文章编号:1007-5321(2017)05-0061-06 DOI:10.13190/j.jbupt.2017-037

改进基于记忆的人工蜂群算法

杜振鑫[1,2], 刘广钟[2], 韩德志[2]
(1. 韩山师范学院 计算机与信息工程学院, 广东 潮州 521041; 2. 上海海事大学 信息工程学院, 上海 201306)

摘要: 基于记忆的人工蜂群算法(ABCM)通过记住成功使用的邻居和系数指导人工蜂群下一步的搜索,需消耗多次函数评价收敛到吸引子,且始终使用与上次相同的排斥系数,造成收敛速度不快、多样性不足,易陷入局部最优解. 提出一种改进 ABCM(IABCM),当使用吸引系数时,候选解只消耗一次函数评价收敛到吸引子,如果候选解好于当前解,则替换当前解,否则直接删除该记忆,这样可以利用尽量小的代价得到尽量大的收益. 当使用排斥系数时,该系数的数值部分重新随机生成,以增加多样性和随机性,有利于算法跳出局部最优解. 在 22 个不同类型函数上的实验表明,IABCM 在收敛速度和精度方面明显优于 ABCM.

关 键 词: 人工蜂群算法; 记忆; 收敛速度; 函数优化
中图分类号: TP18 **文献标志码**: A

An Improved Artificial Bee Colony Algorithm with Memory

DU Zhen-xin[1,2], LIU Guang-zhong[2], HAN De-zhi[2]
(1. School of Computer Information Engineering, Hanshan Normal University, Guangdong Chaozhou 521041, China;
2. College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China)

Abstract: Artificial bee colony algorithm with memory (ABCM) memorizes successful coefficients and neighbors to guide the further foraging of the artiflcial bees. ABCM consumes many function evaluations to converge to the attractors and use the same rejection coefficients as last time, which easily results in slow convergence, low population diversity and falling into the local minima. In the improved ABCM (IABCM), the candidates converge to the attractors consuming only one function evaluation, and the candidate will replace the current solution if the former is better than the latter. Otherwise, the memory will be deleted directly. By doing so, IABCM can get the most profit at a minimum cost. When the rejection coefficients are used, the numeric parts will be regenerated randomly to enhance the diversity and randomness, which is beneficial to help the algorithm to escape the local minima. Experiments on 22 functions with different characteristics demonstrate that the IABCM is significantly better than ABC and ABCM in terms of solution quality and convergence speed.

Key words: artiflcial bee colony; memory; convergence speed; function optimization

进化算法在解决图像处理、数据挖掘、特征选择等领域的复杂优化问题方面取得了显著的成绩,受到越来越多的关注. 主要的进化算法包括遗传算法[1]、粒子群算法[2]、蚁群算法[3]、差分进化算法[4]、人工蜂群算法[5](ABC, artificial bee colony)等. Karaboga 等[6-8]的实验结果表明 ABC 的性能好

收稿日期: 2017-03-27
基金项目: 国家自然科学基金项目(61672338,61373028)
作者简介: 杜振鑫(1976—), 男, 讲师, E-mail: duzhenxinmail@163.com.

于、至少相当于其他群体智能优化算法,加上 ABC 比较容易实施,因此得到了广泛的研究和应用[9-10]. 在 ABC 中,可以用先前的成功经验指导蜂群下一步的搜索. Kiran 等[11]通过增加记忆库记住精英解,使用精英解作为搜索公式中的邻居来增加 ABC 的开采性能. Gao 等[12]也提出了一种类似于记忆的机制来决定每个个体每次搜索使用的参数,统计上一代的所有个体搜索成功时使用的参数的均值决定下一次某个个体使用的参数. 以上几种改进都只是在解的水平,属于粗粒度记忆,没有针对某一个具体的个体. Li 等[13]提出了一种带记忆的 ABC(ABCM, artificial bee colony with memory),采用不同于上述文献的记忆机制,进一步细化了记忆机制,同时记住每个个体每一次成功搜索时使用的邻居和参数,取得了一定的优势. 但是,ABCM 仍然存在收敛速度不够快、精度不够高的缺点. 笔者在 ABCM 的基础上,进一步细化了记忆机制,在保持复杂度不变的情况下,改变 ABCM 中记忆的使用方式,在 22 个不同类型函数上的测试结果表明,改进算法明显提高了 ABCM 的性能.

