基于模拟退火与高斯扰动的烟花优化算法

韩守飞 李席广 拱长青
(沈阳航空航天大学计算机学院 沈阳 110136)

摘 要 烟花算法(Fireworks Algorithm,FWA)是一种群体智能优化算法,具有求解复杂问题的全局最优解的能力。为了提高FWA求解全局最优解的能力,将模拟退火的思想引入到烟花优化算法中,并对FWA中某些单个烟花个体进行高斯扰动,提出了一种基于模拟退火与高斯扰动的烟花优化算法(SAFWA)。分别把烟花算法(FWA)、标准粒子群算法(SPSO)、增强烟花算法(EFWA)和SAFWA在10个典型的基准测试函数中进行仿真对比,结果表明,在收敛速度、计算精度以及稳定性方面,SAFWA均优于其他3种算法。

关键词 烟花算法,模拟退火,高斯扰动,标准粒子群算法,增强烟花算法

中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.05.046

Fireworks Optimization Algorithm Based on Simulated Annealing and Gaussian Perturbations

HAN Shou-fei LI Xi-guang GONG Chang-qing
(College of Computer Science,Shenyang Aerospace University,Shenyang 110136,China)

Abstract Fireworks algorithm (FWA) is a kind of swarm intelligence optimization algorithm for solving complex problems with a global capacity of the optimal solution. This paper introduced both simulated annealing and Gaussian perturbations into the standard FWA so as to improve its ability to solve global optimal solution. As a result, this paper proposed a simulated annealing Gauss fireworks algorithm (SAFWA) for global optimization. In our simulation, we compared FWA, SPSO, EFWA and SAFWA with 10 typical benchmark functions. The results show that SAFWA is better than FWA, SPSO, and EFWA in terms of convergence speed, accuracy and stability.

Keywords Fireworks algorithm, Simulated annealing, Gaussian perturbations, SPSO, EFWA

烟花算法(FWA)是谭营教授于2010年受烟花在夜空中爆炸的启发而提出的一种群体智能优化算法。FWA通过模拟烟花在空中爆炸的行为建立相应的数学模型,通过引入随机因素和选择策略形成一种并行爆炸式搜索方式,进而发展成为一种能够求解复杂问题最优解的全局概率搜索方法。FWA与一般的群体智能优化算法类似,首先随机初始化 N 个烟花,每个烟花经历爆炸和变异操作,并应用映射规则保证变异后的个体仍然在可行域内;保留最优的烟花,并应用选择策略从剩下的烟花中选择出 $N-1$ 个烟花,与最优的烟花组成群体进行下一次迭代。目前,烟花算法已被应用于许多实际优化问题的求解中,应用领域包括方程组求解、非负矩阵分解的计算、垃圾邮件检测算法中的参数优化等。

烟花算法自提出后就受到诸多科研工作者的关注,许多优化算法相继被提出。文献[1]提出对烟花算法的改进主要有两类,一类是在原有烟花算法的基础上进行算子的分析和改进,另一类是与其他启发式算法的混合。文献[2]针对基本烟花算法的不足,提出了增强的烟花算法(EFWA)。文献[3]提出了使用差分演化算法和烟花算法进行混合的新型算法(FWA-DE)。文献[4]使用差分变异替换基本烟花的高斯变异。文献[5]提出将文化算法和烟花算法进行混合的文化烟花算法(CA-FWA)。文献[6]将生物地理学优化算法引入烟花算法,提出了混合算法(BBO-FWA)。本文将模拟退火算法与烟花算法结合,并引入高斯扰动,提出了基于模拟退火与高斯扰动的烟花优化算法(SAFWA)。该算法既保持了烟花算法的特征,又在收敛速度和计算精度上有一定的提高,并且具有更好的稳定性。

1 烟花算法

燃放烟花(fireworks)或爆竹(crackers)是中国人最重要的节日活动之一,具有上千年的历史。每逢中国的传统节日,成千上万的烟花或爆竹在夜空爆炸并产生五彩缤纷的图案。

烟花算法的基本原则:若烟花对应的适应度函数值越小,则该烟花爆炸产生的火花数量越多,爆炸幅度越小;反之,若烟花对应的适应度函数值越大,则该烟花爆炸产生的火花数

到稿日期:2016-04-26 返修日期:2016-07-06 本文受辽宁省教育厅科学基金(L2013064),中航工业技术创新基金(基础研究类)(2013S60109R)资助。

韩守飞(1990—),男,硕士生,主要研究方向为智能算法优化和云计算安全,E-mail:1492519973@qq.com;李席广(1979—),男,硕士,讲师,主要研究方向为网络安全和云计算安全,E-mail:lixiguang@sau.edu.cn;拱长青(1965—),男,博士,教授,主要研究方向为网络通信和云计算安全,E-mail:gongchangqing@sau.edu.cn。

