基于衰减因子和动态学习的改进樽海鞘群算法

陈 雷[1,2†], 蔺 悦[1], 康志龙[3]

(1. 天津大学 微电子学院, 天津 300072; 2. 天津商业大学 信息工程学院, 天津 300134;

3. 河北工业大学 电子信息工程学院, 天津 300401)

摘要: 樽海鞘群算法是一种新型的群智能优化算法. 与其他智能优化算法相比, 樽海鞘群算法的优化求解策略仍有待改进, 以进一步提高该算法的求解精度和寻优效率. 本文提出一种基于衰减因子和动态学习的改进樽海鞘群算法, 通过在领导者更新阶段添加衰减因子, 提高算法的局部开发能力, 在跟随者更新阶段引入动态学习策略, 提高算法的全局搜索能力. 本文对16个测试函数进行实验, 将提出的改进算法与其他智能优化算法比较, 实验结果表明, 本文提出的改进算法在收敛精度和收敛速度方面有较大提升, 具有良好的优化性能.

关键词: 樽海鞘群算法; 衰减因子; 动态学习; 群智能; 优化算法

引用格式: 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. 控制理论与应用, 2020, 37(8): 1766 – 1780

DOI: 10.7641/CTA.2020.90766

Improved salp swarm algorithm based on reduction factor and dynamic learning

CHEN Lei[1,2†], LIN Yue[1], KANG Zhi-long[3]

(1. School of Microelectronic, Tianjin University, Tianjin 300072, China;
2. School of Information Engineering, Tianjin University of Commerce, Tianjin 300134, China;
3. School of Electronic Information Engineering, Hebei University of Technology, Tianjin 300401, China)

Abstract: Salp swarm algorithm is one of the most recently proposed swarm intelligent optimization algorithms. Compared with other intelligent optimization algorithms, the optimization strategy of salp swarm algorithm needs to be improved to enhance the convergence accuracy and speed. This paper proposes an improved salp swarm algorithm based on reduction factor and dynamic learning. Firstly, reduction factor is added in the update stage of leader salps, in order to improve the local exploitation ability. Next, dynamic learning strategy is imported in the update stage of follower salps, aiming to improve the global search ability. In this paper, 16 test functions are conducted in the experiment of comparison between the proposed improved algorithm and other intelligent optimization algorithms. The results show that the proposed improved algorithm has a good improvement in convergence accuracy and speed, which has good optimization performance.

Key words: salp swarm algorithm; reduction factor; dynamic learning; swarm intelligence; optimization algorithm

Citation: CHEN Lei, LIN Yue, KANG Zhilong. Improved salp swarm algorithm based on reduction factor and dynamic learning. *Control Theory & Applications*, 2020, 37(8): 1766 – 1780

1 引言

樽海鞘群算法(salp swarm algorithm, SSA)[1]是澳大利亚学者Mirjalili在2017年提出的一种新型群智能优化算法, 该算法模拟了海洋动物樽海鞘的群体觅食行为, 机制简单易懂, 操作方便, 易于实现, 已经成为国内外很多学者的研究热点. 目前, 该算法已经被应用于光伏系统优化[2–4]、网络系统管理[5]、特征选择[6–7]、图像处理[8–10]和训练神经网络[11]等实际问题中.

SSA相较于其他一些智能优化算法具有一定优势, 参数设置简单, 收敛速度快, 鲁棒性强, 处理低维问题优化性能良好. 但是SSA也存在迭代后期搜索精度不

收稿日期: 2019−09−11; 录用日期: 2020−04−13.
†通信作者. E-mail: chenlei@tjcu.edu.cn; Tel.: +86 13512999888.
本文责任编委: 李少远.
国家自然科学基金项目(61401307), 河北省高等学校科学技术研究项目(ZD2018045), 天津市企业科技特派员项目(18JCTPJC57500)资助.
Supported by the National Natural Science Foundation of China (61401307), the Colleges and Universities in Hebei Province Science and Technology Research Project (ZD2018045) and the Tianjin Research Program of Science and Technology Commissioner of China (18JCTPJC57500).

高, 种群多样性较差等缺点, 限制了算法的局部开发能力和全局探索能力. 为改善其优化性能, 许多研究学者提出多种改进算法. 从改进参数的角度: Sayed等人[12]将混沌映射引入SSA, 用混沌变量代替重要随机参数, 加强随机参数的动态特性, 提高了SSA的开发和探索能力. Wang等人[13]提出一种基于单纯形法的改进樽海鞘群算法, 改变随机策略, 增加种群的多样性, 加强了算法的局部搜索能力. Qais等人[14]调整领导者趋向食物源的重要参数, 并且在跟随者位置更新公式中添加随机参数, 帮助领导者更好的趋向最优值, 改善了SSA求解精度不足的缺陷. 从与其他智能优化算法结合的角度: Ibrahim等人[15]将SSA与粒子群优化算法结合, 提高在探索时期的灵活性和多样性, 达到了更高的寻优效率. Khamees等人[16]将SSA与模拟退火算法结合, 在解决多目标优化问题上表现出色.

以上改进算法均从不同角度增强了SSA算法的寻优性能. 但在实际问题中, 往往是高维单峰和高维多峰的复杂问题. 对于解决这类复杂的优化问题, SSA的求解精度和寻优效率仍需进一步提高. 针对上述存在的问题, 本文提出一种基于衰减因子和动态学习的改进樽海鞘群算法(salp swarm algorithm based on reduction factor and dynamic learning, RDSSA), 在领导者位置更新阶段添加衰减因子, 提高算法在迭代后期的局部开发能力; 在跟随者位置更新阶段引入动态学习策略, 提高算法的全局探索能力, 解决了算法在高维多峰优化问题上求解精度不高, 寻优效率低的问题.

通过对16个测试函数进行实验, 并与其他改进樽海鞘群算法及其他智能优化算法比较, 验证RDSSA的优化性能.

2　樽海鞘群算法

SSA建立了一种用于求解优化问题的樽海鞘链模型, 将樽海鞘群分为两类: 领导者和跟随者, 领导者是处在链的前面起带领作用的个体, 跟随者直接或间接地相互追随. 樽海鞘的位置在D维搜索空间中定义, 食物源F作为樽海鞘群的觅食目标.

领导者的位置更新公式如下:

$$x_j^1=\begin{cases}F_j+c_1((ub_j-lb_j)c_2+lb_j),\ c_3\geqslant 0.5,\\ F_j-c_1((ub_j-lb_j)c_2+lb_j),\ c_3<0.5,\end{cases}\tag{1}$$

其中: x_j^1表示第1个樽海鞘个体(领导者), 在第j维的位置; F_j表示在第j维食物源的位置; ub_j和lb_j分别表示在第j维搜索空间上、下界; 系数c_1是SSA中的一个重要参数, 定义如下:

$$c_1=2\mathrm{e}^{-(\frac{4l}{L})^2},\tag{2}$$

其中: l表示当前迭代次数, L表示总迭代次数. 参数c_2和c_3取$[0,1]$的随机数, 它们分别决定了在j维位置更新的移动步长, 以及前进还是后退.

跟随者的位置更新公式如下:

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

其中: x_j^i表示第i个樽海鞘个体(跟随者), 在第j维的位置. 当$i=2$, 跟随者x_j^2的更新与领导者x_j^1和跟随者本身x_j^2有关, 由领导者直接引领; 当$i>2$, 跟随者x_j^i的更新与跟随者x_j^{i-1}和x_j^i有关, 由领导者间接引领.

SSA采用优胜劣汰策略, 通过计算所有个体的适应值, 比较当前迭代次数的适应值与先前最优适应值, 不断接近食物源位置. 由此可以模拟樽海鞘群的觅食行为, 解决最优化问题.

3　基于衰减因子和动态学习的改进樽海鞘群算法

通过研究SSA的原理发现, 由于位置更新搜索范围无约束, 且精英个体的影响权重小, 导致SSA在迭代后期不能进行很精确的搜索, 跟随者不能很好的协助个体位置更新. 因此, 本文的改进思路从两个方面考虑: 针对SSA在领导者更新阶段搜索范围不受限的问题, 添加衰减因子, 增强迭代后期的局部开发能力; 针对跟随者位置更新的局限性, 引入动态学习策略, 提高全局探索能力.

3.1　添加衰减因子的樽海鞘群算法

基本SSA在领导者位置更新阶段, 个体在食物源附近移动, 搜索范围不受限制, 使得收敛后期个体不能在极值点进行精确搜索, 还有可能跳出极值点. 为了改善这一问题, 本文提出将衰减因子引入SSA, 得到添加衰减因子的樽海鞘群算法(reduction factor salp swarm algorithm, RSSA), 使得领导者位置更新范围随着迭代次数的增加而逐渐减小, 收敛前期避免陷入局部极值, 收敛后期越来越逼近最优值, 达到更高的求解精度.

添加衰减因子的领导者位置更新公式(4)如下:

$$x_j^1=\begin{cases}A(l)[F_j+c_1((ub_j-lb_j)c_2+lb_j)],\ c_3\geqslant 0.5,\\ A(l)[F_j-c_1((ub_j-lb_j)c_2+lb_j)],\ c_3<0.5,\end{cases}\tag{4}$$

其中控制搜索范围的衰减因子$A(l)$是一个非线性递减函数, 定义如下:

$$A(l)=\mathrm{e}^{-30(\frac{l}{L})}.\tag{5}$$

收敛前期, 搜索范围不受限, 个体可以充分在全局移动, 充分发挥算法的全局搜索能力, 避免陷入局部极值; 收敛后期, 随着个体越来越逼近最优值, 搜索范围也逐渐减小, 个体在限制范围内进行精确搜索, 增

强局部搜索能力, 以达到更高的求解精度.