1 ABC

在 ABC[5]中,雇佣蜂负责勘探食物源,并将食物源的信息通过舞蹈的形式通知观察蜂;观察蜂根据雇佣蜂带来的信息,按照一定的概率选择某个食物源继续进行开采. 食物源被选中的概率与食物源的质量成正比. 如果一个食物源的适应度连续 l 次未能成功更新,则需要放弃该位置. 这时,该食物源对应的雇佣蜂转变为侦察蜂,随机侦察(初始化)一个新的食物源代替该食物源. 在 ABC 中,雇佣蜂和观察蜂的数量都等于 N. 设待优化问题是求最小值,首先按照式(1)随机生成 N 个初始解.

$$x_i^j = x_{\min}^j + \mathrm{rand}(0,1)(x_{\max}^j - x_{\min}^j) \tag{1}$$

其中:$\mathrm{rand}(0,1)$是$[0,1]$之间的随机数;$i=1,2,\cdots,N$; $j=1,2,\cdots,D$,D 是决策变量的数量,也就是待优化问题的维数;$x_{\max}^j$和$x_{\min}^j$分别是第 j 维的上界和下界. 初始化之后,ABC 按以下 3 个阶段搜索问题的最优解.

1) 雇佣蜂阶段:每只雇佣蜂在相应的食物源 x_i 处,利用式(2)随机选择第 j 维更新,生成新的食物源 v_i.

$$v_i^j = x_i^j + \phi_i^j(x_k^j - x_i^j) \tag{2}$$

其中:ϕ_i^j是$[-1,1]$之间的随机数,$j \in \{1,2,\cdots,D\}$是随机选择的一个维,k是$\{1,2,\cdots,N\}$中随机选择的一个不同于 i 的食物源. 发现新的食物源 v_i之后,如果 v_i的函数值 $f(v_i)$小于 x_i的函数值 $f(x_i)$,就称搜索 x_i^j 成功,用 v_i 替换 x_i,否则 x_i 保持不变.

2) 观察蜂阶段:在所有雇佣蜂完成了勘探之后,观察蜂根据接收到的信息随机选择一个食物源继续开采. 观察蜂按照式(3)选择食物源.

$$p_i = a_i / \sum_{i=1}^{N} a_i \tag{3}$$

其中 a_i是食物源 i 的适应度值. 选择食物源之后,观察蜂用式(2)搜索新的候选解,如果搜索的新解的适应度更好,则替换掉原食物源,否则原食物源保持不变. 对于最小值优化问题,a_i由式(4)给出.

$$a_i = \begin{cases} 1/(1+o_i), & \text{若 } o_i \geqslant 0 \\ 1+|o_i|, & \text{其他} \end{cases} \tag{4}$$

其中 o_i 是食物源 i 的目标函数值.

3) 侦察蜂阶段:若每个食物源 i 使用式(2)搜索失败,则失败次数 $T_i = T_i + 1$. 在每一轮迭代中,只选择 T 值最大且超过 l 的食物源初始化. 即如果 $\max(T_i) > l$,则食物源 i 被抛弃,该食物源对应的雇佣蜂成为侦察蜂,该侦察蜂按照式(1)通过初始化被抛弃的食物源代替被抛弃的食物源 i.

