基于排名映射概率的混沌人工蜂群算法

张新明 李晓安 何文涛 王鲜芳
(河南师范大学计算机与信息工程学院 新乡 453007)

摘 要 针对人工蜂群算法(Artificial Bee Colony algorithm,ABC)因直接采用函数值映射的概率选择食物源而引起过早收敛和陷入局部最优以及优化精度不高的问题,提出一种基于排名映射概率的混沌人工蜂群算法(Chaotic Artificial Bee Colony algorithm based on Rank mapping probability,CABC-R)。首先利用目标函数值的排名映射获取选择食物源的概率,然后构建基于排名映射概率的人工蜂群算法以便能够维持种群的多样性,获得较好的全局最优解,最后创建较高寻优精度的新型局部混沌优化算法精确寻找最优解。对10个标准测试函数进行了仿真,结果表明,CABC-R算法不仅优化效果更准确而且更能跳出局部最优,有效地找到全局最优解,优于标准的ABC、JADE、MSEP和RABC算法。

关键词 优化方法,人工蜂群算法,混沌优化算法,排名映射概率,直接映射概率
中图法分类号 TP181 **文献标识码** A

Chaotic Artificial Bee Colony Algorithm Based on Rank Mapping Probability

ZHANG Xin-ming LI Xiao-an HE Wen-tao WANG Xian-fang
(College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China)

Abstract In view of the shortcomings of artificial bee colony algorithms, such as the low convergence rate and being trapped into local optimums owing to choosing the food source based on direct mapping probability, and low optimization precision, a chaotic artificial bee colony optimization algorithm based on rank mapping probability (CABC-R) was proposed in this paper. The proposed search process was divided into two different phases: in the first one an ABC global optimizer based on rank mapping probability was created to get a global solution, in the second one the local chaotic optimization algorithm was gotten to obtain more precise an optimum. The simulation results on 10 standard test complicated functions indicate that the proposed optimization algorithm is rapid and effective, and that it outperforms the current global optimization algorithms such as ABC, JADE, MSEP and RABC.

Keywords Optimization method, Artificial bee colony algorithm (ABC), Chaotic optimization algorithm (COA), Rank mapping probability, Direct mapping probability

1 引言

人工蜂群算法(artificial bee colony,ABC)[1]是一种模拟蜜蜂采蜜行为的新型群智能算法,它具有操作简便、易于实现及控制参数少等特点,成为现代智能优化领域等的研究热点[2-7]。ABC算法是一种新颖的群集智能优化算法,因此还有诸如收敛速度慢、易陷于局部最优和优化精度不高等许多问题亟待解决。混沌运动所具有的内在随机性、遍历性、"规律"等特点,使混沌搜索能在一定范围内按其自身的"规律"不重复地遍历每一个状态,因此混沌优化算法(chaotic optimization algorithm,COA)[8]有较强的局部搜索能力,在某些场合更容易跳出"谷"、"沟"、"槽"等局部极值点,加之其算法简单易用,在数字图像处理等许多领域中得到应用[8,9]。因此为了提升ABC算法的搜索能力,许多学者使用混沌特性对ABC算法进行了不同的尝试。如文献[10]使用混沌初始化,其基本思想是利用混沌序列生成初始食物源种群,从而使得初始种群在定义域内的分布更加均匀;文献[10,11]采用混沌侦察蜂,其思想是在侦察蜂阶段,针对陷入局部极值的食物源,利用混沌序列进行局部搜索,进而产生较优的食物源。文献[12]在文献[10,11]的基础上,在观察蜂阶段结束之后执行混沌局部搜索算子,增强算法的局部搜索能力。本文针对标准的ABC算法存在的问题,提出了一种基于排名映射概率的混沌人工蜂群优化算法(Chaotic Artificial Bee Colony algorithm based on Rank mapping probability,CABC-R)。首先在观察蜂阶段使用排名映射概率选择新食物源,由此构建排名映射概率的人工蜂群优化算法(Artificial Bee Colony algo-

到稿日期:2013-05-28 返修日期:2013-07-07 本文受国家自然科学基金项目(61173071),河南省高校创新人才支持计划项目(2012HASTIT011)资助。

张新明(1963-),男,副教授,硕士生导师,CCF会员,主要研究方向为智能优化算法、数字图像处理和模式识别等,E-mail:xinmingzhang@126.com;李晓安(1961-),男,讲师,主要研究方向为工程优化;何文涛(1986-),男,硕士生,主要研究方向为数字图像处理和计算机教学论;王鲜芳(1969-),女,教授,硕士生导师,主要研究方向为机器学习、非线性系统的建模与优化控制等。

rithm based on Rank mapping probability，ABC-R)，克服标准ABC算法收敛速度慢、容易出现“早熟”等缺点，然后构造新型的混沌优化算法，最后将ABC-R全局粗搜索和COA局部精搜索相结合，以获得更具有竞争力的优化算法和优化结果。

