Pure Mathematics 理论数学, 2023, 13(3), 694-711
Published Online March 2023 in Hans. https://www.hanspub.org/journal/pm
https://doi.org/10.12677/pm.2023.133074

Hans汉斯

基于突变策略的自适应骨干粒子群算法

张嘉文[1]，舒慧生[1*]，阚　秀[2]

[1]东华大学理学院，上海
[2]上海工程技术大学电子电气工程学院，上海

收稿日期：2023年2月23日；录用日期：2023年3月24日；发布日期：2023年3月31日

摘　要

骨干粒子群算法是由标准粒子群算法演变而来的，其在粒子位置更新方面采用了高斯采样策略。针对骨干粒子群算法在解决高维优化问题时存在的易陷入局部最优的问题，文中引入了具有下降趋势的时变因子，提出了一种基于突变策略的带有自适应扰动值的骨干粒子群算法。该算法在高斯分布的均值项中引入两个服从均匀分布的随机数，在高斯分布的标准差中引入了一个自适应扰动值，且给出了突变策略进一步保证粒子收敛到全局最优解。改进后的算法与其他5种粒子群算法在9个经典测试函数上进行仿真实验，结果表明改进的算法在收敛速度和收敛精度方面的综合表现都优于其它算法。

关键词

骨干粒子群算法，自适应扰动，突变策略，时变因子，全局收敛

A Self-Adaptive Bare-Bones Particle Swarm Optimization Algorithm Based on Mutation Strategy

Jiawen Zhang[1], Huisheng Shu[1*], Xiu Kan[2]

[1]College of Science, Donghua University, Shanghai
[2]School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai

Received: Feb. 23rd, 2023; accepted: Mar. 24th, 2023; published: Mar. 31st, 2023

Abstract

The bare-bones particle swarm optimization algorithm is evolved from the standard particle swarm optimization algorithm, which adopts the Gaussian sampling strategy when particles' posi-

*通讯作者。

文章引用: 张嘉文, 舒慧生, 阚秀. 基于突变策略的自适应骨干粒子群算法[J]. 理论数学, 2023, 13(3): 694-711.
DOI: 10.12677/pm.2023.133074

tion update. To improve the problem of premature convergence when solving the high-dimensional optimization problems, a time-varying factor with downward trend is introduced and a self-adaptive mutation bare-bones particle swarm optimization (AMBPSO) is proposed where the particle would mutate according to adaptive probability when it becomes stagnant after updating with the perturbation strategy in order to jump out of the local optimum. The algorithm introduces two random numbers that obey uniform distribution in the mean term of the Gaussian distribution and an adaptive perturbation value in the standard deviation of the Gaussian distribution, and gives a mutation strategy to further ensure that the particles converge to the global optimum. The proposed algorithm and other five particle swarm optimization algorithms are simulated on nine classical benchmark functions, whose experimental results show that the proposed AMBPSO algorithm is more competitive than some existing popular variants of PSO algorithms in terms of convergence speed and convergence accuracy.

Keywords

Bare-Bones Particle Swarm Optimization, Adaptive Disturbance, Mutation Strategy, Time-Varying Factor, Global Convergence

1. 引言

骨干粒子群算法(Bare-bones Particle Swarm Optimization, BBPSO)在2003年首先被Kennedy提出[1]，与标准粒子群算法[2]相比，骨干粒子群算法的位置更新方程中没有速度项，在后续迭代过程中粒子位置更新服从高斯采样策略。由于骨干粒子群算法的简单性和强大的竞争力，它成为了最方便的粒子群算法，并逐渐引起了越来越多的学者的关注。不同学者对于 BBPSO 算法研究的侧重点也各有不同[3]。在理论研究方面，文献[4]运用数学归纳法证明了标准粒子群算法可以转化为骨干粒子群算法，且给出了BBPSO 算法更为通用的形式；文献[5]中提出了一种带有自适应扰动值的 BBPSO，并利用随机过程理论证明了改进算法的收敛性；文献[6]认为获得的保证种群均方收敛到全局最优的参数条件并不适用于所有函数，并提出了利用首达时来分析种群的收敛性。在应用方面，文献[7]通过将改进的骨干粒子群优化算法和被称为定向混沌搜索的局部搜索器相结合，解决了具有阀点效应的动态经济调度问题；由于大多数粒子群算法中采用的传统粒子更新机制和群初始化策略限制了其处理高维特征选择问题的性能，文献[8]提出了一种基于互信息的骨干粒子群算法的特征选择算法；文献[9]利用改进的骨干粒子群算法解决了评论员的分配问题。

虽然骨干粒子群算法在实现过程中省去了调节多个参数这一步骤，更易于实现，但和粒子群算法一样，其在多峰函数上寻优时，仍存在容易停滞、陷入局部最优的问题，导致求解的结果不够理想。为了进一步改善骨干粒子群算法的性能，学者们主要从改进高斯采样的均值或方差部分[10] [11]、引入突变算子[12]、与其他群智能算法结合[13] [14] [15]等方法出发，在提高粒子多样性的基础上，来改善粒子的全局和局部搜索能力。

针对骨干粒子群算法过早收敛，易于陷入局部最优的问题，本文提出了一种搜索中心随机，带有自适应扰动值，且在粒子停滞时依概率发生突变的自适应突变骨干粒子群算法(AMBPSO)。自适应扰动值的大小由每个粒子的收敛程度和种群的多样性共同决定，在扰动值中引入了 sigmoid 函数，使得扰动值最终收敛到 0。根据高斯分布更新后的粒子，将依概率发生突变，突变概率由停滞次数所决定。采用 9

个经典的测试函数对所提出的算法进行验证，并与其他 5 种粒子群算法(PSO, BBPSO, ABPSO, SLBPSO, AWPSO)进行了比较。

本文的其余内容组织如下：第一部分介绍了传统的骨干粒子群算法；第二部分提出了改进的基于自适应突变的骨干粒子群算法(AMBPSO)，并给出了详细的算法流程和参数选择方式；第三部分通过仿真实验验证了提出的算法的有效性；第四部分给出了文章的最终结论及未来展望。

2. 骨干粒子群算法

Clerc 和 Kennedy [16]基于对标准粒子群算法中粒子运行轨迹的研究，得出了粒子以其个体局部最优和群体全局最优位置的加权平均值为中心进行振荡的结论。

根据这一结论，骨干粒子群算法[1]由 Kennedy 于 2003 年提出，其比粒子群算法更为简单、且基本无参数。骨干粒子群算法中粒子的位置采用高斯采样策略进行更新。通常，第 i 个粒子的位置表示为 $x_i(t)=\left(x_{i,1}(t),x_{i,2}(t),\cdots,x_{i,D}(t)\right)$，则粒子在搜索过程中的位置更新方程为

$$x_{i,j}(t+1)=N\left(u_{i,j}(t),\sigma_{i,j}^2(t)\right) \tag{1}$$

其中，$u_{i,j}(t)=0.5\left(Pb_{i,j}(t)+Gb_j(t)\right)$为高斯分布的均值，$\sigma_{i,j}(t)=\left|Pb_{i,j}(t)-Gb_j(t)\right|$为高斯分布的标准差。$Pb_i(t)=\left(pb_{i,1}(t),pb_{i,2}(t),\cdots,pb_{i,D}(t)\right)$表示第 i 个粒子当前的局部最优位置，通常记为 Pbest；$Gb_i(t)=\left(gb_{i,1}(t),gb_{i,2}(t),\cdots,gb_{i,D}(t)\right)$表示当前粒子领域内找到的全局最优位置，通常记为 Gbest。