2 ABCM

Giurfa[14]对蜜蜂等昆虫的记忆行为从神经科学的角度进行了深入的研究,结论表明真实的蜜蜂具有非凡的联想记忆能力,从而可以利用先前成功的经验指导后续搜索行为. 但是,这种记忆能力在基本 ABC 算法及其改进版本中没有得到体现,如式(2)所示,每一个雇佣蜂或者观察蜂都随机地选择一个邻居 k 开采. Li 等[13]通过引入记忆结构,模仿了蜜蜂的记忆能力,允许蜜蜂记住先前的成功搜索经验. 记忆结构随时更新,可以用来指导蜜蜂的后续搜索行为,使得搜索比 ABC 更加有效.

2.1 记忆结构

Li 等[13]提出的 ABCM 中,x_i^j的记忆用 M_i^j 表示,其中 $i=1,2,\cdots,N$ 表示食物源序号,$j=1,2,\cdots,D$ 表示变量的维数,整个记忆库表示为式(5). 开始

$$\boldsymbol{B} = \begin{bmatrix} M_1^1 & M_1^2 & \cdots & M_1^D \\ M_2^1 & M_2^2 & \cdots & M_2^D \\ \vdots & \vdots & \ddots & \vdots \\ M_N^1 & M_N^2 & \cdots & M_N^D \end{bmatrix} \tag{5}$$

时，每个记忆 M_i^j 为空，随着 ABC 算法的不断迭代，M_i^j 不断地根据雇佣蜂和观察蜂的成功搜索经验更新. 具体地，如果一个雇佣蜂或者观察蜂利用式(2)搜索 x_i^j 成功($o(v_i)<o(x_i)$)，相应的更新信息被视为成功的搜索经验并且保存在记忆 M_i^j 中. M_i^j 将记住式(2)的 2 个成分：所使用的邻居 k；所使用的系数 ϕ_i^j.

换句话说，如果式(2)用过的值 k 和 ϕ_i^j 能够成功地更新 x_i^j，则 k 和 ϕ_i^j 将被作为成功经验保存到记忆 M_i^j 中，以供下次再次使用. 表 1 给出了 x_i^j 的记忆结构 M_i^j. 其中，M 表示记忆容量，对于 x_i^j，其对应的记忆结构 M_i^j 包含成功搜索时的邻居 $k_i^j(m)$ 和成功搜索时的系数 $\phi_i^j(m)$，$m=1,2,\cdots,M$，$i=1,2,\cdots,N$，$j=1,2,\cdots,D$.

表 1　记忆的结构

记忆	成功邻居	成功系数
1	$k_i^j(1)$	$\phi_i^j(1)$
2	$k_i^j(2)$	$\phi_i^j(2)$
⋮	⋮	⋮
M	$k_i^j(M)$	$\phi_i^j(M)$

2.2　记忆的使用

在基本 ABC 中，当完成式(2)的时候，随机选择邻居 k 和系数 ϕ_i^j. 在 ABCM 中，当记忆未满的时候，与 ABC 一样随机选择邻居 k 和系数 ϕ_i^j；当记忆已满的时候，改用记忆中对应的邻居 k 和系数 ϕ_i^j 完成式(2)，也即此时蜜蜂继续使用以前成功的经验.

2.3　记忆的更新

开始的时候，每个食物源的每一维的记忆 M_i^j 为空. 在进化过程中，如果 x_i^j 使用式(2)搜索成功，则使用过的邻居 k 和系数 ϕ_i^j 被保存到 x_i^j 对应的记忆 M_i^j 中. 当记忆未满的时候，ABCM 与 ABC 一样随机选择式(2)的邻居 k 和系数 ϕ_i^j. 当记忆已满的时候，邻居 k 和系数 ϕ_i^j 从 M_i^j 的 M 条记忆中随机选择一条(表 1 中的任意一行)记忆. 如果使用该记忆搜索成功，则该条记忆继续保持，否则该条记忆删除.