2 标准的人工蜂群算法

ABC算法模拟自然界蜜蜂采蜜的过程，通过不同工种蜜蜂之间的合作完成采蜜过程的各阶段任务，并通过食物源信息的收集与共享寻找问题的最优解，与传统的优化方法相比，ABC算法具有更好的优化性能[3,4]。算法将蜜蜂分为侦查蜂、采蜜蜂和观察蜂3种工种，采蜜蜂和观察蜂的数量与食物源的数量相等。食物源的位置代表优化问题的一个可能的解，食物源的花蜜量对应解的适应度。首先，ABC算法随机产生初始解，初始解的个数为N，即N个采蜜蜂、观察蜂和食物源。每个解x_i是一个D维向量，其中$i=1,2,\cdots,N$，D是优化参数的个数。完成初始化后，蜜蜂开始对初始解进行循环搜索。采蜜蜂对记忆中的食物源(原始解)位置产生改变从而找到一个新的食物源(新解)，并确认新的食物源的花蜜量，即计算新解的适应度。如果新解的适应度高于原始解的适应度，则采蜜蜂将记忆新解，忘记原始解。所有采蜜蜂完成搜索后回到蜂巢，将食物源信息(解的位置和适应度)与观察蜂共享。观察蜂根据搜集到的信息，按照与适应度相关的概率选择一个食物源位置，并像采蜜蜂一样对记忆中的位置进行改变得到新解并确认新解的适应度。如果新解的适应度高于记忆中解的适应度，则用新解替代记忆中的解。观察蜂选择食物源的概率p_i可根据式(1)进行计算。

$$p_i=fit_i/\sum_{n=1}^{N}fit_n \tag{1}$$

式中，fit_i为第i个解的适应度。

采蜜蜂和观察蜂对原始解的邻域进行搜索的操作为：

$$v_{ij}=x_{ij}+r_{ij}(x_{ij}-x_{kj}) \tag{2}$$

式中，v_{ij}为新的食物源位置，x_{ij}为食物源的第j维位置，x_{kj}为随机选择的不等于i的食物源的第j维位置，r_{ij}为$[-1,1]$间的随机数。如果在采蜜过程中，食物源经若干次搜索不变，相应的采蜜蜂变成侦查蜂，随机搜索新食物源，用其代替初始标记食物源中的相应位置，确定最终食物源。按照上述方式反复循环迭代，直到达到算法的终止条件。

3 基于排名映射概率的混沌人工蜂群算法

在标准的ABC算法中，观察蜂选择食物源时通过式(3)计算适应度，然后通过式(1)获取食物源的概率(简称直接映射概率)，然后依据轮盘赌方式选择，这是一种基于贪婪策略的选择方式，会使种群多样性降低，从而导致过早收敛和提前停滞的现象[7]。另外，ABC算法的优化精度不高，这些问题必须得到解决。本文利用排名映射概率来提高ABC算法的全局搜索能力，利用混沌优化算法提高局部搜索能力和寻优精度。

3.1 排名映射概率

在标准的ABC算法中，对于函数优化问题，如求函数最小值，假如f_i是第i个解的目标函数值，则第i个解的适应度常常表示为：

$$fit_i=1/abs(f_i) \tag{3}$$

式中，$abs(f_i)$是求f_i的绝对值。

但这种方式会导致如下两种情况发生：(1)当f_i为inf或者说无意义时，式(3)无法计算；而f_i为0时，式(3)无意义，导致ABC算法无限循环；(2)当某个f_i特别小，大大小于其他函数值时，通过式(3)和式(1)计算的概率接近1，而其它解的概率接近0，如此在每次迭代中，观察蜂通过概率p_i仅仅只选择第i个解的邻域进行搜索，使种群多样性降低到最小，从而引起过早收敛和提前停滞。

鉴于以上情况，本文提出基于排名映射获取选择概率的方法(求函数的最小值)，其过程描述如下：

(1)计算每个食物源的目标函数值f_i。

(2)处理无意义的函数值和排序。首先定义无意义的函数值，如果f_i无意义，则令$f_i=inf$；随后对所有解的函数值按由小到大的顺序进行排名，得到各个解的排名s_i，很显然无意义函数值映射的名次较大。

$$s_i=\mathrm{Sort}(f_i),s_i\in[1,2,\cdots,N] \tag{4}$$

(3)归一化。为了使适应度值整体变化较为平缓，对整个排名进行归一化。

$$u_i=s_i/N,u_i\in[1/N,2/N,\cdots,1] \tag{5}$$

(4)依据式(6)计算适应度值。

$$fit_i=1/u_i \tag{6}$$

(5)依据式(1)得到选择食物源的概率p_i。

以上过程可以看出：①每个解的适应度值与其函数值没有直接的关系，而是通过排名映射获取，与其函数值的排名成反比，函数值越大，其对应的排名越大，反之排名越小；②对于无意义的函数值其排名靠后，适应度较小，排名映射概率较小，但仍能保证有机会更新其旧解；③由于各个解的排名不会相同，即使函数值相同(在函数值相同时，会按照原解的顺序排名)，各个解的概率也决不相同；④由于每个解都有一个排名，因此排名映射概率绝不会等于0，也不会等于1，更不会为inf。

因此，综上所述，新的优化算法不会出现基于直接映射概率的标准ABC算法所出现的情况，保证种群的多样性，避免出现“早熟”和提前停滞现象的发生。

3.2 ABC-R算法

ABC-R算法的基本步骤如下：

步骤0 初始化参数。先设定$limit$、M和N的值，其中M为最大迭代次数，N为采蜜蜂、观察蜂和食物源的数量，$limit$为一个食物源被放弃之前搜索不变的最大次数。

