控制与决策
Control and Decision

基于疯狂自适应的樽海鞘群算法

张达敏， 陈忠云， 辛梓芸， 张绘娟， 闫威

引用本文:
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): 2112−2120.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2019.0012

基于疯狂自适应的樽海鞘群算法

张达敏†, 陈忠云, 辛梓芸, 张绘娟, 闫 威

(贵州大学 大数据与信息工程学院，贵阳 550025)

摘 要: 针对樽海鞘群算法求解精度不高和收敛速度慢等缺点,提出一种基于疯狂自适应的樽海鞘群算法. 引入Tent混沌序列生成初始种群,以增加初始个体的多样性;在食物源位置上引入疯狂算子,增强种群的多样性;在追随者位置更新公式中引入自适应惯性权重,使算法的全局搜索和局部搜索能力得到更好的平衡. 使用统计分析、收敛速度分析、Wilcoxon检验、经典基准函数和CEC2014函数的标准差评估改进樽海鞘群算法的效率. 结果表明,改进算法具有更好的全局搜索能力和求解鲁棒性,同时,寻优精度和收敛速度也比原来算法有所增强,尤其在求解高维和多峰测试函数上,改进算法拥有更好的性能.

关键词: 混沌映射；疯狂算子；惯性权重；樽海鞘群算法；函数优化

中图分类号: TP301 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2019.0012

引用格式: 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): 2112-2120.

Salp swarm algorithm based on craziness and adaptive

ZHANG Da-min†, CHEN Zhong-yun, XIN Zi-yun, ZHANG Hui-juan, YAN Wei

(School of Big Date and Information Engineering，Guizhou University，Guiyang 550025，China)

Abstract: In order to solve the problem of that the standard salp swarm algorithm (SSA) has slow convergence velocity and low result precision in the evolutionary process, an improved algorithm, called crazy and adaptive salp swarm algorithm (CASSA), is proposed in this paper. The Tent chaotic sequence is used to initiate the individuals' position, which can strengthen the diversity of initiate the individuals. The crazy operator is introduced at the food source position to increase the diversity of the population. The adaptive inertial weight is introduced into the follower position update formula to balance the global search and local search ability of the algorithm. The efficiency of the CASSA is evaluated by using statistical analysis, convergence rate analysis, Wilcoxon's test, standard deviations on classical benchmark functions and modern CEC 2014 functions. The results show that the CASSA has better global search ability and solving robustness, and meanwhile, the optimization accuracy and convergence speed are also more powerful than the standard algorithm. Especially, in solving the high-dimension and multimodal function optimization problem, the improved algorithm has better performance.

Keywords: chaotic map；crazy operator；inertia weight；salp swarm algorithm；function optimization

0 引 言

近年来,在工程、商业和经济学的各个领域出现许多复杂优化问题,这些问题通过传统方法在一定的时间或精度内得到最优解较为困难,运用智能算法可以处理这类优化问题. 在过去几十年中,研究人员提出各种启发式优化算法,模拟一些生物行为或物理现象,如粒子群算法(particle swarm optimization, PSO)[1]、正弦余弦算法(sine cosine algorithm, SCA)[2]、蝴蝶优化算法(butterfly optimization algorithm, BOA)[3]、灰狼优化算法(grey wolf optimization, GWO)[4]等. 这些算法已成功应用于各种科学领域,如过程控制、生物医学信号处理、图像处理以及许多其他工程设计问题.

樽海鞘群算法(salp swarm algorithm, SSA)是由Mirjalili等[5]提出的一种新型启发式智能算法. 虽然该算法求解大部分优化问题具有优越性,但与其他群智能算法一样,仍存在求解精度低和收敛速度慢等缺陷. 文献[6]提出固定惯性权重,可以加快搜索过程中的收敛速度,并应用于特征选择问题. 文献[7]将樽海鞘群算法与混沌理论结合提出混沌樽海鞘群算法,在解决特征提取问题时,能发现最优特征子集,最大程度地提高分类精度,最小化所选特征的数目. 文献[8]

收稿日期: 2019-01-03；**修回日期:** 2019-04-29.

基金项目: 贵州省自然科学基金项目(黔科合基础[2017]1047号).

责任编委: 刘德荣.

†通讯作者. E-mail: 1203813362@qq.com.

提出基于樽海鞘群算法的无缘时差定位,利用SSA解决TDOA定位结算问题,验证了算法在多站时差定位问题上的有效性和优越性. 文献[9]提出基于Tent映射的灰狼优化算法,通过混沌映射产生初始种群,增加了种群个体的多样性. 文献[10]通过引入疯狂算子来增强粒子群算法种群的多样性. 文献[11]在速度公式中引入自适应惯性权重,使蝙蝠在搜索过程中能动态地调整速度.

为解决标准樽海鞘群算法存在的求解精度不高和收敛速度慢等问题,本文提出一种疯狂自适应的樽海鞘群算法(crazy and adaptive salp swarm algorithm, CASSA). 改进算法的主要贡献有: 1) 采用Tent映射生成樽海鞘群算法的初始种群,可以使种群均匀分布在搜索空间,增强初始个体的多样性,提高算法前期收敛速度; 2) 在标准樽海鞘群算法食物源位置更新公式中加入疯狂算子,对食物源位置产生一定扰动,以此增加群体多样性; 3) 在追随者位置更新公式中引入自适应惯性权重,惯性权重较大时,有利于提升算法探索能力,惯性权重较小时,有利于增强算法开发能力,能更好地权衡探索能力和开发能力. 通过求解12个典型测试函数和CEC2014测试函数的最优解,验证了改进算法(CASSA) 的有效性和鲁棒性.

1　樽海鞘群算法

樽海鞘群算法[1]是一种全新的智能优化算法,该算法的思想出自于樽海鞘的聚集行为,即樽海鞘链. 以水中的浮游植物(海藻等)为食,通过吸入喷出海水完成在水中的移动. 在SSA中,樽海鞘链由两种类型的樽海鞘组成: 领导者和追随者. 领导者位于樽海鞘链的最前面,其他个体则为追随者的角色.

