求解函数优化问题的改进的人工蜂群算法

葛 宇[1] 梁 静[2] 王学平[3] 谢小川[1]
(四川师范大学基础教学学院 成都 610068)[1] (成都工业学院网络中心 成都 610031)[2]
(四川师范大学数学与软件科学学院 成都 610068)[3]

摘 要 为提高人工蜂群算法求解复杂函数优化问题的性能,分析了算法中侦察蜂逃逸行为的不足,并对其进行改进:定义了逃逸指标,使其能准确地反映个体状态对算法早熟的影响;重新设计选择机制,让侦察蜂不需要参数控制,能自适应地选择可能导致算法早熟收敛的个体执行逃逸操作;改进了逃逸算子,降低了逃逸操作的盲目性。通过9个典型测试问题的实验结果表明:在指定误差精度下,本改进算法均能有效收敛;同时与基本人工蜂群算法和已有的典型改进相比,本改进算法在收敛精度和速度上均有明显提高。说明提出的改进策略能有效提高算法求解复杂函数优化问题的能力。

关键词 人工蜂群算法,早熟收敛,逃逸指标,选择机制,逃逸算子
中图法分类号 TP18 **文献标识码** A

Improved Artificial Bee Colony Algorithms for Function Optimization

GE Yu[1] LIANG Jing[2] WANG Xue-ping[3] XIE Xiao-chuan[1]
(College of Fundamental, Sichuan Normal University, Chengdu 610068, China)[1]
(Network Center, Chengdu Technological University, Chengdu 610031, China)[2]
(College of Mathematics and Soft Science, Sichuan Normal University, Chengdu 610068, China)[3]

Abstract In order to enhance the performance of artificial bee colony algorithm in solving complex function optimization problems, this paper analysed the shortcoming of escape behavior of scout bees, and improved it. The improved algorithm defines escape index, making it precisely reflecting the effect of individual status on the premature convergence of algorithm, redesigns the selection scheme, making scout bees choosing individual escape operation that might result in algorithm premature convergence adaptively, improves the escape operator, reducing the blindness of escape operation. Nine typical experiments prove that the improved algorithm could converge efficiently under assignment convergence accuracy, and the improved algorithm could converge with more convergence accuracy and speed compared with basic artificial colony algorithm and existing typical improved versions, thus proves the improved strategy proposed in this paper could boost capability of solving complex function optimization problems.

Keywords Artificial bee colony algorithm, Premature convergence, Escape index, Selection scheme, Escape operator

1 引言

人工蜂群算法[1]是Karaboga D提出的一种新兴智能优化算法,具有控制参数少、易于实现、计算简单等优点[2],近年来已引起了国内外学者的广泛关注,并在许多领域得到了应用[3-7]。然而和其他智能优化算法一样,人工蜂群算法在求解复杂函数优化问题中也存在早熟收敛的现象,从而导致算法性能降低。为此,不少学者提出了改进方案。如:文献[8]采用双种群机制并结合差分进化策略,帮助算法有效跳出早熟收敛状态,并快速搜寻到最优解。文献[9]利用混沌搜索的遍历性防止算法早熟收敛。文献[10]采用"分段搜索"方式对个体进行贪婪更新,避免算法早熟收敛,提高收敛速度。和以上改进方法不同,本文通过改进侦察蜂逃逸行为来增强算法跳出早熟收敛的能力,使算法的收敛精度和速度得到有效提高。具体地,本文设计了个体逃逸指标、逃逸算子和选择机制,让侦察蜂能自适应地选择可能导致算法早熟收敛的个体执行变尺度逃逸操作。实验结果表明,本文改进方案能帮助算法有效逃离早熟收敛,提高了人工蜂群算法的收敛速度和寻优精度。

2 人工蜂群算法描述

人工蜂群算法的寻优过程[1,11]模拟蜜蜂寻找优质花蜜源的行为,把蜜蜂分为引领蜂、跟随蜂和侦察蜂3种角色,通过3种蜜蜂间的协作使算法朝最优方向进化。算法的一次迭代

到稿日期:2012-11-27 返修日期:2013-03-11 本文受四川省教育厅项目(12ZB112)资助。

葛 宇(1981-),男,硕士,讲师,CCF会员,主要研究方向为计算智能,E-mail:geyufly@yahoo.com.cn;梁 静(1979-),女,硕士,讲师,CCF会员,主要研究方向为图形图像;王学平(1965-),男,博士,教授,博士生导师,主要研究方向为不确定性的数学理论及算法;谢小川(1975-),男,硕士,实验师,主要研究方向为计算机智能。