步骤1 按照式(7)随机产生$2N$个位置。

$$v_{ij}=a_j+rand(0,1)\times(b_j-a_j) \tag{7}$$

式中，v_{ij}为第i个蜜蜂第j维搜索后的位置；b_j和a_j代表第j维参数的上下界，$rand(0,1)$是$[0,1]$之间的随机数。

步骤2 计算适应度。对于以上$2N$个食物源，计算其目标函数值，选取适应度值高的N个位置作为循环搜索的初始食物源位置。

步骤3 初始化食物源状态变化数组lim，即令$\mathrm{lim}_i=0,i=1,\cdots,N$。

步骤4 采蜜蜂在食物源附近按式(2)搜索新食物源，并

对新食物源按式(8)作越界处理。

$$v_{ij}=\max(a_j,\ v_{ij}),v_{ij}=\min(b_j,v_{ij}) \tag{8}$$

步骤 5　计算新食物源的适应度，比较前后食物源优劣，若搜索后食物源优于搜索前食物源，则代替先前食物源，并设 $\lim_i=0$，否则 $\lim_i=\lim_i+1$。

步骤 6　观察蜂根据采蜜蜂释放的花蜜信息，计算排名映射概率，依据此概率按轮盘赌方式选择食物源，并在其附近按式(2)搜索新食物源，对新食物源按式(8)作越界处理。

步骤 7　同步骤 5。

步骤 8　判断是否出现侦查蜂，若某些食物源经 *limit* 次搜索不变，放弃该食物源，即当 $\max(\lim)\geqslant limit$ 时，相应的采蜜蜂变成侦查蜂，按式(7)随机产生新食物源。

步骤 9　从步骤 4 开始重新搜索，直到满足终止条件(即迭代次数超过了 M)，停止搜索，获得全局最优解。

3.3　新型混沌优化子模块

混沌优化算法[8,9]的基本思想就是把混沌变量线性映射到优化变量的取值区间，然后进行混沌搜索。本文创建新型的混沌优化算法采用 Logistic 混沌映射系统，它的模型如式(9)所示：

$$z_{k+1}=4z_k(1-z_k) \tag{9}$$

受文献[13]的随机局部优化模块的启示，首先提出一种新型的混沌局部优化子模块，其主要步骤如下：

步骤 1　设置迭代次数为 M_2、精度控制参数 c_0 和混沌初始值 $z_{0,j}$。

步骤 2　外循环。$j_0=1$，其中 j_0 为外循环变量。

步骤 3　内循环。设 $k=1$，其中 k 为内循环变量。

步骤 4　按式(9)进行混沌映射。

步骤 5　混沌载波。按式(10)进行载波。

$$x_{k,j}{}'=x^*_{k,j}+\lambda_{k,j}(2z_{k,j}-1) \tag{10}$$

式中，$\lambda_{k,j}$ 为调节参数，在此混沌优化子模块中，我们规定 $\lambda_{k,j}=|x^*_{k,j}|/(c_0{}^{j_0})$，$x_k^*$ 为当前最优解，$\lambda_{k,j}$ 用到了当前最优解，这样更有利于局部优化。

步骤 6　精搜索。令 $x=x_k{}'$ 并按式(8)作越界处理；计算相应的性能指标 $f(x)$；if $f(x)\leqslant f^*$，then $f^*=f(x)$，$x_k^*=x$ else if $f(x)>f^*$，then 放弃 x，其中，f^* 为当前的最优值。

步骤 7　如果 $k<M_2$，则 $k=k+1$ 并且转到步骤 4，否则转到步骤 8。

步骤 8　如果 $j_0>5$，则终止搜索，输出最优解 x_k^* 和最优值 f^*，否则 $j_0=j_0+1$ 并且转到步骤 2。

3.4　CABC-R 算法流程

为了得到精度更高的最优解，将上节介绍的 M_1 个混沌局部优化子模块串接起来构造新型的 COA。那么 CABC-R 算法的主要步骤是：首先运行 ABC-R 算法获得全局粗略解，然后运行新型的 COA，即，运行上节介绍的混沌局部优化子模块 M_1 次，进行精确的局部优化得到问题的最终解。在运行混沌局部优化子模块时，精度控制参数按式(11)作调整，这样更利于提高优化精度。

