控制与决策
Control and Decision

面向全局搜索的自适应领导者樽海鞘群算法

刘景森， 袁蒙蒙， 左方

引用本文:
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): 2152−2160.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2020.0090

面向全局搜索的自适应领导者樽海鞘群算法

刘景森[1,2], 袁蒙蒙[2], 左 方[2,3†]

(1. 河南大学 智能网络系统研究所，河南 开封 475004；2. 河南大学 软件学院，河南 开封 475004；
3. 河南大学 河南省智能网络理论与关键技术国际联合实验室，河南 开封 475004)

摘 要: 为了进一步改善基本樽海鞘群算法容易陷入局部最优、寻优精度有时不高、求解结果不太稳定的不足,提出一种面向全局搜索的自适应领导者樽海鞘群算法. 首先,在领导者位置更新公式中引入上一代樽海鞘群位置,增强全局搜索的充分性,有效避免算法陷入局部极值;然后,在领导者位置更新公式中加入惯性权重,并在全局和局部搜索的选择上引入领导者-跟随者数量自适应调整策略,使算法在迭代前期领导者数目较多且受全局最优解影响较大,能以较大的全局搜索步幅快速收敛到全局最优区域,而在迭代后期领导者步幅较小且跟随者数量较多,可以在最优解附近深度挖掘,提高算法的收敛精度;随后给出算法流程并对时间复杂度进行理论分析;最后,通过5种代表性对比算法在12个不同特征基准测试函数多个维度上的函数优化仿真实验,表明所提出的改进算法的寻优精度和稳定性均有明显提升.

关键词: 樽海鞘群算法；全局搜索；自适应调整策略；寻优精度；收敛曲线

中图分类号: TP301.6 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2020.0090 **开放科学(资源服务)标识码(OSID):**

引用格式: 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): 2152-2160.

Global search-oriented adaptive leader salp swarm algorithm

LIU Jing-sen[1,2], *YUAN Meng-meng*[2], *ZUO Fang*[2,3†]

(1. Institute of Intelligent Networks System，Henan University，Kaifeng 475004，China；2. College of Software，Henan University，Kaifeng 475004，China；3. Henan International Joint Laboratory of Theories and Key Technologies on Intelligence Networks，Henan University，Kaifeng 475004，China)

Abstract: In order to further improve the shortcomings that the basic salp swarm algorithm is easy to fall into the local optimum, the optimization accuracy is sometimes not high, and the solution results are not stable, a global search-oriented adaptive leader salp swarm algorithm is proposed. The location of the last generation salp swarm group is introduced into the leader position update formula, which enhances the sufficiency of global search and effectively avoids the algorithm falling into local extremum. Then the inertia weight is added to the leader position update formula, and the leader-follower adaptive adjustment strategy is introduced to the choice of global and local search, so that the algorithm has a large number of leaders in the early iteration and is greatly influenced by the global optimal solution, it can quickly converge to the global optimal region with a larger global search step. At the end of the iteration, the leader's stride is small and the number of followers is large, so the algorithm can be mined deeply near the optimal solution to improve the convergence accuracy. Then the algorithm flow is given and the time complexity is analyzed theoretically. Finally, through the simulation experiment of function optimization of 5 representative comparison algorithms on multiple dimensions of 10 different feature benchmark functions, the test results show that the optimization accuracy and stability of the improved algorithm are significantly improved.

Keywords: salp swarm algorithm；global search；adaptive adjustment strategy；optimization accuracy；convergence curve

0 引 言

随着人类社会和认知的不断发展,各类应用问题和科学计算的规模与复杂性也在日益增长,传统的数值优化计算方法自身缺陷暴露,难以在合理有效时间内满足人们的求解需求. 近年来,基于仿生学的启发式智能优化算法因其操作简单、机制灵活、求

收稿日期: 2020-01-19；**修回日期:** 2020-03-22.

基金项目: 河南省重点研发与推广专项项目(182102310886)；河南省高等教育教学改革研究与实践项目(2019 SJGLX080Y)；河南大学研究生教育创新与质量提升项目(SYL18060145, SYL18020105).

责任编委: 夏元清.

†通讯作者. E-mail: zuofang@henu.edu.cn.

解高效等特点受到众多学者的青睐和研究. 如: 受自然界鸟群觅食过程启发提出的粒子群算法(particle swarm optimization, PSO)[1-2]; 受蝙蝠利用声呐来探测猎物、绕过障碍物行为启发提出的蝙蝠算法(bat algorithm, BA)[3]; 受自然界花朵授粉过程启发提出的花朵授粉算法(flower pollination algorithm, FPA)[4]; 受狼群领导和狩猎行为启发提出的灰狼优化算法(grey wolf optimizer, GWO)[5]; 受座头鲸捕食行为启发提出的鲸鱼算法(whale optimization algorithm, WOA)[6]等等. 这些智能优化算法的提出为解决大规模复杂问题提供了新思路, 并被广泛应用于多个领域.

樽海鞘群算法(salp swarm algorithm, SSA)[7]是由Mirjalili等提出的一种新型元启发式智能算法, 该算法中樽海鞘群以链的形式进行搜索, 结构简单, 几乎不涉及参数设置, 且求解性能较好, 已被成功应用于无源时差定位[8]、目标分类[9]、数字微分器[10]、混合动力系统[11]、光伏单元双二极管模型[12]等问题的求解之中.

