文章编号: 1001-0920 (2015) 02-0316-05 DOI: 10.13195/j.kzyjc.2013.1442

基于分治策略的改进人工蜂群算法

李田来, 刘方爱, 王新华

(山东师范大学 信息科学与工程学院，济南 250014)

摘 要: 人工蜂群 (ABC) 算法存在着收敛速度不够快、易陷入局部最优的缺陷. 针对这一问题, 提出一种改进的人工蜂群 (DCABC) 算法. 应用反学习的初始化方法产生初始解, 引入分治策略对蜜源进行优化, 在采蜜蜂发布更新的蜜源信息后, 跟随蜂选择最优蜜源, 并采用分治策略进行迭代优化. 通过对经典测试函数的反复实验及与其他算法的比较, 表明了所提出的算法具有良好的加速收敛效果, 提高了全局搜索能力与效率.

关键词: 人工蜂群；改进算法；分治策略；反学习

中图分类号: TP301.6 **文献标志码:** A

Modified artificial bee colony algorithm based on divide-and-conquer strategy

LI Tian-lai, *LIU Fang-ai*, *WANG Xin-hua*

(School of Information Science and Engineering，Shandong Normal University，Ji'nan 250014，China. Correspondent: LI Tian-lai，E-mail：litianlai0313@163.com)

Abstract: As a kind of swarm optimization algorithm with good performance, the artificial bee colony (ABC) algorithm is presented in recent years. However, it exist some disadvantages, such as the convergence speed is not fast enough, easy to fall into local optimum and etc. In order to solve this problem, an improved algorithm called DCABC is presented. In this algorithm, the opposition-based learning method is employed when producing the initial population, the divide-and-conquer strategy is adopted to greed update food resources. After employed bees releasing updated food source information, onlookers choose optimal resource based on the divide-and-conquer strategy. Experiments are conducted on a set of 6 benchmark functions, and the results show that DCABC has better performance than several other ABC-based algorithms, especially on the accelerating convergence and the global search ability and efficiency.

Keywords: artificial bee colony；modified algorithm；divide-and-conquer strategy；opposition-based learning

0 引 言

人工蜂群 (ABC) 算法是受蜜蜂采蜜的群体过程启发而提出的仿生智能算法. 较为完善的ABC算法由Karaboga于2005年提出[1], 并且已经成功地应用于数值优化等多个方面[2-5], 取得了较好的测试结果. 此外, Pan等[6]提出了用ABC算法求解流水车间调度问题, Singh[7]用ABC算法解决最小生成树问题, 均验证了该算法的优越性. 尽管如此, ABC算法仍然存在着过早收敛、易陷入局部最优、精度不高等缺点. 针对上述问题, 学者们提出了多种改进策略来优化其性能[8-11]. 然而, 到目前为止, 如何在提高算法收敛性的同时避免陷入局部最优仍然是一个难题.

本文受分治策略的启发, 提出一种改进的人工蜂群 (DCABC) 算法. 首先采用反学习方法进行初始化, 应用分治策略优化蜜源, 跟随蜂通过迭代搜索实现充分优化. 在Matlab编程环境下, 选取经典高维函数进行反复实验, 通过比较传统的ABC算法与改进的ABC算法, 证明了该算法能够显著提高进化效率和全局优化性能.

1 传统的人工蜂群算法

ABC算法[12]的基本原理由3部分构成: 蜜源、采蜜蜂 (EF)、待工蜂 (UF). 其中, 待工蜂又分为侦察蜂和跟随蜂. 蜜蜂的基本行为模式有3种: 搜索蜜源、蜜

收稿日期: 2013-10-17；**修回日期:** 2014-01-21.

基金项目: 国家自然科学基金项目(90612003)；山东省自然科学基金项目(ZR2013FM008)；山东省科技发展计划项目(2011GGH20123).

作者简介: 李田来(1982−), 男, 博士生, 从事无线网络、智能计算的研究；刘方爱(1962−), 男, 教授, 博士生导师, 从事分布式计算、无线网络等研究.

