文章编号:1007-5321(2020)01-0054-07 DOI:10.13190/j.jbupt.2019-063

基于马尔可夫链的人工蜂群算法

郭 佳[1,2], 马朝斌[1,2], 苗萌萌[2], 张绍博[2]
(1. 北京交通大学 计算机与信息技术学院, 北京 100044; 2. 国家保密科技测评中心, 北京 100044)

摘要: 分析了人工蜂群算法及部分国内外学者提出的改进算法,针对局部搜索能力差和容易陷入局部最优解的缺点,根据马尔可夫链预测已知解空间的发展趋势,提出了一种基于马尔可夫链的改进人工蜂群算法(MABC),通过伪代码给出了算法的运行过程,从收敛性能和算法复杂度 2 个方面分析了人工蜂群算法、一种典型的改进算法和 MABC 算法的性能. 最后以 10 个典型函数为测试用例,从结果精度、收敛速度、分割参数和运行时间 4 个方面进行验证,实验结果表明,MABC 算法在求解精度和收敛速度上高于 ABC 算法,但运行时间略长,验证了理论分析的结果.

关 键 词: 马尔可夫链; 人工蜂群算法; 函数优化
中图分类号: TP389 **文献标志码**: A

Markov Chain Based Artificial Bee Colony Algorithm

GUO Jia[1,2], MA Chao-bin[1,2], MIAO Meng-meng[2], ZHANG Shao-bo[2]
(1. School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China;
2. National Secrecy Science and Technology Evaluation Center, Beijing 100044, China)

Abstract: To overcome the shortcomings of existing local search ability and to easily obtain the local optimal solution of artificial bee colony algorithm (ABC), a new modified artificial bee colony algorithm (MABC) is proposed using the development trend of known solution space predicted by Markov Chain. The running process of the algorithm is provided through a pseudo code. The performances of the ABC and MABC are analyzed from two aspects: convergence performance and algorithm complexity. Using 10 typical functions as test cases, Experiments are carried out in four aspects: result precision, convergence speed, segmentation parameters and running time. It is shown that the MABC algorithm is superior to the ABC algorithm in terms of accuracy and convergence speed.

Key words: Markov chain; artificial bee colony algorithm; function optimization

Karaboga 等[1]首次提出了人工蜂群算法(ABC, artificial bee colony algorithm),该算法具有控制参数少、开拓性强和适应度高的特点[2],但因 ABC 重勘探,轻开采,在函数求解中存在收敛精度低和易于早熟的缺点,国内外学者提出了很多改进方法.

在改进搜索方程方面,Li 等[3]通过更新记忆结构,提出 ABC with memory;Zhu 等[4]用全局最优解 gbest 代替随机生成的邻域解,提出 Gbest-guided ABC(GABC);Banharnsakun 等[5]提出基于当前最优解某一维分量的改进算法 Best-so-far ABC;Jadon 等[6]提出了基于局部和全局信息的 ABC with global and local neighborhoods;Sharma 等[7]在雇佣蜂阶段加入局部最优解,在跟随蜂阶段加入全局最优解,提出 Lbest Gbest ABC. 这些方法利用相对范围内的最优值作为引导,易使算法依赖该值,陷入局部最优. 在此基础上,周新宇等[8]加入了开采到限值后放弃

收稿日期: 2019-04-17
作者简介: 郭 佳(1983—), 女, 工程师, E-mail:m13581902161@163.com.

的解信息，并采用正交实验设计生成新的食物源；Kiran 等[9]以前次解的更新方向作为引导，提出 Directed ABC；Du 等[10]将一般解和全局最优解同时引入搜索方程中. 这些方法仍然没有充分利用整个解空间的信息. 另一方面，一些算法将 ABC 算法与其他算法相结合，如粒子群、灰色系统理论[11]、蚁群算法[12]、萤火虫算法[13]和蝙蝠算法[14]等，在一定程度上改进了 ABC 算法的收敛精度和寻优速度，但并没有从根本上改变 ABC 算法的搜索方程.

笔者提出利用马尔可夫链(Markov chain)改进 ABC 的新方法 Markov chain ABC(MABC)，从搜索过程内在机制出发，利用解空间的内在联系将已有解依循环时步顺序转换为马尔可夫链，进一步根据发展规律进行预测，解决了在提高算法收敛性的同时如何增加种群多样性、避免陷入局部最优的问题.