过程归纳为引领蜂、跟随蜂和侦察蜂3个阶段，具体描述为：

对于最小值优化问题

$$\text{Min}(f(X)), X\in S \tag{1}$$

式中，$f(X)$为适应度函数，$X=(x_1,x_2,\cdots,x_D)^T$为式(1)的任意可行解，$S=\{X=(x_1,x_2,\cdots,x_D)^T \mid x_j\in[a_j,b_j], a_j<b_j, j=1,2,\cdots,D\}$为式(1)的解空间，$W=\{X_1,X_2,\cdots,X_Q\}$表示包含$Q$个可行解(个体)的种群。

(1)引领蜂

引领蜂对种群W中的每个个体依次执行更新操作。具体地，对个体X_i的一维分量按式(2)进行变异，产生新个体。

$$x_{ij}'=x_{ij}+R\times(x_{ij}-x_{kj}) \tag{2}$$

式中，x_{ij}表示个体X_i的第j维分量；x_{kj}为个体X_k的第j维分量元；j,k均随机选择，且$k\neq i$；R为[-1,1]间的随机数。x_{ij}'表示新产生的第j维分量，其对应的新个体记为X_i'。算法根据贪婪选择算子式(3)更新X_i，并按式(4)记录个体未更新次数$trial_i$。

$$X_i=\begin{cases}X_i', & f(X_i')<f(X_i)\\ X_i, & f(X_i')\geqslant f(X_i)\end{cases} \tag{3}$$

$$trial_i=\begin{cases}0, & f(X_i')<f(X_i)\\ trial+1, & f(X_i')\geqslant f(X_i)\end{cases} \tag{4}$$

(2)跟随蜂

跟随蜂在优秀个体附近作局部搜索。即：算法进行Q次选择，每次均采用轮盘赌方式选出种群W中优秀个体执行式(2)—式(4)。

(3)侦察蜂

侦察蜂对可能导致算法早熟收敛的个体执行逃逸操作。侦察蜂根据条件$\text{MAX}(trial_i)>limit$选择个体$X_i$，利用逃逸算子式(5)更新早熟个体$X_i$中的每一维分量。

$$x_{ij}=b_j+R\times(b_j-a_j) \tag{5}$$

式中，R表示区间[0,1]内的随机数，b_j和a_j是第j维解空间的上下边界。

3 改进人工蜂群算法

人工蜂群算法中侦察蜂负责发现可能导致算法早熟收敛的个体，并对其进行更新，从而降低算法出现早熟的概率。但是，文献[10]指出人工蜂群算法仍然存在早熟收敛的现象，这说明侦察蜂的逃逸性能还有待提高。下文就侦察蜂的逃逸行为进行分析，并提出改进方案。

3.1 侦察蜂逃逸行为的讨论

侦察蜂选择$\text{MAX}(trial_i)$与参数$limit$比较，若$\text{MAX}(trial_i)>limit$成立，则认为对应个体$X_i$会导致算法早熟收敛，用逃逸算子(见式(5))对$X_i$进行更新。上述侦察蜂行为不能帮助算法有效逃离早熟收敛，而且会导致算法通用性低，其原因有如下3点：

1. 个体选择条件$\text{MAX}(trial_i)>limit$依赖参数$limit$，对于不同的优化问题，很难找到统一的标准。若$limit$设置不当，就会影响算法跳出早熟收敛的能力。因此在实际应用中，需要针对具体的应用环境来调试$limit$值，这必然会导致算法通用性低，不利于算法的推广。

2. 早熟收敛是指在算法未进化到理论最优解并且在种群多样性急剧下降的前提下，个体进化停止(即算法多次迭代都不能产生更好的个体)[12]。因此，在选择可能导致算法早熟的个体时，仅考虑个体未更新次数($trial$)，而忽略个体位置对种群多样性的影响，显然存在片面性。

3. 逃逸算子(见式(5))随机产生新个体，无法有效地控制逃逸尺度，使逃逸操作存在盲目性，从而导致算法的收敛精度和速度降低。例如：在算法进化后期，应采用较小尺度的逃逸算子才不会干扰算法的深度寻优，而式(5)无法调整逃逸尺度，所以不可避免地会影响到算法后期的深度寻优性能。

基于以上分析不难看出：以上3点不足会降低侦察蜂的逃逸性能，使算法不能有效逃离早熟收敛，从而导致算法寻优精度低，收敛缓慢甚至停滞。

3.2 侦察蜂逃逸行为的改进方案

为提高侦察蜂的逃逸能力，本文对侦察蜂的逃逸行为进行改进，具体思路是：定义了逃逸指标，使其能全面反映个体对算法早熟的影响程度；改进逃逸算子，降低逃逸操作的盲目性；设计了自适应选择机制，让侦察蜂不需要参数控制就能选出个体进行逃逸(即执行逃逸算子)。以下分别进行描述。