在樽海鞘群算法中,定义每个樽海鞘个体的位置矢量X用于在N维空间中搜索,其中N为决策变量的数目. 樽海鞘群算法中位置向量X将由维度为d的N个樽海鞘个体组成,因此,种群向量由$N\times d$维矩阵构成,即

$$X_i=\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix}. \tag{1}$$

在樽海鞘群算法中,食物源的位置是所有樽海鞘个体的目标位置,因此领导者的位置更新公式为

$$x_j^1=\begin{cases} F_j+c_1((\mathrm{ub}_j-\mathrm{lb}_j)c_2+\mathrm{lb}_j),\ c_3\geqslant 0.5; \\ F_j-c_1((\mathrm{ub}_j-\mathrm{lb}_j)c_2+\mathrm{lb}_j),\ c_3<0.5. \end{cases} \tag{2}$$

其中: x_j^1为第1只樽海鞘(领导者)在第j维空间的位置; F_j为食物源在第j维空间的位置; ub_j为第j维空间的上限; lb_j为第j维空间的下限; c_2和c_3为区间[0, 1]内产生的随机数. 参数c_2和c_3决定第j维空间的下一个位置应该朝正向还是负向以及移动的长度.

由式(2)可知,领导者位置更新主要受到食物源位置影响,c_1定义为

$$c_1=2\mathrm{e}^{-(4t/T_{\max})^2}. \tag{3}$$

其中: t为当前迭代次数, $T_{\max}$为最大迭代次数. 参数c_1可以使SSA的探索能力和开发能力处于较好的状态,因此系数c_1是樽海鞘群算法中最重要的参数.

为更新追随者的位置,使用如下公式(牛顿运动定理):

$$x_j^i=\frac{1}{2}at^2+v_0\Delta t. \tag{4}$$

其中: $i\geqslant 2$; x_j^i为第i只追随者在第j维空间的位置; Δt为时间; v_0为初速度; 加速度$a=(v_{\mathrm{final}}-v_0)/\Delta t$, $v_{\mathrm{final}}=(x_j^{i-1}-x_j^i)/\Delta t$, x_j^{i-1}为第$i-1$只樽海鞘在j维空间的位置. 因为时间即为迭代次数之差,所以$\Delta t=1$,且初速度$v_0=0$,式(4)可以表示为

$$x_j^i=\frac{1}{2}(x_j^i+x_j^{i-1}). \tag{5}$$

由式(2)和(5)可以模拟樽海鞘链的行为机制.

2　基于疯狂自适应樽海鞘群算法

2.1　Tent映射的种群初始化

樽海鞘群体的初始化对SSA算法的收敛速度与寻优精度至关重要. 在樽海鞘群初始时,由于没有任何先验知识可使用,基本上大部分群智能算法的初始位置均采用随机生成. 文献[12]提出初始种群均匀分布在搜索空间,对提高算法寻优有很大帮助.

混沌序列具有随机性、遍历性和规律性等特点,通过其产生的樽海鞘群体有较好的多样性. 基本思路是通过映射关系在[0, 1]区间产生混沌序列,将混沌序列转化到个体的搜索空间. 产生混沌序列的模型有许多,Tent映射比Logistic映射能够生成更好的均匀序列[13]. 本文采用Tent映射生成的混沌序列初始化樽海鞘群算法群体,其数学表达式为

$$y_{j+1}^i=\begin{cases} \mu y_j^i,\ y_j^i<0.5; \\ \mu(1-y_j^i),\ y_j^i\geqslant 0.5. \end{cases} \tag{6}$$

其中: $\mu\in(0,2]$为混沌参数, μ越大,混沌性越好,本文$\mu=2$; $i=1,2,\ldots,N$表示种群规模; $j=1,2,\ldots,d$表示混沌变量序号.

Tent混沌映射对初始值的选取非常敏感,为式(6)选取d个具有微小差异的初始值,可得到d个混沌

序列y_j^i,将其作逆映射到相应的个体搜索空间得到变量x_j^i,有

$$x_j^i = \mathrm{lb}_i + (\mathrm{ub}_i - \mathrm{lb}_i)y_j^i, \tag{7}$$

其中$[\mathrm{lb}_i, \mathrm{ub}_i]$为$x_j^i$的搜索范围.

2.2 疯狂算子

在樽海鞘群算法中,种群的食物源位置有重要作用,引导着群体向最优解移动,但若食物源位置陷入局部最优,则容易导致群体出现搜索停止,即群体内多样性缺失. 樽海鞘在移动的过程中,食物源不可能一直保持其位置不变,它们可能会突然变换位置,以此来增加种群的意外行为. 本文采用“疯狂”因素来描述这种行为,其核心思想是通过疯狂变量对其进行建模. 为了减少SSA算法出现早熟的收敛现象,本文提出在樽海鞘群算法领导者的位置更新公式中引入一个疯狂算子[11],确保樽海鞘在预先设定的疯狂概率下,对食物源位置产生一定扰动,以此维持个体的多样性. 新的领导者更新公式如下:

$$x_j^i = \begin{cases} F_j + P(c_4)\mathrm{sign}(c_4)x_{\mathrm{craziness}} + \\ c_1((\mathrm{ub}_j - \mathrm{lb}_j)c_2 + \mathrm{lb}_j),\ c_3 \geqslant 0.5; \\ F_j + P(c_4)\mathrm{sign}(c_4)x_{\mathrm{craziness}} - \\ c_1((\mathrm{ub}_j - \mathrm{lb}_j)c_2 + \mathrm{lb}_j),\ c_3 < 0.5. \end{cases} \tag{8}$$

其中:c_4为服从[0, 1]间均匀分布的随机数;$x_{\mathrm{craziness}}$通常取较小的常数;$P(c_4)$和$\mathrm{sign}(c_4)$分别定义为

$$P(c_4) = \begin{cases} 1,\ c_4 \leqslant P_{\mathrm{cr}}; \\ 0,\ \text{otherwise}. \end{cases} \tag{9}$$

$$\mathrm{sign}(c_4) = \begin{cases} -1,\ c_4 \geqslant 0.5; \\ 1,\ \text{otherwise}. \end{cases} \tag{10}$$

P_{cr}为设定的疯狂概率,$x_{\mathrm{craziness}}$为一个非常小的值(= 0.000 1). 樽海鞘在移动过程中食物源位置发生移动的可能性较小,在这种情况下,如果P_{cr}取值较小(= 0.3),则随机数c_4将有很大概率大于P_{cr},疯狂因子$P(c_4)$也将为0.

