DOI:10.11918/202111074

基于正交设计的折射反向学习樽海鞘群算法

王宗山[1],丁洪伟[1],王 杰[2],李 波[1],侯 鹏[3],杨志军[1,4]

(1. 云南大学 信息学院,昆明 650500;2. 郑州大学 机械与动力工程学院,郑州 450066;
3. 复旦大学 计算机科学技术学院,上海 201203;4. 云南省教育厅 教育科学研究院,昆明 650021)

摘 要:为克服基本樽海鞘群算法(SSA)存在的收敛速度慢、高维求解精度低等不足,提出正交折射反向学习机制和自适应惯性权重策略,嵌入 SSA 中,得到一种基于正交设计的折射反向学习樽海鞘群算法(OOSSA)。正交折射反向学习策略中,采用基于透镜成像原理的折射反向学习策略以加强对反向解空间的勘探,极大地降低了算法陷入局部最优的概率;采用正交试验设计构建若干部分维上取折射反向值的部分反向解,深度挖掘并保存当前个体和折射反向个体的优势维度信息。此外,在跟随者位置更新阶段引入惯性权重因子,有效地改善跟随者的搜索模式并增强算法的局部开采能力。采用 CEC2017 基准函数进行仿真实验,同时使用 Wilcoxon 秩和检验、Friedman 检验等方法来评价 OOSSA 算法的优化性能,测试结果表明所提算法的寻优精度和收敛速度明显优于基本 SSA 算法、8 种新近的改进 SSA 算法和 9 种前沿的群体智能优化算法。此外,将所提算法应用于一个工程设计问题,结果表明该算法在工程优化方面的性能优于对比算法。最后,针对求解自主移动机器人路径规划问题,提出一种基于 OOSSA 的路径规划算法。在 3 种环境设置下对所提算法进行仿真实验,并与 PSO、ABC、GWO、FA 和 SSA 等算法进行对比。仿真结果表明,本文算法能够规划出最优的无碰撞路径。系统的实验表明 OOSSA 算法可作为问题优化的有效工具。

关键词:樽海鞘群算法;透镜折射学习;正交试验设计;自适应学习;基准函数;工程优化;路径规划
中图分类号:TP301.6 **文献标志码**:A **文章编号**:0367-6234(2022)11-0122-15

Salp swarm algorithm based on orthogonal refracted opposition-based learning

WANG Zongshan[1], DING Hongwei[1], WANG Jie[2], LI Bo[1], HOU Peng[3], YANG Zhijun[1,4]

(1. School of Information Science and Engineering, Yunnan University, Kunming 650500, China;
2. School of Mechanical and Power Engineering, Zhengzhou University, Zhengzhou 450066, China;
3. School of Computer Science, Fudan University, Shanghai 201203, China; 4. Educational and Scientific Institute, Education Department of Yunnan Province, Kunming 650021, China)

Abstract: The basic salp swarm algorithm (SSA) may suffer from the drawbacks of slow convergence and low accuracy of high-dimensional solutions. To solve these limitations, we proposed an improved SSA algorithm (OOSSA) which integrates orthogonal refracted opposition-based learning strategy and self-adaptive inertia weight strategy into SSA. In orthogonal refracted learning strategy, the refracted opposition-based learning based on the optical lens imaging principle was employed to enhance the exploration scope of the inverse solution space, which greatly reduced the probability of the algorithm falling into the local optimum. The orthogonal experimental design was used to construct several partial opposite solutions that take the refracted-based inverse values in part of the dimension, so as to deeply mine and preserve the dominant dimensional information of the current individual and the refracted-based opposite individual. In addition, an adaptive inertia weight was introduced in the follower position update phase to effectively improve the follower search pattern and enhance the local exploitation ability of the algorithm. The CEC2017 benchmark functions were employed for simulation experiments. Also, Wilcoxon's rank-sum test and Friedman test were performed to verify the superiority of the proposed method. Experimental results show that the proposed OOSSA outperformed the basic SSA, eight improved SSA variants, and nine cutting-edge swarm-based intelligence algorithms. Moreover, the algorithm was applied to an engineering design problem, and results show that the algorithm had better performance than other algorithms in engineering optimization. Finally, an OOSSA-based robot path planning approach was developed for solving the path planning problem in autonomous mobile robots. The proposed algorithm was simulated in three environment settings and compared with

收稿日期:2021-11-15
基金项目:国家自然科学基金(61461053,61461054,61562092);云南省教育厅科学研究基金(2022Y008)
作者简介:王宗山(1995—),男,博士研究生;
丁洪伟(1964—),男,教授,博士生导师
通信作者:丁洪伟,wzs_ynu@163.com

other algorithms including particle swarm optimization (PSO), artificial bee colony (ABC), grey wolf optimizer (GWO), firefly algorithm (FA), and SSA. Simulation results show that the proposed algorithm could plan the optimal collision-free paths compared with its competitors. The systematic experiments indicate that the OOSSA algorithm can be an effective tool for problem optimization.

Keywords: salp swarm algorithm; refracted opposition-based learning; orthogonal experimental design; self-adaptive learning; benchmark functions; engineering design optimization; path planning

樽海鞘群算法(salp swarm algorithm,SSA)[1]是2017年提出的一种新型群体智能优化算法,它源于对海洋生物樽海鞘导航和群体觅食行为的模拟,通过领导、跟随等过程,构建一种高效的寻优方案。SSA算法具有寻优机制简单、参数设置少、全局搜索能力强等特点。文献[1]的研究表明,SSA算法的寻优能力优于遗传算法(genetic algorithm,GA)[2]、粒子群优化(particle swarm optimization,PSO)[3]、人工蜂群(artificial bee colony,ABC)算法[4]等传统智能优化算法,与灰狼优化(grey wolf optimizer,GWO)[5]、鲸鱼优化算法(whale optimization algorithm,WOA)[6]、萤火虫算法(firefly algorithm,FA)[7]等新型群体智能算法相比,SSA也有较强的竞争力[1]。自提出以来,SSA算法已被成功应用于目标分类[8]、特征选择[9]、图像处理[10]、混合动力系统[11]等领域中。

然而,与其他元启发式算法相似,SSA算法也存在开发与勘探能力不平衡、寻优精度低、易陷入局部最优等缺点。针对这些问题,研究学者开发出了许多性能较好的SSA变体。文献[12]在领导者位置更新公式中引入一种控制搜索范围的非线性衰减因子,提高算法的局部搜索能力,同时在跟随者位置更新公式中引入动态学习策略,增强精英个体对领导者的协助作用,以此提高算法的寻优能力。文献[13]提出追随者共生概念,可以优化樽海鞘追随者的跟随方式,从而增强算法的全局开采能力,并通过引入非均匀高斯变异提高种群多样性。文献[14]通过引入疯狂算子对食物源进行扰动,以此提高种群多样性,同时提出自适应惯性权重,能够平衡算法的开采与勘探能力。文献[15]提出动态惯性权重,以此优化领导者的位置更新方式,从而提高算法的全局搜素能力,并随算法迭代自适应地调整樽海鞘领导者和追随者的数量,加快收敛速度的同时提高收敛精度。文献[16]在领导者位置更新公式中引入指数系数,并在跟随者位置更新公式中引入随机参数,加快算法收敛速度的同时改善了算法的求解精度。文献[17]引入一种突变机制帮助算法跳出局部最优,改善了算法求解精度不足的问题。文献[18]将SSA算法与PSO算法进行混合,提高算法勘探阶段的灵活性,获得了更高的寻优效率,并成功应用于特征提取问题。

上述SSA变体均在一定程度上增强了SSA算法的优化性能,但早熟收敛、全局开发与局部探索能力不平衡等问题仍然存在。此外,“没有免费午餐”(no-free-lunch,NFL)定理[19]指出,不存在一种适用于所有优化问题的优化算法。因此,提出一种性能较好的SSA变体是非常有价值的。基于上述分析,本文提出一种基于正交设计的折射反向学习樽海鞘群算法(orthogonal opposition-based salp swarm algorithm,OOSSA)。

1　研究背景

1.1　樽海鞘群算法

樽海鞘是一种外形类似水母的胶状海洋生物,通过吸入和排出海水完成移动。研究人员将樽海鞘群体的链式结构数学化,提出SSA。在基本SSA中,位于樽海鞘链最前端的个体为领导者,其余个体为跟随者。领导者根据食物源信息更新自身位置,数学模型为[1]

$$X_j^1=\begin{cases}F_j+c_1((u_j-l_j)c_2+l_j),c_3\geqslant 0.5\\F_j-c_1((u_j-l_j)c_2+l_j),c_3<0.5\end{cases}\tag{1}$$

式中:X_j^1为领导者在第j维上的位置,F_j为食物源在第j维上的位置,u_j和l_j分别为搜索空间第j维的上下限,c_2、c_3为区间[0,1]内的随机数,分别决定领导者的移动步长和移动方向。

c_1为距离控制因子,其值随算法迭代非线性减小到趋于零,计算公式如下:

$$c_1=2\mathrm{e}^{-\left(\frac{4t}{T}\right)^2}\tag{2}$$

式中:t为当前迭代次数,T为最大迭代次数。

跟随者位置更新的数学模型为[1]

$$X_j^i=\frac{1}{2}(X_j^i+X_j^{i-1})\tag{3}$$

式中:X_j^i为第i个跟随者在第j维上的位置,且$i\geqslant 2$。

SSA算法包括全局勘探和局部开采两个阶段。初始化时期,樽海鞘链群在搜索空间中勘探,并锁定最优解区域。随后,算法进入开采阶段,搜索代理在最优解区域内精确搜索,以提高求解精度。其中,距

离控制因子 c_1 对算法的寻优过程有重要影响。迭代前期,较大的 c_1 帮助种群勘探整个解空间; 迭代后期,较小的 c_1 帮助种群在特定的区域内精确开采。由于缺乏关于食物源位置(全局最优解)信息的先验知识,将每轮迭代中的全局最优个体更新为当前食物源的位置。

1.2 反向学习

反向学习(opposition-based learning,OBL)最初由 Tizhoosh[20] 提出,已成功应用于多种智能优化算法,其主要思想是同时评估当前可行解和其反向解,并选择较优解进入下一代的迭代计算。Rahnamayan 等[21] 证明: 相比于当前解,反向解有更大概率接近全局最优。因此,反向学习策略能够有效提高算法性能。根据樽海鞘群算法分析可知,基本 SSA 中,领导者负责带领群体向全局最优解所在的区域移动。因此,领导者的位置一定程度上决定了群体的移动方向。如果领导者陷入局部最优,则群体将跟随至局部极值区域,导致算法早熟收敛。将反向学习策略应用于领导者个体上避免其陷入局部最优,则该个体能够更好的带领群体移动至更有前景的搜索区域。

