基于鱼鹰搜索策略蛇优化算法的无人机三维航迹规划

陈海洋，温仕琪，张江祺，都 威
（西安工程大学电子信息学院，西安，710048）

摘要 针对智能优化算法在求解无人机三维航迹规划问题时存在搜索能力不足、收敛速度慢和易陷入局部最优等问题，提出了一种鱼鹰搜索策略蛇优化（OSSO）算法。首先，引入 Bernoulli 混沌映射初始化种群，扩大个体搜索范围，丰富种群的多样性；其次，结合鱼鹰优化算法下潜捕食、随机步长和精确开采思想，改进算法搜索策略，增强其全局搜索能力；然后，通过动态折射反向学习策略进行种群更新，提升算法对局部极值的处理能力，平衡算法全局搜索和局部开采能力；最后，分别采用函数法和高程数据构建 2 种三维模型，将航迹长度、威胁区距离和无人机本体约束作为评判指标进行仿真实验。实验结果表明：OSSO 算法具有较强的鲁棒性，对于求解三维航迹规划问题具有良好的稳定性和有效性。

关键词 无人机；航迹规划；蛇优化算法；鱼鹰搜索策略；反向学习
DOI 10.3969/j.issn.2097-1915.2025.02.011
中图分类号 TP301.6 **文献标志码** A **文章编号** 2097-1915(2025)02-0089-11

A Three-Dimensional Path Planning for Drone Based on Osprey Strategy Snake Optimizer

CHEN Haiyang, WEN Shiqi, ZHANG Jiangqi, DU Wei
(School of Information and Electronics, Xi'an Polytechnic University, Xi'an 710048, China)

Abstract Aimed at the problems that search ability is inadequate in search, convergence is slow at speed, and susceptible to local optima in the intelligent optimization algorithm for solving the UAV 3D flight planning problem, an Osprey Strategy Snake Optimizer (OSSO) is proposed. Firstly, Bernoulli chaotic mapping is introduced to initialize the population, expand the individual search range, and enrich the diversity of the population; Secondly, the search strategy is improved in combination with the ideas of submerged predations, stochastic step and precise mining in the Osprey Strategy Snake Optimizer, and the global search capability is enhanced; And then, the dynamic opposition-based learning is utilized for updating population, balancing the algorithm's global exploration and local mined ability, and improving the algorithm's ability to deal with local optima. Finally, two 3D models are constructed by the function method and elevation data respectively, and the simulation experiment is performed by taking the length of trajectory, the distance in threat zone and the drone physical constraints as the judging indexes. The experimental results show that the OSSO algorithm is rugged, and good in stability and in effectiveness in solving the three-dimensional track planning problems.

收稿日期：2024-07-15
基金项目：国家自然科学基金(51905405)
作者简介：陈海洋(1967－)，男，陕西西安人，副教授，博士，研究方向为人工智能。E-mail：chy_00@163.com
通信作者：温仕琪(1998－)，男，河南洛阳人，硕士生，研究方向为人工智能。E-mail：428033923@qq.com

引用格式：*陈海洋，温仕琪，张江祺，等. 基于鱼鹰搜索策略蛇优化算法的无人机三维航迹规划[J]. 空军工程大学学报，2025，26(2)：89-99.*
CHEN Haiyang, WEN Shiqi, ZHANG Jiangqi, et al. A Three-Dimensional Path Planning for Drone Based on Osprey Strategy Snake Optimizer [J]. Journal of Air Force Engineering University, 2025, 26(2): 89-99.

Key words drone;path planning;snake optimizer;osprey strategy;opposition-based learning

现代化军事行动是以通信技术,网络技术为基础,利用各种无人化、智能化设备,以高效、精确、隐蔽的方式完成作战任务[1]。随着智能化程度的提升,无人机(unmanned aerial vehicle,UAV)被定义为具备自主决策能力的机器人。同时,由于其高机动性、易回收、低成本、长续航等特点,被各国广泛应用在打击目标、侦查、突防等多种作战任务中[2-3]。因此,无人机在信息化、现代化的军事行动中起到了中流砥柱的作用[4]。航迹规划问题是无人机导航和控制领域的核心问题之一,也是无人机任务分配中最重要的组成部分[5]。航迹规划和无人机的任务完成度、生存概率等都有着直接关系。但二维航迹规划难以满足无人机作战任务需求,三维航迹规划由于可以综合考虑无人机的各种性能约束、地形威胁代价等因素,因此,如何在三维环境中更加高效、精确地规划出安全、可靠的航迹,是目前急需解决的重要问题。

航迹规划算法分为传统经典算法和智能优化算法[6]。常用的传统经典算法有 A* 算法[7]和快速扩展随机树(rapidly-exploring random trees, RRT)算法[8]。传统经典算法具有快速和结构简单等特点,但针对复杂三维航迹规划问题时,难以满足规划时间和精度的要求。而智能优化算法具有强鲁棒性和高并行性,可以更加高效、精确地完成三维航迹规划任务。智能优化算法包括粒子群优化(particles swarm optimization, PSO)算法[9]和沙猫群优化(sand cat swarm optimization, SCSO)算法[10]等。文献[11]将粒子群算法和差分进化(differential evolution algorithm, DE)算法结合,引入正态扰动策略,并将其运用到无人机城市规划中,实验结果表明,该算法可以规划出高效且平滑的航迹,但三维环境较为简单,难以满足实际需求。文献[12]提出了一种基于非线性调整机制和自适应莱维飞行策略的沙猫群优化算法,同时构建了山地和城市 2 种三维环境来验证算法性能,实验结果表明,该算法具有良好的收敛速度和搜索精度,但未与其他改进算法进行对比。针对上述算法存在的问题,需要一种搜索能力强和规划时间短的智能优化算法。蛇优化(snake optimizer,SO)算法[13]具有参数设置简单、寻优精度高的优点,适合解决三维航迹规划问题,但也存在收敛速度慢、易陷入局部最优、种群初始化随机程度严重的问题。

为此,本文建立了 2 种三维航迹规划模型,并提出了一种融合鱼鹰搜索思想和动态折射反向学习的蛇优化(osprey strategy snake optimizer, OSSO)算法来解决 SO 算法的不足。通过和其他 3 种算法进行对比,验证 OSSO 算法在求解航迹规划问题的有效性和优越性。

1 无人机航迹规划建模

1.1 环境建模

目前,无人机航迹规划在三维空间中的建模方法主要有 Voronoi 图法和栅格法[14]。将三维地理环境离散化,将地图空间划分为多个体积的立方体网格。本文的搜索空间用 $n\times n$ 的矩阵来表示,其中 n 为地图空间划分的栅格个数,矩阵中 z_{ij} 为该栅格范围中的地形高度,z_{tk} 为威胁区域的高度。因此,规划空间 $\boldsymbol{\sigma}$ 可表示为:

$$\boldsymbol{\sigma}=\begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1n}+z_{tm} \\ z_{21} & z_{22}+z_{t1} & \cdots & z_{2n} \\ \vdots & \vdots & & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nn} \end{pmatrix} \tag{1}$$