3　改进算法

在 ABCM 中，当 x_i^j 的记忆 M_i^j 中的系数 ϕ_i^j 大于 0 时，笔者借用物理学中的概念，称为吸引系数，对应的邻居 x_k 称为吸引子；当 ϕ_i^j 小于 0 时称为排斥系数，对应的邻居 x_k 称为排斥子. ABCM 第 1 次使用吸引系数与吸引子的方式为

$$v_i^j=x_i^j(1)+\phi_i^j(x_k^j-x_i^j(1)) \tag{6}$$

其中：第 1 次使用该记忆时的 x_i^j 称为 $x_i^j(1)$. 假设第 1 次记忆使用成功，根据式(2)，应该用 v_i^j 代替 x_i^j，此时 $x_i^j=v_i^j=x_i^j(1)+\phi_i^j(x_k^j-x_i^j(1))$，$x_i^j$ 得到成功更新. 同时，根据 ABCM 的原理，该记忆被保持在 M_i^j 中，邻居 k 和系数 ϕ_i^j 保持不变，供下次继续使用. 第 2 次使用该记忆时：

$$\begin{aligned}v_i^j&=x_i^j(2)+\phi_i^j(x_k^j-x_i^j(2))=\\&\underbrace{x_i^j(1)+\phi_i^j(x_k^j-x_i^j(1))}_{x_i^j(2)}+\\&\phi_i^j[x_k^j-\underbrace{(x_i^j(1)+\phi_i^j(x_k^j-x_i^j(1)))}_{x_i^j(2)}]=\\&\underbrace{x_i^j(1)+2\phi_i^j(x_k^j-x_i^j(1))-(\phi_i^j)^2[x_k^j-x_i^j(1)]}_{x_i^j(3)}\end{aligned} \tag{7}$$

由于 $\phi_i^j<1$，所以第 3 项的系数 $(\phi_i^j)^2$ 相对于第 2 项的系数 $2\phi_i^j$ 是一个很小的数值，第 3 项可以忽略不计，式(7)可以简化为

$$v_i^j=x_i^j(1)+2\phi_i^j(x_k^j-x_i^j(1)) \tag{8}$$

同理，可以推得第 3 次使用该记忆的递推公式：

$$v_i^j=x_i^j(1)+3\phi_i^j(x_k^j-x_i^j(1)) \tag{9}$$

类似地，可得出第 n 次使用记忆的近似递推公式为

$$v_i^j=x_i^j(1)+n\phi_i^j(x_k^j-x_i^j(1)) \tag{10}$$

其中：$n\phi_i^j$ 是吸引系数，x_k^j 是 x_i^j 的吸引子. 当吸引系数 $n\phi_i^j>1$ 时无意义，因此 $n\phi_i^j$ 最多取 1. 也就是说，若 x_i^j 的记忆可以使用 n 次，将消耗 n 次函数评价，而其最终结果与只使用式(11)一次近似等价.

$$v_i^j=x_i^j(1)+1\times(x_k^j-x_i^j(1)) \tag{11}$$

式(11)相当于式(2)中 $\phi_i^j=1$ 的情况. 这意味着，如果在使用记忆的时候，令 $\phi_i^j=1$，直接使用式(11)，将会节省中间 $n-1$ 次函数评价，而最终的结果近似相同.

根据基本 ABC 的搜索式(2)，当系数 $\phi_i^j<0$ 时，式(2)右边的第 2 项 $\phi_i^j(x_k^j-x_i^j)$ 代表排斥项，此时的 x_k 与 $\phi_i^j>0$ 时的意义不同，不代表好于 x_i 的吸引子，相反，代表差于 x_i 的排斥子. 与吸引系数不同，排斥系数 ϕ_i^j 的数值部分 $|\phi_i^j|$ 没有具体的意义. 因此在改进 ABCM (IABCM, improved ABCM) 中，当 $\phi_i^j<0$ 时，采用式(12)重新生成新的排斥系数 ϕ_i^j.

$$\phi_i^j=\mathrm{rand}(-1,0) \tag{12}$$

其中 rand(-1,0) 代表[-1,0]之间的随机数. 与 ABCM 相比,IABCM 采用式(12),每次使用排斥系数 ϕ_i^j 的时候,只使用了 ϕ_i^j 的符号,而其数值部分 $|\phi_i^j|$ 重新随机生成,增加了多样性和随机性.