3.2 引入动态学习的樽海鞘群算法

基本SSA算法没有参数影响跟随者的位置更新, 跟随者的移动由个体自身位置和前一个个体位置综合决定, 精英个体的影响权重小, 使得跟随者对领导者的协助作用很小. 为了增强精英个体的影响权重, 本文将动态学习策略引入SSA, 得到引入动态学习的樽海鞘群算法(dynamic learning salp swarm algorithm, DSSA), 先比较x_j^i与x_j^{i-1}的适应值, 在适应值较大的位置(即离最优值距离较远的位置)上添加削弱因子k, 以削弱较差位置个体的影响权重, 增强较优位置个体的影响权重.

引入动态学习策略的跟随者位置更新公式如下:

$$x_j^i = \begin{cases} \dfrac{1}{2}(k \times x_j^i + x_j^{i-1}),\ f(x_j^i) \geqslant f(x_j^{i-1}), \\ \dfrac{1}{2}(x_j^i + k \times x_j^{i-1}),\ f(x_j^i) < f(x_j^{i-1}), \end{cases} \quad i \geqslant 2, \tag{6}$$

其中: $f(x_j^i)$和$f(x_j^{i-1})$分别表示两位置的适应值, k是服从参数为0.5的指数分布随机数.

在收敛过程中, 精英个体能更好的发挥协助作用, 帮助领导者做决策, 不断向食物源逼近, 提高寻优效率.

3.3 RDSSA的实现流程

将RSSA和DSSA结合, 同时改变领导者和跟随者的位置更新公式, 可以得到新的优化算法RDSSA, 提高领导者局部开发能力, 增强跟随者的协助作用, 以提高收敛速度, 获得更好的优化效果. RDSSA的实现流程如下所示.

Step 1 设定种群规模N、迭代次数Iteration、维数D和上、下边界;

Step 2 初始化樽海鞘群个体的位置, 并计算各个体的适应值Fitness, 将最小适应值个体的位置确定为食物源位置FoodPosition;

Step 3 生成衰减因子$A(l)$, 根据式(4)更新领导者位置;

Step 4 生成随机数k, 根据式(6)更新跟随者位置;

Step 5 计算更新位置后的个体适应值, 若小于当前FoodPosition, 则更新FoodPosition;

Step 6 判断当前迭代次数是否达到预设迭代次数, 若已达到, 结束迭代, 否则返回执行Step 3;

Step 7 输出FoodPosition位置及该位置上的适应值Fitness.

RDSSA的伪代码如下所示:

1) 随机生成初始种群$x_i(i = 1, 2, \cdots, N)$
2) **while** $l < \text{Iteration}$
3) 计算每个个体适应值Fitness
4) FoodPosition = 最优个体位置
5) 依据式(5)生成$A(l)$
6) **for** 每个个体x_i
7) **if** $(i == 1)$
8) 依据式(4)更新领导者位置
9) **else**
10) 依据式(6)更新跟随者位置
11) **end**
12) 计算每个个体适应值Fitness, 并更新FoodPosition
13) **end**
14) 确保每个个体位置都在搜索空间上、下界之内
15) **end**
16) 返回FoodPosition和该位置上的适应值Fitness.

4 测试实验分析

4.1 实验设计

为了验证本文提出的RSSA, DSSA和RDSSA的性能, 实验选取16个典型全局优化测试函数进行实验. 测试函数的形式、搜索范围、理论极值和目标收敛精度如表1所示, 其中$f_1 \sim f_8$为单峰函数, $f_9 \sim f_{16}$为多峰函数. 本文实验分为3个部分: 1) 与基本SSA比较, 固定迭代次数, 观察收敛速度和收敛精度; 2) 与基本SSA比较, 固定目标收敛精度, 观察达到目标收敛精度所需的迭代次数; 3) 与其他算法比较, 固定迭代次数, 观察收敛速度和收敛精度.

实验硬件配置为Intel(R) Core(TM) i5–3210M, CPU 2.50 GHz, RAM 4.00 GB, Windows 64位操作系统, 算法调试运行基于MATLAB 2014a. 为了遵循实验公平性原则, 实验基本参数保持一致: 种群数量N设为50, 函数维数D分别设为30, 45和60, 迭代次数Iteration设为500, 每次实验结果为独立运行30次的平均值.

4.2 实验结果及分析

4.2.1 固定迭代次数下的收敛精度

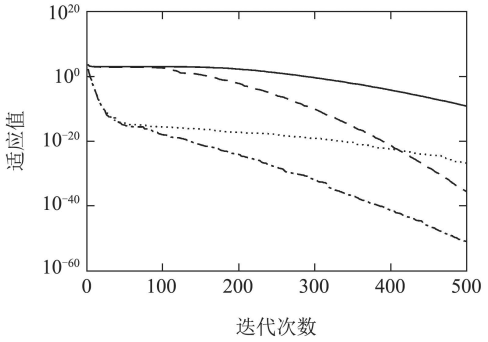
固定迭代次数的实验结果如图1所示, 显示了SSA, RSSA, DSSA和RDSSA4种算法对以上16个测试函数的收敛曲线. 为了便于观察, 将收敛曲线的纵轴刻度设置为以10为底的指数形式.

表 1 典型测试函数

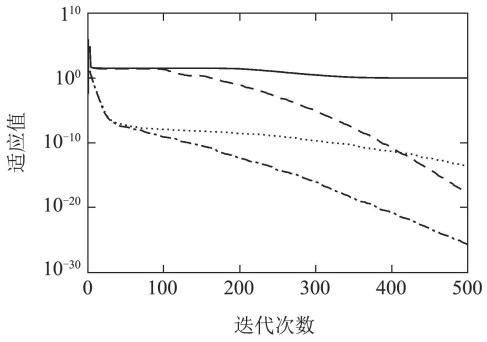
Table 1 Benchmark test functions

函数形式	搜索范围	理论极值	目标精度
$f_1(x)=\sum\limits_{i=1}^{n}x_i^2$	$[-100,100]$	0	1×10^{-27}
$f_2(x)=\sum\limits_{i=1}^{n}\lvert x_i\rvert+\prod\limits_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]$	0	1×10^{-14}
$f_3(x)=\sum\limits_{i=1}^{n}(\sum\limits_{j=1}^{i}x_j)^2$	$[-100,100]$	0	1×10^{-24}
$f_4(x)=\max\{\lvert x_i\rvert,\ 1\leqslant i\leqslant n\}$	$[-100,100]$	0	1×10^{-14}
$f_5(x)=\sum\limits_{i=1}^{n}ix_i^2$	$[-10,10]$	0	1×10^{-25}
$f_6(x)=\sum\limits_{i=1}^{n}\lvert x_i\sin x_i+0.1x_i\rvert$	$[-10,10]$	0	1×10^{-15}
$f_7(x)=\sum\limits_{i=1}^{n}\lvert x_i\rvert^{i+1}$	$[-10,10]$	0	1×10^{-50}
$f_8(x)=\sum\limits_{i=1}^{n}(10^6)^{(i-1)/(n-1)}x_i^2$	$[-100,100]$	0	1×10^{-18}
$f_9(x)=\sum\limits_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]$	0	1×10^{-30}
$f_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum\limits_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum\limits_{i=1}^{n}\cos(2\pi x_i))+20+e$	$[-32,32]$	0	1×10^{-13}
$f_{11}(x)=\frac{1}{4000}\sum\limits_{i=1}^{n}x_i^2-\prod\limits_{i=1}^{n}\cos\frac{x_i}{\sqrt{i}}+1$	$[-600,600]$	0	1×10^{-30}
$f_{12}(x)=(\sum\limits_{i=1}^{n}x_i^2)^{0.25}[\sin^2(50(\sum\limits_{i=1}^{n}x_i^2)^{0.1})+1]$	$[-100,100]$	0	1×10^{-7}
$f_{13}(x)=\sum\limits_{i=1}^{n}(y_i^2-10(\cos(2\pi y_i))+10),\ y_i=\begin{cases}x_i, & \lvert x_i\rvert<\frac{1}{2}\\ \frac{\mathrm{round}(2x_i)}{2}, & \lvert x_i\rvert\geqslant\frac{1}{2}\end{cases}$	$[-5.12,5.12]$	0	1×10^{-30}
$f_{14}(x)=1+0.1\sqrt{\sum\limits_{i=1}^{n}x_i^2}-\cos(2\pi\sqrt{\sum\limits_{i=1}^{n}x_i^2})$	$[-100,100]$	0	1×10^{-14}
$f_{15}(x)=\sum\limits_{i=1}^{n/4}[(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-2x_{4i-1})^4+10(x_{4i-3}+x_{4i})^4]$	$[-5,5]$	0	1×10^{-27}
$f_{16}(x)=\sum\limits_{i=1}^{n}x_i^2+(\sum\limits_{i=1}^{n}(0.5ix_i))^2+(\sum\limits_{i=1}^{n}(0.5ix_i))^4$	$[-5,10]$	0	1×10^{-26}

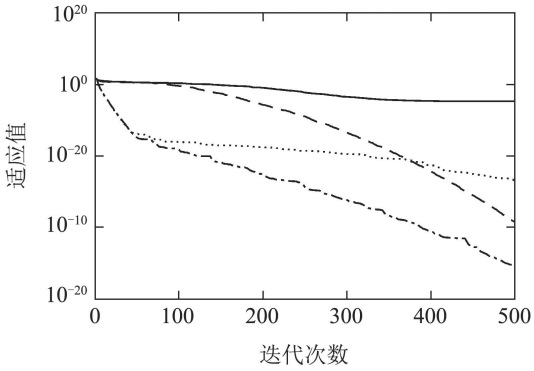
由图1可以看出, 在经过500次迭代之后, RSSA, DSSA和RDSSA都能接近测试函数的理论极值, 且收敛速度和收敛精度都比传统SSA要高. RSSA在200次迭代之后, 收敛速度明显变快, 尤其是对于多峰测试函数, 在避免陷入局部极值、后期迭代速度方面相较于SSA优势明显. DSSA在前期迭代速度快, 后期迭代优化稳定性强, 尤其在测试函数f_7, f_{11}和f_{13}上, 获得明显优于SSA的准确性和高效性. 综合来看, RDSSA具有最快的收敛速度和最精确的收敛精度, 说明RDSSA具有更好的寻优性能.



