DOI:10.19652/j.cnki.femt.2406049

融合对偶学习的动态蜘蛛蜂优化算法及其应用*

沈倩雯 张达敏
(贵州大学大数据与信息工程学院 贵阳 550000)

摘 要:针对经典蜘蛛蜂优化算法初始种群分布不合理、搜索与开发之间的转换不平衡、易陷入局部最优等问题,提出了一种融合对偶学习的动态蜘蛛蜂优化算法(dynamic spider wasp optimizer combined with duality learning, CLDSWO)。首先,结合 Tent 和 Sinusoidal 映射,设计了 TS(Tent-Sinusoidal)映射,并采用 TS 映射生成分布更广泛且均匀的初始蜘蛛蜂种群。其次,设计了一个动态权衡因子,自适应地调整狩猎和交配行为之间的转换,实现全局搜索和局部优化之间的平衡。引入了基于对偶学习的变异机制,在对偶学习的过程中,引入逐维变异机制,加速算法的收敛,增强逃离局部最优的能力。为了验证 CLDSWO 算法的有效性,利用 10 个基准函数和 CEC2017 函数进行实验,并通过 Wilcoxon 检验证实仿真结果的显著性,实验结果表明,CLDSWO 在平衡收敛精度和速度方面更具竞争力。将 CLDSWO 算法应用至压力容器设计问题和无源时差定位问题中,结果表明 CLDSWO 的精度分别提升了 1.28%和 36.67%,验证了 CLDSWO 算法在求解实际工程应用问题中的有效性。

关键词:蜘蛛蜂优化算法;动态权衡因子;对偶学习;逐维变异;工程应用
中图分类号: TP301.6 **文献标识码**:A **国家标准学科分类代码**: 52010

Dynamic spider wasp optimizer incorporating duality learning and its application

Shen Qianwen Zhang Damin
(School of Big Data and Information Engineering, Guizhou University, Guiyang 550000, China)

Abstract: The spider wasp optimizer has problems such as irrational initial population distribution, unbalanced transition between search and exploitation, and a tendency to fall into local optimization. Therefore, a dynamic spider wasp optimizer combined with duality learning (CLDSWO) is proposed to solve these problems. Firstly, the Tent-Sinusoidal (TS) mapping which combines the Tent and Sinusoidal mapping is designed to generate the initial spider-wasp population with a wider and uniform distribution. Secondly, a dynamic tradeoff factor is developed to adaptively adjust the tradeoff between hunting and mating behaviors to achieve a balance between global search and local optimization. Finally, a mutation mechanism based on duality learning is introduced to accelerate the convergence and enhance the ability to escape from the local optimum. To verify the effectiveness of CLDSWO, 10 benchmark functions, CEC2017 functions, and Wilcoxon tests are carried out. The results show that CLDSWO is more competitive in balancing convergence accuracy and speed. The CLDSWO algorithm is applied to the pressure vessel design problem and the time difference of arrival localization problem. The results show that the accuracy of CLDSWO was improved by 1.28% and 36.67%, respectively, validating the effectiveness of CLDSWO in solving practical engineering applications.

Keywords: spider wasp optimizer; dynamic tradeoff factor; duality learning; dimensional mutation; engineering applications

收稿日期:2024-03-11
* **基金项目**:国家自然科学科学基金(62166006)项目资助

0 引 言

随着人工智能领域的发展,研究学者开始关注模拟自然界生物行为的优化算法,以解决复杂的优化问题。群智能优化算法通过模拟动物社会行为,达到全局搜索和优化目标函数的目的。常见的群智能优化算法包括鲸鱼优化算法(whale optimization algorithm, WOA)[1]、哈里斯鹰优化算法(Harris Hawks optimization, HHO)[2]、海洋捕食者算法(marine predator algorithm, MPA)[3]、蜜獾优化算法(honey badger algorithm, HBA)[4]、蜣螂优化算法(dung beetle optimizer, DBO)[5]等,它们在不同的领域和问题上都展现出优越的性能。尽管这类算法由于高灵活性、鲁棒性和易于实现被广泛关注,但它们也存在一系列局限性和问题。首先,群智能算法的性能高度依赖于问题的特性,对于某些复杂的多模态问题,算法可能陷入局部最优解,难以收敛。其次,算法的收敛速度和搜索效率极易受到问题维度的影响,导致高维空间中无法找到全局最优解。另一方面,群智能算法对问题的初始解敏感,不同的初始条件可能导致不同的最优解或收敛速度,这使得算法在实际的工程应用中难以稳定地工作,需要选择有效的启动策略。

针对上述问题,许多学者提出了多种改进策略,以增强群智能算法的搜索性能。Song 等[6]为了提高哈里斯鹰优化算法的性能,提出了一种高斯变异和维数决定的哈里斯鹰算法(dimension decided Harris Hawks optimization with Gaussian mutation, GCHHO),提高算法跳出局部最优的能力,充分挖掘搜索区域。Wang 等[7]提出了一种增强型的樽海鞘群算法,采用无标度网络和自适应权重实现探索和开发的平衡。张帅等[8]在哈里斯鹰算法中集成了正态云模型和动态扰动策略(improved Harris Hawks optimization, IHHO),提高了种群多样性和收敛速度。朱文昌等[9]在麻雀搜索算法中集成了 Iterative 混沌映射、惯性权重因子和 Levy 变异策略。文昌俊等[10]提出了一种引入改进迭代局部搜索的灰狼算法,通过佳点集策略、双收敛因子、欧氏动态权重等措施,提高了算法的收敛速度和搜索精度。陈麒羽等[11]针对非洲秃鹫优化算法的缺陷,提出了一种双重动态调整策略,提高了算法的多样性和候选解的质量。

然而,尽管群智能算法的改进取得了一定的成功,但其发展并未达到巨大的规模。这是因为群智能算法对于所解决问题的依赖性较高,不存在一种算法适合于所有的优化问题,这也激励学者们不断开发新的群智能优化算法,以应对不同优化问题的挑战。蜘蛛蜂优化算法(spider wasp optimizer,SWO)[12]是最近提出的一种新型群智能算法。该算法受雌性蜘蛛黄蜂在自然界中的狩猎、筑巢和交配行为启发。首先,雌性蜘蛛黄蜂在周围环境中寻找合适的猎物(蜘蛛),在发现猎物后,猎物可能会试图逃离,因此,雌黄蜂会跟随它们,伺机将它们麻痹并拖到预先准备好的巢穴中。在找到合适的猎物并把猎物拖进巢穴后,黄蜂会在蜘蛛腹部产卵并关闭巢穴。此外,在搜索空间中,雄黄蜂和雌黄蜂之间以特定的概率进行统一交叉操作来孵化后代。蜘蛛蜂优化算法具有多种不同的更新策略,具有搜索速度快、求解精度高等优势。同时,算法采用一种动态种群策略,当雌黄蜂产卵后关闭巢穴后,其功能被委托给其他黄蜂,在迭代过程中,终止这些黄蜂有助于加速收敛至最优解。

但是,作为受自然界生物启发的群智能算法,蜘蛛蜂优化算法仍具有收敛速度较慢、求解精度不足等缺陷。为了克服这些局限性,国内外的学者对蜘蛛蜂优化算法进行了改进,并将其成功应用于各种工程问题中。Mostafa 等[13]提出将 SWO 算法的机制集成到 DE 的突变策略中,用于求解特征选择问题。Fang 等[14]将 SWO 算法应用至输气压缩机设计问题中。Saber 等[15]通过将 SWO 算法与局部搜索策略集成,进一步增强了算法的搜索能力。然而,这些开发的蜘蛛蜂优化器大多基于特定的问题框架,其在更广泛的工程问题中的性能尚未得到充分验证。因此,对蜘蛛蜂优化算法的进一步研究和改进仍然是必要的,以提高其应用的范围和效果。

本文针对 SWO 的求解精度不高和搜索性能低等缺点,提出了一种融合对偶学习的动态蜘蛛蜂优化算法(CLDSWO)。首先,将 Tent 和 Sinusoidal 混沌映射结合,产生初始化种群,使用动态的权衡因子来平衡蜘蛛蜂勘探和开发之间的能力,最后提出了一种基于对偶学习的变异机制,在对偶学习过程中引入逐维变异策略,促进算法的收敛和优化过程。利用 10 个基准测试函数和 10 个 CEC2017 测试函数进行性能验证,同时利用 Wilcoxon 秩和检验来验证算法差距的显著性,最后将 CLDSWO 应用至压力容器设计问题和无源时差定位(TDOA)优化问题中,验证改进算法在实际工程问题中的有效性。

1 经典蜘蛛蜂优化算法

蜘蛛蜂优化算法的灵感来自蜘蛛蜂的狩猎、筑巢和交配行为。

1.1 初始种群的产生

在 SWO 算法中,每只雌性蜘蛛黄蜂代表当前一代的一个解,假设在 D 维搜索空间中随机生成 N 个蜘蛛蜂个体作为初始的蜘蛛蜂种群,在搜索空间中随机生成任意解如下:

$$SW_i^t = LB + r \times (UB - LB) \tag{1}$$

式中:SW_i^t 为第 t 次迭代中第 i 个个体的位置;r 是 0～1 之间的随机数;UB 和 LB 是预定义的参数上下界。

1.2 狩猎和筑巢行为

雌性蜘蛛蜂在搜索空间内寻找合适的猎物,这一过程为搜索阶段。之后,它会追逐猎物,这一阶段称为围追阶段。最后一个阶段,蜘蛛蜂会把捕捉到的蜘蛛拖进预先准备的巢穴,在它的腹部产卵。

