文章编号: 1001-0920 (2014) 11-2041-07 DOI: 10.13195/j.kzyjc.2013.1003

具有自适应全局最优引导快速搜索策略的人工蜂群算法

赵 辉, 李牧东, 翁兴伟

(空军工程大学 航空航天工程学院，西安 710038)

摘 要: 针对人工蜂群算法存在开发与探索能力不平衡的缺点, 提出了具有自适应全局最优引导快速搜索策略的改进算法. 在该策略中, 首先采蜜蜂利用自适应搜索方程平衡了不同搜索方法的探索和开发能力; 其次跟随蜂利用全局最优引导邻域搜索方程对蜜源进行精细化搜索, 以提高其收敛精度和全局搜索能力. 14 个标准测试函数的仿真结果表明, 相比其他算法, 所提出的改进算法有效平衡了算法的开发与探索能力, 并提高了其最优解的精度及收敛速度.

关键词: 人工蜂群算法；自适应；邻域搜索；函数优化；最优引导

中图分类号: TP301 **文献标志码:** A

Improved artificial bee colony algorithm with self-adaptive global best-guided quick searching strategy

ZHAO Hui, *LI Mu-dong*, *WENG Xing-wei*

(College of Aeronautics and Astronautics Engineering, Air Force Engineering University, Xi'an 710038, China. Correspondent: LI Mu-dong, E-mail: modern_lee@163.com)

Abstract: For the problems of the unbalanced capability between exploration and exploitation of artificial bee colony(ABC) algorithm, an improved ABC algorithm with the self-adaptive global best-guided quick searching strategy(ABCSGQ) is proposed. On the one hand, the self-adaptive search equation is used for employed bees so as to balance the exploration and exploitation of two different solution searching methods. On the other hand, the global best-guided neighborhood search method is adopted for onlooker bees in order to improve the convergence precision and the global search ability. The simulation on 14 benchmark functions shows that the proposed algorithm fully utilizes and balances the exploration and exploitation, and greatly improves the accuracy of optima solutions and convergence speed compared with other current improved ABC algorithms for optimization.

Key words: artificial bee colony algorithm; adaptive method; neighborhood search; function optimization; global best-guided

0 引 言

人工蜂群 (ABC) 算法[1]是继粒子群算法[2]、遗传算法[3]和蚁群算法[4]后提出的又一种群集智能优化算法. 相比于遗传算法、粒子群算法和蚁群算法, 该算法由于模拟蜂群的采蜜过程, 对蜂群进行了功能划分, 在求解函数最优化问题方面表现出了更加优越的性能[5-6]; 同时该算法具有操作简单、易于实现、参数设置较少等特点. ABC 算法一经提出便受到众多学者的关注和研究, 并在函数优化[7]、聚类分析[8]、神经网络[9]等领域得到了广泛的应用. 但是, 标准 ABC 算法同样存在易于陷入局部最优、算法后期收敛速度较慢、搜索精度不高以及对于复杂函数优化问题无法搜索到全局最优解等问题, 从而大大影响了 ABC 算法的实际应用.

针对上述缺点, 学者们分别提出了相应的策略来改善 ABC 算法的性能. 文献 [10] 提出的反向学习法对蜂群进行初始化, 从而改善了种群的多样性; 文献 [11] 提出了全局引导搜索策略对蜂群搜索方程进行改进, 提高了算法的搜索精度和收敛速度; 文献 [12] 提出的邻域搜索法对跟随蜂的蜜源更新公式进行改进, 提高了算法的局部搜索能力; 文献 [13] 引入 Rosenbrock 旋转方向的方法对采蜜蜂的搜索方程进行改进, 以提高其收敛速度; 文献 [14] 通过概率 P 选择两种搜索方程来达到一定程度上平衡算法

收稿日期: 2013-07-21；**修回日期:** 2013-10-21.

基金项目: 航空科学基金项目(20105196016)；中国博士后科学基金项目(2012M521807).

作者简介: 赵辉(1973−), 男, 教授, 博士生导师, 从事武器系统运用与工程、智能优化算法等研究；李牧东(1987−), 男, 博士生, 从事无人飞行器武器系统总体技术、智能优化算法的研究.

开发与探索能力的目的. 上述改进算法虽然在某一方面对ABC算法的性能进行了优化, 但是在算法收敛速度与陷入局部最优以及搜索精度方面仍然很难取得平衡.

针对这一问题, 本文提出了一种具有自适应搜索策略和全局引导邻域搜索策略新型快速人工蜂群算法(ABCSGQ). 首先, 在文献[11]的基础上, 受到文献[15]提出的自适应搜索方法的启发, 针对采蜜蜂提出了自适应搜索方程, 在兼顾算法探索能力的同时, 提高了算法的开发能力; 然后, 在文献[12]的基础上针对跟随蜂的行为特点提出了全局邻域引导搜索策略, 从而提高了算法的搜索精度和收敛速度; 最后通过对14个测试函数的仿真及与其他改进ABC算法的比较, 验证了本文改进算法的性能. 仿真结果表明, 本文提出的算法能够有效提高复杂、高维函数的优化效率, 有效平衡算法的探索和开发能力.