源招募、放弃蜜源. 在初始阶段, 蜜蜂分为采蜜蜂和跟随蜂两种, 分别各占一半. 每只采蜜蜂在原蜜源采蜜, 同时随机搜索新蜜源, 每个蜜源代表解空间范围内的一个可能解, 蜜源通过一个适应度值来衡量优劣, 并进行优胜劣汰; 然后, 采蜜蜂招募跟随蜂按照其得到的蜜源适应度值成比例的概率选择蜜源进行采蜜, 同时随机搜索其他蜜源, 当新蜜源的适应度值更高时, 跟随蜂变为采蜜蜂, 并取代原蜜源位置. 当某个蜜源在规定的迭代次数后仍没有找到更好的蜜源时, 则强制放弃该蜜源, 重新随机搜索一个新的蜜源.

假设蜜蜂总数为 $N_{\rm s}$, 其中采蜜蜂种群规模为 $N_{\rm e}$, 跟随蜂种群规模为 $N_{\rm u}$ (一般定义 $N_{\rm e}=N_{\rm u}$), 个体向量的维度为 D, $S=R^D$ 为个体搜索空间. $S^{N_{\rm e}}$ 为采蜜蜂种群空间. 若 $X_i\in S\,(i\leqslant N_{\rm e})$ 是 $N_{\rm e}$ 个个体, 则 $X=(X_1,X_2,\cdots,X_{N_{\rm e}})$ 代表一个采蜜蜂种群. 用 $X(i)$ 表示第 i 代采蜜蜂种群, 适应度函数为 $f:S\to R^+$.

$n=0$ 时, 采用下式随机生成 $N_{\rm s}$ 个可行解 $(X_1, X_2,\cdots,X_{N_{\rm s}})$, $j\in\{1,2,\cdots,D\}$, 并将 $N_{\rm s}$ 个解按适应度函数值排名, 前 $N_{\rm e}$ 个解构成 $X(0)$:

$$X_i^j=X_{\min}^j+\mathrm{rand}(0,1)(X_{\max}^j-X_{\min}^j). \tag{1}$$

对于第 n 步的采蜜蜂, 采用下式在当前位置向量附近邻域搜索新的位置:

$$V_i^j=X_i^j+\phi_i^j(X_i^j-X_k^j). \tag{2}$$

其中: $j\in\{1,2,\cdots,D\}$, $k\in\{1,2,\cdots,N_{\rm e}\}$, 且 $k\neq i$, k、j 均自动生成为 $[-1,1]$ 之间的随机数, 同时应该保证 $V\in S$.

采用下式在新位置向量 V_i 和原向量 X_i 中进行贪婪选择, 记作 $T_{\rm s}=S^2\to S$:

$$P\{T_{\rm s}(X_i,V_i)=V_i\}=\begin{cases}1,\ f(V_i)\geqslant f(X_i);\\ 0,\ f(V_i)<f(X_i).\end{cases} \tag{3}$$

各跟随蜂根据下式选择采蜜蜂, 并在其邻域内执行式 (2) 的操作:

$$P\{T_{\rm s1}(X)=X_i\}=\frac{f(X_i)}{\sum\limits_{m=1}^{N_{\rm e}}f(X_m)}. \tag{4}$$

重复式 (2) 和 (3) 的相应操作, 记下最优适应度值 $f_{\rm best}$ 以及与之对应的 $j\in\{1,2,\cdots,D\}$.

2　基于分治策略的改进蜂群算法

2.1　反学习的初始化方法

种群的初始化工作直接影响着算法的全局收敛速度和解的质量. 在传统的 ABC 算法中, 开始阶段都是随机生成初始解, 这种方法比较简单, 但在一定程度上会限制 ABC 算法的求解效率. 为解决上述问题, 本文采用文献 [13] 中的基于反向学习的群体初始化, 提出了算法 1. 首先在开始阶段随机生成初始可行解, 然后为每个初始可行解计算相对应的反向解, 最后对这两类解进行比较, 把适应度值优的解作为初始种群, 从而提高初始解的质量. 算法 1 的流程如图 1 所示.