粒子的局部最优位置 Pbest 和群体全局最优位置 Gbest 的更新方程为

$$Pb_i(t+1)\begin{cases}Pb_i(t), f\left(Pb_i(t)\right)\le f\left(x_i(t+1)\right)\\ x_i(t+1), f\left(Pb_i(t)\right)>f\left(x_i(t+1)\right)\end{cases} \tag{2}$$

$$Gb(t+1)=\arg\min\left\{f\left(Pb_i(t+1)\right)\mid i=1,2,\cdots,N\right\} \tag{3}$$

3. 基于自适应突变的骨干粒子群算法

3.1. 自适应扰动策略

根据粒子的位置更新方程(1)可知，粒子的搜索中心不够灵活，一定程度上限制了粒子的搜索区域。为了在早期增强粒子的全局搜索能力，在高斯分布的均值项中引入两个[0, 1]之间的均匀随机数。为了避免过多的粒子停滞，需要使高斯分布的标准差不为 0。因此，改进的算法将自适应扰动值引入到了高斯分布的标准差部分。由于过大的扰动值会极大的影响粒子的收敛速度，使粒子难以快速达到收敛状态；扰动值过小则使得粒子仍易发生停滞，故设置该扰动值由在[0, 1]上的均匀分布数、粒子的多样性和收敛程度共同决定，这在一定程度上能使粒子的收敛速度和粒子的多样性维持平衡。

设第 i 个粒子所在的位置为 $x_i(t)=\left(x_{i,1}(t),x_{i,2}(t),\cdots,x_{i,D}(t)\right)$，后续迭代过程中粒子位置的更新方程为

$$x_{i,j}(t+1)=N\left(u_{i,j}(t),\sigma_{i,j}^2(t)\right)\cdot I_{\{rand<prob\}}+x_{i,j}(t)\cdot I_{\{rand\ge prob\}}$$

$$u_{i,j}(t)=rand_1\times Pb_{i,j}(t)+\left(1-rand_2\right)\times Gb_j(t)$$

$$\sigma_{i,j}(t)=\left|Pb_{i,j}(t)-Gb_j(t)\right|+\Delta$$

$$\Delta=rand_3\times\left|Pb_{k_1,j}(t)-Pb_{k_2,j}(t)\right|\times S\left(f\left(Gb(t)\right)-f\left(X_i(t)\right)\right) \tag{4}$$

其中，$rand_1$、$rand_2$ 和 $rand_3$ 是在[0, 1]上服从均匀分布的随机数；Pb_{k_1} 和 Pb_{k_2} 是从所有粒子中无放回地挑选出来的任意两个粒子的局部最优位置；$S(x)$ 为 sigmoid 函数；$f(\cdot)$ 表示要优化的目标函数；概率

$0 \le prob \le 1$。

上述位置更新策略中，扰动值 Δ 由随机的均匀分布数、两个随机挑选的 Pbest 的差、Gbest 和 X_i 适应度值的差三者共同决定。由式(4)可知，不考虑 $rand_3$ 时，假设 $\left|Pb_{k_1,j}(t)-Pb_{k_2,j}(t)\right|$ 保持不变，则当 $f\left(Gb(t)\right)=f\left(X_i(t)\right)$ 时，该粒子的扰动值最大，以帮助粒子跳出局部最优。根据 Pbest 和 Gbest 的更新方程，AMBPSO 经过多次迭代，所有粒子都将收敛到同一个位置 Gbest，即 $\lim\limits_{t\to\infty}\left|Pb_{k_1,j}(t)-Pb_{k_2,j}(t)\right|=0$，此时 $\lim\limits_{t\to\infty}\Delta=0$，这也是保证种群收敛的必要条件之一。

3.2. 自适应突变策略

在粒子位置未停止更新时，所得到的 Gbest 的值仍可能为局部最优值，故仅依靠提出的自适应扰动策略难以确保粒子收敛到全局最优。因此，将自适应突变策略引入加了自适应扰动策略的骨干粒子群算法中。对于每一个粒子 i，其都依概率发生突变，突变粒子的突变维数也由突变概率决定，设计了时变因子 L 来控制粒子的突变，来帮助粒子跳出局部最优。时变因子 L 的更新方程为

$$L=\tan\left(\frac{\pi}{4}\times\left(1-\frac{t}{it_{\max}}\right)\right) \tag{5}$$

其中，$it_{\max}$ 为最大迭代次数，且 L 随着迭代次数的增加而减小。

以粒子 i 为例，其发生突变的概率为

$$p(t)=0.3\times\left(1-3^{-N(t)}\right) \tag{6}$$

其中，t 为迭代的次数；$N(t)$ 为粒子停滞的次数，即

$$N(t)=\begin{cases}N(t-1)+1, & \left|f\left(Gb(t)\right)-f\left(Gb(t-1)\right)\right|=0\\ 0, & \left|f\left(Gb(t)\right)-f\left(Gb(t-1)\right)\right|\neq 0\end{cases} \tag{7}$$

自适应突变的伪代码如下：

Table 1. Adaptive mutation
表 1. 自适应突变

自适应突变伪代码
1. for $i=1:N$
2. if $p(t)>rand$: //*rand~U(0,1)*//
3. $M\_d=floor\left(p(t)\times D\right)$
4. $m=randperm\left(D,M\_d\right)$
5. for $j=1:length(m)$
6. $x_{i,m(j)}=L\times Pb_{i,m(j)}+(1-L)\times Gb_{m(j)}$
7. end for
8. end if
9. end for

由表 1 可知，突变概率的大小影响了突变的速度，大的突变概率使粒子更容易发生突变，过小的突变概率会使粒子难以快速跳出局部最优。每个粒子都有一定的概率会发生突变，从而增加粒子的多样性，为粒子跳出局部最优做准备。为了缓解突变时对 Pbest 和 Gbest 的经验知识的破坏，粒子突变后的位置为随迭代次数自适应的 Pb 和 Gb 的加权平均值，且迭代前期粒子偏向于对自身经验的认知，增强了粒子的全局搜索能力；迭代后期粒子偏向于对群体经验的认知，进一步保证了粒子的收敛性。