1 标准ABC算法

在ABC算法中, 人工蜂群按照分工可分为3种, 即采蜜蜂、跟随蜂和侦查蜂, 其中采蜜蜂和跟随蜂各占蜂群总数的一半, 并且每一个蜜源仅有一个采蜜蜂工作. 算法初始化时首先按下式:

$$x_{m,i} = L_i + \mathrm{rand}(0,1)(U_i - L_i) \tag{1}$$

随机生成SN个初始解, 并计算每个蜜源位置的花粉量或适应度值, 同时记录全局最优值. 其中: $x_{m,i}$ 为蜜源位置, $m = 1, 2, \cdots,$ SN, $i \in$ Dim; L_i 和 U_i 是蜂群搜索空间的下界和上界. 适应度值计算公式为

$$\mathrm{fit}(x_m) = \begin{cases} \dfrac{1}{f(x_m)}, & f(x_m) \geqslant 0; \\ 1 + \mathrm{abs}(f(x_m)), & f(x_m) < 0. \end{cases} \tag{2}$$

其中 $f(x_m)$ 为蜜源 x_m 的目标函数值. 之后进行对最优解的搜索, 其具体搜索过程如下:

1) 采蜜蜂对蜜源进行搜索并记忆蜜源的花蜜量, 即问题解的质量(适应度);

2) 跟随蜂根据从采蜜蜂处获得的蜜源信息通过判断收益率来选择一个蜜源, 而后对记忆的位置进行更新;

3) 当蜜源因蜂蜜开采完而被放弃时, 产生侦察蜂并搜索新的蜜源来替代旧蜜源.

在算法中, 为了根据记忆位置 X_m 产生一个新的候选位置 V_m, 采用下式进行更新:

$$v_{m,i} = x_{m,i} + \phi_{m,i}(x_{m,i} - x_{k,i}). \tag{3}$$

其中: $k(k \in 1, 2, \cdots,$ SN 且 k 不等于 $m)$ 为随机选择的下标, $\phi_{m,i}$ 为 $[-1, 1]$ 之间的随机数. 这里称式(3)为ABC算法的蜜源搜索方程. 根据蜜源的蜂蜜量, 跟随蜂选择某个蜜源的概率为

$$P_m = \frac{\mathrm{fit}(x_m)}{\sum\limits_{m=1}^{\mathrm{SN}} \mathrm{fit}(x_m)}. \tag{4}$$

若经过"limit"次后, 蜜源位置没有被更新, 则放弃该位置, 而该位置的采蜜蜂成为侦察蜂, 并按式(1)产生新的蜜源.

2 具有自适应全局最优引导快速搜索策略的ABC算法

对于一个群体智能优化算法而言, 如何权衡算法的"探索与开发"能力决定了算法的优化性能[11]. 开发能力是指在某个特定区域内搜索并能够提炼出较好解的能力, 而探索能力是指算法探究搜索空间的不同区域以便确定一个较好解的能力. ABC算法的蜜源搜索方程因其选择的随机性而具有较强的探索能力. 但同时可以发现, ABC算法的开发能力较差, 因而存在收敛速度慢、搜索精度差的问题. 针对这一问题, 提出了自适应全局最优引导快速搜索策略, 该策略主要根据采蜜蜂和跟随蜂不同的搜索机制, 提出了采蜜蜂自适应搜索方程和跟随蜂全局最优引导邻域搜索方程. 下面对这一策略作具体说明.

2.1 采蜜蜂自适应搜索方程

在初步研究如何平衡ABC算法开发与探索能力时, 文献[11]根据粒子群算法的搜索方程, 提出了全局最优引导搜索方程, 即

$$v_{m,i} = x_{m,i} + \phi_{m,i}(x_{m,i} - x_{k,i}) + \psi_{m,i}(V_{\mathrm{gbest},i} - x_{m,i}). \tag{5}$$

其中: m, i, k 和 $\phi_{m,i}$ 的选取同式(3), $\psi_{m,i}$ 为[0,1.5]之间的随机数; $V_{\mathrm{gbest},i}$ 为当前循环次数下的全局最优解.

