基于极值优化策略的改进的人工蜂群算法

葛 宇[1] 梁 静[2] 王学平[3]
(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)

摘 要 为提高人工蜂群算法在求解优化问题中的性能,结合极值优化策略提出一种改进的人工蜂群算法。改进算法基于极值优化策略高效率的寻优机制重新设计了原算法中跟随蜂的局部搜索方案,并具体给出了新方案的组元变异算子和最差组元判定规则。通过对优化问题中8个典型测试函数的仿真实验表明,与基本人工蜂群算法和已有的典型改进算法相比,改进算法在寻优精度和收敛速度上均有明显提高,在优化问题求解中体现出较强的寻优能力。

关键词 人工蜂群算法,极值优化策略,搜索方案,局部搜索
中图法分类号 TP18 **文献标识码** A

Improved Artificial Bee Colony Algorithms Based on Extremal Optimization Strategy

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

Abstract In order to enhance the performance of artificial bee colony algorithm in solving optimization problems, this paper proposed an improved artificial bee colony algorithm. The improved algorithm redesigns local search scheme of onlook bees based on evolution method of extremal optimization strategy, and implements operators of component mutations, formulates rules of worst component judgment. The simulation results of eight typical functions of optimization problems show that the proposed algorithm can attain significant improvement on accuracy and convergent speed, has a better solution capability, compared with the basic artificial bee colony algorithm and known improved algorithm.

Keywords Artificial bee colony algorithm, Extremal optimization strategy, Search scheme, Local search

1 引言

人工蜂群算法[1]是Karabog于2005年提出的一种群智能优化算法,具有控制参数少、易于实现、计算简单等优点[2]。近年来,人工蜂群算法广泛受到国内外学者的关注,并在神经网络[3]、图像处理[4]、传感器网络[5]等领域得到了应用。然而,和其他群智能优化算法一样,人工蜂群算法在优化问题求解中也存在收敛速度慢、计算精度不高的问题。针对上述缺点,不少学者提出了改进方案。如文献[6,7]提出了基于混沌序列的改进方法,该方法利用混沌序列的遍历性与随机性提高算法的搜索精度;文献[8]提出了一种采用levy变异的改进方案,该方法利用levy算子高效率的搜索特性来提高算法搜索速度;文献[9]为人工蜂群算法的变异操作增加一些调节项,以提高算法寻优效率;文献[10]将差分进化算法融合到人工蜂群算法中,将种群中个体随机分成两组,分别用两种算法同时寻优,以提高算法收敛速度。和以上改进方法不同,本文结合极值优化策略寻优机制来改进人工蜂群算法,重新设计了跟随蜂的局部搜索方案,以提高跟随蜂的局部搜索效率,从而提高算法收敛速度和寻优精度。仿真实验结果表明,本文改进方法能有效提高人工蜂群算法的性能。

2 人工蜂群算法

人工蜂群算法模拟蜜蜂群体寻找优质花蜜源的过程,把蜜蜂分为引领蜂、跟随蜂和侦察蜂3种角色[1,2],其中引领蜂负责全局搜索,跟随蜂负责局部搜索,侦察蜂则负责处理进化停滞的个体,算法利用3种蜜蜂间的协作来实现对最优花蜜源的搜索。在实际应用中,花蜜源代表优化问题的候选解(即种群中的个体),花蜜源的价值对应个体的适应度,蜜蜂采蜜过程就是算法搜索最优解的过程。

对最小值优化问题 $\mathrm{Min}(f(X))$,人工蜂群算法首先随机生成包含 SN 个个体的初始种群,每个个体 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})$ 是一个 D 维向量,其每一维分量也称为一个组元。根据角色分工不同,算法一次迭代的步骤如下:

Step 1 引领蜂从所有个体出发进行全局搜索。即对种群中每一个体 X_i 利用式(1)将其中一维分量进行变异,产生新个体。

到稿日期:2012-09-15 返修日期:2012-12-20 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。

$$x_{ij}'=x_{ij}+\phi\times(x_{ij}-x_{kj}) \quad (1)$$

式中，x_{ij} 表示个体 X_i 的第 j 维分量；x_{kj} 表示个体 X_k 的第 j 维分量元；j,k 均随机选择，且 $k\neq i$；ϕ 为[−1,1]间的随机数，控制搜索范围。x_{ij}'表示新产生的第 j 维分量，其对应的新个体记为 X_i'，根据式(2)判断是否被接受。

$$X_i=\begin{cases}X_i', & f(X_i')<f(X_i)\\ X_i, & f(X_i')\geq f(X_i)\end{cases} \quad (2)$$

Step 2　跟随蜂针对部分优秀个体的邻域作局部搜索：

Step 2-1　根据引领蜂的搜索结果，从种群中选择个体 X_i，其中适应度越好的个体被选择的概率越大。

Step 2-2　随机选择 X_i 的第 j 维分量 x_{ij}，根据式(1)对其进行变异，产生新个体 X_i'。

Step 2-3　根据式(2)决定是否接受新个体 X_i'。

Step 2-4　若未达到跟随蜂的最大搜索次数 SN(跟随蜂搜索次数与个体数 SN 相同)，则转向 Step 2-1；否则执行 Step 3。

Step 3　侦察蜂处理进化停滞的个体：如个体 X_i 连续 *limit* 次搜索都没有得到改善，说明此个体进化停滞，侦察蜂就用式(3)随机产生新的个体来代替之。

$$X_i=X_{\min}+R\times(X_{\max}-X_{\min}) \quad (3)$$

式中，R 表示区间[0,1]内的随机数，$X_{\max}$ 和 $X_{\min}$ 是解空间的上、下边界。

3　人工蜂群算法改进

在人工蜂群算的寻优过程中，引领蜂的工作是全局搜索，而跟随蜂主要在优秀个体附近作局部搜索，促使优秀个体朝目标位置演化。因此，跟随蜂的搜索效率直接影响到算法的收敛速度和寻优精度。以下先讨论极值优化策略的寻优机制；然后分析了跟随蜂的局部搜索方案，指出其中的不足，并进行改进；最后对改进算法计算量作了分析。

3.1　极值优化策略寻优机制

极值优化策略[11]是 Boettcher 提出的一种具有较高收敛精度的局部快速寻优方法，其寻优机制是：选择个体最差部分进行随机变异，并无条件接受变异产生的新个体。不同于已有的进化算法，极值优化策略源于复杂系统自组织临界的思想，从待求解问题内部变量之间的联系出发，将寻优过程视为复杂系统的演变过程[11]，即不用调整任何参数，仅通过个体内最差组元的变异，使个体总是朝最优结构演化。文献[12]指出，正是这种对个体内最差部分变异的演化机制大大增加了变异操作进一步产生优秀个体的可能性，使极值优化策略表现出很快的收敛速度和较高的寻优精度；同时配合个体无条件接受规则，让极值优化策略始终保持一定的波动性，有效避免了个体演化停滞。目前，极值优化策略已被成功应用到了粒子群算法[13]、混合蛙跳算法[14]等进化算法中，在避免算法停滞和提高算法收敛速度、寻优精度方面取得了很好的效果。

3.2　跟随蜂局部搜索方案的讨论与改进思路

人工蜂群算法中跟随蜂采用的搜索方案(Step 2-2、Step 2-3)会导致其局部搜索效率不高，具体分析如下：

(1)跟随蜂随机选择一个组元(即一维分量)进行变异，这会使个体演化出现盲目性，降低了变异操作产生更优个体的可能性。

(2)跟随蜂根据式(2)有条件地接受新个体。若式(1)中没能产生比变异前更优秀的个体，跟随蜂就放弃对该个体的更新。

从以上两点分析不难看出：跟随蜂盲目地选择组元进行变异，降低了产生更优秀个体的概率，使个体不被更新的可能性变大，增加了无效搜索次数，使跟随蜂的搜索效率下降，从而导致算法寻优精度不高，收敛缓慢甚至停滞。

