计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(4):196−201 [doi: 10.15888/j.cnki.csa.006312]
©中国科学院软件研究所版权所有.
E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

融合混沌反学习与蜂群搜索算子的引力搜索算法①

丁知平

(清远职业技术学院 信息技术与创意设计学院, 清远 511510)

摘　要: 引力搜索算法是最近提出的一种较有竞争力的群智能优化技术, 然而, 标准引力算法存在的收敛速度慢、容易在进化过程中陷入停滞状态. 针对上述问题, 提出一种改进的引力搜索算法. 该算法采用混沌反学习策略初始化种群, 以便获得遍历整个解空间的初始种群, 进而提高算法的收敛速度和解的精度. 此外, 该算法利用人工蜂群搜索策略很强的探索能力, 对种群进行引导以帮助算法快速跳出局部最优点. 通过对 13 个非线性基准函数进行仿真实验, 验证了改进的引力搜索算法的有效性和优越性.

关键词: 引力搜索算法; 人工蜂群算法; 混沌反学习; 数值实验; 函数优化

引用格式: 丁知平.融合混沌反学习与蜂群搜索算子的引力搜索算法.计算机系统应用,2018,27(4):196–201. http://www.c-s-a.org.cn/1003-3254/6312.html

Improved Gravitational Search Algorithm with Chaotic Opposition-Based Learning and Artificial Bee Colony Search Operator

DING Zhi-Ping

(College of Information Technology and Creative Design, Qingyuan Polytechnic, Qingyuan 511510, China)

Abstract: The gravitational search algorithm (GSA) is a relatively novel swarm intelligence optimization technique which has been shown to be competitive to other population-based intelligence optimization algorithms. However, there is still an insufficiency that is the low convergence speed of the standard gravitational search algorithm, and its being stalled easily in the evolutionary process. Considering those problems, an improved gravitational search algorithm is presented. A strategy of chaotic opposition-based learning is employed to generate an initial population, which makes it possible for the algorithm to achieve a better initial population, thus accelerating the convergence speed. In addition, the method makes full use of the exploration ability of the search strategy of artificial bee colony algorithm to guide the algorithm to jump out of the likely local optima. The results of numerical simulation experiment on a suite of 13 benchmark functions demonstrate the effectiveness and superiority of the improved gravitational search algorithm.

Key words: gravitational search algorithm; artificial bee colony algorithm; chaotic opposition-based learning; numerical experiment; function optimization

引言

引力搜索算法 (Gravitational Search Algorithm, GSA)[1]是继遗传算法[2]、粒子群算法[3,4]、混合蛙跳算法[5]之后的新的群智能优化算法. GSA 算法源于对物理学中万有引力定律进行模拟而提出, 它为解决复杂优化问题提供了新的思路和手段. 由于具有结构简单、易于实现、参数较少等优点, GSA 算法一经提出便受到众多学者的关注和研究, 并取得广泛应用[6–8].

在处理部分复杂、多维优化问题时, 典型 GSA 算法容易陷入局部极值、收敛速度慢等不足. 文献[9]提

① 基金项目: 广东省高等学校优秀青年教师培养对象项目; 清远市科技计划项目 (2016B002)
收稿时间: 2017-08-07; 修改时间: 2017-08-28; 采用时间: 2017-09-04; csa 在线出版时间: 2018-03-31

© 中国科学院软件研究所 http://www.c-s-a.org.cn

出将反向学习策略、边界变异等策略引入算法, 起到调节算法全局搜索和局部搜索能力的作用, 获得了较好的优化能力. 文献[10]利用免疫信息处理机制来改善引力搜索算法的局部优化性能, 明显改善了 GSA 易陷入局部极值的问题. 文献[11]提出一种基于权值的 WGSA 算法来提高标准 GSA 算法的性能, 使得算法在进化初期搜索能力较强. 上述改进策略仍然受到初始种群的影响, 并且, 随着优化问题规模和复杂程度不断扩大, 这些改进策略均很难在提高算法收敛速度和避免早熟收敛两方面取得平衡.