该搜索方程虽然在一定程度上提高了算法的开发能力, 但并没有充分考虑ABC算法在循环初期需要保持较强的开发能力而在循环后期需要保持较高的开发能力这一问题. 为此, 本文在文献[11]的基础上, 结合文献[15]在研究粒子群算法时所提出的自适应思想, 提出了自适应蜜源搜索方程

$$v_{m,i} = x_{m,i} + \mu(t)(x_{j,i} - x_{k,i}) + (1 - \mu(t))\psi_{m,i}(V_{\mathrm{gbest},i} - x_{m,i}), \tag{6}$$

$$\mu(t) = 1 - \mathrm{rand}^{(1 - t/\mathrm{maxcycle})^2}. \tag{7}$$

其中: $x_{j,i}$ 为第 i 维上不同于采蜜蜂 m 和 k 的任一采蜜蜂; $\mu(t)$ 为自适应系数, 它决定了蜜源搜索方程(3)与全局最优引导搜索方程(5)对循环次数的依赖程度, t 为当前循环次数; maxcycle 为最大循环次数. 由式(7)可知: 在循环初期, $\mu(t) \approx 1$, 此时式(6)以蜜源搜索方程为主, 能够保持较强的探索能力, 提高其

全局搜索能力, 避免陷入局部最优; 随着循环次数的增加, $\mu(t)$ 逐渐趋近于0, 此时式 (6) 以全局最优引导搜索方程为主, 满足了对特定区域进行精细化搜索的要求, 从而提高了算法的收敛速度与收敛精度. 通过上述改进, 采蜜蜂能够自适应地选择两种不同的搜索方程, 从而有效地平衡并充分利用了算法的探索与开发能力.

2.2 跟随蜂全局最优引导邻域搜索方程

ABC 算法是模拟蜜蜂采蜜行为而设计出的智能群集优化算法, 而在蜜蜂采蜜过程中, 采蜜蜂负责在特定范围内搜索蜜源并将所搜索到的蜜源信息共享给跟随蜂, 而跟随蜂则通过判断蜜源的蜂蜜量选择较好的蜜源进一步搜索, 以寻找到最优的蜜源. 由对 ABC 算法原理的分析可知, 采蜜蜂和跟随蜂使用相同的搜索方程, 这与算法模拟蜂群采蜜的过程相矛盾. 针对这一问题, Dervis Karaboga 提出了跟随蜂的搜索方程[12]

$$v_{N_m,i}^{\text{best}} = x_{N_m,i}^{\text{best}} + r_{m,i}(x_{N_m,i}^{\text{best}} - x_{k,i}). \tag{8}$$

其中: SN 为蜜源个数; $x_{N_m,i}^{\text{best}}$ 是 x_m 邻域内的最优解, 而 x_m的邻域是通过计算最优解与 x_m 之间的平均距离定义的, 即

$$md_m = \frac{\sum_{j=1}^{\text{SN}} d(m,j)}{\text{SN}-1}, \tag{9}$$

$d(m,j)$ 为蜜源 x_m 与 x_j 之间的欧式距离. 当 $d(m,j) < r \times md_m$ 时, 认为该蜜源位置为 x_m 的邻域, r 为邻域系数, 一般 $r = 1$. 若该邻域存在 S 个蜜源位置, 则通过下式:

$$\begin{aligned}&\text{fit}(x_{N_m}^{\text{best}}) = \\ &\max(\text{fit}(x_{N_m}^1), \text{fit}(x_{N_m}^2), \cdots, \text{fit}(x_{N_m}^S))\end{aligned} \tag{10}$$

选出最优的蜜源位置作为跟随蜂的邻域搜索位置, 而后再根据式(8)更新蜜源位置. 通过上述改进, 在一定程度上提高了 ABC 算法的收敛速度. 同时可发现, 跟随蜂的主要目的是为了对蜜源位置进行精细化搜索, 以期提高全局最优解的质量, 而文献 [12] 提出的跟随蜂搜索方程仍存在较大的随机性. 本文受到文献 [11] 的启发, 对式 (8)进行了改进, 提出了全局最优引导邻域搜索策略

$$\begin{aligned}v_{N_m,i}^{\text{best}} = &x_{N_m,i}^{\text{best}} + \delta_{m,i}(x_{N_m,i}^{\text{best}} - x_{k,i}) + \\ &\psi_{m,i}(X_{m,i}^{\text{gbest}} - x_{N_m,i}^{\text{best}}).\end{aligned} \tag{11}$$

其中: $\delta_{m,i}$ 为服从高斯分布 $N(1,0)$ 的随机数; $\psi_{m,i}$ 为 [0,1.5] 之间的随机数; $X_{m,i}^{\text{gbest}}$ 为当前所有采蜜蜂个体的最优蜜源位置, 即全局最优位置, 能够使跟随蜂在当前全局最优解的引导下进行较优蜜源的邻域搜索. 通过上述改进不仅有效地保持了跟随蜂局部搜索的特性, 同时通过全局最优引导策略改善了最优搜索的精细化程度, 以提高其收敛精度和收敛速度.