在求解测试函数优化问题中,搜索范围变化较大. 为了使樽海鞘群算法前期搜索具有更好的全局性和随机性,本文选取多个领导者进行更新,但领导者太多,算法随机性较强,会导致算法稳定性降低,因此,为了权衡算法的随机性和稳定性,选取一半的樽海鞘个体作为领导者.

2.3 自适应惯性权重

惯性权重ω体现的是追随者继承前一个樽海鞘位置的能力. 文献[14]将惯性权重ω引入PSO算法中,分析指出:当惯性权重值较大时,有助于提升探索能力;当惯性权重较小时,有助于具体开发能力. 由式(5)追随者位置更新公式可知,第i只樽海鞘位置会根据第i和第$i-1$只樽海鞘位置进行更新,对先前个体依懒性较强. 若追随者的位置是局部最优解,则会容易陷入局部最优,出现停滞. 为了更好地权衡樽海鞘群算法的探索能力与开发能力,引入线性递减的惯性权重,它决定了先前个体对当前个体的影响程度. 新的追随者位置公式为

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

$$\omega(t) = \omega_s(\omega_s - \omega_e)(T_{\max} - t)/T_{\max}. \tag{12}$$

其中:ω_s为初始惯性权重,ω_e为更迭至最大更迭代数时的惯性权重,t为当前更迭代数,$T_{\max}$为最大更迭代数. 惯性权重$\omega_s = 0.9, \omega_e = 0.4$时算法具有最佳性能,因此随着更迭的进行,惯性权重从0.9线性递减至0.4,更迭开始较大的惯性权重使算法保持较好的探索能力,而更迭后期较小的惯性权重有助于算法具有更好的开发能力.

2.4 CASSA算法步骤

算法1 CASSA算法.

begin

设置算法参数:种群大小N,最大迭代次数$T_{\max}$,惯性权重ω_s和ω_s,疯狂概率P_{cr}.

利用Tent映射初始化种群,计算每个樽海鞘个体适应度值,并选出最优个体作为食物源位置.

while $t \leqslant T_{\max}$ do

for i = 1 to N do

if $i \leqslant N$/2 do

由式(8)更新领导者位置.

else

由式(11)更新追随者位置.

end if

end for

计算种群个体的适应度值,更新食物源位置.

$t = t + 1$.

end while

end

CASSA算法首先使用第2.1节中Tent映射初始化种群,产生较均匀的初始种群;然后根据第2.2节式(8)的位置公式更新领导者位置,增强种群多样性;最后根据第2.3节式(11)的位置公式更新追随者位置,提高局部和全局能力. 本文提出的疯狂自适应樽海鞘群算法(CASSA)的步骤如算法1所示.

3　仿真实验与结果分析

为了验证本文提出的CASSA在求解优化问题时的有效性和鲁棒性,将CASSA算法与加入疯狂算子的樽海鞘群算法(记为CSSA)、加入自适应权重的樽海鞘算法(记为ASSA)、SSA、BOA[3]、GWO[4]、SCA[2]和PSO[1]在12个典型的标准测试函数[15]最优值求解上独立进行50次对比实验.

实验1　采用如表1所示12个复杂函数作为适应度函数. 选取的测试函数中包含单峰(unimodal)、多峰(multimodal)、 可分(separable)和不可分(non-separable)等不同特征的测试函数. 单峰函数为在定义上下限内只有一个严格上的极大值(或极小值),通常用来检测算法的收敛速度. 多峰函数为含有多个局部最优解或全局最优解的函数,经常用于检测算法的探索能力和开发能力. 另外,测试函数的求解维度也是一个重要因素,算法对低维度求解质量好时对高维度不一定也好,表1中测试函数维度为2～200维,这样可以更加全面地验证算法性能.

表1　基准函数

函数	维度	特征	定义域	最佳值
f_1 Schwefel 1.2	10	UN	$[-100, 100]$	0
f_2 Step	10	US	$[-100, 100]$	0
f_3 Sphere	60	US	$[-100, 100]$	0
f_4 Quartic	60	US	$[-1.28, 1.28]$	0
f_5 Schwefel 2.22	120	UN	$[-10, 10]$	0
f_6 Schwefel 2.21	200	US	$[-100, 100]$	0
f_7 Schaffer	2	MN	$[-100, 100]$	0
f_8 Rastrigin	10	MS	$[-5.12, 5.12]$	0
f_9 Kowalik	4	MN	$[-5, 5]$	3.075e-4
f_{10} Ackley	60	MN	$[-32, 32]$	0
f_{11} Griewank	120	MN	$[-600, 600]$	0
f_{12} Penalized	200	MN	$[-50, 50]$	0

实验环境: Windows　7 系统, 8 G 内存, CPU2.5 GHz, 算法基于Matlab14b用M语言编写. 实验最大迭代次数为1 000, 种群个数为40, 各算法其余的参数设置如表2所示.

通过50次独立实验后从每种算法获得结果的最佳值、平均值、标准差、成功率(Successful, SR%)和平均耗时等多个实验对比数据如表3所示. 其中求解成功率为计算成功次数除以本次实验的求解次数. 判断一次求解是否成功按照下式进行:

$$\begin{cases} |F_A - F_T|/F_T < 1\mathrm{e}^{-5}\,, F_T \neq 0; \\ |F_A - F_T| < 1\mathrm{e}^{-5},\ F_T = 0. \end{cases} \tag{13}$$

表2　算法参数设置

算法	主要参数
CASSA	$\omega_s = 0.9, \omega_e = 0.4, P_{\mathrm{cr}} = 0.3, x_{\mathrm{craziness}} = 0.000\,1$
CSSA	$P_{\mathrm{cr}} = 0.3, x_{\mathrm{craziness}} = 0.000\,1$
ASSA	$\omega_s = 0.9, \omega_e = 0.4$
BOA	$p = 0.8, c = 0.01, a = 0.1$
GWO	$a_{\max} = 2, a_{\min} = 0$
SCA	$a = 2$
PSO	$c_1 = c_2 = 1.5$

其中: F_A 为每次实际求解最佳值, F_T 为测试函数理论最佳值.