IABCM 算法的流程如下:

步骤 1 初始化种群,设置记忆库 $\boldsymbol{B}$ 为空,$g=1$.

//雇佣蜂阶段

步骤 2 当迭代代数 g 小于最大允许迭代次数时,执行以下步骤.

步骤 2.1 每个食物源 $i(1\leqslant i\leqslant N)$ 执行以下步骤:

1) 如果 M_i^j 未满,则执行搜索式(2),若搜索到更好的解,则将使用过的 k 与 ϕ_i^j 保存到 M_i^j.

2) 如果 M_i^j 已满,随机从记忆 M_i^j 中取出一条记忆,包含成功使用过的邻居 k 与对应的系数 ϕ_i^j. 对取出的系数 ϕ_i^j 做如下修正:若 $\phi_i^j>0$,则令 $\phi_i^j=1$;否则,令 ϕ_i^j 为[-1,0]之间的随机数. 利用取出的邻居 k 与修正的系数 ϕ_i^j 完成式(2). 如果未搜到更好的解或者系数 $\phi_i^j>0$,从记忆 M_i^j 中删除该次使用的邻居 k 及相应的系数 ϕ_i^j.

步骤 2.2 计算每个食物源 i 对应的选择概率 p_i.

//观察蜂阶段

步骤 2.3 对每只观察蜂 $t(1\leqslant t\leqslant N)$ 执行以下操作:

根据概率 p_i 轮盘赌选择一个食物源 i,执行步骤 2.1.

//侦察蜂阶段

步骤 2.4 每一轮迭代选择一个 T 值最大的食物源 i,若 $T_i>l$,利用式(1)初始化食物源 i,其相应的记忆置空.

步骤 2.5 $g=g+1$,若满足终止条件,转到步骤 3,否则返回步骤 2.

步骤 3 输出全局最优解,算法结束.

IABCM 与 ABCM 的区别仅在于使用记忆的方式不同. IABCM 使用记忆的过程如下:

1) 当 x_i^j 的记忆 M_i^j 未满的时候,每当使用式(2)搜索到一个更好的解 v_i,则将使用的邻居 k 和系数 ϕ_i^j 保存到记忆 M_i^j 中. (见步骤 2.1 的 1))

2) 当 x_i^j 的记忆 M_i^j 已满的时候,随机从 M_i^j 中取出一条记忆,该条记忆包含对应的邻居 k 和系数 ϕ_i^j. 如果 ϕ_i^j 大于 0(吸引系数),则 ϕ_i^j 赋值为 1 再使用. 如果 ϕ_i^j 小于 0(排斥系数),则 ϕ_i^j 仍然保持符号为负号,而其数值部分在[-1,0]之间随机生成. 根据邻居 k 和修改的系数 ϕ_i^j 完成式(2). 之后,如果 ϕ_i^j 是吸引系数,则不管搜索是否成功,都会删除该系数对应的整条记忆;如果 ϕ_i^j 是排斥系数,仅在搜索失败的时候删除整条记忆.

4 实验与分析

4.1 测试函数与参数设置

实验采用 Gao 等[12,15-16]等广泛使用的 22 个测试函数,这些函数的定义见文献[16],包含 22 个不同类型的函数,其中 $f_1\sim f_6$ 与 f_8 是连续的单峰函数,f_7 是不连续的函数,f_9 是含有噪声的函数,f_{10} 当 $D>3$ 时是多峰函数,$f_{11}\sim f_{22}$ 是多峰函数,测试结果如表 2 所示. 表 2 最后一行给出了 Wilcoxon[17] 非参数统计结果(显著性水平为 0.05),"+"、"="、"-"的含义分别表示 IABCM 好于、等于、差于被比较的算法. 设置参数 $N=50$,$D=30$,$M=2$,$l=D\times N$. maxFES 表示最大函数评价次数,作为算法的结束条件,设置 maxFES = 50 000. 实验在相同条件下重复 30 次.