2.3 改进算法分析及步骤

在标准 ABC 算法基础上, 改进算法采用所提出的自适应全局最优引导快速搜索策略, 首先对采蜜蜂通过引入自适应系数对其搜索方程进行了改进, 即自适应搜索方程, 该方程能够综合考虑 ABC 算法在算法初期要求较好的探索能力而在算法后期要求较好的开发能力的问题, 随着循环次数的增加, 自适应地平衡算法的开发与探索能力; 其次在研究了跟随蜂采蜜机制的基础上, 为了进一步提高其收敛速度和搜索精度, 使跟随蜂采用所提出的最优引导邻域搜索方程. 这些操作增强了算法逃离局部最优值的能力, 提高了全局搜索性能, 在收敛速度和跳出局部最优方面取得了较好平衡. 具体步骤如下:

1) 设置算法的各个参数;

2) 初始化规模为 SN 的蜂群, 并计算每个采蜜蜂对应蜜源位置的适应度值并记录全局最优值;

3) While 算法终止条件不满足,

4) for 采蜜蜂,

5) 根据式 (6) 对蜜源位置进行更新并计算新蜜源的适应度值;

6) 应用贪婪机制选择较优的蜜源;

7) 若蜜源位置得到更新, 则 $\text{trail}(i) = 0$;

8) 否则 $\text{trail}(i) = \text{trail}(i) + 1$;

9) End for

10) 计算更新后蜜源的适应度值, 并按式 (4) 计算选择概率 P_m;

11) for 每个蜂群中的跟随蜂,

12) if $\text{rand}(0,1) < P_m$,

13) 利用式 (11) 产生新的蜜源;

14) 应用贪婪机制选择较优的蜜源;

15) 若蜜源位置得到更新, 则 $\text{trail}(i) = 0$;

16) 否则, $\text{trail}(i) = \text{trail}(i) + 1$;

16) End if

17) End for

18) if $\max(\text{trail}(i)) > \text{limit}$,

19) 利用式 (1) 产生新的蜜源, 替换被采蜜蜂丢弃的蜜源;

20) End if

21) End While($\text{globalmin} \leqslant \varepsilon$, $t > \text{maxcycle}$)

22) 输出最优解及最优值.

3 仿真实验

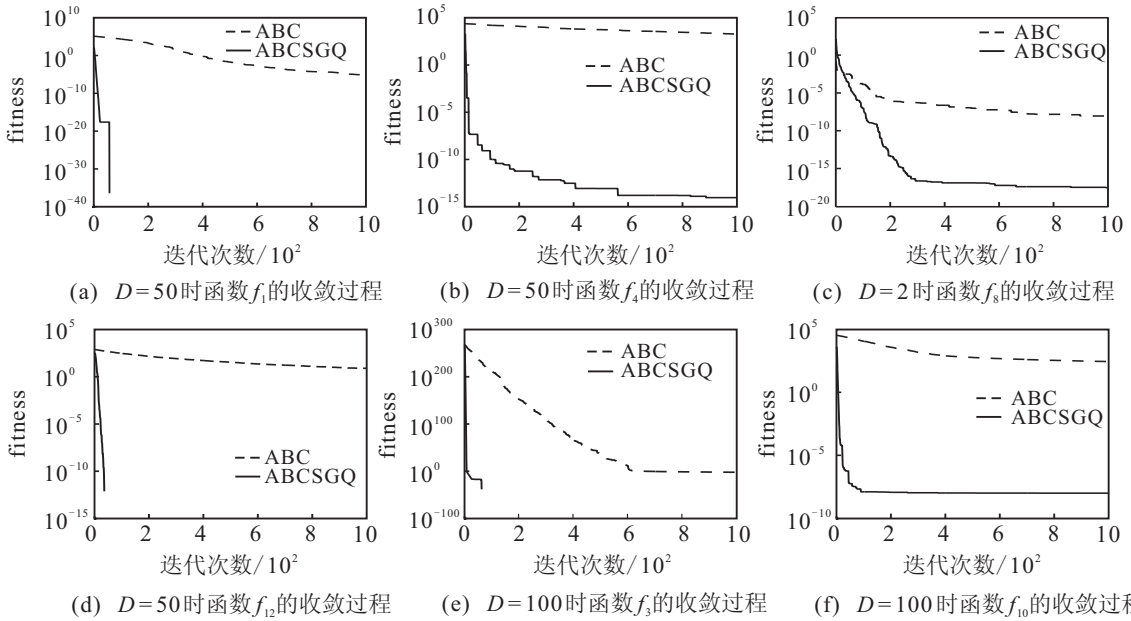
为了验证本文提出的 ABCSGQ 算法的性能, 选

取了14个基准函数作为测试集[14-15], 并与标准ABC算法[1]、GABC算法[11]、qABC算法[12]以及IABC算法[14]的测试结果进行比较. 实验用Matlab实现, 针对所有基准函数, 设种群规模SN = 40, 循环次数maxcycle = 1 000, 算法精度值ε =1e-40, 实验次数time = 50. 通过计算50次独立实验中的最优值、平均值、最小值、中间值、最大值以及方差和时间等参数, 评价算法的优化性能.