针对典型引力搜索算法寻优能力不足的问题, 本文提出了一种改进的引力搜索算法 (GSA with Chaotic Opposition-based Learning and Artificial Bee Colony Operator, CA-GSA) 以改善 GSA 算法的寻优能力. 首先采用混沌反学习的初始化种群以加快算法的全局收敛速度; 然后, 在 GSA 中引入人工蜂群搜索算子对种群进行引导搜索, 从而使种群中的个体尽快跳出局部最优点, 平衡 GSA 算法的探索和开发能力, 避免种群因个体陷入局部最优而导致早熟收敛的不足. 最后, 通过 13 个基准测试函数进行仿真测试, 验证了 CA-GSA 算法的有效性和优越性.

1 引力算法

假设由 N 个质点组成的种群, 其中第 i 个质点的位置为 $X_i=(x_i^1,\cdots,x_i^d,\cdots,x_i^s)$, $i=1,\cdots,N$, x_i^d 表示质点 i 在 d 维空间的位置, s 为优化问题的维数.

依据牛顿引力定理, 在 d 时刻, 第 i 个质点收到第 j 个质点的作用力 F 为

$$F_{ij}^d(t)=G(t)\frac{M_i(t)\times M_j(t)}{R_{ij}(t)+\varepsilon}\left(x_j^d(t)-x_i^d(t)\right) \tag{1}$$

其中, M_i和M_j分别为质点 i、j 的质量; ε为较小的常数; $R_{ij}(t)=\|\mathbf{X}_i(t),\mathbf{X}_j(t)\|_2$; $G(t)$为引力常数.

则作用在第 i 个质点的合力为

$$F_i^d(t)=\sum_{j=1,j\neq i}^{N} rand_j F_{ij}^d(t) \tag{2}$$

式中, $rand_j$是[0,1]中的一个随机数.

基于牛顿第二定理, 质点 i 在 d 维的加速度为

$$a_i^d(t)=F_i^d(t)/M_i(t) \tag{3}$$

在 GSA 算法的每一次迭代过程中, 按式 (4) 更新质点的速度v_i^d, 按式 (5) 更新质点的位置x_i^d:

$$v_i^d(t+1)=rand_i\times v_i^d(t)+a_i^d(t) \tag{4}$$

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1) \tag{5}$$

GSA 是以适应度值为基础的, 依据适应度函数 fit_i给出质点的质量.

2 改进的引力算法

2.1 混沌反学习种群初始化方法

GSA 算法开始于初始解, 然后朝着改善解的方向进行优化, 因此种群初始化方法是算法设计中一项至关重要的环节, 其优劣程度会影响算法的收敛速度和解的精度. 在缺乏有关最优解的先验信息时, GSA 通常采用随机初始种群. 然而, 如果初始解远离最优解, 甚至最优解在初始解的对立解的周围, 随机初始化的方式将会制约算法的收敛性能. 反向学习策略 (Opposition-Based Learning, OBL)[12]是计算智能领域出现的一种新技术并取得广泛应用, 其主要思想是: 对一个可行解, 同时计算并评估其反向解, 从中选择较优的解作为下一代个体. 因此, 采用 OBL 方法进行种群初始化可以改善初始种群的质量. 除此之外, 种群初始化尽可能均匀分布在可能的解空间, 可以有效地提高寻找最优解的效率. 利用混沌算法的遍历性产生分布均匀的初始种群可以得到质量较好的初始解群[13], 进一步提高提高 GSA 算法寻优的计算效率.

式 (6) 所示为 1 维混沌自映射的表达式:

$$\begin{aligned} &x_{n+1}=\sin(2/x_n),\\ &n=0,1,2,\cdots,N,\ -1\le x_n\le 1,\ x_n\neq 0 \end{aligned} \tag{6}$$

式 (6) 中, 混沌变量x_n在其相空间内总会有线性或非线性折叠现象以产生混沌, 其中, 混沌扰动的幅度为 $[-1,1]$, 且迭代初值 $x_0\neq 0$, 本文方法的初值设置 $x_0=0.255$, 经过多次混沌迭代, 系统输出将遍历整个解空间.

混沌反学习初始化种群算法描述如下所示.

