Computer Science and Application 计算机科学与应用, 2023, 13(7), 1363-1372
Published Online July 2023 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2023.137134

Hans汉斯

融合随机森林决策的海鸥优化算法

张天颖，刘　萍，吕佳硕

河北地质大学信息工程学院，河北　石家庄

收稿日期：2023年6月11日；录用日期：2023年7月7日；发布日期：2023年7月13日

摘　要

针对海鸥优化算法(SOA)在求解优化问题时容易早熟、算法性能过于依赖参数以及解的精确度较低等问题，提出了一种融合随机森林决策的海鸥优化算法(RFSOA)。首先，在海鸥前期迁徙阶段，引入非线性递减参数A对海鸥位置进行改变，调整算法的全局寻优能力，避免在迭代过程前期算法便会陷入局部最优的情况；其次，在海鸥攻击阶段，引入机器学习中随机森林思想，将每个海鸥个体位置看作任一决策树的分支，利用其构造决策树进行海鸥位置的改变，位置变化时采取螺旋状位置更新策略或最优高斯游走策略，增加海鸥位置改变的随机性，并利用贪心策略保留优质个体，从而提高寻优精度。选择Benchmark测试函数及0-1背包问题部分算例进行算法性能测试，结果显示RFSOA具有寻优能力强、精确度高的优点，更适合求解较高维度的连续函数优化问题及带有约束的实际应用问题。

关键词

海鸥优化算法，非线性参数，随机森林，高斯随机游走，Benchmark基准测试函数

A Seagull Optimization Algorithm Incorporating Random Forest Decision Making

Tianying Zhang, Ping Liu, Jiashuo Lyu

College of Information Engineering, Hebei GEO University, Shijiazhuang Hebei

Received: Jun. 11th, 2023; accepted: Jul. 7th, 2023; published: Jul. 13th, 2023

Abstract

A seagull optimization algorithm (RFSOA) that incorporates random forest decision making is proposed to address the problems that the seagull optimization algorithm (SOA) tends to be pre-

文章引用: 张天颖, 刘萍, 吕佳硕. 融合随机森林决策的海鸥优化算法[J]. 计算机科学与应用, 2023, 13(7): 1363-1372.
DOI: 10.12677/csa.2023.137134

mature in solving optimization problems, the performance of the algorithm is too dependent on parameters, and the accuracy of the solution is low. Firstly, in the pre-migration stage of the seagull, a nonlinear decreasing parameter A is introduced to change the position of the seagull to adjust the global optimization capability of the algorithm and avoid the situation that the algorithm will fall into the local optimum at the early stage of the iterative process; secondly, in the attack stage of the seagull, the random forest idea in machine learning is introduced to consider each individual position of the seagull as a branch of any decision tree and use it to construct a decision tree to change the position of the seagull. The spiral position update strategy or optimal Gaussian wandering strategy is adopted when the position changes to increase the randomness of the gull position change, and the greedy strategy is used to retain the high quality individuals, so as to improve the optimization seeking accuracy. Benchmark test function and some examples of 0-1 knapsack problem are selected to test the performance of the algorithm. The results show that RFSOA is more suitable for solving continuous function optimization problems with higher dimensionality and practical application problems with constraints.

Keywords

Seagull Optimization Algorithm, Nonlinear Parameters, Random Forest, Gaussian Random Walk, Benchmark Function

1. 引言

受自然界生物群体行为的启发，诸多群智能算法被提出并用于求解优化问题，如粒子群优化算法(Particle Swarm Optimization, PSO) [1]、鲸鱼优化算法(Whale Optimization algorithm, WOA) [2]、哈里斯鹰优化算法(Harris Hawks Optimization, HHO) [3]、黑猩猩优化算法(Chimp Optimization Algorithm, ChOA) [4]、海洋捕食者算法(Marine Predators Algorithm, MPA) [5]等。

海鸥优化算法(Seagull Optimization Algorithm, SOA) [6]是由 Dhiman 和 Kumar 教授在 2019 年提出的一种新的元启发式算法，海鸥优化算法参数较为简单容易实现，目前常被用于求解较大规模的优化类问题，但是在迭代寻优过程中也存在倾向于局部最优、算法性能依靠参数以及寻优的精确度低等缺陷。为了解决海鸥优化算法存在的缺陷，提高算法性能，诸多学者在海鸥优化算法的基础上提出改进策略。