定义 1 反向数(opposite number)[22]。假设 $x \in [a,b]$ 为一个实数,则 x 的反向数 $\tilde{x}$ 定义为

$$\tilde{x} = a + b - x \tag{4}$$

将反向数的概念推广到多维空间,给出反向点的定义。

定义 2 反向点(opposite point)[22]。假设 $X = (x_1, x_2, \cdots, x_D)$ 为 D 维空间中的一点,$x_1, x_2, \cdots, x_D \in R$ 且 $x_j \in [a_j, b_j]\ (j = 1,2,\cdots,D)$,则反向点 $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_D)$ 定义为

$$\tilde{x}_j = a_j + b_j - x_j \tag{5}$$

基于透镜成像原理的折射反向学习策略(lens opposition-based learning,LOBL)是一种新型智能技术,其本质是在 OBL 的基础上,结合光学透镜成像原理所设计的一种动态反向学习策略,以帮助算法寻找更优的候选解,已成功与粒子群算法[23]、灰狼优化算法[24] 等结合,并大幅提高了算法性能。

定义 3 基点(cardinal point)[23]: 若 D 维空间中存在若干点 $o_1, o_2, \cdots, o_m$,任意点 $X = (x_1, x_2, \cdots, x_D)$ 和其反向点 $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_D)$ 到点 $o_i\ (i = 1, 2, \cdots, m)$ 的欧氏距离分别为 d_i 和 d_i^*,令 $k = d_i / d_i^*$,且 $k = 1,2,\cdots,n$,则称 o_i 为 X 与 $\tilde{X}$ 在 $k = i$ 时的基点。

以一维空间为例,假设坐标轴上 x 处有高度为 h 的物体 M,且 $x \in [a,b]$,基点位置 o(本文取 $[a,b]$ 中点作为基点)上放置焦距为 r 的透镜,基于透镜成像可得到高度为 h^* 的像 M^*,透镜成像原理见图 1。

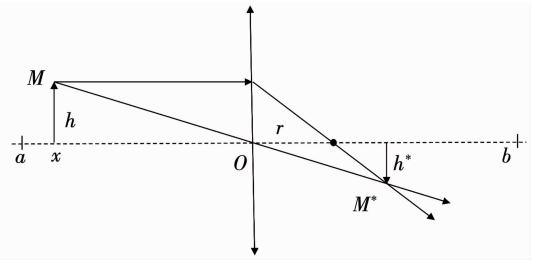


图 1 透镜成像原理示意

Fig. 1 Optical lens imaging principle

图 1 中,x 以 o 为基点得到对应的反向点 $\tilde{x}$,根据透镜成像原理可得出

$$\frac{(a+b)/2 - x}{\tilde{x} - (a+b)/2} = \frac{h}{h^*} \tag{6}$$

令 $h/h^* = k$,k 为缩放因子。变换式(6)即可得到反向点的计算公式为

$$\tilde{x} = (a+b)/2 + (a+b)/2k - x/k \tag{7}$$

当 $k = 1$ 时,式(7)即为中心位置在坐标点 $[-(a+b)/2, (a+b)/2]$ 的 OBL[22]。根据 OBL 策略得到的反向个体是固定的,而根据 k 值不同,通过 LOBL 得到的反向个体是动态的。

将式(7)推广到 D 维空间,一般化透镜折射反向学习策略得到

$$\tilde{x}_j = (a_j + b_j)/2 + (a_j + b_j)/2k - x_j/k \tag{8}$$

式中:x_j 和 $\tilde{x}_j$ 分别为 x 和 $\tilde{x}$ 的第 j 维分量,a_j 和 b_j 分别为解空间上下限的第 j 维分量。

1.3 正交试验设计

正交试验设计(orthogonal experimental design,OED)能够通过少量试验次数找到多因素、多水平实验的最优试验组合[25]。例如,对于 2 水平 7 因素的试验,若采用测试的方法找出最优组合,需 $2^7 = 128$ 次试验。若采用正交试验设计,根据正交表 $L_8(2^7)$,如式(9),仅通过 8 次试验即可找出最优或较优组合,大幅提高了试验效率。但根据正交试验特性,并不能保证正交表中的试验解包含该试验的最优解[25]。因此在使用正交表时,通常需要进行因素分析来找到试验的理论最优组合,并结合正交表中的所有组合一起确定试验的最优解。因此,对于 2 水平 7 因素的试验,需首先根据正交表 $L_8(2^7)$ 得到 8 组候选最优试验解,其次进行因素分析,找到一组理论最优组合,最后对 9 种组合进行评估,找出该实验的最优组合。

$$L_8(2^7)=\begin{bmatrix}1&1&1&1&1&1&1\\1&1&1&2&2&2&2\\1&2&2&1&1&2&2\\1&2&2&2&2&1&1\\2&1&2&1&2&1&2\\2&1&2&2&1&2&1\\2&2&1&1&2&2&1\\2&2&1&2&1&1&2\end{bmatrix} \tag{9}$$

2　正交反向学习樽海鞘群算法 OOSSA

2.1　正交反向学习策略

由式(1)可知，领导者仅在食物源的引导下更新自身位置，由于缺乏先验知识，难以确定当前食物源是否处于全局最优。如果食物源处于局部最优，则整个樽海鞘链群都将聚集在局部最优区域，种群多样性缺失，导致算法收敛至局部最优。为增强 SSA 算法跳出局部最优的能力，提出一种正交反向学习策略(orthogonal lens opposition-based learning，OLOBL)，并将其应用到领导者个体上产生新的候选个体。

OLOBL 是结合 OED 与 LOBL 技术所设计的一种策略。领导者执行 OLOBL 策略，跳跃到更有前景的搜索区域，从而提高种群多样性，降低算法陷入局部最优的概率。但文献[26]的研究表明：对于一个个体，其反向个体仅在部分维度上的值优于当前个体。因此，对个体的所有维度均取基于 OLOBL 的折射反向值会造成维度退化现象，即部分维度反而远离全局最优解。针对这一问题，结合 OED 和 LOBL 技术，设计一种正交折射反向学习策略，充分挖掘当前个体与反向个体的每一维分量，并对二者的优势维度进行组合，产生部分折射反向解。

将 OLOBL 策略嵌入 SSA 算法，优化问题的维度 D 对应正交试验设计中的因素，个体与折射反向个体即为正交试验设计中的两个水平。构建部分反向解的具体描述过程为：对当前个体和其折射反向个体设计一个 2 水平 D 因素的正交试验，产生 M 个部分折射反向解，M 按照式(10)进行计算。其中，根据正交试验表产生部分反向解时，若正交表的元素为 1，试验解对应维上取当前个体的值；若正交表的元素为 2，试验解对应维上取折射反向个体的值。

$$M=2^{\lceil \log_2(D+1)\rceil} \tag{10}$$

根据正交试验特性，正交表中第一行的元素全为 1，这就代表第一组试验解即为个体本身，不需要评估。另外 $M-1$ 组试验解是当前个体与其反向个体优势维度的组合，即部分折射反向解，需要评估。在使用正交试验设计时，需进行因素分析，找出一组不存在于正交表中的理论最优组合，需要评估。因此，执行一次 OLOBL 策略需要的函数评估次数为 M 次。在进化迭代过程中，仅对领导者执行 OLOBL 策略，并从领导者和其正交折射反向个体中选择较优个体进入下一代，这样可以有效增强算法的全局勘探能力，同时减少函数的评价次数，从而改善算法的整体性能。

2.2　惯性权重策略

由追随者位置更新模型可知，基本 SSA 中追随者在保留自身特征的同时学习前一个个体，完成位置更新。其位置更新机制较为单一，一旦领导者陷入局部最优，追随者必然跟随至该局部极值区域。为增强跟随者位置更新机制的灵活性，本文引入一种非线性递减惯性权重来评价前一个个体对当前追随者的影响程度。新的追随者位置更新公式为

$$X_j^i=\frac{1}{2}(X_j^i+\omega X_j^{i-1}) \tag{11}$$

$$\omega=\omega_{\max}-(\omega_{\max}-\omega_{\min})\frac{2}{2+\mathrm{e}^{10-0.04t}} \tag{12}$$

式中：ω 为本文设计的惯性权重，$\omega_{\max}$ 和 $\omega_{\min}$ 分别为初始惯性权重和最终惯性权重。

2.3　OOSSA 算法

所提 OOSSA 算法的实现流程如下：

Step1　设置算法参数：种群规模 N，最大函数评估次数 $F_{\max}$，搜索维度 D，搜索区域上下边界 u、l；在搜索空间中随机生成 N 个樽海鞘个体构成初始种群；

Step2　计算初始种群的适应值并排序，首个个体定义为领导者，其余个体为跟随者，并将适应值最优个体的位置作为当前食物源；

Step3　判断樽海鞘个体角色：若为领导者则进入 Step4；若为跟随者则进入 Step5；

Step4　根据式(1)更新领导者位置，产生候选解 1，同时执行 OLOBL 策略，产生候选解 2，根据适应值保留候选解 1 和候选解 2 中的较优个体；

Step5　根据式(11)更新跟随者位置；

Step6　比较食物源和当前最优个体的适应值，取较优者作为新的食物源；

Step7　若函数评估次数达到最大函数评估次数 $F_{\max}$，则输出食物源位置，即为最优解；否则，返回 Step3 继续迭代。

OOSSA 在保持 SSA 算法基本框架和整体流程不变的情况下，在领导者位置更新的同时执行 OLOBL 策略，同时修改了跟随者的位置更新公式。

设函数维度为 D,种群规模为 N,最大迭代次数为 T。樽海鞘群体移位操作的时间复杂度为 $O(TND)$,领导者执行 OLOBL 操作的时间复杂度为 $O(TD^2)$,因此,OOSSA 算法总的时间复杂度为 $O(TN^2)$,与原 SSA 算法时间复杂度一致,这表明在改进算法的同时并没有增加计算开销。

3 仿真实验及结果分析

3.1 测试函数

为了验证 OOSSA 算法的有效性,实验选取 12 个CEC2017 测试函数进行,见表 1。其中,函数维度设置为 100 维。表 1 中,f_1 ~f_6为单峰函数,此类函数没有局部极值点,只有一个全局最优解,用于测试算法的局部开采能力;f_7 ~f_{12}为复杂多峰函数,此类函数有一个全局最优解的同时包含多个局部极值点,用于测试算法平衡全局勘探和局部开采的能力。

实验环境为: Windows 10 操作系统,Intel(R) Core(TM) i7-7700 处理器,主频 3.60 GHz,8 GB 内存,算法基于 Matlab R2014b 用 M 语言编写。

表 1 CEC2017 测试函数

Tab. 1 CEC2017 benchmark functions

序号	函数名	最优值
f_1	Sphere	0
f_2	Schwefel 2.22	0
f_3	Schwefel 1.2	0
f_4	Schwefel 2.21	0
f_5	Quartic	0
f_6	Bent Cigar	0
f_7	Generalized Penalized Function	0
f_8	Rastrigin	0
f_9	Ackley	0
f_{10}	Griewank	0
f_{11}	Zakharov	0
f_{12}	Schaffer's F7	0