算法. 混沌反学习初始化种群算法

/* 混沌阶段 */

1) 初始化种群规模 N 和混沌迭代步数$K\ge 300$;

2) 随机产生混沌变量初值$ch(0)$, $-1\le ch(0)\le 1, ch(0)\neq 0$, 令$k=1$;

3) 按照公式$ch(k+1)=\sin\frac{2}{ch(k)}$进行混沌运动, 直到$k=K$, 产生 K 个混沌变量;

4) 基于混沌变量$ch(k)$产生 N 个初始化种群:$X(n)=X_{\min}(n)+ch(n).*(X_{\max}(n)-X_{\min}(n))$, $1\le n\le N$; $X_{\min}(n)$和$X_{\max}(n)$为待优化问题变量的最小值和最大值;

/* 反向学习阶段 */

© 中国科学院软件研究所 http://www.c-s-a.org.cn

5) 按照反向学习模型, 生成 N 个反向解$\{X^*(N)\}$: $X^*(n)=X_{\max}(n)+X_{\min}(n)-X(n)$, $1\le n\le N$;
6) 计算$\{X(N)\cup X^*(N)\}$集合中 $2N$ 个质点的适应度值;
7) 选择适应度值最好的 N 个质点作为 GSA 算法的初始种群.

结合二者优势, 提出混沌反学习算法来初始化 GSA 的种群. 基于混沌反向学习初始化方法, 按照混沌运动自身规律和特性进行, 无疑会比随机种群更优越, 因而优化到最优解的可能性也越大. 下面介绍算法具体步骤.

2.2 人工蜂群搜索策略

人工蜂群算法 (Artificial Bee Colony Algorithm, ABC) 是源于蜜蜂采蜜而提出的智能算法[14,15], ABC 算法具有较好的优化性能, 这主要得力于该算法具有很强的探索能力. 文献[16]在原始 ABC 算法的基础上给出了一个新的搜索策略, 即

$$v_{ij}=x_{ij}w_{ij}+2\left(\phi_{ij}-0.5\right)\left(x_{ij}-x_{kj}\right)\Phi_1+\varphi_{ij}\left(x_g-x_{kj}\right)\Phi_2 \tag{7}$$

其中, w_{ij}是惯性权值, 其大小反映了对候选解x_{ij}的依赖程度; x_g为最优位置; ϕ_{ij}和φ_{ij}是[0,1]之间的随机数; Φ_1和Φ_2是控制最大步长的加速因子; x_{kj}是随机选择的个体.

可以看出, 新的搜索算子在随机参数和最优位置x_g的引导下, 在保证探索能力同时能够提高开发能力. 当处理复杂优化问题时, GSA 算法由于探索能力较弱而致使算法易陷入局部最优点, 进一步引导其余个体搜索轨迹向局部最优点靠近而出现早熟现象. 因此, 如何提高 GSA 的探索能力以帮助算法跳出局部最优是改善算法优化性能的关键. 本文结合 ABC 算法强的探索能力, 采用最优候选解x_g帮助 GSA 种群中陷入局部最优的质点快速跳出局部最优, 避免算法早熟.

2.3 改进的引力搜索算法

针对典型引力搜索算法寻优能力不足的问题, 提出一种改进的 GSA(CA-GSA) 算法. CA-GSA 是在标准 GSA 算法基础上, 引进了混沌反学习和人工蜂群搜索算子, 使 CA-GSA 在继承典型 GSA 算法出色的搜索能力下兼具上述两种策略的优势, 以达到改善 GSA 算法优化性能的目的. CA-GSA 算法的程序执行流程如图 1 所示. 具体步骤如下:

1) 初始化参数: 种群规模 N=50、混沌迭代步数 K=300、GSA 算法的迭代次数 NS=1000;

2) 初始化种群: 在待优化问题的空间中, 采用混沌反学习算法产生 N 个质点组成的种群;

3) 计算适应度值, 更新种群中最优质点 x_g;

4) 更新数据$G(t)$, $best(t)$, $worst(t)$和M_i;

5) 计算质点在所受的合力及加速度a_i^d;

6) 依据式 (7) 更新v_{ij}^d;

7) 按式 (5) 更新质点的位置;

8) 重复步骤 3) 到步骤 8), 直到满足终止条件;

9) 输出最优解及最优值, 运算结束.