1　ABC 算法的改进

1.1　马尔可夫链

马尔可夫链是一种描述动态随机现象的数学模型，它建立在系统“状态”和“状态转移”的概念之上[15]，其定义为：对 $n>1$，任意 $i_1,i_2,\cdots,i_{n-1},j\in S$ 恒有 $P\{X_n=j\mid X_1=i_1,X_2=i_2,\cdots,X_{n-1}=i_{n-1}\}=P\{X_n=j\mid X_{n-1}=i_{n-1}\}$，则称离散型随机过程 $\{X_t,t\in T\}$ 为马尔可夫链. 若假设状态空间是有限集，则称为有限状态马尔可夫链. 若在时刻 t_n，系统的状态为 $X_n=i$ 的条件下，下一时刻 t_{n+1} 系统状态为 $X_{n+1}=j$ 的概率 $P_{ij}(n)$ 与 n 无关，称马尔可夫链是齐次马尔可夫链[16]，由状态 S_i 经过 n 步转移到状态 S_j 的转移概率为

$$P_{ij}(n)=\frac{l_{ij}^{(n)}}{L_i} \tag{1}$$

其中：L_i 表示状态 S_i 出现的总次数，$l_{ij}^{(n)}$ 表示状态 S_i 经过 n 步转移到状态 S_j 的次数.

1.2　人工蜂群算法

人工蜂群算法用蜜源的位置表示解，用蜜源的蜂蜜量表示适应值，算法步骤如下.

1) 初始阶段. 确定蜜源数量 Q，解空间维度 D，候选解 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})$，其中 $i\in\{1,2,\cdots,Q\}$，蜜源范围为 $[x^{\min},x^{\max}]$，则初始蜜源为

$$x_{ij}=x_j^{\min}+\phi_{ij}(x_j^{\max}-x_j^{\min}) \tag{2}$$

其中：ϕ_{ij} 为 $(-1,1)$ 间的随机数，$j\in\{1,2,\cdots,D\}$，每次求解后，判断解的优劣，适应度函数为

$$\delta=\begin{cases}\dfrac{1}{1+f(x_{ij})}, & f(x_{ij})\geqslant 0\\ 1+|f(x_{ij})|, & f(x_{ij})<0\end{cases} \tag{3}$$

2) 雇佣蜂阶段. 产生随机候选解为

$$v_{ij}=x_{ij}+\phi_{ij}(x_{ij}-x_{kj}) \tag{4}$$

其中：x_{ij} 为第 i 个蜜源的第 j 维分量，x_{kj} 为随机选取的不同于 x_{ij} 的蜜源，$k\in\{1,2,\cdots,Q\}$，ϕ_{ij} 为 $(-1,1)$ 间的随机数，通过贪婪选择决定是否替换.

3) 跟随蜂阶段. 跟随蜂接收雇佣蜂的蜜源信息后进一步进行开采. 通过轮盘赌算法，根据蜜源的适应度 δ 按式(5)计算食物源被选中的概率 P_i，适应度越高，被选中的概率越大.

$$P_i=\frac{y_i}{\sum_{i=1}^{SN}y_i} \tag{5}$$

4) 侦察蜂阶段. 达到开采上限时，若适应度仍未更新则被淘汰，按式(2)随机生成新蜜源.

1.3　改进人工蜂群算法

将马尔可夫链与人工蜂群算法相结合，提出最优解空间的概念，应用于跟随蜂阶段，得到 MABC 算法. 算法分为两段，分割参数记为 ω，运行 ω 次后将已求得的解记为 Markov 解空间，依次形成按时步递进的 Markov 链，最小时步生成的解在最前面. 将 Marko 解空间划分 N 个状态子空间 S_n，有

$$S_n=[R^n,R^{n+1}] \tag{6}$$

$$R^n=X^{\omega,\min}+\frac{n-1}{N}(X^{\omega,\max}-X^{\omega,\min}) \tag{7}$$

其中：$n\in\{1,2,\cdots,N+1\}$，$X^{\omega,\max}$ 为 Markov 解空间内的最大值，$X^{\omega,\min}$ 为 Markov 解空间内的最小值.

将 Markov 解空间内的解 $X=(X^1,X^2,\cdots,X^Q)$ 分别置于状态子空间中，记为 $S_{n_{\text{start}}}$. 按式(1)计算 M 步状态转移概率(一般选取 $M=N$)，形成状态转移概率矩阵 $\boldsymbol{P}^{(m)}$，$m\in\{1,2,3,\cdots,M\}$. 选取离预测值最近的 M 个时步的状态进行预测，以离预测时步的远近为顺序，在 m 步状态转移矩阵中，选取与 m 时步相对应的起始状态所在的行向量，组成预测矩阵，转移到状态子空间 S_n 的总概率为

$$P_n=\sum_{m=1}^{M}P^{(m)}(S_{n_{\text{start}}},S_n) \tag{8}$$

其中 $P^{(m)}(S_{n_{\text{start}}},S_n)$ 表示 m 步状态转移概率矩阵中第 $S_{n_{\text{start}}}$ 行第 S_n 列的值. 概率最大的空间状态即为预测解所在的子空间.