量越少，且爆炸幅度越大。

一般地，烟花算法由爆炸算子、变异爆炸、映射规则和选择策略 4 个部分组成。

基于以上原则，将烟花算法的基本步骤概括如下：

1. 初始化随机选择 N 个烟花位置；
2. while（当前迭代次数<最大迭代次数）do
3. 对于 N 个烟花；
4. for 所有烟花 x_i do
5. 计算每个烟花产生的火花数 S_i；
6. 计算每个烟花产生火花的幅度 A_i；
7. 随机产生火花；
8. end for
9. for k=1->m do//m 是烟花高斯变异产生的火花数
10. 随机选择一个烟花 x_i 并产生一个火花；
11. end for
12. 依据映射规则对火花进行映射；
13. 应用选择策略得到下一代烟花群体；
14. end while

2 模拟退火

模拟退火（Simulated Annealing，SA）是一种通用概率算法，被用于在一个大的搜寻空间内找寻命题的最优解，可以跳出局部最优解的限制。模拟退火算法是模拟热力学系统中的退火过程，在退火过程中将目标数作为能量函数。大致过程为首先初始化一个高温，温度依据降温的规则缓慢降低，最后终止在设置的最低温度，这时目标函数最小。模拟退火算法也是贪心算法，但是在其过程中引入了随机因素，以一定的概率接受一个比当前解差的解，并且这个概率随着时间的推移而逐渐降低。其伪代码如下：

1. 初始化初始温度 T、初始解 S_0、终止温度 T_f 和最大的迭代次数 N；
2. while(T>T_f,n<N)
3. 产生一个新解 S*；
4. 计算增量 Δt=C(S*)-C(S_0)，其中 C 为评价函数；
5. if Δt<0 则接受当前解为新解；
6. else 产生一个 0 和 1 之间的随机数 r；
7. if(exp[-Δt/T]>r) 接受新解；
8. end if
9. 更新最优解和最优值；
10. n=n+1
11. T=a * T (0<a<1)
12. end while

目前许多学者把模拟退火算法与其他智能算法相结合，使其他算法在收敛速度和精度等方面有所提高，但目前还没有把模拟退火与烟花算法相结合的算法。

3 基于模拟退火与高斯扰动的烟花优化算法

本文将模拟退火思想引入到烟花算法中，提出了一种基于模拟退火与高斯扰动的烟花优化算法（SAFWA）。其基本执行过程为：先随机产生初始种群，设置初始温度，利用模拟退火思想和高斯扰动得到一个当前种群适应值最差的替代值，再根据爆炸算子、变异算子、映射规则以及选择策略选出新一代火花[14]，继续下一步搜索。

基于以上思想，SAFWA 由替换规则、爆炸算子、变异算子、映射规则和选择策略组成。

3.1 替换规则

先初始化烟花种群，找出当前种群中最差的烟花个体，然后引入模拟退火思想，确定初始温度，在每次降温的过程中，引入高斯变异，对当前选出的最差个体的每一维进行高斯变异，即

$$x_i^k = x_i^k g, k=1,2,\cdots,d \tag{1}$$

其中，x_i^k 表示个体 i 的第 k 维，g 是服从均值和方差都为 1 的正态分布，即 $g\sim N(1,1)$，d 表示每个个体设置的维数。

运用模拟退火思想和高斯扰动得到的个体去替换当前种群中最差的个体，实现种群初始化的优化操作。

3.2 爆炸算子

烟花算法初始化随机的 N 个烟花，对于生成的 N 个烟花，应用爆炸算子产生新的火花。爆炸算子是烟花算法的核心，包括爆炸强度、爆炸幅度和位移操作。

3.2.1 爆炸强度

在烟花算法中，产生火花个数的公式为：

$$S_i = m\frac{Y_{max}-f(x_i)+\varepsilon}{\sum_{i=1}^{N}(Y_{max}-f(x_i))+\varepsilon} \tag{2}$$

其中，S_i 为第 i 个烟花产生的火花个数；m 为常数，用来限制产生的火花总数；Y_{max} 是当前种群中适应度最差的个体的适应度值；$f(x_i)$ 为个体 x_i 的适应度值；ε 为一个极小的常数，避免出现分母为零的情况。

3.2.2 爆炸幅度

烟花爆炸幅度范围的计算公式为：

$$A_i = \begin{cases} \hat{A}\dfrac{f(x_i)-Y_{min}+\varepsilon}{\sum_{i=1}^{N}(f(x_i)-Y_{min})+\varepsilon}, & A_i \geqslant A_{min} \\ A_{min}, & \text{其他} \end{cases} \tag{3}$$

