基于自适应进化策略的人工蜂群优化算法

张 强*，李盼池，王 梅

(东北石油大学计算机与信息技术学院 黑龙江 大庆 163318)

【摘要】提出一种自适应进化策略的人工蜂群优化算法来提高基本人工蜂群优化算法的性能。算法中每个引领蜂拥有4种进化策略，在迭代过程中通过计算每种进化策略的立即价值、未来价值和综合奖励来决定引领蜂个体的进化行为，并通过多策略进化概率变异方式来提升个体寻优速度或避免陷入局部最优解。典型高维复杂函数测试表明，该算法具有很好的收敛精度和计算速度。

关 键 词 人工蜂群算法； 进化策略； 极限学习机； 优化； 上限置信区间

中图分类号 TP312 **文献标志码** A doi:10.3969/j.issn.1001-0548.2019.04.013

Artificial Bee Colony Optimization Algorithm Based on Adaptive Evolution Strategy

ZHANG Qiang*, LI Pan-chi, and WANG Mei

(School of Computer and Information Technology, Northeast Petroleum University Daqing Heilongjiang 163318)

Abstract An artificial bee colony optimization algorithm based on adaptive evolution strategy is proposed to improve the performance of the artificial bee colony algorithm. Each leader individual has four evolutionary strategies in the algorithm. In the iteration process, the evolutionary behavior of the leader individual is determined by calculating the immediate value, the future value and the comprehensive reward of each evolutionary strategy. And then a multi-strategy evolutionary probability mutation method is proposed to improve the individual search speed or to avoid falling into the local optimal solution. Typical high-dimensional complex function tests show that the algorithm has good convergence accuracy and computational speed.

Key words artificial bee colony algorithm; evolution strategy; extreme learning machine; optimization; upper confidence bound

2005年，人工蜂群算法[1] (artificial bee colony algorithm, ABC)被提出用来解决函数优化问题。其原理是模拟蜜蜂的采蜜机制，通过不同分工的蜂群相互协作完成进化搜索工作，具有操作简单、设置参数少、收敛速度快等优点。从最初的解决非线性函数优化问题，逐步发展到神经网络优化[2]、特征选择[3]、生产调度[4]、多目标优化[5]和聚类分析[6]等领域。但是ABC也存在早熟收敛、陷入局部最优解的问题。为提高其性能，许多学者提出了改进版本，主要体现在种群多样性[7]、群体拓扑结构[8]、参数选择[9]、进化策略改进[10-13]、与其他智能算法融合方面[14-15]。文献[10]受粒子群算法启发，将全局最优解引入到个体进化公式中，并通过一个随机数控制全局最优解的影响程度。文献[11]将差分进化算法的DE/best/1变异策略引入到个体进化公式中，并构建了一个历史最优解存放池，随机从池里取出一个个体作为best来产生新个体。文献[12]则通过反向学习算法和混沌理论初始化种群，先按DE/best/1变异策略产生新个体，如果新个体不如父代个体，则按照ABC原有方式进行进化，虽然这些改进算法提升了寻优效率，但整个群体的当前最优解仅是算法迭代过程中某个个体自身所能找到的最优位置，只能代表所有个体的当前最好水平，而不能代表当前的整体进化水平。特别是在求解高维复杂的优化问题时，种群当前最优解很可能是一个局部极值，仅利用这一局部极值来指导整个群体的学习，就易陷入局部最优，较难达到目标求解效果。蜜蜂算法[16] (bees algorithm, BA)通过模拟自然界中蜜蜂的觅食行为的另一种蜂群算法，该算法采用基于邻域搜索的随机探索方式，但这种算法的参数设置较多。虽然这

收稿日期：2018－03－21；修回日期：2018－09－15
基金项目：国家自然科学基金(61702093)；黑龙江省自然科学基金(F2015020，F2015021)
作者简介：张强(1982－)，男，副教授，主要从事智能进化算法、神经网络方面的研究. E-mail: dqpi_zq@163.com

些算法对个体进化策略的改进方式各不相同，但本质上都是通过改变个体进化的方式来提升算法性能。本文提出基于自适应进化策略的人工蜂群优化算法，将个体赋予一定的自主性，在进化的过程中自主选择进化策略，以求在每次进化过程中都能使个体适应值得到改进。

1　人工蜂群算法基本原理

人工蜂群算法将蜂群分为3类：引领蜂、跟随蜂和侦察蜂，以最小化问题为例简要阐述其寻优过程。

1.1　引领蜂进化方式

将种群中的个体按照适应度值从小到大排序，取前 $N/2$ 组构成引领蜂种群，后 $N/2$ 组构成跟随蜂种群。对于当前代引领蜂种群中的个体 $x_{ij}(t)$，随机选择个体 $k\in[1,2,\cdots,N/2]$，$i\neq k$，按式(1)进行交叉搜索生成新个体 $v_{ij}(t)$：

$$v_{ij}(t)=x_{ij}(t)+\text{rand}(x_{ij}(t)-x_{kj}(t)) \tag{1}$$

式中，rand 为[−1,1]区间上的随机数；j 为优化空间维数范围内的随机整数。按式(2)选择较优的个体更新引领蜂种群：

$$x_i(t+1)=\begin{cases}v_i(t) & f(x_i(t))>f(v_i(t))\\ x_i(t) & f(x_i(t))\leqslant f(v_i(t))\end{cases} \tag{2}$$