1.4　改进算法伪代码

MABC 主函数代码与 ABC 算法一致，在跟随蜂

阶段插入 Markov 预测子函数,伪代码如算法 1.

算法 1 Markov 预测子函数伪代码

Markov(输入为已有蜜源 X,输出为新的蜜源 V)
1 设置 Markov 子空间的个数和时步 N
2 按照当前解的时步顺序进行排序依次形成按时步递进的 Markov 链
3 while $(n<N)$
4 按式(7)计算状态空间边界
5 end
6 记录所有解落入的状态空间
7 $i=0$
8 while$(i<N)$
9 记录经过 i 个时步后解所属状态空间的变化
10 end
11 得出 n 个状态空间转移次数矩阵
12 按式(1)计算出 n 个状态转移概率矩阵
13 按式(8)计算转移到状态子空间的总概率
14 获得最优状态子空间后,取其平均值得出 V

2 算法性能分析

2.1 收敛性分析

ABC 算法的雇佣蜂和跟随蜂按式(4)更新蜜源,式中参数 x_{kj} 和 ϕ_{ij} 均为随机产生,因此更新的解具有很强随机性,蜂群搜索经验丢失,且当 x_{ij} 接近最优蜜源时,较大的 $\phi_{ij}(x_{ij}-x_{kj})$ 会引起算法震荡. GABC 搜索方程为 $v_{ij}=x_{ij}+\phi_{ij}(x_{ij}-x_{kj})+\phi_{ij}(y_j-x_{ij})$,虽然加入了 j 维空间的最优解 y_j,引导了方程的搜索方向,减少了随机性,但在一定程度上降低了算法的全局寻优能力,尤其在多峰类型函数中,若 y_j 位于局部最优解邻域,算法会逐渐趋于局部最优解. 改进的 MABC 算法将最优解空间的信息加入搜索过程,一方面充分保留已有解经验,在具有最大转移概率的子解空间内生成新解,保留有益信息,具有继承性;另一方面不依赖某一个或几个最优解,具有一定的开拓性,避免陷入局部最优. 在全局扩展性能和局部收敛性能之间找到了平衡点.

2.2 算法复杂度分析

设定人工蜂群算法中食物源的个数为 q,维度为 d,迭代次数为 t,则算法初始化的复杂度为 qd.

在 ABC 和 GABC 算法的雇佣蜂阶段,搜索复杂度为 qd,适应度值计算复杂度为 q;跟随蜂阶段,贪婪选择复杂度为 q,设选中一半跟随蜂,则搜索复杂度为 $\frac{qd}{2}$,适应度值计算复杂度为 q;若达到开采上限后仍未更新,则侦查蜂随机搜索复杂度为 d. 整体复杂度为:$T_{\mathrm{ABC}}=qd+t\left(\frac{3}{2}qd+3qd\right)$.

MABC 算法的主要区别是搜索复杂度,也就是 Markov 过程的复杂度,其中解所属状态空间判断复杂度为 qd,状态转移概率矩阵计算复杂度为 $\frac{d(d-1)}{2}q$,预测新解仅需 1 步,不计入,则整体复杂度为 $T_{\mathrm{MABC}}=qd+t\left(qd+\frac{d(d-1)}{2}q+3q+d\right)$. 记算复杂度之差为 $T_{\mathrm{MABC}}-T_{\mathrm{ABC}}=\frac{d(d-2)}{2}qt$.

当维度 d 大于 2 时,MABC 算法更加复杂,且随着维度和食物源增加,复杂度也随之增加. 可见,MABC 算法虽然可以提高算法的精度,却要牺牲一定的计算资源. 但在寻优过程中,往往会在算法达到一定精度后触发停止条件,所以 T_{MABC} 中的 t 会小于 T_{ABC},从而降低 MABC 算法的复杂度.

3 实验验证

设计 5 个实验,分别从结果精度、收敛速度、分割参数和 CPU 运行时间 4 个方面分析. 选取 10 个经典测试函数进行实验,如表 1 所示,不同函数的公共参数采取相同设置.