为此，本文受极值优化策略的寻优机制启发，重新设计跟随蜂的局部搜索方案。具体思路是：让跟随蜂有目的地从优秀个体中选择最差组元进行变异，提高变异操作产生更优秀个体的效率；同时，采用无条件接受新个体的规则，使跟随蜂搜索保持活力，防止算法停滞。

3.3　基于极值优化策略改进后的跟随蜂局部搜索方案

为了提高跟随蜂的搜索效率，本文根据极值优化策略的寻优机制重新设计了跟随蜂的局部搜索方案。新方案具体描述为如下 3 步。

(1)最差组元判定规则依据组元变异的结果选出个体 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})$ 的最差组元 x^{worst}，其中 $x^{worst}\in\{x_{i1},x_{i2},\cdots,x_{iD}\}$。

(2)记录由 x^{worst} 变异产生的新个体 X^{new}。

(3)执行 $X_i=X^{new}$，无条件接受新个体 X^{new}。

下面具体讨论新方案中涉及的组元变异算子和最差组元判定规则。

3.3.1　组元变异算子的设计

为了让极值优化策略有更高的寻优效率，不少学者在实际应用中改进了其原有的随机组元变异算子[13,14]。本文基于尽量降低改进后算法计算量的考虑，在式(1)的基础上设计了一种带方向引导的组元变异算子，如式(4)：

$$x_{ij}'=x_{ij}+R\times(x_{bj}-x_{ij}),R\in[-1,1] \quad (4)$$

式中，待变异组元 x_{ij} 表示个体 X_i 的第 j 维分量；x_{bj} 为算法记录的历史最优个体 X_b 的第 j 维组元，用于引导变异方向。

可以看出：式(4)仅在式(1)的基础上替换了一个分量，有效控制了新搜索方案的计算量。

3.3.2　最差组元判定规则

个体 X_i 是一个 D 维向量，可以看作是由 D 个组元构成的非空集合 $X_i=\{x_{i1},x_{i2}\cdots x_{iD}\}$。判定个体的最差组元就需要对集合 X_i 进行操作。设有一映射 σ 使：

$$X_i\xrightarrow{\sigma}S \quad (5)$$

式中，映射 σ 具体描述为：对个体 X_i 的每个组元 x_{ij} 分别按式(4)进行变异，同时保持其他组元不变，得到对应新个体 s_j，记作 $\sigma(x_{ij})=s_j,j\in\{1,2\cdots D\}$，对于映射 σ，x_{ij} 就是 s_j 的原象；集合 $S=\{s_1,s_2,\cdots,s_D\}$表示 X_i 经映射 σ 产生的新个体集合。

基于以上说明，针对最小值优化问题，指定 $f(X)$为个体的适应度函数，本文采用的最差组元判定规则用如下步骤描述：

步骤 1　由式(5)将组元集合 X_i(即个体 X_i)映射到新个体集合 $S=\{s_1,s_2,\cdots,s_D\}$。

步骤 2　根据适应度函数 $f(X)$计算集合 S 中每个个体的适应度值 $f(s_j),j\in\{1,2,\cdots,D\}$。

步骤 3　结合步骤 2 的结果，从集合 S 中找出个体 s_w，其满足：$\exists s_w\in S$ 使 $f(s_w)=\mathrm{Min}(f(s_j)),j\in\{1,2,\cdots,D\}$。

步骤 4　定义映射 σ 中 s_w 的原象 x_{ij} 为个体 X_i 的最差组

元，记作 x^{worst}，同时 s_w 记录到新个体 X^{new} 中。

3.4 改进后的算法流程

对最小值优化问题 Min($f(X)$)，改进后的人工蜂群算法步骤如下：

Step 1 算法初始化，随机产生 SN 个个体，每个个体 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})$ 是一个 D 维向量，并指定侦察蜂控制变量 $limit$ 的值，用于判断个体是否陷入停滞，指定跟随蜂搜索次数和个体数 SN 相同，算法结束条件是达到最大迭代次数或达到指定精度。