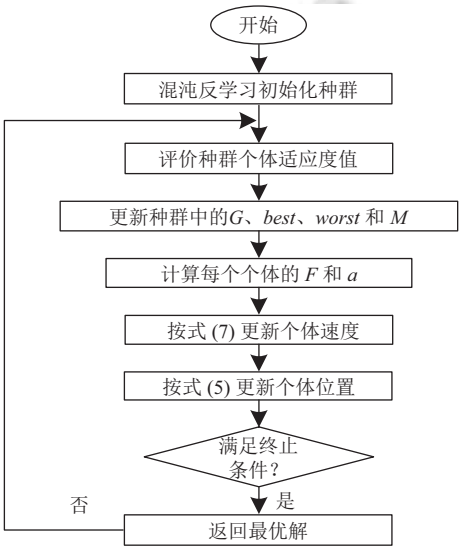

图 1 CA-GSA 程序流程图

3 仿真实验

为了验证 CA-GSA 的效果, 对表 1 中 13 个标准基准函数进行仿真实验. 表 1 给出了基准函数的搜索空间和最优解, 其中 f_1~f_7 为单模态函数, 主要用来考察算法的执行能力并测试算法的寻优精度; f_8~f_{13} 是优化领域中公认较难优化的多模态函数, 具有大量的局部最优点, 它们主要用来检验算法是否具备避免早熟并搜索到全局最优解的能力.

为了说明 CA-GSA 算法的有效, 将 CA-GSA 与标准 GSA、基于人工蜂群搜索算子的 GSA(记为 A-GSA) 进行对比研究. 为了比较的公平性, 3 种算法的迭代次数为 1000, 种群规模为 50. 为有效减少随机干扰的影响, 均在相同条件下独立运行 30 次. 表 2 记录了 3 种算法测试基准函数分别在 30 维和 50 维情况下的平均收敛次数 (C.I)、最优解 (Best)、平均最优适应度值 (Mean) 和标准差 (SD) 的统计数据. C.I 反映了算法的收敛速度, Best 和 Mean 显示了在给定函数评价次数下算法所能达到的精度, SD 反映了算法的稳定性.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

表 1 测试函数

函数名称	搜索范围	最小值
$f_1=\sum_{i=1}^{n}x_i^2$	$[-100,100]^n$	0
$f_2=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]^n$	0
$f_3=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	$[-100,100]^n$	0
$f_4=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	$[-100,100]^n$	0
$f_5=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30,30]^n$	0
$f_6=\sum_{i=1}^{n}([x_i+0.5])^2$	$[-100,100]^n$	0
$f_7=\sum_{i=1}^{n}ix_i^4+random[0,1)$	$[-1.28,1.28]^n$	0
$f_8=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]^n$	–418.98*n
$f_9=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]^n$	0
$f_{10}=-20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32,32]^n$	0
$f_{11}=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]^n$	0
$f_{12}=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4};\quad u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\0 & -a<x_i<a\\k(-x_i-a)^m & x_i<-a\end{cases}$	$[-50,50]^n$	0
$f_{13}=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	$[-50,50]^n$	0

从表 2 可以看出, 无论是解的精度还是收敛速度, A-GSA 和 CA-GSA 算法比标准 GSA 算法均有很大提高. 仿真结果可以看出, CA-GSA 算法几乎在所有基准函数上取得了最好的优化结果. 所有函数的最优值、平均值都等于或非常接近于全局最优值, 而且标准差都相当小. 具体地, 对于 f_1、f_2、f_3、f_4、f_9、f_{11}, 均搜索到了全局最优值. 特别地, 当 n=50, CA-GSA 的优化精度和标准差基本上接近 n=30 的精度和标准差. 进一步表明了 CA-GSA 的优化效果、稳定性和鲁棒性并没有随着问题复杂程度的增加而减弱. GSA 和 A-GSA 随着维数的增加, 绝大部分的优化效果都呈现下降趋势. 尤其是 GSA 算法, 从表 2 可以看出, 虽然标准 GSA 算法对大部分基准函数能够优化到较好的结果, 但随着问题复杂程度的增加, 其稳定性和优化精度下降明显.

相比 GSA 算法, A-GSA 除了在 f_6 和 f_{12} 函数上性能略逊于标准 GSA 算法, 在其他测试函数上, A-GSA 优化精度等于或非常接近全局最优值且优于 GSA 算法. 尤其是 f_8, GSA 无论在 30 维还是 50 维, 其解均陷入局部最优, 而 A-GSA 算法则能跳出局部最优, 获得较为满意的解. 究其原因, 是因为 A-GSA 算法引入了人工蜂群搜索算子, 增强了算法的探索能力, 进而增加了算法获得全局最优解的能力.

比较 A-GSA 和 CA-GSA 的性能, A-GSA 和 CA-GSA 的区别在于 A-GSA 算法中没有采取混沌反学习进行种群初始化. 对于 f_1、f_3、f_9、f_{10}、f_{11}, 二者虽然都能获得相同的优化结果, 但 CA-GSA 的收敛速度较 A-GSA 的收敛速度更快; 对于其余函数, CA-GSA 算法在全局收敛速度和优化精度上均优于 A-GSA. 上述表明, 混沌反向学习种群初始化方法能够改善 GSA 算法的寻优能力.