这 10 个函数包括单峰、多峰以及旋转 3 类[17-19]. 其中 f_1、f_6、f_8 和 f_9 是单峰函数,仅有一个极值点,f_4 是很难极小化的典型病态非凸单峰函数,难以找到全局最优点;f_3、f_7 和 f_{10} 是多峰函数,其中 f_7 是连续的平滑多峰函数,f_3 是具有大量局部最优点的复杂多峰函数,f_{10} 在距离全局最优点大约 3.14 的范围内存在无限多的局部最优点,函数强烈震荡,难以收敛. f_2 和 f_5 为旋转、不可分离的多峰函数,具有大量的局部最优点.

3.1 结果精度比较

3.1.1 不同算法结果精度比较

取维度为 30,分割参数为 100,后续实验会针对这 2 个参数进行展开比对. 取食物源个数为 10,开采上限有多种典型的取值[8],为达到最高的函数执行频率,取 100. 最大循环次数为 500,进行 30 次实验,测试结果如表 2 所示,其中最后一列为 wilcoxon[20] 非参数统计结果(显著水平为 0.05),“+”、“=”、“-”的含义分别表示 MABC 好于、等于、差于

表 1　实验用函数

函数	数学描述	搜索范围	理论最优值
f_1	$\min f(x_i)=\sum_{i=1}^{D}x_i^2$	$[-100,100]$	0
f_2	$\min f(x_i)=\sum_{i=1}^{D}\frac{x_i^2}{4\,000}-\prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]$	0
f_3	$\min f(x_i)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]$	0
f_4	$\min f(x_i)=\sum_{i=1}^{D-1}[100(x_i^2-x_{i+1})^2+(x_i-1)^2]$	$[-30,30]$	0
f_5	$\min f(x_i)=-20\,e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D}x_i^2}}-e^{\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)}+22.712\,82$	$[-32.768,32.768]$	0
f_6	$\min f(x_i)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]$	0
f_7	$\min f(x_i)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-100,100]$	0
f_8	$\min f(x_i)=\sum_{i=1}^{D}x_i^2+\left(\frac{1}{2}\sum_{i=1}^{D}ix_i\right)^2+\left(\frac{1}{2}\sum_{i=1}^{D}ix_i\right)^4$	$[-5,10]$	0
f_9	$\min f(x_i)=\sum_{i=1}^{D}ix_i^2$	$[-10,10]$	0
f_{10}	$\min f(x_i)=0.5+\frac{\sin^2\left(\sqrt{\sum_{i=1}^{D}x_i^2}\right)-0.5}{1+0.001\left(\sqrt{\sum_{i=1}^{D}x_i^2}\right)}$	$[-100,100]$	0

表 2　不同算法寻优精度结果比较

函数	算法	平均值	标准差	最优值	最差值	显著性
f_1	ABC	5.67×10^{-4}	9.43×10^{-4}	1.62×10^{-5}	4.14×10^{-3}	+
	MABC	1.37×10^{-6}	6.02×10^{-6}	1.11×10^{-12}	3.36×10^{-5}	
f_2	ABC	2.27×10^{-2}	2.53×10^{-2}	6.93×10^{-5}	9.86×10^{-2}	+
	MABC	1.94×10^{-7}	6.28×10^{-7}	1.80×10^{-14}	3.00×10^{-6}	
f_3	ABC	1.11×10	3.72	2.57	2.09×10	+
	MABC	1.50×10^{-2}	7.91×10^{-2}	4.28×10^{-8}	4.41×10^{-1}	
f_4	ABC	4.40×10^{2}	9.41×10^{2}	1.25×10	4.76×10^{3}	+
	MABC	4.46×10	3.35×10	4.86	1.03×10^{2}	
f_5	ABC	2.00×10	7.25×10^{-4}	2.00×10	2.00×10	+
	MABC	2.83×10^{-4}	5.49×10^{-4}	9.59×10^{-8}	2.27×10^{-3}	
f_6	ABC	6.67×10^{-1}	5.47×10^{-1}	0	2.00	+
	MABC	1.33×10^{-1}	3.46×10^{-1}	0	1.00	
f_7	ABC	3.97×10^{-2}	2.16×10^{-2}	1.53×10^{-2}	1.06×10^{-1}	+
	MABC	8.61×10^{-3}	3.9×10^{-3}	1.77×10^{-3}	2.02×10^{-2}	
f_8	ABC	1.64×10^{-4}	3.41×10^{-4}	7.16×10^{-6}	1.77×10^{-3}	+
	MABC	1.04×10^{-6}	2.85×10^{-6}	1.33×10^{-11}	1.28×10^{-5}	
f_9	ABC	1.17×10^{-3}	1.18×10^{-3}	1.04×10^{-4}	4.69×10^{-3}	+
	MABC	2.14×10^{-5}	6.22×10^{-5}	4.80×10^{-12}	2.57×10^{-4}	
f_{10}	ABC	3.24×10^{-1}	4.38×10^{-2}	1.94×10^{-1}	4.02×10^{-1}	+
	MABC	4.98×10^{-2}	7.73×10^{-3}	3.02×10^{-2}	6.80×10^{-2}	