3.3. 算法流程

基于自适应突变的骨干粒子群算法的具体步骤如下：

步骤 1 设置种群参数；

步骤 2 伪随机数法初始化粒子；

步骤 3 评估整个种群，更新粒子的个体最优位置 Pbest 和种群的全局最优位置 Gbest；

步骤 4 根据式(4)更新每个粒子的位置；

步骤 5 计算粒子全局最优位置 Gbest 在迭代过程中的停滞次数；

步骤 6 计算粒子突变的概率及发生突变粒子的突变维度，进而更新粒子的位置；

步骤 7 重新评估种群，计算粒子的个体最优位置和种群的全局最优位置；

步骤 8 判断是否满足进化终止条件，若不满足则返回步骤 4；若满足，则算法终止。

改进后的骨干粒子群算法的流程图如图 1 所示：

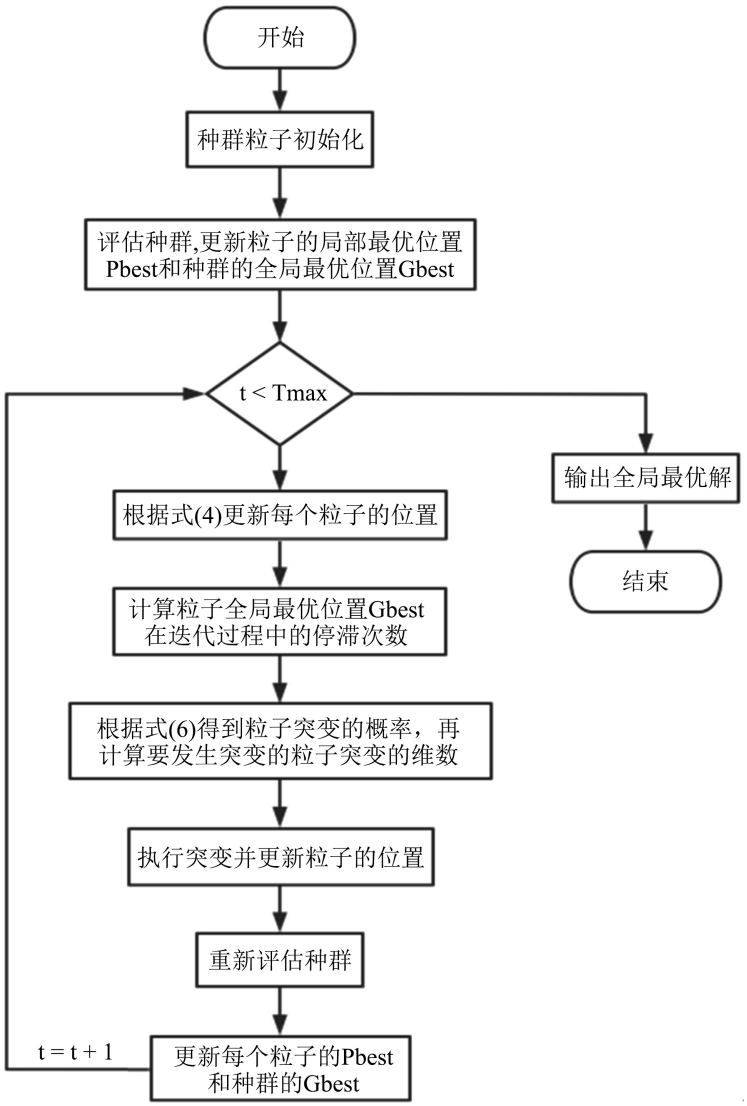

Figure 1. The flowchart of the AMBPSO algorithm
图 1. 自适应突变骨干粒子群算法流程图

3.4. 参数选择

Sphere 函数是最简单的测试函数，其只有一个全局最优解，故利用 AMBPSO 算法在 Sphere 函数上进行测试，以确定的 *prob* 的最佳值。实验过程中，将粒子的个数和问题的维度均设置为 30，搜索空间为 $[-100,100]^{30}$。为了简化测试过程，每次迭代过程中所有粒子适应度值的标准差被用来计算粒子的多样性，各粒子当前位置和群体全局最优位置间的最大距离代表种群的收敛程度。实验结果如图 2 和图 3 所示。

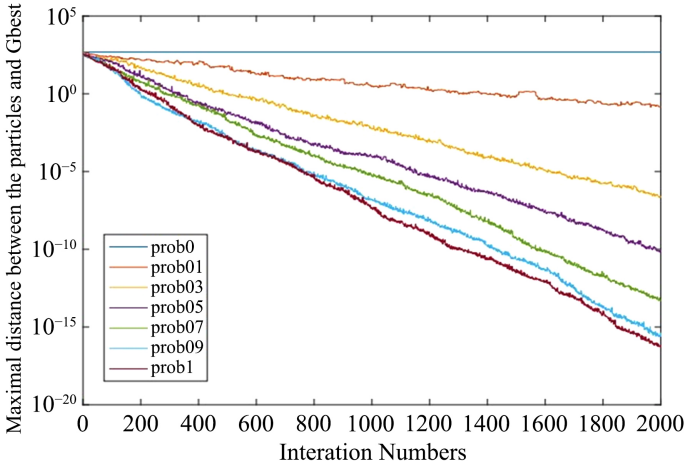

Figure 2. The maximum distance between particles and the global optimal position of the swarm
图 2. 各粒子和种群全局最优位置间的最大距离

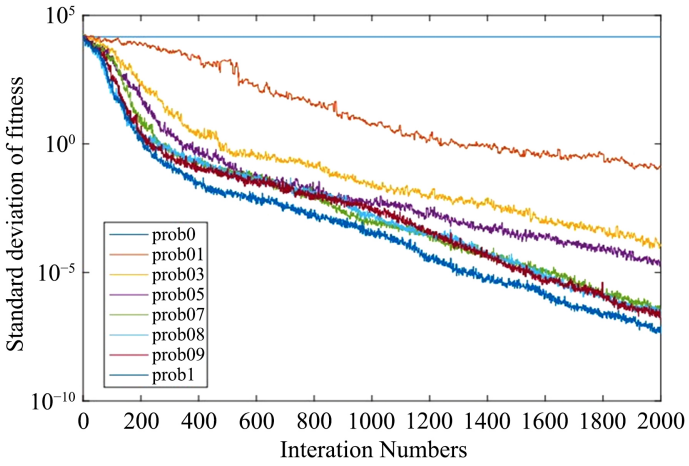

Figure 3. Standard deviation of the fitness of all particles in each iteration
图 3. 每次迭代过程中所有粒子适应度值的标准差

由实验结果可得，当 $prob>0$，所有粒子和 Gbest 间的最大距离随着 *prob* 增大而减小。当 $prob=1$ 时，粒子的收敛程度和收敛速度都最大。由种群中所有粒子适应度值的标准差曲线可得，粒子的多样性也随着 *prob* 的增大而减小。当 *prob* 的值在 0.7 和 0.9 之间时，粒子的多样性曲线基本相同。经过多次试验可得，当 *prob* 的取值范围为[0.7, 0.9]时，实验结果达到最优。因此，在后续的试验过程中将 *prob* 设置为 0.8。

4. 仿真实验

为了进一步验证改进算法的性能，将 AMBPSO 算法与五种典型的粒子群算法进行比较。所有用于对比实验的算法，其相关描述如表 2 所示。在进行对比实验时，将粒子个数设置为 30，问题维度也为 30，最大迭代次数设置为 2000。对于每个测试函数，每种算法进行 30 次独立实验，避免随机性对实验结果的影响。

Table 2. Algorithms and descriptions for conducting experiments
表 2. 进行实验的算法及其描述

序号	算法	算法描述	参考文献
1	BBPSO	由 Kennedy 首先提出的标准 BBPSO 算法	[1]
2	PSO	标准 PSO 算法	[2]
3	ABPSO	带有自适应扰动值的 BBPSO 算法	[5]
4	SLBPSO	自学习 BBPSO 算法	[11]
5	AWPSO	基于 sigmoid 函数的自适应权重 PSO 算法	[17]
6	AMBPSO	基于自适应突变策略的 BBPSO 算法	改进的算法

4.1. 测试函数

为了测试 AMBPSO 算法在收敛速度和精度方面的提升，采用 9 个典型的测试函数对算法进行仿真实验。测试函数如表 3 所示。

Table 3. Benchmark functions
表 3. 测试函数

函数名称	公式	问题维度	搜索空间	最优值
Sphere	$F_1(x)=\sum_{i=1}^{D}x_i^2$	30	[−100, 100]	0
Schwefel 1.2	$F_2(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j^2\right)^2$	30	[−100, 100]	0
Rosenbrock	$F_3(x)=\sum_{j=1}^{D-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	30	[−30, 30]	0
Schwefel 2.22	$F_4(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30	[−10, 10]	0
Quartic Function *i.e.* Noise	$F_5(x)=\sum_{i=1}^{D}ix_i^4+random[0,1)$	30	[−1.28, 1.28]	0