/*反学习种群初始化*/
1. 随机产生规模为 $N_{\rm s}$ 的种群;
2. for $i=1$ to $N_{\rm s}$ do
3. for $j=1$ to D do
4. $x[i][j]=x[\min][j]+\mathrm{rand}(0,1)(x[\max][j]-x[i][j])$
/*反学习种群初始化结束*/
/*反学习进化跳转*/
5. for $i=1$ to $N_{\rm s}$ do
6. for $j=1$ to D do
7. $ox[i][j]=x[\min][j]+x[\max][j]-x[i][j]$
8. 选择集合 $X(N_{\rm s})$,$OX(N_{\rm s})$ 中适应度函数值最好的 $N_{\rm s}$ 个个体作为初始种群
9. 选取适应度函数值排名前二分之一的 $X[i]$ 作为采蜜蜂初始种群 $X(0)$
/*反学习进化跳转结束*/

图 1　算法 1 的流程

2.2　基于分治策略的进化过程

传统的 ABC 算法中, 文献 [14] 分析到采蜜蜂在当前位置向量附近邻域进行搜索新的位置时, 按式 (2) 搜索具有比较大的随机性, 从而导致更新不稳定. 如图 2 所示.

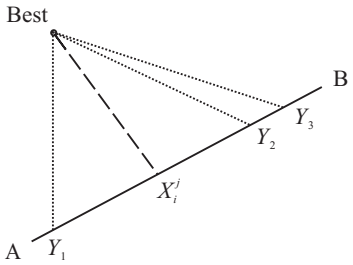

图 2　随机搜索过程

AB 代表以 X_i^j 为中心的 $\pm(X_i^j-X_k^j)$ 搜索范围, 由于 ϕ 为 [−1,1] 的随机数, 其搜索结果可能取到 AB 内的任意一个值. 若取到比 X_i^j 更优的 Y_1, 则在进行贪婪选择时替代 X_i^j, 更新蜜源; 若取到比 X_i^j 更差的 Y_2 或者 Y_3, 则不会更新. 显然, 这种随机选择的方式

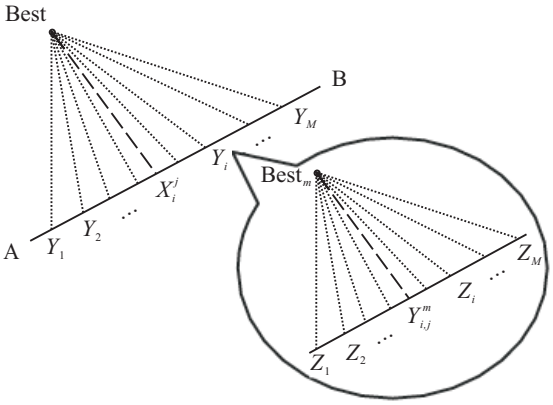

图 3　分治搜索过程

使得食物源的更新成功率较低. 为此, 该文献提出了分段搜索策略, 将该维空间分成了若干段进行贪婪搜索, 提高了搜索效率, 但也存在陷入局部最优的风险.

本文受文献[14]的启发, 将分治策略应用到该问题中, 如图3所示.

首先用下式将该维空间划分为 M 个区间, m 的取值范围为 $[0, M]$:

$$Y_{i,j}^{m} = X_i^j + \frac{2m-M}{M}(X_i^j - X_k^j),\ m \in [0, M]. \quad (5)$$

然后再用下式将每一个区间 $[Y_{i,j}^{m}, Y_{i,j}^{m+1}]$ 进一步划分为 N 个子区间:

$$V_{i,j}^{m,n} = Y_{i,j}^{m} + \sin\left(\frac{(n-\mathrm{rand}(0,1))}{2N}\pi\right)\times (Y_{i,j}^{m} - Y_{i,j}^{m+1}),\ n \in [1, N]. \quad (6)$$

其中: n 的取值范围为 $[1, N]$, 正弦函数的取值范围为 $[0, 1]$.