表3中的最优值、平均值都可以反映算法的收敛精度和寻优能力. 对于6个单峰函数(f_1 ～ f_6), CASSA在求解精度上最高达到1e-96. 同时,随着搜索空间维度的增加,寻优收敛精度6种算法有所下降,因为伴随着求解维度的增加,算法求解难度也呈指数级别递增,所以算法的收敛精度有所降低属于正常现象. 另外,SSA、SCA、PSO算法当维度增加到120维(Schwefel 2.22)和200维(Schwefel 2.21)时,求解精度较差,并与理论最优值存在1e+01级的误差. 对于120维的Schwefel 2.22函数,BOA甚至与理想值存在1e+51级的误差. 除了 f_2 中CSSA最优值较差,其余CSSA和ASSA的寻优精度较标准SSA都要好,这表明引入不同策略可增强算法性能. 而CASSA相对于其他几种算法寻优精度要高很多,且与其他算法精度最大能达到1e-98级的差距. 对于6个多峰函数($f_7 \sim f_{12}$),算法求解精度相对于单峰函数要低一些. 同样,随着维度增加算法求解精度也有所降低. 当维度增加到200维(Penalized)时,SSA算法与理论最优值存在1e+01级的误差. 在求解多峰函数Schaffer(2维)和Rastrigin(10维)时,CASSA、 CSSA、ASSA、 BOA、 GWO、 SCA达到理论最优值0. 在求解多峰函数Griewank(120维)时,CASSA、CSSA、ASSA、GWO算法都达到理论最优值0,求解其他函数时,CASSA算法比其他算法的精度都高. 另外,CSSA和ASSA比标准SSA寻优结果都要好,表明加入不同算子对算法性能有所提升. 可见,CASSA算法在求解单峰、多峰、可分、不可分以及高维的基准函数时都有明显的优势.

表3中标准差和成功率可以反映算法的稳定性和跳出局部最优的能力. 除了 f_2 中CASSA标准差较差,CASSA算法独立50次都很接近理论最优值,标准差也较小. 这表明CASSA的寻优求解有着一定的稳定性. 另外,CASSA算法标准差始终都要比另外几种