3.2 与基本 SSA 和改进 SSA 算法的比较

利用 OOSSA 算法求解表 1 中的 12 个经典测试函数,并与基本 SSA 算法(SSA,2017)[1]、多子群的共生非均匀高斯变异 SSA 算法(MSNSSA,2020)[13]、生命周期启发的 SSA 算法(LSSA,2020)[27]、自适应 SSA 算法(ASSA,2021)[28]、高斯 SSA 算法(GSSA,2021)[29]、改进反向学习的 SSA 算法(OBSSA,2021)[30]、具有惯性权重的自适应 SSA 算法(ASSO,2021)[31]、随机取代的双惯性权重 SSA 算法(RDSSA,2021)[32]、混合鲸鱼优化的 SSA 算法(IWOSSA,2021)[33]进行比较。为保证比较的公平性,所有算法均使用相同的种群规模 $N=30$,最大函数评价次数 15 000。对比算法的其他参数设置与各自原文献保持一致。在 OOSSA 算法中,缩放因子 $k=10\ 000$。每种算法对每个测试函数独立运行 30 次实验,分别记录它们的平均值和标准差,并采用 Friedman 检验[34]对算法进行排名,见表 2。

由表 2 可知,除了函数 f_5、f_7、f_9、f_{11},OOSSA 算法在其他 8 个测试函数上均一致收敛到全局最优。与 SSA、LSSA、ASSA、GSSA 和 IWOSSA 算法相比,OOSSA 在所有基准函数上取得了较好的结果,且收敛精度与求解稳定性远优于对比算法。与 MSNSSA 算法相比,OOSSA 分别在 9 个和 2 个函数上取得了较好和相似的结果。OOSSA 分别在 7 个和 4 个测试函数上获得了比 OBSSA 算法较好和相似的结果;然而,对于 f_4,OBSSA 得到了较好的结果。与 ASSO 算法相比,OOSSA 分别在 9 个和 3 个测试函数上获得了较好和相似的结果。与 RDSSA 算法相比,OOSSA 分别在 9 个和 2 个测试函数上取得了较好和相似的结果;然而,RDSSA 在 f_7 上获得了稍优的结果。根据表 2 中 Friedman 排名检验结果,OOSSA 算法排名第一,RDSSA 排名第二,其次为 OBSSA、ASSO、MSNSSA、GSSA、ASSA、LSSA、IWOSSA 和 SSA。证明不同 SSA 变体的改进策略均有一定效果,求解性能优于基本 SSA 算法,但不如 OOSSA 算法。

此外,本文采用 Wilcoxon 秩和检验[34](显著性水平设为 0.05)从统计意义上对 10 种算法的寻优结果进行评价。在表 3 中给出所有基准函数的 OOSSA 和其他对比算法的 Wilcoxon 秩和检验中计算的 p 值。例如,如果最优算法为 OOSSA,则在 OOSSA versus SSA,OOSSA versus LSSA 等之间进行成对比较。其中,N/A 表示不适用,意味着相应算法在该函数上表现最优,没有统计数据与自身进行比较。符号"-""+""="分别表示 OOSSA 劣于、优于、相当于各算法的结果。当 $p<0.05$ 时,拒绝零假设,即认为两种对比算法有显著差异[34]。

由表 3 可以看出,除了 f_5 的 OOSSA versus OBSSA 和 OOSSA versus ASSO 这两种情形下 $p>0.05$,其他对比算法的 $p<0.05$,根据 Wilcoxon 秩和检验,这说明 OOSSA 算法的整体性能明显优于基本 SSA 算法和其他 8 种改进 SSA 算法,且 OOSSA 算法的优越性在统计上是显著的。总体来说,相比于基本 SSA 算法和其他最新的 SSA 变体,OOSSA 算法能够提供竞争性的性能。

表 2　OOSSA 与多种 SSA 变体对 12 个 100 维测试函数的结果比较

Tab. 2　Comparison of OOSSA and SSA variants on twelve 100-dimensional test functions

函数	统计结果	SSA	LSSA	MSNSSA	ASSA	GSSA	OBSSA	ASSO	RDSSA	IWOSSA	OOSSA
f_1	平均值	1.49×10^{3}	2.60×10^{-3}	1.82×10^{-22}	5.06×10^{-2}	1.08×10^{-14}	3.96×10^{-32}	1.97×10^{-26}	2.10×10^{-52}	1.28×10^{-6}	0.00
	标准差	4.07×10^{2}	2.70×10^{-3}	4.57×10^{-23}	2.37×10^{-2}	5.55×10^{-14}	3.81×10^{-32}	2.29×10^{-27}	1.15×10^{-51}	1.11×10^{-6}	0.00
	排名	10.00	8.00	5.00	9.00	6.00	3.00	4.00	2.00	7.00	1.00
f_2	平均值	4.66×10^{1}	8.46×10^{-2}	1.06×10^{-11}	2.22×10^{-2}	3.55×10^{-17}	1.85×10^{-16}	1.11×10^{-13}	5.45×10^{-22}	4.90×10^{-3}	0.00
	标准差	7.65×10^{1}	3.87×10^{-2}	1.76×10^{-12}	5.90×10^{-3}	5.40×10^{-17}	1.07×10^{-16}	7.72×10^{-15}	2.99×10^{-21}	2.46×10^{-2}	0.00
	排名	10.00	9.00	6.00	8.00	3.00	4.00	5.00	2.00	7.00	1.00
f_3	平均值	4.70×10^{4}	7.18×10^{3}	6.27×10^{-21}	1.67×10^{4}	1.68×10^{4}	4.81×10^{-30}	1.93×10^{-25}	7.50×10^{-42}	1.03×10^{5}	0.00
	标准差	2.36×10^{4}	5.78×10^{3}	4.30×10^{-21}	1.05×10^{3}	1.06×10^{3}	5.60×10^{-30}	1.15×10^{-25}	3.29×10^{-41}	2.96×10^{4}	0.00
	排名	9.00	6.00	5.00	7.00	8.00	3.00	4.00	2.00	10.00	1.00
f_4	平均值	2.80×10^{1}	2.30×10^{1}	4.00×10^{-12}	1.05×10^{1}	2.33×10^{1}	5.54×10^{-17}	3.61×10^{-14}	3.77×10^{-20}	4.59×10^{1}	0.00
	标准差	2.84	4.54	1.01×10^{-12}	2.69	3.08	3.38×10^{-17}	3.37×10^{-15}	2.07×10^{-19}	6.33	0.00
	排名	9.00	7.00	5.00	6.00	8.00	3.00	4.00	2.00	10.00	1.00
f_5	平均值	2.70	6.60×10^{-1}	2.55×10^{-4}	8.99×10^{-2}	2.00×10^{-1}	1.08×10^{-4}	1.07×10^{-4}	7.34×10^{-4}	8.51×10^{-2}	6.81×10^{-5}
	标准差	6.00×10^{-1}	1.95×10^{-1}	2.14×10^{-4}	2.51×10^{-2}	3.40×10^{-1}	1.20×10^{-4}	1.22×10^{-4}	4.78×10^{-4}	3.37×10^{-2}	6.67×10^{-5}
	排名	10.00	9.00	4.00	7.00	8.00	3.00	2.00	5.00	6.00	1.00
f_6	平均值	1.32×10^{7}	6.16×10^{3}	1.56×10^{-18}	4.32×10^{2}	3.62×10^{-11}	6.36×10^{-28}	2.08×10^{-22}	2.65×10^{-52}	1.32×10^{-2}	0.00
	标准差	3.91×10^{6}	4.80×10^{3}	3.73×10^{-19}	2.07×10^{2}	1.03×10^{-10}	6.15×10^{-28}	2.56×10^{-23}	1.45×10^{-51}	1.04×10^{-2}	0.00
	排名	10.00	9.00	5.00	8.00	6.00	3.00	4.00	2.00	7.00	1.00
f_7	平均值	3.22×10^{1}	8.58	1.78×10^{-1}	1.49	2.31×10^{1}	1.63×10^{-1}	9.65×10^{-1}	2.60×10^{-3}	2.03×10^{1}	1.94×10^{-1}
	标准差	9.00	4.39	4.18×10^{-2}	1.10	8.50	2.99×10^{-2}	5.14×10^{-2}	6.10×10^{-3}	1.19×10^{1}	4.57×10^{-2}
	排名	10.00	7.00	3.00	6.00	9.00	2.00	5.00	1.00	8.00	4.00
f_8	平均值	2.41×10^{2}	1.84×10^{2}	0.00	2.10×10^{2}	6.50×10^{-10}	0.00	0.00	0.00	3.22×10^{2}	0.00
	标准差	5.96×10^{1}	1.06×10^{2}	0.00	4.87×10^{1}	3.55×10^{-9}	0.00	0.00	0.00	1.12×10^{2}	0.00
	排名	9.00	7.00	1.00	8.00	6.00	1.00	1.00	1.00	10.00	1.00
f_9	平均值	9.93	2.58×10^{-2}	1.68×10^{-12}	3.08×10^{-2}	2.68×10^{-9}	8.88×10^{-16}	1.88×10^{-14}	4.20×10^{-15}	6.67×10^{-1}	8.88×10^{-16}
	标准差	1.17	1.75×10^{-2}	3.06×10^{-13}	9.60×10^{-3}	6.09×10^{-9}	0.00	2.37×10^{-15}	9.01×10^{-16}	3.65	0.00
	排名	10.00	7.00	5.00	8.00	6.00	1.00	4.00	3.00	9.00	1.00
f_{10}	平均值	1.29×10^{1}	3.57×10^{-2}	0.00	4.06×10^{-2}	1.29×10^{-14}	0.00	0.00	0.00	8.00×10^{-3}	0.00
	标准差	3.31	5.68×10^{-2}	0.00	4.07×10^{-2}	3.65×10^{-14}	0.00	0.00	0.00	1.62×10^{-2}	0.00
	排名	10.00	8.00	1.00	9.00	6.00	1.00	1.00	1.00	7.00	1.00
f_{11}	平均值	7.68×10^{-1}	2.17×10^{-2}	1.36×10^{-24}	8.60×10^{-3}	9.84×10^{-13}	1.98×10^{2}	1.69×10^{-28}	1.10×10^{-67}	1.24×10^{-2}	3.59×10^{-128}
	标准差	2.05×10^{-1}	1.31×10^{-2}	3.99×10^{-24}	3.70×10^{-3}	4.10×10^{-12}	3.35×10^{1}	4.80×10^{-29}	5.52×10^{-67}	1.91×10^{-2}	1.88×10^{-127}
	排名	9.00	8.00	4.00	6.00	5.00	10.00	3.00	2.00	7.00	1.00
f_{12}	平均值	5.07	3.22×10^{-1}	1.27×10^{-6}	1.63	3.38×10^{-10}	4.83×10^{-9}	1.29×10^{-7}	2.40×10^{-13}	3.32×10^{-2}	0.00
	标准差	2.30×10^{-1}	2.12×10^{-1}	9.52×10^{-8}	3.64×10^{-1}	2.84×10^{-10}	1.18×10^{-9}	4.42×10^{-9}	9.75×10^{-13}	3.25×10^{-2}	0.00
	排名	10.00	8.00	6.00	9.00	3.00	4.00	5.00	2.00	7.00	1.00
平均排名		9.67	7.75	4.17	7.58	6.17	3.17	3.50	2.08	7.92	1.25
Friedman		10.00	8.00	5.00	7.00	6.00	3.00	4.00	2.00	9.00	1.00