$$c_0=1.05c_0 \tag{11}$$

整个 CABC-R 算法的运行流程如图 1 所示。

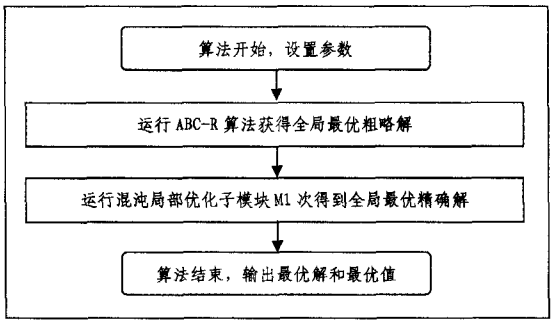

图 1　本文提出算法的流程图

从 CABC-R 算法流程可以看到，与传统的 COA 相比，(1)新型 COA 中的混沌优化子模块由于选用的调节参数是依据当前最优解的绝对值，而且每次内循环中运行混沌优化子模块都是基于最新的 x_k^*，在 $|x_k^*|$ 的尺度下自动调整参数，$|x_k^*|$ 越大，搜索空间越大，反之，越小，这样比传统的 COA 采用固定的 $\lambda_{k,j}$ 更能提高搜索精度。(2)一方面，随着 j_0 的增大，局部搜索空间减少，搜索精度提高；另一方面，在局部优化子模块迭代次数增加的同时，精度控制参数 c_0 也会慢慢增加，见式(11)，搜索空间进一步减少，更有利于搜索精度的提高。所以新型 COA 能极大地提高优化精度。

4　仿真实验及结果分析

为了验证 CABC-R 算法的有效性，限于篇幅，选取 10 个常用测试优化算法性能的典型高维函数进行比较，考察它们搜索得到的全局最优解、优化精度及效率等情况。在本实验中，算法采用 MATLAB R2010A 语言实现，所有实验在 DELL Intel 酷睿 370 主频为 2.4G 的 CPU 和内存为 2G DDR3 RAM 的笔记本上进行。算法的参数设置为：CABC-R、ABC-R 和 ABC 3 种算法中 N 都为 20；*limit* 都为 $N*D$；各算法的最大迭代次数为：对于 CABC-R，有两部分，分别是 ABC-R 和 COA，ABC-R 迭代 3000 次或者 1000 次，具体来说，对于 f_{01}、f_{02}、f_{04}、f_{08}、f_{09} 和 f_{10}，$M=3000$，而其它函数，$M=1000$。混沌局部优化算法参数设置是：局部混沌优化子模块迭代次数为 50 次，即 $M_1=50$，$M_2=20$，c_0 初始值设置为 2。对于 ABC-R 和 ABC，为了公平起见，保证它们的函数评价次数与 CABC-R 算法相等，所以对于 f_{01}、f_{02}、f_{04}、f_{08}、f_{09} 和 f_{10}，$M=3100$，而其它函数，$M=1100$。这样对应的函数评价次数见表 1 第 3 列，其中，这些评价次数都未包含初始化位置后的评价次数 $2N$。各算法均随机运行 50 次，对于实验 1，选取测试函数搜索的最好值 Best、搜索的平均值 Mean、搜索的最差值 Worst、方差 Std、最大目标函数的评价次数(Max_FFS)及运行时间(Time)为评价标准来考察各算法的寻优性能。对于实验 2，选取测试函数的平均值和方差来考察各算法的寻优性能。为了使比较更具有普适性，这 10 个函数代表着不同的情况，如有单峰函数和多峰函数、可分离和不可分离、连续和不连续等，这 10 个函数表达式和全局最优解等情况如下：

$$f_{01}=\sum_{i=1}^{30}x_i^2,\ -100\leqslant x_i\leqslant 100$$

f_{01} 为 Sphere 函数，为可分离的高维对称单峰函数，其维数为 30，全局最优值为：$\min(f_{01})=f_{01}(0,0,\cdots,0)=0$。

$f_{02}=\sum_{i=1}^{30}|x_i|+\prod_{i=1}^{30}|x_i|$，$-10\leqslant x_i\leqslant 10$

f_{02}为 Schwefel's Prolem 2.22 函数，为多极值函数，其维数为 30，全局最优值为：$\min(f_{02})=f_{02}(0,0,\cdots,0)=0$。

$f_{03}=\sum_{i=1}^{30}(\lfloor x_i+0.5\rfloor)^2$，$-100\leqslant x_i\leqslant 100$

f_{03}为 Step 阶跃函数，是由很多平滑的高地和陡脊组成的离散单峰函数，其维数为 30，全局最优值为：$\min(f_{03})=f_{03}(0,0,\cdots,0)=0$。

$f_{04}=-\sum_{i=1}^{30}(x_i\sin\sqrt{|x_i|})$，$-500\leqslant x_i\leqslant 500$

f_{04}为 Schwefel's Prolem 2.26 函数，为不可分离的多峰函数，且带有一定的欺骗性，其维数为 30，全局最优值为：$\min(f_{04})=f_{04}(420.968746,\cdots)\approx -12569.5$。

$f_{05}=\sum_{i=1}^{30}(x_i^2-10\cos(2\pi x_i)+10)$，$-100\leqslant x_i\leqslant 100$

f_{05}为 Rastrigin 函数，此函数是在 Sphere 函数的基础上，使用了余弦函数来产生大量的局部最小值，是一个典型的具有大量局部最优点的复杂多峰函数，其维数为 30，全局最优值为：$\min(f_{05})=f_{05}(0,0,\cdots,0)=0$。

$f_{06}=20+e-20\exp[-\frac{1}{5}\sqrt{\frac{1}{30}\sum_{i=1}^{30}x_i^2}]-\exp[\frac{1}{30}\sum_{i=1}^{30}\cos(2\pi x_i)]$，$-32\leqslant x_i\leqslant 32$

f_{06}为 Ackley 函数，是连续、旋转、不可分离的高维多峰函数，其维数为 30，全局最优值为：$\min(f_{06})=f_{06}(0,0,\cdots,0)=0$。