Step 2 引领蜂对种群中每个个体 X_i 随机选择一组元 x_{ij} 利用式(1)进行变异，用式(2)对其更新。

Step3-1 跟随蜂按概率选择一个优秀个体 X_i。

Step3-2 按照 3.3.2 节中的判定规则找出最差组元，并记录其对应新个体 X^{new}。

Step3-3 执行 $X_i=X^{new}$，无条件接受新个体 X^{new}。

Step3-4 判断跟随蜂是否完成了 SN 次搜索，若完成，则执行 Step 4，否则执行 Step3-1。

Step 4 侦察蜂找出种群中未更新次数达到阈值 $limit$ 的个体，按式(3)随机更新。

Step 5 记录历史最优个体 X_b。

Step 6 判断是否满足算法终止条件，若满足，输出最优结果，否则转到 Step2。

3.5 改进方案的计算量分析

与原有的人工蜂群算法相比，本文只修改了跟随蜂的搜索方案，因此，本文对改进前后算法一次迭代中跟随蜂的计算量进行分析。算法的计算量通常以迭代次数再乘上每次迭代所消耗的乘法和加法次数来衡量。设一次加法计算量 T_1；一次乘法计算量 T_2；个体中组元个数(维数) D；跟随蜂的搜索次数与种群中个体数量 SN 相同。因适应度计算与具体问题有关，所以设适应度计算操作由 M 次加法和 N 次乘法组成。改进前后跟随蜂计算量分析如下。

改进前，跟随蜂只随机选择一个组元变异并计算适应度，需要执行 2 次加法、1 次乘法、1 次适应度计算操作。故改进前跟随蜂的计算量为：

$$O=SN\times[(2+M)\times T_1+(1+N)\times T_2] \tag{6}$$

改进后，跟随蜂要对每个组元变异，并计算其对应个体的适应度，需要做 $2D$ 次加法、D 次乘法、D 次个体适应度计算。故改进后算法的计算量为：

$$O'=SN\times[(2+M)\times T_1+(1+N)\times T_2]\times D \tag{7}$$

由此可以看出，本文改进方案的一次迭代计算量是原算法的 D 倍。从一次算法迭代来看计算量有所增加，但是改进算法能加快算法收敛，即减少了迭代次数。因此在固定精度的优化问题求解中改进方案增加的运算时间可以忽略，甚至还能比改进前有所减少，这一点在下文的实验数据(见表 3)中得到了验证。

4 仿真实验

4.1 实验设计

采用如下方法讨论本文提出的改进算法(以下记为 EABC)性能：(1)固定算法进化迭代次数，对比 EABC 与基本人工蜂群算法(以下记为 ABC)和文献[6]提出的一种典型的改进算法(以下记为 CABC)，评估算法的收敛速度和收敛精度；(2)固定算法收敛精度目标值，通过对比 EABC 与 ABC、CABC 的实验结果，评估算法达到该精度的收敛速度；(3) EABC 与文献报道的改进算法结果进行比较，进一步评估本文改进算法的效果。

实验选择了优化问题中 8 个典型的测试函数，如表 1 所列，其中 f_1-f_4 是单模态函数，f_5-f_8 是多模态函数。为了方便比较，本文调整了部分函数的形式，使其理论最优值都为 0。

本文中实验参数为：种群中个体数 $SN=40$，跟随蜂控制变量 $limit=D*SN$(其中 D 为待求解问题的维数)。实验在 2.6G 主频 CPU、2G 内存的计算机上完成。