ABC 算法.

根据表 2 可以看出,在函数寻优过程中,改进 MABC 算法的结果精度均比 ABC 算法有明显提高.

3.1.2　扩展维度结果精度比较

函数维度对寻优过程影响较大,一般情况下,维度越高寻优难度越大. 实验通过设置不同维度来对比算法的寻优精度. 最大循环次数、食物源个数和分割参数取值同 3.1.1 节. 分别选取单峰函数 f_1、多峰函数 f_3 和旋转多峰函数 f_2 进行 30 次试验,3.1.1 节已对维度为 30 的情况进行了实验,故本次实验分别比对维度为 10、20、40 和 50 时 2 种算法的结果精度. 测试结果如表 3 所示.

根据表 3 可以看出,在维度为 10 时,函数 f_1 和 f_3 上运行 MABC 算法的寻优精度低于 ABC 算法,在高维度时均高于 ABC 算法. 对于函数 f_1,首先因为在解为低维时单峰函数 f_1 较容易得到全局最优解,其次本实验中,参数 ω 设置为 100,根据 3.3 节实验可知,函数 f_1 在 ω 较低时寻优精度较低. 当把 ω 设置为 300 时再进行一次维度为 10 的实验,MABC 算法的寻优精度便明显优于 ABC 算法. 对于函 f_3,是在基于 f_1 的基础上,使用了余弦函数来产生大量的

表3 不同维度寻优精度结果比较

函数	维度	算法	平均值	标准差	最优值	最差值	显著性
f_1	10	ABC	1.15×10^{-17}	1.78×10^{-17}	5.52×10^{-21}	9.11×10^{-17}	−
		MABC	1.90×10^{-16}	7.37×10^{-17}	8.24×10^{-17}	3.06×10^{-16}	
	20	ABC	4.95×10^{-7}	9.08×10^{-7}	7.08×10^{-9}	4.64×10^{-6}	+
		MABC	6.93×10^{-10}	3.66×10^{-9}	5.12×10^{-16}	2.01×10^{-8}	
	40	ABC	6.47×10^{-3}	6.32×10^{-3}	6.15×10^{-4}	2.76×10^{-2}	+
		MABC	7.79×10^{-6}	3.52×10^{-5}	6.38×10^{-11}	1.93×10^{-4}	
	50	ABC	2.88×10^{-1}	5.22×10^{-1}	1.33×10^{-2}	2.29	+
		MABC	9.33×10^{-6}	1.68×10^{-5}	9.45×10^{-11}	7.58×10^{-5}	
f_2	10	ABC	1.68×10^{-2}	1.21×10^{-2}	1.32×10^{-7}	4.19×10^{-2}	+
		MABC	2.75×10^{-3}	6.58×10^{-3}	0	2.34×10^{-2}	
	20	ABC	1.10×10^{-2}	1.17×10^{-2}	8.00×10^{-8}	3.46×10^{-2}	+
		MABC	2.87×10^{-4}	1.41×10^{-3}	1.11×10^{-1}	7.72×10^{-3}	
	40	ABC	7.64×10^{-2}	6.67×10^{-2}	8.87×10^{-4}	2.53×10^{-1}	+
		MABC	1.35×10^{-4}	4.32×10^{-4}	5.55×10^{-13}	2.14×10^{-3}	
	50	ABC	3.14×10^{-1}	1.63×10^{-1}	5.75×10^{-2}	6.42×10^{-1}	+
		MABC	1.19×10^{-3}	6.22×10^{-3}	1.25×10^{-12}	3.41×10^{-2}	
f_3	10	ABC	1.06×10^{-13}	1.53×10^{-13}	0	7.11×10^{-13}	−
		MABC	5.48×10^{-6}	1.39×10^{-5}	1.11×10^{-9}	5.81×10^{-5}	
	20	ABC	3.67×10^{-1}	5.55×10^{-1}	2.73×10^{-6}	1.99	+
		MABC	5.68×10^{-4}	1.08×10^{-3}	1.18×10^{-6}	4.69×10^{-3}	
	40	ABC	1.04×10	3.81	5.43	2.05×10	+
		MABC	2.21×10^{-3}	5.66×10^{-3}	5.06×10^{-9}	2.87×10^{-2}	
	50	ABC	2.38×10	8.12	1.04×10	4.28×10	+
		MABC	2.98×10^{-3}	4.02×10^{-3}	4.13×10^{-5}	1.54×10^{-2}	