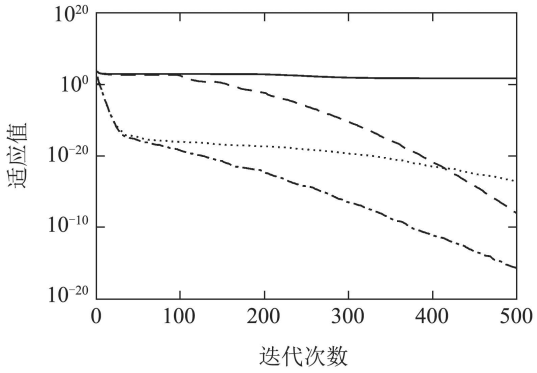
(a) f_1



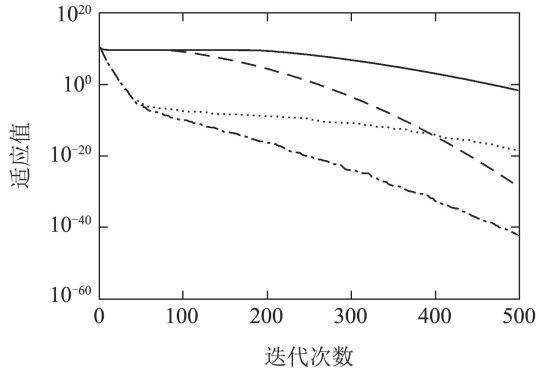
(b) f_2



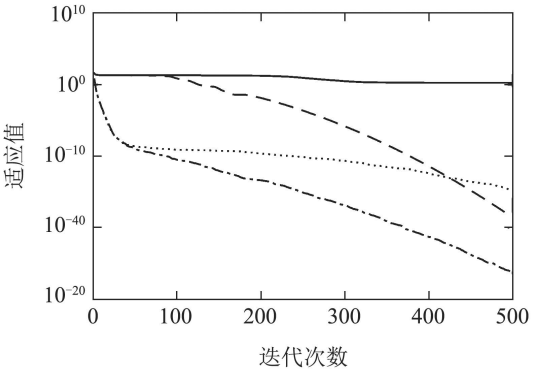
(c) f_3



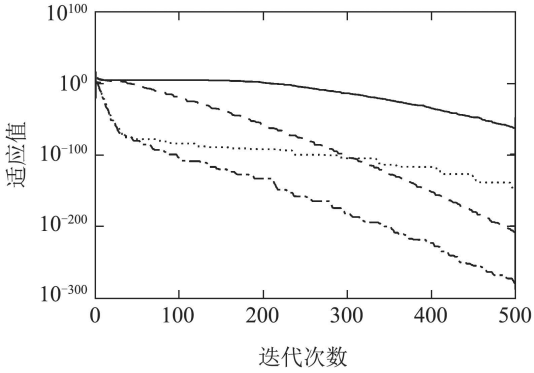
(d) f_4



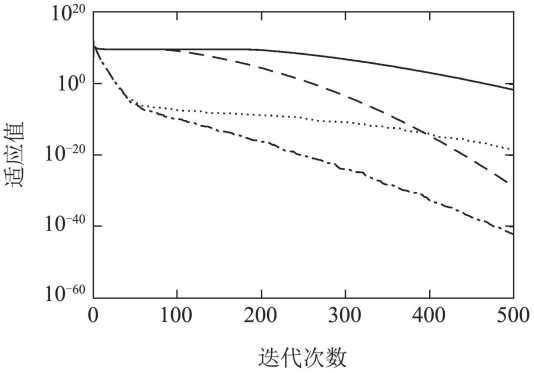
(e) f_5



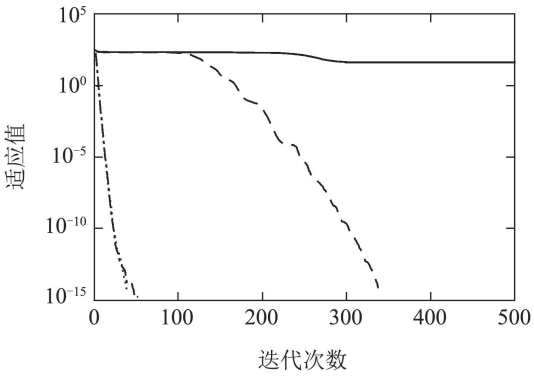
(f) f_6



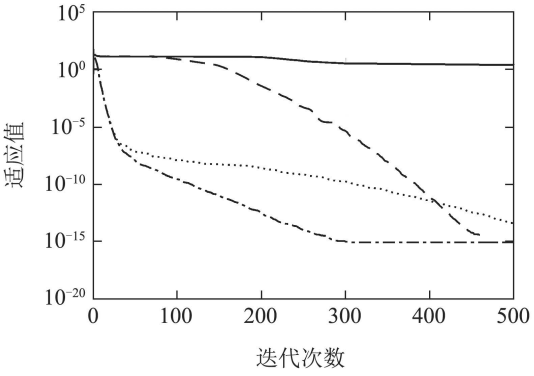
(g) f_7



(h) f_8



(i) f_9



(j) f_{10}

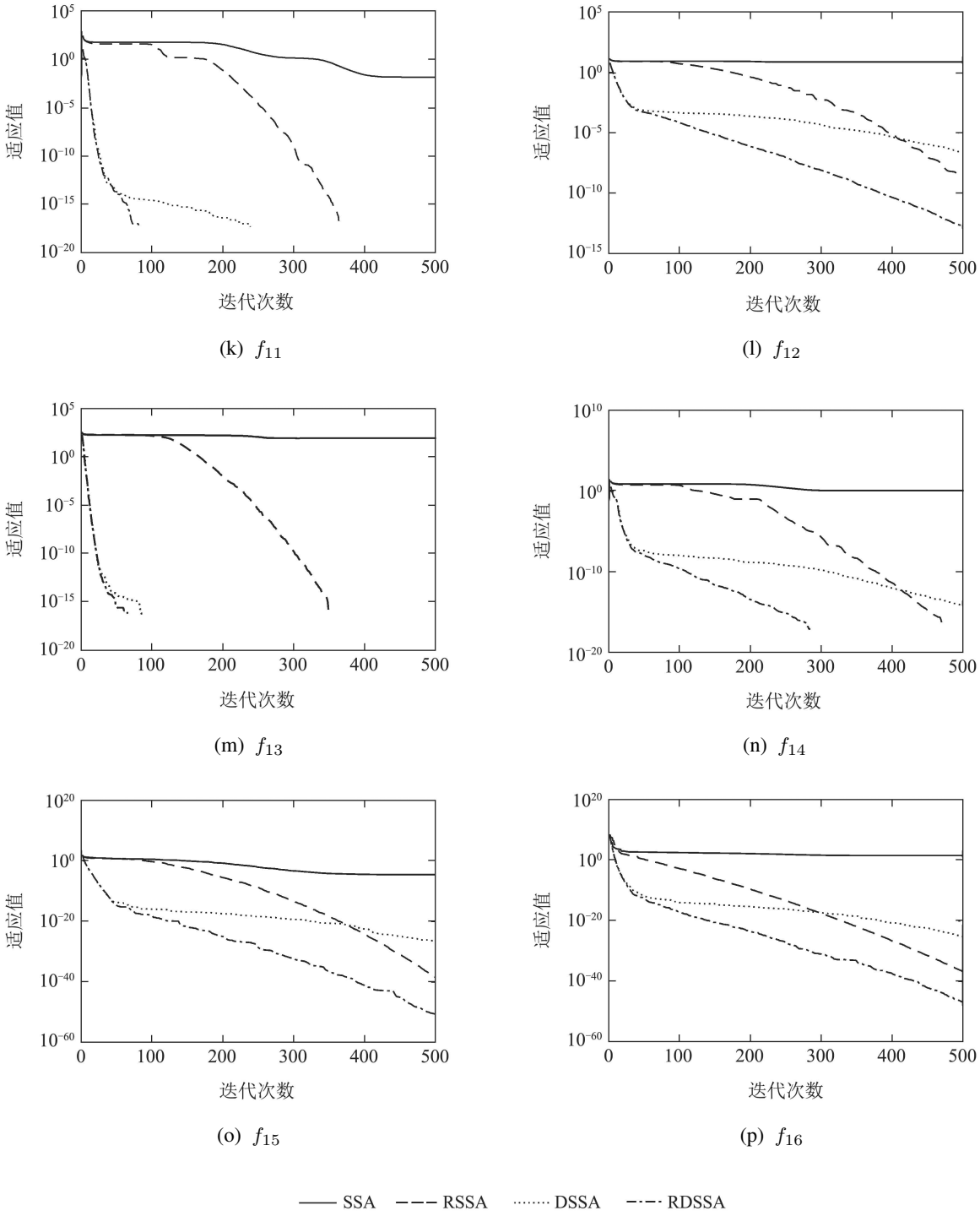


图 1　固定迭代次数下的收敛曲线

Fig. 1　Convergence curves in fixed iteration

表2显示了4种算法对16个测试函数在固定迭代次数下的收敛精度结果比较. 计算得到适应值的最大值(Max)、最小值(Min)、平均值(Ave)和标准差(Std), 分别表示算法所能达到的求解精确度的最差值、最优值、平均值和求解稳定性. 由于本实验所用的计算机能表示的最小浮点数为2.2251E–308, 当收敛精度达到E–308量级时, 结果显示为数值0. p-value表示SSA分别与RSSA, DSSA和RDSSA的Wilcoxon秩和检验结果, 体现算法间的显著性差异, 当p-value的值小于0.05时, 说明改进算法效果明显优于SSA.

通过分析表2数据, RSSA与DSSA的收敛精度相比较于SSA都有改善, 除了f_{10}, RDSSA的优化效果基本与RSSA持平, 其余测试函数RDSSA的收敛精度最高.

对于多峰测试函数f_9, f_{11}, f_{13}和f_{14}, RDSSA的收敛精度远超SSA. p-value值基本都小于0.05, 说明RDSSA比SSA有明显优越性.

表 2 固定迭代次数下的收敛精度(30维)

Table 2 Convergence accuracy in fixed iteration (30 dimensions)

函数	算法	Max	Min	Ave	Std	p-value
f_1	SSA	1.09283E–09	3.63345E–10	7.75733E–10	1.95589E–10	—
	RSSA	1.03031E–35	1.67069E–37	2.40596E–36	2.51896E–36	<0.05
	DSSA	5.36666E–27	1.39616E–29	1.54154E–27	1.68085E–27	<0.05
	RDSSA	**2.03645E–50**	**2.32205E–54**	**1.09011E–51**	**3.72291E–51**	<0.05
f_2	SSA	5.17320E+00	2.23125E–02	9.80966E–01	1.13416E+00	—
	RSSA	1.70786E–18	3.82217E–19	9.04822E–19	3.99585E–19	<0.05
	DSSA	5.14813E–14	1.09341E–15	1.48814E–14	1.42526E–14	<0.05
	RDSSA	**1.11112E–25**	**5.09600E–28**	**1.60705E–26**	**2.20753E–26**	<0.05
f_3	SSA	1.39929E+03	1.17494E+02	5.03150E+02	3.23925E+02	—
	RSSA	1.68287E–34	1.37837E–36	3.37668E–35	3.34282E–35	<0.05
	DSSA	7.82656E–24	5.88939E–27	1.67506E–24	2.35236E–24	<0.05
	RDSSA	**1.28840E–47**	**6.45989E–52**	**1.42996E–48**	**2.81908E–48**	<0.05
f_4	SSA	1.04805E+01	3.45091E–01	6.59604E+00	2.37580E+00	—
	RSSA	1.89953E–18	3.03118E–19	8.54312E–19	4.74942E–19	<0.05
	DSSA	6.03764E–14	4.21933E–15	2.30153E–14	1.78086E–14	<0.05
	RDSSA	**1.20831E–25**	**4.34987E–28**	**1.78016E–26**	**2.48387E–26**	<0.05
f_5	SSA	3.70941E+01	1.27487E–04	5.02205E+00	7.81307E+00	—
	RSSA	7.61201E–35	2.58154E–36	1.66190E–35	1.98275E–35	<0.05
	DSSA	9.44972E–24	3.63357E–27	8.41187E–25	2.00463E–24	<0.05
	RDSSA	**2.38185E–47**	**1.61540E–51**	**2.13280E–48**	**4.67995E–48**	<0.05