表 1 测试函数

函数定义	变量范围
$f_1(x)=\sum_{i=1}^{D}x_i^2$	$\lvert x_i\rvert\leqslant 100$
$f_2(x)=\max\{\lvert x_i\rvert\}$	$\lvert x_i\rvert\leqslant 100$
$f_3(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$\lvert x_i\rvert\leqslant 100$
$f_4(x)=\sum_{i=1}^{D}ix_i^4+\mathrm{random}[0,1)$	$\lvert x_i\rvert\leqslant 1.28$
$f_5(x)=1+\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}(\cos\frac{x_i}{\sqrt{i}})$	$\lvert x_i\rvert\leqslant 600$
$f_6(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$\lvert x_i\rvert\leqslant 5.12$
$f_7=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$	$\lvert x_i\rvert\leqslant 32$
$f_8(x)=418.9829D-\sum_{i=1}^{D}(x_i\sin\sqrt{\lvert x_i\rvert})$	$\lvert x_i\rvert\leqslant 500$

4.2 固定迭代次数下的收敛精度、速度比较

固定算法迭代次数为 1000，实验用 ABC、CABC、EABC 3 种算法求解 8 个测试函数，在不同维数下的实验结果如表 2 所列。同时给出了 3 种算法对 6 个测试函数的进化曲线，如图 1—图 6 所示。