3.2.1 个体逃逸指标

为了全面衡量个体对算法早熟的影响，本文结合“个体未更新次数”和“个体位置对种群多样性的影响程度”两方面因素定义个体逃逸指标为Esp、Esp_i值越大，表示个体X_i越应该被选中进行逃逸。计算公式见式(6)。

$$Esp_i=trial_i-dis_i \tag{6}$$

式中：

(1)$trial_i$表示个体X_i的未被更新次数，见式(4)。

(2)dis_i表示个体X_i位置对种群多样性的影响程度，见式(7)。

$$dis_i=|f_i-\bar{f}| \tag{7}$$

式中，f_i为个体X_i的适应度(由式(1)的适应度函数$f(X)$计算出)，$\bar{f}$为种群中个体适应度的平均值。可见，式(7)是用个体X_i的适应度与种群平均适应度的偏离程度来评估X_i位置对种群多样性的影响，dis_i越小，说明X_i的位置导致种群多样性降低的可能性越大。

3.2.2 变尺度逃逸算子

为了有效地控制逃逸尺度，降低逃逸操作的盲目性，新逃逸算子的设计思路为：算法进化初期应进行大尺度逃逸，以保持种群的多样性，防止算法早熟收敛；随着算法进化的推进，逃逸尺度应逐渐变小，便于进化后期算法的深度寻优。具体地，本文结合当前个体位置以及当前算法的进化程度来产生新个体。设要进行逃逸的个体为X_i，G为算法最大进化代数，g为当前算法的进化代数，R是[-1,1]间随机数，x_{ij}为X_i的第j维分量，改进后的逃逸算子见式(8)(对X_i的每一维分量都更新)。

$$x_{ij}^{new}=x_{ij}+R\times\left(1-\frac{g}{G}\right)\times x_{ij} \tag{8}$$

式中，调节项$\left(1-\frac{g}{G}\right)$利用进化代数信息控制逃逸尺度。另外，为了保证$x_{ij}^{new}$为式(1)的可行解，本文采用反射策略对其

进行处理，即当 x_{ij}^{new} 超越可行解边界时，x_{ij}^{new} 被反弹回来，具体描述见式(9)，原理如图1所示。

$$x_{ij}^{new}=\begin{cases}b_j-(x_{ij}^{new}-b_j)\bmod(b_j-a_j), & x_{ij}^{new}>b_j\\ a_j+(a_j-x_{ij}^{new})\bmod(b_j-a_j), & x_{ij}^{new}<a_j\\ x_{ij}^{new}, & \text{其它}\end{cases} \tag{9}$$

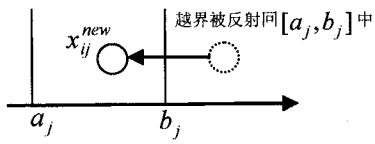

图1 反射策略原理

3.2.3 自适应选择机制

为避免侦察蜂受参数 *limit* 控制，并保证 *Esp* 值越大的个体被选中的概率越大，本文基于轮盘赌法则的特性——值越大被选概率越大，设计了一种自适应选择机制：

$$X_i=Select(M),\exists P_i\geqslant R \tag{10}$$

式中，M 代表种群且 $X_i\in M$；X_i 表示从种群 M 中选出的个体；R 为[0,1]内的随机数；P_i 为个体 X_i 的属性，表示 X_i 的被选择的概率，定义见式(11)：

$$P_i=\frac{Esp_i-\mathrm{Min}(Esp_i)}{\mathrm{Max}(Esp_i)-\mathrm{Min}(Esp_i)} \tag{11}$$

3.2.4 改进后的侦察蜂逃逸行为

改进后的侦察蜂逃逸行为用伪码描述如下：

```
i=1;
while(i<=Q){
  产生区间[0,1]内的一个随机数 R;
  if(R≤Pi){
    由式(10)选个体 Xi 执行式(8)、式(9);
    trial_i=0;
    break;}
  if(i==Q)
    i=0;
  else
    i=i+1;}
```

3.3 改进算法流程

改进后的人工蜂群算法流程如下：

步骤1 算法初始化，随机产生 Q 个个体，每个个体 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})^{\mathrm{T}}$ 是一个 D 维向量，指定算法结束条件(最大进化代数或求解精度)。