4.2 实验结果与分析

根据表 2,IABCM 在 22 个函数上,有 17 个函数的求解精度明显好于 ABC 和 ABCM,1 个函数(f_7)上 3 种算法求解精度相同,显示 IABCM 具有较大的优势. 在单峰函数 $f_1\sim f_9$ 上,IABCM 的求解精度只在 f_9 上不如 ABCM,在其余 8 个函数上都取得了最好的结果. 这主要是因为当 ϕ_i^j 大于 0 时,直接取 $\phi_i^j=1$,节省了函数评价次数. 所有算法在 f_7 函数上都求得了理论最优解 0,这主要是因为 f_7 的理论最优解是一个区域,而不是一个点,因此非常容易求解[16]. 在多峰函数 $f_{11}\sim f_{22}$ 上,只有在 f_{11}、f_{12}、f_{14} 上稍差于其他算法,在其余 9 个函数上都取得了最好的结果. 这主要是因为 IABCM 在使用记忆的时候,当系数 ϕ_i^j 小于 0 时,只使用系数 ϕ_i^j 的符号(负号),而其数值部分在每次使用的时候都随机生成,增加了多样性和随机性,有利于算法在多峰函数中跳出局部最优解. 根据无免费午餐定理,一种策略带来优点的同时,总会带来一定的缺点,IABCM 由于直接取吸引系数 $\phi_i^j=1$,有利于算法快速向吸引子收敛,减小了算法的盲目搜索,但也会导致算法比较贪婪,

表 2　ABC、ABCM 与 IABCM 在 22 个测试函数上的实验结果($D=30$)

函数	ABC			ABCM			IABCM	
	平均值	标准差	显著性	平均值	标准差	显著性	平均值	标准差
f_1	5.07×10^{-5}	7.14×10^{-6}	+	5.07×10^{-5}	2.51×10^{-4}	+	9.30×10^{-8}	5.81×10^{-8}
f_2	1.54×10^{-2}	2.03×10^{-2}	+	2.63×10^{-2}	7.44×10^{-2}	+	2.91×10^{-4}	4.80×10^{-4}
f_3	3.47×10^{-6}	5.64×10^{-6}	+	2.07×10^{-6}	3.72×10^{-6}	+	1.38×10^{-8}	1.29×10^{-8}
f_4	1.34×10^{-13}	2.29×10^{-13}	+	7.99×10^{-15}	2.79×10^{-14}	+	1.03×10^{-17}	1.62×10^{17}
f_5	2.49×10^{-3}	9.74×10^{-4}	+	1.45×10^{-3}	1.20×10^{-3}	+	$1.09\times10^{-}4$	4.85×10^{-5}
f_6	3.63×10^{1}	4.13	+	3.83×10^{1}	5.16	+	3.53×10^{1}	4.83
f_7	0	0	=	0	0	=	0	0
f_8	4.58×10^{-9}	1.00×10^{-13}	+	4.58×10^{-9}	9.83×10^{-13}	+	4.56×10^{-9}	2.6×10^{-17}
f_9	1.86×10^{-1}	4.46×10^{-2}	+	1.82×10^{-1}	4.32×10^{-2}	−	1.85×10^{-1}	3.63×10^{-2}
f_{10}	6.06	7.31	+	5.97	7.08	+	3.09	2.88
f_{11}	3.73	1.86	−	4.72	1.59	+	3.88	1.72
f_{12}	6.70	1.79	+	5.38	1.46	−	5.75	1.30
f_{13}	9.47×10^{-3}	1.14×10^{-2}	+	1.66×10^{-2}	1.55×10^{-2}	+	2.25×10^{-5}	7.93×10^{-5}
f_{14}	7.62×10^{2}	1.66×10^{2}	−	7.49×10^{2}	1.25×10^{2}	−	8.03×10^{2}	1.48×10^{2}
f_{15}	4.96×10^{-2}	3.87×10^{-2}	+	6.09×10^{-2}	1.161×10^{-1}	+	6.09×10^{-3}	5.89×10^{-3}
f_{16}	2.61×10^{-6}	2.76×10^{-6}	+	2.49×10^{-6}	5.83×10^{-6}	+	8.99×10^{-9}	1.09×10^{-8}
f_{17}	6.97×10^{-6}	5.56×10^{-6}	+	3.34×10^{-6}	8.86×10^{-6}	+	2.06×10^{-7}	2.49×10^{-7}
f_{18}	2.35×10^{-2}	2.15×10^{-2}	+	1.96×10^{-2}	1.61×10^{-2}	+	1.10×10^{-2}	9.33×10^{-3}
f_{19}	9.39×10^{-5}	1.20×10^{-4}	+	1.94×10^{-5}	2.21×10^{-5}	+	1.83×10^{-6}	2.05×10^{-6}
f_{20}	1.87×10^{-1}	8.25×10^{-2}	+	8.26×10^{-2}	3.92×10^{-2}	+	7.27×10^{-2}	4.58×10^{-2}
f_{21}	−78.312	1.14×10^{-2}	+	−78.316	1.80×10^{-1}	+	−78.331	$1.197\,4\times10^{-4}$
f_{22}	−27.53	2.48×10^{-1}	+	−27.67	2.92×10^{-1}	+	−27.88	3.56×10^{-1}
+/=/−			19/1/2			18/1/3		