1.2　跟随蜂进化方式

各跟随蜂依据引领蜂适应度的大小按选择概率式(3)在引领蜂种群中选择引领蜂 $x_k(t)$，$k\in[1,2,\cdots,N/2]$，并在其邻域内按式(1)进行新位置的搜索，产生新个体 $x_k(t)$，$k\in[N/2+1,\ N/2+2,\cdots,N]$，形成跟随蜂种群：

$$P_i=\frac{\text{fit}_i}{\sum_{i=1}^{N/2}\text{fit}_i} \tag{3}$$

1.3　侦察蜂进化方式

为了避免在迭代进化后期丧失种群的多样性，ABC将经过连续 limit 代进化适应度不变的个体转换成侦察蜂，并重新产生新个体。通过引领蜂种群、跟随蜂种群及侦察蜂的进化搜索，经过反复循环寻优直到算法迭代到最大迭代次数或种群的最优解达到预定误差精度时算法结束。

2　基于自适应进化策略的人工蜂群优化算法原理

ABC主要是通过引领蜂和跟随蜂的搜索行为实现的，由算法原理可知，搜索范围是随机确定的，算法不能控制搜索范围，这将导致搜索精度不高，算法的寻优速度也可能很慢。进化论研究表明：生物对环境有巨大的适应能力，环境的变化会引起生物的变化，生物会由此改进其行为，环境的多样化是生物多化的根本原因，而ABC算法个体所要面临的环境就是其他个体传递给它的知识环境，个体参照的知识不同进化行为也有所不同，即个体的进化行为在整个寻优过程中不应该是一成不变的。通过分析可知，ABC中个体在整个进化过程中能够获得知识的来源主要分为4个部分：1) 个体自身的知识；2) 当前最佳个体的知识；3) 所有个体的平均知识；4) 其他个体的知识。本文针对这4类知识提出4种策略变异行为来完成个体差分变异。

2.1　多策略变异进化方式

2.1.1　ABC原有的进化方式

这种进化方式有利于基本人工蜂群算法原有的寻优性能，即仍按式(1)对个体进行进化。

2.1.2　基于个体自身、当前最佳个体和和其他个体知识学习的进化方式

这种方式表明进化个体在决定个体寻优行为时充分考虑了自身、当前最佳个体和其他个体的影响，可以采用差分算法的DE/rand-to-best/1来表示，不同文献对于缩放因子F的选择具有一些固定的优选值，但是针对不同的优化问题，这个参数的设置对求解问题的结果具有一定的影响，本文不考虑F的影响，赋予个体较大的自主性，提出改进自适应权重进化方式：

$$v_{ij}=\omega x_{ij}+\text{rand}(\text{Gbest}_j-x_{ij})+\text{rand}(x_{r1,j}-x_{r2,j}) \tag{4}$$

式中，Gbest 为当前群体中的最佳个体。

2.1.3　基于群体平均知识学习的进化方式

社会中的个体通常具有趋同性，也就是说个体的行为容易受到群体共知所影响，可以用“群体位置质心”来代替群体平均知识，为简单计算采用所有个体位置的均值作为质心：

$$\begin{cases}p_j(t)=\sum_{i=1}^{\text{Num}}x_{ij}\Big/\text{Num}\\ v_{ij}(t)=\omega x_{ij}(t)+\text{rand}(p_j(t)-x_{ij}(t))\end{cases} \tag{5}$$

2.1.4　基于其他个体知识学习的进化方式

这种随机选取个体进行变异的方式等价于智能进化算法中的个体突变，并且这种突变方式与一些基于混沌理论进行变异方式相比较，更具有目的性和方向性。可以采用差分算法的DE/rand/1进化方式来表示：

$$v_{ij}(t)=x_{r1j}(t)+\text{rand}(x_{r2j}(t)-x_{r3j}(t)) \tag{6}$$

式中，$r1 \neq r2 \neq r3$ 为在群体中随机选择的3个个体。

式(4)和(5)引入了粒子群算法中的惯性权重 ω，用来描述父代个体对子代个体的影响。ω 值较大时表示全局寻优能力强，反之则局部寻优能力强。当求解问题空间较大时，为了在搜索速度和搜索精度之间达到平衡，常用的做法是使算法在前期具有较高的全局搜索能力，而在后期具有较高的局部搜索能力以提高收敛精度，采用线性递减的方式进行赋值：

$$\omega = \omega_{\max} - \frac{t}{T}(\omega_{\max} - \omega_{\min}) \tag{7}$$

2.2 计算各种策略进化行为立即价值

所谓立即价值是指采用某种策略进化行为后，所产生的适应度 $f(x_i^{\text{new}},t)$ 较上一次迭代计算的适应度 $f(x_i,t)$ 的提高程度，可以使用式(8)进行计算：

$$\text{value}_i(t) = \frac{f(x_i,t) - \min(f(x_i,t), f(x_i^{\text{new}},t))}{f(x_i,t)} \tag{8}$$