1)搜索阶段(探索)

在这一阶段,雌蜂以恒定步长随机探索空间,每只雌蜂的位置更新如下:

$$SW_i^{t+1} = SW_i^t + \mu_1 \cdot (SW_a^t - SW_b^t) \tag{2}$$

式中:SW_a 和 SW_b 是从种群中随机选取的两个个体;利用 μ_1 确定通过当前方向的恒定运动。

$$\mu_1 = |rn| \cdot r_1 \tag{3}$$

式中:r_1 是在 0～1 之间生成的随机数;rn 是使用正态分布生成的随机数。

雌蜂有时会找不到从球上掉下的蜘蛛,因此,他们需要搜索蜘蛛掉落的确切地点周围的整个区域,此时雌蜂的位置更新方程为:

$$SW_i^{t+1} = SW_c^t + \mu_2 \cdot (LB + r_2 \cdot (UB - LB)) \tag{4}$$

$$\mu_2 = B \cdot \cos 2\pi l \tag{5}$$

$$B = \frac{1}{1+e^l} \tag{6}$$

式中:SW_c 是随机选取的雌蜂位置,代表掉落蜘蛛的位置;l 是 1～−2 之间的随机数。

最后,下一代雌蜂的位置更新随机实现如下:

$$S\_SW_i^{t+1} = \begin{cases} SW_i^t + \mu_1 \cdot (SW_a^t - SW_b^t), & r_3 < r_4 \\ SW_c^t + \mu_2 \cdot (LB + r_2 \cdot (UB - LB)), & \text{其他} \end{cases} \tag{7}$$

式中:$S\_SW_i^{t+1}$ 是在搜索阶段随机生成的雌蜂个体;r_3 和 r_4 是 0～1 的两个随机数。

2)围追阶段(探索和开发)

在找到猎物后,雌蜂会对蜘蛛进行诱捕。此时,雌蜂的位置更新公式为:

$$SW_i^{t+1} = SW_i^t + C \cdot |2 \cdot \boldsymbol{r}_5 \cdot SW_a^t - SW_i^t| \tag{8}$$

$$C = \left(2 - 2\left(\frac{t}{t_{\max}}\right)\right) \cdot r_6 \tag{9}$$

式中:SW_a 是随机选取的雌蜂个体;C 是决定雌蜂速度的距离控制因子;t 和 $t_{\max}$ 代表当前迭代次数和最大迭代次数;$\boldsymbol{r}_5$ 是一个向量,其值在 0～1 随机生成;r_6 是 0～1 的随机数。

在诱捕时,蜘蛛会进行逃离。此时,雌蜂与蜘蛛间的距离逐渐增加,使用以下公式模拟此行为:

$$SW_i^{t+1} = SW_i^t \cdot \boldsymbol{vc} \tag{10}$$

式中:$\boldsymbol{vc}$ 是在 k ～ $-k$ 由正态分布产生的向量,k 由式(11)生成。

$$k = 1 - \left(\frac{t}{t_{\max}}\right) \tag{11}$$

这两种不同行为趋势之间的权衡由下式实现:

$$F\_SW_i^{t+1} = \begin{cases} SW_i^t + C \cdot |2 \cdot \boldsymbol{r}_5 \cdot SW_a^t - SW_i^t|, & r_3 < r_4 \\ SW_i^t \cdot \boldsymbol{vc}, & \text{其他} \end{cases} \tag{12}$$

式中:$F\_SW_i^{t+1}$ 是在围追阶段随机生成的雌蜂个体;r_3 和 r_4 是 0～1 的两个随机数。

在优化过程开始时,所有雌蜂将应用探索机制对优化问题进行全局探索,然后采取围追机制,在迭代过程中探索和利用当前雌蜂周围的区域,避免陷入局部最小值。最后,根据以下公式来调整搜索阶段和围追阶段的转换:

$$SF\_SW_i^{t+1} = \begin{cases} S\_SW_i^{t+1}, & p < k \\ F\_SW_i^{t+1}, & \text{其他} \end{cases} \tag{13}$$

式中:$SF\_SW_i^{t+1}$ 是经过搜索或围追后随机生成的雌蜂个体;p 是 0～1 之间的随机数。

3)筑巢行为(开发)

蜘蛛蜂具有不同的筑巢行为,第 1 种是将蜘蛛拉向最适合蜘蛛的区域,并认为它是建造巢穴的最佳地点,模拟这种行为的方程式如下:

$$SW_i^{t+1} = SW^* + \cos(2\pi l) \cdot (SW^* - SW_i^t) \tag{14}$$

式中:SW^* 代表获得的局部最优解。第 2 个方程将在从种群中随机选择的雌蜂位置上建造巢穴,同时使用额外的步长来避免在同一位置上建造两个巢穴,方程设计如下:

$$SW_i^{t+1} = SW_a^t + r_3 \cdot |\gamma| \cdot (SW_a^t - SW_i^t) + (1 - r_3) \cdot U \cdot (SW_b^t - SW_c^t) \tag{15}$$

$$U = \begin{cases} 0, & r_4 > r_5 \\ 1, & \text{其他} \end{cases} \tag{16}$$

式中:r_3、r_4 和 r_5 都是在 0～1 之间产生的随机数;γ 是根据莱维飞行产生的数;SW_a、SW_b 和 SW_c 是从种群中随机选取的 3 个解。式(14)和(15)随机转换为:

$$N\_SW_i^{t+1} = \begin{cases} SW^* + \cos(2\pi l) \cdot (SW^* - SW_i^t), & r_3 < r_4 \\ SW_a^t + r_3 \cdot |\gamma| \cdot (SW_a^t - SW_i^t) + (1 - r_3) \cdot U \cdot (SW_b^t - SW_c^t), & \text{其他} \end{cases} \tag{17}$$

式中:$N\_SW_i^{t+1}$ 是在筑巢阶段随机生成的雌蜂个体。

最后,狩猎和筑巢行为间的权衡由下式实现:

$$SW_i^{t+1} = \begin{cases} SF\_SW_i^{t+1}, & i < Nk \\ N\_SW_i^{t+1}, & \text{其他} \end{cases} \tag{18}$$

式中:SW_i^{t+1} 为经过不同阶段最终生成的雌蜂个体;N 为种群数量,k 由式(11)定义。

1.3 交配行为

雌蜂与雄蜂以一定概率发生交配行为,蜘蛛蜂卵的产生公式如下:

$$SW_i^{t+1} = Crossover(SW_i^t, SW_m^t, CR) \tag{19}$$

式中:$Crossover$ 表示在 SW_i 和 SW_m 之间应用均匀交叉算子,其概率称为交叉率(CR);SW_i 和 SW_m 分别代表雌性和雄性蜘蛛蜂。雄性蜘蛛蜂生成公式为:

$$SW_m^{t+1} = SW_i^t + e^l \cdot |\beta| \cdot v_1 + (1 - e^l) \cdot |\beta_1| \cdot v_2 \tag{20}$$

$$v_1 = \begin{cases} x_a - x_i, & f(x_a) < f(x_i) \\ x_i - x_a, & \text{其他} \end{cases} \tag{21}$$

$$v_2 = \begin{cases} x_b - x_c, & f(x_b) < f(x_c) \\ x_c - x_b, & \text{其他} \end{cases} \tag{22}$$

式中：β 和 β_1 是根据正态分布随机生成的两个数；e 是指数常数；x_a、x_b 和 x_c 为从种群中随机选取的 3 个解，并且满足 $i \neq a \neq b \neq c$。

狩猎和交配行为之间的转换是基于一个预定义的因素，称为权衡率(TR)，当 $rand < TR$ 时，群体进行狩猎和筑巢行为，反之，则进行交配行为。

1.4 动态种群策略

在迭代过程中，群体中的雌蜂数量是动态变化的，当雌蜂完成产卵行为后，它会关闭巢穴，这意味着这只雌蜂在优化过程中的作用已经接近完成，将其功能委托给其他雌蜂有助于加速收敛，采取动态种群策略可以在算法的不同迭代阶段中适应性地调整种群数量，以避免算法陷入局部最优解中。在优化过程的早期阶段，种群数量较大，算法具有较强的全局搜索能力，可以发现问题空间中的潜在解空间。而在优化过程的后期阶段，种群数量逐渐减少，算法更加注重当前已经找到的最优解的精细化搜索，从而更容易收敛到全局最优解。新种群的长度更新如下：

$$N = N_{\min} + (N - N_{\min}) \times k \tag{23}$$

式中：$N_{\min}$ 表示在优化过程的不同阶段，为避免陷入局部最优使用的最小总体数量；$k = 1 - t/t_{\max}$ 控制着种群数量的动态调整速度，迭代初期较快地减少，以加快全局搜索速度，而在迭代后期，种群数量的变化速度减缓，使得算法更加稳定地收敛到最优解附近。