从而在少数多峰函数 f_{11}、f_{12}、f_{14} 上稍差于 ABCM，但是从总体上看，IABCM 在多数函数上仍然明显好于 ABCM. 图 1 给出了 ABC、ABCM 与 IABCM 的收敛曲线，从图中可以看出，IABCM 比 ABC、ABCM 有更快的收敛速度，且 IABCM 对比 ABCM 的优势明显大于 ABCM 对于 ABC 的优势.

5　结束语

ABC 算法因为简单高效、容易实施，从而获得了广泛应用. 如何在保持算法简单性的同时提高算法的性能，是改进算法的挑战. 笔者在 ABCM 的基础上，没有引入新的结构，也没有增加算法复杂度，仅仅改变了 ABCM 中记忆的使用方式，明显提高了 ABCM 的性能，改进算法依然保持了 ABCM 的简单、易实施的特点，容易作为通用框架推广到其他改进 ABC 算法.

参考文献：

[1] 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.

[2] Kennedy J, Eberhart R. Particle swarm optimization[C]// Proc of IEEE Int Conf on Neural Networks. Piscataway: IEEE, 1995: 1942-1948.

[3] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 1996, 26(1): 29-41.

[4] Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[5] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Kayseri: Engineering Faculty

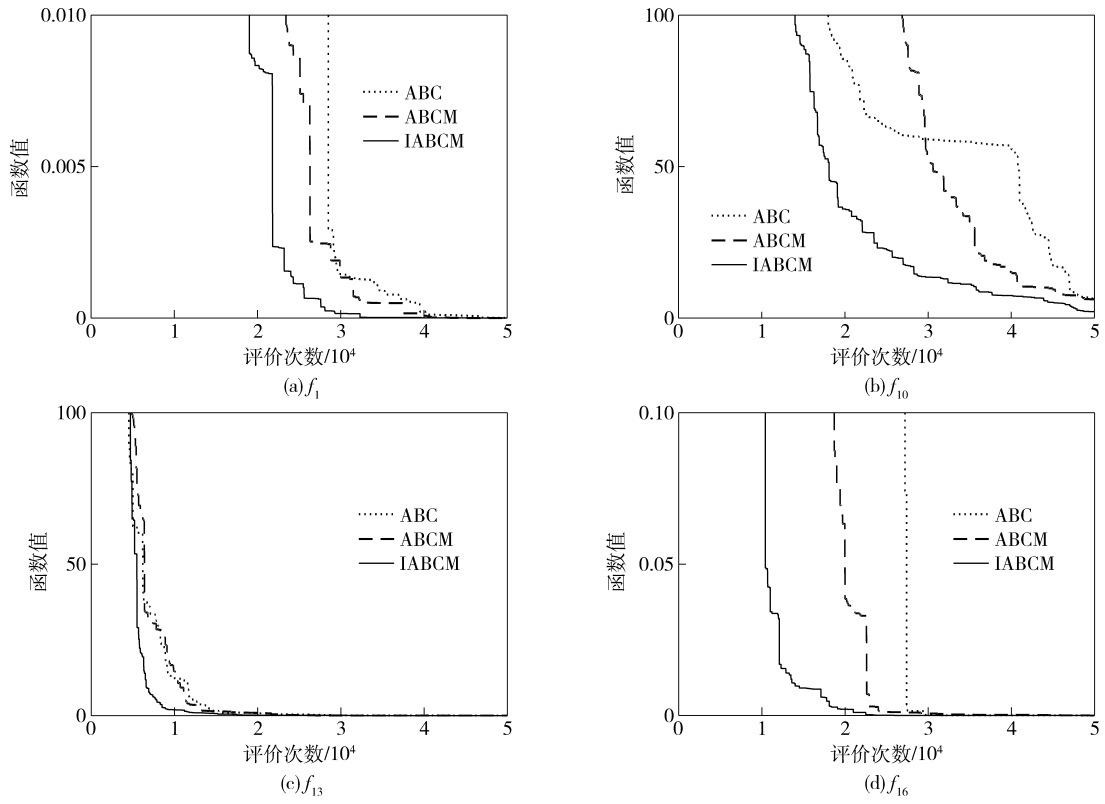

图1 3种算法在4个函数上的收敛曲线

Computer Engineering Department, Ereiyes University, 2005.

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

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

[8] Pholdee N, Bureerat S. Comparative performance of metaheuristic algorithm for mass minimission of trusses with dynamic constraints[J]. Advances in Engineering Software, 2014, 75(9): 1-13.

[9] Tasgetiren M F, Pan Q K, Suganthan P N, et al. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops[J]. Information Sciences, 2011, 181(16): 3459-3475.