步骤2 种群中每个个体 X_i 随机选择一维分量 x_{ij}，利用式(2)进行变异，用式(3)对其更新，并用式(4)记录其未更新次数 $trial_i$。

步骤3 执行 Q 次选择，每次选出一个适应度较好的个体 X_i 执行式(2)、式(3)，并按式(4)计算其 $trial_i$ 值。

步骤4 按式(6)分别计算 Q 个个体的逃逸指标 Esp，并按式(11)计算每个个体的被选择概率 P。

步骤5 执行3.2.4节描述的改进侦察蜂逃逸行为。

步骤6 判断是否达到算法结束条件，若达到，输出最优个体，否则转到步骤2。

3.4 改进算法的计算量分析

与原有的人工蜂群算法相比，本文提出的改进方案仅修改了侦察蜂逃逸行为。以下对改进前后侦察蜂的计算量作分析，以评估改进方案对计算复杂度的影响。

算法的计算量通常以迭代所消耗的乘法和加法次数来衡量，在具体分析中，将除法、求余运算归为乘法；减法运算归为加法。设一次加法的计算量为 T_1；一次乘法的计算量为 T_2；种群中有 Q 个个体，个体维数为 D，算法最大进化代数为 G。改进前后侦察蜂的计算量分析如下：

1. 改进前侦察蜂在满足条件 $\mathrm{MAX}(trial_i)>limit$ 时执行式(5)操作(D 次加法、D 次减法和 D 次乘法)，因此改进前侦察蜂的最大计算量为：

$$O=G\times D\times(2T_1+T_2) \tag{12}$$

2. 改进后侦察蜂的最大计算量为：

$$O'=G\times[(3Q+5D+1)\times T_1+(Q+3D+2)\times T_2] \tag{13}$$

具体分析如下：

(1)式(6)要执行一次减法，在算法一次迭代中式(6)产生的计算量为：$Q\times T_1$。

(2)式(7)中的绝对值运算可以看作：当 $f_i>\overline{f}$ 时 $dis_i=f_i-\overline{f}$；当 $f_i<\overline{f}$ 时 $dis_i=\overline{f}-f_i$。另外，平均适应度 $\overline{f}$ 的计算需要 $Q-1$ 次加法和1次除法，而个体适应度 f_i 在之前算法步骤中已经求得。因此，式(7)在算法一次迭代时的计算量为：$Q\times T_1+T_2$。

(3)式(8)会执行 D 次加法、D 次减法、1次除法和 $2D$ 次乘法，故算法一次迭代中式(8)产生的最大计算量为：$2D\times T_1+2D\times T_2+T_2$。

(4)式(9)最多可能有 $3D$ 次减法和 D 次求余，故算法一次迭代中式(9)产生的最大计算量为：$3D\times T_1+D\times T_2$。

(6)式(11)要执行 $Q+1$ 次减法、Q 次除法，在算法一次迭代中式(11)产生的计算量是：$(Q+1)\times T_1+Q\times T_2$。

从式(12)、式(13)的对比可以看出，本文改进方案造成了计算量的增加，但不是指数级增长，在实际应用中改进算法增加的这部分计算量几乎不会影响到算法的运行时间，这一点在下文的实验数据(见表2)中得到了验证。

4 数值实验

4.1 实验设计

为了评估本文改进算法(以下记为 SABC)的收敛性、逃离早熟收敛的能力、寻优性能(即收敛精度、速度)和适应性，本文在 CPU2.6G、内存 2G 的计算机上将 SABC 与基本人工蜂群算法(以下记为 ABC)进行对比实验。实验从如下4个方面进行：(1)指定误差精度，评价算法的收敛性；(2)指定算法的进化代数，对比 SABC 与 ABC 逃离早熟收敛的能力以及收敛速度、精度；(3)在参数(种群中个体数量)变化的情况下讨论本文改进算法 SABC 的适应性；(4)SABC 与文献报道的改进算法结果进行比较，进一步评估本文的改进效果。其中，ABC 的 *limit* 参数值参考文献[13]，设置为：$Q*D$。

实验选择了如下9个典型的函数优化问题，其中 x^* 表示最优解。另外，为了增加求解复杂度，测试函数的维数 D 选择 50 或 100。

(1)Sphere Model

$f_1(x)=\sum_{i=1}^{D}x_i^2, |x_i|\leqslant 100, f(x^*)=0$

(2)Quartic Function

$f_2(x)=\sum_{i=1}^{D}ix_i^4+random[0,1), |x_i|\leqslant 1.28, f(x^*)=0$

(3)Step Function

$f_3(x)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2, |x_i|\leqslant 100, f(x^*)=0$

(4)Schwefel's Problem 2.21

$f_4(x)=\max\{|x_i|\}, |x_i|\leqslant 100, f(x^*)=0$

(5)Schwefel's Problem 2.22

$f_5(x)=\sum_{i=1}^{D}|x_i|+\prod_{i=1}^{D}|x_i|, |x_i|\leqslant 10, f(x^*)=0$

(6)SumSquares Function

$f_6(x)=\sum_{i=1}^{n}ix_i^2, |x_i|\leqslant 10, f(x^*)=0$

(7)Generalized Griewank Function

$f_7(x)=1+\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}(\cos\frac{x_i}{\sqrt{i}}) |x_i|\leqslant 600, f(x^*)=0$