不过, 樽海鞘群算法与其他启发式算法一样, 其本身同样存在容易陷入局部最优, 寻优精度有时不高, 求解结果不太稳定的问题. 为此, 许多学者针对樽海鞘群算法的不足之处作出了相应改进. Faris等[13]在领导者位置更新公式中引入8个传递函数将连续型SSA算法转换为离散型二进制, 并在传递函数中使用交叉算子来代替平均值, 增强了算法的探索能力. 王梦秋等[14]在跟随者位置更新公式中引入自适应评估移动策略和基于冯诺依曼拓扑结构的邻域最优引领策略, 加强个体间信息交流与协作, 同时引入反向学习策略以一定变异概率对樽海鞘位置进行扰动, 丰富樽海鞘种群多样性. Elaziz等[15]引入差分进化算法, 将差分进化算法的算子作为局部搜索的操作符, 增强樽海鞘群算法的特征挖掘能力. Yang等[16]在SSA算法的基础上扩展了多个独立的樽海鞘链, 引入基于虚拟种群的重组操作来实现不同樽海鞘链之间的全局协调, 提高算法的收敛稳定性. Gholami等[17]引入了一种突变机制来摆脱局部极小值, 提高种群的多样性. 邢致恺等[18]在领导者位置更新公式中引入莱维飞行轨迹, 使得该算法具有更加优秀的全局搜索能力和更强的收敛能力. Qais等[19]在领导者位置更新公式中引入平方指数, 提高了算法的收敛速度. 张达敏等[20]在食物源位置更新公式中引入疯狂算子, 对食物源位置产生一定扰动, 并在跟随者位置更新公式中引入自适应惯性权重, 更好地平衡了探索能力和开发能力. Kang等[21]提出了一种SSA结合Jaya优化器和支持向量机的改进算法, 用于在大坝健康监测建模中准确模拟温度效应; 丁力等[22]在跟随者位置更新过程中采用自适应算子来帮助SSA算法跳出局部最优的束缚, 使樽海鞘个体在前期有较强的全局收敛能力, 在后期能够得到相对准确的结果.

这些改进使算法在各自应用领域的求解性能得到了提升, 但樽海鞘群算法仍有进一步改进的空间. 本文提出一种面向全局搜索的自适应领导者樽海鞘群算法(an adaptive leader salp swarm algorithm for global search, ALSSA). 首先, 在领导者位置更新公式中引入上一代樽海鞘领导者位置, 使领导者位置受上一代领导者位置和上一代全局最优解的共同影响, 增强了全局搜索的充分性, 提高了算法跳出局部极值的能力. 然后, 在领导者位置更新公式中引入惯性权重, 并在领导者-跟随者数量比例中引入自适应调整策略, 使得算法在迭代前期领导者数量较多且受全局最优解影响较大, 可以快速收敛到全局最优区域; 而在迭代后期跟随者数量较多且领导者步幅较小, 便于在最优解附近精细挖掘, 很好地平衡了算法前期的广度搜索和后期的深度挖掘能力, 提高了算法的寻优精度. 理论分析证明了本文改进方法没有增加算法的时间复杂度, 而对多个不同寻优特征的CEC基准函数进行的多维度、多算法对比测试表明, 改进后算法的收敛性能、寻优精度和求解稳定性均有明显提升.

1　基本樽海鞘群算法

基本樽海鞘群算法SSA流程如下.

step 1: 初始化种群. 根据搜索空间每一维的上下限, 初始化樽海鞘的位置$X_j^i(i = 1, 2, \ldots, N, j = 1, 2, \ldots, D)$, 即

$$X_j^i = \mathrm{rand}(N, D) \times (ub(j) - lb(j)) + lb(j). \quad (1)$$

其中: N为樽海鞘群的种群规模, D为空间维度.

step 2: 根据目标函数计算每个樽海鞘的适应度值.

step 3: 选定食物源的初始位置. 对樽海鞘的适应度值进行排序, 最优樽海鞘的位置即为食物源的位置.

step 4: 确定领导者和跟随者. 位于樽海鞘链前一半数目的樽海鞘为领导者, 其余的为跟随者.

step 5: 更新樽海鞘领导者的位置

$$X_j^i(t) = \begin{cases} F_j + c_1((ub_j - lb_j)c_2 + lb_j), \ c_3 \geqslant 0.5; \\ F_j - c_1((ub_j - lb_j)c_2 + lb_j), \ c_3 < 0.5. \end{cases} \quad (2)$$

其中: t为当前迭代次数, $X_j^i(t)$为当前代樽海鞘领导者i在第 j维空间的位置, F_j为当前代食物源在第j

维空间的位置,ub_j和lb_j为第j维搜索空间的上、下限,c_2和c_3为$(0,1)$之间均匀分布的随机数,c_1随迭代次数增加而自适应递减. c_1的取值如下:

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

step 6: 更新樽海鞘跟随者的位置.

$$X_j^i(t) = \frac{1}{2}(X_j^i(t-1) + X_j^{i-1}(t-1)). \tag{4}$$

其中: t为当前迭代次数, $X_j^i(t)$为当前代樽海鞘跟随者i在第j维空间的位置, $X_j^i(t-1)$和$X_j^{i-1}(t-1)$分别为上一代中樽海鞘跟随者i、$i-1$在第j维的位置.

step 7: 对更新后个体的每一维进行边界处理,并根据更新后新的全局最优樽海鞘位置,更新食物源的位置.

step 8: 判断是否满足迭代次数,若是,则输出结果;否则,转step 4继续迭代进化.

2 改进算法ALSSA

2.1 改进领导者位置更新公式

2.1.1 引入上一代樽海鞘领导者位置

在基本樽海鞘群算法中,樽海鞘领导者从迭代开始就奔向全局最优值,导致全局搜索不充分,容易陷入局部极值区域,造成算法有时收敛精度较低. 本文在领导者位置更新公式中引入上一代樽海鞘领导者位置,使得领导者在位置更新阶段既受上一代樽海鞘领导者位置的影响,同时又受上一代全局最优解的影响,有效地避免了基本算法易陷入局部极值的问题,提高了算法的寻优精度. 改进后的樽海鞘领导者位置更新公式为

$$X_j^i(t) = X_j^i(t-1) + (\mathrm{FoodPosition}_j(t-1) - X_j^i(t-1)) \cdot \mathrm{rand}(). \tag{5}$$

其中: $X_j^i(t-1)$为上一代中第i个樽海鞘领导者在第j维的位置, $\mathrm{FoodPosition}_j(t-1)$为上一代中第$j$维的全局最优解,通过引入上一代的位置,樽海鞘领导者能够更有效地进行全局搜索,增强算法跳出局部极值的能力.