另外，若某一策略收益计算如果为零，说明该策略没有取得很好的寻优效果，这只是瞬时效益，不能代表此种策略不好，则还需计算其未来价值。

2.3 计算各种策略进化行为未来价值

个体在没有任何先验知识的前提下，在每次确定所要采取的何种策略行为时，可通过对已获得知识的利用和对新知识的探索协调来确定，以便更快捷地做出最优决策。信心上界算法[18-19](upper confidence bound, UCB)的上限置信区间由两部分之和构成，其中一部分是当前已有回报值，另一部分和回报值在单侧置信区间的大小有关，以保证所期望的回报能以极大的可能性落在回报范围之内。该算法主要利用UCB算法根据棋盘当前局面的立即价值和可选落子点的未来价值来计算上限置信区间最大的落子点作为当前局面的落子点，因此个体的进化策略选择可以借鉴围棋的落子点决策。该算法已被应用到计算机围棋程序中并取得了很好的效果。

令 $N_i^p(t)$ 为个体采用第 i 种策略在第 t 次迭代之前的成功次数，$M_i^p(t)$ 为个体采用第 i 种策略在第 t 次迭代之前总的执行次数，$N_i^g(t)$ 为所有个体采用第 i 种策略在第 t 次迭代之前的成功次数，$M_i^g(t)$ 为所有个体采用第 i 种策略在第 t 次迭代之前总的执行次数，分两种情况按照UCB公式来计算未来价值，一种是单个个体执行某一策略的成功率，还有就是该策略在整个种群的成功率：

$$\text{success}_i(t) = \alpha\left(\frac{N_i^p(t)}{M_i^p(t)} + \sqrt{\frac{C_0\log(t)}{N_i^p(t)}}\right) + (1-\alpha)\left(\frac{N_i^g(t)}{M_i^g(t)} + \sqrt{\frac{C_0\log(\text{Num}\times t)}{\sum_{i=1}^{\text{Num}} N_i^p}}\right) \tag{9}$$

式中，t 是当前迭代的次数；Num 是种群规模；α 和 C_0 是一个常数，α 的作用是平衡个体未来价值与全局未来价值在策略行为预测的重要性，C_0 的作用是平衡个体所获得知识的利用和探索需求，C_0 越大就越偏向于广度搜索，C_0 越小就越偏向于深度搜索，文中按照UCB算法的取值 $C_0 = 2$。

2.4 计算各种策略进化行为综合奖励

在考察一种策略是否有效，需考虑实施该策略的立即价值、估算的未来价值，同时还要考虑之前的历史经验 $P_i(t)$，即上一次迭代采用该策略的概率，故采用式(10)来计算某一个体采用某种策略所能获得的综合奖励：

$$\text{score}_i(t) = \text{value}_i(t) + \text{success}_i(t) + P_i(t) \tag{10}$$

式中，$1 \leqslant i \leqslant M$，$M$ 为多策略进化行为的个数。

2.5 更新各种策略进化行为选择概率

每个个体在分别计算完采用 M 种进化策略的综合奖励后，通过式(11)来更新在第 $t+1$ 次进化过程中采用某种进化策略的概率用以指示要采取哪种策略：

$$P_i(t+1) = P_{\min} + \frac{\text{score}_i(t)}{\sum_{i=1}^{N}\text{score}_i(t)}(1 - MP_{\min}) \tag{11}$$

为保证某种策略都存在概率不为0的情况，需要设置一个最小概率 $P_{\min} < 1/M$。

根据上述方法计算个体使用每个策略进化行为的立即价值、未来价值、综合奖励，并更新每种进化行为策略的选择概率，个体采用选择概率最大的进化行为来对个体进行进化，并利用贪心选择策略来更新个体，同时更新全局最优解。

2.6 多策略进化概率变异方式

在算法的进化过程中会存在个体一直采取某种策略进化行为，虽然每次进化都会对个体适应值有所提升，但提升幅度不大，这有可能造成个体寻优速度缓慢或造成陷入局部最优的情况，所以需要对这种情况进行判断来降低当前所采取策略进化行为的选择概率。文中利用适应值方差来反映个体的收敛程度，方差越小说明或以陷入局部最优或以发现全局最优解，故算法采用个体适应值的方差来判断：

$$\sigma^2 = \frac{\sum_{i=t-\text{count}}^{t}(f_i - \bar{f})^2}{\text{count}} \tag{12}$$

式中，t 代表当前的迭代次数；count 为第 t 次迭代前的适应值个数(本文取10)；f_i 为每次迭代的适应值；$\overline{f}$ 为 count 个适应值的均值。如果 σ^2 小于某一阈值，则将该策略行为的概率置为 $P_{\min}$。

3　数值实验及分析

3.1　函数极值优化

为验证本文所提算法的特点，选取几种与文中算法4种策略相关且具有代表性的算法进行比较。将其与ABC、差分进化算法[20](DE/rand/1、DE/best/1)、GABC[10]、NABC[11]、MABC[12]、BA[16]进行寻优性能对比。实验环境为：Windows 7操作系统，Intel 酷睿i5处理器，主频2.5 Hz，4 G内存，开发工具为 Matlab R2014a。针对8个函数极值求解问题，各算法的参数设置如下：种群个数 Num = 30；各自迭代1 000次，每个算法独立运行10次；AES_ABC 的 $\alpha = 0.5$；DE/rand/1、DE/best/1的 $F = 0.5, \mathrm{CR} = 0.3$ [20]；GABC、NABC 、MABC和BA参数分别按照文献[10]、[11]、[12]、[16]设置，各种蜂群算法的 limit = 100。测试函数如表1所示，分别对每种算法的不同维数(10,30,60,100)进行寻优，限于篇幅文中只列出30维和100维的优化结果如表2～9所示。