局部小值,故在低维时 ABC 算法很容易接近全局最优值. 函数 f_2 随着维数增加,局部最优的范围越来越窄,故此函数从低维到高维的寻优难度没有一个明显的增加,因此,不同维数下 MABC 算法的寻优精度均高于 ABC 算法.

3.2 收敛速度比较

本次实验对比 MABC 算法与 ABC 算法的收敛速度. 最大循环次数、维度、分割参数和食物源个数取值同 3.1.1 节,分别选取单峰函数 f_1、多峰函数 f_3 和旋转多峰函数 f_2 进行 30 次试验. 首先对每个测试函数设定预定精度,若在最大循环次数内达到了该精度,则认为此次寻优成功,否则认为失败. 测试结果如表 4 所示.

根据表 4 可以看出,在指定预设精度前提下,MABC 算法的成功率明显高于 ABC 算法,也就说明 MABC 算法的收敛速度高于 ABC.

表4 给定精度下算法成功率比较

函数	预设精度	算法	平均值	标准差	算法成功率/%
f_1	5.00×10^{-5}	ABC	2.94×10^{-4}	3.64×10^{-4}	16.67
		MABC	1.84×10^{-5}	1.72×10^{-5}	100.00
f_2	5.00×10^{-5}	ABC	2.36×10^{-2}	2.80×10^{-2}	6.67
		MABC	3.20×10^{-5}	4.50×10^{-5}	96.67
f_3	1.00×10^{-2}	ABC	1.06×10	4.70	0
		MABC	6.04×10^{-2}	8.39×10^{-2}	96.67

3.3 分割参数对结果的影响

不同分割参数的取值对函数寻优结果有直接影响,本次实验通过设置不同分割参数值来验证对 MABC 算法性能的影响. 设置维度为 30,最大循环次数、维度和食物源个数取值同 3.1.1 节,分割参数从 10 开始,每增加 20 运行一次,当增至 500 时,代

表不运行 MABC 算法，与 ABC 算法等同. 分别选取单峰函数 f_1、多峰函数 f_3 和旋转多峰函数 f_2 进行 30 次试验，测试结果如表 5 所示.

表 5　不同分割参数寻优精度结果比较

函数	分割次数	平均值	标准差	最优值	最差值
	50	3.00×10^{-7}	1.33×10^{-6}	4.92×10^{-13}	7.32×10^{-6}
	250	1.45×10^{-8}	2.76×10^{-8}	3.71×10^{-16}	9.62×10^{-8}
	400	1.38×10^{-9}	4.72×10^{-9}	2.37×10^{-15}	2.42×10^{-8}
	410	9.06×10^{-10}	2.32×10^{-9}	1.87×10^{-14}	1.12×10^{-8}
f_1	430	8.53×10^{-11}	1.95×10^{-10}	1.78×10^{-15}	1.05×10^{-9}
	450	9.01×10^{-10}	1.88×10^{-9}	1.36×10^{-13}	6.48×10^{-9}
	470	1.85×10^{-9}	5.34×10^{-9}	3.04×10^{-14}	2.60×10^{-8}
	490	6.34×10^{-9}	1.08×10^{-8}	4.88×10^{-12}	5.41×10^{-8}
	500	1.56×10^{-5}	1.72×10^{-5}	6.87×10^{-7}	6.85×10^{-5}
	10	8.04×10^{-7}	4.37×10^{-6}	2.09×10^{-14}	2.39×10^{-5}
	30	4.36×10^{-7}	1.91×10^{-6}	6.99×10^{-14}	1.03×10^{-5}
	50	9.42×10^{-9}	2.71×10^{-8}	3.20×10^{-14}	1.22×10^{-7}
	70	5.63×10^{-7}	1.80×10^{-6}	6.66×10^{-16}	8.30×10^{-6}
f_2	90	1.83×10^{-6}	4.52×10^{-6}	6.33×10^{-15}	1.77×10^{-5}
	100	6.63×10^{-4}	3.46×10^{-3}	3.47×10^{-13}	1.90×10^{-2}
	250	1.51×10^{-4}	5.46×10^{-4}	2.05×10^{-14}	2.74×10^{-3}
	400	6.80×10^{-4}	2.87×10^{-3}	5.88×10^{-15}	1.57×10^{-2}
	500	1.38×10^{-3}	3.96×10^{-3}	2.00×10^{-7}	1.77×10^{-2}
	10	4.95×10^{-4}	7.13×10^{-4}	2.78×10^{-6}	2.98×10^{-3}
	30	9.46×10^{-4}	1.21×10^{-3}	3.81×10^{-8}	5.03×10^{-3}
	50	6.41×10^{-4}	1.07×10^{-3}	4.49×10^{-9}	4.85×10^{-3}
	70	7.23×10^{-4}	1.06×10^{-3}	1.26×10^{-7}	4.82×10^{-3}
f_3	90	1.22×10^{-3}	3.62×10^{-3}	8.82×10^{-9}	1.99×10^{-2}
	100	8.06×10^{-4}	1.18×10^{-3}	9.81×10^{-7}	5.58×10^{-3}
	250	2.88×10^{-3}	5.63×10^{-3}	1.25×10^{-6}	2.63×10^{-2}
	400	1.12×10^{-2}	2.79×10^{-2}	2.64×10^{-8}	1.34×10^{-1}
	500	1.44×10	6.10	3.49	2.91×10