$f_{07}=1+\sum_{i=1}^{30}\frac{x_i^2}{4000}-\prod_{i=1}^{30}\cos(\frac{x_i}{\sqrt{i}})$，$-600\leqslant x_i\leqslant 600$

f_{07}为 Griewank 函数，是连续、旋转和不可分离的高维多峰函数，其维数为 30，全局最优值为：$\min(f_{07})=f_{07}(0,0,\cdots,0)=0$。

$f_{08}=\frac{\pi}{30}\{10\sin^2(\pi y_1)+\sum_{i=1}^{29}[(y_i-1)^2(1+10\sin^2(\pi y_{i+1}+1))]+(y_{30}-1)^2\}+\sum_{i=1}^{30}u(x_i,10,100,4)$，$y_i=1+\frac{x_i+1}{4}$，$i=1,\cdots,30$，$-50\leqslant x_i\leqslant 50$

f_{08}为 Generalized Penalized1 函数，为高维多峰函数，维数为 30，全局最优值为：$\min(f_{08})=f_{08}(1,1,\cdots,1)=0$。

$f_{09}=0.1\{10\sin^2(3\pi x_1)+\sum_{i=1}^{29}[(x_i-1)^2(1+\sin^2(3\pi x_{i+1}+1))]+(x_{30}-1)^2[1+\sin^2(2\pi x_{30})]\}+\sum_{i=1}^{30}u(x_i,5,100,4)$，$-50\leqslant x_i\leqslant 50$

f_{09}为 Generalized Penalized2 函数，为高维多峰函数，其维数为 30，全局最优值为：$\min(f_{09})=f_{09}(1,1,\cdots,1)=0$。

其中，在 f_{08}和 f_{09}中的 u 为：

$$u(x_i,\alpha,k,m)=\begin{cases}k(x_i-\alpha)^m, & x_i>\alpha\\ 0, & -\alpha\leqslant x\leqslant\alpha\\ k(-x_i-\alpha)^m, & x_i<-\alpha\end{cases}$$

$f_{10}=\frac{1}{100}\sum_{i=1}^{100}(x_i^4-16x_i^2+5x_i)$，$-10\leqslant x_i\leqslant 10$

f_{10}为 Himmelblau 函数[6]，为高维多峰函数，维数为 100，全局最优值为：$\min f_{10}=f_{10}(-2.9043,\cdots)\approx -78.3323$。

实验 1　ABC、ABC-R 和 CABC-R 优化效果对比。表 1 列出了 3 种优化算法对 10 个函数的优化结果、目标函数最大评价次数(Max_FFS)和运行时间(Time)，表中优者用黑体表示。由表 1 数据对比可以看出，在所有的标准测试函数中，无论是解的质量，还是算法的收敛精度和稳定性，CABC-R 算法优化性能最好，ABC-R 算法次之，最差的是 ABC 算法。特别地，在这 3 种算法中，CABC-R 算法几乎在所有 10 个测试函数上都取得了最好的优化结果。ABC-R 算法在 f_{03}、f_{04}、f_{08}、f_{09}和 f_{10}上有与 CABC-R 算法几乎媲美的优化性能，而在 f_{01}、f_{02}、f_{05}、f_{06}和 f_{07}上相差甚远。但除 f_{03}外，在其它测试函数上 ABC-R 算法的优化性能优于 ABC 算法，尤其在 f_{01}、f_{02}、f_{08}和 f_{09}上，更是远远胜过 ABC 算法。ABC 算法相比于其它两种算法，除了在 f_{03}上有较为理想的优化效果外，其它函数的优化效果最差，这证明了标准的 ABC 算法由于采用直接映射概率来选择食物源并在其附近按式(2)搜索新食物源会出现"早熟"和提前停滞现象，而采用排名映射的 ABC-R 算法能很好地避免这些现象，但优化精度不高，如除了 f_{03}和 f_{08}外，其它函数的优化精度都不理想。而采用 COA 后使最终的优化精度得以提高。另外，从运行时间和目标函数的评价次数看，评价次数最多的是 f_{01}、f_{02}、f_{04}、f_{08}、f_{09}和 f_{10}这 6 个函数，而 f_{10}的维数最高是 100，其运行时间在 8 秒左右。反过来说，如果 ABC 算法需要达到 CABC-R 算法优化性能，需要增加迭代次数(假定增加迭代次数能够提高优化性能)，也就增加了目标函数的评价次数，增加了计算复杂度，耗时就会增加。所以，在达到同样的优化效果情况下，CABC-R 算法目标评价次数少，运行时间少。此实验说明：本文提出的 ABC-R 算法是可行的，新型的 COA 提高了优化精度，效果是显著的。