表1　优化测试函数

序号	测试函数	搜索空间	最优解
1	$f_1 = \sum_{i=1}^{n} x_i^2$	$[-100,100]^n$	0
2	$f_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10,10]^n$	0
3	$f_3 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)$	$[-100,100]^n$	0
4	$f_4 = \max(\mathrm{abs}(x_i))$	$[-100,100]^n$	0
5	$f_5 = \sum_{i=1}^{n} i x_i^4 + \mathrm{random}[0,1]$	$[-1.28,1.28]^n$	0
6	$f_6 = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] + 10n$	$[-5.12,5.12]^n$	0
7	$f_7 = 20 + \exp(1) - 20\exp\left[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right] - \exp\left[\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i) \right]$	$[-32,32]^n$	0
8	$f_8 = \frac{4}{4\,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left[\frac{x_i}{\sqrt{i}} \right] + 1$	$[-600,600]^n$	0

表2　函数F1不同维数测试结果

算法	维数	最优结果	最差结果	平均结果	方差
AES_ABC	30	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
	100	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
ABC	30	5.08×10^{-12}	2.35×10^{-11}	1.40×10^{-11}	7.37×10^{-12}
	100	5.37×10^{1}	1.14×10^{2}	8.03×10^{1}	2.21×10^{1}
DE/rand/1	30	8.43×10^{-12}	1.57×10^{-11}	1.12×10^{-11}	2.80×10^{-12}
	100	1.01×10^{2}	1.30×10^{2}	1.13×10^{2}	1.05×10^{1}
DE/best/1	30	1.46×10^{-34}	2.61×10^{-33}	7.93×10^{-34}	1.04×10^{-33}
	100	9.28×10^{-7}	1.15×10^{-5}	4.73×10^{-6}	4.08×10^{-6}
BA	30	1.91×10^{4}	2.59×10^{4}	2.32×10^{4}	3.47×10^{3}
	100	1.85×10^{5}	2.01×10^{5}	1.95×10^{5}	6.66×10^{3}
GABC	30	7.78×10^{-20}	4.29×10^{-19}	2.16×10^{-19}	1.55×10^{-19}
	100	7.28×10^{-2}	4.22×10^{-1}	2.56×10^{-1}	1.40×10^{-1}
NABC	30	5.08×10^{-28}	2.18×10^{-27}	1.44×10^{-27}	7.34×10^{-28}
	100	5.44×10^{-4}	1.70×10^{-3}	1.23×10^{-3}	4.25×10^{-4}
MABC	30	4.54×10^{-43}	8.85×10^{-42}	6.03×10^{-42}	3.32×10^{-42}
	100	3.31×10^{-7}	5.84×10^{-7}	4.65×10^{-7}	1.18×10^{-7}

表3　函数F2不同维数测试结果

算法	维数	最优结果	最差结果	平均结果	方差
AES_ABC	30	1.47×10^{-306}	4.65×10^{-259}	9.29×10^{-260}	0.00×10^{0}
	100	1.03×10^{-300}	1.68×10^{-245}	3.36×10^{-246}	0.00×10^{0}
ABC	30	1.55×10^{-6}	3.19×10^{-6}	2.13×10^{-6}	6.62×10^{-7}
	100	3.30×10^{0}	5.60×10^{0}	4.37×10^{0}	8.29×10^{-1}
DE/rand/1	30	1.18×10^{-7}	1.70×10^{-7}	1.41×10^{-7}	1.96×10^{-8}
	100	1.41×10^{1}	1.82×10^{1}	1.64×10^{1}	1.84×10^{0}
DE/best/1	30	1.15×10^{-19}	8.59×10^{-19}	3.63×10^{-19}	2.87×10^{-19}
	100	3.78×10^{-5}	7.45×10^{-4}	2.76×10^{-4}	2.82×10^{-4}
BA	30	5.79×10^{1}	8.68×10^{1}	7.32×10^{1}	1.08×10^{1}
	100	3.86×10^{2}	4.16×10^{2}	4.00×10^{2}	1.06×10^{1}
GABC	30	3.05×10^{-10}	6.15×10^{-10}	4.63×10^{-10}	1.23×10^{-10}
	100	4.03×10^{-1}	4.67×10^{-1}	4.31×10^{-1}	2.39×10^{-2}
NABC	30	5.80×10^{-15}	1.24×10^{-14}	9.09×10^{-15}	2.42×10^{-15}
	100	1.37×10^{-2}	1.81×10^{-2}	1.62×10^{-2}	1.85×10^{-3}
MABC	30	4.09×10^{-22}	1.21×10^{-21}	8.40×10^{-22}	3.45×10^{-22}
	100	2.47×10^{-4}	2.73×10^{-4}	2.60×10^{-4}	9.79×10^{-6}

