控制与决策
Control and Decision

基于精英解和随机个体邻域信息的改进人工蜂群算法

孟红云， 位冰可

引用本文:
孟红云, 位冰可. 基于精英解和随机个体邻域信息的改进人工蜂群算法[J]. 控制与决策, 2020, 35(9): 2169−2174.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2018.1797

基于精英解和随机个体邻域信息的改进人工蜂群算法

孟红云†, 位冰可

(西安电子科技大学 数学与统计学院，西安 710071)

摘 要: 针对人工蜂群(ABC)算法开发能力差、收敛速度慢的缺点,分别提出适用于雇佣蜂和观察蜂阶段的搜索方程,其中前者用到精英解、随机选择个体及其邻域的有益信息,后者用到群体最优解的信息. 所提出的搜索方程在一定程度上不仅能够加快改进算法的收敛速度,而且由于随机选择个体的引入在一定意义上可以保证算法的探索能力. 对22个基准测试函数的仿真实验结果表明,所提出的算法在大多数测试函数上的性能优于对比算法.

关键词: 人工蜂群算法；精英解；邻域信息；欧氏距离

中图分类号: TP18 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2018.1797

引用格式: 孟红云,位冰可. 基于精英解和随机个体邻域信息的改进人工蜂群算法[J]. 控制与决策, 2020, 35(9): 2169-2174.

An improved artificial bee colony algorithm based on elite solution and random individual neighborhood information

MENG Hong-yun†, WEI Bing-ke

(School of Mathematics and Statistics，Xidian University，Xi'an 710071，China)

Abstract: Aiming at the disadvantages of the artificial bee colony (ABC) algorithm, such as poor exploitation ability and slow convergence speed, the search equations for the employed bee phase and the onlooker bee phase are proposed respectively. The former exploits the beneficial information from the elite solution, randomly selected individual and its neighborhood, and the latter exploits the information from the optimal solution of the population. The proposed search equations not only accelerate the convergence speed of the improved algorithm to some extent, but also guarantee the exploration ability of the algorithm in a certain sense due to the introduction of randomly selected individuals. The simulation results of 22 benchmark functions demonstrate that the proposed algorithm is superior to the comparison algorithms on most test functions.

Keywords: artificial bee colony algorithm；elite solution；neighborhood information；Euclidean distance

0 引 言

随着科学技术的发展,优化问题越来越复杂,其特点是具有非凸性、不连续性、不可微性和多模态性[1],利用传统的基于导数的方法求解这些全局优化问题成为一个巨大的挑战. 为了解决这些复杂问题,进化算法作为一种新的无导数优化技术被提出. 在进化算法领域,一些主流算法包括差分进化算法(DE)[2]、遗传算法(GA)[3]、人工蜂群算法(ABC)[4]和粒子群算法(PSO)[5]等.

ABC算法是由Karaboga[4]于2005年提出的一种基于群体智能的随机优化方法,其灵感来源于蜜蜂的摇摆舞和智能觅食行为,已被证明是一种解决复杂优化问题非常有效的方法. 近年来,ABC算法由于其结构简单、实现方便、性能突出等优点,已成功扩展到解决多目标优化问题[6]、二元优化问题[7]、数据聚类问题[8]等众多应用问题中. 然而,ABC算法收敛速度较慢,这主要是由其搜索方程造成的,该搜索方程善于探索,不善于开发,但智能算法的性能评价取决于能否保持探索与开发之间的适当平衡. 因此,为了弥补ABC算法的不足,进一步提高ABC算法的性能,学者们提出了ABC的改进算法,并给出了新的搜索方程[9-10]. 实验表明,改进的ABC算法可以提高ABC算法的性能,然而,不可能有一种方法能有效解决每一个问题,因此ABC算法性能还有待提高.

针对ABC算法的缺点,本文提出两个新的搜索方程,以进一步改善ABC算法的性能. 一般而言,

收稿日期: 2018-12-30；**修回日期:** 2019-04-02.
基金项目: 国家自然科学基金项目(61401322, 61877066).
责任编委: 陈家伟.
†通讯作者. E-mail: menghy@xidian.edu.cn.

