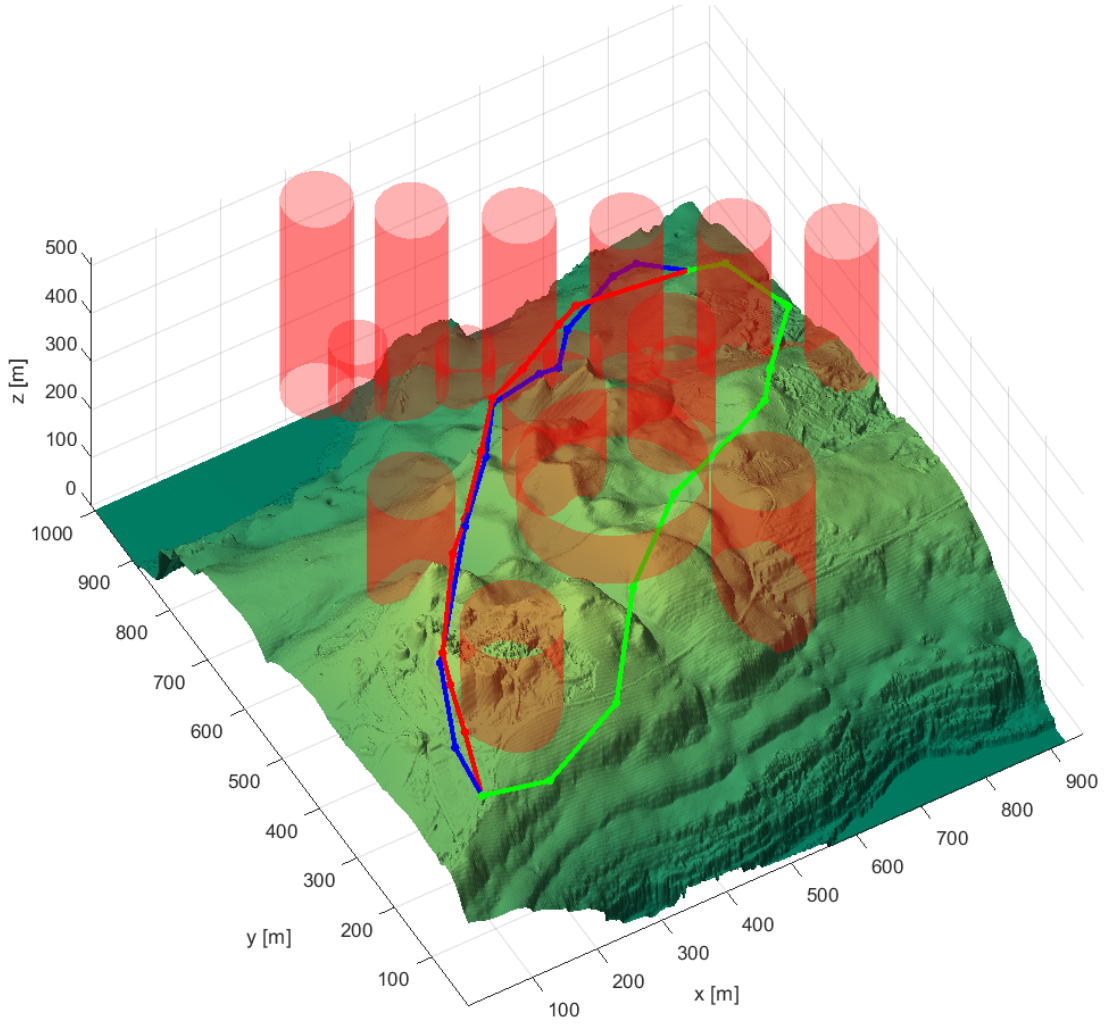

图 13: NMOPSO，GWO 和 MOGWO 的结果路径绘制

Algorithm 3: Pseudocode for MOGWO Path Planner

Input : Model $\mathcal{M}$, MaxIt T, PopSize N, ArchiveSize N_{arc}, GridDiv N_{grid}

Output: Pareto Optimal Set $\mathcal{A}$

Initialize wolf population Γ_i $(i = 1, \ldots, N)$ randomly;
Evaluate Cost $F(\Gamma_i)$ for all wolves;
Determine non-dominated solutions and initialize Archive $\mathcal{A}$;
`CreateGrid(`$\mathcal{A}, N_{grid}$`)`;
for $t = 1$ **to** T **do**
    Calculate convergence factor $a = 2 - 2\frac{t}{T}$;
    `// Select leaders from Archive using Roulette Wheel`
    if $|\mathcal{A}| < 3$ **then**
        $\Gamma_\alpha, \Gamma_\beta, \Gamma_\delta \leftarrow$ best available in $\mathcal{A}$;
    else
        $\Gamma_\alpha \leftarrow$ `SelectLeader(`$\mathcal{A}, \beta$`)`;
        $\Gamma_\beta \leftarrow$ `SelectLeader(`$\mathcal{A}, \beta$`)`;
        $\Gamma_\delta \leftarrow$ `SelectLeader(`$\mathcal{A}, \beta$`)`;
    `// Update positions for the swarm`
    foreach *wolf* $i \in 1 \ldots N$ **do**
        foreach *dim* $j \in \{r, \psi, \phi\}$ **do**
            $X_1 \leftarrow$ position guided by Γ_α using A, C;
            $X_2 \leftarrow$ position guided by Γ_β using A, C;
            $X_3 \leftarrow$ position guided by Γ_δ using A, C;
            $\Gamma_i^j(t+1) \leftarrow (X_1 + X_2 + X_3)/3$;
        Boundary check and evaluate new Cost $F(\Gamma_i)$;
    `// Update Archive and Grid`
    Add non-dominated wolves from population to $\mathcal{A}$;
    Remove dominated solutions from $\mathcal{A}$;
    `CreateGrid(`$\mathcal{A}, N_{grid}$`)`;
    `// Archive Pruning if full`
    while $|\mathcal{A}| > N_{arc}$ **do**
        Select a crowded grid cell using Roulette Wheel;
        Remove one solution from the selected cell;
return $\mathcal{A}$;

References

Coello, C.A.C., G.T. Pulido, and M.S. Lechuga (2004). "Handling multiple objectives with particle swarm optimization". In: *IEEE Transactions on Evolutionary Computation* 8.3, pp. 256–279. DOI: 10.1109/TEVC.2004.826067.

Duong, Thi Thuy Ngan, Duy-Nam Bui, and Manh Duong Phung (Mar. 2025). "Navigation variable-based multi-objective particle swarm optimization for UAV path planning with kinematic constraints". In: *Neural Computing and Applications* 37.7, pp. 5683–5697. ISSN: 1433-3058. DOI: 10.1007/s00521-024-10945-1. URL: https://doi.org/10.1007/s00521-024-10945-1.

Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis (2014). "Grey Wolf Optimizer". In: *Advances in Engineering Software* 69, pp. 46–61. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2013.12.007. URL: https://www.sciencedirect.com/science/article/pii/S0965997813001853.