f_6	SSA	4.65966E+00	3.58059E–01	1.80069E+00	1.15264E+00	—
	RSSA	2.81871E–19	2.05388E–19	2.48098E–19	1.67150E–20	<0.05
	DSSA	4.87169E–15	1.45438E–16	1.50540E–15	1.34735E–15	<0.05
	RDSSA	**2.52668E–26**	**9.88207E–29**	**5.06126E–27**	**6.10721E–27**	<0.05
f_7	SSA	1.48619E–10	2.19192E–33	9.13616E–12	3.02443E–11	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_8	SSA	3.47160E–02	1.34726E–02	2.12088E–02	6.36467E–03	—
	RSSA	2.27624E–29	7.75099E–30	1.46615E–29	3.74767E–30	<0.05
	DSSA	5.09441E–18	1.11707E–22	4.00784E–19	9.73408E–19	<0.05
	RDSSA	1.15023E–41	6.67160E–46	6.59224E–43	2.14968E–42	<0.05
f_9	SSA	7.16369E+01	1.39294E+01	4.22525E+01	1.54043E+01	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{10}	SSA	3.46197E+00	2.76348E–05	1.93652E+00	6.77444E–01	—
	RSSA	9.85849E–16	3.23398E–16	8.45288E–16	8.12093E–16	<0.05
	DSSA	2.10498E–13	4.44089E–15	3.94943E–14	3.86232E–14	<0.05
	RDSSA	**9.88178E–16**	**1.83865E–16**	**8.03178E–16**	**8.02393E–16**	<0.05
f_{11}	SSA	3.69772E–02	6.23011E–06	1.12365E–02	1.00337E–02	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05

(转下页)

(接上页)

f_{12}	SSA	9.39866E+00	6.18973E+00	7.66468E+00	7.98358E–01	—
	RSSA	2.38347E–09	1.83234E–09	2.20125E–09	1.23567E–10	<0.05
	DSSA	4.99328E–07	3.15487E–08	2.22338E–07	1.13229E–07	<0.05
	RDSSA	**5.04621E–13**	**5.19400E–14**	**1.87431E–13**	**1.01715E–13**	<0.05
f_{13}	SSA	1.64000E+02	1.90000E+01	8.32667E+01	3.30609E+01	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{14}	SSA	1.49987E+00	8.99873E–01	1.19987E+00	1.59741E–01	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	2.70894E–14	4.44089E–16	6.20245E–15	6.61599E–15	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{15}	SSA	1.40136E–04	7.21057E–09	2.38257E–05	3.50959E–05	—
	RSSA	1.19618E–38	5.71396E–41	2.81336E–39	3.45145E–39	<0.05
	DSSA	1.46086E–26	1.65616E–30	1.82191E–27	3.34758E–27	<0.05
	RDSSA	**3.86341E–50**	**1.64913E–54**	**2.13009E–51**	**7.45047E–51**	<0.05
f_{16}	SSA	1.18974E+02	1.22554E+01	5.78362E+01	2.47261E+01	—
	RSSA	4.52155E–37	1.85755E–38	2.09254E–37	1.15348E–37	<0.05
	DSSA	1.54162E–24	2.74832E–28	8.85276E–26	2.86400E–25	<0.05
	RDSSA	**6.03575E–47**	**4.33179E–50**	**1.21718E–47**	**1.76460E–47**	<0.05

为了进一步验证RDSSA在解决复杂的高维问题的优化性能, 本文增加了45维和60维的测试函数的实验, 观察固定迭代次数下的收敛精度. 实验结果如表3和表4所示. 通过分析表3和表4的数据, RSSA, DSSA 和 RDSSA在 45 维和 60 维的收敛精度均高于SSA, 充分验证了RDSSA解决高维问题的优势.