实验 2　CABC-R 算法与 MSEP 算法[14]、JADE 算法[15]和 RABC 算法[5]优化效果对比。表 2 是将本文提出的 CABC-R 算法与 MSEP 算法、JADE 算法和 RABC 算法在 9 个函数上进行优化比较的结果。其中，MSEP 算法、JADE 算法和 RABC 算法优化数据来自相应的文献，相比于实验 1，CABC-R 算法在函数 f_{02}上做了调整，即将 M_1 从 50 调整到 100，其它参数保持不变，COA 的目标函数评价次数从 4000 变为 8000，提高了搜索精度，这样总的评价次数为 128000，见表 2 的第 6 列第 4 行。从表 2 看出，(1)在优化效果(均值 Mean 和方差 Std)上，与 RABC 算法相比，CABC-R 算法在 f_{08}和 f_{09}上优化效果稍逊于 RABC 算法，而其他 7 个函数不亚于 RABC 算法的优化效果，5 个函数即 f_{01}、f_{02}、f_{04}、f_{06}和 f_{07}的 CABC-R 算法优化效果优于 RABC 算法，而在 f_{03}和 f_{05}上的优化效果相同。CABC-R 算法与 JADE 算法相比，在 8 个可比较的测试函数中，二者在 f_{03}、f_{05}、f_{07}、f_{08}和 f_{09}上的优化效果相当，而在 f_{01}、f_{02}和 f_{06} 3 个函数上，CABC-R 算法优于 JADE 算法。CABC-R 算法与 MSEP 算法相比，除了在 f_{03}优化效果一样外，而在其他 8 个函数上，CABC-R 算法大幅度优于 MSEP 算法。(2)从 MSEP 算法、JADE 算法和 RABC 算法的 Max_FFS(见表 2 第 2 列)和 CABC-R 算法的 Max_FFS(见表 2 倒数第 2 列)看出，CABC-R 算法评价次数少，

尤其是在 f_{03}、f_{04}、f_{05}、f_{06} 和 f_{07} 上，前者是后者的 4～7 倍，这说明本文提出的 CABC-R 算法收敛速度快，效果明显。

总之，不管从实验 1 还是从实验 2 来看，本文提出的 CABC-R 算法优化性能是出色的，在大多数函数上优于标准的 ABC 算法、RABC 算法、MSEP 算法和 JADE 算法。

表 1　3 种优化算法的计算结果以及耗时

Functions	Algorithms	Max_FFS	Time/s	Mean	Std	Best	Worst
	CABC-R	124,000	3.8943	**5.0024e−94**	**2.1046e−93**	**3.5022e−109**	**1.2769e−92**
f_{01}	ABC-R	124,000	3.9584	1.5372e−40	4.0388e−40	2.8642e−43	2.5390e−39
	ABC	124,000	3.9394	2.0190e−28	6.3119e−28	7.0523e−31	4.4647e−27
	CABC-R	124,000	4.2897	**3.1384e−51**	**1.8168e−50**	**6.3999e−56**	**1.2855e−49**
f_{02}	ABC-R	124,000	4.5335	1.7563e−21	2.3792e−21	1.1664e−22	1.1677e−20
	ABC	124,000	4.5127	8.7921e−15	4.7121e−15	1.1521e−13	1.3321e−15
	CABC-R	044,000	1.3403	0	0	0	0
f_{03}	ABC-R	044,000	1.4800	0	0	0	0
	ABC	044,000	1.4892	0	0	0	0
	CABC-R	124,000	4.2956	**−1.25695e+04**	**5.8799e−12**	**−1.25695e+04**	**−1.25695e+04**
f_{04}	ABC-R	124,000	4.5123	−1.25695e+04	6.1328e−12	−1.25695e+04	−1.25695e+04
	ABC	124,000	4.3616	−1.25695e+04	1.2421e−10	−1.25695e+04	−1.25695e+04
	CABC-R	044,000	1.4180	**0**	**0**	**0**	**0**
f_{05}	ABC-R	044,000	1.5850	0.0398	0.1969	1.8005e−08	0.9950
	ABC	044,000	1.5799	7.6330e−06	1.7087e−05	3.4689e−09	1.0625e−04
	CABC-R	044,000	1.7052	**3.1974e−16**	**1.700e−15**	**−8.8818e−16**	**2.6645e−15**
f_{06}	ABC-R	044,000	1.9104	1.0239e−06	1.06512e−06	1.8508e−07	7.1246e−06
	ABC	044,000	1.7836	3.1206e−06	4.0313e−06	4.9868e−07	2.2505e−05
	CABC-R	044,000	2.5148	**0**	**0**	**0**	**0**
f_{07}	ABC-R	044,000	2.8263	8.9658e−04	0.0032	2.7756e−14	0.0148
	ABC	044,000	2.7790	0.0022	0.0068	1.1551e−12	0.0433
	CABC-R	124,000	6.5936	**1.5705e−32**	**5.5294e−48**	**1.5705e−32**	**1.5705e−32**
f_{08}	ABC-R	124,000	6.7196	**1.5705e−32**	**5.5294e−48**	**1.5705e−32**	**1.5705e−32**
	ABC	124,000	6.7408	2.2788e−26	1.6000e−25	4.2812e−32	1.1315e−24
	CABC-R	124,000	6.3462	**1.3498e−32**	**1.1059e−47**	**1.3498e−32**	**1.3498e−32**
f_{09}	ABC-R	124,000	6.5647	3.3619e−32	1.4228e−31	1.3498e−32	1.0196e−30
	ABC	124,000	6.5096	1.4251e−26	4.4451e−26	7.0285e−30	2.1542e−25
	CABC-R	124,000	7.8508	**−78.3323**	**2.1296e−09**	**−78.3323**	**−78.3323**
f_{10}	ABC-R	124,000	8.1094	−78.3323	4.0434e−08	−78.3323	−78.3323
	ABC	124,000	7.9075	−78.3323	6.7726e−07	−78.3323	−78.3323