其中，A_i 为第 i 个烟花的爆炸幅度范围，即爆炸的火花将在这个范围内随机产生位移，但不能超出这个范围；$\hat{A}$ 为常数，表示最大的爆炸幅度；Y_{min} 是当前种群中适应度最好的个体的适应度值；$f(x_i)$ 为个体 x_i 的适应度值；ε 为一个极小的常数；A_{min} 是最小的爆炸幅度。

3.2.3 位移操作

位移操作是对烟花的每一维进行位移：

$$\Delta x_i^k = x_i^k + rand(0,A_i) \tag{4}$$

其中，$rand(0,A_i)$ 表示在爆炸幅度 A_i 内生成的均匀随机数。

3.3 变异算子

为了保证种群的多样性，在烟花算法中引入了高斯变异。高斯变异的过程如下：首先在烟花种群中随机选择一个烟花，对选择得到的烟花再随机选择一定数量的维度进行高斯变异。在选中的烟花和最好的烟花之间进行高斯变异，产生新的火花。

3.4 映射规则

初始化随机的烟花可能靠近可行域的边界，由于爆炸幅度范围的原因，这些烟花爆炸产生的新烟花可能覆盖到边界

以外的区域。这种烟花对于该算法是无用的，因此需要通过一种规则将其拉回到可行域范围内。映射规则主要包括模运算规则、镜面反射规则和随机映射规则。

这里主要采用模运算映射规则来应对这种情况，其公式为：

$$x_i^k = x_{\min}^k + U(0,1) \cdot (x_{\max}^k - x_{\min}^k) \tag{5}$$

其中，x_i^k 表示超出边界的第 i 个个体在第 k 维上的位置；$x_{\max}^k$ 和 $x_{\min}^k$ 分别表示第 k 维上的边界的上、下界；$U(0,1)$ 表示在 [0,1] 区间上的均匀分布随机数。

3.5 选择策略

运用爆炸算子和变异算子并保证产生的火花在可行域范围之后，需要从产生的火花中选择一部分作为下一代的烟花。烟花算法用到的是精英-随机选择策略。采用以下选择策略来选择进入下一代的个体：每次都留下最优个体，再随机选择其他的 $N-1$ 个个体，即从烟花、爆炸烟花、高斯变异烟花组成的集合中随机选择。

算法的流程图如图 1 所示。

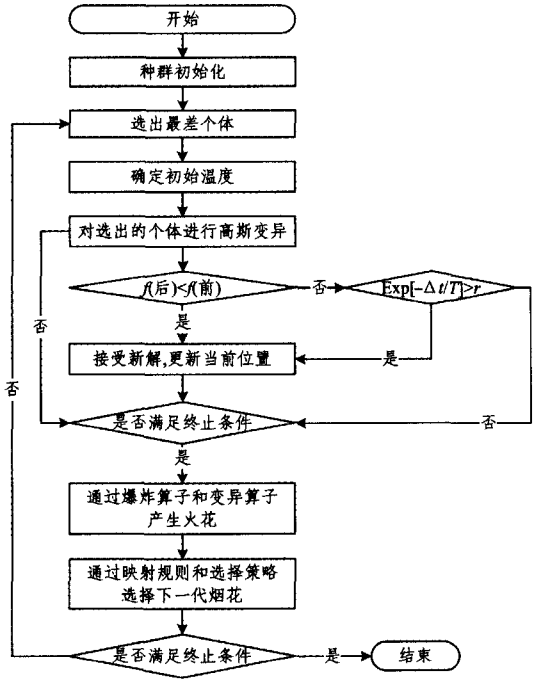


图 1　SAFWA 流程图

通过图 1 可以看出，整个算法可以分成两部分：1）利用模拟退火和高斯扰动找到当前最差的个体的替代值，即精英解；2）利用烟花算法找到下一代迭代的群体，进行进一步搜索。算法的基本步骤如下：