通过栅格图建模后,航迹规划问题就简化为在搜索空间 $\boldsymbol{\sigma}$ 中按照航迹点设置要求,在避开障碍区的同时,依次选择最优的航迹点的问题。无人机航迹规划点集 P 表示为:

$$P=\{S,P_1,P_2,\cdots,P_n,T\} \tag{2}$$

式中:S 为无人机的起始点;T 为无人机的目标点;P_i 为无人机第 $i(i=1,2,\cdots,n)$ 个航迹点。

1.2 无人机航迹规划模型

无人机在实际执行任务过程中,不仅要避开自然地形和威胁区,还要考虑无人机自身性能约束,如高度约束和飞行距离约束,这样规划的航迹才真实可飞。因此,本文综合考虑无人机航迹长度、避障威胁、飞行高度、飞行距离和转角要求等代价函数,建立无人机航迹规划模型。

1.2.1 航迹长度代价函数

航迹长度是评价无人机航迹规划能力的重要指标之一。飞行距离越短,飞行遇到威胁的概率也会越低,任务执行时间也越短。考虑到无人机和障碍可能发生碰撞,无人机航迹长度代价函数表示为:

$$f_{\text{length}}=\begin{cases}\infty, & \text{相交于地图模型}\\ \sum\limits_{i=1}^{n-1}L_i, & \text{不相交}\end{cases} \tag{3}$$

$$L_i=\sqrt{(x_{i+1}-x_i)+(y_{i+1}-y_i)+(z_{i+1}-z_i)} \tag{4}$$

式中：L_i 为第 i 条航迹段的长度；(x_i,y_i,z_i) 为第 i 个航迹点 P_i 的坐标。若航迹段和模型相交，则判定无人机坠毁，无法继续航行。

1.2.2　威胁代价函数

无人机规划的航迹除了要满足航迹长度，还需要尽可能远离威胁区域。建立的威胁区如图 1 所示。

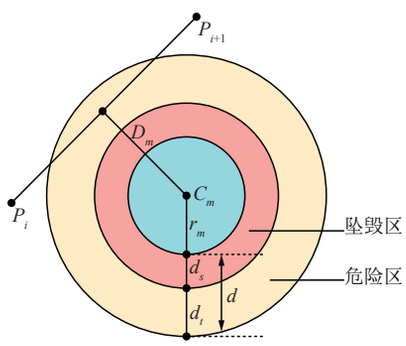


图 1　威胁代价函数

Fig. 1　Threat cost function

将威胁区建模中心 C_m 的坐标设为 (x_m,y_m)，d_m 为威胁区中心到无人机航迹段的距离，也是判断无人机是否穿过威胁区的条件。若 d_m 大于安全距离，则此段威胁代价函数为零；而当判断无人机穿过危险区时，就会给予无人机一定的惩罚；如果无人机穿过坠毁区，就会判断无人机坠毁并且无法继续航行。威胁区建模的函数表示为：

$$f_{\text{threaten}}=\sum_{i=1}^{n-1}\sum_{m=1}^{m}T_m\overrightarrow{(P_{i+1}P_i)} \tag{5}$$

$$T_m\overrightarrow{(P_{i+1}P_i)}=\begin{cases}0, & d_m\geqslant d+r_m\\ (d+r_m)-d_m, & r_m+ds<d_m<r_m+d\\ \infty, & d_m\leqslant r_m+ds\end{cases} \tag{6}$$

式中：T_m 为航迹规划空间中的第 m 个威胁区；$T_m\overrightarrow{(P_{i+1}P_i)}$ 为无人机经过第 m 个威胁区的代价值。

1.2.3　飞行高度约束

无人机在航行时，应降低飞行高度，利用地形来降低被探测雷达发现的概率。但要保持安全距离防止和地形相撞，因此需要设计 1 个最低飞行高度 $Z_{\min}$ 和最高飞行高度 $Z_{\max}$，如图 2 所示。

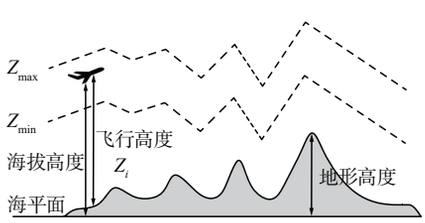


图 2　飞行高度约束

Fig. 2　Flight altitude constraints

由图 2 可知，$Z_{\max}$ 和 $Z_{\min}$ 随着地形高度的变化而不断变化，确保无人机和地形保持安全距离。其代价函数表示为：

$$f_{\text{cost1}}=\sum_{i=1}^{n}Z_i \tag{7}$$

$$Z_i=\begin{cases}0, & Z_{\min}\leqslant z_i<Z_{\max}\\ \infty, & \text{其他}\end{cases} \tag{8}$$

式中：Z_i 为第 i 个航迹段的高度代价；z_i 为无人机相对地面的飞行高度。若 z_i 处于高度约束范围则不进行处罚，反之无人机将无法继续航行。

1.2.4　飞行距离约束

无人机在航行时由于自身携带的电池电量或者载油量有限，因此应设置 1 个最远飞行航迹 $L_{\max}$，无人机航迹长度超过 $L_{\max}$ 便无法继续航行，其代价函数形式表示为：

$$f_{\text{cost2}}=\begin{cases}0, & \sum_{i=1}^{n}L_i\leqslant L_{\max}\\ \infty, & \text{其他}\end{cases} \tag{9}$$

无人机在调整飞行位姿时，需要一定距离进行缓冲调节，因此要求无人机每个航迹段不能超过最小航迹段距离 $L_{\min}$，函数形式表示为：

$$f_{\text{cost3}}=\sum_{i=1}^{n}M_i \tag{10}$$

$$M_i=\begin{cases}0, & L_i\geqslant L_{\min}\\ \infty, & \text{其他}\end{cases} \tag{11}$$

式中：M_i 为第 i 个航迹段的最小航迹段代价。

1.2.5　俯仰角、转弯角约束

由于无人机动力学的性能瓶颈，无人机进行大幅度的俯冲或转弯时，会造成无人机自身状态不稳定，因此需设置 1 个最大俯仰角 $\theta_{\max}$ 和最大转弯角 $\varphi_{\max}$，俯仰角和转弯角的关系如图 3 所示。

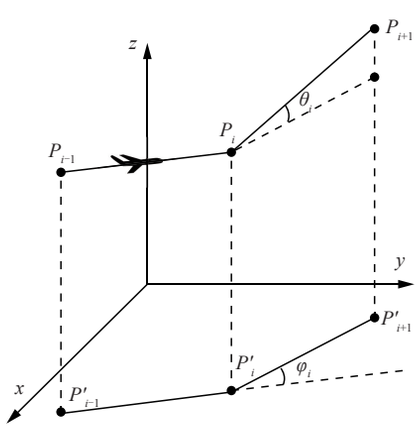


图 3　俯仰角和转弯角约束

Fig. 3　Pitch and turn angle constraints

在图 3 中，P'_i 为航迹点 P_i 映射到二维平面的投影点。若无人机运动时超过了 $\theta_{\max}$ 或 $\varphi_{\max}$，则会增加代价以示惩罚。俯仰角约束的函数形式表示为：

$$f_{\text{cost4}}=\begin{cases}0,\ \theta_i\leqslant\theta_{\max}\\ \sum_{i=1}^{n-1}(|\theta_i|-\theta_{\max}),\ \text{其他}\end{cases}\tag{12}$$

$$\theta_i=\arctan\left(\frac{|z_{i+1}-z_i|}{\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}}\right)\tag{13}$$

式中：(x_i,y_i,z_i)为第 i 个航迹点 P_i 的坐标值；θ_i 为第 i 段航迹俯仰角。当 θ_i 未超过最大俯仰角时，其约束代价为零；反之，其代价值用所有违反约束条件的 θ_i 和最大俯仰角的差值之和来计算。