表 3 固定迭代次数下的收敛精度(45维)

Table 3 Convergence accuracy in fixed iteration (45 dimensions)

函数	算法	Max	Min	Ave	Std	p-value
f_1	SSA	3.24116E–04	6.48165E–06	9.28166E–05	1.18593E–04	—
	RSSA	4.04886E–35	7.44707E–37	9.12060E–36	1.29600E–35	<0.05
	DSSA	1.65407E–26	2.38024E–29	3.09573E–27	5.32886E–27	<0.05
	RDSSA	**3.40549E–50**	**6.95530E–53**	**9.82941E–51**	**1.26556E–50**	<0.05
f_2	SSA	5.15598E+00	1.04527E+00	2.83308E+00	1.52066E+00	—
	RSSA	3.46484E–18	6.85478E–19	1.57314E–18	9.20073E–19	<0.05
	DSSA	9.51721E–14	7.18543E–15	2.72815E–14	2.87014E–14	<0.05
	RDSSA	**1.63655E–25**	**2.15606E–27**	**2.83451E–26**	**4.84893E–26**	<0.05
f_3	SSA	8.86101E+03	9.19686E+02	4.16480E+03	2.32448E+03	—
	RSSA	2.08963E–34	1.83525E–35	9.18265E–35	5.59463E–35	<0.05
	DSSA	1.98383E–23	1.97640E–26	4.53617E–24	6.35640E–24	<0.05
	RDSSA	**3.40166E–48**	**1.99343E–50**	**1.18864E–48**	**1.15230E–48**	<0.05
f_4	SSA	2.18689E+01	9.85777E+00	1.41353E+01	3.77734E+00	—
	RSSA	1.93247E–18	2.66919E–19	9.60554E–19	5.77461E–19	<0.05
	DSSA	3.39544E–14	3.20433E–15	1.79009E–14	1.22464E–14	<0.05
	RDSSA	**2.43232E–26**	**1.09288E–27**	**1.02289E–26**	**8.17594E–27**	<0.05
f_5	SSA	2.45099E+02	2.75518E+00	6.58921E+01	6.69379E+01	—
	RSSA	3.05234E–34	7.47423E–36	6.65283E–35	9.40552E–35	<0.05
	DSSA	2.63770E–24	2.17390E–26	1.04172E–24	1.09243E–24	<0.05
	RDSSA	**7.78317E–48**	**1.22612E–50**	**1.54639E–48**	**2.36580E–48**	<0.05
f_6	SSA	6.20447E+00	2.75622E+00	3.90452E+00	1.16871E+00	—
	RSSA	4.11839E–19	3.51849E–19	3.87285E–19	2.27872E–20	<0.05
	DSSA	1.17354E–14	1.08992E–16	2.72086E–15	3.52970E–15	<0.05
	RDSSA	**5.74654E–26**	**1.17472E–27**	**1.04006E–26**	**1.73290E–26**	<0.05

(转下页)

(接上页)

f_7	SSA	1.64285E+07	1.25389E–14	1.64286E+06	5.19516E+06	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_8	SSA	2.02941E+02	5.34473E+00	4.85142E+01	5.80429E+01	—
	RSSA	2.07250E–29	1.12127E–29	1.56494E–29	2.79949E–30	<0.05
	DSSA	3.27321E–18	2.05669E–21	6.52918E–19	1.07433E–18	<0.05
	RDSSA	**2.64936E–41**	**3.67500E–45**	**2.89917E–42**	**8.30711E–42**	<0.05
f_9	SSA	8.05916E+01	5.07429E+01	6.06925E+01	9.30991E+00	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{10}	SSA	3.85283E+00	2.04965E+00	2.86317E+00	6.13707E–01	—
	RSSA	**8.88178E–16**	**8.88178E–16**	**8.88178E–16**	**0**	<0.05
	DSSA	2.28262E–13	7.99361E–15	5.20473E–14	6.79303E–14	<0.05
	RDSSA	**8.88178E–16**	**8.88178E–16**	**8.88178E–16**	**0**	<0.05
f_{11}	SSA	1.21256E–01	1.52281E–02	5.32092E–02	3.82977E–02	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{12}	SSA	1.00128E+01	7.69401E+00	9.17602E+00	7.34645E–01	—
	RSSA	2.61659E–09	2.28782E–09	2.38779E–09	9.50559E–11	<0.05
	DSSA	4.58243E–07	1.31942E–07	2.66200E–07	9.43361E–08	<0.05
	RDSSA	**2.33000E–13**	**5.99441E–14**	**1.30645E–13**	**6.05739E–14**	<0.05
f_{13}	SSA	2.56500E+02	8.85000E+01	1.65725E+02	5.13358E+01	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{14}	SSA	3.39987E+00	1.89987E+00	2.62987E+00	4.87739E–01	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	1.33227E–14	1.33227E–15	5.70655E–15	3.89834E–15	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{15}	SSA	9.32962E–05	1.71292E–07	1.78475E–05	2.89692E–05	—
	RSSA	3.73352E–39	1.02293E–40	1.28563E–39	1.16681E–39	<0.05
	DSSA	2.96460E–26	2.63897E–29	3.20450E–27	9.29542E–27	<0.05
	RDSSA	**2.75264E–50**	**1.44565E–53**	**5.49712E–51**	**9.35976E–51**	<0.05
f_{16}	SSA	4.14691E+02	2.01067E+02	2.80063E+02	6.87546E+01	—
	RSSA	1.02172E–36	1.31153E–37	4.62371E–37	2.57349E–37	<0.05
	DSSA	5.57029E–25	1.51263E–27	9.99033E–26	1.71140E–25	<0.05
	RDSSA	**3.68542E–47**	**1.08136E–49**	**6.76808E–48**	**1.13657E–47**	<0.05

表 4 固定迭代次数下的收敛精度(60维)

Table 4 Convergence accuracy in fixed iteration (60 dimensions)

函数	算法	Max	Min	Ave	Std	p-value
f_1	SSA	6.47223E–01	1.45892E–01	3.29798E–01	1.61166E–01	—
	RSSA	3.35707E–35	1.28161E–36	6.54384E–36	9.96320E–36	<0.05
	DSSA	1.35664E–25	1.08830E–29	1.77669E–26	4.16852E–26	<0.05
	RDSSA	**1.26935E–50**	**4.46620E–52**	**4.62738E–51**	**4.49486E–51**	<0.05

(转下页)

(接上页)

f_2	SSA	1.33479E+01	2.39358E+00	8.10118E+00	3.41768E+00	—
	RSSA	4.90944E–18	8.28657E–19	2.54740E–18	1.45007E–18	<0.05
	DSSA	1.49660E–13	6.21612E–15	4.65005E–14	4.56258E–14	<0.05
	RDSSA	**2.68574E–26**	**1.21613E–27**	**9.88921E–27**	**7.84865E–27**	<0.05
f_3	SSA	9.76476E+03	3.42358E+03	5.66548E+03	2.00469E+03	—
	RSSA	2.07190E–34	1.00475E–35	1.08954E–34	6.75311E–35	<0.05
	DSSA	5.91364E–23	1.41508E–25	1.22558E–23	1.84580E–23	<0.05
	RDSSA	**1.02326E–46**	**1.26753E–49**	**2.25493E–47**	**3.49603E–47**	<0.05
f_4	SSA	1.94542E+01	1.34816E+01	1.67569E+01	2.09411E+00	—
	RSSA	1.97851E–18	3.37756E–19	1.10637E–18	6.07419E–19	<0.05
	DSSA	2.94685E–14	6.05789E–16	1.07823E–14	8.80814E–15	<0.05
	RDSSA	**2.28490E–25**	**9.09491E–28**	**4.18913E–26**	**7.08808E–26**	<0.05
f_5	SSA	2.35954E+03	9.73046E+01	9.27773E+02	7.03066E+02	—
	RSSA	5.10839E–34	2.23230E–35	2.39517E–34	1.89494E–34	<0.05
	DSSA	1.44444E–23	2.98983E–26	5.10997E–24	5.19037E–24	<0.05
	RDSSA	**1.51311E–47**	**1.26065E–49**	**4.37077E–48**	**5.29876E–48**	<0.05
f_6	SSA	1.30698E+01	3.18395E+00	9.27620E+00	3.03972E+00	—
	RSSA	5.73325E–19	5.03685E–19	5.29915E–19	2.16139E–20	<0.05
	DSSA	8.95233E–15	3.94691E–16	3.15633E–15	2.59359E–15	<0.05
	RDSSA	**4.21249E–26**	**7.92059E–28**	**1.61535E–26**	**1.40685E–26**	<0.05
f_7	SSA	4.44784E+12	5.99030E–04	5.59002E+11	1.40656E+12	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_8	SSA	1.91217E+06	3.00481E+04	3.36328E+05	5.69384E+05	—
	RSSA	2.87060E–29	1.98982E–29	2.39295E–29	3.01237E–30	<0.05
	DSSA	5.31203E–18	4.11957E–21	8.42013E–19	1.73120E–18	<0.05
	RDSSA	**1.69321E–42**	**3.60338E–45**	**3.12237E–43**	**5.02509E–43**	<0.05
f_9	SSA	1.13721E+02	3.49816E+01	7.22400E+01	2.64602E+01	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{10}	SSA	5.05185E+00	3.53123E+00	4.24317E+00	4.12921E–01	—
	RSSA	**8.88178E–16**	**8.88178E–16**	**8.88178E–16**	**0**	<0.05
	DSSA	9.68114E–14	4.44089E–15	4.24549E–14	2.88648E–14	<0.05
	RDSSA	**8.88178E–16**	**8.88178E–16**	**8.88178E–16**	**0**	<0.05
f_{11}	SSA	5.72231E–01	1.78561E–01	3.57856E–01	1.30251E–01	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{12}	SSA	1.06505E+01	1.00128E+01	1.03954E+01	3.29273E–01	—
	RSSA	2.71044E–09	2.45164E–09	2.55609E–09	7.40141E–11	<0.05
	DSSA	5.85522E–07	8.20708E–08	2.64348E–07	1.53139E–07	<0.05
	RDSSA	**3.87803E–13**	**4.79570E–14**	**2.06689E–13**	**1.12959E–13**	<0.05
f_{13}	SSA	3.46000E+02	5.20000E+01	2.19000E+02	1.11687E+02	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	**0**	**0**	**0**	**0**	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05
f_{14}	SSA	4.79987E+00	3.19987E+00	4.00987E+00	5.64604E–01	—
	RSSA	**0**	**0**	**0**	**0**	<0.05
	DSSA	1.50990E–14	2.66454E–15	6.92779E–15	3.75774E–15	<0.05
	RDSSA	**0**	**0**	**0**	**0**	<0.05

(转下页)

(接上页)

f_{15}	SSA	4.40758E–05	7.79350E–08	1.05086E–05	1.64670E–05	—
	RSSA	2.22816E–39	1.89801E–41	8.92821E–40	6.89290E–40	<0.05
	DSSA	2.27712E–27	1.31084E–31	6.92293E–28	9.13056E–28	<0.05
	RDSSA	**1.09613E–50**	**1.98638E–53**	**1.23639E–51**	**3.41835E–51**	<0.05
f_{16}	SSA	8.09641E+02	4.34131E+02	6.56401E+02	1.26222E+02	—
	RSSA	1.19653E–36	3.50322E–37	7.70076E–37	2.45936E–37	<0.05
	DSSA	6.66595E–25	3.00436E–27	2.16233E–25	2.09271E–25	<0.05
	RDSSA	**2.66392E–45**	**3.75113E–49**	**3.43910E–46**	**8.25073E–46**	<0.05

4.2.2 固定目标精度下的迭代次数

固定目标精度下的实验结果如表5所示.