2.1.2 引入惯性权重

本文还在领导者位置更新公式中引入动态惯性权重,随迭代次数自适应递减的惯性权重w表示了樽海鞘领导者受全局最优解影响程度的变化. 在迭代前期,领导者受全局最优解影响较大,有较大的全局搜索步幅,能够更快速地找到全局最优区域. 而在迭代后期,大部分樽海鞘都已达到较优值,领导者受全局最优解影响变小,领导者可以在最优解附近深度挖掘,提高了算法的收敛精度. 本文中惯性权重的计算公式为

$$w = \frac{\mathrm{e}^{2(1-t/\max\_\mathrm{iter})} - \mathrm{e}^{-2(1-t/\max\_\mathrm{iter})}}{\mathrm{e}^{2(1-t/\max\_\mathrm{iter})} + \mathrm{e}^{-2(1-t/\max\_\mathrm{iter})}}. \tag{6}$$

结合式(5)和(6),新的樽海鞘领导者位置更新公式为

$$X_j^i(t) = X_j^i(t-1) + (w \cdot \mathrm{FoodPosition}_j(t-1) - X_j^i(t-1)) \cdot \mathrm{rand}(). \tag{7}$$

其中:惯性权重w为基于双曲正切函数的非线性递减值,其取值范围为$(0,1)$, t为当前迭代次数, max_iter为最大迭代次数, $X_j^i(t-1)$为上一代中第i个樽海鞘领导者在第j维的位置, $\mathrm{FoodPosition}_j(t-1)$为上一代中第$j$维的全局最优解. 通过引入惯性权重,使得改进后的算法能够在全局和局部搜索之间保持较好平衡,樽海鞘领导者更好地发挥领导者作用,提高算法的寻优精度.

2.2 引入领导者-跟随者自适应调整策略

在基本SSA算法中,樽海鞘领导者和跟随者的数目始终是各占种群中个体数的一半,这就使得在迭代前期,执行全局搜索的领导者比例过低,跟随者比例过高,领导者无法有效地进行全局搜索,全局搜索不充分,容易陷入局部最优;而在迭代后期,执行局部搜索的跟随者比例过低,局部搜索不够充分,容易造成寻优精度不高. 针对此问题,本文引入领导者-跟随者自适应调整策略,樽海鞘领导者的数目随迭代次数的增加自适应减少,跟随者数目自适应增加,在算法前期能够保持很强的全局搜索能力,同时兼顾局部搜索,而在算法运行后期,局部搜索逐渐增强,同时也兼顾全局搜索,从整体上提高了算法的收敛精度. 改进后的领导者-跟随者数量计算公式(每代中领导者数量等于rN,跟随者数量等于$(1-r)N$)为

$$r = b\Big(\tan\Big(-\frac{\pi t}{4\max\_\mathrm{iter}} + \frac{\pi}{4}\Big) - k \cdot \mathrm{rand}()\Big). \tag{8}$$

其中: t为当前迭代次数; max_iter为最大迭代次数; b为控制领导者-跟随者数量的比例系数,避免迭代前期的樽海鞘领导者或迭代后期的樽海鞘跟随者比例过高,全局和局部搜索失衡降低寻优性能,易陷入局部极值的现象,经大量实验测试,本文取值为0.75. 分析式(8)可知, r的值随着算法迭代次数的增加呈非线性递减趋势,于是领导者数量逐渐减少,跟随者数量逐渐增加,在迭代后期,更多的樽海鞘跟随者在全局最优值附近深度挖掘. k为扰动偏离因子,结合rand函数对递减的r值进行扰动,经大量实验反复测试, k等于0.2时,寻优效果最佳.

3 ALSSA算法流程与时间复杂度分析

3.1 ALSSA算法流程

算法1 ALSSA算法.

begin

设置算法参数: 种群大小N, 最大迭代次数max_iter, 个体维度D.

初始化种群$X_j^i(i=1,2,\ldots,N,\ j=1,2,\ldots,D)$,

由$f(x)$计算每个樽海鞘个体的适应度值, 并对适应度值进行排序,

选择最优个体作为食物源位置F.

while ($t <$ max_iter+1) do

由式(8)计算r.

按照式(6)计算惯性权重w.

for $i=1$ to N do

if $i \leqslant (r \cdot N)$ then

for $j=1$ to D do

由式(7)更新樽海鞘领导者i在第j维的位置.

end for

else

for $j=1$ to D do

由式(4)更新樽海鞘跟随者i在第j维的位置.

end for

end if

end for

for $i=1$ to N do

对更新后个体的每一维进行边界处理, 并计算樽海鞘的适应度值.

若小于食物源位置的适应度值, 则用当前樽海鞘位置替换食物源位置.

end for

$t=t+1$

end while

end

3.2　ALSSA算法的时间复杂度分析

时间复杂度是体现算法性能的关键因素, 它反映算法的运算效率. 文献[23]和[24]分别对万有引力算法和布谷鸟算法的时间复杂度进行了分析. 本文采用同样思想对ALSSA算法的时间复杂度进行分析.

在基本樽海鞘群算法SSA中, 假设种群规模为N, 个体维度为n, 设置初始参数的时间为η_0, 每一维上产生均匀分布随机数的时间为η_1, 求给定目标函数适应度值的时间为$f(n)$, 对樽海鞘适应度值进行排序并找到当前最优位置的时间为η_2, 则初始化种群和选定食物源初始位置阶段的时间复杂度为

$$T_1 = O(\eta_0 + N(n\eta_1 + f(n)) + \eta_2) = O(n + f(n)).$$

进入迭代后, 迭代次数为max_iter.

在领导者位置更新阶段, 领导者数目为$N/2$. 设由式(3), 计算c_1的时间为η_3, c_1同代相同, 每轮迭代只需计算1次. c_2和c_3为均匀分布的随机数, 每个个体的每一维各不相同, 其生成时间均与η_1一致. 式(2)中, 两种情况的计算时间完全相同, 故由式(2)进行每一维位置更新的时间为η_4, 则此阶段的时间复杂度为

$$T_2 = O\Big(\eta_3 + \Big(\frac{N}{2}\Big)n(\eta_1 + \eta_1 + \eta_4)\Big) = O(n).$$