表 3 100 维基准函数 Wilcoxon 秩和检验的 p 值

Tab. 3 p-value for Wilcoxon's rank-sum test on 100-dimensional benchmark functions

函数	SSA	LSSA	MSNSSA	ASSA	GSSA	OBSSA	ASSO	RDSSA	IWOSSA
f_1	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
f_2	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
f_3	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
f_4	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
f_5	3.02×10^{-11}	3.02×10^{-11}	1.04×10^{-4}	3.02×10^{-11}	3.02×10^{-11}	1.30×10^{-1}	4.46×10^{-1}	1.78×10^{-10}	3.02×10^{-11}
f_6	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
f_7	3.02×10^{-11}	3.02×10^{-11}	1.09×10^{-1}	4.62×10^{-10}	3.02×10^{-11}	7.60×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
f_8	1.21×10^{-12}	1.21×10^{-12}	N/A	1.21×10^{-12}	1.10×10^{-2}	N/A	N/A	N/A	1.21×10^{-12}
f_9	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	N/A	5.65×10^{-13}	7.15×10^{-13}	1.21×10^{-12}
f_{10}	1.21×10^{-12}	1.21×10^{-12}	N/A	1.21×10^{-12}	2.21×10^{-6}	N/A	N/A	N/A	1.21×10^{-12}
f_{11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
f_{12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
+/=/−	12/0/0	12/0/0	9/3/0	12/0/0	12/0/0	7/4/1	9/3/0	9/2/1	12/0/0

3.3 两种改进策略的有效性分析

本文对基本 SSA 算法做了两个方面的改进，分别为 OLOBL 策略和惯性权重策略(inertia weight, IW)，为了验证两种改进策略的有效性，分别将它们嵌入到基本 SSA 算法中，得到 OLOBL-SSA 和 IW-SSA 算法。此外，将结合 OBL 的正交反向学习(orthogonal opposition-based learning, OOBL)嵌入 SSA 中，得到 OOBL-SSA，并与 OLOBL-SSA 进行对比，以验证 LOBL 和 OBL 策略的有效性。本节选取表 1 中的 12 个测试函数进行仿真实验来比较 OOSSA、OLOBL-SSA、IW-SSA、OOBL-SSA 和基本 SSA 算法。设置所有算法的种群规模均为 30，最大函数评价次数均为 15 000，函数维度设置为 100 维，每种算法在每个函数上独立运行 30 次，分别记录它们的平均值、标准差和 Friedman 排名检验结果，见表 4。

表 4 采用不同改进策略的 SSA 算法在 CEC2017 上的实验结果

Tab. 4 Comparison of SSA with different improvement strategies for CEC2017

函数	统计结果	SSA	OLOBL-SSA	IW-SSA	OOBL-SSA	OOSSA
f_1	平均值	1.49×10^{3}	0.00	2.41×10^{-42}	1.84×10^{-22}	0.00
	标准差	4.07×10^{2}	0.00	3.61×10^{-43}	2.29×10^{-22}	0.00
	排名	5.00	1.00	3.00	4.00	1.00
f_2	平均值	4.66×10^{1}	0.00	1.25×10^{-21}	8.70×10^{-12}	0.00
	标准差	7.65	0.00	1.23×10^{-22}	5.85×10^{-12}	0.00
	排名	5.00	1.00	3.00	4.00	1.00
f_3	平均值	4.70×10^{4}	0.00	1.53×10^{-40}	1.61×10^{-22}	0.00
	标准差	2.36×10^{4}	0.00	5.66×10^{-41}	1.39×10^{-22}	0.00
	排名	5.00	1.00	3.00	4.00	1.00
f_4	平均值	2.80×10^{1}	0.00	4.44×10^{-22}	2.44×10^{-12}	0.00
	标准差	2.84	0.00	6.29×10^{-23}	2.61×10^{-12}	0.00
	排名	5.00	1.00	3.00	4.00	1.00
f_5	平均值	2.70	7.40×10^{-5}	7.97×10^{-5}	2.12×10^{-4}	6.03×10^{-5}
	标准差	6.00×10^{-1}	6.67×10^{-5}	6.15×10^{-5}	1.86×10^{-4}	6.08×10^{-5}
	排名	5.00	2.00	3.00	4.00	1.00
f_6	平均值	1.32×10^{7}	0.00	2.34×10^{-38}	1.15×10^{-18}	0.00
	标准差	3.91×10^{6}	0.00	3.56×10^{-39}	1.32×10^{-18}	0.00
	排名	5.00	1.00	3.00	4.00	1.00
f_7	平均值	3.22×10^{1}	1.47×10^{-1}	1.87×10^{-1}	1.71×10^{-1}	2.66×10^{-2}
	标准差	9.00	3.28×10^{-2}	3.38×10^{-2}	4.12×10^{-2}	5.00×10^{-2}
	排名	5.00	2.00	4.00	3.00	1.00
f_8	平均值	2.41×10^{2}	0.00	0.00	0.00	0.00
	标准差	5.96×10^{1}	0.00	0.00	0.00	0.00
	排名	5.00	1.00	1.00	1.00	1.00

续表 4

函数	统计结果	SSA	OLOBL-SSA	IW-SSA	OOBL-SSA	OOSSA
f_9	平均值	9.93	8.88×10^{-16}	4.44×10^{-15}	1.44×10^{-12}	8.88×10^{-16}
	标准差	1.17	0.00	0.00	8.44×10^{-13}	0.00
	排名	5.00	1.00	3.00	4.00	1.00
f_{10}	平均值	1.29×10^{1}	0.00	0.00	0.00	0.00
	标准差	3.31	0.00	0.00	0.00	0.00
	排名	5.00	1.00	1.00	1.00	1.00
f_{11}	平均值	7.68×10^{1}	1.21×10^{3}	2.89×10^{-44}	1.81×10^{2}	1.49×10^{-75}
	标准差	2.05×10^{1}	2.15×10^{2}	7.21×10^{-45}	3.68×10^{1}	8.08×10^{-75}
	排名	5.00	5.00	2.00	4.00	1.00
f_{12}	平均值	5.07	0.00	1.36×10^{-11}	1.12×10^{-6}	0.00
	标准差	2.30×10^{-1}	0.00	8.57×10^{-13}	4.64×10^{-7}	0.00
	排名	5.00	1.00	3.00	4.00	1.00
	平均排名	4.83	1.50	2.67	3.42	1.00
	Friedman	5.00	2.00	3.00	4.00	1.00

从表 4 可知，OLOBL-SSA、IW-SSA 和 OOBL-SSA 算法在所有函数上的求解精度和求解稳定性均优于原 SSA 算法，这证明了改进策略的有效性。与 IW-SSA 和 OOBL-SSA 算法相比，OOSSA 在除 f_8 和 f_{10} 之外的 10 个函数上均取得了较优的结果；对于 f_8 和 f_{10}，3 种算法均达到了理论最优值。OLOBL-SSA 算法则和 OOSSA 算法一样，在多个测试函数上均能达到理论最优值，但对于函数 f_5、f_7 和 f_{11}，OOSSA 的寻优精度和寻优稳定性明显优于 OLOBL-SSA，尤其在函数 f_{11} 上，OOSSA 算法获得了显著的优势。基于上述分析，两种改进策略均具备一定效果，且两种改进策略同时作用能够进一步提高算法的寻优性能。此外，从表 4 可以看出，相比于 OOBL-SSA 算法，OLOBL-SSA 在除 f_8 和 f_{10} 之外的 10 个基准函数上均获得了显著的优势；对于 f_8 和 f_{10}，两种算法均达到了理论最优值。这表明正交反向学习中采用 LOBL 策略能够更加有效地帮助算法跳出局部最优。此外，由表 4 中 Friedman 排名检验结果可知，OOSSA 排名第一，OLOBL-SSA 排名第二，依次是 IW-SSA、OOBL-SSA 和 SSA，这进一步验证了改进策略的有效性以及 OOSSA 算法比单策略改进算法具备明显优势。

3.4　OOSSA 算法求解大规模优化问题的可行性分析

为验证 OOSSA 算法应用于求解大规模优化问题的可行性，本节选取表 1 中 12 个测试函数进行仿真实验，并将函数维度设置为 10 000 维。设置所有算法的种群规模均为 30，最大函数评价次数均为 15 000。表 5 给出了 OOSSA 算法对12 个大规模测试函数 30 次独立实验获得的最优解、最差解、平均值和标准差结果。此外，在实验结果中加入求解成功率指标，用于评价 OOSSA 算法处理大规模优化问题的效率。判断一次求解是否成功的标准为

$$\begin{cases} |f_A - f_T|/f_T < 10^{-5}, & f_T \neq 0 \\ |f_A - f_T| < 10^{-5}, & f_T = 0 \end{cases} \tag{13}$$

式中：f_A 为算法在测试函数上取得的结果，f_T 为测试函数的理论最优值。

表 5　OOSSA 算法求解 10 000 维测试函数的实验结果

Tab. 5　Experimental results of OOSSA for solving 10 000-dimensional test functions

函数	OOSSA				
	最优解	最差解	平均值	标准差	成功率/%
f_1	0.00	0.00	0.00	0.00	100
f_2	0.00	0.00	0.00	0.00	100
f_3	0.00	0.00	0.00	0.00	100
f_4	0.00	0.00	0.00	0.00	100
f_5	4.96×10^{-6}	2.49×10^{-4}	1.11×10^{-4}	9.41×10^{-5}	20
f_6	0.00	0.00	0.00	0.00	100
f_7	2.42×10^{-1}	3.59×10^{-1}	3.01×10^{-1}	4.83×10^{-2}	0
f_8	0.00	0.00	0.00	0.00	100
f_9	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	0.00	100
f_{10}	0.00	0.00	0.00	0.00	100
f_{11}	2.65×10^{-79}	1.33×10^{-76}	2.93×10^{-77}	5.81×10^{-77}	100
f_{12}	0.00	0.00	0.00	0.00	100