φ_i 为第 i 段航迹转弯角，当 φ_i 超过最大转弯角时，使用其和最大转弯角的差值作为代价值。转弯角约束的函数形式为：

$$f_{\text{cost5}}=\begin{cases}0,\ \varphi_i\leqslant\varphi_{\max}\\ \sum_{i=2}^{n-1}(|\varphi_{i-1}|-\varphi_{\max}),\ \text{其他}\end{cases}\tag{14}$$

$$Q_1=(x_{i+1}-x_i)(x_i-x_{i-1})+(y_{i+1}-y_i)(y_i-y_{i-1})\tag{15}$$

$$Q_2=\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}\sqrt{(x_i-x_{i-1})^2+(y_i-y_{i-1})^2}\tag{16}$$

$$\varphi_{i-1}=\arccos\left(\frac{Q_1}{Q_2}\right)\tag{17}$$

1.2.6 目标优化函数

综合考虑航迹代价函数以及无人机自身约束条件，本文构建的目标优化函数 F_{cost} 表示为：

$$F_{\text{cost}}=\omega_1 f_{\text{length}}+\omega_2 f_{\text{threaten}}+\omega_3\sum_{i=1}^{5}f_{\text{cost}\_i}\tag{18}$$

式中：ω_1、ω_2、ω_3 分别为航迹长度代价函数、威胁代价函数、无人机自身性能约束的权重系数，可根据无人机任务侧重点的不同来调节权重大小。

2 鱼鹰搜索策略蛇优化算法

2.1 基本蛇优化算法

2.1.1 初始化种群

蛇优化算法首先在搜索空间中生成随机分布的蛇个体，初始化公式为：

$$\boldsymbol{X}_i=\boldsymbol{X}_{\min}+\text{rand}(\boldsymbol{X}_{\max}-\boldsymbol{X}_{\min})\tag{19}$$

式中：$\boldsymbol{X}_i$ 为第 i 个蛇个体；$\boldsymbol{X}_{\max}$ 和 $\boldsymbol{X}_{\min}$ 分别为待优化问题的上下界，rand 为 0～1 之间的随机数。

2.1.2 控制参数

蛇算法的控制参数由食物源 Q 和温度 Temp 组成，表达式为：

$$Q=C_1\exp\left(\frac{t-T}{T}\right)\tag{20}$$

$$\text{Temp}=\exp\left(\frac{-t}{T}\right)\tag{21}$$

式中：C_1 为常数且等于 0.5；t 为当前迭代次数；T 为最大迭代次数。

2.1.3 探索阶段

蛇算法按照探索阶段进行位置更新，需满足 $Q<\text{Threshold}_1$，其中 $\text{Threshold}_1=0.25$。位置更新公式为：

$$\boldsymbol{X}_{i\_\text{m}}(t+1)=\boldsymbol{X}_{\text{rand}\_\text{m}}(t)+\text{flag}\ C_2\ A_{\text{m}}((\boldsymbol{X}_{\max}-\boldsymbol{X}_{\min})\text{rand}+\boldsymbol{X}_{\min})\tag{22}$$

$$\boldsymbol{X}_{i\_\text{f}}(t+1)=\boldsymbol{X}_{\text{rand}\_\text{f}}(t)+\text{flag}\ C_2\ A_{\text{f}}((\boldsymbol{X}_{\max}-\boldsymbol{X}_{\min})\text{rand}+\boldsymbol{X}_{\min})\tag{23}$$

$$A_{\text{m}}=\exp\left(\frac{-f_{\text{rand}\_\text{m}}}{f_{i\_\text{m}}}\right)\tag{24}$$

$$A_{\text{f}}=\exp\left(\frac{-f_{\text{rand}\_\text{m}}}{f_{i\_\text{f}}}\right)\tag{25}$$

式中：$\boldsymbol{X}_{i\_\text{m}}$ 和 $\boldsymbol{X}_{i\_\text{f}}$ 分别为第 i 个雄性个体和雌性个体的所在位置；$\boldsymbol{X}_{\text{rand}\_\text{m}}$ 和 $\boldsymbol{X}_{\text{rand}\_\text{f}}$ 分别为随机雄性位置和随机雌性位置；flag 是 1 或－1 的随机数；C_2 为常数且等于 0.05；A_{m} 和 A_{f} 分别为雄性和雌性寻找食物的能力；f_{rand_m} 和 f_{rand_f} 分别为 $\boldsymbol{X}_{\text{rand}\_\text{m}}$ 和 $\boldsymbol{X}_{\text{rand}\_\text{f}}$ 的适应度值；f_{i_m} 和 f_{i_f} 分别为第 i 个雄性个体和雌性个体的适应度值。

2.1.4 开发阶段

当 $Q>\text{Threshold}_1$ 且 $\text{Temp}>\text{Threshold}_2$ 时，蛇个体会向食物源移动，其中 $\text{Threshold}_2=0.6$。位置更新公式为：

$$\boldsymbol{X}_{i,j}(t+1)=\boldsymbol{X}_{\text{food}}+\text{flag}\ C_3\ \text{Temp}\ \text{rand}(\boldsymbol{X}_{\text{food}}-\boldsymbol{X}_{i,j}(t))\tag{26}$$

式中：$\boldsymbol{X}_{i,j}$ 为第 i 个个体所在位置；$\boldsymbol{X}_{\text{food}}$ 为最优个体所在位置；C_3 为常数且等于 2。

当 $\text{Temp}<\text{Threshold}_2$ 时，蛇个体会随机进行战斗或交配模式，战斗模式更新公式为：

$$\boldsymbol{X}_{i\_\text{m}}(t+1)=\boldsymbol{X}_{i\_\text{m}}(t)+C_3\ F_{\text{m}}\ \text{rand}(\boldsymbol{Q}\boldsymbol{X}_{\text{best}\_\text{f}}-\boldsymbol{X}_{i\_\text{m}}(t))\tag{27}$$

$$\boldsymbol{X}_{i\_\text{f}}(t+1)=\boldsymbol{X}_{i\_\text{f}}(t)+C_3\ F_{\text{f}}\ \text{rand}(\boldsymbol{Q}\boldsymbol{X}_{\text{best}\_\text{m}}-\boldsymbol{X}_{i\_\text{f}}(t))\tag{28}$$

$$F_{\text{m}}=\exp\left(\frac{-f_{\text{best}\_\text{f}}}{f_{i\_\text{m}}}\right)\tag{29}$$

$$F_{\text{f}}=\exp\left(\frac{-f_{\text{best}\_\text{m}}}{f_{i\_\text{f}}}\right)\tag{30}$$

式中：$\boldsymbol{X}_{\text{best}\_\text{m}}$ 和 $\boldsymbol{X}_{\text{best}\_\text{f}}$ 分别为雄性种群中和雌性种群中的最优个体所在位置；F_{m} 和 F_{f} 分别为雄性个体和雌性个体的战斗能力；f_{best_m} 和 f_{best_f} 分别为 $\boldsymbol{X}_{\text{best}\_\text{m}}$ 和 $\boldsymbol{X}_{\text{best}\_\text{f}}$ 的适应度值。