Step 1　随机初始种群的位置、最大火花数、可行域、高斯变异的火花数。

Step 2　找出初始种群中适应值最差的个体，并记录其位置信息。

Step 3　初始化初始温度、终止温度和最大迭代次数。

Step 4　对当前适应值最差的个体进行高斯变异，得到一个新解。

Step 5　比较高斯扰动前、后的适应值。

Step 6　如果扰动后的目标值更优，则更新当前位置为新解的位置；如果扰动前的目标值更优，则以一定的概率接受新解。

Step 7　进行退温操作。

Step 8　若满足停止条件（一般是达到一定精度或者达到一定的迭代次数），则搜索停止，输出结果；否则，转到 Step 4。

Step 9　通过模拟退火和高斯扰动得到一个精英解，用得到的精英解去替换初始种群的最差烟花个体，组成一个新的烟花种群，并用于下一步计算。

Step 10　通过爆炸算子和变异算子产生烟花。

Step 11　对超出界限的烟花运用映射规则，然后通过选择策略得到下一代烟花群体。

Step 12　若满足停止条件（一般是达到一定精度或者达到一定的迭代次数），则搜索停止，输出结果；否则，转到 Step 2。

该算法的伪代码如下：

```
1. 初始化随机选择 N 个烟花位置；
2. while (当前迭代次数<最大迭代次数) do
3.   对于 N 个烟花；
4.     for 1 to N do
5.       找出当前群体最差的烟花；
6.     end for
7.     初始化初始温度 T、终止温度 Tf 和最大的迭代次数 N；
8.     while(T>Tf,n<N)
9.       对当前个体进行高斯变异，得到新解 S*；
10.      计算增量 Δt=C(S*)-C(S0)，其中 C 为评价函数；
11.      if Δt<0 则接受当前解为新解；
12.      else 产生一个 0 和 1 之间的随机数 r；
13.      if(exp[-Δt/T]>r) 接受新解；
14.      end if
15.      更新最优解和最优值；
16.      n=n+1
17.      T=a * T (0<a<1)
18.    end while
19.    for 所有烟花 xi do
20.      计算每个烟花产生的火花数 Si；
21.      计算每个烟花产生火花的幅度 Ai；
22.      随机产生火花；
23.    end for
24.    for k=1->m do //m 是烟花高斯变异产生的火花数
25.      随机选择一个烟花 xi 并产生一个火花；
26.    end for
27.    依据映射规则对火花进行映射；
28.    应用选择策略得到下一代烟花群体；
29. end while
```

4　仿真实验

为了比较分析 SAFWA 的性能，本文将 SAFWA 算法、

FWA 算法、EFWA 算法以及 SPSO 算法[16]用于 10 个典型的基准测试函数中，并对实验结果进行比对，以直观呈现 4 种算法的性能。

4.1 实验设计

4 种算法的参数设置如表 1 所列。

表 1 参数设置

算法	参数设计
SPSO	粒子群大小为 40，权重设置为 0.72984，$c1$ 为 2.05，$c2$ 为 2.05
FWA	初始烟花个数为 5，烟花总数 50，高斯变异个数为 5，爆炸幅度为 40
EFWA	初始烟花个数为 5，烟花总数 50，高斯变异个数为 5，爆炸幅度为 40，初始爆炸半径系数为 0.02，最终爆炸半径系数为 0.001
SAFWA	初始烟花个数为 5，烟花总数 50，高斯变异个数为 5，爆炸幅度为 40，退火常数为 0.99

本实验采用的 10 个典型的基准测试函数均来自全局优化测试函数库。函数的基本特征如表 2 所列。为了使图表更加清晰，用 $f_1(x)$ 表示 Six-Hump Camel-back 函数，$f_2(x)$ 表示 Axis parallel hyper ellipsoid 函数，$f_3(x)$ 表示 Rotated hyper ellipsoid 函数。* 表示该函数有不止一个最优点。

表 2 基准测试函数的基本特征

函数名	维数	搜索范围	最优点	最优值
Sphere	30	[−100,100]	(0,…,0)	0
Schwefel	30	[−100,100]	(0,…,0)	0
Rosenbrock	30	[−30,30]	(1,…,1)	0
Ackley	30	[−32,32]	(0,…,0)	0
Griewank	30	[−600,600]	(0,…,0)	0
Rastrigin	30	[−5.12,5.12]	(0,…,0)	0
$f_1(x)$	2	[−5,5]	*	−1.0316285
Goldstein-Price	2	[−2,2]	(0,−1)	3
$f_2(x)$	30	[−5.12,5.12]	(0,…,0)	0
$f_3(x)$	30	[−65.536,65.536]	(0,…,0)	0

由于在测试函数中许多函数的最优点在原点位置附近，而基本烟花算法具备这一特点，即具有非常强大的在原点位置附近搜索的能力。因此，实验在该测试集合上增加了位置偏移。位置偏移的大小与优化问题的搜索范围有关，如表 3 所列，其中，Max 与 Min 为优化问题的搜索范围，SI 表示偏移索引，SV 表示偏移值大小。

表 3 基准测试函数的位置偏移描述

SI	SV
0	0
1	$0.05\times\frac{Max-Min}{2}$
2	$0.1\times\frac{Max-Min}{2}$
3	$0.2\times\frac{Max-Min}{2}$
4	$0.3\times\frac{Max-Min}{2}$
5	$0.5\times\frac{Max-Min}{2}$
6	$0.7\times\frac{Max-Min}{2}$