毛清华等[7]提出一种融合改进 Logistics 混沌和正弦余弦算子的自适应 t 分布海鸥算法(ISOA)，采用 Logistics 混沌映射对种群进行初始化，并引入正弦余弦算子以及非线性参数 A 来对算法的全局搜索和局部搜索进行协调，从而加快算法收敛速度，在最优解位置利用自适应 t 分布变异策略进行扰动产生新解，增强了算法跳出局部最优的能力。秦维娜[8]等提出一种基于惯性权重的海鸥优化算法，该算法使用非线性递减的惯性权重对海鸥的位置进行调整，并使用莱维飞行和随机指数值增加海鸥飞行的随机性，从而提高算法寻找最佳方案的全局能力。He [9]等提出的基于反向的海鸥优化算法(OSOA)，将特征选择技术与 SOA 相结合，其总体由基于反向的学习(OBL)算法确定，测试结果证明该算法具有更高的精度和计算效率。Jia H [10]等提出了一种特征选择混合海鸥优化算法，将海鸥优化算法与热交换优化(TEO)算法进行混合，采用 TEO 算法的热交换公式来改进 SOA 的海鸥攻击模式，从而有效的提高了 SOA 的开发能力。

为了进一步提升海鸥优化算法的性能，本文提出一种融合随机森林决策的海鸥优化算法(RFSOA)。该算法在迁徙阶段，引入非线性参数 A，从而提高算法对于搜索空间全局及局部的平衡性；在攻击阶段利用随机森林思想，利用决策树计算海鸥个体的待攻击位置，位置变化过程采取螺旋更新或高斯随机游走策略，增加海鸥位置改变的随机性，最后利用贪心策略保留最优个体，从而提高算法寻优精度。

2. 标准海鸥优化算法

海鸥是遍布全球的候鸟，主要以群居式生活，擅长利用智慧来寻找和攻击猎物。

其最重要的运动行为是迁徙和攻击，迁徙通常被定义为动物为了搜寻更为丰富的食物来源从而获得充足的能量而从一个区域移动到另一个区域的生理需要。在迁移期间，动物呈集群式出行。在进行迁徙时，每只海鸥所处的飞行位置都不同，以避免相互之间发生碰撞。在一个群体中，会有一个最佳位置方向，每个海鸥都会朝着这个方向移动，从而导致自身所在的位置发生改变。海鸥经常会攻击候鸟，在这种情况下，运动群体的位置会围绕螺旋形轨迹发生变化[11]。

2.1. 迁徙(全局搜索)

在进行迁移时，算法模拟处于集群中的海鸥个体如何从一个位置移动到另一个位置。在这一过程中，海鸥应该满足三个条件：

1) 避免碰撞：为了避免与邻居(其他海鸥)碰撞，该算法使用一个额外的变量 A 来进行对于海鸥新位置的计算。

$$C_s(t) = A \times P_s(t) \tag{1}$$

$C_s(t)$ 表示不会和其他海鸥发生位置冲突从而导致碰撞的新位置，$P_s(t)$ 表示海鸥目前所在的位置，t 表示当前迭代，A 表示海鸥在特定搜索空间中移动时的行为。

$$A = f_c - t \times \left(f_c / Max_{iteration} \right) \tag{2}$$

f_c 可以控制变量 A 随迭代次数的改变而发生改变，它的值从 2 线性降低到 0。

2) 最佳位置方向：在避免了与群体中的其他海鸥个体发生位置碰撞冲突之后，海鸥会寻找一个目前相对个体最佳的位置方向，然后朝着其所处的方向移动。

$$M_s(t) = B \times \left(P_{bs}(t) - P_s(t) \right) \tag{3}$$

$M_s(t)$ 表示需要朝着其移动的最佳位置所在的方向，B 是负责平衡全局和局部搜索的随机数。

$$B = 2 \times A^2 \times r_d \tag{4}$$

r_d 是[0, 1]范围内的随机数。

3) 靠近最佳位置：当海鸥移动到不与其他海鸥发生位置冲突的位置并且寻找到最佳位置方向之后，就会朝着该方向进行移动，从而使其位置发生改变。

$$D_s(t) = \left| C_s(t) + M_s(t) \right| \tag{5}$$

$D_s(t)$ 是海鸥在整个迁移改变位置过程中的位移。

2.2. 攻击(局部搜索)