表 5 固定目标精度下的实验结果

Table 5 Test results in fixed precision

函数	统计结果	SSA	RSSA	DSSA	RDSSA
f_1	Ave	500	416.1	486.933	**229.933**
	SR	0	**1**	0.00667	**1**
f_2	Ave	500	423.6	495.267	**243.567**
	SR	0	**1**	0.433	**1**
f_3	Ave	500	395.867	495.7	**221.3**
	SR	0	**1**	0.433	**1**
f_4	Ave	500	420.767	495.633	**236.833**
	SR	0	**1**	0.433	**1**
f_5	Ave	500	405.467	496.567	**228.867**
	SR	0	**1**	0.2	**1**
f_6	Ave	500	431.9	495.267	**247.033**
	SR	0	**1**	0.367	**1**
f_7	Ave	500	85.9	**7.9**	7.96667
	SR	0	**1**	**1**	**1**
f_8	Ave	500	393.4	475.333	**214.267**
	SR	0	**1**	0.833	**1**
f_9	Ave	500	264.933	**32.5**	**32.5**
	SR	0	**1**	**1**	**1**
f_{10}	Ave	500	396.633	476	**212.267**
	SR	0	**1**	0.9	**1**
f_{11}	Ave	500	301.267	137.033	**61**
	SR	0	**1**	**1**	**1**
f_{12}	Ave	500	428.667	499.5	**242.767**
	SR	0	**1**	0.1	**1**
f_{13}	Ave	500	277.567	51.3	**39.6**
	SR	0	**1**	**1**	**1**
f_{14}	Ave	500	408.3	481.233	**212.867**
	SR	0	**1**	0.8333	**1**
f_{15}	Ave	500	380.4	478.267	**220.3**
	SR	0	**1**	0.7	**1**
f_{16}	Ave	500	395.333	490.867	**225.533**
	SR	0	**1**	0.467	**1**

由于测试函数复杂度不同, 设置的目标精度也不同, 具体数值设置如表1所示, 函数维数D为30. 表5中Ave表示算法30次独立运行能够达到目标精度的平均迭代次数, 如果某次独立运行500次迭代后仍未达到目标精度, 则取该次迭代次数为500. SR表示算法能在500次迭代内达到目标精度的成功率. 2 通过表5数据可知, SSA的成功率基本为0, 说明其优化能力有待提高. 对于测试函数f_9~f_{11}和f_{13}~f_{15}, DSSA的成功率均高于70%, 说明DSSA对于多峰优化问题的寻优性能更好. RSSA的成功率均为100%, 说明RSSA具有较快的收敛速度. RDSSA不仅成功率均为100%, 且平均迭代次数远远小于RSSA, 均未超过250, 说明RDSSA具有更快的收敛速度和更好的优化效果.

4.2.3 与其他算法比较

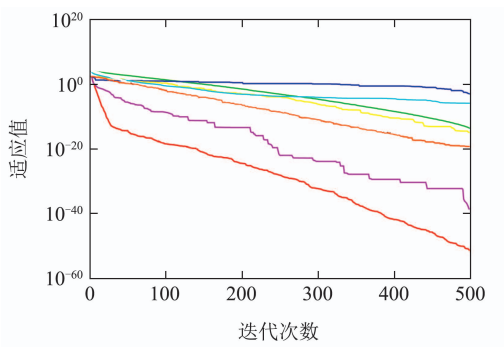
为了进一步验证RDSSA的优化性能, 将其与增强的樽海鞘群算法(enhanced salp swarm algorithm, ESSA)[14]、混沌的樽海鞘群算法(chaotic salp swarm algorithm, CSSA)[12]、飞蛾火焰算法(moth-flame optimization algorithm, MFO)[17]、多重宇宙优化(multi-verse optimizer, MVO)[18]算法、神经网络算法(neural network algorithm, NNA)[19]、粒子群优化(particle swarm optimization, PSO)[20]算法、人工蜂群(artificial bee colony, ABC)[21]算法和差分进化(differential evolution, DE)[22]算法进行比较. 其中, ESSA和CSSA是SSA的改进算法, MFO, MVO和NNA是近3年被提出的新颖的全局优化算法, PSO, ABC和DE是效果较好的经典智能优化算法.为了遵循实验的公平性原则, 种群数量、函数维数、迭代次数和独立运行次数等基本实验参数与文献[12]保持一致, 文献[22]中的参数F设置为[0.2, 0.9]的随机数, CR设置为0.2, 其余文献中的参数均与原文献保持一致. 表6中CSSA数据来自文献[12]. 实验结果如表6和图2所示.

表6中Ave和Std分别代表计算得到的适应值的平均值和方差, 是评判算法收敛精度和稳定性的指标. 由表6可以看出, 除了f_9~f_{11}和f_{13}, RDSSA和ESSA的收敛精度和收敛稳定性相当, 在其余测试函数上, RDSSA都表现出了比其他算法更优秀的收敛能力, 特别是f_7和f_{14}, RDSSA的收敛精度和收敛稳定性远超于其他算法.