Continued

Rastrigin	$F_6(x)=\sum_{i=1}^{D}x_i^2-10\cos(2\pi x_i)+10$	30	[−5.12, 5.12]	0
Ackley	$F_7(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+e$	30	[−32, 32]	0
Griewank	$F_8(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[−600, 600]	0
Schwefel	$F_9(x)=418.9829\times D-\sum_{i=1}^{D}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[−500, 500]	0

4.2. 性能指标

在进行对比试验时，使用三个不同的指标[5] [15]来检验改进算法的性能。

第一个指标为每次实验所得种群收敛的精度的均值(Mean of AC，简写为 Mean)。设测试函数的全局最优解为 x_{opt}，群体在第 t 次迭代时得到的全局最优解为 Gb，则精度(Accuracy，简写 AC)为

$$\mathrm{AC}=\left|f\left(Gb(t)\right)-f\left(x_{opt}\right)\right| \tag{8}$$

第二个指标为实验得到的精度的标准差(Standard Deviation of AC，简写 Std)。由于算法中含有多个随机项，每次实验结果都具有随机性，故该指标用来验证算法寻优的稳定性。

第三个指标为实验的成功率(SR)。该指标指在算法迭代过程没有结束时，找到符合精度(在此设置为 10^{-8})要求的全局最优解的概率。其表达式为

$$\mathrm{SR}=\frac{n'}{n}\times 100\% \tag{9}$$

其中，n' 为实验成功的次数，n 为总实验次数。同时，当 $\mathrm{SR}=100\%$，进一步考虑找到全局最优解的平均函数评估次数(NFE)，其表达式为

$$\mathrm{NFE}=\frac{1}{n}\sum_{i=1}^{n}\mathrm{FE} \tag{10}$$

其中，FE 为每次实验结果达到设置的收敛精度前的函数评估次数。

4.3. 结果与分析

如表 4 所示，通过平均 AC 值、AC 的标准差和平均函数评估次数来评估粒子群优化算法的搜索能力，实验的成功率用于验证算法跳出局部最优的能力。表 3 中提到的所有测试函数用于测试算法是否能够获得其全局最优值。当达到最大迭代次数时，对于相同的测试函数，AC 值越小，算法的收敛精度越高。

由表 4 可知，与其他五种算法相比，AMBSPO 算法在测试函数 $F_1(x)\sim F_5(x)$、$F_7(x)$ 和 $F_9(x)$ 上的平均 AC 值最小。在测试函数 $F_8(x)$ 上，尽管 AMBPSO 算法的平均 AC 值不是最小的，但它明显小于标准 PSO 算法的平均 AC 值，并且它与 BBPSO 算法和 ABPSO 算法中的平均 AC 值非常相近。在测试函数 $F_1(x)\sim F_4(x)$、$F_7(x)$ 和 $F_9(x)$ 上，AMBPSO 算法的标准差最小，这表明改进的算法在搜索这些函数的全局最优值时具有显著的稳定性。对于函数 $F_5(x)$、$F_6(x)$ 和 $F_8(x)$，AMBSPO 算法与其他大多数算法之间的标准差的差异不大，其所得结果的标准差处于理想的状态。

Table 4. Results of algorithms on nine test functions
表 4. 算法在 9 个测试函数上的结果

函数	指标项	PSO	AWPSO	BBPSO	ABPSO	SLBPSO	AMBPSO
F_1	Mean	5.8645E−01	8.2088E−28	3.3333E+02	2.1342E−23	4.6956E−25	**7.4693E−29**
	Std	6.7375E−01	1.9860E−27	1.8257E+03	8.7759E−23	7.8172E−25	**1.8468E−28**
	NFE	-	3.8491E+04	-	2.6654E+04	3.2723E+04	**2.0301E+04**
	SR (%)	0	100	96.67	100	100	**100**
F_2	Mean	1.1792E+04	7.8254E−02	2.7914E+01	4.0679E+01	1.3803E+02	**4.6321E−02**
	Std	8.6615E+03	1.1846E−01	2.4662E+01	2.6876E+01	7.9778E+01	**8.4399E−02**
	NFE	-	-	-	-	-	-
	SR (%)	0	0	0	0	0	**0**
F_3	Mean	1.2524E+04	2.8692E+01	4.3361E+01	4.6073E+01	8.2126E+01	**2.6915E+01**
	Std	3.0949E+04	2.1694E+01	3.3949E+01	3.1197E+01	5.5031E+01	**2.5976E−01**
	NFE	-	-	-	-	-	-
	SR (%)	0	0	0	0	0	**0**
F_4	Mean	1.9390E+01	2.6914E−11	1.5333E+01	1.0019E−17	7.7017E−18	**1.0460E−21**
	Std	1.5792E+01	1.0720E−10	9.7320E+00	2.3703E−17	7.6163E−18	**1.4035E−21**
	NFE	-	4.6068E+04	-	3.0947E+04	3.6778E+04	**2.4657E+04**
	SR (%)	0	100	10	100	100	**100**
F_5	Mean	1.6795E−02	1.3231E−02	1.0956E−01	1.8403E−02	1.7929E−02	**9.3443E−03**
	Std	4.7618E−03	4.9092E−03	4.9988E−01	6.6083E−03	6.2313E−03	**6.2425E−03**
	NFE	-	-	-	-	-	-
	SR (%)	0	0	0	0	0	**0**
F_6	Mean	1.0056E+02	2.9716E+01	4.6730E+01	4.1125E+01	**8.0245E+00**	1.1486E+01
	Std	4.8578E+01	7.1830E+00	1.1572E+01	1.2223E+01	**5.3222E+00**	2.9171E+01
	NFE	-	-	-	-	-	-
	SR (%)	0	0	0	0	**0**	60
F_7	Mean	7.8025E−01	1.0421E−14	4.6003E−01	3.3342E−01	1.8841E−13	**5.9212E−15**
	Std	2.5534E+00	7.1013E−15	8.3027E−01	7.0981E−01	1.2158E−13	**1.7034E−15**
	NFE	-	4.6031E+04	-	-	4.4635E+04	**3.3460E+04**
	SR (%)	0	100	73.33	80	100	**100**

Continued

F_8	Mean	6.6299E+00	9.8373E−03	1.1638E−02	1.0586E−02	**1.4196E−03**	1.1758E−02
	Std	2.2937E+01	1.5217E−02	1.3031E−02	1.3843E−02	**3.2722E−03**	2.6468E−02
	NFE	-	-	-	-	**-**	-
	SR (%)	0	56.67	36.67	36.67	**66.67**	76.67
F_9	Mean	6.1476E+03	6.2923E+03	3.5866E+03	2.5063E+03	3.5882E+04	**2.2451E+03**
	Std	7.4177E+02	7.5982E+02	5.9212E+02	6.1715E+02	1.2356E+04	**4.7662E+02**
	NFE	-	-	-	-	-	**-**
	SR (%)	0	0	0	0	0	**0**

Table 5. Result of t-test
表 5. t 检验结果

	PSO	AWPSO	BBPSO	ABPSO	SLBPSO
F_1	+	+	+	+	+
F_2	+	=	+	+	+
F_3	+	=	+	+	+
F_4	+	+	+	+	+
F_5	+	+	+	+	+
F_6	+	+	+	+	−
F_7	+	+	+	+	+
F_8	+	=	=	=	−
F_9	+	+	+	=	+