由表 5 可以看出，除了 f_5、f_7、f_9、f_{11}，OOSSA 算法在其他 8 个测试函数上均能收敛到理论最优值，且根据标准差能够看出，OOSSA 在保持高求解精度的同时具备同样优越的求解稳定性；对于函数 f_5、f_7、f_9 和 f_{11}，OOSSA 虽未能找到理论最优解，但仍获得了令人满意的结果。这充分说明 OOSSA 算法在处理大规模优化问题时具备较强的鲁棒性。在求解成功率方面，OOSSA 在除 f_5 和 f_7 之外的 10 个测试函数上均能达到 100% 的成功率，这也充分说明 OOSSA 算

法能够有效应用于求解大规模优化问题。

3.5 与其他群体智能算法的比较

为了进一步验证 OOSSA 算法的有效性,将其与 9 种最新的群体智能优化算法进行对比,它们分别是海洋捕食者算法(MPA,2020)[35]、平衡优化器(EO,2020)[36]、基于选择对立的灰狼优化器(SOGWO,2020)[37]、改进的灰狼优化器(IGWO,2021)[38]、算数优化算法(ArOA,2021)[39]、阿基米德优化算法(AOA,2020)[40]、基于多样性和突变机制的飞蛾扑火优化(DMMFO,2021)[41]、双重自适应权重的飞蛾扑火优化(WEMFO,2021)[42]、基于反向学习的灰狼优化器(OGWO,2021)[43]。本节选取表 1 中的 12 个测试函数进行仿真实验,并将函数维度设置为 $D=100$。设置所有算法的种群规模均为 30,最大函数评价次数均为 15 000。对比算法的其他参数设置与原文献保持一致。表 6 给出了 10 种算法对 12 个测试函数 30 次独立实验获得的平均值和标准差,并采用 Friedman 检验对算法进行排名。

表 6 OOSSA 与多种前沿算法对 12 个测试函数的结果比较

Tab. 6 Comparison of OOSSA and nine cutting-edge algorithms for 12 test functions

函数	统计结果	SOGWO	MPA	EO	ArOA	AOA	IGWO	WEMFO	DMMFO	OGWO	OOSSA
f_1	平均值	3.15×10^{-12}	3.02×10^{-19}	5.19×10^{-29}	2.16×10^{-2}	1.34×10^{-77}	2.40×10^{-12}	1.27×10^{-22}	3.22×10^{4}	4.16×10^{-15}	0.00
	标准差	3.80×10^{-12}	4.99×10^{-19}	1.19×10^{-28}	7.90×10^{-3}	4.67×10^{-77}	1.98×10^{-12}	5.61×10^{-22}	6.47×10^{3}	4.72×10^{-15}	0.00
	排名	8.00	5.00	3.00	9.00	2.00	7.00	4.00	10.00	6.00	1.00
f_2	平均值	4.55×10^{-8}	2.31×10^{-11}	2.21×10^{-17}	2.45×10^{-65}	1.58×10^{-40}	1.54×10^{-8}	4.57×10^{-14}	2.23×10^{2}	3.86×10^{-10}	0.00
	标准差	1.69×10^{-8}	2.60×10^{-11}	2.49×10^{-17}	9.23×10^{-65}	7.83×10^{-40}	5.48×10^{-9}	8.47×10^{-14}	5.02×10^{1}	3.07×10^{-10}	0.00
	排名	9.00	6.00	4.00	2.00	3.00	8.00	5.00	10.00	7.00	1.00
f_3	平均值	1.94×10^{3}	1.33×10^{1}	1.36×10^{1}	3.81×10^{3}	6.38×10^{-64}	5.21×10^{3}	5.08×10^{-8}	2.29×10^{5}	9.37×10^{2}	0.00
	标准差	1.63×10^{3}	4.51×10^{1}	5.50×10^{1}	2.09×10^{4}	3.49×10^{-63}	2.16×10^{3}	2.77×10^{-7}	3.23×10^{4}	1.32×10^{3}	0.00
	排名	7.00	4.00	5.00	8.00	2.00	9.00	3.00	10.00	6.00	1.00
f_4	平均值	1.08	1.96×10^{-7}	2.40×10^{-3}	9.31×10^{-1}	1.02×10^{-38}	3.97	3.97×10^{-10}	8.89×10^{1}	1.91	0.00
	标准差	9.33×10^{-1}	6.75×10^{-8}	5.30×10^{-3}	1.29×10^{-2}	5.24×10^{-38}	2.35	8.01×10^{-10}	2.50	1.85	0.00
	排名	7.00	4.00	5.00	6.00	2.00	9.00	3.00	10.00	8.00	1.00
f_5	平均值	7.30×10^{-3}	2.00×10^{-3}	2.70×10^{-3}	6.98×10^{-5}	8.70×10^{-4}	1.26×10^{-2}	1.40×10^{-3}	9.84×10^{1}	2.20×10^{-3}	6.36×10^{-5}
	标准差	2.80×10^{-3}	8.69×10^{-4}	1.40×10^{-3}	4.71×10^{-5}	7.44×10^{-4}	4.00×10^{-3}	7.65×10^{-4}	3.03×10^{1}	1.90×10^{-3}	3.03×10^{-5}
	排名	8.00	5.00	7.00	2.00	3.00	9.00	4.00	10.00	6.00	1.00
f_6	平均值	1.58×10^{-8}	2.18×10^{-15}	4.55×10^{-25}	1.23×10^{-104}	1.72×10^{-71}	1.77×10^{-8}	9.07×10^{-19}	3.23×10^{8}	3.22×10^{-11}	0.00
	标准差	1.26×10^{-8}	2.82×10^{-15}	7.05×10^{-25}	6.71×10^{-104}	9.37×10^{-71}	1.03×10^{-8}	4.18×10^{-18}	7.45×10^{7}	3.71×10^{-11}	0.00
	排名	8.00	6.00	4.00	2.00	3.00	9.00	5.00	10.00	7.00	1.00
f_7	平均值	2.97×10^{-1}	4.58×10^{-2}	3.99×10^{-2}	9.01×10^{-1}	1.11	2.45×10^{-1}	4.07×10^{-1}	9.55×10^{7}	2.59×10^{-1}	1.78×10^{-1}
	标准差	8.41×10^{-2}	1.05×10^{-2}	1.08×10^{-2}	2.48×10^{-2}	5.69×10^{-2}	7.94×10^{-2}	4.34×10^{-2}	4.08×10^{7}	4.89×10^{-2}	3.86×10^{-2}
	排名	6.00	2.00	1.00	8.00	9.00	4.00	7.00	10.00	5.00	3.00
f_8	平均值	1.02×10^{1}	0.00	0.00	0.00	0.00	1.41×10^{2}	5.07×10^{2}	8.43×10^{2}	8.73×10^{-1}	0.00
	标准差	7.45	0.00	0.00	0.00	0.00	5.68×10^{1}	3.71×10^{2}	7.17×10^{1}	2.01	0.00
	排名	7.00	1.00	1.00	1.00	1.00	8.00	9.00	10.00	6.00	1.00
f_9	平均值	1.43×10^{-7}	4.80×10^{-11}	3.53×10^{-14}	2.24×10^{-4}	1×10^{1}	1.78×10^{-7}	1.36	1.97×10^{1}	5.28×10^{-9}	8.88×10^{-16}
	标准差	4.93×10^{-8}	2.56×10^{-11}	5.73×10^{-15}	1.20×10^{-3}	2.34×10^{-5}	7.04×10^{-8}	5.06	1.19×10^{-1}	2.65×10^{-9}	0.00
	排名	5.00	3.00	2.00	7.00	9.00	6.00	8.00	10.00	7.00	1.00
f_{10}	平均值	1.11×10^{-2}	0.00	0.00	6.26×10^{2}	0.00	0.00	1.30×10^{-3}	3.27×10^{2}	3.30×10^{-3}	0.00
	标准差	1.62×10^{-2}	0.00	0.00	1.81×10^{2}	0.00	0.00	4.20×10^{-3}	7.98×10^{1}	1.01×10^{-2}	0.00
	排名	8.00	1.00	1.00	10.00	1.00	1.00	6.00	9.00	7.00	1.00
f_{11}	平均值	2.98×10^{-13}	2.62×10^{-19}	1.68×10^{-28}	1.36×10^{3}	8.77×10^{-73}	1.70×10^{-12}	1.96×10^{-24}	5.22×10^{2}	1.06×10^{-15}	5.00×10^{-130}
	标准差	2.04×10^{-13}	2.67×10^{-19}	2.14×10^{-28}	9.59×10^{1}	3.71×10^{-72}	1.74×10^{-12}	7.38×10^{-24}	7.66×10^{1}	1.35×10^{-15}	2.70×10^{-129}
	排名	7.00	5.00	3.00	10.00	2.00	8.00	4.00	9.00	6.00	1.00
f_{12}	平均值	1.05×10^{-2}	4.22×10^{-7}	1.35×10^{-9}	1.50×10^{-3}	1.72×10^{-22}	1.07×10^{-2}	1.16×10^{-8}	8.44	9.30×10^{-5}	0.00
	标准差	2.40×10^{-3}	3.25×10^{-7}	6.43×10^{-10}	3.30×10^{-3}	5.99×10^{-22}	2.50×10^{-3}	1.09×10^{-8}	3.61×10^{-1}	5.08×10^{-5}	0.00
	排名	8.00	5.00	3.00	7.00	2.00	9.00	4.00	10.00	6.00	1.00
	平均排名	7.33	3.92	3.25	6.00	3.25	7.25	5.17	9.83	6.42	1.25
	Friedman	9.00	4.00	3.00	6.00	3.00	8.00	5.00	10.00	7.00	1.00

比较表 6 中结果可知，与 SOGWO、WEMFO、DMMFO 和 OGWO 算法相比，OOSSA 在所有测试函数上均获得了较好的结果。与 MPA 和 EO 算法相比，OOSSA 在 9 个函数上获得了较好的性能和 2 个函数上得到了相似的性能；然而，对于 f_7，OOSSA 获得稍差的结果。与 ArOA 和 IGWO 算法相比，OOSSA 分别在 11 个和 1 个测试函数上取得了较好和相似的性能。OOSSA 在 10 个测试函数上获得了比 AOA 算法较好的结果；对于函数 f_8 和 f_{10}，两种算法均收敛至全局最优。此外，从 Friedman 排名检验结果可以看出，OOSSA 排名第一，AOA 和 EO 排名并列第二，其余依次是 MPA、WEMFO、ArOA、OGWO、IGWO、SOGWO 和 DMMFO。这进一步验证了 OOSSA 算法的优越性。

3.6　收敛性分析

为进一步验证 OOSSA 算法的收敛性能，图 2 给出了 OOSSA、SSA、和各 SSA 变体对表 1 中部分代表性基准函数的收敛曲线。图 3 给出了 OOSSA 和各前沿算法对表 1 中部分代表性测试函数的收敛曲线。设置所有算法的种群规模均为 30，最大函数评价次数均为 15 000，函数维度设置为 $D=100$。