表1给出了14个基准函数的表达式、搜索范围和理论最优值, 其中f_1~f_3为单峰函数, f_4为添加了噪声的4次函数, f_5~f_8为低维多峰函数, f_9~f_{14}为高维多峰函数, f_1为Sphere函数, f_9为Griewank函数, f_{10}为Rosenbrock函数, f_{11}为Ackley函数, f_{12}为Rastrigin函数, f_{14}为Schaffer函数. 图1给出了典型的6种基

表 1 测试函数

基准测试函数	搜索范围	最优值
$f_1(x)=\sum_{i=1}^{D}x_i^2$	$[-100,100]$	0
$f_2(x)=\sum_{i=1}^{D}(10^6)^{(i-1)/(n-1)}x_i^2$	$[-100,100]$	0
$f_3(x)=\sum_{i=1}^{D}\lvert x_i\rvert^{(i+1)}$	$[-10,10]$	0
$f_4(x)=\sum_{i=1}^{D}ix_i^4+\mathrm{random}[0,1)$	$[-1.28,1.28]$	0
$f_5(x)=\left(4x^2-2.1x^4+\frac{1}{3}x^6+xy-4y^2+4y^4\right)+1.031\,63$	$[-5,5]$	0
$f_6(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	$[-10,10]$	0
$f_7(x)=(x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+10(x_1-x_4)^4$	$[-30,30]$	0
$f_8(x)=(x_1^2+x_2-11)^2+(x_2^2+x_1-7)^2$	$[-6,6]$	0
$f_9(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]$	0
$f_{10}(x)=\sum_{i=1}^{D}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-2.048,2.048]$	0
$f_{11}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)_i^2\right)+20+e$	$[-32,32]$	0
$f_{12}(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]$	0
$f_{13}(x)=\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)+10)$, if $\lvert x_i\rvert<0.5, y_i=x_i$; else $y_i=\frac{\mathrm{round}(2x_i)}{2}$	$[-5.12,5.12]$	0
$f_{14}(x)=0.5+\sin^2\left(\frac{\sqrt{\sum_{i=1}^{D}x_i^2}-0.5}{\left(1+0.001\left(\sum_{i=1}^{D}x_i^2\right)\right)^2}\right)$	$[-100,100]$	0