(8)Generalized Rastrigin's Function

$f_8(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10], |x_i|\leqslant 5.12, f(x^*)=0$

(9)Ackley's Function

$f_9=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))+20+e, |x_i|\leqslant 32, f(x^*)=0$

4.2 指定误差精度时算法的收敛性比较

实验指定算法获得的目标函数值与理论最优值的相对误差小于 1.0E−4 时视为算法收敛，各算法针对 9 个测试函数(D=50)独立运行 10 次，种群内个体数 Q=20，所得结果取平均值，如表 1 所列(其中收敛率为算法收敛次数与总运行次数之比，若算法进化代数达到 1000 都未收敛到指定精度则收敛率记为 0)。

表 1 固定误差精度下的实验结果

函数	算法	平均进化代数	收敛率	平均时耗(s)
Sphere	ABC	690	100%	7.73E−6
	SABC	37	100%	5.91E−7
Quartic	ABC	—	0	1.09e−5
	SABC	972	100%	1.29e−5
Step	ABC	373	100%	4.49E−6
	SABC	30	100%	5.06E−7
Schwefel's 2.21	ABC	—	0	1.09E−5
	SABC	37	100%	5.58E−7
Schwefel's 2.22	ABC	—	0	1.12E−5
	SABC	56	100%	8.15E−7
SumSquares	ABC	558	100%	6.21E−6
	SABC	50	100%	7.27E−7
Griewank	ABC	918	40%	1.81E−5
	SABC	59	100%	1.37E−6
Rastrigin	ABC	—	0	1.28E−5
	SABC	75	100%	1.19E−6
Ackley	ABC	—	0	1.46E−5
	SABC	49	100%	9.21E−7

从表 1 中可以看出，对于所列测试函数，SABC 都实现了 100%收敛，并且进化代数都小于 ABC，尤其是除 Quartic 函数外 SABC 的平均进化代数都在 100 以内。说明本文提出的改进方案并不会降低算法的收敛性能，反而能减少收敛代数，从而使算法具有很快的收敛速度。

4.3 指定进化代数时比较算法逃离早熟收敛的能力以及收敛精度和速度

实验用 ABC、SABC 分别求解 9 个函数优化问题，指定算法进化代数为 1000，种群内个体数 Q=20。分别进行 10 次实验取平均结果，如表 2 所列，同时各函数在 50 维时的平均进化曲线如图 2—图 10 所示。

表 2 指定进化代数下的寻优结果

函数	算法	维数	平均值	标准差	平均时耗(s)
Sphere	ABC	50	2.81E−6	2.66E−6	1.09E−5
		100	3.23E−3	2.58E−3	1.11E−5
	SABC	50	1.32E−245	6.99E−326	1.33E−5
		100	7.71E−242	1.23E−335	1.47E−5
Quartic	ABC	50	3.01E−1	4.95E−2	1.14E−5
		100	9.21E−1	1.75E−1	1.17E−5
	SABC	50	3.43E−5	4.77E−4	1.35E−5
		100	4.09E−4	2.32E−4	1.49E−5
Step	ABC	50	0	0	4.38E−6
		100	0	0	8.78E−6
	SABC	50	0	0	4.51E−7
		100	0	0	3.73E−7
Schwefel's 2.21	ABC	50	56.35	3.93	1.09E−5
		100	81.76	1.91	1.14E−5
	SABC	50	1.99E−69	4.45E−69	1.34E−5
		100	3.77E−96	8.43E−96	4.49E−5
Schwefel's 2.22	ABC	50	2.54E−3	5.95E−4	1.13E−5
		100	7.96E−2	1.61E−2	1.13E−5
	SABC	50	5.01E−116	1.12E−115	1.33E−5
		100	2.45E−105	5.48E−105	1.49E−5
Sum Squares	ABC	50	4.46E−6	3.39E−6	1.09E−5
		100	1.63E−3	5.87E−4	1.15E−5
	SABC	50	2.08E−225	1.09E−329	1.34E−5
		100	5.72E−247	3.31E−332	1.51E−5
Griewank	ABC	50	2.35E−3	3.31E−3	2.01E−5
		100	1.35E−1	1.29E−1	2.18E−5
	SABC	50	0	0	2.29E−6
		100	0	0	2.66E−6
Rastrigin	ABC	50	7.58	2.61	1.28E−5
		100	78.62	5.82	1.42E−5
	SABC	50	0	0	1.54E−6
		100	0	0	1.35E−6
Ackley	ABC	50	6.67E−2	5.01E−2	1.44E−5
		100	3.97	3.04E−1	1.61E−5
	SABC	50	8.88E−16	0	1.68E−5
		100	8.88E−16	0	1.94E−5