表3 基准函数优化结果对比

	算法	最佳值	平均值	标准差	SR/%	耗时/s		算法	最佳值	平均值	标准差	SR/%	耗时/s
f_1	CASSA	**2.29e-96**	**2.59e-95**	**2.45e-95**	100	1.083 1	f_7	CASSA	**0.00e+00**	**0.00e+00**	**0.00e+00**	100	0.741 0
	CSSA	5.25e-17	3.72e-16	3.39e-16	100	1.062 8		CSSA	**0.00e+00**	2.22e-18	1.10e-17	100	0.729 7
	ASSA	1.48e-92	2.02e-91	2.46e-91	100	1.071 0		ASSA	**0.00e+00**	**0.00e+00**	**0.00e+00**	100	0.745 0
	SSA	3.70e-10	1.52e-09	8.43e-10	100	1.061 7		SSA	1.11e-15	2.33e-03	4.19e-03	76	0.727 0
	BOA	1.20e-14	1.51e-14	1.39e-15	100	1.885 2		BOA	**0.00e+00**	8.27e-03	3.65e-03	16	1.233 7
	GWO	7.15e-71	2.93e-61	1.05e-60	100	1.885 2		GWO	**0.00e+00**	3.50e-03	4.71e-03	64	1.233 7
	SCA	2.00e-19	2.03e-10	8.81e-10	100	**0.916 3**		SCA	**0.00e+00**	2.89e-17	2.04e-16	100	**0.527 7**
	PSO	2.80e-4	1.92e-03	1.24e-03	0	0.960 2		PSO	4.28e-09	2.92e-03	4.50e-03	70	0.671 7
f_2	CASSA	**2.59e-10**	**5.74e-10**	1.96e-10	100	0.6463	f_8	CASSA	**0.00e+00**	**0.00e+00**	**0.00e+00**	100	0.611 3
	CSSA	5.70e-02	1.36e-01	5.15e-02	0	0.629 8		CSSA	**0.00e+00**	**0.00e+00**	**0.00e+00**	100	0.602 5
	ASSA	5.17e-10	8.94e-10	**1.91e-10**	100	0.638 0		ASSA	**0.00e+00**	**0.00e+00**	**0.00e+00**	100	0.608 0
	SSA	3.94e-10	7.88e-10	2.42e-10	100	0.626 7		SSA	3.98e+00	1.77e+01	7.90e+00	0	0.600 5
	BOA	4.82e-01	1.03e+00	3.07e-01	0	0.966 2		BOA	**0.00e+00**	2.28e+01	1.93e+01	30	1.481 1
	GWO	3.77e-07	6.78e-07	1.64e-07	100	0.966 2		GWO	**0.00e+00**	8.40e-02	5.94e-01	98	1.481 1
	SCA	1.03e-01	3.44e-01	1.69e-01	0	**0.432 2**		SCA	**0.00e+00**	7.79e-01	5.51e+00	94	0.463 3
	PSO	2.02e-05	1.41e-04	9.22e-05	0	0.515 9		PSO	2.99e+00	9.00e+00	4.79e+00	0	**0.443 7**
f_3	CASSA	**2.07e-95**	**2.92e-95**	**5.25e-96**	100	0.822 1	f_9	CASSA	**3.07e-04**	**3.10e-04**	**7.41e-06**	0	0.713 9
	CSSA	6.57e-16	9.97e-16	2.00e-16	100	0.770 4		CSSA	3.10e-04	6.78e-04	6.38e-04	0	0.704 6
	ASSA	1.40e-91	2.20e-91	3.79e-92	100	0.807 0		ASSA	**3.07e-04**	5.75e-04	3.68e-04	0	0.711 7
	SSA	8.62e-08	1.44e-07	3.78e-08	100	0.766 8		SSA	3.14e-04	8.33e-04	2.93e-04	0	0.705 4
	BOA	1.59e-14	1.78e-14	8.32e-16	100	0.732 0		BOA	3.15e-04	3.48e-04	2.91e-05	0	1.223 5
	GWO	4.49e-45	3.21e-43	5.75e-43	100	0.732 0		GWO	**3.07e-04**	8.39e-03	9.88e-03	0	1.223 5
	SCA	2.69e+00	3.12e+02	5.48e+02	0	0.705 0		SCA	3.20e-04	9.32e-04	3.89e-04	0	**0.525 9**
	PSO	1.87e-01	2.98e-01	5.82e-02	0	**0.446 9**		PSO	3.15e-04	6.71e-04	3.05e-04	0	0.613 2
f_4	CASSA	**3.71e-08**	**1.24e-05**	**1.04e-05**	50	1.354 5	f_{10}	CASSA	**8.88e-16**	**8.88e-16**	**0.00e+00**	100	1.069 6
	CSSA	5.51e-06	4.88e-05	4.59e-05	4	1.302 2		CSSA	4.53e-09	5.34e-09	4.38e-10	100	1.024 3
	ASSA	3.92e-06	5.02e-05	3.83e-05	4	1.332 6		ASSA	**8.88e-16**	**8.88e-16**	**0.00e+00**	100	1.051 6
	SSA	1.73e-01	3.26e-01	1.01e-01	0	1.297 1		SSA	2.27e+00	3.56e+00	8.92e-01	0	1.037 9
	BOA	1.59e-04	6.33e-04	1.94e-04	0	1.664 9		BOA	1.02e-11	1.16e-11	6.19e-13	100	1.216 2
	GWO	2.33e-04	1.30e-03	5.71e-04	0	1.664 9		GWO	2.93e-14	3.53e-14	4.14e-15	100	1.216 2
	SCA	9.56e-02	2.18e+00	2.68e+00	0	1.194 8		SCA	1.29e-01	1.72e+01	6.85e+00	0	1.005 4
	PSO	3.17e-01	7.99e-01	8.91e-01	0	**0.899 8**		PSO	4.60e+00	6.45e+00	7.10e-01	0	**0.734 8**
f_5	CASSA	**1.07e-47**	**1.21e-47**	**8.03e-49**	100	1.294 3	f_{11}	CASSA	**0.00e+00**	**0.00e+00**	**0.00e+00**	100	1.496 0
	CSSA	3.28e-08	4.00e-08	3.20e-09	100	1.192 3		CSSA	**0.00e+00**	9.33e-16	4.50e-16	100	1.409 9
	ASSA	5.07e-46	5.85e-46	3.49e-47	100	1.252 6		ASSA	**0.00e+00**	**0.00e+00**	**0.00e+00**	100	1.458 6
	SSA	1.28e+01	2.59e+01	6.08e+00	0	1.181 9		SSA	5.06e-01	9.68e-01	1.51e-01	0	1.434 1
	BOA	1.57e+51	1.59e+60	8.31e+60	0	0.993 9		BOA	9.99e-16	1.45e-14	5.92e-15	100	1.531 6
	GWO	3.16e-18	8.20e-18	3.60e-18	100	0.993 9		GWO	**0.00e+00**	1.40e-03	4.97e-03	92	1.531 6
	SCA	1.50e-02	3.67e+00	4.32e+00	0	1.211 0		SCA	1.30e+00	8.21e+01	6.00e+01	0	1.497 6
	PSO	2.40e+01	3.39e+01	5.94e+00	0	**0.551 7**		PSO	1.02e+02	1.30e+02	1.34e+01	0	**0.997 0**
f_6	CASSA	**1.71e-48**	**2.10e-48**	**2.66e-49**	100	1.930 7	f_{12}	CASSA	**3.50e-02**	**5.75e-02**	**1.39e-02**	0	5.548 6
	CSSA	9.42e-09	1.21e-08	1.19e-09	100	1.781 3		CSSA	6.76e-01	7.74e-01	3.97e-02	0	5.421 3
	ASSA	1.35e-46	1.79e-46	1.94e-47	100	1.873 3		ASSA	7.37e-02	1.39e-01	3.13e-02	0	5.510 6
	SSA	2.23e+01	2.92e+01	2.65e+00	0	1.768 4		SSA	1.46e+01	3.00e+01	7.68e+00	0	5.422 9
	BOA	1.08e-11	1.21e-11	5.85e-13	100	1.171 4		BOA	9.92e-01	1.10e+00	4.26e-02	0	8.844 1
	GWO	1.09e+00	5.26e+00	3.37e+00	0	1.171 4		GWO	3.03e-01	4.30e-01	5.11e-02	0	8.844 1
	SCA	9.04e+01	9.50e+01	1.29e+00	0	1.916 7		SCA	4.05e+08	9.02e+08	2.56e+08	0	5.466 7
	PSO	9.08e+00	1.12e+01	9.77e-01	0	**0.827 9**		PSO	3.18e+00	4.62e+00	1.01e+00	0	**4.419 3**

算法优秀,进一步验证了CASSA的有效性. 12个基准函数中有单峰、多峰、低维和高维,除了f_4、f_9、f_{12}函数,CASSA在成功率基本上都为100%,而标准SSA除了在f_1、f_2、f_3函数的成功率为100%外,其余基准函数成功率几乎为0. 随着搜索维度的增加,标准SSA在寻优求解能力上表现出很大不足,特别是在求解多维函数时,最优值和成功率都很差,表明标准SSA跳出局部最优的能力较弱. 而在CASSA和ASSA中都引入了惯性权重,这对算法后期跳出局部搜索具有很大作用.