表4　函数F3不同维数测试结果

算法	维数	最优结果	最差结果	平均结果	方差
AES_ABC	30	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
	100	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
ABC	30	1.30×10^{4}	2.10×10^{4}	1.63×10^{4}	2.92×10^{3}
	100	1.88×10^{5}	2.24×10^{5}	2.06×10^{5}	1.42×10^{4}
DE/rand/1	30	2.17×10^{4}	3.31×10^{4}	2.76×10^{4}	4.49×10^{3}
	100	5.77×10^{5}	7.19×10^{5}	6.61×10^{5}	5.99×10^{4}
DE/best/1	30	8.75×10^{2}	4.79×10^{3}	2.89×10^{3}	1.73×10^{3}
	100	2.37×10^{5}	3.68×10^{5}	2.90×10^{5}	4.88×10^{4}
BA	30	1.66×10^{4}	3.11×10^{4}	2.37×10^{4}	5.43×10^{3}
	100	2.82×10^{5}	4.36×10^{5}	3.61×10^{5}	6.71×10^{4}
GABC	30	1.21×10^{4}	1.99×10^{4}	1.65×10^{4}	3.07×10^{3}
	100	1.61×10^{5}	2.47×10^{5}	2.01×10^{5}	3.26×10^{4}
NABC	30	1.68×10^{4}	2.05×10^{4}	1.86×10^{4}	1.73×10^{3}
	100	2.08×10^{5}	2.60×10^{5}	2.34×10^{5}	2.49×10^{4}
MABC	30	1.17×10^{4}	2.71×10^{4}	2.07×10^{4}	5.64×10^{3}
	100	2.65×10^{5}	3.40×10^{5}	2.90×10^{5}	3.42×10^{4}

表5 函数F4不同维数测试结果

算法	维数	最优结果	最差结果	平均结果	方差
AES_ABC	30	1.15×10^{-291}	1.14×10^{-234}	2.28×10^{-235}	0.00×10^{0}
	100	6.90×10^{-289}	8.40×10^{-251}	1.68×10^{-251}	0.00×10^{0}
ABC	30	2.99×10^{1}	5.46×10^{1}	4.44×10^{1}	9.01×10^{0}
	100	8.89×10^{1}	9.09×10^{1}	9.01×10^{1}	8.58×10^{-1}
DE/rand/1	30	2.59×10^{-1}	3.70×10^{-1}	3.00×10^{-1}	4.34×10^{-2}
	100	7.41×10^{1}	8.69×10^{1}	8.19×10^{1}	4.77×10^{0}
DE/best/1	30	1.15×10^{1}	2.93×10^{1}	1.77×10^{1}	7.17×10^{0}
	100	5.19×10^{1}	5.99×10^{1}	5.39×10^{1}	3.39×10^{0}
BA	30	5.15×10^{1}	5.92×10^{1}	5.64×10^{1}	3.27×10^{0}
	100	8.32×10^{1}	8.69×10^{1}	8.47×10^{1}	1.41×10^{0}
GABC	30	2.09×10^{1}	3.03×10^{1}	2.45×10^{1}	3.71×10^{0}
	100	8.37×10^{1}	8.95×10^{1}	8.69×10^{1}	2.38×10^{0}
NABC	30	1.51×10^{1}	1.97×10^{1}	1.82×10^{1}	1.83×10^{0}
	100	8.58×10^{1}	9.03×10^{1}	8.74×10^{1}	1.81×10^{0}
MABC	30	2.01×10^{0}	2.27×10^{0}	2.12×10^{0}	1.05×10^{-1}
	100	7.69×10^{1}	8.25×10^{1}	7.99×10^{1}	2.26×10^{0}

表6 函数F5不同维数测试结果

算法	维数	最优结果	最差结果	平均结果	方差
AES_ABC	30	1.47×10^{-5}	1.46×10^{-4}	9.16×10^{-5}	6.41×10^{-5}
	100	8.55×10^{-5}	2.87×10^{-4}	1.67×10^{-4}	7.79×10^{-5}
ABC	30	2.31×10^{-1}	3.92×10^{-1}	2.80×10^{-1}	6.47×10^{-2}
	100	2.22×10^{0}	3.19×10^{0}	2.98×10^{0}	4.23×10^{-1}
DE/rand/1	30	9.99×10^{-2}	1.55×10^{-1}	1.28×10^{-1}	2.09×10^{-2}
	100	9.47×10^{-1}	1.23×10^{0}	1.12×10^{0}	1.29×10^{-1}
DE/best/1	30	2.92×10^{-2}	3.86×10^{-2}	3.32×10^{-2}	3.97×10^{-3}
	100	1.88×10^{-1}	2.88×10^{-1}	2.59×10^{-1}	4.17×10^{-2}
BA	30	1.10×10^{1}	1.67×10^{1}	1.41×10^{1}	2.11×10^{0}
	100	8.40×10^{2}	1.08×10^{3}	9.44×10^{2}	9.13×10^{1}
GABC	30	8.06×10^{-2}	1.26×10^{-1}	9.84×10^{-2}	1.67×10^{-2}
	100	1.38×10^{0}	1.94×10^{0}	1.72×10^{0}	2.14×10^{-1}
NABC	30	4.65×10^{-2}	7.67×10^{-2}	6.58×10^{-2}	1.14×10^{-2}
	100	8.65×10^{-1}	1.32×10	1.07×10^{0}	1.78×10^{-1}
MABC	30	2.00×10^{-2}	3.47×10^{-2}	2.63×10^{-2}	6.24×10^{-3}
	100	4.42×10^{-1}	5.56×10^{-1}	5.12×10^{-1}	4.85×10^{-2}