ABC算法的雇佣蜂阶段偏向于探索,观察蜂阶段更偏向于开发,本文的两个搜索方程也保持这个原则. 雇佣蜂阶段利用精英解、随机选择个体及其邻域范围内的最佳解指导蜜蜂的搜索,观察蜂阶段利用精英解、当前最优解的信息以及随机选择个体的邻域信息指导蜜蜂的搜索. 通过这种方式不仅加快了算法的收敛速度,而且由于随机选择个体的引入在一定程度上还保证了算法的探索能力. 在观察蜂阶段省去轮盘赌步骤,直接将计算资源分配给质量好的食物源(精英解),在一定程度上节约了计算量.

1 基本人工蜂群算法

ABC算法依次分为4个阶段,每个阶段细节如下所述:

1) 在初始化阶段,初始化几个基本参数(种群数SN,维度数D,最大函数评价次数max.FES,预定周期limit),然后随机产生一组表示优化问题可能解的食物源位置,即

$$x_{i,j}=x_j^{\min}+\mathrm{rand}(0,1)\cdot(x_j^{\max}-x_j^{\min}). \quad (1)$$

其中:$i\in\{1,2,\dots,\mathrm{SN}\},j\in\{1,2,\dots,D\},x_j^{\max}$和$x_j^{\min}$分别为解第$j$维变量的上界和下界.

2) 在雇佣蜂阶段,每只雇佣蜂占据一个食物来源的位置,候选食物源位置(新解)由雇佣蜂根据以下搜索方程生成:

$$v_{i,j}=x_{i,j}+\phi_{i,j}\cdot(x_{i,j}-x_{k,j}). \quad (2)$$

其中:$i\in\{1,2,\dots,\mathrm{SN}\},j\in\{1,2,\dots,D\},k\neq i,\phi_{i,j}$为$[-1,1]$之间的随机数.

3) 在观察蜂阶段,每只观察蜂根据食物源位置的概率值,用轮盘赌方法选择一个食物源位置进行开发,概率计算公式如下:

$$P_i=\mathrm{fitness}_i\Big/\sum_{j=1}^{\mathrm{SN}}\mathrm{fitness}_j. \quad (3)$$

其中:$\mathrm{fitness}_i$为第i个食物源的适应度值,理论上,适应度值越大,选择概率越大.

考虑最小化问题时,食物来源的适应度值一般计算如下:

$$\mathrm{fitness}_i=\begin{cases}\dfrac{1}{1+f_i},\ f_i\geqslant 0;\\ 1+|f_i|,\ \text{otherwise}.\end{cases} \quad (4)$$

其中$\mathrm{fitness}_i$和f_i分别为食物源x_i的适应度值和目标函数值. 一旦食物源被选择,即按照式(2)进行位置更新操作.

4) 在侦察蜂阶段,一旦某种食物源的位置(解)不能在预定周期内被更新,那么这个食物源便会被抛弃,新食物源位置根据式(1)随机生成.

2 基于精英解和随机个体邻域信息的改进人工蜂群算法

2.1 研究动机

本文的研究动机主要来源于文献[11]. 在ABC算法中,每一代蜜蜂都在独立地搜索,而在自然界中,当一只雇佣蜂找到高质量的食物源时,它会用摇摆舞来通知周围的同伴. 因此,在雇佣蜂阶段,搜索方程可用一些其他有用的信息指导搜索,这样可以加快收敛速度,同时也避免陷入局部最优. 为此,本文设计了一个新的搜索方程,用到精英解的有益信息和随机选择的个体以及其邻域最优解的信息. 在ABC算法中,观察蜂可以飞到任何食物源的位置进一步探索. 然而,为了减轻选择的压力和运行时间,省去轮盘赌过程,在观察蜂阶段直接对精英解进行进一步的搜索. 此外,对观察蜂的搜索还可以由整个种群的全局信息来指导. 综上所述,通过改变雇佣蜂和观察蜂的搜索方程,以期提高人工蜂群算法的优化性能.