因实验结果较多，仅列出集中于函数取得寻优结果最优值对应的 ω 及其周边的取值. 从表 5 可以看出 ω 取值小于 500 的实验结果均优于取值等于 500 的结果，进一步证明改进的 MABC 算法性能优于 ABC 算法. 同时可以看出，在函数 f_1 的寻优过程中，最优值出现在 ω 较大值阶段（$\omega=430$），而在 f_2 和 f_3 的寻优过程中，最优值出现在 ω 较小值阶段（$\omega=50$）. 因为函数 f_1 为单峰函数，寻找极值点较容易，原始的 ABC 算法收敛方向比较明确，越到后期，对 Markov 链的预测越有较大的指导意义；而 f_2 和 f_3 均是多峰函数，有多个局部极值点，寻找极值点较困难，故越早使用 MABC 算法，越利于函数寻优.

3.4　算法运行时间比较

本次实验对比 MABC 算法与 ABC 算法的运行时间，验证 2.2 节对算法复杂度的分析. 分为两组实验，一组最大循环次数、维度、分割参数和食物源个数取值同 3.1.1 节；另一组在第一组的基础上设置与 3.2 节实验一致的终止条件. 分别选取单峰函数 f_1、多峰函数 f_3 和旋转多峰函数 f_2 进行 30 次试验，记录每次 CPU 运行时间并计算平均值. 第 1 组测试结果如表 6 所示，第 2 组测试结果如表 7 所示. 算法的运行平台为：Windows 10（15063.1955）操作系统，CPU 为 Intel（R） Core（TM）i7-7700HQ CPU 2.80 GHz，内存为 32.0 GB，编程环境为 Matlab R2016b.

表 6　未给定预设精度下 CPU 运行时间

函数	预设精度	算法	CPU 时间
f_1	–	ABC	0.999
		MABC	4.225
f_2	–	ABC	1.367
		MABC	4.836
f_3	–	ABC	1.110
		MABC	4.621

表 7　给定预设精度下 CPU 运行时间

函数	预设精度	算法	CPU 时间
f_1	5.00×10^{-5}	ABC	1.037
		MABC	1.749
f_2	5.00×10^{-5}	ABC	1.360
		MABC	1.723
f_3	1.00×10^{-2}	ABC	0.912
		MABC	1.524

从表 6 和表 7 可以看出，MABC 算法比 ABC 算法运行时间长很多，但当限定预设精度后，由于 MABC 算法比 ABC 算法提早达到精度，所以循环次数少，总体 CPU 运行时间比 ABC 算法略长，与 2.2 节分析结果一致.

4　结束语

利用马尔可夫链适用于多种复杂因素影响时间序列预测的特性，将马尔可夫链与人工蜂群算法相结合对蜜源进行预测，提出了 MABC 算法，从理论

上分析得出,改进算法提高了原始 ABC 算法的性能,并通过实验验证得出 MABC 算法具有较高的求解精度及收敛速度. 在后期的研究中,将进一步细化 MABC 算法,继续深入研究最优解状态空间的动态变化,可根据最新预测值将状态空间逐渐缩小,进一步提高算法性能. 此外,将提出的算法应用到多目标优化、网络性能优化、图像处理和工程安全等领域,也是值得进一步研究的工作.