为了评估算法的收敛性能，最大的函数评估次数设置为 300000，连续运行 20 次，分别进行 3 次实验。

4.2 实验结果分析

在测试函数集合上进行性能测试，主要通过均值综合的排名来比较不同算法之间的性能。为了验证不同算法在测试函数上的差异显著性，通常使用 T 检验[1]。对此，本文分别进行了 3 次实验来比较 SPSO，FWA，EFWA 和 SAFWA 之间的性能差异。

4.2.1 SPSO，FWA，EFWA 和 SAFWA 的位置偏移曲线的对比

设定相同的迭代次数，对以上 10 个基准函数进行仿真实验，如图 2—图 11 所示。图中横坐标表示位置偏移索引，纵坐标表示平均适应值。由于有的坐标太小，图会产生重合的现象，因此在每幅图下方给出上了函数最优值的表格，如表 4—表 13 所列。

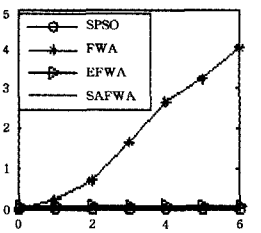


图 2 Sphere 函数曲线对比

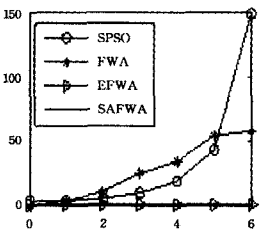


图 3 Schwefe 函数进化曲线对比

表 4 Sphere 函数全局最优平均值的对比（SI 为偏移索引）

算法	SI=0	SI=1	SI=2	SI=3	SI=4	SI=5	SI=6
SPSO	0	0	0	0	0	0	0
FWA	0	0.2348	0.7096	1.666	2.6764	3.2552	4.0253
EFWA	0.0721	0.0661	0.0686	0.0693	0.0648	0.0728	0.0766
SAFWA	0.0032	0.0066	0.0067	0.0087	0.0065	0.0071	0.0082

表 5 Schwefe 函数全局最优平均值的对比（SI 为偏移索引）

算法	SI=0	SI=1	SI=2	SI=3	SI=4	SI=5	SI=6
SPSO	3.721	3.798	5.619	9.738	18.24	42.15	148.71
FWA	0	3.7453	10.123	24.61	33.87	53.28	57.393
EFWA	0.1774	0.185	0.1919	0.1852	0.1872	0.1884	0.1692
SAFWA	0.0475	0.0713	0.0775	0.0737	0.075	0.0774	0.0829

由图 2 和表 4 可知，随着函数位置偏移的变大，相对于 FWA 和 EFWA，SAFWA 取得了更加明显的性能优势，但是 SPSO 的性能比 SAFWA 的性能更好。

由图 3 和表 5 可知，随着函数位置偏移的不断变大，相对于 EFWA 和 SAFWA，SPSO 和 FWA 的性能很差，且全局最优值也在不断变大，稳定性变差。SAFWA 和 EFWA 都具有稳定性，但是 SAFWA 比 EFWA 具有更好的收敛性。

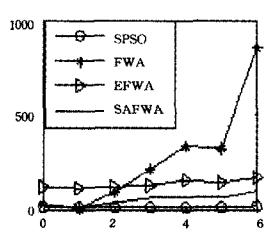


图 4 Rosenbrock 函数进化曲线对比

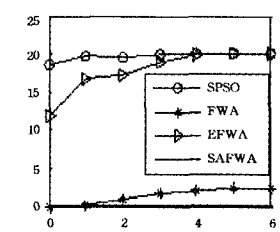


图 5 Ackley 函数进化曲线对比

表 6 Rosenbrock 函数全局最优平均值的对比（SI 为偏移索引）

算法	SI=0	SI=1	SI=2	SI=3	SI=4	SI=5	SI=6
SPSO	9.3145	11.1278	12.258	10.721	12.289	13.864	16.565
FWA	25.557	3.8475	88.025	210.12	335.25	323.34	851.63
EFWA	120.26	110.71	119.43	124.39	158.27	138.28	165.99
SAFWA	27.853	7.7656	30.149	58.489	69.715	62.098	93.5629

表 7　Ackley 函数全局最优平均值的对比（*SI* 为偏移索引）

算法	*SI*=0	*SI*=1	*SI*=2	*SI*=3	*SI*=4	*SI*=5	*SI*=6
SPSO	18.536	19.756	19.528	19.931	19.986	20.001	20.1145
FWA	0	0.2437	0.987	1.737	2.033	2.358	2.4182
EFWA	11.868	16.734	17.209	18.924	19.962	19.989	19.9975
SAFWA	0.0130	0.0234	0.0223	0.0214	0.0209	0.0249	0.0229