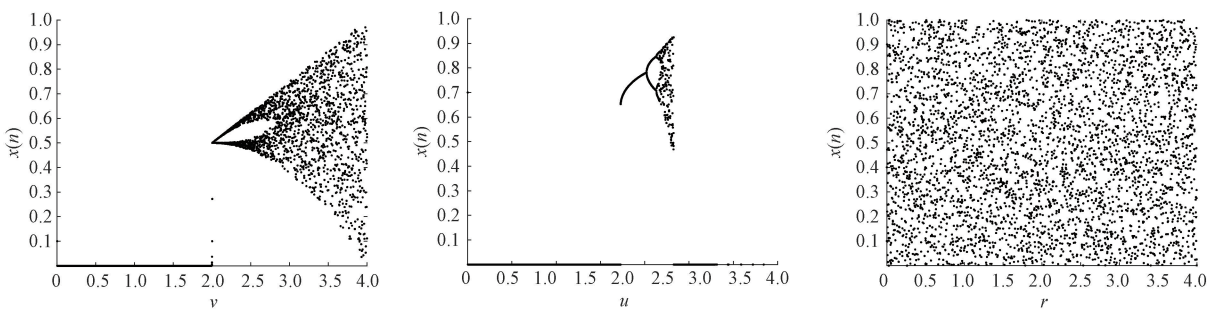
2 改进的蜘蛛蜂优化算法

2.1 Tent-Sinusoidal 混沌映射

群智能算法对于初始条件非常敏感，不同的初始条件可能导致不同的最优解或收敛速度，因此需要精心选择初始条件，确保算法的稳定运行。经典的蜘蛛蜂优化算法随机生成初始解的位置，这种方式简单快速，然而，随机生成可能导致初始解相对分散，采用混沌映射方式生成的初始解具有更多的结构性，有助于更快探索空间。Tent 映射[16]和 Sinusoidal 映射[16]是常用的两种混沌映射方式，本文根据这两种混沌映射方式，设计了一种新的 TS(Tent-Sinusoidal)混沌映射，定义如下：

$$x(t+1) = \begin{cases} [rx(t)^2 \sin \pi x(t) + 50(0.5 - 20r)x(t)] \bmod 1, & x(t) < 0.5 \\ [rx(t)^2 \sin \pi x(t) + 50(0.5 - 20r)(1 - x(t))] \bmod 1, & \text{其他} \end{cases} \tag{24}$$

式中：$r \in (0,4]$。不同映射方式的分岔图如图 1 所示。



(a) Tent 映射
(a) Tent mapping
(b) Sinusoidal 映射
(b) Sinusoidal mapping
(c) TS 映射
(c) TS mapping

图 1 不同映射分岔图

Fig. 1 Bifurcation diagram of different maps

由图 1 可以看到，TS 映射表现出良好混沌特性，输出的序列在[0,1]内均匀分布，因此采用 TS 映射初始化种群，能得到更加均匀且分散的初始位置。

2.2 动态权衡因子

在 SWO 算法中，蜘蛛蜂在迭代的初始阶段狩猎蜘蛛和建筑巢穴，而在迭代的后期进行产卵，这种机制是由蜘蛛蜂的生命周期和行为模式产生的。TR 决定蜘蛛蜂狩猎和交配行为之间的转换，然而，固定的 TR 很难保证算法每次都能得到很好的优化。为了在算法的不同迭代阶段中适应性地调整搜索策略，平衡全局搜索和局部开发的能力，从而提高算法的搜索效率和收敛精度。在算法的早期阶段，应该注重全局的探索和开发，因此 TR 值应该较大，使得搜索策略偏向于探索；而在算法的后期阶段，应该注重当前最优解的精细化搜索，因此 TR 值应该较小，使得搜索策略偏向于开发。同时，由于种群数量的变化会对搜索策略产生影响，因此在计算 TR 值时引入了动态变化的种群数量 N，使得 TR 值随着种群数量的变化而动态调整。按照以上思想，设计的自适应 TR 如下：

$$TR = \cos\left(0.5\pi\left(\frac{t}{t_{\max}}\right)\right) \Big/ \left(1 + \alpha \cdot \frac{N}{N_{\max}}\right) \tag{25}$$

式中：t 和 $t_{\max}$ 分别代表当前迭代次数和最大迭代次数；N 为当前种群数量；$N_{\max}$ 为最大种群数量，本文最大种群数量为初始种群数量；α 为(0,1)之间的影响因子。

2.3 基于对偶学习的变异机制

对偶学习是一种机器学习方法，它的基本理念是通过让两个“对手”在某个任务中进行竞争，从而提高它们的性能。受该思想启发，本文提出了一种基于对偶学习的变异机制，通过变异产生个体的“对手”，让两个个体相互竞争，

以促进优化效率，加快收敛速度。算法过程如下。

算法 1　基于对偶学习的变异机制

1. 随机选取 N_d 个个体作为对偶学习的原始样本 $\{w_k\}$
2. 使用交叉、维度变异策略产生 $\{w_k\}$ 的对偶 $\{w'_k\}$
3. For $i=1:N_d$
4.　If w'_k 优于 w_k
5.　　将种群中的 w_k 替换为 w'_k
6.　End if
7. End for

其中，N_d 为每一代生成对偶的个数，初始值等于种群个数，其值在迭代过程中动态变化，并按照如下公式更新：

$$N_d^{t+1}=\sigma^s N_d^t \tag{26}$$

$$N_d\in(N_{d,\min},N_{d,\max}) \tag{27}$$

$$s=(\delta N_d-N_v)/N \tag{28}$$

$$N_v=|\,w'_k:f(w'_k)<f(w_k)\,| \tag{29}$$

式中：N_v 是来自前一代的有效对偶个数，有效对偶个数是指由某个个体得到的对偶比此个体的性能要好；$\delta\in(0,1)$ 是一个决策阈值，若有效对偶的比例大于 δ，在下一代就生成更多的对偶，反之，在下一代生成对偶的个数会减少；$\sigma\in(0,1)$ 是控制自适应速度的常数。$N_{d,\min}$ 和 $N_{d,\max}$ 是 N_d 可取的最大值和最小值。

在确定每代生成的对偶对数个数后，需要通过交叉、变异操作来产生对偶个体。

1)交叉

本文采用遗传算法中的双点交叉算子[17]进行交叉操作，双点交叉是在两个父代个体的染色体中随机选择两个交叉点，然后交换这两个交叉点之间的基因片段，生成两个新的个体。相比于单点交叉，双点交叉在不显著增加计算成本的情况下，提高了交叉操作的多样性，有助于更好探索空间。

2)变异

本文中采用一种逐维变异策略对当前选择的 N_d 个个体进行变异操作，结合贪婪策略保留最优解，提高全局搜索能力，实现公式如下：

$$MC_{i,d}=\varepsilon\times x_{i,d_1}+(1-\varepsilon)\times x_{i,d_2} \tag{30}$$

式中：$MC_{i,d}$ 为产生的子代个体；d_1 和 d_2 是随机选取的维度指标；ε 是均匀分布在 0～1 内的随机数。

2.4　CLDSWO 实现步骤

CLDSWO 算法流程如图 2 所示，实现步骤如算法 2 所示。

算法 2　CLDSWO 算法

输入：种群规模 N，最小种群数量 $N_{\min}$，最大迭代次数 $t_{\max}$
输出：最优解 SW^*

1. 采用 TS 映射生成初始 N 个雌蜂个体
2. 计算每个雌蜂个体的适应度值 f
3. 找到具有最佳适应度值的个体 SW^*
4. While ($t<t_{\max}$)
5.　根据式(25)计算 TR 的值
6.　if ($r<TR$)
7.　　for $i=1:N$
8.　　按照 2.2 节进行狩猎和筑巢行为，更新个体位置
9.　　重新计算适应度值 $f(SW_i)$，并更新最优解
10.　　end for
11.　else
12.　　for $i=1:N$
13.　　按照 2.3 节进行交配行为，更新个体位置
14.　　重新计算适应度值 $f(SW_i)$，并更新最优解
14.　　end for
15.　end if
16.　按照 3.3 节的算法执行基于对偶学习的变异机制
17.　根据式(23)更新种群数量 N
18. end while

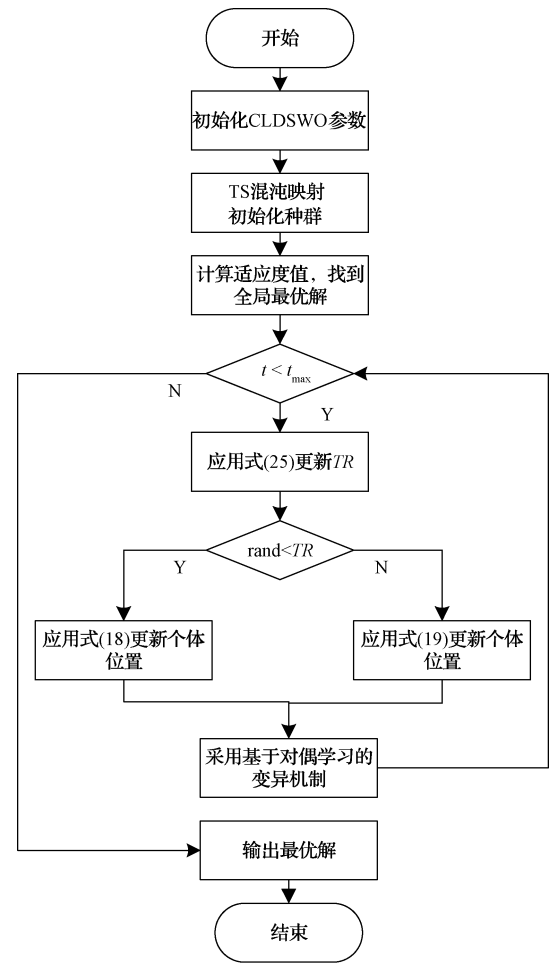


图 2　CLDSWO 算法流程

Fig. 2　Flowchart of the CLDSWO algorithm

2.5　CLDSWO 算法的时间复杂度分析

算法的复杂度是反映算法效率的关键指标之一。

CLDSWO算法的复杂度主要包括如下3个过程：混沌初始化；狩猎和筑巢以及交配行为的位置更新；基于对偶学习的变异机制。假设算法的种群大小为N，维度为D，最大迭代次数为$t_{\max}$。混沌初始化阶段的时间复杂度为$O(N)$；狩猎和筑巢行为的时间复杂度为$O(t_{\max}DN)$；交配行为的时间复杂度为$O(t_{\max}DN)$；基于对偶学习的变异机制的时间复杂度主要由交叉和变异策略的复杂度决定，其中交叉操作的时间复杂度为$O(t_{\max}NN)$，变异操作的时间复杂度为$O(t_{\max}ND)$。综上，CLDSWO算法的总体时间复杂度为$O(CLDSWO)=O(N+t_{\max}N(N+D))$。与SWO算法相比，尽管CLDSWO算法增加了时间开销，但是性能得到了优化。