在跟随者位置更新阶段, 跟随者数目为其余$N/2$. 设樽海鞘跟随者每一维位置更新所需的时间为η_5, 则这一阶段的时间复杂度为

$$T_3 = O\Big(\Big(\frac{N}{2}\Big)n\eta_5\Big) = O(n).$$

在边界处理和更新食物源位置阶段, 设每个樽海鞘个体每一维边界处理的时间为η_6, 个体计算适应度值的时间为$f(n)$, 与食物源所在位置适应度值的比较替换时间为η_7, 则此阶段的时间复杂度为

$$T_4 = O(N(n\eta_6 + f(n) + \eta_7)) = O(n + f(n)).$$

综上所述, SSA算法的总时间复杂度为

$$T(n) = T_1 + \text{max\_iter}(T_2 + T_3 + T_4) = O(n + f(n)).$$

在ALSSA改进算法中, 算法的种群规模、个体维度、参数设置时间、求给定目标函数适应度值的时间、对樽海鞘适应度值进行排序并找到当前最优位置的时间均与基本樽海鞘群算法一致, 因此ALSSA在初始化种群和选定食物源初始位置阶段的时间复杂度与基本樽海鞘群算法相同, 即

$$T_1' = T_1 = O(\eta_0 + N(n\eta_1 + f(n)) + \eta_2) = O(n + f(n)).$$

进入迭代后, 迭代次数为max_iter.

设由式(8)产生r的时间为t_1, 由式(6)产生自适应惯性权重w的时间为t_2. 选定领导者-跟随者时, ALSSA引入领导者-跟随者自适应调整策略, 故而在领导者位置更新阶段, 领导者数目为rN, 按式(7)对每一维进行位置更新的时间为t_3, 则这一阶段的时间复杂度为

$$T_2' = O(t_1 + t_2 + (rN)n(t_3)) = O(n).$$

在跟随者位置更新阶段, 跟随者数目为$(1-r)N$, 对樽海鞘跟随者每一维进行位置更新的时间与基本算法相同, 则这一阶段的时间复杂度为

$$T_3' = O((1-r)Nn\eta_5) = O(n).$$

在边界处理和更新食物源位置阶段, 每个樽海鞘个体每一维边界处理的时间、个体计算适应度值的

时间、与食物源所在位置适应度值的比较替换时间均与基本算法一致,故这一阶段的时间复杂度为

$$T_4' = T_4 = O(n + f(n)).$$

综上所述,ALSSA算法的总时间复杂度为

$$T(n) = T_1' + \max\_\text{iter}(T_2' + T_3' + T_4') = O(n + f(n)).$$

由此可知,与基本SSA相比,本文算法ALSSA并未增加时间复杂度,两者相同,执行效率没有下降.

4 仿真实验

为了全面检验本文改进算法的寻优能力,选取12个具有不同寻优特征的CEC2017基准测试函数,将本文算法ALSSA与SSA(2017)、Improved Salp Swarm Algorithm (ISSA, 2018)[14]、Enhanced Salp Swarm Algorithm(ESSA, 2019)[19]和Phasor Particle Swarm Optimization(PPSO, 2019)[25]共5种算法,分别在10维、50维和100维上进行对比测试.

4.1 测试函数

具体测试函数如表1所示. 表1中的12个测试函数均为CEC2017基准测试函数,为了方便测试,在CEC2017中,所有基准测试函数的自变量各维取值范围都是$[-100, 100]$.其中: $f_1(x), f_2(x), f_3(x), f_{11}(x)$是单峰函数,此类函数没有局部极小值,只有一个全局最小值,主要为了验证算法的收敛速度; $f_4(x) \sim f_{10}(x), f_{12}(x)$为复杂多峰函数,这类函数有较多局部极值点,主要用于测试算法跳出局部极值的能力.

4.2 寻优精度分析

为了测试改进算法ALSSA的寻优性能,本文将测试函数$f_1(x) \sim f_{12}(x)$的维度分别设置为$d = 5/10/100$,在不同维度下进行测试. 为保证实验的公平性与客观性,5种算法均在相同条件下独立运行30次,种群大小均为30,最大进化代数$\max_\text{iter} = 1\,000$. 算法参数设置方面,4种樽海鞘群类算法无需另外设置参数,而PPSO算法中学习因子c_1与c_2均为1.5,惯性权重w为0.8,与原文献和算法取值相同. 表2统计了10维、50维、100维下各算法运行30次得到的寻优结果的最优解、最差解和平均值.

由表2的数据可以看出,本文算法ALSSA在各函数不同维数下的求解精度和稳定性明显优于其他4种算法,尤其是在$f_2(x)$、$f_4(x)$、$f_5(x)$、$f_7(x)$、$f_8(x)$、$f_{10}(x)$这6个函数的各个维数上,得到的寻优结果的最优值、最差值和平均值都是理论最优值.

在10维条件下,对于函数$f_2(x)$、$f_4(x)$、$f_5(x)$、$f_7(x)$、$f_8(x)$、$f_{10}(x)$,本文算法ALSSA, 30次寻优结果的最优值、最差值、平均值都为理论最优值,求解效

表1 测试函数

序号	函数名	函数公式	最优值
$f_1(x)$	Bent Cigar	$f_1(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$	0
$f_2(x)$	Sum of Different Power	$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	0
$f_3(x)$	Zakharov	$f_3(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^4$	0
$f_4(x)$	Griewank's	$f_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4\,000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0
$f_5(x)$	Rastrigrin	$f_5(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	0
$f_6(x)$	Expanded Schaffer's	$g(x,y) = 0.5 + \frac{\sin^2(\sqrt{x^2+y^2}) - 0.5}{(1 + 0.001(x^2+y^2))^2}$, $f_6(x) = g(x_1,x_2) + g(x_2,x_3) + \ldots + g(x_{D-1},x_D) + g(x_D,x_1)$	0
$f_7(x)$	Non-continuous Rotated Rastrigin's	$f_7(x) = \sum_{i=1}^{D} [y_i^2 - 10\cos(2\pi y_i) + 10]$, $y_i = \begin{cases} x_i \longrightarrow \lvert x_i \rvert < \frac{1}{2}; \\ \frac{\text{round}(2x_i)}{2} \longrightarrow \lvert x_i \rvert \geqslant \frac{1}{2} \end{cases}$	0
$f_8(x)$	Rosenbrock's	$f_8(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	0
$f_9(x)$	Discus	$f_9(x) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$	0
$f_{10}(x)$	Ackley	$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$	0
$f_{11}(x)$	High Conditioned	$f_{11}(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	0
$f_{12}(x)$	Schaffer's F_7	$f_{12}(x) = \left[\frac{1}{D-1}\sum_{i=1}^{D-1} (\sqrt{s_i} * (\sin(50.0 s_i^{0.2}) + 1))\right]^2$	0