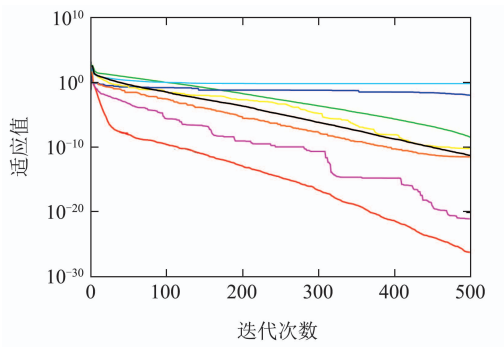
表 6 与其他算法的收敛精度比较

Table 6 Comparison of convergence accuracy with other algorithms

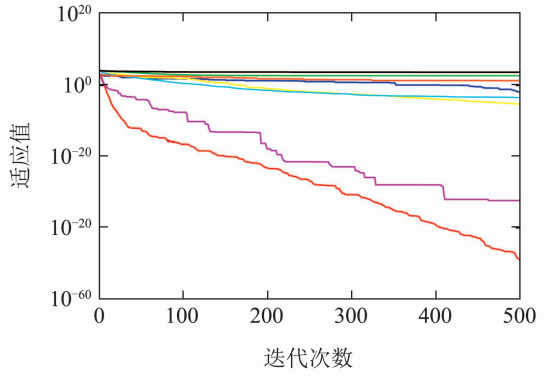
函数	统计结果	RDSSA	ESSA	CSSA	MFO	MVO	NNA	PSO	ABC	DE
f_1	Ave	**3.56E–52**	4.55E–39	1.40E–09	3.38E–14	1.16E–03	1.92E–15	1.88E–06	1.17E–16	5.86E–17
	Std	**8.40E–52**	1.81E–38	3.31E–10	3.99E–14	1.64E–03	1.05E–14	1.03E–05	9.24E–20	6.26E–20
f_2	Ave	**6.49E–27**	1.02E–21	4.14E–06	3.47E–09	1.09E–02	6.73E–11	8.34E–01	5.48E–12	3.48E–12
	Std	**7.20E–27**	4.86E–21	1.68E–06	2.82E–09	3.94E–03	1.29E–10	6.03E–01	3.68E–12	7.24E–13
f_3	Ave	**1.01E–49**	4.13E–33	5.40E–10	1.01E–01	9.57E–03	6.11E–06	3.52E–04	4.66E+01	2.05E+01
	Std	**2.28E–49**	1.31E–32	1.94E–10	3.16E–01	8.24E–03	1.19E–05	6.64E–04	1.71E+01	3.59E+00
f_4	Ave	**8.47E–27**	5.02E–19	1.08E–05	9.28E–03	2.74E–02	3.72E–04	3.03E+00	3.44E+01	2.09E–03
	Std	**5.66E–27**	1.59E–18	1.55E–06	1.60E–02	1.13E–02	3.35E–04	3.10E+00	9.91E+00	7.29E–04
f_5	Ave	**1.44E–50**	1.28E–42	1.77E–10	2.90E–14	1.07E–03	8.68E–16	9.44E–04	1.43E+02	2.28E–20
	Std	**2.19E–50**	4.03E–42	3.89E–11	4.82E–14	7.65E–04	2.74E–15	2.92E–03	6.38E+01	1.30E–20
f_6	Ave	**2.90E–27**	1.13E–24	—	1.17E–09	6.50E–02	7.47E–06	7.58E–01	5.76E+00	8.37E–06
	Std	**3.65E–27**	3.56E–24	—	2.09E–09	3.82E–02	1.27E–05	4.25E–01	2.23E+00	1.72E–05
f_7	Ave	**1.30E–276**	4.27E–224	—	1.76E–52	7.80E–25	2.03E–61	2.12E–13	4.45E+06	4.45E–97
	Std	**0**	**0**	—	5.16E–52	2.46E–24	6.43E–61	6.71E–13	4.45E+06	6.84E–97
f_8	Ave	**1.41E–43**	9.30E–30	—	2.33E–08	2.56E+03	5.34E–09	3.92E–02	4.02E+09	4.11E–14
	Std	**1.36E–43**	2.94E–29	—	3.91E–08	2.71E+03	1.69E–08	7.89E–02	1.90E+09	2.28E–15
f_9	Ave	**0**	**0**	1.30E–10	1.85E+01	2.39E+02	8.50E+01	2.59E+02	3.71E+01	3.12E+01
	Std	**0**	**0**	2.26E–10	9.71E+00	1.23E+00	2.17E+01	3.46E+01	3.80E+01	1.12E+01
f_{10}	Ave	**8.88E–16**	9.17E–16	7.03E–06	6.72E–08	1.46E–02	4.84E–09	3.17E+00	1.62E+01	1.09E–10
	Std	**0**	**0**	2.41E–06	5.22E–08	8.39E–03	2.54E–08	1.43E+00	2.57E+00	3.38E–11
f_{11}	Ave	**0**	**0**	1.64E–09	1.91E–01	3.49E–01	9.67E–02	2.62E–01	4.11E+01	1.30E+01
	Std	**0**	**0**	8.19E–10	1.06E–01	2.29E–03	5.40E–02	1.52E–01	1.30E+01	6.29E–05
f_{12}	Ave	**1.04E–13**	4.25E–10	—	2.20E+00	2.93E–01	1.91E–01	3.93E+00	9.02E+00	7.82E–01
	Std	**5.61E–14**	2.11E–09	—	1.47E+00	8.34E–02	5.30E–02	8.41E–01	6.96E–01	7.47E–02
f_{13}	Ave	**0**	**0**	—	6.60E+00	1.53E+00	1.00E+00	1.87E+01	1.80E+01	1.00E+00
	Std	**0**	**0**	—	1.03E+01	2.08E+00	0.00E+00	8.98E+00	7.48E+00	0.00E+00
f_{14}	Ave	**0**	1.48E–17	—	3.17E–01	1.20E–01	4.07E–02	8.50E–01	7.20E+00	1.03E–01
	Std	**0**	8.11E–17	—	9.50E–02	4.07E–02	4.07E–02	3.25E–01	1.32E+00	1.83E–02
f_{15}	Ave	**1.88E–51**	4.92E–37	—	1.43E–05	3.99E–06	9.38E–07	8.36E–39	3.16E+00	1.27E–06
	Std	**5.05E–51**	2.69E–36	—	1.17E–05	4.45E–06	6.66E–07	2.87E–38	2.15E+00	7.22E–07
f_{16}	Ave	**8.20E–50**	1.11E–34	—	1.50E+01	7.65E–05	5.86E–09	5.42E–02	5.52E+01	2.24E+00
	Std	**1.78E–49**	5.98E–34	—	1.65E+01	9.39E–05	2.85E–08	2.18E–01	1.23E+01	1.07E+00