3 实验仿真与结果分析

3.1 实验设置

为了验证CLDSWO算法的性能，本文采用10个基准测试函数(表1)进行了测试，其中$F1\sim F7$为单峰函数，$F8\sim F10$为多峰函数。单峰函数只有一个最优解，用于测试算法的收敛速度。多峰函数则更具挑战性，它们包含了多个局部最优解，测试算法跳出局部最优的能力。本文将最优值、均值和标准差作为算法性能的评估指标。种群规模设置为30，最大迭代次数为1 000，为了公平起见，每个函数独立运行30次，各算法的参数设置如表2所示。

表1 10个基准测试函数

Table 1 10 benchmark test functions

函数	名称	维度	区间	最优值
$F1$	Sphere	30	$[-100,100]$	0
$F2$	Schwefel 2.22	30	$[-10,10]$	0
$F3$	Schwefell 1.2	30	$[-100,100]$	0
$F4$	Schwefel 2.21	30	$[-100,100]$	0
$F5$	Rosebrock	30	$[-30,30]$	0
$F6$	Step	30	$[-100,100]$	0
$F7$	Quartic	30	$[-1.28,1.28]$	0
$F8$	Ackley	30	$[-32,32]$	0
$F9$	Penalized1	30	$[-50,50]$	0
$F10$	Penalized2	30	$[-50,50]$	0

表2 算法参数设置

Table 2 The parameter settings

算法	参数
CLDSWO	$N_{\min}=20,N_{\max}=N_{初始},N_{d,\min}=1,N_{d,\max}=N$ $\delta=0.9,\sigma=0.5$
DBO	$k=0.1,b=0.3$
HBA	$\beta=6,C=2$
MPA	$FADs=0.2$
GCHHO	$p=0.5,J\in[0,2],E_0\in[-1,1],\theta=0$
IHHO	$p=0.5,J\in[0,2],E_0\in[-1,1],\lambda=0.3,\zeta=2$

3.2 消融实验

为验证不同改进策略的有效性，将标准SWO算法、加入TS混沌映射的SWO算法(CSWO)、加入动态权衡因子的SWO算法(DSWO)、加入基于对偶学习的变异机制的SWO算法(LSWO)以及CLDSWO算法进行比较。不同改进策略在10个基准测试函数下得到的结果如表3所示。从表3可以看出，单个不同策略改进的算法都在一定程度上增强了原始SWO算法的性能，并且混合多种策略的CLDSWO算法在所有函数上的表现都是最优的。在$F1\sim F7$上，除了CLDSWO外，LSWO算法获得了排名第2的位置，DSWO算法获得了排名第3的位置。尤其对于$F1$、$F3$和$F6$，LSWO算法和CLDSWO算法都收敛到最优值0。对于较为复杂的多峰函数$F9$和$F10$，LSWO和CLDSWO算法的收敛精度一样，并且都是最佳的，表明提出的基于对偶学习的变异机制增强了算法跳出局部最优的能力。综上，不同策略组合应用的CLDSWO算法在所有基准测试函数上具有更好的适应性。

3.3 与其他智能算法的比较

为进一步验证CLDSWO的有效性，选取其他5个智能优化算法进行对比，包括3种经典的优化算法(DBO算法[5]、HBA算法[4]、MPA算法[3])以及两种改进的优化算法(GCHHO算法[6]和IHHO算法[8])。不同算法在10个基准测试函数上的测试结果如表4所示。

从表4可以看出，CLDSWO算法在大部分基准测试函数上的性能都是最优的，尤其在$F1$、$F3$和$F6$上取得了理论最优值0。从最优值指标看，CLDSWO算法在大部分单峰测试函数和所有多峰测试函数上具有最佳收敛精度。从均值结果来看，CLDSWO算法在$F1\sim F3$、$F6$、$F8\sim F10$上的收敛精度排名第1，在$F4$、$F5$和$F7$上分别排名第2和第3，仅次于IHHO算法和GCHHO算法。对于标准差数据，CLDSWO在$F1\sim F4$、$F6$和$F8\sim F10$上的数值达到最小值，表明CLDSWO算法的稳定性较好。整体而言，相比于其他算法，CLDSWO算法具有优越性和鲁棒性。

3.4 算法的收敛曲线分析

CLDSWO和其他对比算法在一些测试函数上的收敛曲线如图3所示，进一步显示了各自的特点。从图3可以看出，CLDSWO算法在所有函数中的收敛速度和收敛精度都是最高的。对于$F1$、$F2$、$F6$而言，算法的收敛曲线在迭代前期和后期的下降速度都很快，并最终收敛到理论最优值。对于$F8$，CLDSWO在前期的收敛速度最快，后期的收敛精度最优。对于$F9$和$F10$，可以观察到CLDSWO算法在收敛过程中曲线出现了多个拐点，表明算法在不断跳出局部最优并继续收敛，进一步验证了改进策略的有效性。综上，CLDSWO算法比其他优化算法具有更好的收敛性能。

表 3 不同改进策略测试结果

Table 3 The results of different improvement strategies

函数	算法	最优值	均值	标准差	函数	算法	最优值	均值	标准差
	SWO	4.28×10^{-221}	4.37×10^{-171}	**0**		SWO	3.75×10^{-12}	1.45×10^{-10}	2.84×10^{-10}
	CSWO	1.98×10^{-245}	2.44×10^{-173}	**0**		CSWO	3.18×10^{-12}	2.00×10^{-10}	3.37×10^{-10}
$F1$	DSWO	1.37×10^{-269}	9.95×10^{-229}	**0**	$F6$	DSWO	1.20×10^{-10}	1.31×10^{-8}	4.68×10^{-8}
	LSWO	**0**	**0**	**0**		LSWO	**0**	**0**	**0**
	CLDSWO	**0**	**0**	**0**		CLDSWO	**0**	**0**	**0**
	SWO	3.42×10^{-123}	1.99×10^{-83}	1.08×10^{-82}		SWO	5.69×10^{-5}	5.85×10^{-4}	4.38×10^{-4}
	CSWO	6.94×10^{-110}	1.67×10^{-84}	8.88×10^{-84}		CSWO	5.34×10^{-5}	4.89×10^{-4}	3.31×10^{-4}
$F2$	DSWO	4.06×10^{-140}	4.72×10^{-114}	2.58×10^{-113}	$F7$	DSWO	4.52×10^{-5}	2.50×10^{-4}	1.78×10^{-4}
	LSWO	6.46×10^{-296}	2.69×10^{-270}	**0**		LSWO	1.35×10^{-5}	1.56×10^{-4}	$\mathbf{1.00\times10^{-4}}$
	CLDSWO	**0**	$\mathbf{1.10\times10^{-320}}$	**0**		CLDSWO	$\mathbf{1.25\times10^{-6}}$	$\mathbf{1.31\times10^{-4}}$	1.39×10^{-4}
	SWO	5.99×10^{-188}	5.46×10^{-133}	2.99×10^{-132}		SWO	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**
	CSWO	2.18×10^{-193}	5.19×10^{-137}	2.49×10^{-136}		CSWO	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**
$F3$	DSWO	5.28×10^{-250}	1.77×10^{-193}	**0**	$F8$	DSWO	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**
	LSWO	**0**	**0**	**0**		LSWO	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**
	CLDSWO	**0**	**0**	**0**		CLDSWO	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**
	SWO	3.97×10^{-96}	4.45×10^{-65}	2.44×10^{-64}		SWO	9.79×10^{-14}	1.19×10^{-11}	4.46×10^{-11}
	CSWO	1.81×10^{-108}	2.43×10^{-72}	1.33×10^{-71}		CSWO	1.36×10^{-13}	4.57×10^{-12}	8.94×10^{-12}
$F4$	DSWO	2.74×10^{-129}	5.36×10^{-107}	2.93×10^{-106}	$F9$	DSWO	3.10×10^{-12}	1.15×10^{-10}	1.78×10^{-10}
	LSWO	4.93×10^{-207}	3.05×10^{-166}	**0**		LSWO	$\mathbf{1.57\times10^{-32}}$	$\mathbf{1.57\times10^{-32}}$	$\mathbf{5.57\times10^{-48}}$
	CLDSWO	3.66×10^{-266}	6.22×10^{-230}	**0**		CLDSWO	$\mathbf{1.57\times10^{-32}}$	$\mathbf{1.57\times10^{-32}}$	$\mathbf{5.57\times10^{-48}}$
	SWO	2.10×10^{1}	2.21×10^{1}	5.05×10^{-1}		SWO	8.18×10^{-12}	3.46×10^{-10}	6.38×10^{-10}
	CSWO	2.13×10^{1}	2.22×10^{1}	$\mathbf{4.63\times10^{-1}}$		CSWO	6.85×10^{-12}	3.66×10^{-4}	2.01×10^{-3}
$F5$	DSWO	2.10×10^{1}	2.24×10^{1}	5.80×10^{-1}	$F10$	DSWO	4.25×10^{-10}	2.56×10^{-3}	8.50×10^{-3}
	LSWO	**0**	7.23×10^{0}	1.04×10^{1}		LSWO	$\mathbf{1.35\times10^{-32}}$	$\mathbf{1.35\times10^{-32}}$	$\mathbf{5.57\times10^{-48}}$
	CLDSWO	**0**	$\mathbf{5.93\times10^{0}}$	1.00×10^{1}		CLDSWO	$\mathbf{1.35\times10^{-32}}$	$\mathbf{1.35\times10^{-32}}$	$\mathbf{5.57\times10^{-48}}$