2.2 雇佣蜂的搜索方程

在ABC算法中,雇佣蜂在自己的食物源位置附近盲目地寻找. 为了在不影响种群多样性的前提下加快收敛速度,将目标函数值从小到大排序,排名在前$T(T=p\cdot\mathrm{SN},p\in(0,1))$的目标函数值所对应的食物源作为精英解,在雇佣蜂阶段用到精英解的有益信息和随机选择的个体k及其邻域的信息. 精英解的作用是可以加快收敛,k的引入可以加强探索,同时避免陷入局部最优. 新的雇佣蜂搜索方程为

$$v_{i,j}=\frac{x_{\mathrm{knbest},j}+x_{e,j}}{2}+\phi_{i,j}\cdot(x_{\mathrm{knbest},j}-x_{k,j}). \quad (5)$$

其中:x_e为随机选择的一个精英解,$e\neq i;x_k$从种群中随机选择,$k\in\{1,2,\dots,\mathrm{SN}\},k\neq e\neq i;x_{\mathrm{knbest},j}$为第$k$个个体的邻域范围内目标函数值最小的最佳食物源位置;j为随机选择的下标,$j\in\{1,2,\dots,D\}$;$\phi_{i,j}$为$[-1,1]$之间的随机数.

如果x_{knbest}为空,则用ABC算法的搜索方程式(2)进行搜索.

2.3 观察蜂的搜索方程

ABC算法中,在观察蜂阶段,每只观察蜂根据食物源位置的概率值用轮盘赌方法选择一个食物源位置进行开发,虽然轮盘赌方法选到高质量食物源的概率较大,但仍有一定的概率选到质量比较差的食物源,浪费了一定的评价次数,收敛速度较慢. 因此,在观察蜂阶段省去轮盘赌环节,直接对精英解进行更新. 新的观察蜂搜索方程为

$$v_{i,j}=\frac{x_{\mathrm{best},j}+x_{e,j}}{2}+\phi_{e,j}\cdot(x_{\mathrm{knbest},j}-x_{k,j}). \quad (6)$$

其中: x_e 为随机选择的一个精英解, $k \neq e$; $x_{\rm best}$ 为当代最优解; $\phi_{e,j}$ 为 $[-1, 1]$ 之间的随机数.

如果 $x_{\rm knbest}$ 为空, 则用ABC算法的搜索方程式(2)进行搜索.

注意到, 式(6)与文献[11]观察蜂的搜索方程很相似, 即

$$v_{i,j} = \frac{x_{{\rm best},j} + x_{e,j}}{2} + \phi_{e,j}(x_{{\rm best},j} - x_{k,j}). \quad (7)$$

式(7)利用当代最优解的信息指导蜜蜂的搜索, 易陷入局部最优; 而式(6)用随机选择个体邻域最优解的信息指导蜜蜂的搜索, 具有一定的探索能力.

2.4　邻域范围及算法伪代码

本文与文献[12]一样, 将平均欧氏距离md作为个体的邻域范围. 特别地, 对于第 i 个个体, 其邻域范围计算如下:

$$\mathrm{md}_i = \frac{\sum_{j=1}^{\rm SN} d_{i,j}}{\rm SN - 1}. \quad (8)$$

其中: SN为食物源的个数, $d_{i,j}(i \neq j)$ 为食物源位置 x_i 与 x_j 之间的欧氏距离. 如果 x_j 与 x_i 之间的欧几里得距离小于平均欧氏距离 md_i, 则食物源位置 x_j 位于第 i 个食物源的邻域范围内.

算法1　本文算法的伪代码.

00 参数初始化(种群数, 维度, 最大函数评价次数, 预定周期).
01 由式(1)产生初始化种群, 并计算其目标函数值.
02 while (FES < max.FES)
03　对个体目标函数值进行排序, 找到精英解.
04　根据式 (8) 计算每个个体的邻域距离.
05　for $i = 1$ to SN
06　　根据式(5)或(2)产生候选解 v_i.
07　　if $f(v_i) < f(x_i)$
08　　$x_i = v_i$; $\mathrm{counter}(i) = 0$
09　　else
10　　$\mathrm{counter}(i) = \mathrm{counter}(i) + 1$
11　　end if
12　end for
13　for $i = 1$ to SN
14　　随机从精英解中选择 x_e
15　　根据式(6)或(2)产生候选解 v_e.
16　　if $f(v_e) < f(x_e)$
17　　$x_e = v_e$; $\mathrm{counter}(e) = 0$
18　　else
19　　$\mathrm{counter}(e) = \mathrm{counter}(e) + 1$
20　　end if
21　end for
22　FES = FES + 2SN
23　if $\mathrm{counter(max)} > \mathrm{limit}$
24　　根据式(1)随机生成一个解替换 $x_{\max}$.
25　end if
26 end while