海鸥在迁徙过程中用翅膀和重量保持高度，攻击角度和速度会不断发生改变。当进行对猎物的攻击时，它们就在空中整体的运动轨迹呈螺旋形状变化。利用 x、y 和 z 平面描述的运动行为如下：

$$\begin{cases} x = r \times \cos(\theta) \\ y = r \times \sin(\theta) \\ z = r \times \theta \\ r = u \times e^{\theta v} \end{cases} \tag{6}$$

其中 r 是每个螺旋的半径，θ 是[0, 2π]范围内的随机角度值。u 和 v 是螺旋形状的相关常数，e 是自然对数的底数。

$$P_s(t) = P_{bs}(t) + D_s(t) \times x \times y \times z \tag{7}$$

$P_s(t)$ 是海鸥的攻击位置。

综上所述，进行位置变换之后的海鸥粒子位置如下所示：

$$P_s(t) = P_{bs}(t) + \left| B \times P_{bs}(t) - (B - A) \times P_s(t) \right| \times x \times y \times z \tag{8}$$

3. 融合随机森林决策的海鸥优化算法

3.1. 非线性参数 A

标准海鸥优化算法中引入辅助变量 A 作为避免海鸥发生位置碰撞的变量，A 的值随着 f_c 发生变化，A 的取值从 2 线性下降到 0，导致对于解空间的搜索呈线性收敛。经过测试发现，呈线性收敛的 A 并不适合算法对于非线性空间中解的有效搜索。为了进一步提高算法对于解的全局及局部搜索的平衡能力，将 A 设置成非线性参数，由公式(9)描述：

$$A = \frac{A_{\text{start}} - A_{\text{final}}}{1 + e^{\frac{2t - T_{\max}}{T_{\max}} * 5}} \tag{9}$$

t 代表当前迭代，$T_{\max}$ 代表最大迭代次数。

如图 1 所示，改进后的参数 A 呈非线性下降，在算法迭代前期，A 的值较大，因此算法具有较好的全局搜索能力；在迭代后期，A 的值迅速下降，使得算法具有较强的局部搜索能力。相比于标准 SOA 中线性下降的参数 A，改进后的 A 有效平衡了算法的全局及局部搜索，理论上可以有效提升算法性能。

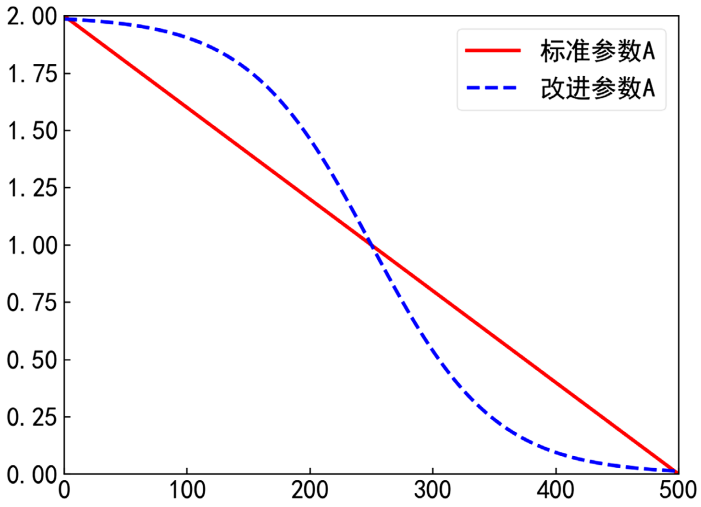

Figure 1. Variation curve of parameter A with the number of iterations
图 1. 参数 A 随迭代次数的变化曲线

3.2. 基于随机森林的位置更新

随机森林算法中采用的 bootstrap 重采样技术是一种有放回的随机采样，从原始的训练样本集中随机有放回地抽取与其等数量的样本组成 1 个采样集，重复 n 轮得到 n 个相互独立的采样集；然后利用每个采样集分别生成决策树，n 个决策树连接形成“森林”[12]。