表 4 不同对比算法测试结果

Table 4 The results of different comparative algorithms

函数	算法	最优值	均值	标准差	函数	算法	最优值	均值	标准差
	CLDSWO	**0**	**0**	**0**		CLDSWO	**0**	**0**	**0**
	DBO	4.25×10^{-314}	7.21×10^{-233}	**0**		DBO	6.59×10^{-12}	6.79×10^{-8}	2.09×10^{-7}
$F1$	HBA	**0**	**0**	**0**	$F6$	HBA	2.44×10^{-13}	1.55×10^{-9}	4.59×10^{-9}
	MPA	2.21×10^{-52}	8.11×10^{-50}	1.35×10^{-49}		MPA	5.20×10^{-10}	1.67×10^{-9}	8.97×10^{-10}
	GCHHO	4.05×10^{-205}	8.53×10^{-176}	**0**		GCHHO	8.81×10^{-10}	3.12×10^{-8}	5.56×10^{-8}
	IHHO	**0**	**0**	**0**		IHHO	9.24×10^{-32}	1.86×10^{-25}	4.05×10^{-25}
	CLDSWO	**0**	$\mathbf{2.47\times10^{-322}}$	**0**		CLDSWO	8.31×10^{-6}	1.43×10^{-4}	1.11×10^{-4}
	DBO	3.57×10^{-153}	1.07×10^{-118}	5.88×10^{-118}		DBO	1.20×10^{-4}	7.64×10^{-4}	6.53×10^{-4}
$F2$	HBA	1.27×10^{-185}	5.57×10^{-180}	**0**	$F7$	HBA	2.23×10^{-5}	1.69×10^{-4}	1.15×10^{-4}
	MPA	2.09×10^{-30}	3.88×10^{-28}	6.87×10^{-28}		MPA	9.17×10^{-5}	6.97×10^{-4}	5.17×10^{-4}
	GCHHO	2.60×10^{-105}	4.51×10^{-93}	2.30×10^{-92}		GCHHO	6.15×10^{-6}	1.32×10^{-4}	1.39×10^{-4}
	IHHO	**0**	7.36×10^{-209}	**0**		IHHO	$\mathbf{2.09\times10^{-6}}$	$\mathbf{8.24\times10^{-5}}$	$\mathbf{7.56\times10^{-5}}$
	CLDSWO	**0**	**0**	**0**		CLDSWO	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**
	DBO	3.75×10^{-264}	1.13×10^{-137}	6.16×10^{-137}		DBO	$\mathbf{8.88\times10^{-16}}$	1.13×10^{-15}	9.01×10^{-16}
$F3$	HBA	3.42×10^{-251}	2.53×10^{-230}	**0**	$F8$	HBA	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**
	MPA	1.82×10^{-17}	1.02×10^{-11}	4.28×10^{-11}		MPA	$\mathbf{8.88\times10^{-16}}$	4.09×10^{-15}	1.08×10^{-15}
	GCHHO	5.09×10^{-155}	5.80×10^{-107}	3.17×10^{-106}		GCHHO	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**
	IHHO	**0**	**0**	**0**		IHHO	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	**0**

续表

函数	算法	最优值	均值	标准差	函数	算法	最优值	均值	标准差
$F4$	CLDSWO	7.34×10^{-256}	4.47×10^{-227}	**0**	$F9$	CLDSWO	$\mathbf{1.57\times10^{-32}}$	$\mathbf{1.57\times10^{-32}}$	$\mathbf{5.57\times10^{-48}}$
	DBO	4.69×10^{-152}	4.21×10^{-97}	2.30×10^{-96}		DBO	8.36×10^{-14}	1.46×10^{-8}	7.53×10^{-8}
	HBA	1.94×10^{-146}	2.09×10^{-139}	7.86×10^{-139}		HBA	6.84×10^{-15}	3.59×10^{-10}	1.61×10^{-9}
	MPA	2.91×10^{-20}	3.08×10^{-19}	2.62×10^{-19}		MPA	7.67×10^{-11}	1.77×10^{-10}	8.91×10^{-11}
	GCHHO	3.63×10^{-100}	1.02×10^{-89}	2.41×10^{-89}		GCHHO	2.62×10^{-10}	3.08×10^{-9}	2.91×10^{-9}
	IHHO	**0**	$\mathbf{1.22\times10^{-237}}$	**0**		IHHO	2.54×10^{-32}	5.85×10^{-26}	2.24×10^{-25}
$F5$	CLDSWO	**0**	7.36×10^{0}	1.06×10^{1}	$F10$	CLDSWO	$\mathbf{1.35\times10^{-32}}$	$\mathbf{1.35\times10^{-32}}$	$\mathbf{5.57\times10^{-48}}$
	DBO	2.44×10^{1}	2.49×10^{1}	2.48×10^{-1}		DBO	7.70×10^{-5}	3.11×10^{-1}	3.29×10^{-1}
	HBA	1.81×10^{1}	1.96×10^{1}	8.03×10^{-1}		HBA	3.64×10^{-12}	8.12×10^{-2}	1.37×10^{-1}
	MPA	2.26×10^{1}	2.33×10^{1}	3.64×10^{-1}		MPA	9.75×10^{-10}	5.70×10^{-4}	2.14×10^{-3}
	GCHHO	2.00×10^{-5}	1.59×10^{-4}	1.12×10^{-4}		GCHHO	1.91×10^{-9}	5.96×10^{-8}	5.69×10^{-8}
	IHHO	**0**	$\mathbf{4.14\times10^{-23}}$	$\mathbf{9.51\times10^{-23}}$		IHHO	$\mathbf{1.35\times10^{-32}}$	3.13×10^{-24}	1.45×10^{-23}

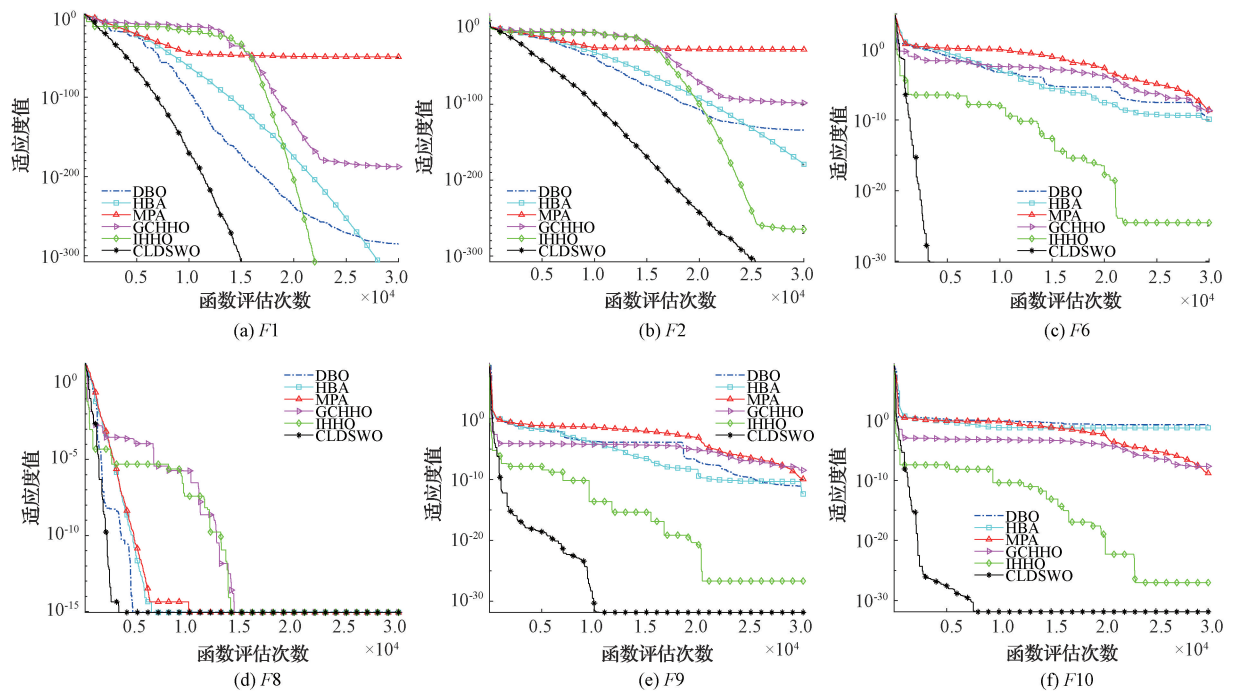


图 3　不同算法收敛曲线

Fig. 3　The convergence curves of different algorithms

3.5　Wilcoxon 秩和检验

采用 Wilcoxon 秩和检验[18]确定算法之间是否存在显著性差异。算法独立运行 30 次，数据量满足统计分析要求。表 5 为 $\alpha=5\%$ 显著性水平下的 Wilcoxon 秩和检验的 p 值，若 $p<0.05$，则拒绝原假设，表表明这两种算法有显著性差异，反之则接受原假设。“+/=/−”分别表示 CLDSWO 算法的性能优于、相当于和劣于其他算法，其中“NaN”表示这两种算法的性能相当。