2.5　本文算法的时间复杂度分析

与ABC算法相比, 本文算法在选择精英解(对总体进行排序)和个体最佳邻居时引入了额外计算, 为此, 以下给出本文算法的时间复杂度分析. 在一次迭代过程中, 选择精英解(对总体进行排序)的时间复杂度为 $O(\mathrm{SN} \cdot \log(\mathrm{SN}))$. 由于ABC算法的复杂度为 $O(\mathrm{SN} \cdot D)$, 加上排序的整体复杂度为 $O(\mathrm{SN} \cdot \log(\mathrm{SN}) + \mathrm{SN} \cdot D)$. 当 D 远大于 $\log(\mathrm{SN})$ 时, 可以简化为 $O(\mathrm{SN} \cdot D)$. 此外, 群体中个体与其他个体之间距离计算的复杂度为 $O(\mathrm{SN}^2 \cdot D)$. 每个个体在自己的邻域范围内找到目标函数值最小的邻居, 该过程的复杂度为 $O(\mathrm{SN}^2 + \mathrm{SN})$. 所以本文算法的总体复杂度为 $O(\mathrm{SN}^2 \cdot D)$. 此外, 在函数评价代价高昂的情况下, 本文算法的计算负担主要由函数评价决定. Cai等[13]认为, 通过计算个体之间的成对距离所增加的计算成本对于计算代价高昂的函数而言是可以忽略不计的, 因此在函数评价成本较高的情况下, 引入的操作造成的额外开销相对较小.

3　仿真实验

为了测试本文算法的性能, 对文献[14]的22个基准测试函数进行仿真实验. 表1为每个函数的搜索范围、全局最优值和可接受值. 在Windows7 64位操作系统, Matlab R2017b环境下运行对比算法ABC[4]、GABC[9]、EABC[15]、CABC[16]. 为了公平比较, 所有算法独立运行25次, 所有比较算法的相关参数设置与原论文相同, 参数设置如表2所示, 并设置max.FES = $5\,000D$ 作为终止条件.

本文采用3个指标评价所提出算法的性能. Mean(std)表示25次独立实验的平均最佳目标函数值和相应的标准差, 用来评价不同算法得到的解的质量. AVEN记录了达到可接受值所需的函数评价的平均次数, 该值用于评估收敛速度, 如果算法在所有运行次数中无法找到目标函数值小于可接受值的任何解, 则AVEN将用NaN表示. 成功率(SR %)表示算法在25次独立实验中, 能够找到目标函数值小于可接受值的解的比例, 用于评价不同算法的鲁棒性.

为了测试本文算法的有效性, 在 $D = 30$、50、100的测试函数上进行仿真实验, 限于篇幅, 只给出 $D = 30$、100时部分函数的实验结果. 为了更清晰地表明收敛速度的变化, 图1绘制了 $D = 30$ 部分函

数的收敛曲线.