由表 2 中数据可以看出：SABC 求得的平均最优值均优于 ABC，尤其是对 Griewank 和 Rastring 问题，SABC 在 50 和 100 维下都能达到理论最优解，说明 SABC 具有较高的收敛精度。同时，SABC 得到的标准差也较 ABC 低，说明 SABC 的稳定性也高于 ABC。另外，表 2 显示 SABC 的运行时间相比 ABC 未见明显增加，说明改进后算法增加的计算量不会对算法运行时间造成明显影响。

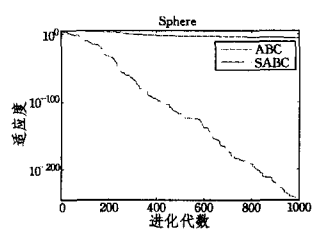

图 2 Sphere 函数平均进化曲线

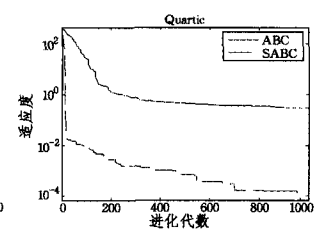

图 3 Quartic 函数平均进化曲线

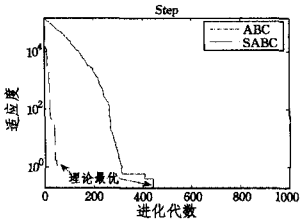

图 4 Step 函数平均进化曲线

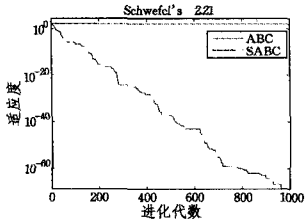

图 5 Schwefel's 2.21 函数平均进化曲线

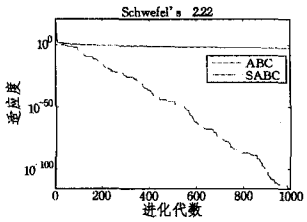

图 6 Schwefel's 2.22 函数平均进化曲线

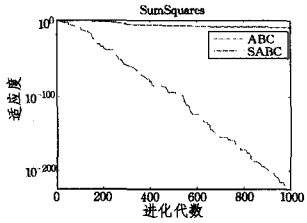

图 7 SumSquares 函数平均进化曲线

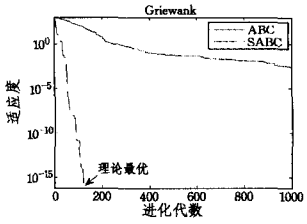

图 8 Griewank 函数平均进化曲线

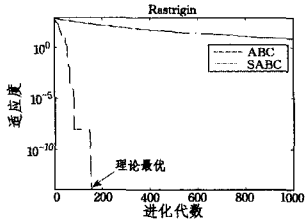

图 9 Rastrigin 函数平均进化曲线

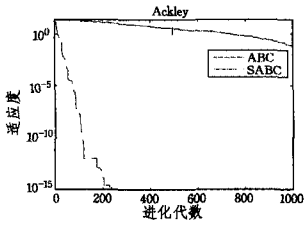

图 10 Ackley 函数平均进化曲线

从图 2—图 10 可以看出：SABC 仅需要较少的进化代数就能达到比 ABC 高的寻优精度，并且 SABC 未出现进化停滞的现象，表现出比 ABC 更强的收敛趋势和更快的速度（用进化代数表示），尤其是对 Griewank、Rastrigin 和 Step 函数，SABC 能在 200 次迭代内收敛到理论最优。从 9 个函数的平均进化曲线可以看出：SABC 能有效逃离早熟收敛，使算法在进化中始终保持朝理论最优快速靠近的趋势，让 SABC 具有较快的收敛速度，且逃离早熟收敛的能力强于 ABC。

4.4 参数变化时算法的适应性讨论

改进后的侦察蜂逃逸行为不再需要参数 *limit* 控制，SABC 只需要指定参数 Q（种群内个体数）。为评估 SABC 对参数的适应性，实验对参数 Q 取不同值时 SABC 与 ABC 的寻优精度进行了比较，指定函数维数为 100，种群内个体数量 Q 从 20 变化到 50，算法进化代数为 1000。限于篇幅，本文选择了两个有代表性的函数：Quartic（单模态函数）和 Ackley（非线性多模态函数）的实验数据进行对比，如表 3 所列（表 3 中的数据为算法运行 10 次求得的平均值，括号中数据为其标准差）。