(a) D=50时函数f_1的收敛过程 (b) D=50时函数f_4的收敛过程 (c) D=2时函数f_8的收敛过程

(d) D=50时函数f_{12}的收敛过程 (e) D=100时函数f_3的收敛过程 (f) D=100时函数f_{10}的收敛过程

图 1 不同基准函数的测试收敛过程

表 2　ABCSGQ 算法和 ABC 算法对 $f_1 \sim f_{14}$ 函数的测试结果

F	D	Method	Best	Median	Worst	Mean	Std	t/s
f_1	50	ABC	5.701 16e-07	2 027 422e-06	2.560 4e-05	7.183 7e-06	8.448 5e-06	0.001 006
		ABCSGQ	0	0	0	0	0	0.000 285 7
	100	ABC	9.289 91e-03	2.336 59e-02	9.018 63e-02	3.115 8e-02	2.485 17e-02	0.001 031
		ABCSGQ	0	0	0	0	0	0.003 080 5
f_2	50	ABC	1.560 03e-02	5.423 68e-02	3.884 83e-01	9.968 03e-02	1.251 61e-01	0.001 002
		ABCSGQ	0	0	0	0	0	0.002 960 1
	100	ABC	4.775 87e+03	1.061 3e+04	2.326 7e+04	1.151 88e+04	6.085 89e+03	0.001 024
		ABCSGQ	0	0	0	0	0	0.003 307
f_3	50	ABC	2.955 67e-07	1.014 52e-06	2.815 86e-06	1.162 8e-06	8.055 2e-07	0.002 208
		ABCSGQ	0	0	0	0	0	0.004 174 7
	100	ABC	2.983 8e-03	6.871 33e-03	1.714 74e-02	7.559 02e-03	4.182 76e-03	0.003 202
		ABCSGQ	0	0	0	0	0	0.008 101 6
f_4	50	ABC	6.947 42e+02	1.348 31e+03	4.184 14e+03	1.907 28e+03	1.262 75e+03	0.001 517
		ABCSGQ	4.576 4e-23	2.936 47e-16	6.775 01e-14	9.174 99e-015	2.136 58e-14	0.017 086
	100	ABC	8.353 62e+04	1.130 41e+05	1.426 91e+05	1.140 12e+05	2.170 83e+04	0.002 028
		ABCSGQ	8.709 34e-12	2.754 78e-06	8.892 93e-01	9.362 83e-02	2.798 74e-01	0.018 253
f_5	2	ABC	1.546 51e-06	1.526 51e-06	1.526 51e-06	1.526 51e-06	1.072 58e-16	0.000 969
		ABCSGQ	1.267 47e-21	1.375 76e-21	1.375 76e-21	1.375 76e-21	1.013 64e-31	0.008 422 3
f_6	2	ABC	1.900 81e-13	1.8805 8e-10	2.077 91e-09	4.374 91e-10	6.368 77e-10	0.001 137
		ABCSGQ	1.606 16e-19	1.559 95e-17	4.416 6e-17	1.878 72e-17	1.540 82e-17	0.009 207
f_7	2	ABC	2.842 93e-04	1.016 03e-03	1.940 61e-02	4.041 09e-03	5.953 66e-03	0.001 178
		ABCSGQ	0	0	0	0	0	0.001 023 1
f_8	2	ABC	1.333 44e-13	2.984 56e-10	9.456 24e-08	9.978 76e-09	2.973 55e-08	0.008 503 2
		ABCSGQ	1.447 62e-19	1.534 35e-18	1.383 97e-17	3.327 28e-18	4.307 02e-18	0.012 932 1
f_9	50	ABC	8.354 8e-06	2.933 43e-04	1.054 98e-02	2.731 64e-03	4.378 2e-03	0.001 956
		ABCSGQ	0	0	0	0	0	0.000 913
	100	ABC	2.594 75e-02	1.581 65e-01	4.702 52e-01	2.042 21e-01	1.464 6e-01	0.002 105
		ABCSGQ	0	0	0	0	0	0.000 903
f_{10}	50	ABC	0.830 645	3.394 45e+01	9.163 57e+01	3.704e+01	3.269 95e+01	0.001 221
		ABCSGQ	4.248 01e-13	2.980 83e-12	1.141 19e-10	1.565 01e-11	3.502 45e-11	0.013 414
	100	ABC	1.904 52e+02	2.842 91e+02	3.793 08e+02	2.856 58e+02	5.885 39e+01	0.001 271
		ABCSGQ	6.822 02e-13	5.445 11e-11	1.012 35e-07	1.022 92e-08	3.197 67e-08	0.017 116
f_{11}	50	ABC	5.023 96e-03	1.377 41e-02	2.151 3e-01	3.460 66e-02	6.395 66e-02	0.001 252
		ABCSGQ	8.881 78e-16	8.881 78e-16	8.881 78e-16	8.881 78e-16	0	0.009 497
	100	ABC	2.220 34	3.448 24	3.723 67	3.304 27	0.442 473	0.001 343
		ABCSGQ	8.881 78e-16	8.881 78e-16	8.881 78e-16	8.881 78e-16	0	0.012 416
f_{12}	50	ABC	2.073 92	8.763 19	1.189 75e+01	7.850 28	3.253 33	0.001 126
		ABCSGQ	0	0	0	0	0	0.000 535
	100	ABC	5.701 37e+01	6.577 9e+01	9.574 15e+01	7.015 32e+01	1.400 94e+01	0.001 213
		ABCSGQ	0	0	0	0	0	0.000 147
f_{13}	50	ABC	0	2.048 38	5.110 7	2.352 73	1.615 75	0.002 691
		ABCSGQ	0	0	0	0	0	0.000 438
	100	ABC	1.799 79e+01	3.391 58e+01	5.601 9e+01	3.517 85e+01	1.072 37e+01	0.005 676
		ABCSGQ	0	0	0	0	0	0.000 627
f_{14}	50	ABC	7.925 08e-06	7.925 08e-06	7.925 08e-06	7.925 08e-06	0	0.001 113
		ABCSGQ	6.693 14e-18	3.151 78e-17	4.728 38e-17	1.321 97e-17	1.715 39e-17	0.003 215
	100	ABC	9.950 14e-07	9.950 14e-07	1.000 13e-06	9.956 67e-07	1.625 22e-09	0.005 806
		ABCSGQ	3.202 28e-17	2.184 88e-10	7.210 93e-09	9.273 13e-10	2.223 71e-09	0.035 345

准函数的测试收敛过程, 表 2 给出了测试结果.

从图 1 可以看出, 相比于 ABC 算法, ABCSGQ 算法在 6 种不同基准函数不同维度的收敛过程中具有明显的优越性, 其中在图 1(a)、1(d)、1(e) 中, 改进算法在较少循环次数下便能收敛至理论最优值, 而图 1(b)、1(c)、1(f) 中, ABCSGQ 在循环截止时虽然没能搜索到最优解, 但相比于 ABC 算法, 仍具有较快的收敛速度及搜索精度. 从表 2 可以看出, ABCSGQ 算

法对于函数 $f_1 \sim f_3$, f_7, f_9, f_{12} 和 f_{13} 在50维和100维两种条件下均能收敛至理论最优值, 而其他函数虽然无法达到理论最优值, 但相比于ABC算法已经非常接近最优值. 同时可以看出, 由于本文算法一方面采用了自适应搜索方程, 大大增加了算法在开发和探索方面的平衡能力; 另一方面采用了全局最优引导邻域搜索方程, 提高了收敛速度. 另外可以看出, 在对函数 f_1, f_9, f_{11}, f_{12} 和 f_{13} 的优化中, 改进算法由于较快收敛至理论全局最优解, 平均运行时间优于ABC算法. 因此, 该算法在略微增加运行时间的基础上, 其所有函数的最优值、中间值、最差值和平均值以及方差均明显优于ABC算法.