表 2　4 种优化算法的计算结果

Functions	Max_FFS	MSEP	JADE	RABC	CABC-R	
		Mean/ Std	Mean/ Std	Mean/ Std	Max_FFS	Mean/ Std
f_{01}	150,000	1.0e−4/1.3e−5	1.8e−60/8.4e−60	9.1148e−61/2.1063e−60	124,000	**5.0024e−94/2.1046e−93**
f_{02}	200,000	4.1e−4/2.1e−4	1.8e−25/8.8e−25	3.2100e−74/1.9714e−73	128,000	**8.3893e−82/4.8610e−81**
f_{03}	150,000	0/0	0/0	0/0	044,000	0/0
f_{04}	900,000	−12569.5/7.3.4e−4	−/−	−12569.5/7.2852e−12	124,000	**−12569.5/5.8799e−12**
f_{05}	500,000	2.5e−5/2.1e−5	0/0	0/0	044,000	0/0
f_{06}	150,000	1.7e−3/4.3e−4	4.4e−15/0	3.8247e−14/4.3659e−15	044,000	**3.1974e−16/1.700e−15**
f_{07}	200,000	8.5e−4/2.3e−3	0/0	2.0200e−04/1.4283e−03	044,000	**0/0**
f_{08}	150,000	7.5e−7/4.0e−7	1.6e−32/5.5e−48	**1.5704e−32/1.6588e−47**	124,000	1.5705e−32/5.5294e−48
f_{09}	150,000	1.2e−5/1.1e−5	1.4e−32/1.1e−47	**1.3497e−32/8.2941e−48**	124,000	1.3498e−32/1.1059e−47

结束语　(1)针对 ABC 存在过早收敛和易陷入局部最优以及优化精度不高的问题，提出一种 ABC-R 算法，使得 ABC 算法能够跳出局部最优，避免算法过早收敛。实验结果说明本方法是普遍有效的，在保留原算法简单易行、搜索能力强的基础上，更具目的性和灵活性，为进一步改善及应用打下了良好基础。(2)与文献[10-12]不同，本文利用混沌特性创建了新型的 COA，并将 ABC-R 算法与这种新型的 COA 有机结合来构建 CABC-R 算法，取长补短，先使用 ABC-R 算法获得全局解，然后采用 COA 进行局部搜索，这样既较好地获得了全局最优解，又大幅度提高了解的精度，当然可以将文献[10-12]等方法用于 CABC-R 算法中，以更进一步提高优化效果。由此可见，本文提出的 CABC-R 算法在解决优化问题上优势明显，是一种具有较大竞争力和潜力的智能优化方法。

参 考 文 献

[1] Karaboga D. An idea based on honey bee swarm for numerical optimization [R]. Technical Report-TR06. Erciyes University, Kayseri, Turkey, 2005

[2] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm [J]. Journal of Global Optimization, 2007, 39(3): 459-171

[3] Banharnsakun A, Achalakul T, Sirinaovakul B. The best-so-far selection in artificial bee colony algorithm [J]. Applied Soft Computing, 2011, 11: 2888-2901

[4] Li G Q, Niu P F, Xiao X J. Development and investigation of efficient artificial bee colony algorithm for numerical function opti-

mization [J]. Applied Soft Computing, 2012, 12: 320-332

[5] Kang F, Li J J, Ma Z Y. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions [J]. Information Sciences, 2011, 181(1): 3508-3531

[6] Gao W F, Liu S Y. Improved artificial bee colony algorithm for global optimization [J]. Information Processing Letters, 2011, 111(17): 871-882

[7] 毕晓君，王艳娇. 加速收敛的人工蜂群算法[J]. 系统工程与电子技术，2011，33(12)：2755-2761

[8] 张新明，孙印杰. 基于混沌优化的自适应中值滤波[J]. 电子技术应用，2007，33(9)：63-65

[9] 张新明，徐久成. 基于混沌理论和支持向量机的人脸识别方法[J]. 高技术通讯，2009，19(5)：494-500

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

[11] 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

[12] 李志勇，李玲玲，王翔，等. 基于 Memetic 框架的混沌人工蜂群算法[J]. 计算机应用研究，2012，29(11)：4045-4049

[13] 张新明，雷冠军，闫林，等. 一种新型快速的直接随机优化算法[J]. 吉林大学学报：理学版，2012，50(4)：750-756

[14] Dong H B, He J, Huang H K, et al. Evolutionary programming using a mixed mutation strategy [J]. Information Sciences, 2007, 177(1): 312-327

[15] Zhang J, Sanderson A C. JADE: adaptive differential evolution with optional external archive [J]. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 945-958