为更直观地反映算法的寻优效果, 将 CA-GSA、A-GSA 和标准 GSA 算法进行比较. 3 种算法对测试函数 f_1、f_4、f_5、f_9、f_{11}、f_{13}6 个函数在 Dim=30 情况下的寻优曲线如图 2 所示. 图 2 可以看出, 针对单峰测试函数 f_1、f_4、f_5, CA-GSA 具有较快的收敛速度和更优秀的解的精度; A-GSA 算法由于没有采用混沌反学习测量, 影响了收敛速度, 由于采用了蜂群搜索算子, 其优化性能明显强于标准 GSA 算法. 针对多峰测试函数,

© 中国科学院软件研究所 http://www.c-s-a.org.cn

由于采用了混沌反学习的初始化种群策略和蜂群搜索算子, CA-GSA 算法的优化性能明显优于 A-GSA 和 GSA 算法. 综上所述, CA-GSA 算法在优化性能上得到了明显改善.

表 2 3 种优化算法性能比较

函数	算法	D=30				D=50			
		C.I	Best	Mean	SD	C.I	Best	Mean	SD
f_1	GSA	1000	9.6397×10^{-18}	1.6535×10^{-17}	3.1255×10^{-17}	1000	6.1760×10^{-17}	6.8852×10^{-17}	3.8521×10^{-17}
	A-GSA	946	0	0	0	951	0	0	0
	CA-GSA	674	0	0	0	682	0	0	0
f_2	GSA	1000	1.8394×10^{-8}	2.3617×10^{-8}	2.4311×10^{-8}	1000	1.1399×10^{-6}	1.8237×10^{-6}	3.2107×10^{-6}
	A-GSA	957	6.3551×10^{-240}	6.3551×10^{-240}	0	960	7.4507×10^{-40}	7.4507×10^{-240}	0
	CA-GSA	871	0	0	0	887	0	0	0
f_3	GSA	1000	213.8583	245.2465	124.1475	1000	856.0238	1.0389×10^{3}	8.0389×10^{2}
	A-GSA	957	0	0	0	965	0	0	0
	CA-GSA	655	0	0	0	669	0	0	0
f_4	GSA	1000	2.8288×10^{-9}	3.4666×10^{-9}	4.6413×10^{-9}	1000	3.1509	3.6869	1.3257
	A-GSA	1000	3.3665×10^{-241}	3.3665×10^{-241}	0	1000	3.2135×10^{-241}	3.2135×10^{-241}	0
	CA-GSA	892	0	0	0	903	0	0	0
f_5	GSA	1000	25.7921	26.2523	3.2792	1000	50.9788	51.2594	1.4218
	A-GSA	277	6.5892	6.5915	0.0335	312	27.7344	27.8057	0.2319
	CA-GSA	294	1.0389×10^{-4}	1.3980×10^{-4}	2.6081×10^{-4}	309	1.2355×10^{-4}	1.6806×10^{-4}	2.7493×10^{-4}
f_6	GSA	1000	1.4927×10^{-17}	1.8800×10^{-17}	0.9811×10^{-17}	1000	6.0914×10^{-17}	6.6512×10^{-17}	4.1126×10^{-17}
	A-GSA	161	3.0118	3.1384	0.3688	168	9.9137	10.1299	0.6813
	CA-GSA	159	2.2475×10^{-7}	2.4897×10^{-7}	1.2477×10^{-6}	169	1.8447×10^{-6}	2.3945×10^{-6}	3.2935×10^{-6}
f_7	GSA	1000	0.0259	0.0264	2.2791×10^{-3}	1000	0.0965	0.1206	0.0925
	A-GSA	723	1.6703×10^{-4}	2.3735×10^{-4}	1.8252×10^{-4}	739	1.9180×10^{-4}	2.6899×10^{-4}	1.2419×10^{-4}
	CA-GSA	459	3.4275×10^{-5}	4.2652×10^{-5}	3.5183×10^{-5}	462	5.6901×10^{-5}	6.2318×10^{-5}	2.3617×10^{-5}
f_8	GSA	1000	-2.8403×10^{3}	-2.8251×10^{3}	72.2457	1000	-3.7552×10^{3}	-3.7336×10^{3}	93.6284
	A-GSA	896	–10 652.68	–10 639.28	67.3552	890	–26 336.57	–26 314.83	84.8538
	CA-GSA	889	–12 563.82	–12 556.23	58.5992	878	–20 947.99	–20 944.63	60.0375
f_9	GSA	1000	14.3566	16.0801	6.0467	1000	35.8226	36.9129	3.6118
	A-GSA	393	0	0	0	398	0	0	0
	CA-GSA	362	0	0	0	361	0	0	0
f_{10}	GSA	1000	2.5073×10^{-9}	3.4686×10^{-9}	2.3746×10^{-9}	1000	3.7160×10^{-9}	4.9160×10^{-9}	4.2701×10^{-9}
	A-GSA	736	8.8817×10^{-16}	8.8818×10^{-16}	0	768	8.8817×10^{-16}	8.8818×10^{-16}	0
	CA-GSA	592	8.8817×10^{-16}	8.8817×10^{-16}	0	582	8.8817×10^{-16}	8.8817×10^{-16}	0
f_{11}	GSA	1000	1.9082	1.9239	0.1092	1000	18.2104	18.2615	0.4593
	A-GSA	359	0	0	0	374	0	0	0
	CA-GSA	298	0	0	0	305	0	0	0
f_{12}	GSA	1000	0.0351	0.0374	0.0102	1000	0.4457	0.446	0.0043
	A-GSA	1000	0.4954	0.5733	0.2041	1000	0.7913	0.7983	0.0548
	CA-GSA	1000	9.0147×10^{-9}	1.1408×10^{-8}	1.9984×10^{-8}	1000	1.3863×10^{-8}	1.4563×10^{-8}	3.9127×10^{-8}
f_{13}	GSA	1000	1.4318	1.6696	0.9618	1000	0.2538	0.2697	0.0612
	A-GSA	973	0.0029	0.0055	0.0557	982	0.0011	0.0038	0.0871
	CA-GSA	957	3.1576×10^{-5}	3.7965×10^{-5}	4.4721×10^{-5}	965	2.3941×10^{-5}	4.4210×10^{-5}	5.3627×10^{-4}