表 1 22 个基准测试函数

函数	名称	搜索空间	最优值	可接受值
f_1	Sphere	$[-100, 100]$	0	1×10^{-8}
f_2	Elliptic	$[-100, 100]$	0	1×10^{-8}
f_3	SumSquare	$[-10, 10]$	0	1×10^{-8}
f_4	sumPower	$[-1, 1]$	0	1×10^{-8}
f_5	Schwefel 2.22	$[-10, 10]$	0	1×10^{-8}
f_6	Schwefel 2.21	$[-100, 100]$	0	1×10^{0}
f_7	Step	$[-100, 100]$	0	1×10^{-8}
f_8	Exponential	$[-10, 10]$	0	1×10^{-8}
f_9	Quartic	$[-1.28, 1.28]$	0	1×10^{-1}
f_{10}	Rosenbrock	$[-5, 10]$	0	1×10^{-1}
f_{11}	Rastrigin	$[-5.12, 5.12]$	0	1×10^{-8}
f_{12}	NCRastrigin	$[-5.12, 5.12]$	0	1×10^{-8}
f_{13}	Griewank	$[-600, 600]$	0	1×10^{-8}
f_{14}	Schwefel 2.26	$[-500, 500]$	0	1×10^{-8}
f_{15}	Ackley	$[-50, 50]$	0	1×10^{-8}
f_{16}	Penalized1	$[-100, 100]$	0	1×10^{-8}
f_{17}	Penalized2	$[-100, 100]$	0	1×10^{-8}
f_{18}	Alpine	$[-10, 10]$	0	1×10^{-8}
f_{19}	Levy	$[-10, 10]$	0	1×10^{-8}
f_{20}	Weierstrass	$[-1, 1]$	0	1×10^{-8}
f_{21}	Himmelblau	$[-5, 5]$	−78.332 36	−78
f_{22}	Michalewicz	$[0, \pi]$	$-D$	$-D+1$

表 2 各种人工蜂群算法参数设置表

算法	参数设置
ABC	$\mathrm{SN}=50, \mathrm{limit}=\mathrm{SN}\cdot D$
GABC	$\mathrm{SN}=50, \mathrm{limit}=\mathrm{SN}\cdot D$
EABC	$\mathrm{SN}=50, \mathrm{limit}=\mathrm{SN}\cdot D, \mu=0.3, A=1, \delta=0.3$
CABC	$\mathrm{SN}=50, \mathrm{limit}=\mathrm{SN}\cdot D$
本文算法	$\mathrm{SN}=50, \mathrm{limit}=\mathrm{SN}\cdot D, p=0.1$

表3记录了 $D=100$ 时的实验结果,对于单模态函数 $f_1 \sim f_7$ 而言,除 f_7 外,本文算法的求解质量和收敛速度均优于其他比较算法. 由于 f_7 的实际全局最优解不是一个点,而是一个区域,使得 f_7 更容易求解. 所有算法均找到了最优解,但本文算法的收敛速度更快. 同样,对于 f_8,除EABC算法外,其他算法均得到了相同结果. 对于 f_9,所有算法都只能得到一个近似全局最优解,这是因为 f_9 是一个噪声4次函数,很难找到 f_9 的实际全局最优解. f_{10} 是Rosenbrock函数,具有多种特征,其全局最优解位于狭长的抛物线形平坦山谷内,变量之间具有很强的依赖性,梯度一般不指向最优解[17]. 因此,利用当前最优解的信息或其他好的信息很容易陷入局部最优区域,这便是本文算法在 f_{10} 中表现不佳的原因. 对于多模态函数 $f_{11} \sim f_{22}$,从收敛速度看,虽然EABC算法在 f_{14}、f_{20} 上的收敛速度比本文算法快一点,但成功率没有本文算法高;从解的准确性和鲁棒性来看,本文算法在多数测试函数上优于或至少不次于其他算法.