[10] Lozano M, García-Martínez C, Rodríguez F J, et al. Optimizing network attacks by artificial bee colony[J]. Information Sciences, 2017, 377(1): 30-50.

[11] Kiran M S, Babalik A. Improved artificial bee colony algorithm for continuous optimization problems[J]. Journal of Computer and Communications, 2014, 2(04): 108-116.

[12] Gao Weifeng, Chan F T S, Huang Lingling, et al. Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood[J]. Information Sciences, 2015, 316(18): 180-200.

[13] Li Xianneng, Yang Guangfei. Artificial bee colony algorithm with memory[J]. Applied Soft Computing, 2016, 41(4): 362-372.

[14] Giurfa M, Zhang Shaowu, Jenett A, et al. The concepts of 'sameness' and 'difference' in an insect[J]. Nature, 2001, 410(6831): 930-933.

[15] Gao Weifeng, Liu Sanyang, Huang Lingling. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning[J]. IEEE Transactions on Cybernetics, 2013, 43(3): 1011-1024.

[16] Cui Laizhong, Li Genghui, Lin Qiuzhen, et al. A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation[J]. Information Sciences, 2016, 367(18): 1012-1044.

[17] Derrac J, García S, Molina D, et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms[J]. Swarm and Evolutionary Computation, 2011, 1(1): 3-18.