交配模式更新公式为：

$$\boldsymbol{X}_{i\_\text{m}}(t+1)=\boldsymbol{X}_{i\_\text{m}}(t)+C_3\ M_{\text{m}}\ \text{rand}(\boldsymbol{Q}\boldsymbol{X}_{i\_\text{f}}(t)-\boldsymbol{X}_{i\_\text{m}}(t))\tag{31}$$

$$\boldsymbol{X}_{i\_f}(t+1)=\boldsymbol{X}_{i\_f}(t)+C_3\ M_f\ \mathrm{rand}(\boldsymbol{Q}\boldsymbol{X}_{i\_m}(t)-\boldsymbol{X}_{i\_f}(t)) \tag{32}$$

$$M_m=\exp\left(\frac{-f_{i\_f}}{f_{i\_m}}\right) \tag{33}$$

$$M_f=\exp\left(\frac{-f_{i\_m}}{f_{i\_f}}\right) \tag{34}$$

式中：M_m 和 M_f 分别为雄性个体和雌性个体的交配能力。

雄蛇和雌蛇交配会产生新的个体，新个体随机生成并替换当代最差个体，表示为：

$$\boldsymbol{X}_{worst\_m}=\boldsymbol{X}_{min}+\mathrm{rand}(\boldsymbol{X}_{max}-\boldsymbol{X}_{min}) \tag{35}$$

$$\boldsymbol{X}_{worst\_f}=\boldsymbol{X}_{min}+\mathrm{rand}(\boldsymbol{X}_{max}-\boldsymbol{X}_{min}) \tag{36}$$

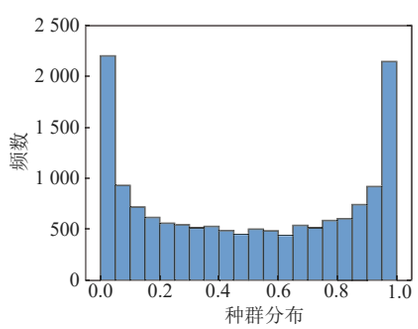
式中：$\boldsymbol{X}_{worst\_m}$ 和 $\boldsymbol{X}_{worst\_f}$ 分别为最差雄性个体和最差雌性个体的所在位置。

2.2　鱼鹰搜索策略蛇优化算法

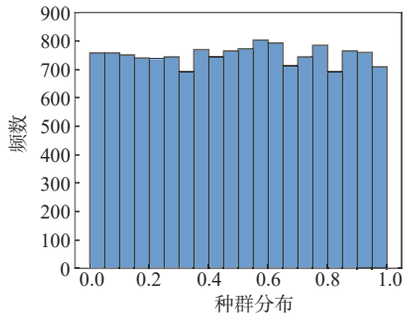
在传统蛇优化算法中，种群初始化采用随机分布生成，造成算法种群多样性降低；其次，算法在探索阶段按随机方向进行无规律探索，个体寻优存在一定的盲目性，会导致算法前期陷入停滞；最后，算法在交配模式随机产生新个体，具有不稳定性，一定程度上会导致算法全局搜索能力较弱并影响算法局部开发能力。为解决上述问题，本文引入了 3 种改进措施：Bernoulli 混沌映射、鱼鹰搜索策略、动态折射反向学习。

2.2.1　Bernoulli 混沌映射

和其他群智能算法一致，SO 算法其初始解分布是通过 rand 函数生成的，更容易出现初始解粘连，导致种群的搜索范围受限，从而降低算法的寻优性能。混沌映射则不同于 rand 函数，具有更复杂的动态特性，通过迭代产生的混沌序列具有不可预测性，从而增加个体的差异并提升种群多样性。现有文献大部分采用 Logistic 混沌映射初始化种群，Bernoulli 混沌映射相较于常见的 Logistic 混沌映射，其搜索遍历范围更广，可以更均匀地产生初始解。2 种混沌序列如图 4 所示。



(a) Logistic 混沌序列



(b) Bernoulli 混沌序列

图 4　2 种混沌映射对比

Fig. 4　Comparison of two chaotic

在图 4 中，Logistic 混沌映射的种群集中分布在两端，导致两端的初始解过多并出现粘连现象，而 Bernoulli 混沌映射是分段式映射，且具有非线性和非周期性等特点，因此对初始条件的变化更加敏感，使其在迭代过程中初始种群的分布更加均匀，从而产生差异更明显的初始解，能够最大程度地提升算法的多样性。同时 Bernoulli 混沌映射相较于 Logistic 混沌映射能快速地在不同状态之间切换，使得算法更加迅速地探索搜索空间，在各种复杂的优化问题中具有更好的适应性。

因此本文引入 Bernoulli 混沌映射来初始化种群，可使种群更加均匀地分布在搜索空间中，增强算法的全局搜索能力，其表达式为：

$$x(t+1)=\begin{cases}\dfrac{x(t)}{1-\alpha}, & 0<x(t)\leqslant 1-\alpha\\[2ex] \dfrac{x(t)-(1-\alpha)}{\alpha}, & 1-\alpha<x(t)\leqslant 1\end{cases} \tag{37}$$

式中：α 为混沌系数，通常取 0.4。

采用混沌映射策略初始化种群，首先需要算法随机生成初始解，然后使用式(37)对初始解进行迭代计算并生成混沌序列，最后将混沌序列映射到 SO 算法的初始种群中。映射公式表示为：

$$\boldsymbol{X}=\boldsymbol{X}_{min}+x(\boldsymbol{X}_{max}-\boldsymbol{X}_{min}) \tag{38}$$

式中：$\boldsymbol{X}$ 为混沌映射后的初始解；$\boldsymbol{X}_{max}$ 和 $\boldsymbol{X}_{min}$ 分别为待优化问题的上下界。

2.2.2　鱼鹰搜索策略

蛇优化算法早期迭代采用随机搜索对区域进行探索，探索不确定性高且搜索范围小，导致算法前期易陷入停滞且搜索能力弱。而鱼鹰优化(osprey optimization algorithm，OOA)算法[15]具有控制参数少、寻优能力强和稳定性好等优点。因此引入 OOA 算法相关思想，对 SO 算法进行改进。

在 SO 算法中，食物源 Q 是切换探索阶段和开

发阶段的重要参数。算法前期由于食物源的匮乏，从而处于探索阶段并随机搜索。随着迭代次数增加，食物源 Q 也会非线性地增加，在大于食物源阈值后，算法切换至开发阶段并进行全局搜索，导致算法前期探索易陷入停滞。因此结合 OOA 算法下潜捕猎思想，提出了非线性减小的食物源更新方式，表示为：

$$Q=\alpha_1 C_1-(\alpha_1-\alpha_2)\frac{2}{2+\exp(\alpha_3-\alpha_4 t)} \tag{39}$$

式中：α_1、α_2、α_3、α_4 均为常数。α_1 影响 Q 的初始值，α_2、α_3、α_4 影响 Q 的减缓速度。多次实验后发现，当 $\alpha_1=0.9$、$\alpha_2=0.4$、$\alpha_3=10$、$\alpha_4=0.04$ 时，算法在全局探索和局部开发能力之间达到了最佳平衡。

在迭代前期，个体会向最优食物位置移动，解决了原算法前期停滞不前的问题。但原算法的位置更新公式步长固定，导致全局搜索能力较弱。本文引入 OOA 算法随机步长思想，改进原算法的全局随机探索方式，表达式为：

$$\boldsymbol{X}_{i,j}(t+1)=\boldsymbol{X}_{i,j}(t)+\text{flag}\ C_3\ \text{Temp}\ \text{rand}(\boldsymbol{X}_{\text{food}}-I_{i,j}\boldsymbol{X}_{i,j}(t)) \tag{40}$$