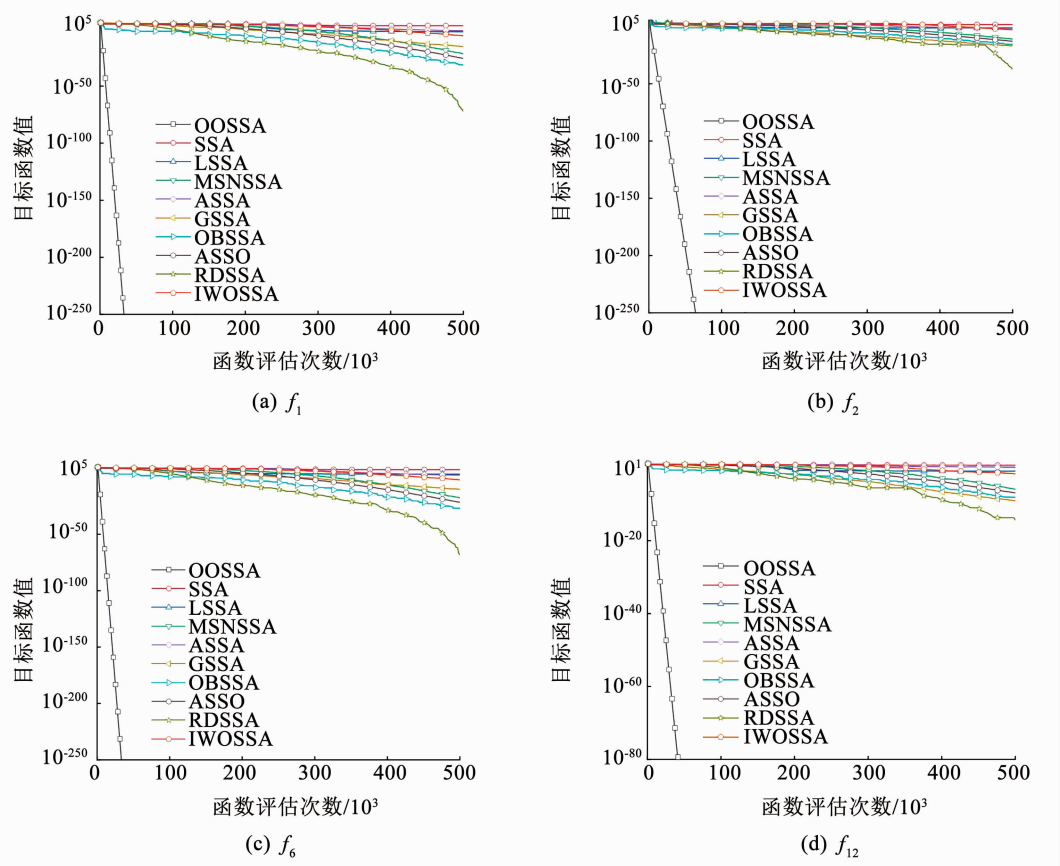
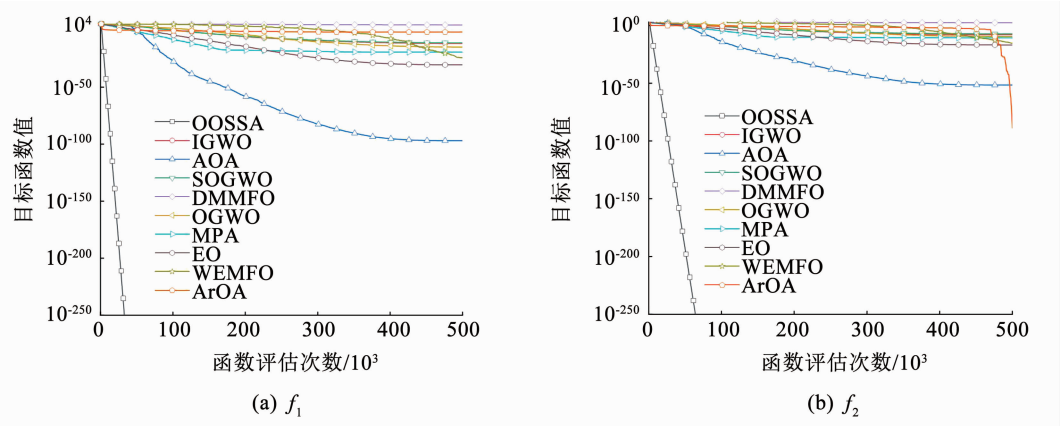


图 2　多种 SSA 变体对 4 个代表性测试函数的收敛曲线

Fig. 2　Convergence curves of different SSA variants for four representative test functions



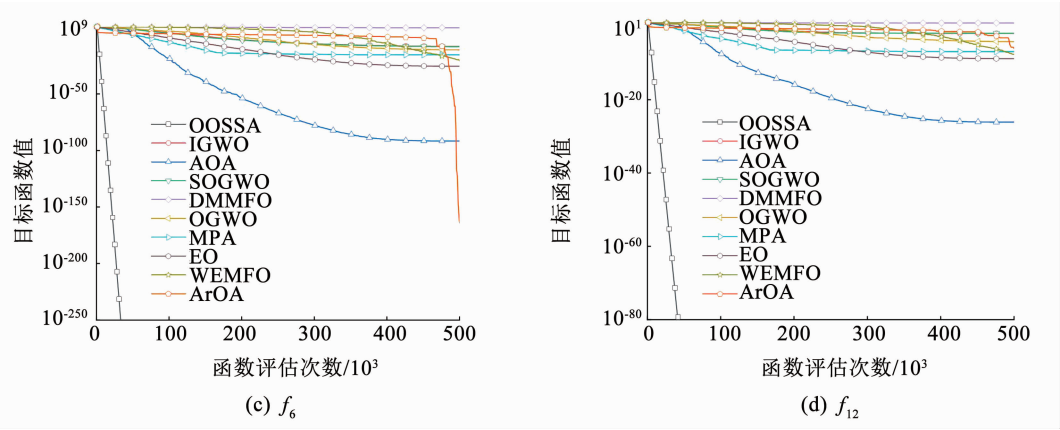


图 3 多种前沿算法对 4 个代表性测试函数的收敛曲线

Fig. 3 Convergence curves of different cutting-edge algorithms for four representative test functions

从图 2 和图 3 可以看出,与基本 SSA 和 8 种 SSA 变体相比,OOSSA 算法在所有函数上均获得了较高的解精度和收敛速度。与其他 9 种最新的群体智能算法相比,OOSSA 在求解精度和收敛速度方面同样表现出较大优势。

4 OOSSA 的工程优化应用案例

4.1 压力容器设计问题

从文献[44]中选取圆柱形压力容器设计优化问题来测试 OOSSA 算法解决实际工程设计问题的能力。压力容器设计问题的目标为最小化制作成本。其中制作成本包括焊接、材料和成型 3 部分。为解决该问题,需要确定管壁厚度 T_s,顶盖壁厚 T_h,内壁直径 R 和容器管身长度 L 这 4 个设计变量的最佳值。首先,将设计变量编号,得到 $\boldsymbol{x}=[x_1\ x_2\ x_3\ x_4]=[T_s\ T_h\ R\ L]$。问题的目标函数 f、优化约束 g 和变量取值范围如下所示:

$$f(\boldsymbol{x})=0.622\,4x_1x_2x_3+1.778\,1x_2x_3^2+3.166\,1x_1^2x_4+19.84x_1^2x_3$$

$$\begin{cases} g_1(\boldsymbol{x})=-x_1+0.019\,3x_3\leqslant 0 \\ g_2(\boldsymbol{x})=-x_3+0.009\,54x_3\leqslant 0 \\ g_3(\boldsymbol{x})=-\pi x_3^2x_4-\dfrac{4}{3}\pi x_3^3+1\,296\,000\leqslant 0 \\ g_4(\boldsymbol{x})=x_4-240\leqslant 0 \end{cases}$$

式中:$x_1\leqslant 0$,$x_2\leqslant 99$,$x_3\leqslant 10$,$x_4\leqslant 200$。

利用 OOSSA 算法求解压力容器设计问题,并与文献[44]中的几种先进算法进行比较。根据文献[44],将 OOSSA 的种群规模设置为 30,最大函数评价次数设置为 15 000,独立运行 30 次后取最优解。将所得结果以及文献[44]中已有的实验结果列于表 7。从表中可以看出,OOSSA 获得了最低的设计成本 5 933.766 65,明显优于其他 10 种对比算法,进一步证明 OOSSA 算法能够有效应用于工程优化问题。

表 7 各算法的压力容器设计结果对比

Tab. 7 Comparison of optimization results of different algorithms for press vessel design

算法	顶盖壁厚 T_h	管壁厚度 T_s	管身长度 L	内壁直径 R	成本
OOSSA	0.790 568	0.399 794	40.962 097 37	191.245 501 0	5 933.766 65
SSA	0.790 678	0.390 834	40.967 738 75	195.918 220 0	6 012.188 50
GWO	0.812 500	0.434 500	42.098 181 00	176.758 730 0	6 051.563 90
WOA	0.812 500	0.437 500	42.098 269 90	176.639 000 0	6 059.741 00
MFO	0.812 500	0.437 500	42.098 445 00	176.636 596 0	6 059.714 30
MVO	0.812 500	0.437 500	42.090 738 20	176.738 690 0	6 060.806 60
MPA	0.812 500	0.437 500	42.098 445 00	176.636 607 0	6 059.714 40
EO	0.812 500	0.437 500	42.098 445 60	176.636 595 8	6 059.714 30
HHO	0.817 584	0.407 293	42.009 174 57	176.071 964 0	6 000.462 59
SCA	0.812 500	0.433 750	42.048 610 00	177.707 800 0	6 076.365 10
CSS	0.812 500	0.437 500	42.103 624 00	176.572 656 0	6 059.088 80

5　基于 OOSSA 的移动机器人路径规划

随着智能化时代的到来，自主移动机器人被广泛应用到越来越多的领域中，如智能送餐、智能仓储、军事战争等。在机器人学中，路径规划是一项关键技术。传统的路径规划算法在面对复杂地形时难以生成理想的轨迹。因此，研究人员试图使用群体智能优化算法解决机器人路径规划问题。智能优化算法在搜索过程中表现出较强的随机性，因此在规划路径时表现出的性能有较大差距。提出一种收敛速度快、鲁棒性强、环境适应能力好的算法是非常有意义的。基于上述分析，本节提出一种基于 OOSSA 的自主移动机器人路径规划算法。

5.1　编码

在设计路径规划算法时，采用导航点模型构建机器人的工作环境设置，并采用三次样条插值对路径进行平滑。其中，导航点能够反映所规划路径的转向次数，因此将路径上所有导航点编码为一个樽海鞘个体。