表7 函数F6不同维数测试结果

算法	维数	最优结果	最差结果	平均结果	方差
AES_ABC	30	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
	100	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
ABC	30	1.00×10^{0}	2.26×10^{0}	1.51×10^{0}	5.73×10^{-1}
	100	9.37×10^{1}	1.11×10^{2}	1.00×10^{2}	7.91×10^{0}
DE/rand/1	30	7.19×10^{1}	9.94×10^{1}	8.71×10^{1}	1.01×10^{1}
	100	7.74×10^{2}	8.56×10^{2}	8.12×10^{2}	3.02×10^{1}
DE/best/1	30	7.96×10^{0}	1.79×10^{1}	1.27×10^{1}	3.88×10^{0}
	100	2.11×10^{2}	5.42×10^{2}	3.84×10^{2}	1.37×10^{2}
BA	30	1.15×10^{2}	1.48×10^{2}	1.34×10^{2}	1.35×10^{1}
	100	8.62×10^{2}	9.49×10^{2}	9.13×10^{2}	3.54×10^{1}
GABC	30	4.43×10^{-8}	1.90×10^{-6}	5.49×10^{-7}	7.80×10^{-7}
	100	5.21×10^{1}	6.18×10^{1}	5.77×10^{1}	3.95×10^{0}
NABC	30	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
	100	3.69×10^{-1}	3.46×10^{0}	1.72×10^{0}	1.50×10^{0}
MABC	30	0.00×10^{0}	5.68×10^{-14}	1.14×10^{-14}	2.54×10^{-14}
	100	1.13×10^{-4}	1.99×10^{0}	1.39×10^{0}	8.90×10^{-1}

表8 函数F7不同维数测试结果

算法	维数	最优结果	最差结果	平均结果	方差
AES_ABC	30	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	0.00×10^{0}
	100	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	0.00×10^{0}
ABC	30	4.20×10^{-6}	1.52×10^{-4}	5.84×10^{-5}	5.69×10^{-5}
	100	3.77×10^{0}	5.19×10^{0}	4.36×10^{0}	6.02×10^{-1}
DE/rand/1	30	7.28×10^{-7}	9.77×10^{-7}	8.90×10^{-7}	9.63×10^{-8}
	100	3.00×10^{0}	3.77×10^{0}	3.37×10^{0}	2.91×10^{-1}
DE/best/1	30	7.99×10^{-15}	1.51×10^{-14}	1.23×10^{-14}	3.89×10^{-15}
	100	1.21×10^{0}	2.70×10^{0}	1.94×10^{0}	6.00×10^{-1}
BA	30	1.79×10^{1}	1.83×10^{1}	1.80×10^{1}	1.48×10^{-1}
	100	1.92×10^{1}	1.93×10^{1}	1.92×10^{1}	5.35×10^{-2}
GABC	30	2.04×10^{-10}	4.89×10^{-10}	4.02×10^{-10}	1.15×10^{-10}
	100	7.79×10^{-1}	9.38×10^{-1}	8.57×10^{-1}	7.12×10^{-2}
NABC	30	7.55×10^{-14}	1.50×10^{-13}	9.97×10^{-14}	3.00×10^{-14}
	100	1.36×10^{-2}	1.91×10^{-2}	1.67×10^{-2}	2.26×10^{-3}
MABC	30	2.93×10^{-14}	3.29×10^{-14}	3.22×10^{-14}	1.59×10^{-15}
	100	7.42×10^{-4}	1.89×10^{-3}	1.28×10^{-3}	4.35×10^{-4}

表9 函数F8不同维数测试结果

算法	维数	最优结果	最差结果	平均结果	方差
AES_ABC	30	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
	100	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
ABC	30	1.35×10^{-8}	6.11×10^{-5}	1.37×10^{-5}	2.65×10^{-5}
	100	1.47×10^{0}	1.98×10^{0}	1.67×10^{0}	2.03×10^{-1}
DE/rand/1	30	2.28×10^{-11}	8.85×10^{-10}	3.01×10^{-10}	3.45×10^{-10}
	100	1.90×10^{0}	2.31×10^{0}	2.01×10^{0}	1.70×10^{-1}
DE/best/1	30	0.00×10^{0}	2.21×10^{-2}	9.85×10^{-3}	7.97×10^{-3}
	100	1.02×10^{-7}	3.19×10^{-2}	6.38×10^{-3}	1.43×10^{-2}
BA	30	1.49×10^{2}	2.53×10^{2}	2.05×10^{2}	4.19×10^{1}
	100	1.75×10^{3}	1.99×10^{3}	1.88×10^{3}	9.70×10^{1}
GABC	30	3.79×10^{-11}	2.42×10^{-6}	4.84×10^{-7}	1.08×10^{-6}
	100	1.41×10^{-1}	4.64×10^{-1}	2.90×10^{-1}	1.26×10^{-1}
NABC	30	5.55×10^{-16}	3.97×10^{-3}	7.94×10^{-4}	1.77×10^{-3}
	100	3.05×10^{-3}	3.93×10^{-2}	1.40×10^{-2}	1.52×10^{-2}
MABC	30	2.22×10^{-16}	3.17×10^{-4}	6.35×10^{-5}	1.42×10^{-4}
	100	2.43×10^{-5}	6.70×10^{-5}	4.79×10^{-5}	1.98×10^{-5}