在此选用正弦函数的优点在于正弦函数并不是平均划分子区间, 这样体现了该子区间寻优过程的随机性. 计算 N 个子区间的适应度函数值, 通过下式选出目标函数值最好的作为该区间对应的代表值:

$$U_{i,j}^{m} = \mathrm{best}\{f(V_{i,j}^{m,n})\},\ n \in [1, N]. \quad (7)$$

将上述得到的 M 个代表值 $U_{i,j}^{m}$ 代入蜜源计算适应度函数值, 然后与原值 X_i^j 的适应度函数值按照下式进行计算:

$$W_i^j = \min\{f(U_{i,j}^{m}) - f(X_i^j)\},\quad f(U_{i,j}^{m}) - f(X_i^j) > 0,\ m \in [1, M]. \quad (8)$$

首先要求差值必须为正值, 然后从中选择最小值 W_i^j, 其对应的区间代表值 $U_{i,j}^{m}$ 即作为精英个体更新种群, 在此选择正数最小值 W_i^j 是为了保证在进化方向不后退的前提下, 避免过早陷入局部最优, 特别是多峰函数. 基于分治策略的改进蜂群算法(算法2)如图4所示.

```
/*算法2 (DCABC)*/
1. 随机产生N_s个蜜源, 函数循环次数为Limit
2. 用算法1产生初始种群, 并与采蜜蜂一一对应
3. while (FE< Limit) do
4.    for i=1 to N_s do
5.       Updated(i) = false
6. 按照式(5)划分为M个子区间
7. for i=1 to M do
8.    按照式(6)随机挑选N个区间代表值Z_1, Z_2, ..., Z_N]
9.    应用式(7) 进行最优选择
10.   fbest(m) = f(Z_1), zbest = Z_1
11.   for k = 2 to N do
12.      if f(Z_k) > fbest(m) do
13.         fbest(m) = f(Z_k), zbest = Z_k
14. 将选出的M个代表值代入式(8)进行计算
15. 找到对应的W_i^j和U_ij^m进行更新
16. 各跟随蜂选择蜜源的最优解 f(X_i)best, 按照分治策略进行迭代更新
17. for i=1 to N_s
18.   if (! Update(i)) then
19.      跟随蜂用算法1重新初始化该蜂源
20. goto 3, 直到更新当前的最优解
/*算法2 (DCABC) 结束*/
```

图 4 算法 2 (DCABC) 的流程

3 仿真实验

3.1 DCABC 与 ABC 的收敛性能比较

为了测试本文提出的基于分治策略的改进蜂群算法的性能, 选取参考文献[14-15]中的6个高维Benchmark函数进行实验, 如表1所示. 分别对测试函数、搜索范围和最优值进行了定义, 设定空间维数为40, 种群规模为60, 最大迭代次数为2 000, 最大循环次数为200, DCABC算法的 M、N 取4. 在Matlab编程环境下, 选择相同条件, 将DCABC算法和ABC算法以及采用反学习初始化方法的人工蜂群(OABC)算法分别独立实验20次, 得到实验结果如表2所示.

表 1 测试函数

Function defination	Range
$f_1(X) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$f_2(X) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-10, 10]^n$
$f_3(X) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2\times [1+10\sin^2(\pi y_{i+1})] + (y_n-1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	
$y_i = 1 + \frac{1}{4}(x_i + 1)$	
$u_{x_i,a,k,m} = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0, & -a \leqslant x_i \leqslant a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$	$[-50, 50]^n$
$f_4(X) = \sum_{i=1}^{n} i x_i^2$	$[-10, 10]^n$
$f_5(X) = \sum_{i=1}^{n-1} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$
$f_6(X) = \frac{1}{4\,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$

表 2 维数为 40、迭代次数为 2 000 时, 3 种算法的性能比较

Fun	Dim	Alg	Best	Mean	SD
f_1	40	ABC	1.748 32 e-10	9.662 3 e-10	4.237 9 e-10
		OABC	3.296 6 e-14	1.719 0 e-11	1.254 6 e-11
		DCABC	3.640 1 e-23	1.409 3 e-22	1.175 9 e-22
f_2	40	ABC	0.941 3	2.091 2	2.158 7
		OABC	0.654 0	1.824 7	1.725 1
		DCABC	0.173 2	0.472 9	0.425 2
f_3	40	ABC	5.648 7 e-11	1.934 7 e-10	1.249 3 e-11
		OABC	7.250 2 e-13	5.832 9 e-11	8.339 1 e-12
		DCABC	4.573 6 e-16	9.354 9 e-14	8.752 5 e-14
f_4	40	ABC	2.796 2 e-14	1.438 2 e-12	4.324 1 e-13
		OABC	8.224 8 e-16	4.821 4 e-13	1.912 8 e-15
		DCABC	2.975 9 e-22	1.031 0 e-22	6.135 4 e-22
f_5	40	ABC	7.563 8 e-16	3.361 7 e-14	8.349 1 e-15
		OABC	5.835 2 e-19	9.221 5 e-15	4.724 8 e-16
		DCABC	0	0	0
f_6	40	ABC	2.357 7 e-13	5.489 3 e-12	8.921 1 e-13
		OABC	9.368 1 e-16	7.974 2 e-13	1.019 7 e-13
		DCABC	0	0	0