由图 4 和表 6 可知，随着函数位置偏移的变大，4 种算法的全局最优值都在不断变大，但是相比其他 3 种算法，SPSO 具有更好的性能；相对于 EFWA 和 FWA，SAFWA 具有更好的收敛性和稳定性。

由图 5 和表 7 可知，随着函数的位置偏移变大，相对于 SPSO，EFWA 和 FWA，SAFWA 具有更好的收敛性和稳定性。

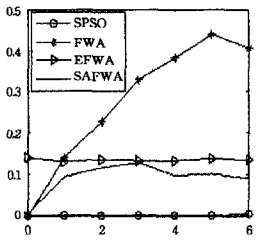


图 6　Griewank 函数进化曲线对比

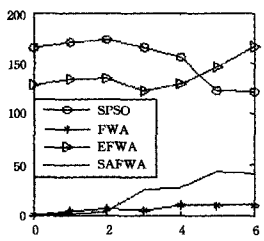


图 7　Rastrigin 函数进化曲线对比

表 8　Griewank 函数全局最优平均值的对比（*SI* 为偏移索引）

算法	*SI*=0	*SI*=1	*SI*=2	*SI*=3	*SI*=4	*SI*=5	*SI*=6
SPSO	0.0004	0.0008	0.0005	0.0009	0.0011	0.0020	0.00612
FWA	0	0.1406	0.2282	0.3274	0.3814	0.4388	0.40572
EFWA	0.1398	0.131	0.1361	0.1329	0.1346	0.1377	0.1365
SAFWA	0.0072	0.0942	0.1176	0.1276	0.0981	0.1026	0.0909

表 9　Rastrigin 函数全局最优平均值的对比（*SI* 为偏移索引）

算法	*SI*=0	*SI*=1	*SI*=2	*SI*=3	*SI*=4	*SI*=5	*SI*=6
SPSO	165.1500	170.2300	172.6300	165.4100	155.6200	122.1300	120.5900
FWA	0	3.4446	6.7645	4.4409	8.8902	9.0594	9.6911
EFWA	128.1800	133.8500	135.2600	122.1800	128.9500	145.2300	166.5200
SAFWA	0.0025	0.3028	3.0395	25.4750	26.7690	42.7880	40.5991

由图 6 和表 8 可知，随着函数位置偏移的变大，相对于 EFWA 和 FWA，SAFWA 具有更好的收敛性和稳定性。相对于 SAFWA，SPSO 具有更好的性能优势。

由图 7 和表 9 可知，随着函数位置偏移的变大，相对于其他 3 种算法，FWA 具有更好的收敛性。SAFWA 在收敛性和稳定性上比 EFWA 和 SPSO 更具优势。

由图 8 和表 10 可知，随着函数位置偏移的变大，SAFWA 与其他 3 种算法的收敛性和稳定性都一样。

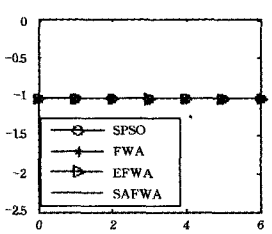


图 8　$f_1(x)$函数进化曲线对比

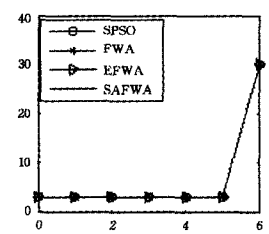


图 9　Goldstein-Price 函数进化曲线对比

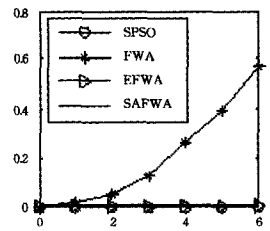
表 10　$f_1(x)$函数全局最优平均值的对比（*SI* 为偏移索引）

算法	*SI*=0	*SI*=1	*SI*=2	*SI*=3	*SI*=4	*SI*=5	*SI*=6
SPSO	−1.032	−1.032	−1.032	−1.032	−1.032	−1.032	−1.032
FWA	−1.032	−1.032	−1.032	−1.032	−1.032	−1.032	−1.032
EFWA	−1.032	−1.032	−1.032	−1.032	−1.032	−1.032	−1.032
SAFWA	−1.031	−1.031	−1.031	−1.032	−1.032	−1.032	−1.032

表 11　Goldstein-Price 函数全局最优平均值的对比（*SI* 为偏移索引）

算法	*SI*=0	*SI*=1	*SI*=2	*SI*=3	*SI*=4	*SI*=5	*SI*=6
SPSO	3	3	3	3	3	3	30
FWA	3.0001	3.0002	3.0005	3.0003	3.0001	3.00012	30.0013
EFWA	3	3	3	3	3	3	30
SAFWA	3	3	3	3	3	3	30