从平均耗时看,由表3可知,SCA和PSO平均耗时最短,改进的CASSA、CSSA、ASSA这3种算法相对于标准SSA的平均耗时都要大. 出现这种情况也很合理,这是因为算法中引入了更多的算子,使得算法能够搜索到更多的解,导致运行时间变长. 总体来看,CASSA平均耗时比另外两种算法增加的并不是很大,在允许范围内.

基于50次独立运行算法的平均值和标准差不会比较每次运行结果. Derrac等[16]提出,对于改进进化算法性能的评估,应该进行统计检验,换言之,仅基于平均值和标准偏差值[17]比较算法还不够,需要进行统计检验以验证所提出的改进算法比其他现有算法具有显著的改进优势. 为了判断CASSA的每次结果是否统计上显著地与其他算法的最佳结果不同,采用Wilcoxon统计检验[18]在5%的显著性水平下进行. 表4给出了所有基准函数的CASSA与其他算法的Wilcoxon秩和检验中计算的p值. 例如,如果最佳算法是CASSA,则在CASSA vs. ASSA、CASSA vs. SSA等之间进行成对比较. 由于最佳算法无法与自身进行比较,针对每个函数中的最佳算法标记为N/A,表示“不适用”. 这意味着相应的算法可以在秩和检验中没有统计数据与自身进行比较,符号“+”“−”和“=”分别表示CASSA的性能优于、劣于和相当于对比算法. 根据文献[16],$p < 0.05$可以被认为是拒绝零假设的有力验证.

由表4可见,CASSA的p值基本小于0.05,这表明该算法的优越性在统计上是显著的,即认为CESSA算法比其他算法具有更高的收敛精度. p值大于0.05的用粗体标出,N/A是因为算法CASSA与对比算法中最佳值都为0.

所有算法的定量分析是基于12个基准函数的平均绝对误差(mean absolute error, MAE). 文献[19]提出对优化算法进行排序,MAE是一种有效且可行的性能指标. 表5给出了这些基准函数的MAE排序,其计算公式如下:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{N_f} |m_i - o_i|}{N_f}. \tag{14}$$

其中: m_i为算法产生的最优结果的平均值, o_i为相应基准函数的理论最优值, N_f为基准函数个数.

表4　基准函数Wilcoxon秩和检验的p值

函数	CASSA vs. CSSA	CASSA vs. ASSA	CASSA vs. SSA	CASSA vs. BOA	CASSA vs. GWO	CASSA vs. SCA	CASSA vs. PSO
	p-value win	p-value win	p-value win	p-value win	p-value win	p-value win	p-value win
f_1	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +
f_2	7.01e-18 +	5.13e-11 +	2.01e-05 +	7.01e-18 +	7.01e-18 +	7.01e-18 +	7.01e-18 +
f_3	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +
f_4	1.41e-10 +	8.95e-11 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +
f_5	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +
f_6	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +
f_7	**1.59e-01**−	N/A =	3.30e-20 +	1.26e-19 +	3.72e-06 +	**3.27e-01**−	3.31e-20 +
f_8	N/A =	N/A =	3.31e-20 +	7.37e-16 +	**1.59e-01**−	6.54e-03 +	3.31e-20 +
f_9	1.27e-16 +	2.81e-12 +	9.54e-18 +	4.01e-16 +	8.98e-08 +	8.46e-18 +	1.08e-17 +
f_{10}	3.31e-20 +	N/A =	3.31e-20 +	3.31e-20 +	1.53e-20 +	3.31e-20 +	3.31e-20 +
f_{11}	1.21e-19 +	N/A =	3.31e-20 +	3.30e-20 +	4.33e-02 +	3.31e-20 +	3.31e-20 +
f_{12}	7.07e-18 +	1.08e-17 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +	7.07e-18 +
+/=/−	10/1/1	8/4/0	12/0/0	12/0/0	11/0/1	11/0/1	12/0/0

表5 MAE算法排名

算法	MAE	rank
CASSA	3.491 554e+02	1
ASSA	3.491 586e+02	2
CSSA	3.492 135e+02	3
GWO	3.492 687e+02	4
SSA	3.538 743e+02	5
PSO	3.613 199e+02	6
SCA	3.370 946e+07	7
BOA	1.309 594e+50	8

由表5可见,CASSA排名为1,与另外7种算法相比,CASSA提供了最小的MAE,进一步表明CASSA的有效性.

图1给出了12个基准函数的平均收敛曲线. 由于CASSA收敛精度较高,为了便于观察收敛情况,对寻优适应度值(纵坐标)取10为底的对数. 由图1(a)～图1(f)可见,在迭代前期,CASSA、 CSSA和ASSA三种算法有一小段时间收敛曲线下降很快,表明引入Tent映射初始化种群,增加种群多样性,使得算法一开始收敛速度就较快. 随着更迭次数的增加持续寻优,未出现停止状况,且寻优精度较高. 除了GWO,其余算法曲线下降比较平缓,在迭代后期出现不同程度的停滞,且寻优精度较低.