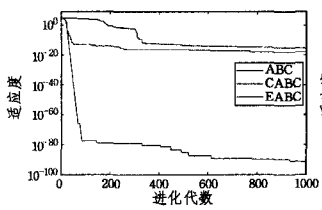

图 1 f_1 函数进化曲线

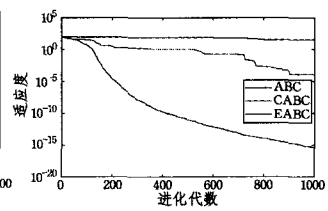

图 2 f_2 函数进化曲线

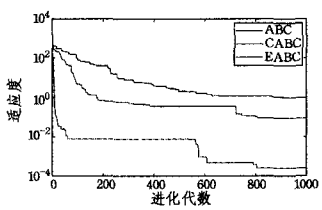

图 3 f_4 函数进化曲线

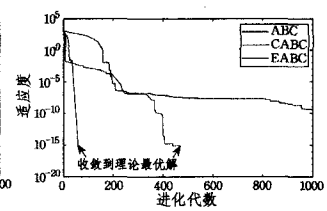

图 4 f_5 函数进化曲线

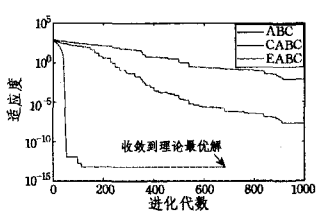

图 5 f_6 函数进化曲线

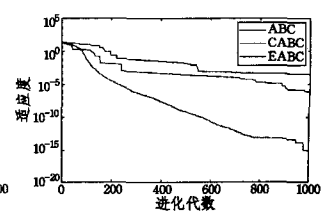

图 6 f_7 函数进化曲线

表 2　固定迭代次数下的寻优结果

函数	算法	维数	平均值	标准差	平均时耗(s)
f_1	ABC	50	4.18E−16	4.92E−16	1.31E−5
		100	9.18E−14	8.67E−14	5.94E−5
	CABC	50	2.51E−17	4.11E−17	3.25E−5
		100	8.91E−17	1.33E−17	9.76E−5
	EABC	50	4.59E−92	3.78E−92	2.22E−4
		100	1.31E−84	6.08E−84	9.10E−4
f_2	ABC	50	3.26E+1	3.05	2.53E−5
		100	6.83E+1	3.45	6.28E−5
	CABC	50	1.13E−4	1.01E−4	4.11E−5
		100	3.52E−3	2.32E−3	7.06E−5
	EABC	50	4.61E−16	1.14E−16	1.52E−4
		100	9.43E−11	1.76E−11	8.29E−4
f_3	ABC	50	2.86E−7	1.29E−7	4.21E−5
		100	3.93E−4	4.87E−4	6.55E−5
	CABC	50	1.64E−34	2.52E−34	5.32E−5
		100	3.28E−33	3.14E−33	7.62E−5
	EABC	50	1.04E−68	8.69E−68	7.24E−4
		100	8.66E−62	6.11E−62	9.35E−4
f_4	ABC	50	9.65E−1	7.71E−1	6.17E−5
		100	8.03E−1	2.55E−1	7.42E−5
	CABC	50	8.76E−2	8.05E−2	8.94E−5
		100	4.24E−1	3.31E−1	9.62E−5
	EABC	50	2.63E−4	3.82E−5	1.21E−4
		100	6.75E−4	1.85E−4	9.76E−4
f_5	ABC	50	4.52E−10	1.54E−10	9.12E−5
		100	3.75E−10	9.16E−10	9.87E−5
	CABC	50	0	0	7.22E−5
		100	0	0	7.26E−5
	EABC	50	0	0	8.11E−6
		100	0	0	8.96E−6
f_6	ABC	50	6.86E−3	8.45E−3	6.32E−5
		100	9.32E−1	3.55E−1	6.90E−5
	CABC	50	1.71E−8	2.64E−8	3.23E−5
		100	3.46E−8	8.21E−7	3.45E−5
	EABC	50	0	0	1.08E−5
		100	0	0	2.86E−5
f_7	ABC	50	3.21E−4	3.88E−7	6.95E−5
		100	2.53E−3	3.24E−3	7.32E−5
	CABC	50	7.25E−7	2.58E−7	8.15E−5
		100	5.48E−8	9.18E−7	9.45E−5
	EABC	50	8.88E−16	0	1.42E−4
		100	8.88E−16	0	2.33E−4
f_8	ABC	50	8.18E+2	3.03E+2	6.78E−5
		100	1.75E+3	4.39E+2	7.02E−5
	CABC	50	7.25	3.26	8.15E−5
		100	1.48E+1	1.18E+1	9.45E−5
	EABC	50	1.18E−6	1.59E−6	1.62E−4
		100	7.85E−6	7.43E−6	2.75E−4

由表 2 中的平均值和标准差结果可以看出：在 f_6、f_7 函数中 EABC 算法收敛到了理论最优解 0，其余测试函数中 EABC 算法也得出了比 ABC、CABC 算法更接近理论最优解的结果，且标准差也都优于另外两种算法，说明 EABC 算法有很好的收敛精度和稳定性。另外，表 2 显示 EABC 算法的运行时间大于另外两种算法，这是因为算法在跟随蜂搜索阶段增加了计算量所致。

从图 1－图 6 中的进化曲线可以看出：EABC 算法仅需要较少的迭代次数就能达到比 CABC 和 ABC 高的寻优精度，并且在进化中 EABC 始终保持朝最优解进化的趋势。说明 EABC 算法的收敛速度和避免算法停滞的能力都高于 ABC、CABC 算法。

4.3　固定收敛精度下的收敛速度和成功率比较

EABC、CABC、ABC 3 种算法在指定收敛精度下经过 10 次独立运行后的平均迭代次数如表 3 所列。其中：成功率＝达到精度的实验次数/总实验次数。若算法迭代 1000 次都未达到指定精度，则视为寻优失败，成功率记为 0。

表 3　固定目标精度下的实验结果

函数	收敛精度	算法	平均迭代次数	成功率	平均时耗(s)
f_1	1.0E−30	ABC	—	0	1.32E−5
		CABC	—	0	3.28E−5
		EABC	45	100%	8.94E−6
f_2	1.0E−5	ABC	—	0	5.55E−5
		CABC	879	30%	4.16E−5
		EABC	215	100%	4.06E−5
f_3	1.0E−10	ABC	976	20%	4.21E−5
		CABC	385	100%	5.32E−5
		EABC	29	100%	7.21E−6
f_4	1.0E−1	ABC	—	0	6.13E−5
		CABC	724	100%	8.89E−5
		EABC	24	100%	7.15E−6
f_5	0	ABC	—	0	9.19E−5
		CABC	471	100%	7.24E−5
		EABC	60	100%	8.14E−6
f_6	0	ABC	—	0	6.37E−5
		CABC	—	0	3.26E−5
		EABC	686	100%	9.11E−5
f_7	1.0E−10	ABC	—	0	6.98E−5
		CABC	984	10%	8.19E−5
		EABC	570	100%	9.42E−5
f_8	1.0E−3	ABC	—	0	6.71E−5
		CABC	—	0	8.13E−5
		EABC	55	100%	8.25E−6