4 结论

针对 GSA 在复杂优化问题中寻优能力的不足的问题, 提出了一种融合混沌反学习和人工蜂群搜索算子的引力搜索算法 (CA-GSA). CA-GSA 的初始种群在保持随机性的前提下, 提高了种群的遍历性, 进而改善了收敛速度; 同时, 人工蜂群搜索算子的引入, 平衡了 GSA 算法探索和开采能力, 进一步改善了 GSA 的优化能力. 通过对 13 基准测试函数进行仿真测试, 验证了 CA-GSA 算法的有效性和优越性.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

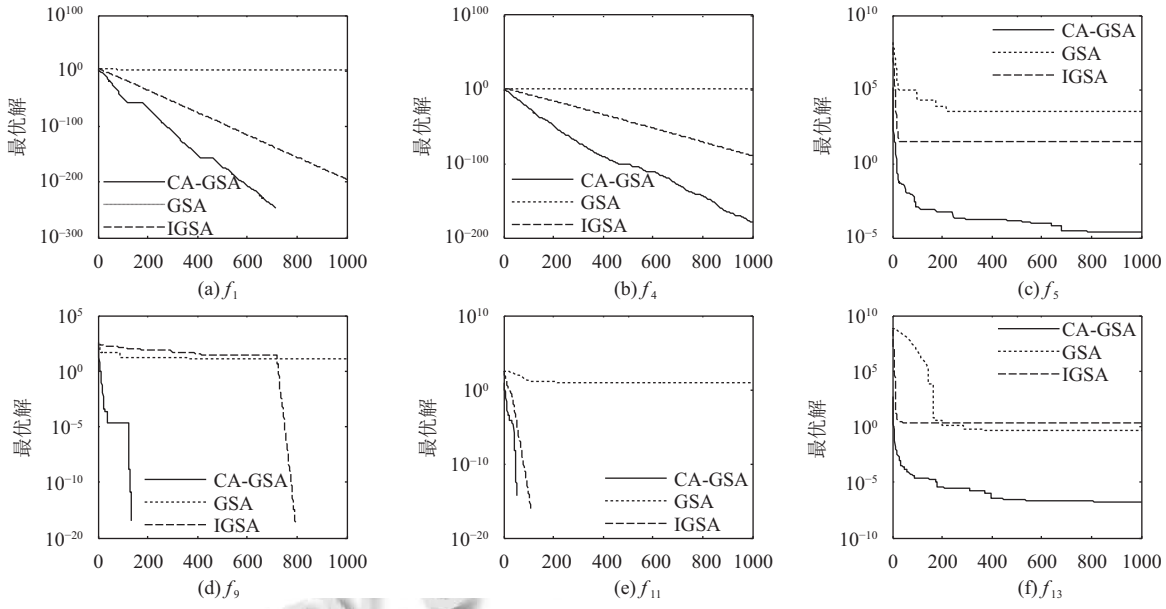

图 2 3 种算法对函数 f_1、f_4、f_5、f_9、f_{11}、f_{13} (n=30) 的优化性能比较

参考文献

1 Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: A gravitational search algorithm. Information Sciences, 2009, 179(13): 2232–2248. [doi: 10.1016/j.ins.2009.03.004]

2 Mahmoodabadi MJ, Nemati AR. A novel adaptive genetic algorithm for global optimization of mathematical test functions and real-world problems. Engineering Science and Technology, an International Journal, 2016, 19(4): 2002–2021. [doi: 10.1016/j.jestch.2016.10.012]

3 Garg H. A hybrid PSO-GA algorithm for constrained optimization problems. Applied Mathematics and Computation, 2016, 274: 292–305. [doi: 10.1016/j.amc.2015.11.001]

4 付强, 葛洪伟, 苏树智. 引入萤火虫行为和 Levy 飞行的粒子群优化算法. 计算机应用, 2016, 36(12): 3298–3302. [doi: 10.11772/j.issn.1001-9081.2016.12.3298]

5 Liu C, Niu PF, Li GQ, *et al*. Enhanced shuffled frog-leaping algorithm for solving numerical function optimization problems. Journal of Intelligent Manufacturing, 2015. [doi: 10.1007/s10845-015-1164-z]

6 Xiao JH, Niu YY, Chen P, *et al*. An improved gravitational search algorithm for green partner selection in virtual enterprises. Neurocomputing, 2016, 217: 103–109. [doi: 10.1016/j.neucom.2016.03.092]