从表 5 可以看出，大部分算法的秩和检验 p 值小于 0.05，这说明 CLDSWO 算法的性能与其他算法有显著差异。因此，CLDSWO 算法具有良好的寻优性能。

3.6　CEC2017 函数测试

为进一步验证 CLDSWO 的性能，使用 10 个更具挑战性的 CEC2017 测试函数（表 6）对上述算法进行测试。种群规模设置为 30，最大迭代次数为 500 次，每个算法独立运行 30 次。同时，还进行了 Wilcoxon 秩和符号检验，其中符号“gm”表示符号“+”和“−”数量之差的分数。表 7 为 CLDSWO 和其他算法的实验结果。表 7 中，CLDSWO 在 CEC01、CEC02、CEC06 和 CEC07 上排名第 1，在 CEC03、CEC04、CEC05、CEC08、CEC09 和 CEC10 上的排名仅次于 MPA 算法，排在第 2 位。MPA 算法受海洋捕食者行为启发，因其独特的觅食策略使得该算法在 CEC2017

表 5 基准测试函数下 Wilcoxon 秩和检验

Table 5 The results of Wilcoxon rank sum test

CLDSWO	DBO	HBA	MPA	GCHHO	IHHO
$F1$	1.21×10^{-12}	NaN	1.21×10^{-12}	1.21×10^{-12}	NaN
$F2$	1.95×10^{-11}	1.95×10^{-11}	1.95×10^{-11}	1.95×10^{-11}	5.14×10^{-8}
$F3$	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	NaN
$F4$	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	7.74×10^{-9}
$F5$	2.72×10^{-11}	2.21×10^{-2}	3.33×10^{-11}	2.67×10^{-2}	1.04×10^{-1}
$F6$	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
$F7$	1.10×10^{-8}	2.84×10^{-1}	9.26×10^{-9}	3.63×10^{-1}	9.47×10^{-3}
$F8$	1.61×10^{-1}	NaN	3.94×10^{-12}	NaN	NaN
$F9$	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
$F10$	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	4.57×10^{-12}
+/=/−	9/0/1	8/2/0	10/0/0	8/1/1	6/3/1

表 6 10 个 CEC2017 基准测试函数

Table 6 10 CEC2017 benchmark test functions

编号	函数名称	特征	最优值
CEC01	偏移旋转 Bent Cigar 函数	单峰函数	100
CEC02	偏移旋转 Zakharov 函数		300
CEC03	偏移旋转 Rosenbrork's 函数	基本多峰函数	400
CEC04	偏移旋转 Levy 函数		900
CEC05	混合函数 2 ($N=3$)	混合函数	1 200
CEC06	混合函数 3($N=3$)		1 300
CEC07	混合函数 6 ($N=5$)		1 800
CEC08	复合函数 2($N=3$)	复合函数	2 200
CEC09	复合函数 4($N=4$)		2 400
CEC10	复合函数 10($N=3$)		3 000

表 7 CEC2017 测试函数测试结果

Table 7 The results of CEC2017 benchmark functions

函数	指标	CLDSWO	DBO	HBA	MPA	GCHHO	IHHO
CEC01	均值	**1.00×10^{2}**	2.13×10^{6}	4.14×10^{3}	1.15×10^{2}	2.45×10^{3}	2.00×10^{10}
	标准差	**4.04×10^{-1}**	8.60×10^{6}	3.99×10^{3}	3.54×10^{1}	2.08×10^{3}	3.36×10^{9}
	p 值	3.01×10^{-11}	3.02×10^{-11}	1.09×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.01×10^{-11}
	符号秩	+	+	+	+	+	+
CEC02	均值	**3.00×10^{2}**	6.77×10^{2}	**3.00×10^{2}**	**3.00×10^{2}**	3.01×10^{2}	1.75×10^{4}
	标准差	2.96×10^{-4}	6.29×10^{2}	**1.03×10^{-5}**	1.18×10^{-4}	2.22×10^{0}	1.32×10^{3}
	p 值	3.02×10^{-11}	3.20×10^{-9}	1.49×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	符号秩	+	+	−	+	+	+
CEC03	均值	**4.01×10^{2}**	4.28×10^{2}	4.04×10^{2}	**4.01×10^{2}**	4.12×10^{2}	2.56×10^{3}
	标准差	1.17×10^{0}	4.45×10^{1}	1.21×10^{1}	**9.62×10^{-1}**	2.36×10^{1}	8.19×10^{2}
	p 值	1.17×10^{-9}	2.64×10^{-1}	7.20×10^{-5}	2.07×10^{-2}	3.02×10^{-11}	1.17×10^{-9}
	符号秩	+	−	+	+	+	+
CEC04	均值	9.01×10^{2}	9.88×10^{2}	9.17×10^{2}	**9.00×10^{2}**	9.84×10^{2}	2.20×10^{3}
	标准差	1.56×10^{0}	9.66×10^{1}	2.88×10^{1}	**8.82×10^{-2}**	1.05×10^{2}	1.37×10^{2}

续表

函数	指标	CLDSWO	DBO	HBA	MPA	GCHHO	IHHO
	p 值	7.85×10^{-11}	3.29×10^{-5}	5.89×10^{-1}	7.11×10^{-11}	2.90×10^{-11}	7.85×10^{-11}
	符号秩	+	+	−	+	+	+
CEC05	均值	4.09×10^{3}	3.26×10^{6}	1.90×10^{4}	$\mathbf{1.32\times10^{3}}$	6.27×10^{5}	1.30×10^{9}
	标准差	3.23×10^{3}	5.06×10^{6}	1.69×10^{4}	$\mathbf{7.76\times10^{1}}$	9.06×10^{5}	4.40×10^{8}
	p 值	1.33×10^{-10}	4.80×10^{-7}	8.99×10^{-11}	7.39×10^{-11}	3.02×10^{-11}	1.33×10^{-10}
	符号秩	+	+	+	+	+	+
CEC06	均值	$\mathbf{1.31\times10^{3}}$	1.27×10^{4}	1.13×10^{4}	$\mathbf{1.31\times10^{3}}$	1.34×10^{4}	1.92×10^{8}
	标准差	5.27×10^{0}	1.33×10^{4}	9.94×10^{3}	$\mathbf{3.07\times10^{0}}$	1.11×10^{4}	1.65×10^{8}
	p 值	3.02×10^{-11}	3.02×10^{-11}	6.74×10^{-6}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	符号秩	+	+	+	+	+	+
CEC07	均值	$\mathbf{1.80\times10^{3}}$	1.66×10^{4}	1.71×10^{4}	1.81×10^{3}	1.77×10^{4}	7.35×10^{8}
	标准差	$\mathbf{4.24\times10^{0}}$	1.34×10^{4}	1.40×10^{4}	5.84×10^{0}	1.23×10^{4}	5.68×10^{8}
	p 值	3.02×10^{-11}	3.02×10^{-11}	7.22×10^{-6}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
	符号秩	+	+	+	+	+	+
CEC08	均值	2.29×10^{3}	2.31×10^{3}	2.30×10^{3}	$\mathbf{2.27\times10^{3}}$	2.30×10^{3}	3.89×10^{3}
	标准差	2.79×10^{1}	$\mathbf{1.36\times10^{1}}$	1.87×10^{1}	3.80×10^{1}	2.07×10^{1}	4.14×10^{2}
	p 值	2.03×10^{-7}	1.77×10^{-3}	9.47×10^{-3}	6.05×10^{-7}	3.02×10^{-11}	2.03×10^{-7}
	符号秩	+	+	+	+	+	+
CEC09	均值	2.56×10^{3}	2.73×10^{3}	2.76×10^{3}	$\mathbf{2.50\times10^{3}}$	2.73×10^{3}	3.03×10^{3}
	标准差	1.05×10^{2}	8.90×10^{1}	2.47×10^{1}	$\mathbf{1.86\times10^{1}}$	9.46×10^{1}	9.06×10^{1}
	p 值	1.43×10^{-8}	1.29×10^{-9}	7.62×10^{-3}	1.86×10^{-9}	3.02×10^{-11}	1.43×10^{-8}
	符号秩	+	+	+	+	+	+
CEC10	均值	6.33×10^{3}	9.55×10^{5}	4.86×10^{6}	$\mathbf{4.22\times10^{3}}$	1.71×10^{5}	6.44×10^{7}
	标准差	3.10×10^{3}	1.02×10^{6}	9.28×10^{6}	$\mathbf{2.74\times10^{3}}$	2.96×10^{5}	2.88×10^{7}
	p 值	8.99×10^{-11}	2.38×10^{-8}	9.83×10^{-8}	6.70×10^{-11}	3.02×10^{-11}	8.99×10^{-11}
	符号秩	+	+	+	+	+	+
+/−/=/gm				57/3/0/54			

的比赛中取得胜利。尽管 CLDSWO 算法在均值指标上的结果比 MPA 稍显弱势，但是从 Wilcoxon 秩和符号检验结果来看，CLDSWO 算法明显优于其他算法。总体而言，CEC2017 测试结果表明 CLDSWO 是有效的，并且适用于一些工程问题。