从表 3 中可以看出，对于所有测试函数，EABC 算法都能取得 100% 的求解成功率，并且迭代次数都小于 ABC 和 CABC 算法，尤其在部分函数中 EABC 只迭代了 20 多次，说明本文改进算法有很快的收敛速度。正是由于这一点，表 3 中除了 f_6、f_7 函数外 EABC 算法表现出了最少的运行时耗。

4.4　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 EABC 算法在 10 次实验中求出的平均结果与文献[7-10]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：EABC 的迭代次数是 1000 次，而其余算法迭代次数均在 2000 以上；对于测试函数的维数，本文选择的是 100，而文献[7,9]中仅为 30。由表 4 可以看出，EABC 以 1000 次迭代就取得了不低于其它 4 种算法 2000 次以上迭代得到的效果，说明 EABC 算法有更好的收敛速度和寻优精度。

表 4　EABC 与参考文献对比求解结果

函数＼算法	EABC	文献[7]	文献[8]	文献[9]	文献[10]
f_1	1.31E−84	2.99E−16	2.65E−15	5.21E−16	1.26E−33
f_4	6.75E−4	—	—	—	2.5E−1
f_5	0	2.70E−16	1.22E−15	7.40E−17	7.22E−17
f_6	0	0	4.95E−10	0	1.37E−16
f_7	8.88E−16	2.94E−14	2.41E−8	2.96E−14	3.41E−13
f_8	7.85E−6	3.82E−4	6.04E+2	—	—

结束语　本文改进人工蜂群算法，在对算法分析的基础上，指出跟随蜂搜索方案的不足，基于极值优化策略重新设计了跟随蜂搜索方案，并具体实现了新搜索方案下跟随蜂的组元变异算子和最差组元判定规则。实验结果表明，本文改进方法通过提高跟随蜂的搜索效率，有效避免了算法停滞，使算

法的收敛速度和求解精度得到了提高，是一种简单高效的改进方法。

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] Irani R, Nasimi R. Application of artificial bee colony-based neural network in bottom hole pressure prediction in underbalanced drilling[J]. Journal of Petroleum Science and Engineering, 2011, 78(1): 6-12

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

[5] Öztürk C, Karaböga D, Görkemli B. Artificial bee colony algorithm for dynamic deployment of wireless sensor networks[J]. Turkish Journal of Electrical Engineering and Computer Sciences, 2012, 20(2): 1-8

[6] 罗钧，李研. 具有混沌搜索策略的蜂群优化算法[J]. 控制与决策，2010，25(12)：1913-1916

[7] Wu Bin, Fan Shu-hai. Improved artificial bee colony algorithm with chaos[A]// Computer Science for Environmental Engineering and Ecoinformatics, 2011[C]. Berlin: Springer, 2011: 51-56

[8] Rajasekhar A, Abraham A, Pant M. Levy mutated artificial bee colony algorithm for global optimization[A]// IEEE International Cnference on Systems, Man and Cybernetics, 2011[C]. Anchorage: IEEE, 2011: 655-662

[9] Guo Peng, Cheng Wen-ming, Liang Jian. Global artificial bee colony search algorithm for numerical function optimization[A]// 7th International Conference on Natural Computation, 2011[C]. Shanghai: IEEE, 2011: 1280-1283

[10] 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

[11] Boettcher S, Percus A G. Extremal optimization: Methods derived from co-evolution[C]// Proc. of the Genetic and Evolutionary Computation Conf. San Francisco: Morgan Kaufmann. 1999: 825-832