（上接第 69 页）

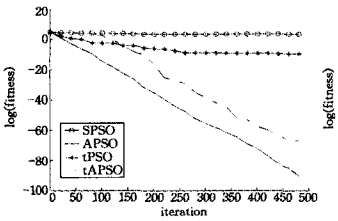

图 1　函数 f_1 寻优进化曲线

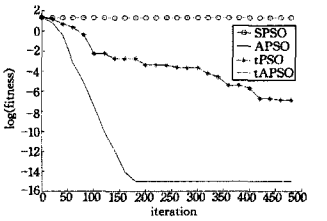

图 2　函数 f_2 寻优进化曲线

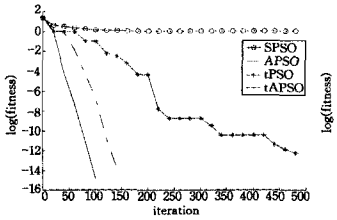

图 3　函数 f_3 寻优进化曲线

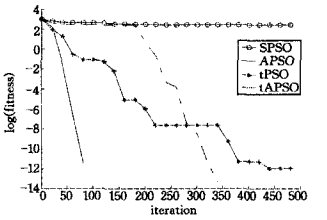

图 4　函数 f_4 寻优进化曲线

表 1　测试结果

函数	算法	最佳值	最差值	平均值
f1	SPSO	39.7475	1.0347e+004	753.0247
	tPSO	2.8097e−020	2.3204e−0011	9.7722e−012
	APSO	1.6456e−70	1.2172e−37	4.0573e−039
	tAPSO	1.1700e−101	1.4614e−089	5.4920e−091
f2	SPSO	20	20	20
	tPSO	2.8050e−011	1.3377e−005	1.3159e−006
	APSO	20	20	20
	tAPSO	8.8818e−016	8.8818e−016	8.8818e−016
f3	SPSO	0.9760	3.1590	1.1514
	tPSO	0	4.1296e−012	3.5261e−013
	APSO	0	0	0
	tAPSO	0	0	0
f4	SPSO	113.5420	370.7751	221.7465
	tPSO	8.8818e−015	1.7926e−009	9.0128e−011
	APSO	0	214.3522	24.1629
	tAPSO	0	0	0

从图 1 可以看出 tAPSO 的收敛精度在指数级别上优于其它 3 种算法，收敛速度也更快，能够收敛于很接近理论最优值的全局最优解，具有很好的优化性能。从图 2 可以看出，只有 tAPSO 算法在进化的前期突破了障碍物，越过了局部最优点，从而在进化的末期能够收敛到比较接近理论最优值的全局最优解，具备良好的收敛效果。从图 3 可以看出：SPSO 算法不能找到全局最优解；APSO 算法绕过了一部分的局部最优点，但是进化的过程比较坎坷，优化性能不够稳定；tPSO 算法和 tAPSO 算都能够最后收敛于比较接近理论最优解的全局最优解，并且收敛速度很快，获得了较好的收敛效果。从图 4 可以发现：SPSO 算法优化性能不好；APSO 算法虽然取到全局最优值，但优化效果不明显；tPSO 算法最终收敛到全局最优，且收敛的速度也较快，收敛精度也较高，具有良好的优化性能；tAPSO 算法具有比较突出的优势，算法的收敛速度、收敛精度和平均收敛值都要优于其它的算法。

从表 1 和图 1—图 4 中可以看出：对于每一个测试函数，本文提出的 tAPSO 算法在收敛速度和收敛精度上都明显优于其它 3 种算法。

结束语　本文在标准 PSO 算法的基础上进行改进，提出了带扰动因子的自适应粒子群优化算法。实验研究表明，提出的新算法具有较强的全局搜索能力、较高的收敛精度和较快的收敛速度。

参 考 文 献

[1] Kennedy J, Eberhart R. Particle swarm optimization[C]// IEEE International Conference on Neural Networks. Perth, 1995: 1942-1948

[2] 吴伟，李楠，郭茂耘. 粗糙集及 PSO 优化 BP 网络的故障诊断研究[J]. 计算机科学，2011，11(38)：200-203

[3] 苗启广，王明静，王宝树. 基于归一化互信息与模糊自适应 PSO 的图像自动配准方法[J]. 计算机科学，2008，6(35)：175-177

[4] 秦洪德，石丽丽. PSO 算法在油船双层结构优化设计中的应用研究[J]. 哈尔滨工程大学学报，2010，8(31)：1007-1011

[5] 肖奔贤. 基于改进 PSO 算法的过热汽温神经网络预测控制[J]. 控制理论与应用，2008，3(25)：569-573

[6] 刘军民，高岳林. 混沌优化算法[J]. 计算机应用，2008，28(2)：322-325

[7] 胡旺，李志蜀. 一种更简化而高效的粒子群优化算法[J]. 软件学报，2007，18(4)：861-868

[8] 赵志刚，常成. 简化的自适应粒子群优化算法[J]. 广西大学学报，2010，35(5)：793-798

[9] Shi Y, Eberhart C. A modified particle swarm optimizer[C]// IEEE International Conference Evolutionary Computation. Anchorage, Alaska, 1998, 5: 4-9

[10] 安晓会，高岳林. 混合变异算子的自适应粒子群优化算法[J]. 计算机应用，2008，28(6)：28-30