由图 9 和表 11 可知，随着函数位置偏移的变大，SAFWA 和 SPSO 以及 EFWA 算法的收敛性和稳定性都一样，比 FWA 性能好。



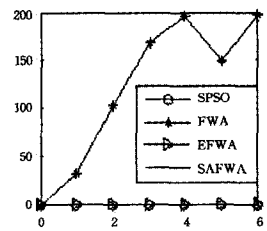


图 10　$f_2(x)$函数进化曲线对比　　图 11　$f_3(x)$函数进化曲线对比

表 12　$f_2(x)$函数全局最优平均值的对比（*SI* 为偏移索引）

算法	*SI*=0	*SI*=1	*SI*=2	*SI*=3	*SI*=4	*SI*=5	*SI*=6
SPSO	0	0	0	0	0	0	0
FWA	0	0.017	0.0531	0.125	0.2608	0.392	0.5702
EFWA	0.0031	0.0031	0.0031	0.0029	0.0032	0.0029	0.0029
SAFWA	1.27e−4	3.11e−4	3.48e−4	3.7e−4	3.1e−4	3.4e−4	3.3e−4

表 13　$f_3(x)$函数全局最优平均值的对比（*SI* 为偏移索引）

算法	*SI*=0	*SI*=1	*SI*=2	*SI*=3	*SI*=4	*SI*=5	*SI*=6
SPSO	0	0	0	0	0	0	0
FWA	0	32.257	103.02	167.71	195.91	148.6	197.4
EFWA	0.4127	0.4741	0.4824	0.4829	0.4171	0.4717	0.4824
SAFWA	0.0967	0.3414	0.36196	0.3566	0.31969	0.3952	0.3507

由图 10 和表 12 可知，随着函数位置偏移的变大，相对于 FWA 和 EFWA，SAFWA 取得了更加明显的性能优势，但是性能不及 SPSO。

由图 11 和表 13 可知，随着函数位置偏移的变大，相对于 FWA 和 EFWA，SAFWA 取得了更加明显的性能优势；相对于 SAFWA，SPSO 具有更好的收敛性和稳定性。

4.2.2　SPSO，FWA，EFWA 和 SAFWA 在测试函数上的平均排名

为了进一步研究 SAFWA 算法的性能，在同等实验条件下，设置最大函数评估次数为 30000，连续运行 20 次。取 20 次得出的全局最优值的综合平均值的排名作对比，结果如表 14 所列。

由表 14 的平均排名结果可知，SAFWA 在 4 种算法中排名第一，远远高于 EFWA 和 FWA 的排名，也比 SPSO 的排名更高，即基于模拟退火与高斯扰动的烟花优化算法的搜索能力强于 FWA，EFWA 和 SPSO。

表 14 SPSO,FWA,EFWA 和 SAFWA 在测试函数上的平均排名

函数名	SPSO	FWA	EFWA	SAFWA
Sphere	1	4	3	2
Schwefel	4	3	2	1
Rosenbrock	1	3	4	2
Ackley	4	2	3	1
Griewank	1	4	3	2
Rastrigin	3	2	4	1
$f_1(x)$	1	1	1	1
Goldstein-Price	1	2	1	1
$f_2(x)$	1	4	3	2
$f_3(x)$	1	4	3	2
平均排名				
	1.8	2.9	2.7	1.5

4.2.3 统计检验

4 种算法在 10 个基准函数独立运行 20 次,因而在每个测试函数上每种算法都有 20 个样本。本文采用 T 检验来比较分析 FWA 和 SAFWA 的性能。该 T 检验采用自由度为 20、显著水平为 0.05 的双尾检验,其结果如表 15 所列。

表 15 SAFWA 算法和 FWA,EFWA,SPSO 的 T 检验

函数名	FWA	EFWA	SPSO
Sphere	+	+	−
Schwefel	+	+	+
Rosenbrock	+	+	−
Ackley	+	+	+
Griewank	+	+	−
Rastrigin	−	+	+
$f_1(x)$	=	=	=
Goldstein-Price	+	=	=
$f_2(x)$	+	+	−
$f_3(x)$	+	+	−

表 15 的检验结果中,“+”表示 SAFWA 算法在相应的测试函数上的效果优于该列算法,“=”表示两种算法在该行的测试函数上的效果一致,“−”表示 SAFWA 算法在这行的测试函数上的效果不如该列算法。