7 Zhang AZ, Sun GY, Ren JC, *et al*. A dynamic neighborhood learning-based gravitational search algorithm. IEEE Transactions on Cybernetics, 2016. [doi: 10.1109/TCYB.2016.2641986]

8 Niu PF, Liu C, Li PF, *et al*. Optimized support vector regression model by improved gravitational search algorithm for flatness pattern recognition. Neural Computing and Applications, 2015, 26(5): 1167–1177. [doi: 10.1007/s00521-014-1798-3]

9 张维平, 任雪飞, 李国强, 等. 改进的万有引力搜索算法在函数优化中的应用. 计算机应用, 2013, 33(5): 1317–1320.

10 杨晶, 黎放, 狄鹏. 免疫万有引力搜索算法的研究与仿真. 兵工学报, 2012, 33(12): 1533–1538.

11 徐遥, 王士同. 引力搜索算法的改进. 计算机工程与应用, 2011, 47(35): 188–192. [doi: 10.3778/j.issn.1002-8331.2011.35.053]

12 Gao WF, Liu SY, Huang LL. Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. Communications in Nonlinear Science and Numerical Simulation, 2012, 17(11): 4316–4327. [doi: 10.1016/j.cnsns.2012.03.015]

13 樊友洪, 邓韧, 李生林, 等. 基于混沌遗传算子的人工鱼群算法. 计算机系统应用, 2017, 26(3): 214–218. [doi: 10.15888/j.csa.005664]

14 马卫, 孙正兴. 基于精英蜂群搜索策略的人工蜂群算法. 计算机应用, 2014, 34(8): 2299–2305. [doi: 10.11772/j.issn.1001-9081.2014.08.2299]

15 王东云, 徐艳平, 瞿博阳. 基于改进蜂群算法的机器人路径规划. 计算机系统应用, 2017, 26(2): 145–150. [doi: 10.15888/j.csa.005601]

16 Li GQ, Niu PF, Xiao XJ. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. Applied Soft Computing, 2012, 12(1): 320–332. [doi: 10.1016/j.asoc.2011.08.040]

© 中国科学院软件研究所 http://www.c-s-a.org.cn