为了更详细地验证 AMBPSO 的有效性，根据每次实验获得的 AC 值，利用显著性水平为 0.05 ($\alpha = 0.05$)的双样本双尾 t 检验，对其他五种算法获得的结果和 AMBPSO 算法获得的结果进行了检验。在表 5 中，“+”表示 AMBPSO 算法显著优于对比的算法，“−”与之相反；“=”表明 AMBPSO 与比较的算法没有显著差异。故由表 5 可得，改进的算法在测试函数 $F_1(x)$，$F_4(x)$，$F_5(x)$ 和 $F_7(x)$ 上明显优于其他所有算法。在测试函数 $F_2(x)$ 和 $F_3(x)$ 上，AMBPSO 算法优于除 AWPSO 算法外的其他四种算法。对于函数 $F_6(x)$，AMBPSO 算法所得结果仅和 SLBPSO 算法在 t 检验结果上对比时没有显著优势。对于函数 $F_8(x)$，尽管三种算法(AWPSO、BBPSO、ABPSO)的 AC 值小于 AMBPSO 算法的 AC 值，但它们和 AMBPSO 算法没有显著差异，且 AMBPSO 算法显著优于 PSO 算法。对于函数 $F_9(x)$，AMBPSO 算法的 AC 值显著小于除 ABPSO 算法外的其他四种算法，和 ABPSO 算法所得结果没有显著差异。通过比较平均 AC 值，可以得出 AMBSPO 算法在绝大多数测试函数上比其他五种算法达到更高的收敛精度。