在实验结果中, 最优值和最差值体现了解的质量; 平均值体现了算法达到的精度, 反映了算法的收敛速度; 标准差反映了算法的稳定性和鲁棒性. 从表2可以看出: 对于本文选取的6个函数, OABC算法的结果优于ABC算法, 证明反学习初始化方法有助于提高解的质量和收敛速度; DCABC算法的结果优于OABC算法和ABC算法, 能够获得更好的最优值, 并且DCABC算法的平均值和标准差更小, 这说明虽然反学习初始化方法有助于提高解的质量和收敛速度, 但本文引入的分治策略更能够有效地提高解的质量、收敛速度以及稳定性. DCABC算法和ABC算法的部分进化曲线对比如图5～图7所示.

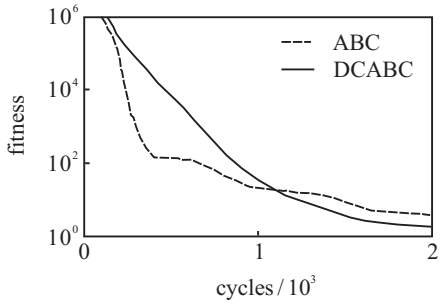

图 5　f_2 的收敛性能比较

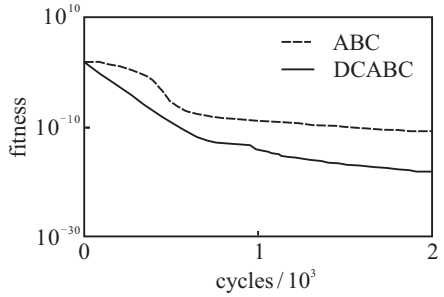

图 6　f_4 的收敛性能比较

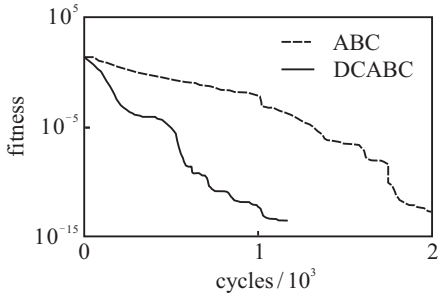

图 7　f_6 的收敛性能比较

3.2　DCABC与几种改进的ABC算法比较

本文提出的DCABC算法重点在于提高收敛能力. 为了进一步考察该算法的性能, 本文与文献[14-15]中所提到的基于不同选择机制的改进蜂群算法进行比较, 选用 $f_1(x)$、$f_2(x)$ 作为测试函数, 参照文献[16-17]的参数设置, 即种群规模为50, 函数的搜索范围分别为 $[-100, 100]$ 和 $[-30, 30]$. 维数为30时, 最大迭代次数为1 000; 维数为50时, 最大迭代次数为2 000.

本文提出的DCABC算法在初始化种群过程中进行了反学习初始化, 而文献[14-15]中并未进行. 为进一步分析分治策略对收敛性能的作用, 特将未进行反学习初始化的DCABC算法(DCABC-O)加入到实验中, 各种算法分别运行30次, 两个测试函数的进化性能结果如表3和表4所示.