式中：$I_{i,j}$ 为 1～2 之间的随机整数。由于改进的位置更新公式加入了随机步长 $I_{i,j}$，使 OSSO 算法在搜索空间中以不同幅度进行跳跃式搜索，增加算法的搜索范围，提高发现潜在最优解区域的概率，从而提升了算法的全局探索能力。同时不同个体在每次迭代可能采用不同步长进行位置更新，从而在搜索空间中分散开来，防止其过度聚集在当前最优解附近，因此降低了算法受当前最优位置的影响，使其不容易出现早熟现象。

算法在局部开发阶段，个体会沿当代最优位置进行局部开发，会造成算法陷入局部极值。因此借鉴 OOA 算法精确开采思想，改进原算法的局部开发方式，表达式为：

$$\boldsymbol{X}_{i,j}(t+1)=\boldsymbol{X}_{i,j}(t)+\frac{\boldsymbol{X}_{\min}+\text{rand}(\boldsymbol{X}_{\max}-\boldsymbol{X}_{\min})}{t} \tag{41}$$

式中：t 为算法当前迭代次数。随着迭代次数的增加算法的搜索步长逐渐减小，使其不易遗漏最优解，有利于后期精确寻优。同时会使种群位置产生细微变化，增强算法局部开发能力，并向当前最优解附近的更优解逐渐收敛。

2.2.3 动态折射反向学习策略

在 SO 算法的局部开发中会过于依赖最优个体的所在位置，使算法陷入局部极值并无法摆脱。因此将动态折射反向学习引入 SO 算法的种群更新公式中，来增强算法跳出局部极值的能力。折射反向学习策略是通过凸透镜折射原理而提出的，其原理如图 5 所示。

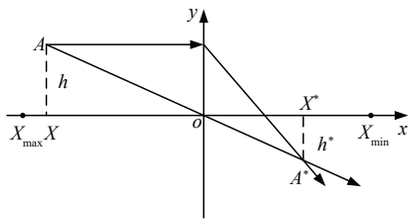


图 5 动态折射反向学习原理图

Fig. 5 dynamic opposition-based learning schematic

在图 5 中，$X_{\max}$ 和 $X_{\min}$ 为问题的上下限，A^* 为 A 折射后的位置，A^* 和 A 在 x 轴上的投影分别为 X^* 和 X，高度为 h^* 和 h。因此，关于 $\boldsymbol{X}(t)$ 的反向解 $\boldsymbol{X}^*(t)$ 的表达式为：

$$\boldsymbol{X}^*(t)=\frac{\boldsymbol{X}_{\max}+\boldsymbol{X}_{\min}}{2}+\frac{\boldsymbol{X}_{\max}+\boldsymbol{X}_{\min}}{2n}-\frac{\boldsymbol{X}(t)}{n} \tag{42}$$

式中：n 为缩放系数。其表达式为：

$$n=h/h^* \tag{43}$$

折射反向学习中，n 一般取常数。但在迭代后期，其反向解也可能陷入局部最优。因此引入了一种动态折射反向学习，其 n 的表达式为：

$$n=(1+(t/T)^{0.5})^{10} \tag{44}$$

随着迭代次数增加，缩放系数 n 也会增加，其反向解范围将从大到小，从而更加适合前期全局大范围搜索和后期局部精确搜索的 SO 算法。

使用动态折射反向学习策略更新种群，首先，按照式(35)、式(36)生成新个体的位置。然后，将新个体位置通过式(42)生成反向解，最后，将反向解适应度值和新个体适应度值进行比较，选择适应度值更小的一方作为新生成个体。

2.3 OSSO 算法步骤

OSSO 算法步骤如下：

步骤 1 设置 OSSO 算法基本参数，种群规模 N，搜索维度 d，最大迭代次数 T。

步骤 2 基于 Bernoulli 混沌映射生成初始种群 $\{\boldsymbol{X}_i\}$，并将种群平分为雌雄 2 个种群，分别计算出 2 个种群中每个个体的适应度值，并记录雄蛇和雌蛇的最优位置。

步骤 3 分别使用式(21)、式(39)计算温度阈值 Temp 和食物量 Q。

步骤 4 若 $Q>0.25$，判断 Temp 是否大于 0.6 成立，根据式(40)进行全局探索，转入步骤 8。

步骤 5 若 Temp$<$0.6，判断随机数 rand 是否大于 0.6 成立，根据式(27)、式(28)选择更优的个体，转入步骤 8。

步骤 6 若 rand$<$0.6，根据式(31)、式(32)产

生新个体，并通过镜透成像反向学习生成反向种群，选择 2 个种群中更优的作为新个体，转入步骤 8。

步骤 7　若 $Q<0.25$，根据式(41)进行局部探索，转入步骤 8。

步骤 8　对比个体适应度值，淘汰最差个体。

步骤 9　若 $t<T$，则令 $t=t+1$，转至步骤 3；否则算法迭代结束，输出最优位置。

3　仿真结果分析

3.1　仿真环境

仿真实验采用的计算机环境为 Windows11 操作系统，运行内存为 16 G，处理器为 Intel(R) Core i9-12900H，主频率 5 GHz，并在 MATLAB2021a 上运行。

3.2　改进策略有效性分析

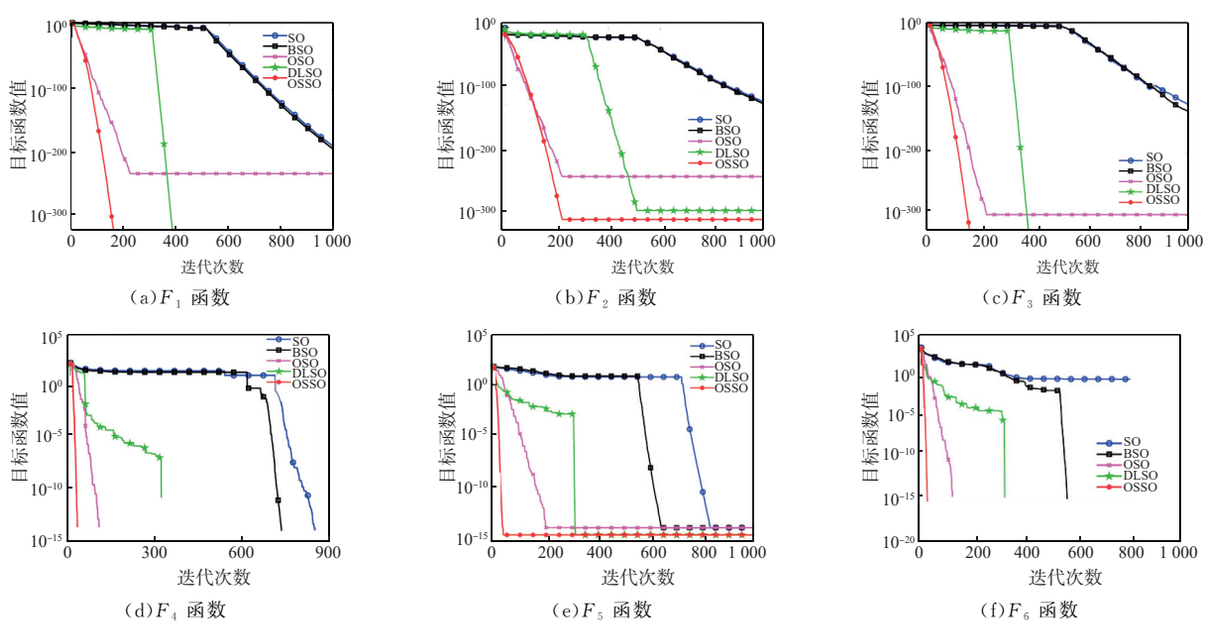
为了验证 Bernoulli 混沌映射、鱼鹰搜索策略和动态折射反向学习对 SO 算法改进的有效性，本文对加入单一策略的改进算法进行对比实验。选择 6 个基准测试函数进行寻优对比测试，其中，$F_1\sim F_3$ 为单峰函数，用于测试算法的收敛速度和搜索精度，$F_4\sim F_6$ 为多峰函数，可有效检验算法的全局搜索性能和跳出局部最优的能力。测试函数相关信息如表 1 所示。