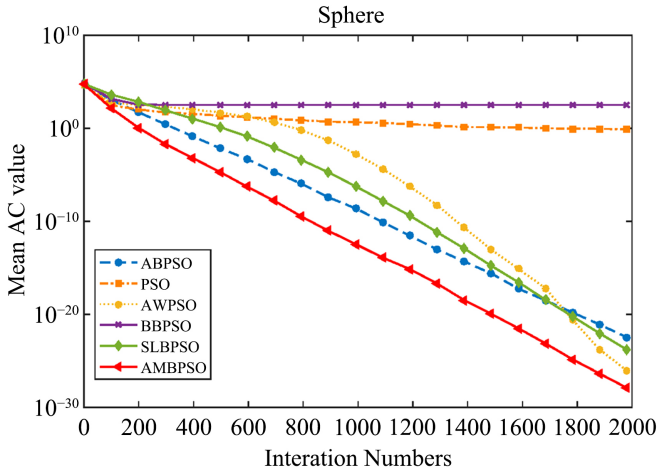

Figure 4. Convergence curve of the function $F_1(x)$
图 4. 函数 $F_1(x)$收敛性曲线

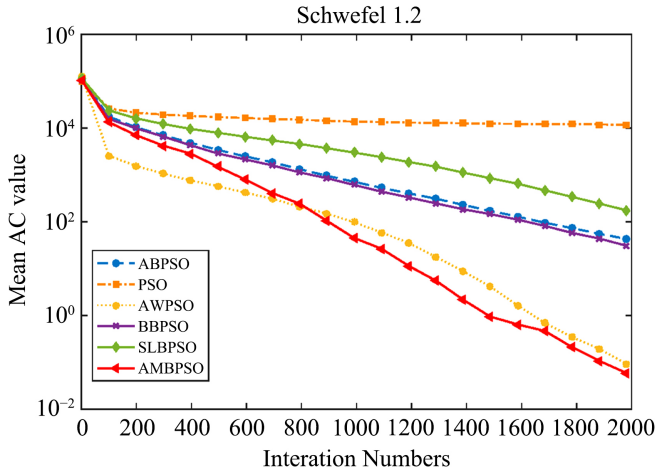

Figure 5. Convergence curve of the function $F_2(x)$
图 5. 函数 $F_2(x)$收敛性曲线

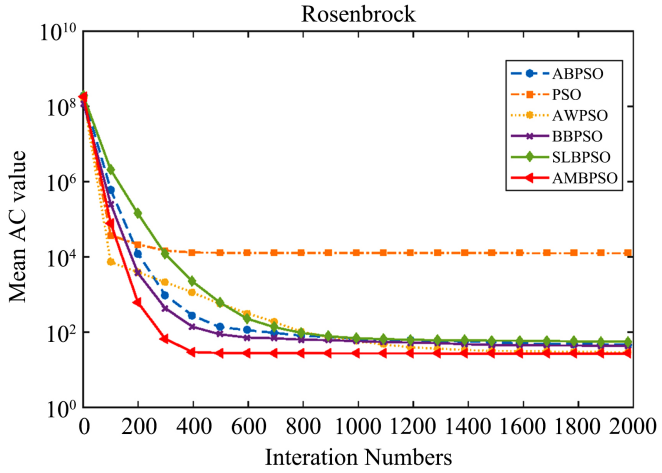

Figure 6. Convergence curve of the function $F_3(x)$
图 6. 函数 $F_3(x)$收敛性曲线

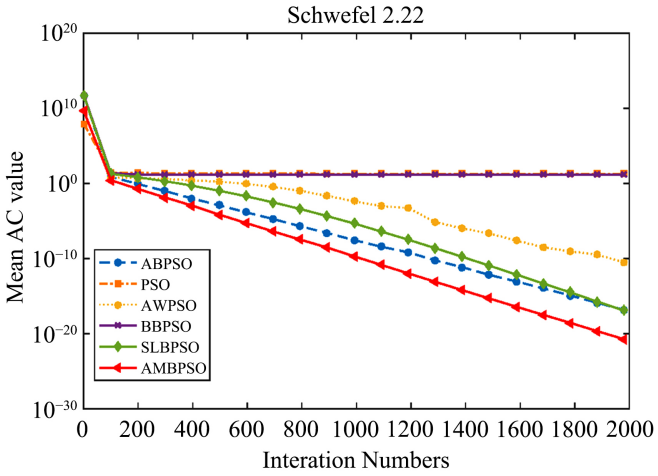

Figure 7. Convergence curve of the function $F_4(x)$
图 7. 函数 $F_4(x)$收敛性曲线

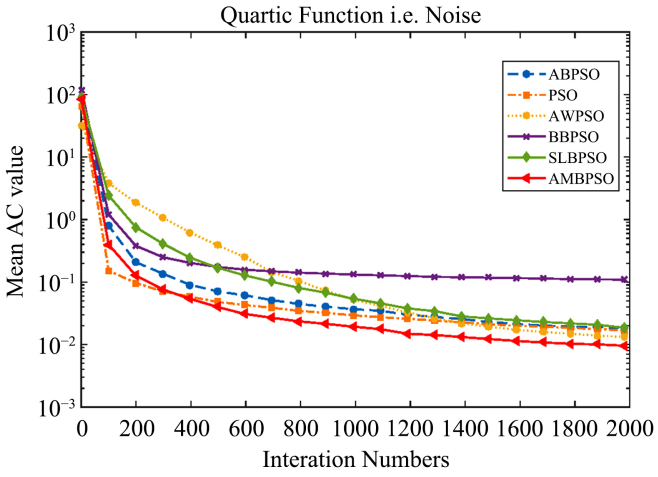

Figure 8. Convergence curve of the function $F_5(x)$
图 8. 函数 $F_5(x)$收敛性曲线

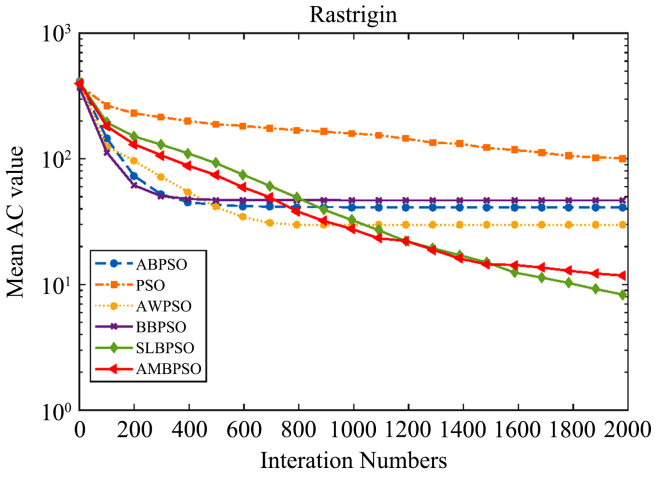

Figure 9. Convergence curve of the function $F_6(x)$
图 9. 函数 $F_6(x)$收敛性曲线

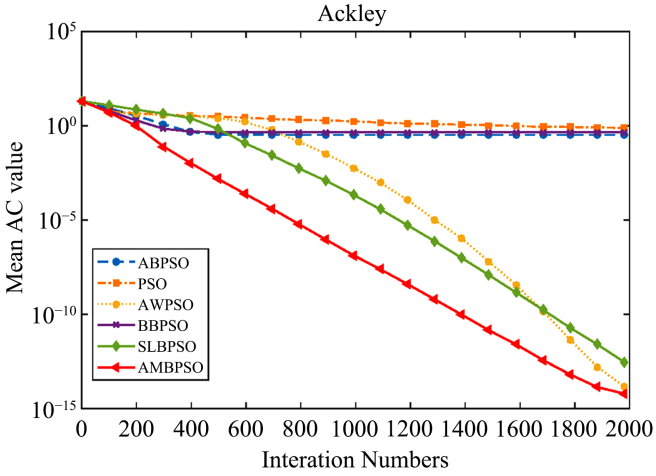

Figure 10. Convergence curve of the function $F_7(x)$
图 10. 函数 $F_7(x)$收敛性曲线

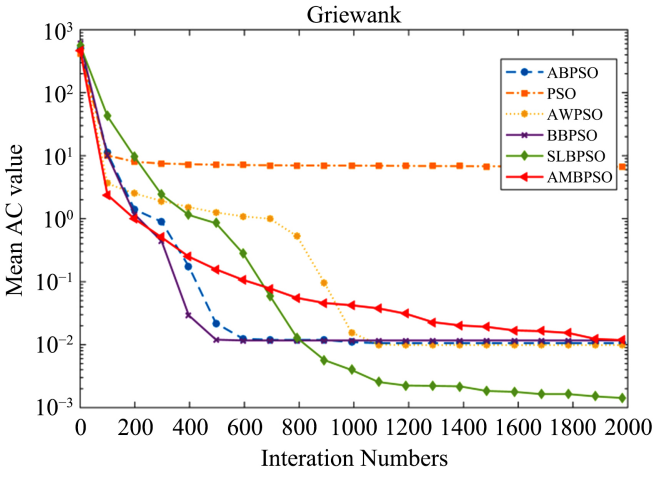

Figure 11. Convergence curve of the function $F_8(x)$
图 11. 函数 $F_8(x)$收敛性曲线

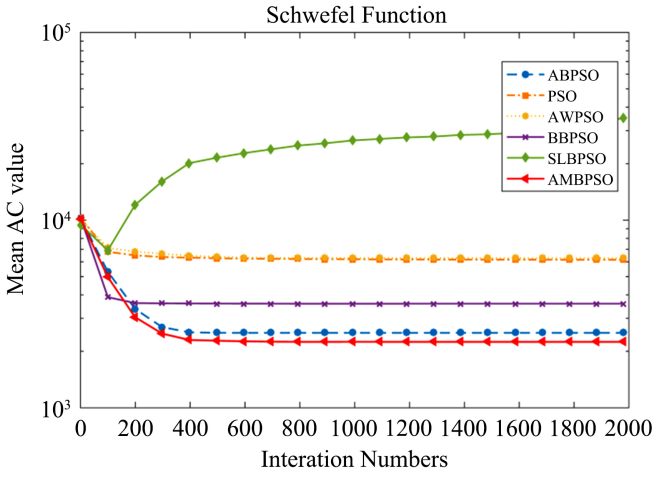

Figure 12. Convergence curve of the function $F_9(x)$
图 12. 函数 $F_9(x)$收敛性曲线

实验的成功率 SR 对评估算法效果同样非常重要。从表 4 中可以看出，只有 AMBSPO 算法在测试函数 $F_6(x)$ 上达到 60%的成功率，而其他算法在达到最大迭代次数时无法收敛到指定的精度。测试函数 $F_8(x)$ 是多峰函数且具有广泛的局部极小值，这是实验无法实现 100%的成功率的原因，但 AMBSPO 算法获得了最大的 SR，且远远超过了其他算法。对于函数 $F_1(x)$ 和 $F_4(x)$，AMBPSO、AWPSO、ABPSO 和 SLBPSO 算法的实验成功率均达到 100%，但通过比较它们的平均函数评估次数，可以看出 AMBPSO 算法的收敛速度比其他算法快得多。对于函数 $F_7(x)$，AWPSO、SLBPSO 和 AMBPSO 算法达到 100%的成功率，但 AMBPSO 算法的平均函数评估次数为 3.3460E+04，比 AWPSO 和 SLBPSO 算法的平均函数评估次数少近 10,000 次，表明 AMBPSO 算法的收敛速度明显快于其他两种算法。六种算法在每个函数上的收敛曲线如图 4~12 所示。横轴表示迭代次数，纵轴表示平均 AC 值，可以清楚地显示每个算法的收敛速度。显然，AMBPSO 算法的收敛速度和收敛精度明显优于其他算法。尽管 SLBPSO 算法在函数 $F_6(x)$ 和 $F_8(x)$ 上比 AMBPSO 算法获得更好的平均 AC 值，但 AMBPSO 算法的实验成功率远高于 SLBPSO 算法。

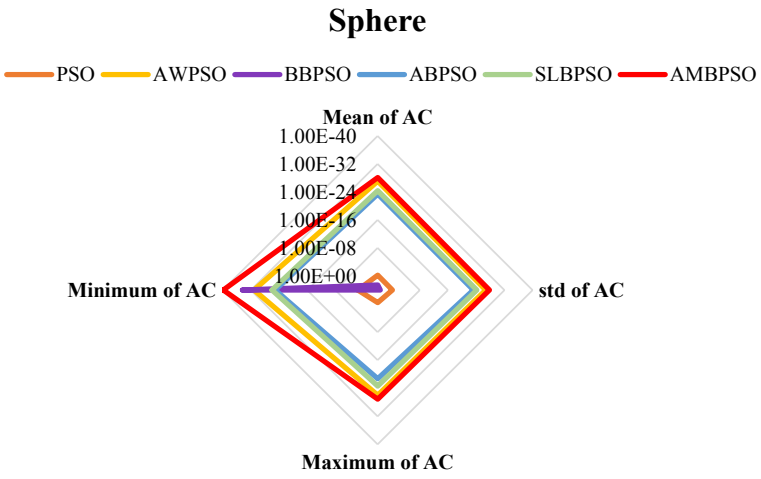

Figure 13. Comparison of statistical results on function $F_1(x)$
图 13. 函数 $F_1(x)$上实验结果统计值比较