设置收敛精度为1×10^{-10}，迭代次数设置为300，每个算法独立运行10次取平均值，对比本文算法在100维函数的收敛平均迭代次数和运行时间如表10和表11所示。

表10 函数F1-F4收敛迭代次数与运行时间对比

算法	F1		F2		F3		F4	
	迭代次数	运行时间/s	迭代次数	运行时间/s	迭代次数	运行时间/s	迭代次数	运行时间/s
AES_ABC	130	0.666	148	0.652	133	2.624	143	0.672
ABC	300	0.761	300	0.677	300	4.884	300	0.758
DE/rand/1	300	1.928	300	1.883	300	10.820	300	2.136
DE/best/1	300	1.966	300	1.944	300	11.110	300	2.062
BA	300	9.650	300	9.860	300	13.748	300	9.260
GABC	300	0.703	300	0.671	300	5.077	300	0.783
NABC	300	0.814	300	0.880	300	5.005	300	0.918
MABC	300	1.020	300	0.959	300	8.235	300	1.260

表11　函数F5-F8收敛迭代次数与运行时间对比

算法	F5		F6		F7		F8	
	迭代次数	运行时间/s	迭代次数	运行时间/s	迭代次数	运行时间/s	迭代次数	运行时间/s
AES_ABC	300	2.522	187	0.957	131	0.970	130	0.990
ABC	300	1.144	300	1.363	300	1.056	300	1.064
DE/rand/1	300	3.042	300	2.528	300	2.552	300	2.788
DE/best/1	300	3.060	300	2.432	300	2.467	300	2.664
BA	300	10.610	300	10.130	300	10.280	300	11.940
GABC	300	1.179	300	1.293	300	1.142	300	1.144
NABC	300	1.221	300	1.391	300	1.226	300	1.414
MABC	300	1.898	300	1.425	300	1.526	300	1.630

从表2～表9可以看出，在增加维数的情况下寻优性能相对稳定，都可以获得较为理想的解。从表10和表11可知，在高维函数优化中，AES_ABC可以用较少的迭代次数获取较好的寻优结果，而其他算法在指定迭代次数内都没有收敛到指定精度。这主要因为AES_ABC寻优个体在迭代过程中，可以通过计算立即价值、估算的未来价值和历史经验来判断如何从个体自身、当前最佳个体、整个群体或其他个体获取知识，并组合这些知识用来指导策略的选择。可以让算法在自身感知的情况下自主选用较优的进化行为，力争个体每次进化都能改进自身适应度。同时多策略进化概率变异算法可以避免某种选用策略造成寻优性能缓慢的情况，使个体通过不断选择进化策略来避免陷入局部最优，有利于获得全局最优解，进而解决了算法在求解某些复杂函数时收敛速度慢、容易陷入局部最优的缺点。

3.2　极限学习机网络权值优化

极限学习机[21](extreme learning machine, ELM)是一种针对单隐藏层前馈神经网络的新算法，具有学习速度快且泛化性能好的优点。虽然极限学习机在大部分情况下可以获得良好的性能，但在一些实际应用中需要大量的隐层节点才能达到预期的效果，而隐层节点过多会增加网络复杂度，容易产生过拟合现象，并且造成极限学习机的泛化能力降低。传统ELM算法的输入权值及隐层偏差都是随机赋值的，因此无法保证这些参数都是最优的，并且广义逆矩阵的计算量会随着矩阵规模的变大而增加，其计算复杂度要多于本文的综合奖励计算量。为此，本文采用所提算法对输入权值及隐层偏差进行优化建立分类模型，并与4种蜂群进化算法进行对比，选择6种UCI数据集进行实验，算法运行参数设置参照3.1，迭代次数设置为300，EML的隐层节点数设置为20，每次取数据集的70%作为训练数据集，剩下的30%作为测试数据集，分别运行10次，取10次运行结果的平均值作为算法的优化结果。每个数据集及分类性能对比如表12和表13所示。

表12　6种UCI数据集

UCI数据	样本总数	特征维数	类别数量
Wine	178	12	3
Heart	270	12	2
Kr_V_kp	3 196	36	2
Pendigits	10 992	16	10
Tic_tac_toc	958	9	2
Balance	625	4	3

表13　各类方法分类性能对比

UCI数据	算法	训练数据识别率/%	测试数据识别率/%
Wine	ELM	64.23	61.82
	ABC_ ELM	95.12	89.09
	AES_ ABC _ ELM	100.00	98.18
	GABC_ ELM	98.37	89.09
	NABC_ ELM	95.93	89.12
	MABC_ ELM	95.12	90.91
Heart	ELM	69.31	65.14
	ABC_ ELM	74.60	66.08
	AES_ ABC _ ELM	86.24	85.19
	GABC_ ELM	77.78	66.67
	NABC_ ELM	78.84	67.90
	MABC_ ELM	79.89	68.35
Kr_V_kp	ELM	74.28	71.83
	ABC_ ELM	83.63	83.75
	AES_ ABC _ ELM	94.45	94.06
	GABC_ ELM	87.25	84.35
	NABC_ ELM	88.28	86.25
	MABC_ ELM	88.14	85.42
Pendigits	ELM	74.48	66.32
	ABC_ ELM	85.71	85.34
	AES_ ABC _ ELM	91.46	90.91
	GABC_ ELM	88.62	88.70
	NABC_ ELM	87.52	87.49
	MABC_ ELM	87.39	86.37
Tic_tac_toc	ELM	73.13	67.71
	ABC_ ELM	79.25	72.57
	AES_ ABC _ ELM	90.54	85.39
	GABC_ ELM	81.34	73.26
	NABC_ ELM	84.03	78.13
	MABC_ ELM	83.54	73.92
Balance	ELM	88.60	82.83
	ABC_ ELM	92.43	88.36
	AES_ ABC _ ELM	96.56	93.17
	GABC_ ELM	93.12	90.48
	NABC_ ELM	92.89	89.95
	MABC_ ELM	94.27	90.89