将该机制引入海鸥优化算法的种群更新，应用随机森林构建多个决策树同时进行决策，最终取最优结果。

设种群内有个体数量 n，利用 $P=\left(P_1,P_2,\cdots,P_n\right)$ 表示种群个体。随机且有放回的取其中 n 个个体构成数据集 1，利用 $TP_1=\left(TP_{11},TP_{12},\cdots,TP_{1n}\right)$ 表示，记其中不重复的个体数量为 r_1；数据集 2 同理，利用 $TP_2=\left(TP_{21},TP_{22},\cdots,TP_{2n}\right)$ 表示，其中不重复的个体数量为 r_2；数据集 3 中包含的个体为种群 P 除 TP_1,TP_2 以外剩余的个体，个体数量为 $n-r_1-r_2$。分别将数据集 1、数据集 2 及数据集 3 的个体视为其对应决策树的分支。对第一棵决策树包含的 r_1 个个体选择第一种位置更新策略，即公式(10)螺旋状更新策略；对第二棵决策树包含的 r_2 个个体选择第二种位置更新策略，即公式(11)高斯随机游走策略；对第三棵决策树包含的个体进行随机概率取值，每个个体对应一个随机数 ρ，$\rho\in(0,1)$，若 $0<\rho\le 0.5$，则该个体采用公式(10)进行位置更新；若 $0.5<\rho<1$，则利用公式(11)进行位置更新。三棵决策树共同进行决策，判断个体位置变化后适应值优劣，并利用贪心策略保留适应值优越的个体。

3.2.1. 位置更新策略

a) 按照标准 SOA 中螺旋状位置更新进行。

$$P_s(t)=P_{bs}(t)+\left|B\times P_{bs}(t)-(B-A)\times P_s(t)\right|\times x\times y\times z \tag{10}$$

b) 采用最优高斯游走策略进行位置更新。

高斯随机游走模型是最具有代表性的随机游走模型之一，具有较高的探索能力。因此，以当前的最优个体作为引导，利用高斯随机游走的特点，产生新的个体，可以平衡算法在进行探索与开发时的能力。下式为基于当前最优个体的高斯随机游走策略：

$$P_s(t)=Gaussian\left(P_{bs}(t),\tau\right)+\left|B\times P_{bs}(t)-(B-A)\times P_s(t)\right| \tag{11}$$

步长 τ 表示为

$$\tau=\frac{\log(t)}{t}\left(p_s(t)-p_{bs}(t)\right) \tag{12}$$

通过当前最优个体的引导，种群会产生接近最优个体的新个体，从而使算法较快的发生收敛；同时，利用高斯分布的方差 τ 对给定空间中的搜索进行自适应控制，在算法迭代初期，t 的值偏小，τ 的值较大，因此算法具有较强的全局探索能力；进入到迭代后期之后，随着迭代次数的不断增加，τ 的值逐渐减小，从而算法具有了较强的局部开发能力。

3.2.2. 贪心策略保留优质个体

利用式(13)贪心策略对每次迭代进行位置变换之后的海鸥个体进行选择保留，具有较高适应值的个体将被保留下来，避免了标准 SOA 中位置更新之后产生并保留劣质个体的情况，从而使得迭代过程可以拥有相对较高的平均适应值，同时个体分布更加广泛，有助于提高寻找整体最优值的效率，降低了过早地发生局部收敛而无法跳出的概率。

$$P_s(t+1)=\begin{cases}P_s(t) & \text{if } fit\left(P_s(t)\right)>fit\left(P_s(t+1)\right)\\ P_s(t+1) & \text{else}\end{cases} \tag{13}$$

3.3. 算法描述

算法 1. 融合随机森林决策的海鸥优化算法(RFSOA)：

输入：种群规模 n，迭代次数 $Maxiter$

输出：最优解 X_{best}

步骤 1. 初始化算法问题参数、迭代次数以及种群规模；

步骤 2. 随机生成初始化种群，产生初始个体编码与海鸥位置编码；

步骤 3. 计算种群个体适应值，获得当前种群中最优个体位置 $P_{best}(t)$；

步骤 4. 利用公式(9)计算参数 A，公式(1)~(5)计算海鸥迁移最佳位置方向 $M_s(t)$ 以及 $D_s(t)$；

步骤 5. 采用随机森林策略，利用公式(10)以及(11)进行海鸥攻击位置变化，得到 $P_s(t)$；计算 $P_s(t)$ 适应值，利用式(13)保留适应值较高的个体；