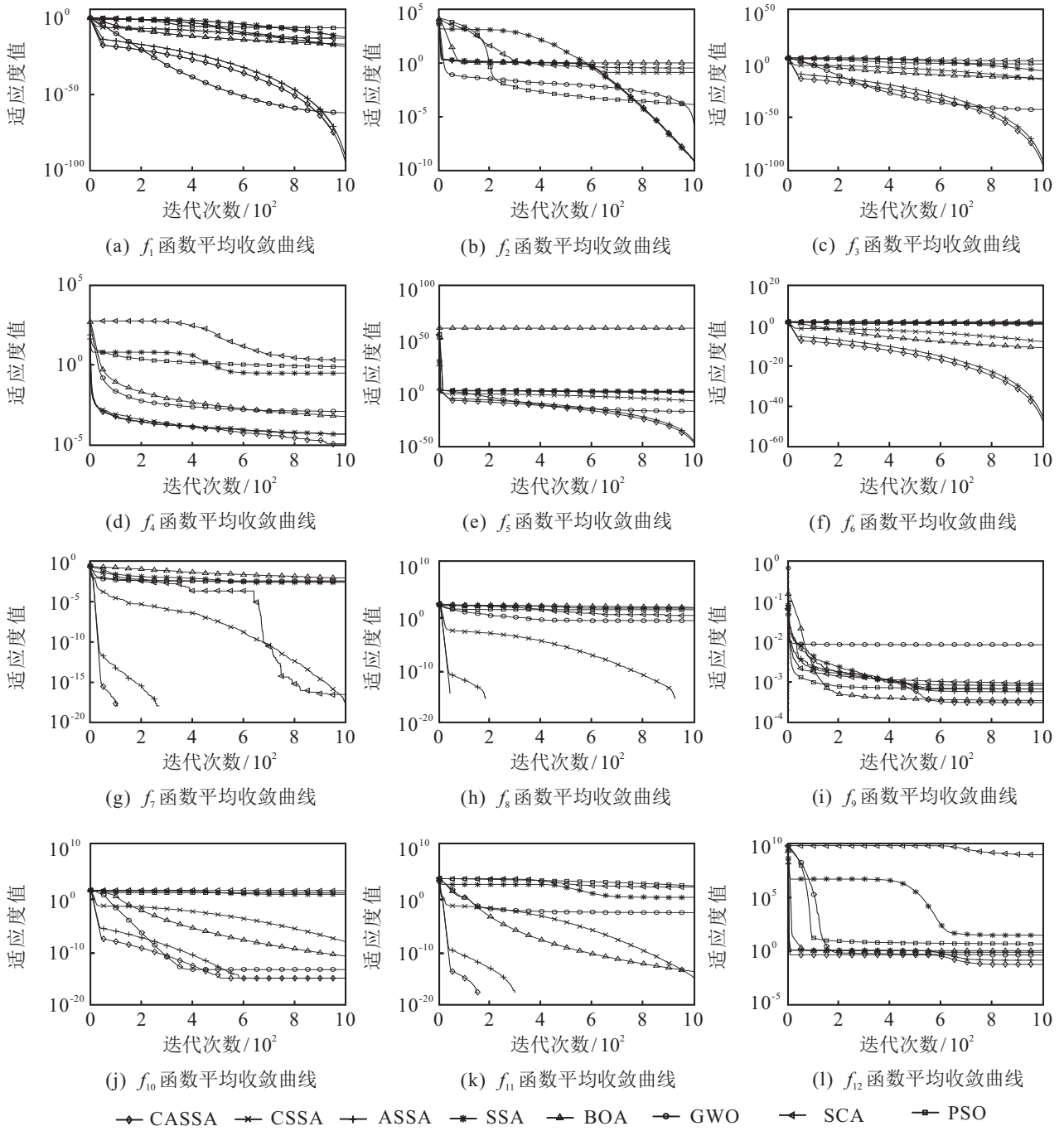

图1 基准函数平均收敛曲线

图1(g)～图1(l)为多峰高维函数的平均收敛曲线. 由图1(g)～图1(l)可看出,CASSA、CSSA和ASSA三个引入Tent初始化种群算法的收敛性在 $f_7 \sim f_{12}$ 函数上迭代前期收敛速度很快,这也进一步表明了使用Tent映射初始化种群的作用. 在迭代后期CASSA收敛速度比其余算法快很多,而SSA算法收敛速度慢,出现不同程度的停滞,且寻优精度较低. 其余大部分算法收敛曲线整体比较平缓,随着更迭次数的增加持续寻优,且出现停止状况. CASSA在函数 f_7、f_8、f_{11} 上寻优精度达到理论最优值0. 由图1(l)可见,CASSA在前期收敛很快陷入局部最优,后期又跳出局部最优,进行继续寻优,最终CASSA的寻优精度较其他算法表现较好.

无论单峰、多峰,还是低维和高维,对于每个函数,CASSA比另外7种算法的收敛速度和寻优精度都要好. 由表3可见,对于函数 f_7、f_8、f_{11},CASSA的最佳值为0. 所以在图1(g)、图1(h)和图1(k)中,CASSA的曲线后面部分没有显示.

实验2 为更好地评估CASSA的有效性和稳定性,本文在CEC2014基准函数中选取部分单峰、多峰、混合(Hybrid)和复合(Composition)类型的函数进行优化求解,如表6所示. 在该实验中,种群规模为40,最大迭代次数为1 000,维度为30.

表6　CEC2014基准函数

函数	维度	特征	定义域	最佳值
CEC01	30	UN	[−100, 100]	100
CEC04	30	MN	[−100, 100]	400
CEC18	30	HF	[−100, 100]	1 800
CEC23	30	CF	[−100, 100]	2 300
CEC24	30	CF	[−100, 100]	2 400
CEC25	30	CF	[−100, 100]	2 500

表7记录了CEC 2014中部分函数独立运行30次后每种算法的平均值和标准差结果. CEC2014函数具有复杂的特征,因此所有算法都较难找到函数最优值. 根据表7结果显示,CASSA在6个基准函数上都求得比其他5个算法更优的结果,验证了CASSA具有较好的有效性和鲁棒性.

表7　CEC2014优化结果对比

函数	指标	CASSA	SSA	BOA	GWO	SCA	PSO
CEC01	Mean	4.718 56e+06	1.966 83e+07	1.252 80e+09	7.385 69e+07	3.738 81e+08	7.264 55e+06
	Std	2.405 21e+06	7.165 94e+06	3.577 44e+08	3.919 47e+07	1.130 34e+08	5.269 61e+06
CEC04	Mean	4.662 39e+02	5.239 20e+02	1.532 28e+04	6.746 43e+02	2.156 51e+03	5.053 68e+02
	Std	1.018 88e+01	2.990 43e+01	2.412 74e+03	9.962 94e+01	5.195 47e+02	2.736 32e+01
CEC18	Mean	2.519 02e+03	9.002 62e+03	2.988 58e+09	1.098 42e+07	2.680 87e+08	5.168 04e+03
	Std	3.472 18e+02	7.696 68e+03	1.253 44e+09	2.354 02e+07	1.498 48e+08	2.395 48e+03
CEC23	Mean	2.500 00e+03	2.629 22e+03	2.500 00e+03	2.636 81e+03	2.695 12e+03	2.615 51e+03
	Std	0.000 00E+00	6.733 15e+00	0.000 00e+00	9.158 77e+00	1.891 03e+01	1.454 46e-01
CEC24	Mean	2.600 00e+03	2.638 79e+03	2.600 00e+03	2.600 02e+03	2.604 17e+03	2.631 37e+03
	Std	0.000 00e+00	6.764 11e+00	0.000 00e+00	8.049 48e-03	4.890 58e+00	8.356 36e+00
CEC25	Mean	2.700 00e+03	2.714 95e+03	2.700 00e+03	2.711 41e+03	2.739 32e+03	2.721 22e+03
	Std	0.000 00e+00	2.984 69e+00	0.000 00e+00	5.913 96e+00	9.771 87e+00	7.240 20e+00