表 2 6种算法在固定迭代次数下的优化性能比较

函数	算法	$d=10$			$d=50$			$d=100$		
		最优解	最差解	平均值	最优解	最差解	平均值	最优解	最差解	平均值
$f_1(x)$	SSA	1.214 7	8.48e+03	2.02e+03	0.139 74	6 078.9	1 334.501 0	3.04e+05	4.88e+06	1.64e+06
	ISSA	6.01	9.82e+03	2.15e+03	1.54	9.18e+03	2.35e+03	3.95e+04	7.20e+05	2.03e+05
	ESSA	0	1.68e+09	3.21e+08	0	8.96e+10	2.55e+10	0	2.43e+11	8.04e+10
	PPSO	1.24e-05	3.65e+00	2.62e-01	1.13e+01	5.63e+04	1.02e+04	3.77e+02	1.77e+05	4.57e+04
	ALSSA	4.33e-214	3.37e-208	3.32e-209	2.52e-187	4.22e-184	6.68e-185	4.81e-181	3.62e-178	4.78e-179
$f_2(x)$	SSA	8.42e-20	1.52e-15	2.30e-16	2.93e-22	3.05e-16	7.53e-17	2.56e-19	1.54e-15	3.07e-16
	ISSA	9.81e-45	2.46e-37	1.92e-38	4.40e-46	6.66e-39	4.85e-40	1.82e-44	2.80e-37	2.21e-38
	ESSA	0	9.95e-14	6.42e-15	0	5.46e-29	3.41e-30	0	2.15e-32	1.34e-33
	PPSO	1.27e-231	4.70e-166	2.93e-167	3.11e-222	8.44e-156	5.30e-157	1.60e-215	5.12e-163	3.20e-164
	ALSSA	0	0	0	0	0	0	0	0	0
$f_3(x)$	SSA	6.44e-10	2.48e-09	1.56e-09	8.47e+03	6.79e+04	3.19e+04	2.75e+04	3.03e+05	1.35e+05
	ISSA	7.98e-10	5.06e-09	1.98e-09	8.93e+02	7.88e+03	4.14e+03	4.24e+04	1.39e+05	8.68e+04
	ESSA	0	4.57e+04	9.14e+03	0	4.43e+05	1.56e+05	0	9.33e+05	2.30e+05
	PPSO	3.58e-08	6.08e-05	7.69e-06	3.31e-04	1.02e+04	6.36e+02	1.07e-02	5.24e+04	1.20e+04
	ALSSA	1.83e-211	4.26e-206	2.87e-207	1.02e-180	2.19e-176	2.22e-177	7.74e-172	2.66e-167	2.85e-168
$f_4(x)$	SSA	8.87e-02	9.82e-01	3.09e-01	3.98e-05	2.26e-02	7.55e-03	1.03e-01	3.72e-01	1.93e-01
	ISSA	6.64e-02	1.47e+00	6.55e-01	1.86e-05	2.47e-02	5.43e-03	4.44e-02	1.65e-01	1.10e-01
	ESSA	0	2.43e+00	4.15e-01	0	2.26e+01	6.94e+00	0	5.58e+01	1.53e+01
	PPSO	1.26e-10	1.38e-01	1.80e-02	1.72e-06	3.23e-03	7.73e-04	1.00e-05	9.46e-03	1.98e-03
	ALSSA	0	0	0	0	0	0	0	0	0
$f_5(x)$	SSA	1.19e+01	7.56e+01	2.69e+01	2.85e+02	6.55e+02	4.84e+02	1.33e+03	2.59e+03	1.97e+03
	ISSA	1.39e+01	1.29e+02	7.19e+01	4.57e+02	2.99e+03	1.48e+03	1.97e+03	1.15e+04	3.96e+03
	ESSA	0	3.69e+03	5.73e+02	0	9.94e+04	2.60e+04	0	2.34e+05	6.25e+04
	PPSO	4.07e-08	7.96e+00	1.28e+00	1.99e-03	1.27e+02	1.75e+01	8.55e-03	3.13e+02	4.16e+01
	ALSSA	0	0	0	0	0	0	0	0	0
$f_6(x)$	SSA	1.057 4	3.204 6	2.370 993 75	11.974	19.679	17.059 75	28.351	42.321	36.454 625
	ISSA	2.382 7	3.941 7	3.412 875	21.204	23.249	22.460 593 7	43.639	48.005	45.922 375
	ESSA	0	4.446 1	1.730 487 5	0	24.216	12.894 437 5	0	48.23	16.889 437
	PPSO	4.43e-11	9.96e-01	8.17e-02	2.40e-05	1.46e+00	2.87e-01	4.98e-08	7.11e+00	1.20e+00
	ALSSA	0	2.23e-08	1.91e-09	0	1.97e-04	1.30e-05	0	2.65e-04	2.34e-05
$f_7(x)$	SSA	8	64	3.24e+01	3.92e+02	1.07e+03	7.10e+02	2.09e+03	4.43e+03	3.10e+03
	ISSA	1.70e+01	1.19e+02	5.76e+01	6.74e+02	2.16e+03	1.33e+03	2.73e+03	5.28e+03	4.14e+03
	ESSA	0	2.81e+03	3.69e+02	0	8.20e+04	2.43e+04	0	2.29e+05	6.65e+04
	PPSO	1.27e-09	8.00e+00	1.94e+00	7.43e-05	4.80e+01	1.23e+01	4.99e-06	9.90e+01	1.88e+01
	ALSSA	0	0	0	0	0	0	0	0	0
$f_8(x)$	SSA	4.322 2	3 657.5	621.475 40	45.911	9.19e+04	9.27e+03	2.39e+03	4.47e+05	7.46e+04
	ISSA	9.44e-01	8.73e+03	1.45e+03	4.54e+01	3.73e+04	6.09e+03	1.47e+03	5.05e+05	8.97e+04
	ESSA	0	1.84e+08	1.80e+07	0	4.14e+10	1.01e+10	0	1.32e+11	4.29e+10
	PPSO	1.87e-06	8.04e+01	6.45e+00	5.60e-04	9.12e+01	9.34e+00	2.58e-01	1.47e+02	2.60e+01
	ALSSA	0	0	0	0	0	0	0	0	0
$f_9(x)$	SSA	5.14e+02	1.19e+04	4.92e+03	8.92e+03	7.48e+04	4.08e+04	2.26e+04	1.71e+05	8.46e+04
	ISSA	1.17e-01	1.07e+01	2.12e+00	4.86e+03	2.67e+04	1.23e+04	3.80e+04	7.79e+04	6.01e+04
	ESSA	0	6.57e+04	1.70e+04	0	3.21e+05	1.10e+05	0	6.38e+05	1.99e+05
	PPSO	3.68e-04	4.29e-01	3.94e-02	2.75e-01	5.07e+01	1.52e+01	4.96e-01	2.01e+04	1.92e+03
	ALSSA	2.55e-217	2.24e-213	2.72e-214	1.3e-192	1.5e-189	3.275e-190	2.35e-186	6.96e-184	1.20e-184
$f_{10}(x)$	SSA	2.16e-05	20	12.182 153 4	3.146 1	20	18.453 515 6	20.003	20.064	20.015 656 2
	ISSA	20	20	20	20	20	20	20	20	20
	ESSA	0	20	10	0	20	11.250 067 0	0	20	12.5
	PPSO	0	2.00e+01	1.25e+01	0	2.00e+01	1.44e+01	8.95e-06	2.00e+01	1.80e+01
	ALSSA	0	0	0	0	0	0	0	0	0
$f_{11}(x)$	SSA	2.057 1e+04	9.162 9e+05	2.730 7e+05	3.380 9e+06	2.669 3e+07	1.170 6e+07	1.757 6e+07	1.221 0e+08	4.629 3e+07
	ISSA	2.294 2e+04	6.053 4e+05	1.832 3e+05	6.206 0e+06	2.814 9e+07	1.396 5e+07	3.509 1e+07	1.050 4e+08	6.232 8e+07
	ESSA	0	7.178 4e+07	9.737 5e+06	0	2.003 0e+09	2.707 6e+08	0	9.409 6e+09	1.623 7e+09
	PPSO	7.882 9e-03	1.353 2e+01	1.613 5e+00	2.167 9e-01	3.747 9e+03	4.974 1e+02	1.769 0e+01	2.566 9e+04	4.171 9e+03
	ALSSA	6.972 3e-216	1.187 8e-210	1.223 5e-211	1.046 8e-188	4.350 7e-186	6.177 7e-187	1.499 7e-182	2.771 3e-180	3.134 8e-181
$f_{12}(x)$	SSA	2.300 0e-01	2.129 5e+01	6.301 8e+00	1.426 4e+01	2.611 5e+01	2.120 9e+01	1.915 6e+01	2.872 4e+01	2.389 6e+01
	ISSA	3.772 3e+00	7.084 2e+01	3.140 0e+01	4.582 7e+01	8.138 8e+01	6.685 4e+01	6.071 4e+01	8.833 5e+01	7.628 8e+01
	ESSA	0	4.969 6e+01	1.084 9e+01	0	1.105 4e+02	5.012 2e+01	0	1.331 9e+02	6.768 3e+01
	PPSO	5.134 6e-04	3.655 4e-01	5.093 9e-03	5.192 4e-04	8.180 4e-01	1.197 8e-01	9.193 3e-03	4.841 8e-01	9.757 81e-02
	ALSSA	7.179 4e-75	4.120 4e-51	2.575 3e-52	1.423 3e-89	5.205 7e-73	3.280 5e-74	2.287 6e-89	5.163 0e-80	4.257 8e-81