5.2　构建适应度函数

为机器人规划路径时，需要考虑两个因素：1）路径长度最短；2）避开所有障碍物。基于这两个因素，本文设计了一种高效的适应度函数，以评价路径的质量，见式（14）。

$$f = L \cdot (1 + \rho \cdot \tau) \tag{14}$$

式中 ρ 为路径非法度放大系数，用于排除有碰撞的路径，取值为 1 000。L 为当前路径的长度，按照下式进行计算

$$L = \sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - x_i)^2} \tag{15}$$

式中 (x_i, y_i) 为插值点 i 的坐标。

τ 为路径非法度，按照下式进行计算

$$\tau = \sum_{k=1}^{s} \sum_{j=1}^{m} \max\left(1 - \frac{d_{j,k}}{r_k}, 0\right) \tag{16}$$

式中：s 为障碍物的数量，m 为路径中插值点的数量，$d_{j,k}$ 为第 j 个插值点到第 k 个障碍物的距离，r_k 为第 k 个障碍物的半径。根据式（16）可知，如果路径过障碍，则 $\tau > 0$，反之，$\tau = 0$。

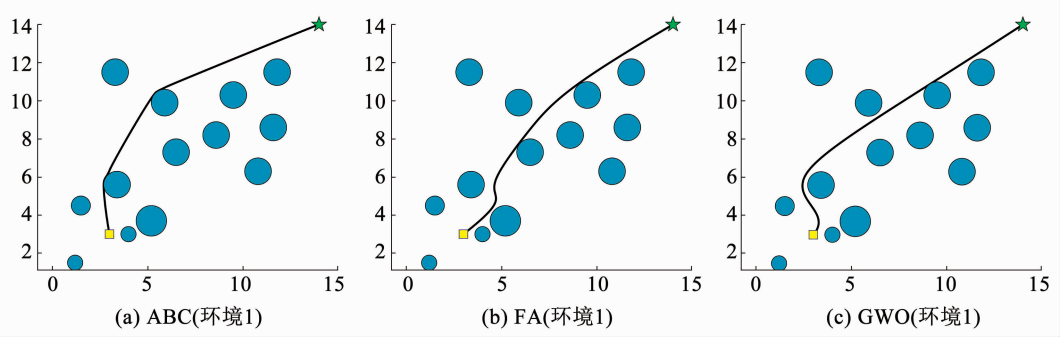
5.3　实验设置

本节对所提算法进行仿真实验，并与基本 SSA 算法、人工蜂群算法（ABC）、粒子群算法（PSO）、灰狼优化器（GWO）和萤火虫算法（FA）进行对比。设置所有算法的种群规模均为 30，最大函数评价次数均为 15 000，对比算法的参数与原始文献保持一致。为使实验结果更加可靠，选取文献[45]中的两种测试环境进行仿真，每种算法为每个地形独立规划 30 次路径，并记录最优路径长度和平均寻优耗时。比较结果见表 8。各算法获得的路径见图 4。

表 8　不同环境下 6 种算法所求路径长度与平均寻优耗时比较

Tab. 8　Comparison of path lengths and average search time of six algorithms in different environments

测试环境	指标	PSO	FA	ABC	GWO	SSA	OOSSA
环境 1	路径长度	17.221 9	16.153 9	17.481 8	17.351 7	16.997 9	15.874 4
	时间/s	0.240 8	2.483 1	0.291 1	0.244 9	0.251 3	0.274 5
环境 2	路径长度	16.192 5	16.303 1	16.291 9	16.328 1	16.815 7	15.933 9
	时间/s	0.339 6	3.146 5	0.406 6	0.354 7	0.356 9	0.371 8



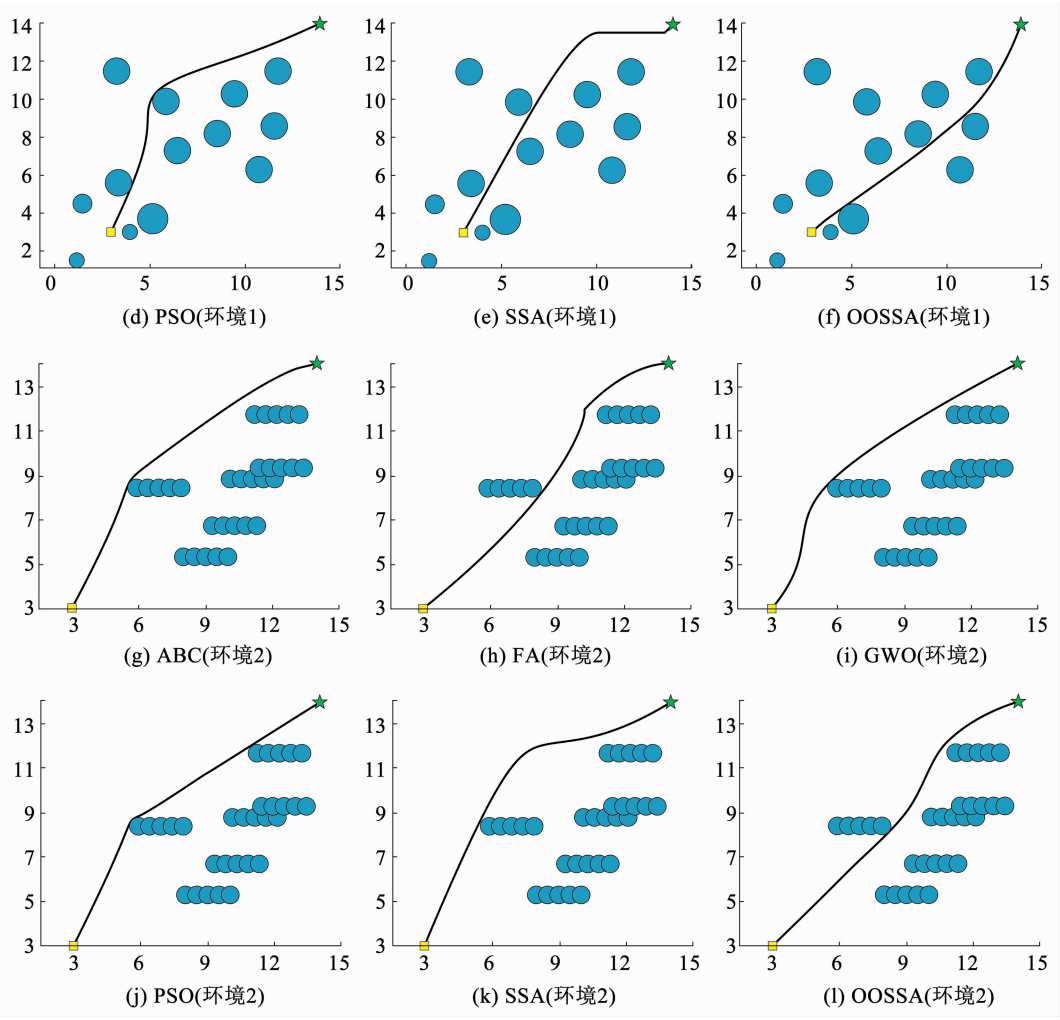


图 4　不同算法的规划路径对比

Fig. 4　Comparison of paths planned by different algorithms

根据表 8 可知,在两种不同的地形设置中,所提算法得到的最优路径长度均优于对比算法。在平均寻优耗时方面,OOSSA 算法与 PSO、ABC、GWO 和 SSA 算法耗时相近,FA 算法则消耗了较长的时间。从图 4 可以看出,6 种算法在两种环境设置中所规划的路径均能避开所有障碍物。而 OOSSA 均可生成最合理的轨迹以帮助机器人从起点移动到终点,这是因为 OLOBL 策略能够降低算法陷入局部最优的概率,使算法在迭代前期能够规划出较为合理的轨迹。在迭代后期,引入非线性惯性权重的跟随者位置更新模式能够帮助算法不断调整该路径,进一步降低了路径长度。

6　结　论

本文提出一种基于正交设计的折射学习樽海鞘群算法。首先利用透镜折射反向学习和正交试验设计构造了一种正交折射学习策略 OLOBL,并应用到 SSA 中,提高算法勘探能力的同时解决了反向学习存在的维度退化问题。其次,在跟随者位置更新阶段引入自适应惯性权重因子,以平衡算法的全局勘探和局部开采能力。对 12 个 CEC2017 基准测试函数、1 个工程优化问题和移动机器人路径规划问题进行仿真实验研究,并通过多种方式详细分析了所提算法的全局勘探、局部开采和跳出局部最优的能力。研究结果表明:OOSSA 算法能够有效提升基本 SSA 算法的性能,其在整体性能上要明显优于多种当前最先进的 SSA 变体和多种最新开发的群体智能优化算法。OOSSA 在解决工程设计优化问题和机器人路径规划问题时,同样获得了满意的结果。在未来的研究中,OOSSA 算法将与光伏电池模块参数识别等问题相结合,以期解决更多实际工程设计优化问题。

参考文献

[1] MIRJALILI S, GANDOMI A H, MIRJALILI S Z, et al. Salp swarm algorithm: a bio-inspired optimizer for engineering design problems [J]. Advances in Engineering Software, 2017, 114: 163. DOI: 10.1016/j.advengsoft.2017.07.002

[2] LIANG Haibo, ZOU Jialing, ZUO Kai, et al. An improved genetic

algorithm optimization fuzzy controller applied to the wellhead back pressure control system [J]. Mechanical Systems and Signal Processing, 2020, 142: 106708. DOI: 10.1016/j.ymssp.2020.106708

[3] 张艺瀛，金志刚. 一种高维多模态优化的量子粒子群优化算法[J]. 哈尔滨工业大学学报，2018，50(11)：50
ZHANG Yiying, JIN Zhigang. Quantum particle swarmoptimization algorithm for high-dimensional multi-model optimization[J]. Journal of Harbin Institute of Technology, 2018, 50(11): 50. DOI: 10.11918/j.issn.0367-6234.201806065

[4] WANG Zongshan, DING Hongwei, LI Bo, et al. An energy efficient routing protocol based on improved artificial bee colony algorithm for wireless sensor networks[J]. IEEE Access, 2020, 8: 133577. DOI: 10.1109/ACCESS.2020.3010313

[5] 李长安，谢宗奎，吴忠强，等. 改进灰狼算法及其在港口泊位调度中的应用[J]. 哈尔滨工业大学学报，2021，53(1)：101
LI Chang'an, XIE Zongkui, WU Zhongqiang, et al. Improved grey wolf algorithm and its application in port berth scheduling [J]. Journal of Harbin Institute of Technology, 2021, 53(1): 101. DOI: 10.11918/201911117

[6] MIRJALILI S, LEWIS A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51. DOI: 10.1016/j.advengsoft.2016.01.008

[7] WANG Zongshan, DING Hongwei, LI Bo, et al. Energy efficient cluster based routing protocol for WSN using firefly algorithm and ant colony optimization[J]. Wireless Personal Communications, 2022: 1. DOI: 10.1007/s11277-022-09651-9