表 3 种群内个体数变化的实验结果

	算法	Q=20	Q=30	Q=40	Q=50
Quartic	ABC	9.2E−1 (1.8E−1)	8.9E−1 (3.5E−1)	6.6E−1 (4.5E−1)	5.6E−1 (9.6E−2)
	SABC	4.1E−4 (2.3E−4)	5.7E−4 (3.9E−4)	7.2E−4 (5.1E−4)	3.2E−4 (3.4E−4)
Ackley	ABC	3.97 (3.0E−1)	4.10 (6.6E−1)	3.21 (3.9E−1)	3.27 (4.8E−1)
	SABC	8.9E−16 (0)	8.9E−16 (0)	8.9E−16 (0)	8.9E−16 (0)

从表 3 数据可以看出，随着种群内个体数量的变化，SABC 依然能得到比 ABC 高的寻优精度，说明参数改变不会对 SABC 的寻优效果造成明显影响。

4.5 与参考文献中的寻优结果比较

为了进一步验证本文改进算法的性能，选择相同的测试函数，将 SABC 算法在 10 次实验中求出的平均结果与文献中给出的结果作对比，如表 4 所列。需要特别说明的是，本文的实验条件比参考文献中的要苛刻很多，如：SABC 的进化代数是 1000 次，而文献[8-10]的算法进化代数均在 1500 次以上；对于测试函数的维数，本文选择的是 100 维，而文献[4,9,10]均不超过 50 维。由表 4 可以看出：SABC 仅以 1000 次进化就取得了不低于文献[8-10]进化 1500 次以上的结果，说明 SABC 相比上述文献中的改进方案具有更高的收敛速度和寻优精度。对于文献[4]，仅有 Sphere 和 Schwefel's 2.22 函数的优化结果优于 SABC，但 SABC 的求解复杂度明显大于文献[4]（SABC 为 100 维，文献[4]为 50 维），并且 SABC 对以上两函数的求解结果也达到了 10^{-100} 以上精度，说明 SABC 与文献[4]的改进策略相比同样占有一定优势。

表 4 SABC 与参考文献的对比结果

函数 \ 算法	SABC	文献[4]	文献[8]	文献[9]	文献[10]
Sphere	7.7E−242	0	1.3E−33	3.0E−16	—
Quartic	4.1E−4	6.5E−2	2.5E−1	—	—
Step	0	1.2E−15	—	—	—
Schwefel's 2.21	3.8E−96	19.67	—	—	—
Schwefel's 2.22	2.5E−105	0	—	—	—
SumSquares	5.7E−247	—	—	—	3.3E−21
Griewank	0	0	7.2E−17	2.7E−16	0
Rastrigin	0	0	1.4E−16	0	6.8E−14
Ackley	8.8E−16	8.8E−16	3.4E−13	2.9E−14	5.3E−13

结束语 为提高人工蜂群算法求解复杂函数优化问题的性能，本文对其进行改进，分析了原算法侦察蜂逃逸行为的不足，并提出了改进方案。具体地，设计了逃逸指标、自适应选择机制，改进了逃逸算子。数值实验结果表明：在函数优化问题求解中，本文改进方案能有效帮助算法逃离早熟收敛，使算法的收敛速度和求解精度得到提高，达到了预期效果。

参 考 文 献

[1] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Kayseri: Erciyes University, 2005

[2] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm[J]. Applied Soft Computing, 2008, 8(1):

687-697

[3] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm[J]. Applied Mathematics and Computation, 2009, 214(1): 108-132

[4] Li C Q, Niu P F, Xiao X J. Development and investigation of efcient articial bee colony algorithm for function optimization numerical function optimization [J]. Applied Soft Computing, 2012, 12: 320-332

[5] 罗钧，王强，付丽. 改进蜂群算法在平面误差评定中的应用[J]. 光学精密工程，2012，20(2)：422-430

[6] Horng M H. Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation[J]. Expert Syst Appl, 2011, 38(11): 13785-13791

[7] Öztürk C, Karabo˘ga D, Görkemli B. Artificial bee colony algorithm for dynamic deployment of wireless sensor networks[J]. Turk J Electr Eng Comput Sci, 2012, 20(2): 1-8

[8] 暴励，曾建潮. 一种双种群差分蜂群算法[J]. 控制理论与应用，2011，28(2)：266-272

[9] Wu B, Fan S H. Improved artificial bee colony algorithm with chaos[M]. Computer science for environmental engineering and ecoinformatics, Berlin: Springer, 2011: 51-56