果极好;而SSA和ISSA算法在这6个函数上的最优值都没有达到理论最优值;PPSO算法在函数$f_{10}(x)$上仅最优值为理论最优值,而在其他5个函数上的最优值、最差值、平均值都没有达到理论最优值;在这6个函数上,ESSA算法的最优值都为理论最优值,但其最差值、平均值的精度明显低于其他4个算法,显示出求解的不稳定性. 对于函数$f_1(x)$、$f_3(x)$、$f_9(x)$、$f_{11}(x)$、$f_{12}(x)$,ALSSA算法30次寻优结果的最优值、最差值、平均值都远远好于SSA、ISSA和PPSO算法,虽然最优值稍逊于ESSA算法找到的理论最优值0,但最差值和平均值的寻优精度皆远高于ESSA算法,寻优精度和稳定性都很出色. 对于函数$f_6(x)$,ALSSA算法的最优值为理论最优值,最差值和平均值精度也远远好于其他4个对比算法.

在50维和100维的高维条件下,5种算法的求解精度都会随着维度的增加而有所降低,但ALSSA算法受维度变化影响较小,寻优精度依然高于其他4个算法. 对于函数$f_2(x)$、$f_4(x)$、$f_5(x)$、$f_7(x)$、$f_8(x)$、$f_{10}(x)$,本文算法ALSSA在50维和100维下30次寻优结果的最优值、最差值、平均值依然都为理论最优值,求解效果十分稳定. 在50维条件下SSA和ISSA算法在这6个函数上的最优值都没有达到理论最优值;PPSO算法只在函数$f_{10}(x)$上的最优值为理论最优值,但最差值和平均值不是,且在其他5个函数上的最优值、最差值、平均值都没有达到理论最优值. 在100维条件下,SSA、ISSA和PPSO算法在这6个函数上的最优值都没有达到理论最优值. 在50维和100维条件下,ESSA算法的最优值都为理论最优值,但最差值和平均值都不是理论最优值,精度更明显低于ALSSA算法,求解稳定性较差. 对于函数$f_6(x)$,在50维和100维条件下,ALSSA算法的最优值为理论最优值,最差值和平均值精度也远远好于其他4个算法. 对于函数$f_1(x)$、$f_3(x)$、$f_9(x)$、$f_{11}(x)$、$f_{12}(x)$,ALSSA算法在50维和100维下的最优值、最差值、平均值都远远好于SSA、ISSA和PPSO算法. 虽然其30次运行结果的最优值稍逊于ESSA算法,未达到理论最优值0,但最差值和平均值的寻优精度皆远高于ESSA算法,寻优精度和稳定性都十分优越.