[8] KHISHE M, MOHAMMADI H. Passive sonar target classification using multi-layer perceptron trained by salp swarm algorithm[J]. Ocean Engineering, 2019, 181: 98. DOI: 10.1016/j.oceaneng.2019.04.013

[9] TUBISHAT M, JA'AFAR S, ALSWAITTI M, et al. Dynamic salp swarm algorithm for feature selection [J]. Expert Systems with Applications, 2020, 164: 113873. DOI: 10.1016/j.eswa.2020.113873

[10] 刘森，贾志成，陈雷，等. 基于樽海鞘群体优化非负矩阵分解的高光谱图像解混算法[J]. 计算机辅助设计与图形学学报，2019，31(2)：315
LIU Sen, JIA Zhicheng, CHEN Lei, et al. Hyperspectral images unmixing based on nonnegative matrix factorization optimized by salp swarm algorithm [J]. Journal of Computer-Aided Design & Computer Graphics, 2019, 31(2): 315. DOI: 10.3724/SP.J.1089.2019.17189

[11] FATHY A, REZK H, NASSEF A M. Robust hydrogen-consumption-minimization strategy based salp swarm algorithm for energy management of fuel cell/supercapacitor/batteries in highly fluctuated load condition [J]. Renewable Energy, 2019, 139: 147. DOI: 10.1016/j.renene.2019.02.076

[12] 陈雷，蔺悦，康志龙. 基于衰减因子和动态学习的改进樽海鞘群算法[J]. 控制理论与应用，2020，37(8)：1766
CHEN Lei, LIN Yue, KANG Zhilong. Improved salp swarm algorithm based on reduction factor and dynamic learning [J]. Control Theory & Applications, 2020, 37(8): 1766. DOI: 10.7641/CTA.2020.90766

[13] 陈忠云，张达敏，辛梓芸. 多子群的共生非均匀高斯变异樽海鞘群算法[J/OL]. 自动化学报[2021-10-20]
CHEN Zhongyun, ZHANG Damin, XIN Ziyun. Multi-subpopulation based symbiosis and non-uniform Gaussian mutation salp swarm algorithm[J/OL]. Acta Automatica Sinica[2021-10-20]. DOI: 10.16383/j.aas.c190684

[14] 张达敏，陈忠云，辛梓芸，等. 基于疯狂自适应的樽海鞘群算法[J]. 控制与决策，2020，35(9)：2112
ZHANG Damin, CHEN Zhongyun, XIN Ziyun, et al. Salp swarm algorithm based on craziness and adaptive [J]. Control and Decision, 2020, 35(9): 2112. DOI: 10.13195/j.kzyjc.2019.0012

[15] 刘景森，袁蒙蒙，左方. 面向全局搜索的自适应领导者樽海鞘群算法[J]. 控制与决策，2021，36(9)：2152
LIU Jingsen, YUAN Mengmeng, ZUO Fang. Global search-oriented adaptive leader salp swarm algorithm[J]. Control and Decision, 2021, 36(9): 2152. DOI: 10.13195/j.kzyjc.2020.0090

[16] QAIS M H, HASANIEN H M, ALGHUWAINEM S. Enhanced salp swarm algorithm: application to variable speed wind generators[J]. Engineering Applications of Artificial Intelligence, 2019, 80: 82. DOI: 10.1016/j.engappai.2019.01.011

[17] GHOLAMI K, PARVANEH M H. A mutated salp swarm algorithm for optimum allocation of active and reactive power sources in radial distribution systems [J]. Applied Soft Computing, 2019, 85: 105833. DOI: 10.1016/j.asoc.2019.105833

[18] IBRAHIM R A, EWEES A A, OLIVA D, et al. Improved salp swarm algorithm based on particle swarm optimization for feature selection [J]. Journal of Ambient Intelligence and Humanized Computing, 2019, 10(8): 3155. DOI: 10.1007/s12652-018-1031-9

[19] WOLPERT D H, MACREADY W G. No free lunch theorems for optimization[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67. DOI: 10.1109/4235.585893

[20] TIZHOOSH H R. Opposition-based learning: a new scheme for machine intelligence[C]//Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce. Vienna: IEEE, 2005: 695. DOI: 10.1109/CIMCA.2005.1631345

[21] RAHNAMAYAN S, TIZHOOSH H R, SALAMA M M A. Opposition versus randomness in soft computing techniques[J]. Applied Soft Computing, 2008, 8(2): 906. DOI: 10.1016/j.asoc.2007.07.010

[22] RAHNAMAYAN S, TIZHOOSH H R, SALAMA M M A. Opposition-based differential evolution[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 64. DOI: 10.1109/TEVC.2007.894200

[23] 喻飞，李元香，魏波，等. 透镜成像反学习策略在粒子群算法中的应用[J]. 电子学报，2014，42(2)：230
YU Fei, LI Yuanxiang, WEI Bo, et al. The application of a novel OBL based on lens imaging principle in PSO. Acta Electronica Sinica, 2014, 42(2): 230. DOI: 10.3969/j.issn.0372-2112.2014.02.004

[24] 龙文，伍铁斌，唐明珠，等. 基于透镜成像学习策略的灰狼优化算法[J]. 自动化学报，2020，46(10)：2148
LONG Wen, WU Tiebin, TANG Mingzhu, et al. Grey wolf optimizer algorithm based on lens imaging learning strategy[J]. Acta Automatica Sinica, 2020, 46(10): 2148. DOI: 10.3969/j.issn.0372-2112.2014.02.004

[25] ZHANG Hongliang, HEIDARI A A, WANG Mingjing, et al. Orthogonal Nelder-Mead moth flame method for parameters identification of photovoltaic modules[J]. Energy Conversion and

Management, 2020, 211: 112764. DOI: 10. 1016/j. enconman. 2020. 112764

[26] PARK S Y, LEE J J. Stochastic opposition-based learning using a beta distribution in differential evolution[J]. IEEE Transactions on Cybernetics, 2016, 46(10): 2184. DOI: 10. 1109/TCYB. 2015. 2469722

[27] BRAIK M, SHETA A, TURABIEH H, et al. A novel lifetime scheme for enhancing the convergence performance of salp swarm algorithm[J]. Soft Computing, 2021, 25(1): 181. DOI: 10. 1007/s00500 - 020 - 05130 - 0

[28] SALGOTRA R, SINGH U, SINGH S, et al. Self-adaptive salp swarm algorithm for engineering optimization problems[J]. Applied Mathematical Modelling, 2020, 89: 188. DOI: 10. 1016/j. apm. 2020. 08. 014

[29] NAUTIYAL B, PRAKASH R, VIMAL V, et al. Improved salp swarm algorithm with mutation schemes for solving global optimization and engineering problems [J]. Engineering with Computers, 2021: 1. DOI: 10. 1007/s00366 - 020 - 01252 - z

[30] HUSSIEN A G. An enhanced opposition-based salp swarm algorithm for global optimization and engineering problems[J]. Journal of Ambient Intelligence and Humanized Computing, 2021, 13(1): 1. DOI: 10. 1007/s12652 - 021 - 02892 - 9

[31] OZBAY F A, ALATAS B. Adaptive salp swarm optimization algorithms with inertia weights for novel fake news detection model in online social media[J]. Multimedia Tools and Applications, 2021, 80(26): 1. DOI: 10. 1007/s11042 - 021 - 11006 - 8

[32] REN Hao, LI Jun, CHEN Huiling, et al. Stability of salp swarm algorithm with random replacement and double adaptive weighting [J]. Applied Mathematical Modelling, 2021, 95: 503. DOI: 10. 1016/j. apm. 2021. 02. 002

[33] SAAFAN M M, EL-GENDY E M. IWOSSA: an improved whale optimization salp swarm algorithm for solving optimization problems [J]. Expert Systems with Applications, 2021, 176: 114901. DOI: 10. 1016/j. eswa. 2021. 114901

[34] DERRAC J, CARCIA S, MOLINA D, et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms [J]. Swarm and Evolutionary Computation, 2011, 1(1): 3. DOI: 10. 1016/j. swevo. 2011. 02. 002

[35] FARAMARZI A, HEIDARINEHAD M, MIRJALILI S, et al. Marine predators algorithm: a nature-inspired metaheuristic[J]. Expert Systems with Applications, 2020, 152: 113377. DOI: 10. 1016/j. eswa. 2020. 113377

[36] FARAMARZI A, HEIDARINEJAD M, STEPHENS B, et al. Equilibrium optimizer: a novel optimization algorithm [J]. Knowledge-Based Systems, 2020, 191: 105190. DOI: 10. 1016/j. knosys. 2019. 105190

[37] DHARGUPTA S, GHOSH M, MIRJALILI S, et al. Selective opposition based grey wolf optimization[J]. Expert Systems with Applications, 2020, 151: 113389. DOI: 10. 1016/j. eswa. 2020. 113389

[38] NADIMI-SHAHRAKI M H, TAGHIAN S, MIRJALILI S. An improved grey wolf optimizer for solving engineering problems[J]. Expert Systems with Applications, 2021, 166: 113917. DOI: 10. 1016/j. eswa. 2020. 113917

[39] ABUALIGAH L, DIABAT A, MIRJALILI S, et al. The arithmetic optimization algorithm [J]. Computer Methods in Applied Mechanics and Engineering, 2021, 376: 113609. DOI: 10. 1016/j. cma. 2020. 113609

[40] HASHIM F A, HUSSAIN K, HOUSSEIN E H, et al. Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems[J]. Applied Intelligence, 2021, 51(3): 1531. DOI: 10. 1007/s10489 - 020 - 01893 - z

[41] MA Lei, WANG Chao, XIE Nenggang, et al. Moth-flame optimization algorithm based on diversity and mutation strategy[J]. Applied Intelligence, 2021, 51(8): 5836. DOI: 10. 1007/s10489 - 020 - 02081 - 9

[42] SHAN Weifeng, QIAO Zenglin, HEIDARI A A, et al. Double adaptive weights for stabilization of moth flame optimizer: balance analysis, engineering cases, and medical diagnosis [J]. Knowledge-Based Systems, 2021, 214: 106728. DOI: 10. 1016/j. knosys. 2020. 106728

[43] YU Xiaobing, XU Wangying, LI Chenliang. Opposition-based learning grey wolf optimizer for global optimization[J]. Knowledge-Based Systems, 2021, 226: 107139. DOI: 10. 1016/j. knosys. 2021. 107139

[44] HEIDARI A A, MIRJALILI S, FARIS H, et al. Harris hawks optimization: algorithm and applications[J]. Future Generation Computer Systems, 2019, 97: 849. DOI: 10. 1016/j. future. 2019. 02. 028

[45] AGARWAL D, BHARTI P S. Implementing modified swarm intelligence algorithm based on Slime moulds for path planning and obstacle avoidance problem in mobile robots[J]. Applied Soft Computing, 2021, 107: 107372. DOI: 10. 1016/j. asoc. 2021. 107372

(编辑 苗秀芝)