[10] 罗钧，肖向海，付丽，等. 基于分段搜索策略的改进蜂群算法[J]. 控制与决策，2012，27(9)：1402-1405

[11] 张超群，郑建国，王翔. 蜂群算法研究综述[J]. 计算机应用研究，2011，28(9)：3201-3205

[12] Chen S H, Chen M C, Chang P C, et al. Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems[J]. Expert Systems with Applications, 2010, 37(9): 6441-6451

[13] Akay B, Karaboga D. Parameter tuning for the artificial bee colony algorithm[C]// International Conference on Computer and Computational Intelligence. 2009, 5796: 608-619

(上接第 244 页)

行分类。本文还通过最小包含球的核心集技巧建立了 DSSVM 与 MEB 之间的联系，在此基础上提出的 DSCVM 算法在解决较大样本分类问题上显现了一定的优势。

参 考 文 献

[1] Cortes C, Vapnik V N. Support vector networks [J]. Machine Learning, 1995, 20(3): 273-297

[2] Chang C-C, L in C-J. Training v-Support Vector Classifiers: Theory and Algorithms[J]. Neural Computation, 2002, 14: 1959-1977

[3] Hu M, Chen Y, Kwok J T. Building sparse multi-kernel SVM classifiers[J]. IEEE Transactions on Neural Networks, 2009, 20(5): 827-839

[4] 丁晓剑，赵银亮，李远成. 基于 SVM 的二次下降有效集算法[J]. 电子学报，2011，39(8)：1766-1770

[5] Tax D M J, Duin R P W. Support Vector Data Description[J]. Machine Learning, 2004, 54(1): 45-66

[6] Kim J S, Scott C D. L2 kernel classification[J]. IEEE Trans. PAMI, 2010, 32(10): 1822-1831

[7] Yang Y, Wright J, Ma Y, et al. Feature Selection in Face Recognition: A Sparse Representation Perspective[J]. IEEE Trans. PAMI, 2009, 31(2): 210-227

[8] Majumdar A, Ward R K. Classification via group sparsity promoting regularization[C]// IEEE International Conference on Acoustics, Speech and Signal Processing. 2009: 861-864

[9] Majumdar A, Ward R K. Improved Group Sparse Classifier[J]. Pattern Recognition Letters, 2010, 31(13): 1959-1964

[10] Tibshirani R. Regression shrinkage and selection via the lasso [J]. J. Royal. Statist. Soc B., 1996, 58(1): 267-288

[11] Donoho D L. For most large underdetermined systems of linear equations the minimal L1-norm solution is also the sparsest solution[J]. Commun. Pure Appl. Math., 2006(6): 797-829

[12] 宋枫溪，程科，杨静宇，等. 最大散度差和大间距线性投影与支持向量机[J]. 自动化学报，2004，30(6)：890-896

[13] Fine S, Scheinberg K. Efficient SVM training using low-rank kernel representations[J]. J. Mach. Learn. Res., 2001, 2: 243-264

[14] Deng Z H, Chung F L, Wang S T. FRSDE: Fast Reduced Set Density Estimator using Minimal Enclosing Ball Approximation [J]. Pattern Recognition, 2008, 41: 1363-1372

[15] Tsang I W, Kwok J T, Zurada J M. Generalized core vector machines[J]. IEEE Transactions on Neural Networks, 2006, 17(5): 1126-1140

[16] Chung F L, Deng Z H, Wang S T. From Minimum Enclosing Ball to Fast Fuzzy Inference System Training on Large Datasets [J]. IEEE Transactions on Fuzzy Systems, 2009, 17(1): 173-184

[17] Bădoiu M, Clarkson K L. Optimal core sets for balls[J]. Computational Geometry: Theory and Applications, 2008, 40(1): 14-22

[18] Bădoiu M, Har-Peled S, Indyk P. Approximate clustering via core sets[C]// Proc. 34th Annu. ACM Symposium Theory Comput., Montreal, QC, Canada, 2002: 250-257

[19] Asharaf S, Murty M N, Shevade S K. Multiclass core vector machine[C]// Proc. 24th ICML. Corvalis, OR, 2007: 41-48

[20] 胡文军，王士同，邓赵红. 适合大样本快速训练的最大夹角间隔核心集向量机[J]. 电子学报，2011，39(5)：1178-1184

[21] Smola A, Schölkopf B. Sparse greedy matrix approximation for machine learning[C]// ICML'00 Proceedings of the 7th International Conference on Mechine Learning Stanford, CA, June 2000: 911-918

[22] Asuncion A, Newman D J. UCI Machine Learning Repository [OL]. http://www.ics.uci.edu/ ~mlearn/MLRepository.html, Irvine, CA: University of California, School of Information and Computer Science