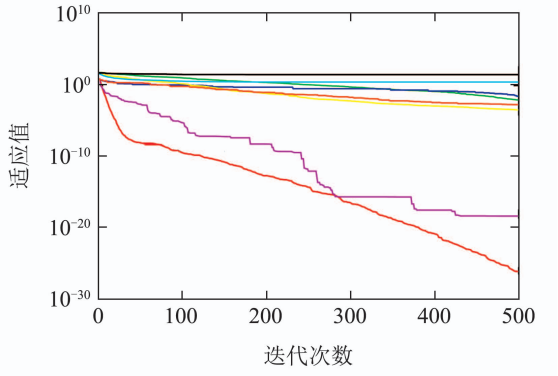
(a) f_1



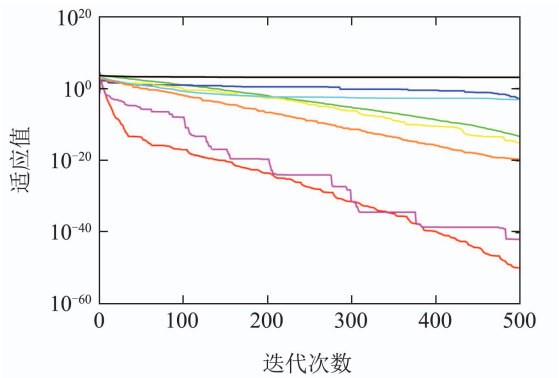
(b) f_2



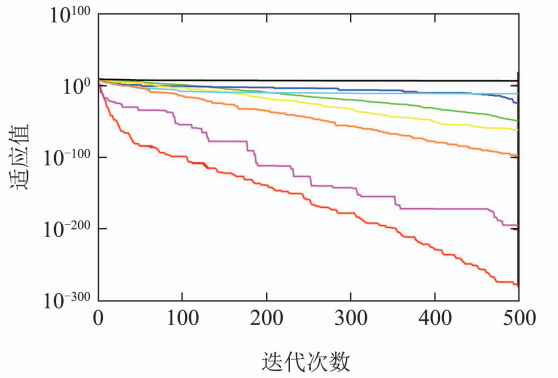
(c) f_3



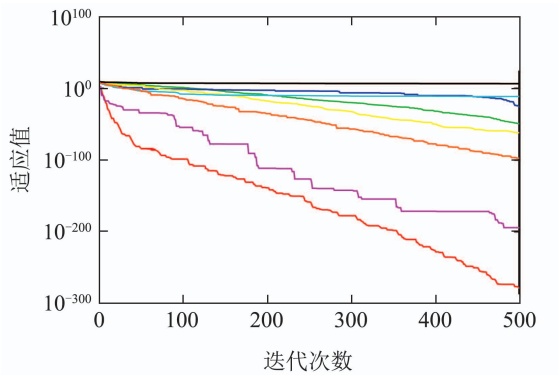
(d) f_4



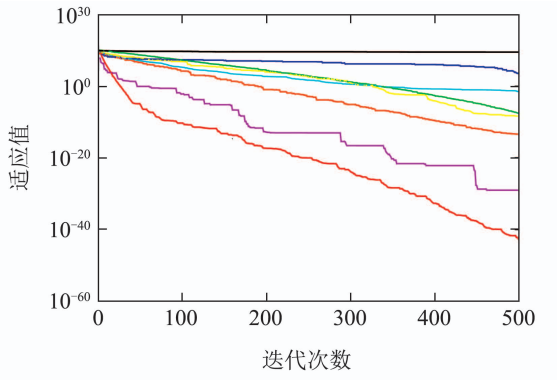
(e) f_5



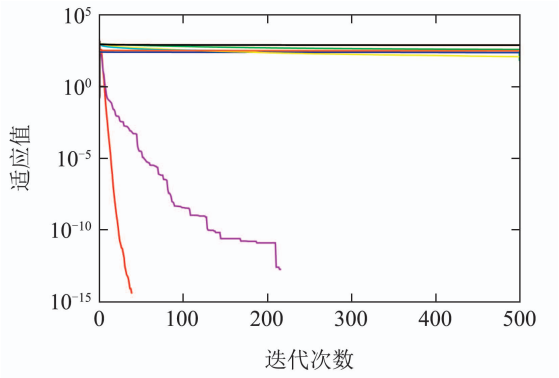
(f) f_6



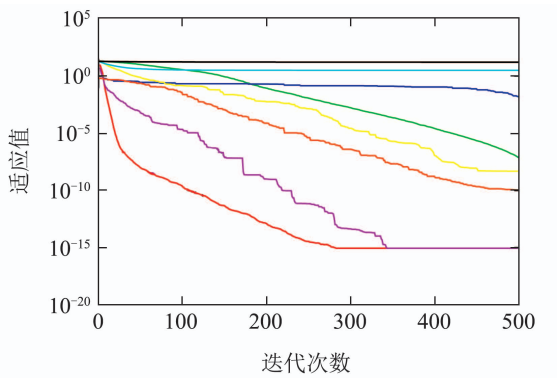
(g) f_7



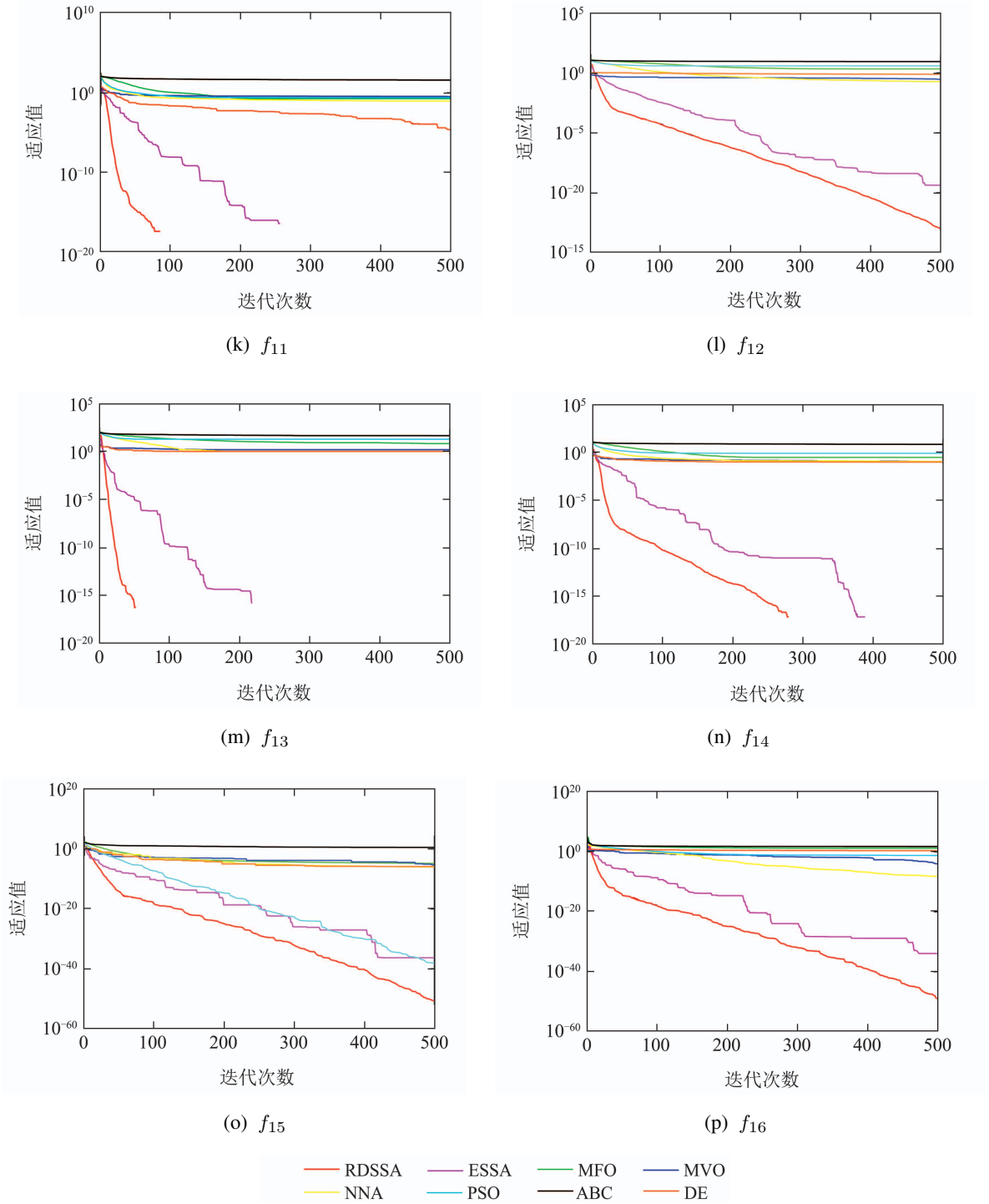
(h) f_8



(i) f_9



(j) f_{10}



(k) f_{11}　(l) f_{12}　(m) f_{13}　(n) f_{14}　(o) f_{15}　(p) f_{16}

图 2　与其他算法收敛曲线比较

Fig. 2　Comparison of convergence curves with other algorithms

图2显示了几种算法的收敛曲线比较, 可以看出对于单峰函数, RDSSA在100次迭代前速度很快, 之后以既快又稳定的速度收敛, 最终达到最好的优化效果, ESSA其次, 也能得到较好的收敛精度, 其余算法收敛速度较慢, 优化效果不佳. 对于多峰函数, 特别是f_9, f_{11}, f_{13}和f_{14}, RDSSA和ESSA的收敛速度优势明显, RDSSA的优化效果最好. 综合来看, RDSSA相比于其他智能优化算法, 具有较快的寻优速度和较好的寻优性能.

5　结论

本文提出了一种基于衰减因子和动态学习的改进樽海鞘群算法, 通过添加衰减因子, 灵活的控制搜索范围大小, 加快了算法收敛速度, 通过引入动态学习策略, 加强了跟随者对于寻优的协助作用, 达到了更高的收敛精度, 改善了SSA的优化性能. 实验结果表

明, RDSSA相较于基本SSA、其他改进SSA和其他智能优化算法, 在收敛精度和收敛速度方面都有较大提升, 在最优化问题领域有着广泛的应用前景.

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. *Advances in Engineering Software*, 2017, 114: 163 – 191.

[2] YANG B, ZHONG L, ZHANG X, et al. Novel bio-inspired memetic salp swarm algorithm and application to MPPT for PV systems considering partial shading condition. *Journal of Cleaner Production*, 2019, 215: 1203 – 1222.

[3] ABBASSI R, ABBASSI A, HEIDARI A A, et al. An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. *Energy Conversion and Management*, 2019, 179: 362 – 372.

[4] YANG Bo, ZHONG Lin'en, ZHU Dena, et al. Modified salp swarm algorithm based maximum power point tracking of power-voltage system under partial shading condition. *Control Theory & Applications*, 2019, 36(3): 339 – 352.
(杨博, 钟林恩, 朱德娜, 等. 部分遮蔽下改进樽海鞘群算法的光伏系统最大功率跟踪. 控制理论与应用, 2019, 36(3): 339 – 352.)

[5] ATEYA A A, MUTHANNA A, VYBORNOVA A, et al. Chaotic salp swarm algorithm for SDN multi-controller networks. *Engineering Science and Technology, an International Journal*, 2019, 22(4): 1001 – 1012.

[6] HEGAZY A E, MAKHLOUF M A, EL-TAWEL G S. Improved salp swarm algorithm for feature selection. *Journal of King Saud University-Computer and Information Sciences*, 2018, 37(1): 72 – 88.

[7] FARIS H, MAFARJA M M, HEIDARI A A, et al. An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. *Knowledge-Based Systems*, 2018, 154: 43 – 67.

[8] IBRAHIM A, AHMED A, HUSSEIN S, et al. Fish image segmentation using salp swarm algorithm. *The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018)*. Berlin, German: Springer, 2018: 42 – 51

[9] LIU Sen, JIA Zhicheng, CHEN Lei, et al. Hyperspectral images unmixing based on nonnegative matrix factorization optimized by salp swarm algorithm. *Journal of Computer-Aided Design and Computer Graphics*, 2019, 31(2): 315 – 323.
(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. 计算机辅助设计与图形学学报, 2019, 31(2): 315 – 323.)

[10] XING Zhikai, JIA Heming, SONG Wenlong. Levy flight trajectory-based salp swarm algorithm for multilevel thresholding image segmentation. *Acta Automatica Sinica*, 2019: 1 – 15, doi: 10.16383/j.aas.c180140
(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. 自动化学报, 2019: 1 – 15, doi: 10.16383/j.aas.c180140)

[11] BAIRATHI D, GOPALANI D. Salp swarm algorithm (SSA) for training feed-forward neural networks. *Soft Computing for Problem Solving*. Singapore: Springer, 2019: 521 – 534.

[12] SAYED G I, KHORIBA G, HAGGAG M H. A novel chaotic salp swarm algorithm for global optimization and feature selection. *Applied Intelligence*. 2018, 48(10): 3462 – 3481.

[13] WANG D Y, ZHOU Y Q, JIANG S Q, et al. A simplex method-based salp swarm algorithm for numerical and engineering optimization. *Intelligent Information Processing*. Berlin, German: Springer, 2018: 189 – 211.

[14] QAIS M H, HASANIEN H M, ALGHUWAINEM S. Enhanced salp swarm algorithm: application to variable speed wind generators. *Engineering Applications of Artificial Intelligence*, 2019, 80: 82 – 96.

[15] IBRAHIM R A, EWEES A A, OLIVA D, et al. Improved salp swarm algorithm based on particle swarm optimization for feature selection. *Journal of Ambient Intelligence and Humanized Computing*, 2018, 10(8): 3155 – 3169.

[16] KHAMEES M, ALBAKRY A, SHAKER K. Multi-objective feature selection: hybrid of salp swarm and simulated annealing approach. *New Trends in Information and Communications Technology Applications*. Berlin, German: Springer, 2018: 129 – 142.

[17] MIRJALILI S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, 2015, 89: 228 – 249.

[18] MIRJALILI S, MIRJALILI S M, HATAMLOU A. multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing & Applications*, 2015, 27(2): 495 – 513.

[19] SADOLLAH A, SAYYAADI H, YADAV A. A dynamic metaheuristic optimization model inspired by biological nervous systems: Neural network algorithm. *Applied Soft Computing*, 2018, 71: 747 – 782.

[20] HART R, KENNEDY J. A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Nagoya, Japan: IEEE, 1995: 39 – 43.

[21] KARABOGA D, BASTURK B. A powerful and efficient algorithm for numerical function optimization : artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 2007, 39(3): 459 – 471.

[22] STORN R, PRICE K. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997, 11(4): 341 – 359.

作者简介:

陈 雷 博士, 教授, 硕士生导师, 主要研究方向为仿生智能计算、高光谱图像处理、盲信号处理, E-mail: chenlei@tjcu.edu.cn;

蔺 悦 硕士研究生, 主要研究方向为仿生智能计算, E-mail: linyue_tju@163.com;

康志龙 副研究员, 主要研究方向为仿生智能计算、高光谱图像处理和半导体光电子学, E-mail: zhlk@hebut.edu.cn.