[12] 齐洁，汪定伟. 极值优化算法综述[J]. 控制与决策，2007，22(10)：1081-1090

[13] 陈泯融. 基于极值动力学的优化方法及其应用研究[D]. 上海：上海交通大学，2008

[14] 骆剑平，陈泯融. 混合蛙跳算法及其改进算法的混合轨迹及收敛性分析[J]. 信号处理，2010，26(9)：1428-1433

(上接第 232 页)

扰，进一步提高识别的精度。

参 考 文 献

[1] Bader G D, et al. BIND-the biomolecular interaction network database[J]. Nucleic Acids Res., 2003, 31(1): 242-245

[2] Peri S, et al. Development of human protein reference database as an initial platform for approaching systems biology in humans [J]. Genome Res., 2003, 13: 2363-2371

[3] U. S. National Library of Medicine. PubMed [OL]. http://www.ncbi.nlm.nih.gov/pubmed/

[4] Ono T, Hishigaki H, et al. Automatic extraction of information on protein-protein interactions from the biological literature[J]. Bioinfornatics, 2001, 17(2): 155-161

[5] Huang M L, Zhu X Y, Hao Y, et al. Discovering patterns to extract protein-protein interactions form full text[J]. Bioinformatics, 2004, 20(18): 3604-3612

[6] Fundel K, et al. RelEx_Relation extraction using dependency parse trees[J]. Bioinformatics, 2007, 23(3): 365-371

[7] Temkin J M, Gllder M R. Extraction of protein interaction information from unstructured text using a context-tree grammar [J]. Bioinformatics, 2003, 19(16): 2046-2053

[8] Bunescu R C, Mooney R J. Subsequence kernels for relation extraction[C]// Proceedings of the 19th Aunual Conference on Neual Information Processing Systems. Cambridge. MA, USA: MIT Press, 2005: 171-178

[9] Niu Y, et al. Evaluation of linguistic features useful in extraction of interactions from PubMed; Application to annotating known, high-throughput and predicted interactions in I²D[J]. Bioinformatics, 2010, 26(1): 111-119

[10] 唐楠，杨志豪，等. 基于多核学习的医学文献蛋白质关系抽取[J]. 计算机工程，2011，37(10)：184-186

[11] Medin D L, Robert L G, Gentner D. Similarity involving attributes and relations: Judgments of similarity and difference are not inverses[J]. Psychological Science, 1990, 1(1): 64-69

[12] Toch E, Reinhartz I, Berger I, et al. Humans, semantic services and similarity: A user study of semantic Web services matching and composition[J]. Journal of Web Semantics: Science, Services and Agents on theWorldWideWeb, 2011, 9: 16-28

[13] Bos J. A survey of Computational Semantics: Representation, Inference and Knowledge in Wide-Coverage Text Understanding [J]. Language and Linguistics Compass, 2011, 5(6): 336-366

[14] Turney P D. Similarity of semantic relations[J]. Computational Linguistics, 2006, 32(3): 379-416

[15] Nakov P, Hearst M A. Solving relational similarity problems using the web as a corpus[C]// Proceedings of ACL-08: HLT. Columbus, Ohio, USA: Association for Computational Linguistics, 2008: 452-460

[16] U. S. National Library of Medicine, Bookshelf [OL]. http://www.ncbi.nlm.nih.gov/books/NBK5500/#chapter1. ESearch

[17] University of Illinois at Urbana Champaign. Sentence Segmentation tool[OL]. http://cogcomp.cs.illinois.edu/page/tools_view/2

[18] Apache Software Foundation. Apache OpenNLP 1.5.2-incubating[OL]. http://opennlp.apache.org/index.html

[19] The Stanford Natural Language Processing Group. Stanford Parser[OL]. http://nlp.stanford.edu/software/lex-parser.shtml

[20] Thorsten Joachims. SVM-light Support Vector Machine[OL]. http://www.cs.comell.edu/people/tj/svm_light