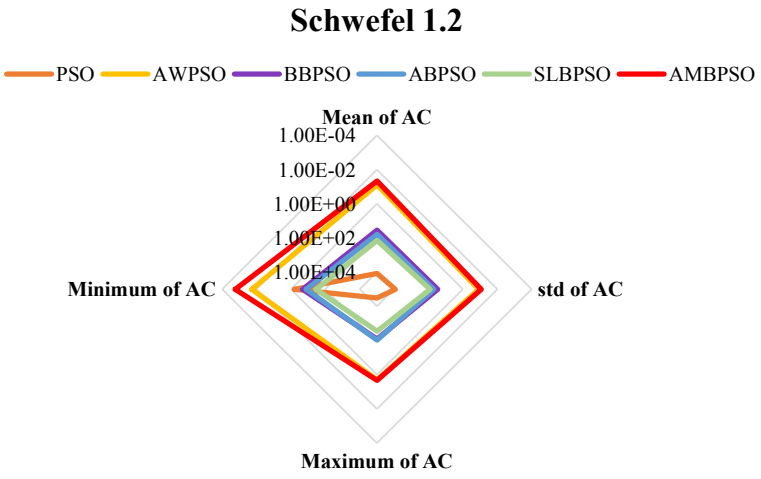

Figure 14. Comparison of statistical results on function $F_2(x)$
图 14. 函数 $F_2(x)$上实验结果统计值比较

在每个测试函数上实验结果统计值的雷达图如图 13~19 所示。对于函数 $F_6(x)$，AMBSPO 算法获得的最小 AC 值为 0。对于函数 $F_8(x)$，所有算法获得的最小 AC 值均为 0。因此，算法在这两个测试函数上获得的结果不再绘制雷达图。由图 13~19 可得，AMBSPO 算法在几乎所有函数上的性能都远远优于其他算法，特别是在最小 AC 值、平均 AC 值和标准差方面。对于函数 $F_3(x)$，虽然 AWPSO 算法的最小 AC 值最小，但其最大 AC 值远大于 AMBPSO 算法，且 AMBPSO 算法的平均 AC 值小于 AWPSO 算法，最大 AC 值是所有算法中最小的，这表明了 AMBPSO 算法的稳定性。对于函数 $F_9(x)$，SLBPSO 算法虽然得到了最小的 AC 值，但它的最大 AC 值也远大于比其他算法。此外，在函数 $F_9(x)$ 上 SLBPSO 算法的平均 AC 值和标准差也大于其他算法，说明 SLBPSO 算法非常不稳定。因此，通过比较四个统计量的值，可得 AMBPSO 算法比其他算法具有显著优势。

综上所述，AMBPSO 算法总体上比其他五种算法具有更快的收敛速度和更高的收敛精度。

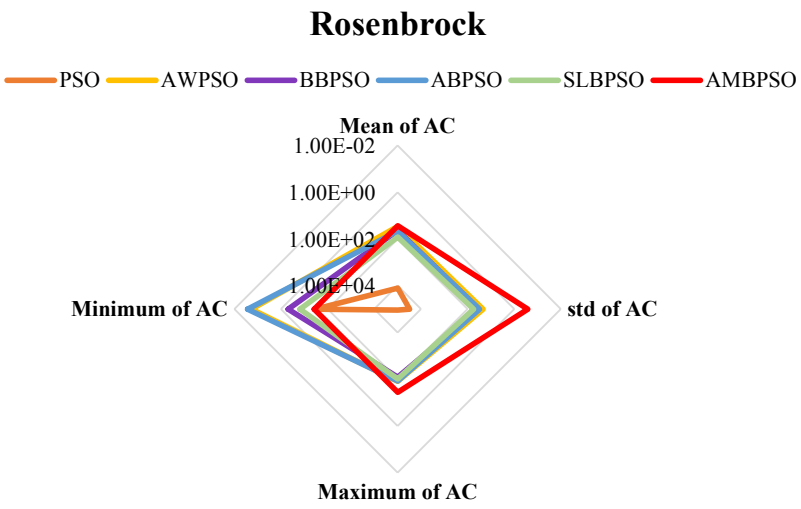

Figure 15. Comparison of statistical results on function $F_3(x)$
图 15. 函数 $F_3(x)$上实验结果统计值比较

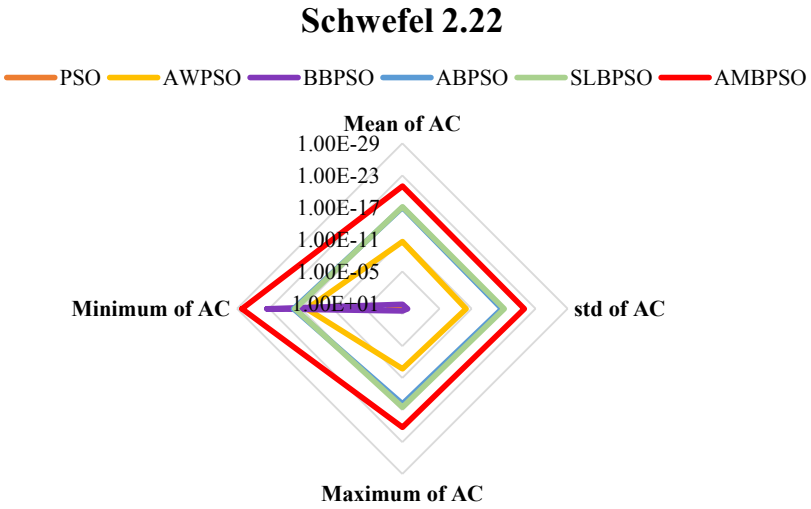

Figure 16. Comparison of statistical results on function $F_4(x)$
图 16. 函数 $F_4(x)$上实验结果统计值比较

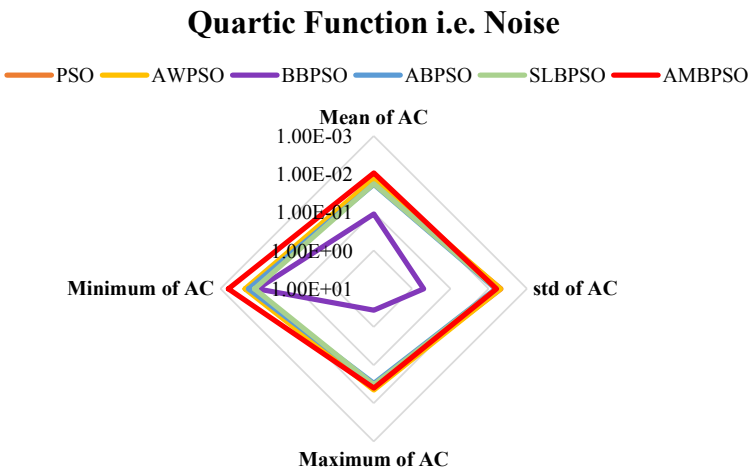

Figure 17. Comparison of statistical results on function $F_5(x)$
图 17. 函数 $F_5(x)$上实验结果统计值比较

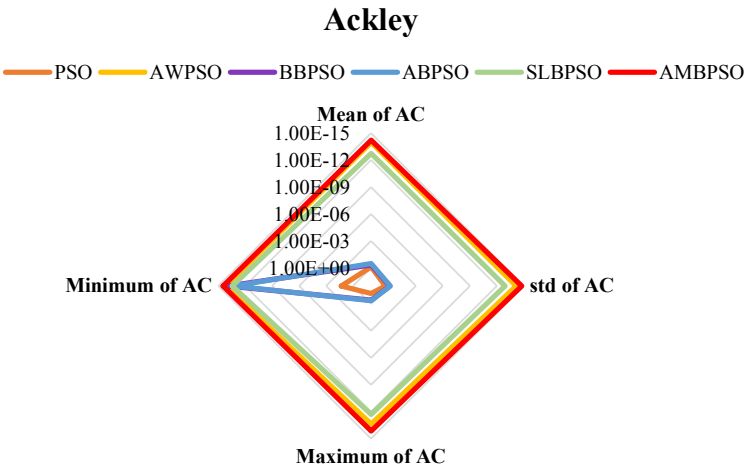

Figure 18. Comparison of statistical results on function $F_7(x)$
图 18. 函数 $F_7(x)$上实验结果统计值比较