步骤 6. 判断是否到达迭代停止条件，如果未到达，则返回执行步骤 3~5；

步骤 7. 停止迭代，输出最优适应值以及其对应最优解 X_{best}。

以上步骤对应具体流程如图 2 所示。

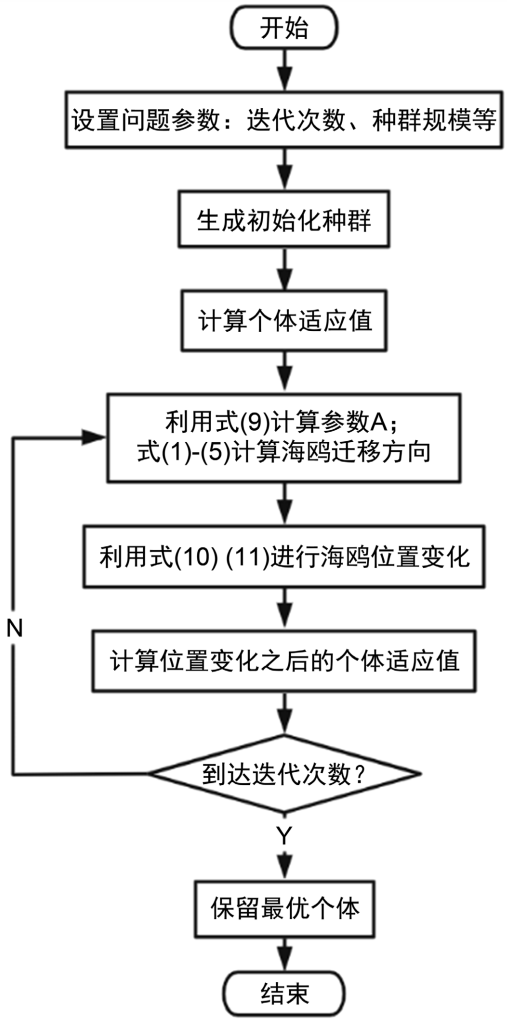

Figure 2. RFSOA specific process
图 2. RFSOA 具体流程

4. 实验与结果分析

4.1. 无约束函数测试

仿真实验的计算机环境为：Windows11，CPU 2.70GHz 处理器，8 GB 内存，应用 python 对算法编码。

为了验证 RFSOA 的性能优劣，选择 7 个 Benchmark 基准测试函数来测试算法的性能，其中 f1~f4 为单峰函数，f5~f7 为多峰函数，具体信息列于表 1 中。以粒子群算法 PSO，遗传算法 GA，基于非线性惯性权重的海鸥优化算法 I-SOA 作为对比算法。RFSOA 的种群规模设置 40，问题维度为 30，算法的迭代次数为 500。将所有算法均独立运行 30 次，以消除偶然因素的影响，取解均值及标准差进行对比，如表 2 所示。

Table 1. Benchmark functions
表 1. Benchmark 基准测试函数

类型	序号	函数名称	搜索域	最优值
Unimodal functions	f1	Sphere	[−100, 100]	0
	f2	Schwefel 2.22	[−10, 10]	0
	f3	Schwefel 1.2	[−100, 100]	0
	f4	Schwefel 2.21	[−100, 100]	0
Simple Multimodal Functions	f5	Generalized Rastrigin	[−5.12, 5.12]	0
	f6	Ackley	[−32, 32]	0
	f7	Generalized Griewand	[−100, 100]	0

Table 2. Comparison with other algorithm tests
表 2. 与其他算法测试对比

函数 30 维	RFSOA		SOA		PSO		GA		I-SOA	
	解均值	标准差	解均值	标准差	解均值	标准差	解均值	标准差	解均值	标准差
f1	9.16E−08	4.75E−07	3.87E+01	8.72E+01	1.51E+00	1.23E+00	5.41E+00	3.09E+00	3.30E−04	1.43E−03
f2	6.43E−09	3.69E−09	1.83E+00	1.27E+00	2.64E−01	1.40E−01	2.06E+00	1.39E+00	2.13E−04	6.40E−04
f3	2.97E−12	8.92E−13	1.43E+05	3.06E+04	8.94E+03	2.31E+03	9.82E+03	1.65E+04	9.65E+04	2.03E+04
f4	4.81E−09	5.52E−10	4.72E+01	2.63E+01	2.09E+01	3.90E+00	6.53E+01	8.52E+00	5.72E+01	3.35E+01
f5	8.56E−10	2.82E−09	1.56E+01	2.17E+01	7.99E+01	1.75E+01	9.41E+01	2.82E+01	1.28E+00	5.64E+00
f6	4.74E−09	3.47E−09	1.38E−01	5.92E−01	1.62E+00	6.97E−01	1.86E+01	5.10E−01	1.21E−05	6.47E−05
f7	1.54E−14	7.64E−13	1.75E+00	1.94E+00	9.28E−01	1.38E−01	1.25E+00	9.98E−02	4.37E−03	1.16E−02