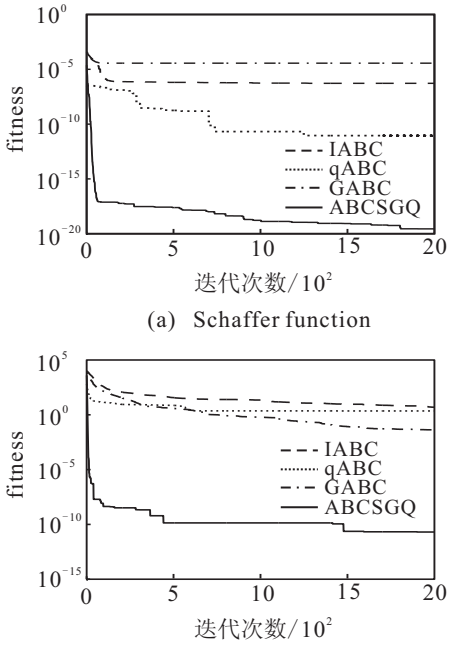
为了进一步说明ABCSGQ算法的优越性, 对IABC算法、GABC算法和qABC进行了比较, 其中参数设置参考文献[14]. 表3给出了测试结果, 图2和图3为4种算法对Sphere函数、Schaffer函数、Rastrigin函数和Rosenbrock函数的收敛过程对比.

从表3可以看出, ABCSGQ算法无论是在收敛速度、搜索精度还是稳定性方面均优于其他3种算法, 其中函数Sphere、Griewank和Rastrigin在评价次数较少时便能搜索到全局最优解, 而其他3种函数虽然没能收敛到最优解, 但搜索精度优于其他算法.

表 3 ABCSGQ与其他3种ABC改进算法的测试对比结果

F	D	Method	max.FE	Mean	Std
Schaffer	30	IABC	1.0e+06	9.95e-07	7.65e-11
		qABC	1.0e+06	8.97e-14	9.03e-14
		GABC	1.0e+06	3.65e-05	0
		ABCSGQ	1.0e+06	2.76e-20	5.22e-20
Rosenbrock	30	IABC	1.0e+06	4.89e-00	8.26e-00
		qABC	1.0e+06	2.24e-00	1.78e-01
		GABC	1.0e+06	4.20e-02	2.83e-02
		ABCSGQ	1.0e+06	8.39e-14	1.69e-13
Sphere	60	IABC	1.0e+06	6.37e-13	6.72e-13
		qABC	1.0e+06	6.95e-16	3.59e-16
		GABC	1.0e+06	1.80e-15	1.70e-16
		ABCSGQ	3.2e+03	0	0
Griewank	60	IABC	1.0e+06	7.49e-09	1.86e-08
		qABC	1.0e+06	2.78e-01	3.23e-06
		GABC	1.0e+06	1.54e-05	4.86e-04
		ABCSGQ	2.6e+03	0	0
Rastrigin	60	IABC	1.0e+06	9.81e-01	1.00e-00
		qABC	1.0e+06	1.59e-13	6.23e-14
		GABC	1.0e+06	9.49e-01	7.58e-01
		ABCSGQ	2.4e+01	0	0
Ackley	60	IABC	1.0e+06	1.01e-13	9.18e-15
		qABC	1.0e+06	3.18e-14	1.80e-14
		GABC	1.0e+06	1.79e-09	4.56e-10
		ABCSGQ	1.0e+06	8.88e-16	1.86e-18

从图2可以看出, 在低维Schaffer和Rosenbrock函数的测试中, ABCSGQ算法在指定循环次数内虽然没有搜索到理论全局最优解, 但其收敛速度及搜索精度方面明显优于其他3种改进ABC算法.