将融入 Bernoulli 混沌映射的 SO 算法记为 BSO，加入鱼鹰搜索策略的 SO 算法记为 OSO，加入动态折射反向学习策略的 SO 算法记为 DLSO。基本 SO 算法、BSO 算法、OSO 算法、DLSO 算法和 OSSO 算法在 6 个测试函数上的寻优对比如图 6 所示。

表 1　基准测试函数

Tab. 1　Basic text function

函数类型	测试函数	搜索范围	极小值
单峰	$F_1(x)=\sum_{i=1}^{D}x_i^2$	$[-100,100]^D$	0
	$F_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10,10]^D$	0
	$F_3(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	$[-100,100]^D$	0
多峰	$F_4(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]^D$	0
	$F_5(x)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+\mathrm{e}$	$[-32,32]^D$	0
	$F_6(x)=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]^D$	0



(a)F_1 函数　(b)F_2 函数　(c)F_3 函数

(d)F_4 函数　(e)F_5 函数　(f)F_6 函数

图 6　函数收敛曲线

Fig. 6　Function convergence graph

由图 6 可知，对于单峰测试函数 F_1～F_3，OSO 算法中采用的鱼鹰搜索策略起主导作用，解决了原算法前期易陷入停滞的问题，同时提升了算法的全局探索能力。动态折射反向学习策略提升了算法的搜索精度，加强了其跳出局部最优的能力。BSO 算法的寻优效果不明显，但和其他改进策略相结合，使得 OSSO 算法的收敛速度和精度均优于其他算法。在 F_5 中，动态折射反向学习策略起决定性作用，增强了算法的局部开发能力，使 DLSO 算法和 OSSO 算法能收敛到更接近理论极值附近。在 F_4 和 F_6 中，所有改进算法均能收敛到理论极值 0，但 OSSO 算法的收敛速度更快。通过实验说明，本文提出的改进策略增强了 OSSO 算法的全局搜索能力，同时有效帮助其跳出局部极值。

3.3 三维环境模型

本文使用 OSSO 算法求解 2 种三维环境，分别为函数模型和高程数据模型，函数模型的计算量较小，考验算法收敛速度和搜索精度，而高程数据模型考验算法平衡全局探索和局部开采能力。

3.3.1 函数模型

函数建模下的无人机航迹规划任务空间设置为 500 m×500 m×100 m，给定的威胁区域参数和山体模型参数分别如表 2 和表 3 所示。无人机起始点坐标为(10 m，100 m，10 m)，任务目标点坐标为(470 m，420 m，60 m)。

表 2 函数模型的威胁区域参数

Tab. 2 Threat area parameterized by the functional model

威胁区中心 (x_t,y_t)/m	威胁区高度 H_t/m	威胁区半径 R_t/m
(270,200)	100	20
(170,350)	100	30
(300,300)	100	25
(350,400)	100	30

表 3 山体模型参数

Tab. 3 Parameters of mountain model

山体中心 (x_i,y_i)/m	山体坡度 (x_{si},y_{si})/m	山体高度 H_i/m
(50,70)	(46.8,46.8)	48
(70,300)	(20.4,20.4)	41
(150,450)	(46.4,46.4)	96
(170,150)	(38.1,38.1)	56
(170,390)	(15.1,15.1)	46
(250,400)	(46.9,46.9)	36
(300,100)	(34.1,34.1)	31
(350,270)	(35.9,35.9)	78
(400,450)	(16.5,16.5)	84
(445,120)	(43.9,43.9)	88

将 OSSO 算法、PSO 算法、SO 算法及融合双向搜索和精英反向学习的蛇优化(bi-directional search and elite opposition-based learning snake optimizer，BEESO)算法[16]求解航迹规划的结果进行对比。各算法种群数设置为 30，最大迭代次数为 100，航迹点个数为 5，并对生成航迹进行 3 次样条插值平滑处理[17]。无人机的最大俯仰角和最大转弯角设为 90°，无人机飞行高度区间为 0～100 m，目标优化函数的权重系数 ω_1、ω_2、ω_3 分别为 0.4、0.3、0.3。

图 7 为无人机在函数模型下，OSSO 算法、PSO 算法、SO 算法和 BEESO 算法分别得到的无人机最优航迹，其中，褐色部分为山体模型，朱红色圆柱为威胁区域，图 8 为其俯视图。表 4 为函数模型的实验统计结果，其中最优值为算法独立运行 30 次后的最小值，平均值为算法独立运行 30 次后的平均值。图 9 为 4 种算法的适应度函数迭代曲线。

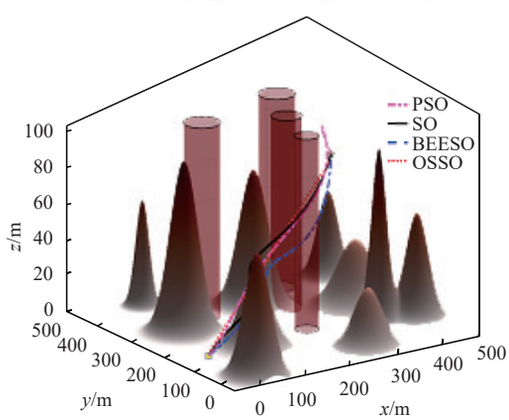


图 7 函数建模下航迹规划对比

Fig. 7 Comparison of track planning under functional model

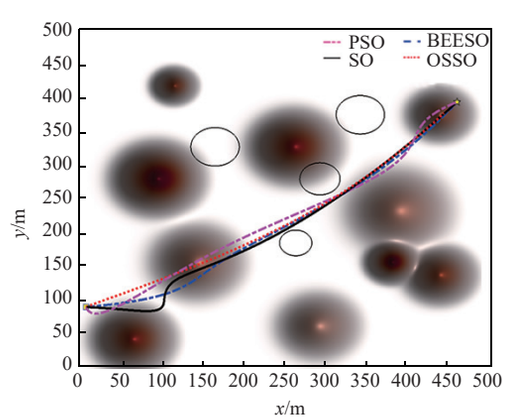


图 8 函数建模下航迹规划对比俯视图

Fig. 8 Top view of track planning comparison under functional modeling

由图 7 和图 8 可知，4 种算法所规划的航迹均可有效避开威胁区和山峰，并顺利到达目标点。但 PSO 算法和 SO 算法规划的航迹有过多无效转向，造成航迹的波动过大，表明 PSO 算法和 SO 算法在迭代前期陷入停滞，使最终航迹效果不佳。BEESO

算法和 OSSO 算法的航迹曲折更小，航迹长度更短，克服了算法前期陷入停滞的问题。相较于 BEESO 算法，OSSO 算法的航迹更平滑、航迹的质量更高。

由表 4 可知，OSSO 算法在最优值和平均值方面均优于 BEESO 算法、SO 算法和 PSO 算法。这表明，OSSO 算法具有更强的收敛速度和鲁棒性。可以在更短时间规划出平滑且能耗低的航迹。OSSO 算法相较于 2 种原始算法平价耗时较长，但大幅提升了算法搜索精度且耗时差较小。