上述求解结果和分析表明,在各测试函数的不同维数上,ALSSA算法均呈现出较好的求解性能,在10维、50维、100维条件下的求解精度和稳定性明显优于其他4个对比算法.

4.3 收敛曲线分析

一个算法性能的优劣,可以直观地通过收敛曲线展现出来,收敛曲线显示了算法陷入局部最优中的次数和收敛速度. 由于多峰函数更为复杂,易使算法陷入局部极值,这些函数的收敛情况更能说明算法的寻优能力. 因此下面列出12个测试函数中所有8个多峰函数的收敛曲线图,而单峰函数的收敛曲线比较简单且结果与多峰函数类似,不再冗赘列出. 图1～图8是ALSSA、ESSA、SSA、PPSO和ISSA五种算法在维度$d=100$时求解函数$f_4(x)\sim f_{10}(x), f_{12}(x)$的收敛情况.

通过8个函数的收敛曲线清晰地展现了ALSSA、ESSA、SSA、PPSO和ISSA算法在进化过程中适应度值的变化趋势.

对于图1、图2、图4和图5,SSA、ESSA和ISSA算法收敛速度很慢,PPSO算法在迭代前期收敛速度

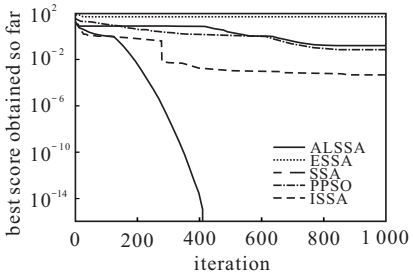

图1 $f_4(x)$函数的收敛曲线

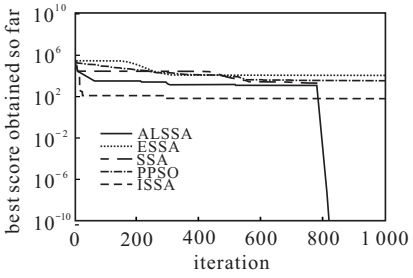

图2 $f_5(x)$函数的收敛曲线

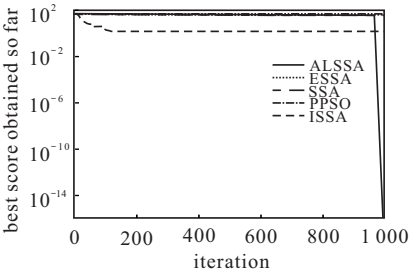

图3 $f_6(x)$函数的收敛曲线

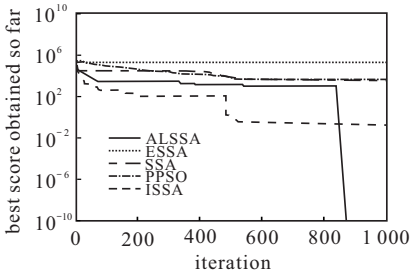

图4 $f_7(x)$函数的收敛曲线

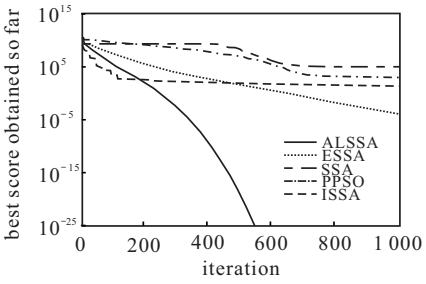

图 5　$f_8(x)$ 函数的收敛曲线

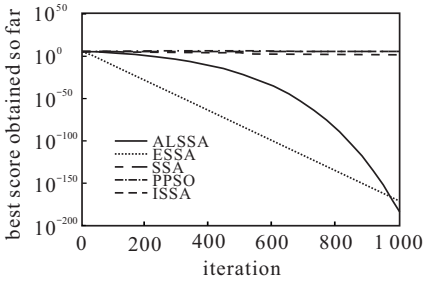

图 6　$f_9(x)$ 函数的收敛曲线

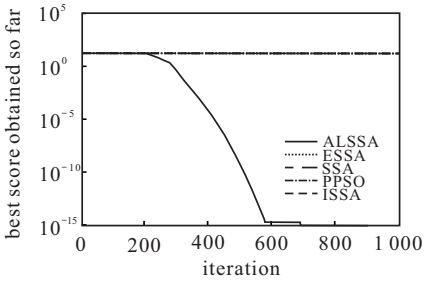

图 7　$f_{10}(x)$ 函数的收敛曲线

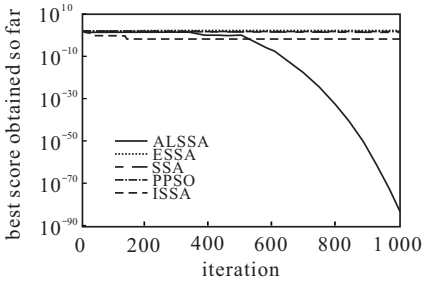

图 8　$f_{12}(x)$ 函数的收敛曲线

较快,但很快陷入局部极值无法跳出,迭代结束时PPSO、ESSA、SSA和ISSA算法的精度依然远远低于ALSSA算法的收敛精度. 对于图3和图8,ESSA、SSA和ISSA算法在迭代初期就陷入局部极值无法跳出,PPSO算法虽然在迭代前期收敛速度很快,但在200代左右时陷入局部极值且直到迭代结束也未能跳出,而ALSSA算法虽在迭代初期也陷入局部极值但之后跳出该极值,并快速收敛到最优解. 对于图6,SSA、PPSO和ISSA算法在迭代初期就陷入局部极值无法跳出,收敛精度较低,ESSA算法虽在前900代收敛速度比ALSSA快,但在迭代至1 000代时其收敛精度已经低于ALSSA算法. 对于图7,ESSA、SSA、PPSO和ISSA算法在迭代初期就陷入局部极值无法跳出,而ALSSA算法从迭代开始就一直保持较快的收敛速度,700代就可以收敛到全局最优解.

由上述分析可知,ALSSA算法的寻优精度和收敛性能明显优于其他4种算法,这主要是因为在领导者位置更新中引入上一代领导者位置,有效地避免了基本算法易陷入局部极值的问题. 而引入惯性权重和领导者-跟随者数量自适应调整策略,使算法在迭代前期能够快速找到全局最优区域;在迭代后期可以在最优解区域精细挖掘,从而在整体上大大提高了算法的收敛性能和求解精度.