从本文算法与4种蜂群算法的分类性能对比结果可知，AES_ABC_ ELM可以使ELM获得很好的训练数据集识别率，尤其对数据样本数和特征维数较多的数据集Kr_V_kp、Pendigits、Tic_tac_toc。根据ELM建模原理，对于这3类UCI数据集需要优化的输入权值及隐含层偏差相对于其他3种Wine、Heart 和Balance要多，广义逆矩阵的计算量也较大，这也说明本文算法对于高维数据寻优具有较好的效果。

4 结 束 语

本文提出基于自适应进化策略的人工蜂群优化算法，使个体在进化过程中不断利用不同的知识源来更新自身位置，采用贪心算法的方式以期在每次进化都能获得更好的收益。基于高维测试函数的测试结果表明，所提算法在整体上具有较好的全局寻优能力且收敛速度较快，可以在较少的迭代次数内获得较为满意的寻优结果，适合于求解一些适应度计算复杂和耗时的优化问题。

参 考 文 献

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

[2] PERUMAL S S, VISHWANATH N, JER LANG H, et al. An effective content based medical image retrieval by using ABC based artificial neural network(ANN)[J]. Current Medical Imaging Reviews, 2017, 12(999): 1.

[3] SHUNMUGAPRIYA P, KANMANI S. A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid)[J]. Swarm & Evolutionary Computation, 2017, 36: 27-36.

[4] ZHANG R, SONG S, WU C. A hybrid artificial bee colony algorithm for the job shop scheduling problem[J]. International Journal of Production Economics, 2013, 141(1): 167-178.

[5] LUO J, LIU Q, YANG Y, et al. An artificial bee colony algorithm for multi-objective optimisation[J]. Applied Soft Computing, 2017, 50: 235-251.

[6] BHARTI K K, SINGH P K. Chaotic Artificial bee colony for text clustering[J]. Soft Computing, 2016, 20(3): 1113-1126.

[7] 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.
LUO Jun, LI Yan. Artificial bee colony algorithm withchaotic-search strategy[J]. Control and Decision, 2010, 25(12): 1913-1916.

[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.
BAO Li, ZENG Jian-chao. A bi-group differential artificial bee colony algorithm[J]. Control Theory and plications, 2011, 28(2): 266-272.

[9] AKAY B, KARABOGA D. Parameter tuning for the artificial bee colony algorithm[M]//Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems. [S.l.]: Springer, 2009: 608-619.

[10] ZHU G, KWONG S. Gbest-guided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics & Computation, 2010, 217(7): 3166-3173.

[11] XU Y, FAN P, YUAN L. A simple and efficient artificial bee colony algorithm[J]. Mathematical Problems in Engineering, 2013, 1: 1-9.

[12] GAO W F, LIU S Y. A modified artificial bee colony algorithm[J]. Computers & Operations Research, 2012, 39(3): 687-697.

[13] GAO W F, HUANG L L, WANG J, et al. Enhanced artificial bee colony algorithm through differential evolution[J]. Applied Soft Computing, 2016, 48: 137-150.

[14] XIANG W, MA S, AN M. HABCDE: A hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution[J]. Applied Mathematics & Computation, 2014, 238(238): 370-386.

[15] LI Z, WANG W, YAN Y, et al. PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems[J]. Expert Systems with Applications, 2015, 42(22): 8881-8895.

[16] PHAM D T, GHANBARZADEH A, KOÇ E, et al. The bees algorithm-a novel tool for complex optimisation problems[J]. Intelligent Production Machines & Systems, 2006, 1: 454-459.

[17] 董文永, 康岚兰, 刘宇航, 等. 带自适应精英扰动及惯性权重的反向粒子群优化算法[J]. 通信学报, 2016, 37(12): 1-10.
DONG Wen-yong, KANG Lan-lan, LIU Yu-hang, et al. An opposition-based particle swarm optimization with adaptive elite mutation and nonlinear inertia weight[J]. Journal on Communications, 2016, 37(12): 1-10.

[18] AUER P, CESA-BIANCHI N, FISCHER P. Finite-time analysis of the multiarmed bandit problem[J]. Mach Learn, 2002, 47(2): 235-256.

[19] STURTEVANT N R. An analysis of UCT in multi-player games[J]. Computers and Games, 2008, 31(4): 37-49.

[20] PRICE K, STORN R, LAMPINEN J. Differential evolution: a practical approach to global optimization [M]. Berlin, Germany: Springer-Verlag, 2005.

[21] HUANG G B, ZHU Q Y., SIEW C K. Extreme learning machine: a new learning scheme of feedforward neural networks[C]//IEEE International Joint Conference on Neural Networks. Budapest, Hungary: IEEE, 2004: 985-990.

编 辑 叶 芳