从表 15 可以看出,相对于 FWA,SAFWA 除了在函数 Rastrigin 上的效果没有 FWA 好,在函数 Six-Hump Camel-back 上两种算法的效果一样,在其他函数上的效果均优于 FWA;相对于 EFWA,SAFWA 在函数 Six-Hump Camel-back 和 Goldstein-Price 上的效果一样,除此之外,在其他函数上其效果均优于 EFWA 算法;对于 SAFWA 和 SPSO,两种算法在 10 个测试函数上的效果有好有坏,再结合表 14 的排名情况,SAFWA 在总体上的效果和稳定性比 SPSO 好。所以从统计的角度可得出结论:SAFWA 算法在大部分测试问题上均优于 FWA,EFWA 和 SPSO。

结束语 对烟花算法的改进研究主要有两类[1],研究烟花算法与其他启发式算法相结合是其中一类。本文将模拟退火算法与烟花算法相结合,对适应值最差的烟花进行高斯扰动,从而得到一个比最差的烟花个体更好的精英解。随着温度的逐渐降低,接受差解的几率越来越小,从而提高了算法的收敛性能。通过对 10 个基准测试函数的仿真实验表明,SAFWA 算法的收敛性能和计算精度不同程度地优于 FWA 算法、EFWA 算法和 SPSO 算法;且随着位移位置的不断增大,相对于其他 3 种算法,SAFWA 具有更好的稳定性。

参 考 文 献

[1] TAN Y,ZHENG S Q. Recent advances in fireworks algorithm [J]. CAAI Transactions on Intelligent Systems,2014,9(5):515-528. (in Chinese)
谭营,郑少秋. 烟花算法研究进展[J]. 智能系统学报,2014,9(5):515-528.

[2] ZHENG S,JANECEK A,TAN Y. Enhanced fireworks algorithm[C]// 2013 IEEE Congress on Evolutionary Computation (CEC). 2013.

[3] ZHENG Y J,XU X L,LING H F,et al. A hybird fireworks optimization method with differential evolution operators[J]. Neurocomputing,2015,148:75-80.

[4] YU C,KELLEY L,ZHENG S,et al. Fireworks algorithm with differential mutation for solving the CEC 2014 Competition problems[C]// 2014 IEEE Congress on Evolutionary Computation. 2014:3238-3245.

[5] GAO H,DIAO M. Cultural fireworks algorithm and its Application for digital filters design[J]. International Journal of Modelling,Identification and Control,2011,14(4):324-331.

[6] ZHANG B,ZHANG M X,ZHENG Y J. A hybird bi-Geography-based optimization and fireworks algorithm[C]// 2014 IEEE Congress on Evolutionary Computation. 2014:3200-3206.

[7] TAN Y,YU C,ZHENG S,et al. Introduction to fireworks algorithm [J]. International Journal of Swarm Intelligence Research,2013,4(4):39-70.

[8] TAN Y,ZHU Y. Fireworks algorithm for optimization [M]. Berlin Springer,2010:355-364.

[9] ZHOU Y,TAN Y. GPU-based parallel particle swarm optimization[C]// IEEE Congress on Evolutionary Computation. 2009:1493-1500.

[10] LIU J,ZHENG S,TAN Y. Analysis on global convergence and time complexity of fireworks algorithm[C]// 2014 IEEE Congress on Evolutionary Computation(CEC). Beijing,China,2014:3207-3213.

[11] CHEN H G,WU J S,WANG J L. Mechanism study of simulated annealing algorithm[J]. Journal of Tongji University(Natural Science),2004,32(6):802-806. (in Chinese)
陈华根,吴建生,王家林. 模拟退火算法机理研究[J]. 同济大学学报(自然科学版),2004,32(6):802-806.

[12] HE X S,DING W J,YANG X S. Bat algorithm based on simulated annealing and Gaussian perturbations[J]. Application Research of Computers,2014,31(2):392-397. (in Chinese)
贺兴时,丁文静,杨新社. 基于模拟退火高斯扰动的蝙蝠优化算法[J]. 计算机应用研究,2014,31(2):392-397.

[13] ZHANG Y,HE X S,YANG X S. A cuckoo search algorithm based on simulated annealing and Gaussian disturbance [J]. Basic Sciences Journal of Textfile Universites,2015,28(4):515-521. (in Chinese)
张毅,贺兴时,杨新社. 基于模拟退火高斯扰动的布谷鸟算法[J]. 纺织高校基础科学学报,2015,28(4):515-521.

[14] 谭营. 烟花算法引论[M]. 北京:科学出版社,2015.

[15] 藤素珍,冯敬海. 数理统计学[M]. 大连:大连理工大学出版社,2005.

[16] BRATTON D,KENNEDY J. Defining a standard for particle swarm optimization[C]// Swarm Intelligence Symposium. 2007.