表 4　函数建模的实验统计结果

Tab. 5　Statistical result of functional modeling

算法	最优值	最差值	平均值	平均耗时/s
PSO	970.361	972.085	971.248	**6.147**
SO	970.397	973.266	972.297	6.824
BEESO	966.495	967.258	966.983	10.716
OSSO	**963.072**	**963.993**	**963.497**	8.241

由图 9 可知，SO 算法生成初始解要优于 PSO 算法，但由于前期随机探索，算法会陷入局部极值，在第 33 代才逐渐克服影响，导致最终适应值接近 PSO 算法。PSO 算法在前期不断收敛，但在中期停滞，陷入局部极值。BEESO 算法前期也陷入了局部极值，但在第 27 代后快速收敛，并优于 SO 算法和 PSO 算法。OSSO 算法在前期短暂陷入停滞，在第 17 代后，呈阶梯状不断迭代收敛，最终适应度值优于其他 3 种算法，这表明 OSSO 算法解决了 SO 算法易陷入停滞的问题，且提高了算法的搜索精度和收敛速度。

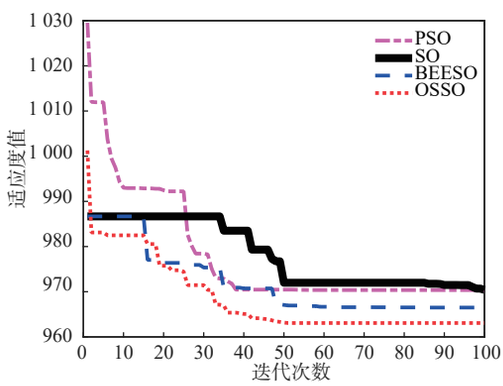


图 9　函数建模下适应度曲线

Fig. 9　Graph of fitness under functional modeling

3.3.2　高程数据模型

为了进一步验证 OSSO 算法的平衡全局探索和局部开发能力，因此引入真实地形的高程数据进行建模。任务空间设置为 1 000 m×1 000 m×400 m，起始点坐标为(200 m，100 m，75 m)，目标点坐标为(900 m，550 m，200 m)。给定的威胁区域参数如表 5 所示。

表 5　高程数据模型的威胁区域参数

Tab. 5　Threat area parameterized by DEM

威胁区中心 (x_t,y_t)/m	威胁区高度 H_t/m	威胁区半径 R_t/m
(350，500)	100	50
(600，200)	150	70
(500，350)	150	50
(350，200)	150	50
(700，550)	150	50
(650，750)	150	50
(800，400)	150	50
(300，350)	100	50
(500，600)	100	50

对比算法和 3.3.1 节保持一致，各算法种群数设置为 50，最大迭代次数为 1 500，航迹点个数为 15，每种算法单独运行 30 次。无人机的最大俯仰角和最大转弯角设为 45°，无人机飞行高度区间为 50～200 m，目标优化函数的权重系数 ω_1、ω_2、ω_3 分别为 0.4、0.3、0.3。

图 10 为无人机在高程模型下，4 种算法分别得到的无人机最优航迹，透明圆柱为威胁区域，图 11 为其俯视图。表 6 为高程模型的统计结果，图 12 为 4 种算法的适应度函数在迭代 1 500 次后的迭代曲线。

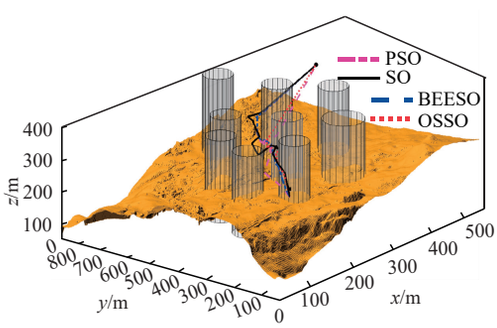


图 10　高程数据建模下航迹规划对比

Fig. 10　Comparison of track planning under DEM

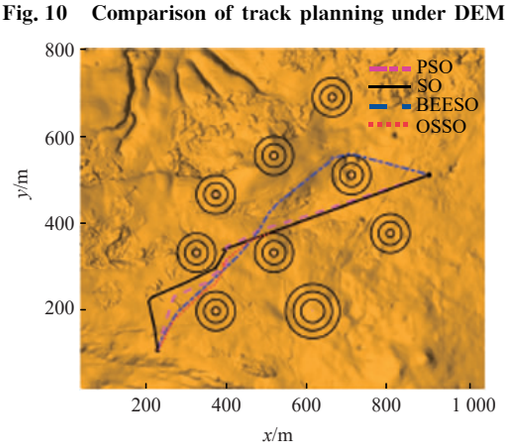


图 11　高程数据建模下航迹规划对比俯视图

Fig. 11　Top view of track planning comparison under DEM

由图 10 和图 11 可知，4 种算法均生成了有效航迹，但 PSO 算法和 SO 算法规划出的航迹过长，

转向波动严重，依然陷入局部最优。BEESO 算法规划出的航迹更加平滑，但为了规避威胁区的影响导致航迹过长。OSSO 算法规划出的航迹更短，在复杂环境中以更合适的转弯率和接近率通过威胁区环境，没有频繁的高度变化且航迹质量更高。

由表 6 可知，OSSO 算法的最优值和平均值均优于其他 3 种算法。平均值相较于 PSO 算法、SO 算法、BEESO 算法，分别提升了 28.44%、22.34%、11.07%。这说明 OSSO 算法所规划的航迹长度冗余少，算法的稳定性更好。结合图 12 可知，虽然 OSSO 算法耗时较长，但收敛速度最快，通过最少的迭代次数可求解出最小的目标函数值，因此可减少迭代次数以达成耗时减小的效果。

表 6 高程数据建模的实验统计结果

Tab. 6 Statistical result of DEM

算法	最优值	最差值	平均值	平均耗时/s
PSO	686.134	718.736	705.066	**73.439**
SO	644.098	667.283	649.656	76.961
BEESO	557.294	586.332	567.363	95.537
OSSO	**498.399**	**541.496**	**504.541**	83.182

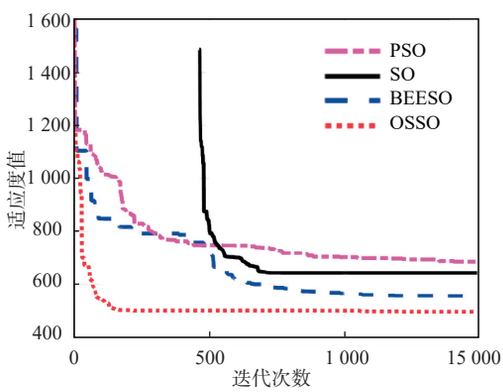


图 12 高程数据建模下适应度曲线

Fig. 12 Graph of fitness under DEM

由图 12 可知，PSO 算法前期收敛速度较快，但其寻优效果最差，并在第 434 次时陷入局部极值，直到 641 代才逐渐克服并缓慢寻优。SO 算法由于前期易陷入停滞，导致初期适应度值为∞，在克服影响后，快速收敛最终优于 PSO 算法。BEESO 算法收敛速度优于 PSO 算法，但在 94 代陷入局部极值，到 420 代后克服并继续快速收敛。OSSO 算法收敛速度优于其他 3 种算法，在 35 代左右便达到了 PSO 算法的寻优精度，在短暂停滞后迅速克服，使其搜索精度优于其他算法。这表明 OSSO 算法在处理复杂航迹规划问题时，具有一定的优越性，更符合实际飞行情况。