综上可知,疯狂自适应樽海鞘群算法对于多种基准函数都有很好的寻优结果,特别是对于高维、多峰的函数,具有较好的稳定性和寻优能力.

4　结　论

本文在标准樽海鞘群算法的基础上,引入Tent映射,使初始化种群均匀分布,疯狂算子对食物源进行扰动,自适应的惯性权重平衡探索能力和开发能力,提出了一种改进的疯狂自适应樽海鞘群算法,并将樽海鞘群算法应用于经典和CEC2014基准函数的寻优问题. 不仅使用最优值、标准差等指标对算法进行检验,还提出使用Wilcoxon秩和检验对算法显著性水平进行验证. 研究表明:疯狂自适应樽海鞘群算法可以获得更好的全局搜索和局部搜索能力,且收敛到质量更好的最优解,算法的有效性和鲁棒性也得到验证. 在后续的研究中,考虑将改进的樽海鞘群算法应用到工程实践问题中,以进一步验证算法的性能.

参考文献(References)

[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proceedings of ICNN'95-IEEE International Conference on Neural Networks. Perth: IEEE, 1995: 1942-1948.

[2] Mirjalili S. SCA: A sine cosine algorithm for solving optimization problems[J]. Knowledge Based Systems, 2016, 96: 120-133.

[3] Arora S, Singh S. Butterfly optimization algorithm: A novel approach for global optimization[J]. Soft Computing, 2019, 23(3): 715-734.

[4] Mirjalili S, Mirjalili S M, Lewis A, et al. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69(3): 46-61.

[5] 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(6): 163-191.

[6] Hegazy A E, Makhlouf M A, Eltawel G S. Improved salp swarm algorithm for feature selection[J]. Journal of King Saud University — Computer and Information Sciences, 2018, 6(3): 1-10.

[7] Sayed G I, Khoriba G, Haggag M H. A novel chaotic salp swarm algorithm for global optimization and feature selection[J]. Applied Intelligence, 2018, 48(3): 1-20.

[8] 陈涛, 王梦馨, 黄湘松. 基于樽海鞘群算法的无源时差定位[J]. 电子与信息学报, 2018, 40(7): 1591-1597.
(Chen T, Wang M X, Huang X S. Time difference of arrival passive location based on salp swarm algorithm[J]. Journal of Electronics & Information Technology, 2018, 40(7): 1591-1597.)

[9] 滕志军, 吕金玲, 郭力文, 等. 一种基于Tent映射的混合灰狼优化的改进算法[J]. 哈尔滨工业大学学报, 2018, 50(11): 46-55.
(Teng Z J, Lv J L, Guo L W, et al. An improved hybrid grey wolf optimization algorithm based on tent mapping[J]. Journal of Harbin Institute of Technology, 2018, 50(11): 46-55.)

[10] Mandal D, Ghoshal S P, Bhattacharjee A K. Radiation pattern optimization for concentric circular antenna array with central element feeding using crazinessbased particle swarm optimization[J]. International Journal of RF and Micro-wave Computer Aided Engineering, 2010, 20(5): 577-586.

[11] Wang X W, Wang W, Wang Y. An adaptive bat algorithm[C]. International Conference on Intelligent Computing. Nanning: Springer, 2013: 216-223.

[12] Haupt R, Haupt S. Practical genetic algorithm[M]. New York: John Wiley and Sons, 2004: 38-39.

[13] 单梁, 强浩, 李军, 等. 基于Tent映射的混沌优化算法[J]. 控制与决策, 2005, 20(2): 179-182.
(Shan L, Qiang H, Li J, et al. Chaotic optimization algorithm based on Tent map[J]. Control and Decision, 2005, 20(2): 179-182.)

[14] Zhou Z, Shi Y. Inertia weight adaption in particle swarm optimization algorithm[C]. International Conference on Advances in Swarm Intelligence. Chongqing: Springer, 2011: 71-79.

[15] Mirjalili S, Mirjalili S M, Yang X S. Binary bat algorithm[J]. Neural Computing and Applications, 2014, 25(3): 663-681.

[16] Derrac J, Garcia 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 & Evolutionary Computation, 2011, 1(1): 3-18.

[17] García S, Molina D, Lozano M, et al. A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: A case study on the CEC'2005 special session on real parameter optimization[J]. Journal of Heuristics, 2009, 15(6): 617-644.

[18] Wilcoxon F. Individual comparisons by ranking methods[J]. Biometrics Bulletin, 1945, 1(6): 80-83.

[19] Emad N. A modified flower pollination algorithm for global optimization[J]. Expert Systems with Applications, 2016, 57(9): 192-203.

作者简介

张达敏(1967–) 男, 教授, 博士, 从事认知无线电、智能算法优化、图像处理等研究, E-mail: 1203813362@qq.com;

陈忠云(1989–), 男, 硕士生, 从事认知无线电、智能算法优化的研究, E-mail: chenzhongyun315@hotmail.com;

辛梓芸(1994–), 女, 硕士生, 从事认知无线电、智能算法优化的研究, E-mail: 623969769@qq.com;

张绘娟(1995–), 女, 硕士生, 从事认知无线网络选择的研究, E-mail: 1448547735@qq.com;

闫威(1993–), 男, 硕士生, 从事认知无线网络选择的研究, E-mail: 349552812@qq.com.

(责任编辑: 郑晓蕾)