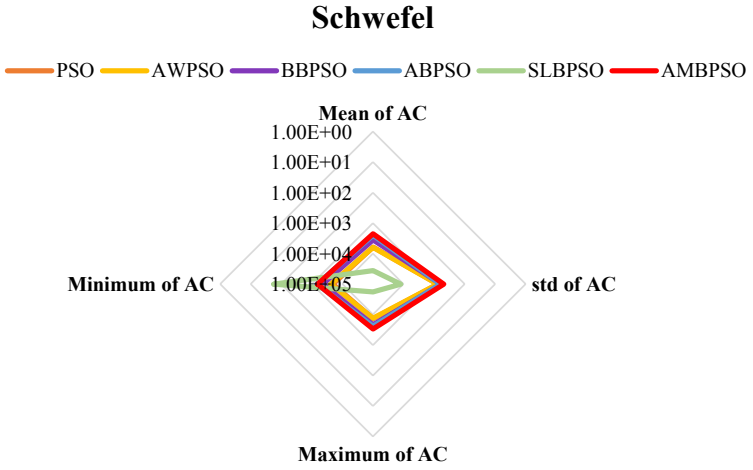

Figure 19. Comparison of statistical results on function $F_9(x)$
图 19. 函数 $F_9(x)$上实验结果统计值比较

5. 结束语

针对骨干粒子群算法易陷入局部最优、收敛速度较慢的问题，本文提出了一种自适应突变骨干粒子群算法。通过在粒子的位置更新方程中引入自适应扰动值，增加了粒子的多样性，帮助粒子跳出局部最优。同时，设计了一种突变策略，使粒子按照适当的概率发生突变。突变过程中引入了具有下降趋势的时变因子，以平衡粒子的勘探和开发能力。采用 9 个经典测试函数对所提出的算法进行了验证，并将提出的 AMBPSO 算法和 PSO，AWPSO，BBPSO，ABPSO，SLBPSO 算法进行了比较。实验结果表明，AMBPSO 算法的寻优能力在多种类型的函数上均优于其他粒子群算法，且提出的算法扩大了粒子群的搜索范围，显著提高了粒子在搜索全局最优解时的收敛速度和收敛精度。如何利用严谨的数学理论推导分析算法的收敛性将是下一步的主要工作。

参考文献

[1] Kennedy, J. (2003) Bare Bones Particle Swarms. *Proceedings of the* 2003 *IEEE Swarm Intelligence Symposium*, Indianapolis, 26-26 April 2003, 80-87.

[2] Kennedy, J. and Eberhart, R. (1995) Particle Swarm Optimization. *Proceedings of ICNN'95-International Conference on Neural Networks*, Perth, 27 November-1 December 1995, 1942-1948.

[3] Houssein, E.H., Gad, A.G., Hussain, K. and Suganthan, P.N. (2021) Major Advances in Particle Swarm Optimization: Theory, Analysis, and Application. *Swarm and Evolutionary Computation*, **63**, 100868. https://doi.org/10.1016/j.swevo.2021.100868

[4] Pan, F., Hu, X., Eberhart, R.C. and Chen, Y. (2008) An Analysis of Bare Bones Particle Swarm. 2008 *IEEE Swarm Intelligence Symposium*, St. Louis, 21-23 September 2008, 21-23. https://doi.org/10.1109/SIS.2008.4668301

[5] Zhang, Y., Gong, Dw., Sun, X.-Y. and Geng, N. (2014) Adaptive Bare-Bones Particle Swarm Optimization Algorithm and Its Convergence Analysis. *Soft Computing*, **18**, 1337-1352. https://doi.org/10.1007/s00500-013-1147-y

[6] Lehre, P.K. and Witt, C. (2013) Finite First Hitting Time versus Stochastic Convergence in Particle Swarm Optimization. Springer, New York, 1-20. https://doi.org/10.1007/978-1-4614-6322-1_1

[7] Zhang, Y., Gong, D.-W., Geng, N. and Sun, X.-Y. (2014) Hybrid Bare-Bones PSO for Dynamic Economic Dispatch with Valve-Point Effects. *Applied Soft Computing*, **18**, 248-260. https://doi.org/10.1016/j.asoc.2014.01.035

[8] Song, X.-F., Zhang, Y., Gong, D.-W. and Sun, X.-Y. (2021) Feature Selection Using Bare-Bones Particle Swarm Optimization with Mutual Information. *Pattern Recognition*, **112**, Article ID: 107804. https://doi.org/10.1016/j.patcog.2020.107804

[9] Yang, C., Liu, T., Yi, W., Chen, X. and Niu, B. (2020) Identifying Expertise through Semantic Modeling: A Modified BBPSO Algorithm for the Reviewer Assignment Problem. *Applied Soft Computing*, **94**, Article ID: 106483. https://doi.org/10.1016/j.asoc.2020.106483

[10] 王东风, 孟丽, 赵文杰. 基于自适应搜索中心的骨干粒子群算法[J]. 计算机学报, 2016, 39(12): 2652-2667.

[11] Chen, J., Shen, Y. and Wang, X. (2015) A Self-Learning Bare-Bones Particle Swarms Optimization Algorithm. In: Tan, Y., Shi, Y., Buarque, F., Gelbukh, A., Das, S. and Engelbrecht, A., Eds., *Advances in Swarm and Computational Intelligence*. *ICSI* 2015. *Lecture Notes in Computer Science*, Vol. 9140, Springer, Cham, 107-114. https://doi.org/10.1007/978-3-319-20466-6_12

[12] Lin, M., Wang, Z., Chen, D. and Zheng, W. (2022) Particle Swarm-Differential Evolution Algorithm with Multiple Random Mutation. *Applied Soft Computing*, **120**, Article ID: 108640. https://doi.org/10.1016/j.asoc.2022.108640

[13] Omran, M., Engelbrecht, A. and Salman, A. (2009) Bare Bones Differential Evolution. *European Journal of Operational Research*, **196**, 128-139. https://doi.org/10.1016/j.ejor.2008.02.035

[14] Xiong, G.J., Shuai, M.H. and Hu, X. (2022) Combined Heat and Power Economic Emission Dispatch Using Improved Bare-Bone Multi-Objective Particle Swarm Optimization. *Energy*, **244**, Article ID: 123108. https://doi.org/10.1016/j.energy.2022.123108

[15] Tuba, I., Veinovic, M., Tuba, E., Capor Hrosik, R. and Tuba, M. (2022) Tuning Convolutional Neural Network Hyperparameters by Bare Bones Fireworks Algorithm. *Studies in Informatics and Control*, **31**, 25-35. https://doi.org/10.24846/v31i1y202203

[16] Clerc, M. and Kennedy, J. (2002) The Particle Swarm-Explosion, Stability and Convergence in a Multidimensional

Complex Space. *IEEE Transactions on Evolutionary Computation*, **6**, 58-73. https://doi.org/10.1109/4235.985692

[17] Liu, W., Wang, Z., Zeng, N., *et al*. (2021) A Novel Sigmoid-Function-Based Adaptive Weighted Particle Swarm Optimizer. *IEEE Transactions on Cybernetics*, **51**, 1085-1093. https://doi.org/10.1109/TCYB.2019.2925015