由表 2 数据可以得到，在维度 30 的情况下，RFSOA 在对 7 个 Benchmark 函数的测试中展示出来的总体效果较好。相比于标准 SOA，改进后的 RFSOA 所寻得的最优解及稳定性均有较大提升，更加接近理论最优。同时由测试结果可以看出，RFSOA 针对测试函数所展示出来的效果也明显优于参与对比的 PSO、GA 及 I-SOA。因此证明了改进后的 RFSOA 针对无约束优化函数的求解的确具有更好的效果。

4.2. 0-1 背包问题测试

背包问题(Knapsack Problem, KP) [13] [14]是由 Markel 和 Hellman 提出的一个经典的组合优化问题。0-1 背包问题(0-1 knapsack problem, 0-1 KP)可以描述为：给定一个背包和 n 种物品，每个物品都有两个参

数：重量和价值，背包有自己的限重，选择部分物品将其装入背包中，在满足背包限重的条件下，使得装入的物品价值最高。

设共有 n 种物品，$i(1 \le i \le n)$ 号物品的价值与重量分别为 v_i 和 w_i，背包的重量限制为 C，其中，v_i、w_i 与 C 均为正数。令 $X=[x_1,x_2,\cdots,x_n] \in \{0,1\}^n$ 表示 0-1 背包问题的一个潜在解向量，其中，当 $x_i=1$ 时，表示 i 号物品被装入背包中；当 $x_i=0$ 时，表示 i 号物品不装入背包。0-1KP 的数学模型为：

$$\max f(X) = \max \sum_{x=1}^{n} v_i x_i \tag{14}$$

$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \le C \tag{15}$$

在上述模型中，任意一个 n 维的 0-1 向量都是 0-1KP 的一个潜在解，当其满足约束条件(15)式时，才能称为 0-1KP 的一个可行解。

利用引入随机森林思想的海鸥优化算法(RFSOA)对 0-1KP 进行求解，测试数据以及对比数据均来源于文献[15]，同样将算法的种群规模设置为 30，最大迭代次数设置为 200，算法独立运行次数为 20。

由表 3 测试结果可以看出，RFSOA 针对 0-1KP 问题具有较好的求解效果，针对文献所给测试数据，其寻优效果达到 100%。对于较高维度的 kp4、kp5 以及 kp6 三组数据，RFSOA 获得的最优解相比于 BSOA 以及 BCSA 也展示出了较明显的优势，相比于标准 SOA，其最优解及平均解提升最高达到了 16.1%和 20.6%。因此证明 RFSOA 针对带有约束的优化问题求解也具有一定提升效果。

Table 3. Comparison of test results of algorithms on 0-1KP problems
表 3. 算法在 0-1KP 问题上测试结果对比

算例	维度 (理论最优)	算法	最优值	最差值	平均值
Kp1	50 (3119)	SOA	3051	2948	2993.5
		BSOA	3097	2999	3026.6
		BCSA	3104	2965	3015.9
		RFSOA	3119	3119	3119.0
Kp2	50 (3103)	SOA	3016	2905	2987.2
		BSOA	3083	2973	3066.4
		BCSA	3054	2938	3006.5
		RFSOA	3103	3103	3103.0
Kp3	60 (1563)	SOA	1426	1259	1295.7
		BSOA	1463	1188	1267.1
		BCSA	1496	1480	1487.2
		RFSOA	1563	1563	1563.0
Kp4	100 (2660)	SOA	2337	2276	2292.4
		BSOA	2379	2315	2355.4
		BCSA	2510	2309	2400.3
		RFSOA	2660	2641	2656.9

Continued

Kp5	200 (5164)	SOA	4539	4392	4451.8
		BSOA	4875	4803	4869.4
		BCSA	4427	4146	4285.3
		RFSOA	5164	5128	5143.6
Kp6	100 (26934)	SOA	25,891	24,728	25036.3
		BSOA	26,029	25,338	26009.7
		BCSA	26,188	23,200	25102.4
		RFSOA	26,934	26,907	26925.3