综上所述,本文提出的ALSSA算法不管是在低维条件下还是在高维条件下都具有较好的进化寻优性能,其求解精度和寻优稳定性均优于ESSA、SSA、PPSO和ISSA四种对比算法.

5　结　论

本文针对基本樽海鞘群算法求解精度不高,易陷入局部极值等不足进行了改进. 首先,在领导者位置更新公式中引入上一代领导者位置,使领导者位置更新既受上一代樽海鞘领导者位置的影响,同时又受上一代全局最优解的影响,提升了算法跳出局部极值的能力. 然后,在领导者位置更新公式中加入惯性权重,并在领导者-跟随者数量比例中引入自适应调整策略,使算法在迭代前期可以快速收敛到全局最优解区域,在迭代后期集中在最优解附近区域精细挖掘,提高算法的收敛精度. 最后,通过对5种算法在12个不同特征基准测试函数上寻优结果及收敛曲线的对比分析,验证了本文改进算法的有效性和优越性. 下一步将继续改进樽海鞘群算法的寻优性能,提高其寻优稳定性和收敛速度,并以此为基础,将其应用于更多实际问题的求解中.

参考文献(References)

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

[2] Coello C A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 256-279.

[3] Yang X S. Bat algorithm for multi-objective optimisation[J]. International Journal of Bio-Inspired Computation, 2011, 3(5): 267-274.

[4] Yang X S. Flower Pollination algorithm for global optimization[C]. International Conference on Unconventional Computation and Natural Computation. Berlin: Springer-Verlag, 2013: 240-249.

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

[6] Mirjalili S, Lewis A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95(5): 51-67.

[7] Mirjalili S, Gandomi A H, Mirjalili S Z. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems[J]. Advances in Engineering Software, 2017, 114(6): 163-191.

[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 and Technology, 2018, 40(7): 1591-1597.)

[9] Khishe M, Mohammadi H. Passive sonar target classification using multi-layer perceptron trained by salp swarm algorithm[J]. Ocean Engineering, 2019, 181: 98-108.

[10] Ali T A A, Xiao Z, Sun J R, et al. Optimal design of IIR wideband digital differentiators and integrators using salp swarm algorithm[J]. Knowledge-Based Systems, 2019, 182: 104834.

[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: 147-160.

[12] Abbassi R, Abbassi A, Heidari A A, et al. An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models[J]. Energy Conversion and Management, 2019, 179: 362-372.

[13] Faris H, Mafarja M M, Heidari A A, et al. An efficient binary salp swarm algorithm with crossover scheme for feature selection problems[J]. Knowledge-Based Systems, 2018, 154: 43-67.

[14] 王梦秋, 王艳, 纪志成. 基于改进樽海鞘群算法的PMSM多参数辨识[J]. 系统仿真学报, 2018, 30(11): 4284-4291.
(Wang M Q, Wang Y, Ji Z C. Multi-parameter identification of PMSM based on improved salp swarm algorithm[J]. Journal of System Simulation, 2018, 30(11): 4284-4291.)

[15] Elaziz M A, Lin L, Jayasena K P N, et al. Multiobjective big data optimization based on a hybrid salp swarm algorithm and differential evolution[J]. Applied Mathematical Modelling, 2020, 80: 929-943.

[16] Yang B, Zhong L E, Shu H C, et al. Novel bio-inspired memetic salp swarm algorithm and application to MPPT for PV systems considering partial shading condition[J]. Journal of Cleaner Production, 2019, 215: 1203-1222.

[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.

[18] 邢致恺, 贾鹤鸣, 宋文龙. 基于莱维飞行樽海鞘群优化算法的多阈值图像分割[J]. 自动化学报, DOI: doi.org/10.16383/j.aas.c180140.
(Xing Z K, Jia H M, Song W L. Multi-threshold image segmentation based on Levy flying salp swarm optimization algorithm[J]. Journal of Automation, DOI: doi.org/10.16383/j.aas.c180140.)

[19] 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: 82-96.

[20] 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.
(Zhang D M, Chen Z Y, Xin Z Y, et al. Salp swarm algorithm based on crazyiness and adaptive[J]. Control and Decision, 2020, 35(9): 2112-2120.)

[21] Kang F, Li J J, Dai J H. Prediction of long-term temperature effect in structural health monitoring of concrete dams using support vector machines with Jaya optimizer and salp swarm algorithms[J]. Advances in Engineering Software, 2019, 131: 60-76.

[22] 丁力, 高振奇, 虞青. 基于改进樽海鞘群算法的四旋翼飞行器姿态优化控制[J]. 农业机械学报, 2019, 50(10): 243-250.
(Ding L, Gao Z Q, Yu Q. Optimal attitude control of a four rotor aircraft based on improved salp swarm algorithm[J]. Journal of Agricultural Machinery, 2019, 50(10): 243-250.)

[23] Liu J S, Xing Y H, Li Y. A gravitational search algorithm with adaptive mixed mutation for function optimization[J]. International Journal of Performability Engineering, 2018, 14(4): 681-690.

[24] 周欢, 李煜. 具有动态惯性权重的布谷鸟搜索算法[J]. 智能系统学报, 2015, 10(4): 645-651.
(Zhou H, Li Y. Cuckoo search algorithm with dynamic inertia weight[J]. CAAI Transactions on Intelligent Systems, 2015, 10(4): 645-651.)

[25] Ghasemi M, Akbari E, Rahimnejad A, et al. Phasor particle swarm optimization: A simple and efficient variant of PSO[J]. Soft Computing, 2013, 23(19): 9701-9718.

作者简介

刘景森(1968−), 男, 教授, 博士, 从事智能算法、优化控制等研究, E-mail: ljs@henu.edu.cn;

袁蒙蒙(1993−), 女, 硕士生, 从事智能算法的研究, E-mail: ymmhenu@163.com;

左方(1980−), 男, 副教授, 博士, 从事计算机网络、智能优化算法等研究, E-mail: zuofang@henu.edu.cn.

(责任编辑：孙艺红)