4 工程应用实验

为进一步验证 CLDSWO 在实际工程应用问题中的性能表现，本文选取压力容器设计问题和无源时差定位优化问题进行测试。

4.1 压力容器设计问题

压力容器设计问题[19]的目标是在满足生产需要的同时使其总费用最少，其 4 个决策变量分别为内半径 $R(x_1)$，容器长度 $L(x_2)$，外壳厚度 $T_s(x_3)$ 以及封头厚度 $T_h(x_4)$，其中 T_s 和 T_h 为 0.625 的整数倍，R 和 L 为连续变量。

1)目标函数：

$$\min f(x)=0.622\,4x_1x_3x_4+1.778\,1x_2x_3^2+3.166\,1x_1^2x_4+19.84x_1^2x_3 \tag{31}$$

2)约束条件：

$$g_1(x)=-x_1+0.019\,3x_3\leqslant 0 \tag{32}$$

$$g_2(x)=-x_2+0.009\,54x_3\leqslant 0 \tag{33}$$

$$g_3(x)=-\pi x_3^2x_4-\frac{4}{3}\pi x_3^2+1\,296\,000\leqslant 0 \tag{34}$$

$$g_4(x)=x_4-240\leqslant 0 \tag{35}$$

3)边界约束：

$$0\leqslant x_1,x_2\leqslant 99 \tag{36}$$

$$10\leqslant x_3,x_4\leqslant 200 \tag{37}$$

实验种群规模设置为 30，最大迭代次数为 500 次，独立运行 30 次。对比算法为 SWO 算法[12]、DBO 算法[5]、

HBA 算法[4]和 GCHHO 算法[6]，参数设置同 3.1 节。结果如表 8 所示。从表 8 可以看出，在均值和标准差指标上，CLDSWO 算法在各对比算法中的表现最优，与经典 SWO 算法相比，均值精度提升了 77.728 5，标准差降低了 20.053 6，验证了 CLDSWO 算法解决实际问题的有效性。

表 8 各算法在压力容器问题中的结果统计

Table 8 The results of algorithms in pressure vessel problems

算法	x_1	x_2	x_3	x_4	均值	标准差
CLDSWO	0.840 8	0.416 4	43.554 4	159.399 2	5985.897 6	136.609 1
SWO	1.021 9	0.505 1	52.946 4	76.570 1	6063.626 1	159.662 7
DBO	1.258 8	0.622 2	65.225 2	10.000 0	6435.055 7	678.771 0
HBA	0.778 3	0.384 7	40.326 4	199.905 4	6013.351 5	138.094 2
GCHHO	1.000 7	0.494 6	51.849 3	84.318 3	6592.163 5	479.346 4

4.2 TDOA 优化问题

TDOA 技术利用目标信号源到达各基站的时间差来确定目标位置。基于智能优化算法求解 TDOA 定位问题已被验证是一种可靠有效的方法[20]。

1)数学模型

本文采取三维定位系统，讨论由 4 个基站、1 个目标源构成的定位系统，定位系统如图 4 所示，设主站位置为 $A(x_0,y_0,z_0)$，3 个辅站位置分别为 $B(x_1,y_1,z_1)$、$C(x_2,y_2,z_2)$、$D(x_3,y_3,z_3)$，目标源的位置为 $E(x,y,z)$，则定位方程为：

$$\begin{cases} r_0^2=(x-x_0)^2+(y-y_0)^2+(z-z_0)^2 \\ r_i^2=(x-x_i)^2+(y-y_i)^2+(z-z_i)^2 \\ r_{i0}=r_i-r_0 \\ r_{i0}=c\cdot\Delta t_i=c\cdot(t_i-t_0) \end{cases} \tag{38}$$

式中：$i=1,2,3$；r_0 表示目标到主站的距离；r_i 表示辅站 i 到目标的距离；r_{i0} 表示目标到主站的距离与目标到辅站 i 距离的差值；c 表示信号传播速度；t_0 为信号到主站的时间；t_i 为信号到达辅站 i 的时间；t_{i0} 表示信号到达主站与辅站 i 的时间差，即 TDOA 测量值。通过联立以上方程组得到最优解的过程就是 TDOA 定位求解的过程。

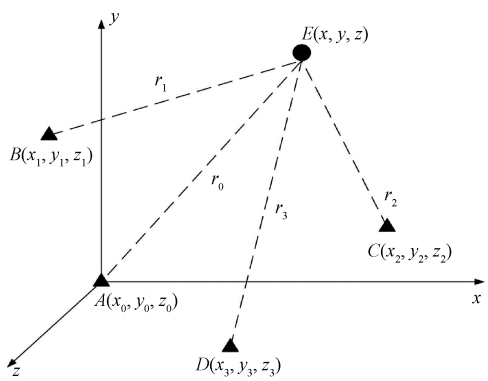


图 4 三维 TDOA 定位原理

Fig. 4 The diagram of three-dimensional TDOA

2)求解过程

利用智能优化算法求解 TDOA 问题，就是将 TDOA 定位模型中的似然函数作为目标函数，经过迭代寻优得到最优解，即最佳目标估计位置。以双曲线定位模型为例，假设有 N 个信号接收站，令目标位置发射的信号到达主站的时间真实值记为 t_0，信号到辅站 i 的时间真实值记为 $t_i(i=1,2,\cdots,N-1)$，信号传播速度为 c，可以得到：

$$\begin{cases} t_0=\dfrac{\sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}}{c} \\ t_i=\dfrac{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}}{c} \end{cases} \tag{39}$$

因此，目标位置信号到达主站与辅站 i 的 TDOA 测量值为：

$$t_{0,i}=t_0-t_i+e_{0,i} \tag{40}$$

式中：$e_{0,i}$ 为主站与第 i 个辅站之间的 TDOA 测量误差，假设其服从均值为 0，方差为 σ_i^2 的高斯分布。

将信号到达主站时间、信号到达辅站 i 的时间、信号到达主站与辅站 i 的 TDOA 测量值和信号到达主站与辅站 i 的 TDOA 测量值误差表示为 $N-1$ 维向量形式，如下：

$$\boldsymbol{t}_0=[t_0,t_0,\cdots,t_0]^{\mathrm{T}} \tag{41}$$

$$\boldsymbol{t}=[t_1,t_2,\cdots,t_{(N-1)0}]^{\mathrm{T}} \tag{42}$$

$$\Delta\boldsymbol{t}=[t_{10},t_{20},\cdots,t_{(N-1)0}]^{\mathrm{T}} \tag{43}$$

$$\boldsymbol{e}=[e_{10},e_{20},\cdots,e_{(N-1)0}]^{\mathrm{T}} \tag{44}$$

简化可得：

$$\Delta\boldsymbol{t}=\boldsymbol{t}-\boldsymbol{t}_0+\boldsymbol{e} \tag{45}$$

利用最大似然估计得出估计目标位置 $\boldsymbol{X}'=[x',y',z']^{\mathrm{T}}$，设各测量值相互独立，则似然函数为：

$$L=\prod_{i=1}^{N-1}\left[\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(t_{i0}-t_i+t_0)^2}{2\sigma^2}\right)\right]=\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{N-1}\cdot\exp\left[-\frac{(\Delta\boldsymbol{t}-\boldsymbol{t}+\boldsymbol{t}_0)^{\mathrm{T}}(\Delta\boldsymbol{t}-\boldsymbol{t}+\boldsymbol{t}_0)}{2\sigma^2}\right] \tag{46}$$

求目标位置使似然函数最大，相当于求：

$$(x',y',z')=\arg\left(\max\left[\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{N-1}\cdot\right.\right.$$

$$\exp\left[-\frac{(\Delta\boldsymbol{t}-\boldsymbol{t}+\boldsymbol{t}_0)^{\mathrm{T}}(\Delta\boldsymbol{t}-\boldsymbol{t}+\boldsymbol{t}_0)}{2\sigma^2}\right]\Bigg\} \tag{47}$$

即：

$$(x',y',z')=\arg\{\min[(\Delta\boldsymbol{t}-\boldsymbol{t}+\boldsymbol{t}_0)^{\mathrm{T}}(\Delta\boldsymbol{t}-\boldsymbol{t}+\boldsymbol{t}_0)]\} \tag{48}$$

根据式(48)可以推出粒子群算法的适应度函数为：

$$f(\boldsymbol{X})=(\Delta\boldsymbol{t}-\boldsymbol{t}+\boldsymbol{t}_0)^{\mathrm{T}}(\Delta\boldsymbol{t}-\boldsymbol{t}+\boldsymbol{t}_0) \tag{49}$$

式中：$\boldsymbol{X}$ 为粒子个体的位置矢量。

3)仿真分析

实验场景设置如下：假设在一个三维空间内，搜索范围上界 $ub=[120,120,50]^{\mathrm{T}}$，下界为 $lb=[-120,-120,0]^{\mathrm{T}}$，信号接收站采用 Y 型布阵，各基站坐标分别为 $A=$[0 km，0 km，0 km]，$B=$[7 km，−7 km，0 km]，$C=$[−7 km，−7 km，0 km]，$D=$[0 km，10 km，0 km]，种群大小为 30，最大迭代次数为 500 次。对比算法为 SWO 算法[12]、WOA 算法[1]、HBA 算法[4]以及 MPA 算法[3]，对不同位置的目标进行 30 次独立实验，定位结果如表 9 所示。

表 9 不同目标位置算法定位正确率对比

Table 9 Comparison of positioning accuracy of different algorithms (%)

目标坐标	[20 km，20 km，25 km]	[75 km，75 km，25 km]	[100 km，100 km，25 km]	[119 km，119 km，25 km]
CLDSWO 定位正确率	100	100	100	100
SWO 定位正确率	100	96.67	80	63.33
WOA 定位正确率	90	53.33	66.67	46.67
HBA 定位正确率	100	46.67	13.33	26.67
MPA 定位正确率	80	23.33	10	23.33