5. 结束语

为了克服标准 SOA 存在的容易早熟、算法性能过于依赖参数以及解的精确度较低等缺陷，本文提出了一种融合随机森林策略的海鸥优化算法(RFSOA)，通过引入非线性参数、随机森林策略以及高斯随机游走策略对 SOA 进行改进，从而更好的平衡算法对于搜索空间全局及局部的探索。针对 Benchmark 函数以及 0-1KP 的一系列测试实验表明，改进后的 RFSOA 在对函数以及实际应用问题进行求解时，可以有效地解决算法过早陷入局部最优值的缺陷，从而找到更具有优越性的解，同时算法鲁棒性得到了很大提升，证明了 RFSOA 的有效性。

在未来的工作中，仍需进一步提升 SOA 的性能，并利用 SOA 对更为复杂的实际应用进行研究，扩展其应用范围。

参考文献

[1] Kennedy, J. and Eberhart, R. (2002) Particle Swarm Optimization. Icnn95-International Conference on Neural Networks. *Proceedings of the IEEE International Conference on Neural Networks*, **4**, 1942-1948.

[2] Mirjalili, S. and Lewis, A. (2016) The Whale Optimization Algorithm. *Advances in Engineering Software*, **95**, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

[3] Heidari, A.A., Mirjalili, S., Faris, H., *et al*. (2019) Harris Hawks Optimization: Algorithm and Applications. *Future Generation Computer Systems*, **97**, 849-872. https://doi.org/10.1016/j.future.2019.02.028

[4] Khishe, M. and Mosavi, M.R. (2020) Chimp Optimization Algorithm. *Expert Systems with Applications*, **149**, Article ID: 113338. https://doi.org/10.1016/j.eswa.2020.113338

[5] Faramarzi, A., Heidarinejad, M., Mirjalili, S., *et al*. (2020) Marine Predators Algorithm: A Nature-Inspired Metaheuristic. *Expert Systems with Applications*, **152**, Article ID: 113377. https://doi.org/10.1016/j.eswa.2020.113377

[6] Gaurav, D. and Kumar, V. (2019) Seagull Optimization Algorithm: Theory and Its Applications for Large-Scale Industrial Engineering Problems. *Knowledge-Based Systems*, **165**, 169-196. https://doi.org/10.1016/j.knosys.2018.11.024

[7] 毛清华, 王迎港. 融合改进 Logistics 混沌和正弦余弦算子的自适应 t 分布海鸥算法[J]. 小型微型计算机系统, 2022, 43(11): 2271-2277.

[8] 秦维娜, 张达敏, 尹德鑫, 等. 一种基于非线性惯性权重的海鸥优化算法[J]. 小型微型计算机系统, 2022, 43(1): 10-14.

[9] He, J., Ye, Y., Ping, W., *et al*. (2020) A Novel Hybrid Classification Method Based on the Opposition-Based Seagull Optimization Algorithm. *IEEE Access*, **8**, 100778-100790. https://doi.org/10.1109/ACCESS.2020.2997791

[10] Jia, H., Xing, Z. and Song, W. (2019) A New Hybrid Seagull Optimization Algorithm for Feature Selection. *IEEE Access*, **7**, 2169-3536. https://doi.org/10.1109/ACCESS.2019.2909945

[11] 韩毅, 徐梓斌, 张亮, 等. 国外新型智能优化算法——海鸥优化算法[J]. 现代营销(经营版), 2019(10): 70-71.

[12] 范昊东. 基于灰狼优化的随机森林模型的研究[J]. 电子测试, 2022, 36(6): 45-47.

[13] Kellerer, H., Pferschy, U. and Pisinger, D. (2004) Knapsack Problems. Springer-Verlag, Berlin, 1-445. https://doi.org/10.1007/978-3-540-24777-7

[14] Merkel, R. and Hellman, M. (1978) Hiding Iformation and Signatures in Trapdoor Knapsacks. *IEEE Transactions on Iformation Theory*, **24**, 525-530. https://doi.org/10.1109/TIT.1978.1055927

[15] 刘雪静, 贺毅朝, 吴聪聪, 等. 求解 0-1 背包问题的混沌二进制乌鸦算法[J]. 计算机工程与应用, 2018, 54(10): 173-179.