参考文献:

[1] Karaboga D, Kaya E. An adaptive and hybrid artificial bee colony algorithm (aABC) for ANFIS training[J]. Neural Computing and Applications, 2014, 25(7-8): 1967-1978.

[2] Goel S, Singh J, Ojha N. Intelligent aircraft landing decision support system using artificial bee colony[C]// Proc of the 3th International Conference on Computing for Sustainable Global Development. Piscataway: IEEE, 2016: 2412-2416.

[3] Li X N, Yang G F. Artificial bee colony algorithm with memory[J]. Applied Soft Computing, 2016, 41(4): 362-372.

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

[5] Banharnsakun A, Sirinaovakul B, Achalakul T. The performance and sensitivity of the parameters setting on the best-so-far ABC[C]//International Conference on Simulated Evolution & Learning. [S.l.]: Springer-Verlag, 2012: 248-257.

[6] Jadon S S, Bansal J C, Tiwari R, et al. Artificial bee colony algorithm with global and local neighborhoods [J]. International Journal of System Assurance Engineering and Management, 2010, 1(3): 189 – 200.

[7] Sharma H, Sharma S, Kumar S. Lbest Gbest artificial bee colony algorithm[C]// International Conference on Advances in Computing. NY: IEEE, 2016: 119-124.

[8] 周新宇, 吴志健, 王文明. 基于正交实验设计的人工蜂群算法[J]. 软件学报, 2015, 26(9): 2167-2190.
Zhou Xinyu, Wu Zhijian, Wang Wenming. Artificial bee colony algorithm based on orthogonal experimental design [J]. Journal of Software, 2015, 26(9): 2167-2190.

[9] Kiran M S, Findik O. A directed artificial bee colony algorithm[J]. Applied Soft Computing, 2015, 46(2): 534-546.

[10] Du Z, Han D, Li K C. Improving the performance of feature selection and data clustering with novel global search and elite-guided artificial bee colony algorithm [J]. The Journal of Supercomputing, 2019, 75(2): 5189-5226.

[11] Rajasingam N, Rasi D, Deepa S N. Optimized deep learning neural network model for doubly fed induction generator in wind energy conversion systems[J]. Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2019, 23(8): 8453-8470.

[12] Mala D J, Kamalapriya M, Shobana R, et al. A non-pheromone based intelligent Swarm Optimization Technique in Software Test Suite Optimization [C]//International Conference on Intelligent Agent & Multi-agent Systems. [S.l.]: IEEE Press, 2009: 1-5.

[13] Tuba M, Bacanin N. Artificial bee colony algorithm hybridized with firefly algorithm for cardinality constrained mean-variance portfolio selection problem[J]. Applied Mathematics & Informaion Sciences, 2014, 8(6): 2831-2844.

[14] Neelima S, Satyanarayana N, Murthy P K. Optimization of association rule mining using hybridized artificial bee colony (ABC) with BAT algorithm[C]//Advance Computing Conference. [S.l.]: IEEE, 2017: 831-834.

[15] Zhiyan Shi, Dan Bao, Yan Fan, et al. The asymptotic equipartition property of Markov chains in single infinite Markovian environment on countable state space[J]. Stochastics, 2019, 91(1): 1-13.

[16] Komorowski T, Peszat S, Szare K. On ergodicity of some Markov processes[J]. Annals of Probability, 2010, 38(4): 1401-1443.

[17] Suganthan P N, Hansen N, Liang J J, et al. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization[R]. Singapore: Nanyang Technological University, 2005.

[18] Zhou X, Wu Z, Wang H, et al. Enhancing differential evolution with role assignment scheme[J]. Soft Computing, 2014, 18(11): 2209-2225.

[19] Xiong G, Shi D, Duan X. Enhancing the performance of biogeography-based optimization using polyphyletic migration operator and orthogonal learning[J]. Computers & Operations Research, 2014, 41(1): 125-139.

[20] 杜振鑫, 刘广钟, 韩德志. 改进基于记忆的人工蜂群算法[J]. 北京邮电大学学报, 2017, 40(5): 61-66.
Du Zhenxin, Liu Guangzhong, Han Dezhi. An improved artificial bee colony algorithm with memory[J]. Journal of Beijing University of Posts and Telecommunications, 2017, 40(5): 61-66.