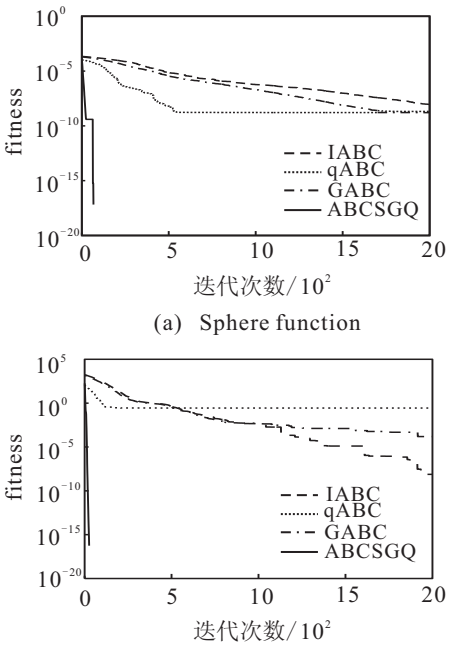


(a) Schaffer function

(b) Rosenbrock function

图 2 $D = 30$ 时4种算法对Schaffer和Rosenbrock函数的收敛过程对比

从图3可以看出, 在对高维单峰Sphere和多峰Griewank函数的优化中, 改进算法在较少循环次数条件下便能搜索到理论最优解, 表现出了优越的搜索性能.



(a) Sphere function

(b) Griewank function

图 3 $D = 60$ 时4种算法对Sphere和Griewank函数的收敛过程对比

4 结 论

为了解决标准ABC算法的收敛速度慢且易陷入早熟收敛等问题, 更好地平衡算法在搜索过程中的探

索与开发能力, 本文在全局最优引导搜索方程的基础上, 结合自适应思想, 提出了具有自适应全局最优引导快速搜索策略的ABCSGQ改进算法. 该算法通过对采蜜蜂采用自适应搜索方程和跟随蜂采用全局最优邻域引导方程的方法, 在提高算法开发能力的同时, 有效地保持了算法良好的探索能力. 通过对14个基准函数寻优及与其他现有改进ABC算法比较的实验结果可以看出, 本文算法在略微增加计算开销的前提下, 在收敛速度、搜索精度、鲁棒性以及快速跳出局部最优等方面优化性能具有较大改善.

可以发现, 对于 f_6, f_{10}, f_{11} 和 f_{14} 等这些很难寻优的复杂函数, 目前的改进ABC算法均无法收敛到理论最优解, 如何在更加复杂的函数上表现出良好的优化性能, 将是下一步的研究内容. 同时, 如何将本文改进算法应用于无人机组网、非线性系统优化以及最优控制等方面, 也是值得进一步研究的工作.

参考文献(References)

[1] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[2] Kennedy J, Eberhart R. Particle swarm optimization[C]. IEEE Int Conf on Neural Networks. Perth, 1995: 1942-1949.

[3] Tang K S, Man K F, Kwong S, et al. Genetic algorithms and their applications[J]. IEEE Signal Processing Magazine, 1996, 13(6): 22-37.

[4] Dorigo M, Stutzle T. Ant colony optimization[M]. Cambrige: MA MIT Press, 2004.

[5] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony(ABC) algorithm[J]. J of Global Optimization, 2007, 39(3): 459-471.

[6] Karaboga D, Basturk B. A comparative study of artificial bee colony algorithm[J]. Applied Mathematics and Computation, 2009, 214(1): 108-32.

[7] Karaboga D, Akay B. A modified artificial bee colony(ABC) algorithm for constrained optimization problems[J]. Applied Soft Computing, 2011, 11(3): 3021-3031.

[8] Karaboga D. A novel clustering approach: Artificial bee colony(ABC) algorithm[J]. Applied Soft Computing, 2011, 11(1): 652-657.

[9] Hsieh T J, Hsiao H F. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm[J]. Applied Soft Computing, 2011, 11(2): 2510-2525.

[10] Gao W F, Liu S Y. A modified artificial bee colony algorithm[J]. Computers and Operations Research, 2012, 39(3): 687-697.

[11] Zhu G P, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics and Computation, 2010, 217(7): 3166-3173.

[12] Karaboga D, Gorkemli B. A quick artificial bee colony(qABC) algorithm for optimization problems[R]. Erciyes University, Engineering Faculty, Computer Engineering Department, 2012.

[13] Kang F, Li J L, Ma Z Y. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions[J]. Information Sciences, 2011, 181(16): 3508-3531.

[14] 高卫峰, 刘三阳, 黄玲玲. 受启发的人工蜂群算法在全局优化问题中的应用[J]. 电子学报, 2012, 40(12): 2396-2403.
(Gao W F, Liu S Y, Huang L L. Inspired artificial bee colony algorithm for global optimization problems[J]. Acta Electronica Sinica, 2012, 40(12): 2396-2403.)

[15] 吴建辉, 章兢, 李仁发, 等. 多子种群微粒群免疫算法及其在函数优化中的应用[J]. 计算机研究与发展, 2012, 49(9): 1883-1898.
(Wu J H, Zhang J, Li R F, et al. A multi-subpopulation PSO immune algorithm and its application on function optimization[J]. J of Computer Research and Development, 2012, 49(9): 1883-1898.

(责任编辑: 孙艺红)