3.3.3 参数敏感性分析

为分析 OSSO 算法的优越性，因此对其进行参数敏感性分析实验，每次仅调整目标优化函数的参数权重占比，其余参数设置和 3.3.2 节一致，每种算法独立运行 30 次，实验统计结果如表 7 所示。

表 7 不同权重参数的实验统计结果

Tab. 7 Statistical result of various weight parameters

参数值	算法	平均值
$\omega_1=0.8$ $\omega_2=0.1$ $\omega_3=0.1$	PSO	1 052.56
	SO	980.73
	BEESO	975.84
	OSSO	**963.53**
$\omega_1=0.1$ $\omega_2=0.8$ $\omega_3=0.1$	PSO	203.17
	SO	225.21
	BEESO	191.68
	OSSO	**181.93**
$\omega_1=0.1$ $\omega_2=0.1$ $\omega_3=0.8$	PSO	377.79
	SO	343.83
	BEESO	**243.14**
	OSSO	260.05

由表 7 可知，航迹长度代价对目标优化函数值的变化影响最大，4 种算法中 PSO 算法对航迹长度代价的敏感度最低，因此规划出的航迹质量较差。BEESO 算法对无人机性能约束最为敏感，在规划航迹中会优先考虑性能约束，导致生成航迹较为冗长。OSSO 算法表现最均衡，规划出的航迹安全且平滑，这表明了 OSSO 算法在处理复杂三维航迹规划问题的有效性。

4 结语

本文提出了基于鱼鹰搜索策略蛇优化算法以解决无人机三维航迹规划问题。在原始 SO 算法的基础上，引入了 Bernoulli 混沌映射、鱼鹰搜索策略和动态折射反向学习等多种策略，提升算法的收敛速度和搜索能力，并提高了算法局部开发能力和多样性。仿真实验表明，OSSO 算法在三维无人机航迹规划过程中，能够以代价值最小的前提下快速穿过威胁区域，并稳定规划出安全且合理的航迹。

下一步将着重研究 OSSO 算法在求解多无人机协同航迹规划问题的研究，构建合理的协同目标函数，在保证多无人机之间不碰撞且同时抵达目标点，将更真实的仿真模型应用于三维航迹规划问题中。

参考文献

[1] AIT S A, SOUKANE A, MERAIHI Y, et al. UAV Path Planning Using Optimization Approaches: A Survey[J]. Archives of Computational Methods in Engineering, 2022, 29(6): 4233-4284.

[2] 隋东，杨振宇，丁松滨，等. 基于 EMSDBO 算法的无人机三维航迹规划[J]. 系统工程与电子技术，2024，46(5)：1756-1766.
SUI D，YANG Z Y，DING S B，et al. Three-Dimensional Path Planning of UAV Based on EMSDBO Algorithm[J]. Systems Engineering and Electronics，2024，46(5)：1756-1766.(in Chinese)

[3] 赵畅，刘允刚，陈琳，等. 面向元启发式算法的多无人机路径规划现状与展望[J]. 控制与决策，2022，37(5)：1102-1115.
ZHAO C，LIU Y G，CHEN L，et al. Research and Development Trend of Multi-UAV Path Planning Based on Metaheuristic Algorithm[J]. Control and Decision，2022，37(5)：1102-1115.(in Chinese)

[4] BAEK J，HAN S I，HAN Y. Energy-Efficient UAV Routing for Wireless Sensor Networks[J]. IEEE Transactions on Vehicular Technology，2020，69(2)：1741-1750.

[5] LU Y C，XUE Z C，XIA G S，et al. A Survey on Vision-Based UAV Navigation[J]. Geo-spatial Information Science，2018，21(1)：21-32.

[6] ZHANG S W，ZHANG R. Radio Map-Based 3D Path Planning for Cellular-Connected UAV[J]. IEEE Transactions on Wireless Communications，2021，20(3)：1975-1989.

[7] WU X L，XU L，ZHEN R，et al. Bi-Directional Adaptive A* Algorithm Toward Optimal Path Planning for Large-Scale UAV under Multi-Constraints[J]. IEEE Access，2020，8：85431-85440.

[8] 陈海洋，王露楠. 基于双向同时无碰撞检测目标偏置 RRT 算法的路径规划方法[J]. 空军工程大学学报(自然科学版)，2022，23(3)：60-67.
CHEN H Y，WANG L N. A Path Planning Algorithm Based on Two-Way Simultaneous No-Collision Goal RRT[J]. Journal of Air Force Engineering University (Natural Science Edition)，2022，23(3)：60-67.(in Chinese)

[9] 巫茜，罗金彪，顾晓群，等. 基于改进 PSO 的无人机三维航迹规划优化算法[J]. 兵器装备工程学报，2021，42(8)：233-238.
WU Q，LUO J B，GU X Q，et al. Optimization Algorithm for UAV 3D Track Planning Based on Improved PSO Method[J]. Journal of Ordnance Equipment Engineering，2021，42(8)：233-238.(in Chinese)

[10] SEYYEDABBASI A，KIANI F. Sand Cat Swarm Optimization：A Nature-Inspired Algorithm to Solve Global Optimization Problems[J]. Engineering with Computers，2023，39(4)：2627-2651.

[11] 鲁亮亮，代冀阳，应进，等. 基于 APSODE-MS 算法的无人机航迹规划[J]. 控制与决策，2022，37(7)：1695-1704.
LU L L，DAI J Y，YING J，et al. UAV Trajectory Planning Based on APSODE-MS Algorithm[J]. Control and Decision，2022，37(7)：1695-1704.(in Chinese)

[12] 王康，司鹏，陈莉，等. 基于改进沙猫群算法的无人机三维航迹规划[J]. 兵工学报，2023，44(11)：3382-3393.
WANG K，SI P，CHEN L，et al. 3D Path Planning of Unmanned Aerial Vehicle Based on Enhanced Sand Cat Swarm Optimization Algorithm[J]. Acta Armamentarii，2023，44(11)：3382-3393.(in Chinese)

[13] HASHIM F A，HUSSIEN A G. Snake Optimizer：a Novel Meta-Heuristic Optimization Algorithm[J]. Knowledge-Based Systems，2022，242：108320.

[14] BEHJATI M，NORDIN R，ZULKIFLEY M A，et al. 3D Global Path Planning Optimization for Cellular-Connected UAVs under Link Reliability Constraint[J]. Sensors，2022，22(22)：8957.

[15] DEHGHANI M，TROJOVSK P. Osprey Optimization Algorithm：A New Bio-Inspired Metaheuristic Algorithm for Solving Engineering Optimization Problems[J]. Frontiers in Mechanical Engineering，2023，8：1126450.

[16] HU G，YANG R，ABBAS M，et al. BEESO：Multi-Strategy Boosted Snake-Inspired Optimizer for Engineering Applications[J]. Journal of Bionic Engineering，2023，20(4)：1791-1827.

[17] 刘景森，吉宏远，李煜. 基于改进蝙蝠算法和三次样条插值的机器人路径规划[J]. 自动化学报，2021，47(7)：1710-1719.
LIU J S，JI H Y，LI Y. Robot Path Planning Based on Improved Bat Algorithm and Cubic Spline Interpolation[J]. Acta Automatica Sinica，2021，47(7)：1710-1719.(in Chinese)

(编辑：杜娟)