表 3　f_1 的性能结果比较

Alg	Dim	Mean	SD
ABC	30	1.829 458 e-09	2.639 270 e-09
	50	9.564 480e-12	1.212 776 e-11
DABC	30	9.798 135 e-12	1.063 763 e-11
	50	2.138 416 e-14	2.318 155 e-14
RABC	30	1.978 102 e-12	2.033 197 e-12
	50	9.066 074 e-16	7.463 074 e-16
TABC	30	1.514 169 e-12	1.380 414 e-12
	50	4.228 150 e-15	3.779 910 e-15
SABC	30	1.730 030 e-20	1.350 080 e-20
	50	1.473 120 e-19	2.964 700 e-19
DCABC-O	30	1.335 471 e-22	1.035 415 e-22
	50	9.376 141 e-22	1.027 468 e-21
DCABC	30	4.742 630 e-23	3.964 826 e-23
	50	6.972 518 e-22	8.457 213 e-22

表 4　f_2 的性能结果比较

Alg	Dim	Mean	SD
ABC	30	5.220 163	5.278 101
	50	3.283 130	3.986 572
DABC	30	4.002 284	2.780 462
	50	3.185 141	2.264 986
RABC	30	2.355 272	1.769 675
	50	1.331 720	1.301 473
TABC	30	2.093 640	1.698 724
	50	1.460 538	1.191 361
SABC	30	0.655 239	0.509 349
	50	0.368 808	0.530 536
DCABC-O	30	0.542 016	0.510 472
	50	0.294 731	0.264 370
DCABC	30	0.524 760	0.495 238
	50	0.253 791	0.21 4863

从表3和表4可以看出: DCABC算法优于DCABC-O算法, 表明了反学习初始化方法有利于提高收敛速度; DCABC-O算法优于文献[14-15]中的算法, 再次表明本文引入分治策略对收敛有明显的加速效果.

4　结　论

本文在ABC算法的基础上, 提出DCABC算法, 通过引入反学习的初始化方法和分治搜索过程, 提高了算法的收敛速度和进化能力, 并且避免了早熟. 对经典函数的寻优实验结果表明, 本文提出的算法优化效率较高, 优化性能稳定, 改善了全局进化能力. 然而, 该算法也有其局限性. 对于分治搜索过程中的两个参数的设置不能实现自适应, 需要根据具体问题具体分

析, 从而达到计算精度与算法复杂度的均衡. 如何将该算法应用到多目标优化、无线网络路由、无线传感器节点定位、图像处理等领域是下一步的研究工作.

参考文献(References)

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

[2] 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.

[3] Karaboga D, Basturk B. On the performance of artificial bee colony(ABC) algorithm[J]. Applied Soft Computing, 2008, 8(1): 687-697.

[4] Karaboga D, Akay B, Ozturk C. Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks[M]. Modeling Decisions for Artificial Intelligence. Berlin: Springer Heidelberg, 2007: 318-329.

[5] Karaboga N. A new design method based on artificial bee colony algorithm for digital IIR filters[J]. J of the Franklin Institute, 2009, 346(4): 328-348.

[6] Pan Q K, Fatih Tasgetiren M, Suganthan P N, et al. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem[J]. Information Sciences, 2011, 181(12): 2455-2468.

[7] Singh A. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem[J]. Applied Soft Computing, 2009, 9(2): 625-631.

[8] Akay B, Karaboga D. A modified artificial bee colony algorithm for real-parameter optimization[J]. Information Sciences, 2012, 192: 120-142.

[9] Kang F, Li J, Xu Q. Structural inverse analysis by hybrid simplex artificial bee colony algorithms[J]. Computers and Structures, 2009, 87(13): 861-870.

[10] Alatas B. Chaotic bee colony algorithms for global numerical optimization[J]. Expert Systems with Applications, 2010, 37(8): 5682-5687.

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

[12] 段海滨, 张祥银. 仿生智能计算[M]. 北京: 科学出版社, 2011: 90-94.
(Duan H B, Zhang X Y. Bionic intelligent computing[M]. Beijing: Science Press, 2011: 90-94.)

[13] Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition-based differential evolution[J]. IEEE Trans on Evolutionary Computation, 2008, 12(1): 64-79.

[14] 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): 1402-1406.
(Luo J, Xiao X H, Fu L, et al. Modified artificial bee colony algorithm based on segmental-search strategy[J]. Control and Decision, 2012, 27(9): 1402-1406.)

[15] Bao L, Zeng J. Comparison and analysis of the selection mechanism in the artificial bee colony algorithm[C]. The 9th Int Conf on Hybrid Intelligent Systems. Shenyang: IEEE, 2009: 411-416.

(责任编辑: 齐 霁)