从表 9 可以看出，CLDSWO 算法在不同目标位置处都能达到 100%的定位准确率。随着目标距离观测站越来越来远，其余算法的定位精度有所下降，当目标坐标为[100 km，100 km，25 km]时，MPA 定位正确率下降到仅有 10%，SWO、WOA 算法正确率高于 60%，而 CLDSWO 算法一直保持 100%的定位准确率，当目标坐标接近边界，为[119 km，119 km，25 km]时，CLDSWO 算法依然保持 100%的定位正确率。综上，CLDSWO 算法定位准确率在所有算法中排名第 1，验证了其在 TDOA 定位中的有效性和稳定性。

5 种算法的近场源和远场源收敛曲线对比如图 5、6 所示。由图 5 可以看出，在近场目标定位中，CLDSWO 算法和 MPA 算法的收敛曲线表现最优，其中 CLDSWO 算法的收敛精度达到最优值。由图 6 可以看出，对于远场目标定位，WOA 算法在前期收敛速度很快，但在后期陷入了局部最优值，而 CLDSWO 算法的收敛曲线的下降速度一直很快，最后收敛到一个极小值。总体而言，CLDSWO 算法的收敛曲线表现较好且较为稳定，说明了 CLDSWO 算法在近远场定位中具有较高的定位精度和稳健性。

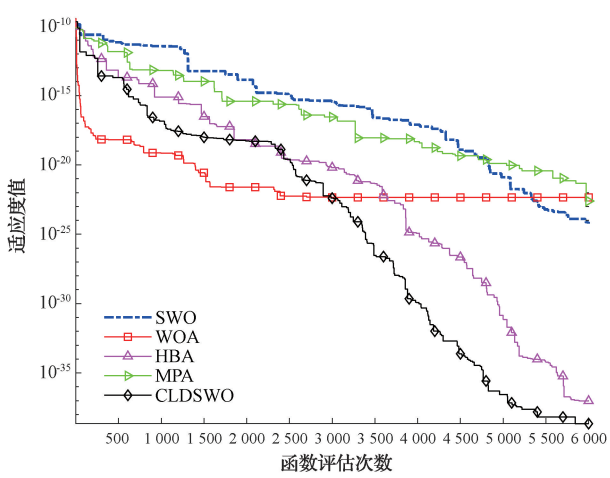


图 5 近场源定位收敛曲线

Fig. 5 Convergence curve of near-field source localization

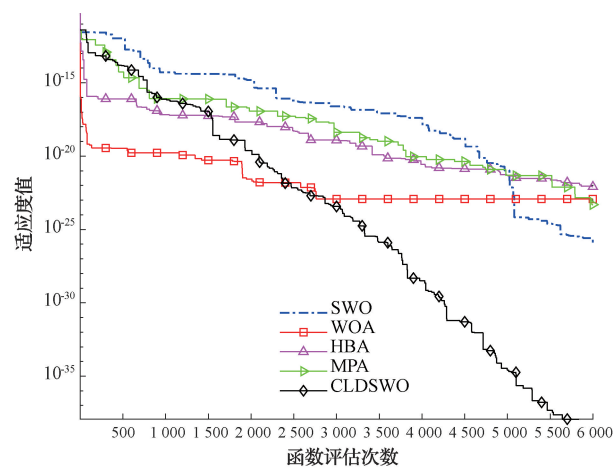


图 6 远场源定位收敛曲线

Fig. 6 Convergence curve of far-field source localization

5 结 论

本文针对经典蜘蛛蜂优化算法初始化方法不合理、勘探和开发之间的转换不平衡、易陷入局部最优等问题，提出了一种融合对偶学习的动态蜘蛛蜂优化算法。首先，引入 Tent-Sinusoidal 混沌映射初始化蜂群，然后采用动态的权衡因子实现搜索和开发之间的平衡，最后引入基于对偶

学习的变异机制，增加算法的多样性，加强算法跳出局部最优的能力。为了验证改进算法的有效性，采用10个基准测试函数和CEC2017测试函数进行了性能测试，并通过Wilcoxon秩和检验测试算法结果的显著性，实验结果表明，CLDSWO展示了其有效性和优越性。通过压力容器设计问题和无源时差定位优化问题的应用，进一步证实了CLDSWO算法在工程应用问题上的强大能力。

参 考 文 献

[1] MIRJALILI S, LEWIS A. The whale optimization algorithm [J]. Advances in Engineering Software, 2016, 95: 51-67.

[2] QU C W, HE W, PENG X N, et al. Harris Hawks optimization with information exchange [J]. Applied Mathematical Modelling, 2020, 84: 52-75.

[3] FARAMARZI A, HEIDARINEJAD M, MIRJALILI S, et al. Marine predators algorithm: A nature-inspired metaheuristic [J]. Expert Systems with Applications, 2020, 152: 113377.

[4] HASHIM F A, HOUSSEIN E H, HUSSAIN K, et al. Honey badger algorithm: New metaheuristic algorithm for solving optimization problems [J]. Mathematics and Computers in Simulation, 2022, 192: 84-110.

[5] XUE J, SHEN B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization [J]. Journal of Supercomputing, 2023, 79 (7): 7305-7336.

[6] SONG S, WANG P, HEIDARI A A, et al. Dimension decided Harris Hawks optimization with Gaussian mutation: Balance analysis and diversity patterns [J]. Knowledge-Based Systems, 2021, 215: 106425.

[7] WANG C, XU R Q, MA L, et al. An efficient salp swarm algorithm based on scale-free informed followers with self-adaption weight [J]. Applied Intelligence, 2023, 53(2): 1759-1791.

[8] 张帅，王俊杰，李爱莲，等. 集成正态云和动态扰动的哈里斯鹰优化算法 [J]. 小型微型计算机系统，2023, 44(6): 1153-1161.
ZHANG SH, WANG J J, LI AI L, et al. Harris Hawks optimization algorithm integrating normal clouds and dynamic perturbations [J]. Journal of Chinese Computer Systems, 2023, 44 (6): 1153-1161.

[9] 朱文昌，李振璧，姜媛媛. 联合VMD与ISSA-ELM的电力电子电路软故障诊断 [J]. 电子测量与仪器学报，2022, 36(5): 223-233.
ZHU W CH, LI ZH B, JIANG Y Y. Combined VMD and ISSA-ELM for soft fault diagnosis of power electronic circuits [J]. Journal of Electronic Measurement and Instrumentation, 2022, 36 (5): 223-233.

[10] 文昌俊，陈凡，陈洋洋，等. 引入改进迭代局部搜索的灰狼算法及应用 [J]. 电子测量技术，2023, 46(23): 30-42.
WEN CH J, CHEN F, CHEN Y Y, et al. Improved iterative local search grey wolf algorithm and its application [J]. Electronic Measurement Technology, 2023, 46 (23): 30-42.

[11] 陈麒羽，邵洁，王超群，等. 基于双重动态调整的改进非洲秃鹫优化算法 [J]. 国外电子测量技术，2024, 43(1): 20-29.
CHEN Q Y, SHAO J, WANG CH Q, et al. Improved African vulture optimization algorithm based on dual dynamic adjustment [J]. Foreign Electronic Measurement Technology, 2024, 43 (1): 20-29.

[12] ABDEL-BASSET M, MOHAMED R, JAMEEL M, et al. Spider wasp optimizer: A novel meta-heuristic optimization algorithm [J]. Artificial Intelligence Review, 2023, 56(10): 11675-11738.

[13] MOSTAFA R R, KHEDR A M, AL AGHBARI Z, et al. An adaptive hybrid mutated differential evolution feature selection method for low and high-dimensional medical datasets [J]. Knowledge-Based Systems, 2024, 283: 111218.

[14] FANG N, CAO Q. Leaf in wind optimization: A new metaheuristic algorithm for solving optimization problems [J]. IEEE Access, 2024, 12: 56291-56308.

[15] SABER S, SALEM S J S. High-performance technique for estimating the unknown parameters of photovoltaic cells and modules based on improved spider wasp optimizer [J]. Sustainable Machine Intelligence Journal, 2023, 5(2): 1-14.

[16] LI Y, LI C, ZHANG S, et al. A self-reproduction hyperchaotic map with compound lattice dynamics [J]. IEEE Transactions on Industrial Electronics, 2022, 69(10): 10564-10572.

[17] MIRJALILI S. Evolutionary Algorithms and Neural Networks [M]. Berlin:Springer, 2019.
[18] ABBASZADEH SHAHRI A, ASHEGHI R, KHORSAND ZAK M. A hybridized intelligence model to improve the predictability level of strength index parameters of rocks [J]. Neural Computing and Applications, 2021, 33(8): 3841-3854.
[19] CHEN H, XU Y, WANG M, et al. A balanced whale optimization algorithm for constrained engineering design problems [J]. Applied Mathematical Modelling, 2019, 71: 45-59.
[20] DONG J, LIAN Z, XU J, et al. An improved adaptive sparrow search algorithm for TDOA-based localization [J]. ISPRS International Journal of Geo-Information, 2023, 12(8): 334.

作 者 简 介

沈倩雯,硕士研究生,主要研究方向为智能优化算法、路径规划等。
E-mail: gs.qwshen22@gzu.edu.cn

张达敏(通信作者),博士,教授,硕士生导师,主要研究方向为智能计算、智能算法优化、认知无线电等。
E-mail: dmzhang@gzu.edu.cn