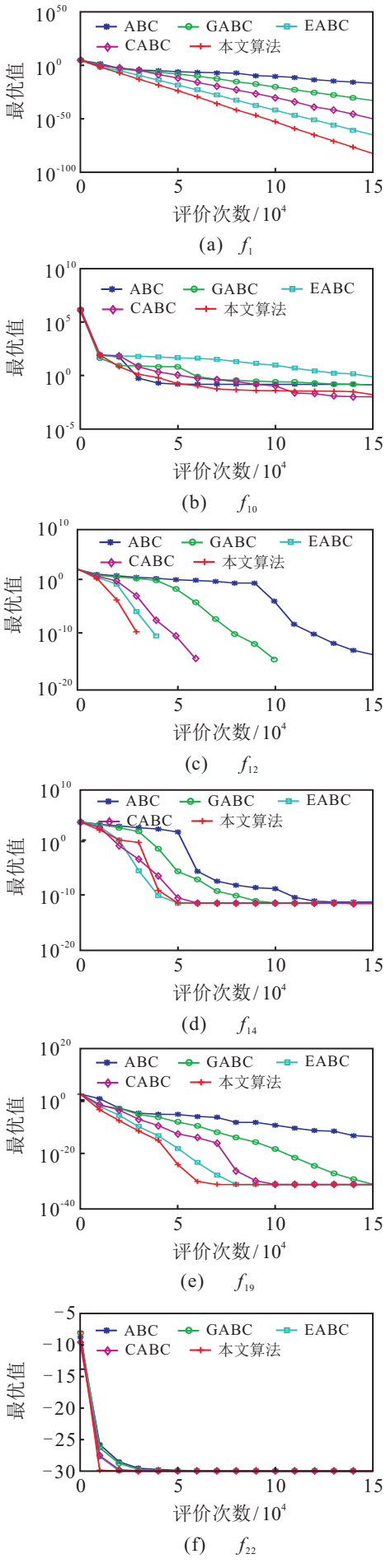

图 1 $D=30$ 时部分函数的收敛性能比较

表 3　22个100维基准测试函数的实验结果

Alg	metric	ABC	GABC	EABC	CABC	本文算法
f_1	Mean(std)	4.17e-16(3.28e-16)	1.91e-30(1.26e-30)	1.63e-59(9.00e-60)	6.85e-49(1.43e-48)	**8.66e-81(1.34e-80)**
	SR(AVEN)	**100**(293 090)	**100**(178 938)	**100**(103 518)	**100**(142 978)	**100(75 730)**
f_2	Mean(std)	5.63e-09(5.47e-09)	7.44e-24(3.54e-24)	3.56e-55(4.94e-55)	3.77e-39(1.35e-38)	**2.31e-77(2.67e-77)**
	SR(AVEN)	84(477 721)	**100**(269 998)	**100**(143 454)	**100**(223 362)	**100(100 170)**
f_3	Mean(std)	8.77e-17(6.90e-17)	6.68e-31(3.09e-31)	6.45e-60(4.14e-60)	1.81e-49(1.62e-49)	**2.59e-81(5.48e-81)**
	SR(AVEN)	**100**(279 570)	**100**(175 022)	**100**(100 738)	**100**(138 914)	**100(73 782)**
f_4	Mean(std)	2.98e-31(9.47e-31)	1.29e-53(5.18e-53)	3.21e-30(1.13e-29)	7.34e-60(3.41e-59)	**2.59e-110(1.29e-109)**
	SR(AVEN)	**100**(74 414)	**100**(44 434)	**100**(28 794)	**100**(41 750)	**100(21 582)**
f_5	Mean(std)	1.28e-09(3.49e-10)	2.10e-16(3.61e-17)	4.05e-31(7.44e-31)	1.29e-25(4.04e-26)	**1.15e-41(6.84e-42)**
	SR(AVEN)	**100**(464 842)	**100**(288 274)	**100**(160 754)	**100**(203 078)	**100(118 426)**
f_6	Mean(std)	2.77e+01(1.38e+00)	2.18e+01(3.35e+00)	4.84e+01(4.15e+00)	3.84e+00(3.44e-01)	**2.74e-02(6.14e-03)**
	SR(AVEN)	0(NaN)	0(NaN)	0(NaN)	0(NaN)	**100(269 326)**
f_7	Mean(std)	**0.00e+00(0.00e+00)**	**0.00e+00(0.00e+00)**	**0.00e+00(0.00e+00)**	**0.00e+00(0.00e+00)**	**0.00e+00(0.00e+00)**
	SR(AVEN)	**100**(52 470)	**100**(45 278)	**100**(30 826)	**100**(34 138)	**100(28 358)**
f_8	Mean(std)	**7.12e-218(0.00e+00)**	**7.12e-218(0.00e+00)**	1.00e-217(0.00e+00)	**7.12e-218(0.00e+00)**	**7.12e-218(0.00e+00)**
	SR(AVEN)	**100(150)**	**100(150)**	**100(150)**	**100(150)**	**100(150)**
f_9	Mean(std)	2.91e-01(4.01e-02)	1.61e-01(1.77e-02)	9.06e-02(1.27e-02)	1.07e-01(1.03e-02)	**5.04e-02(5.48e-03)**
	SR(AVEN)	0(NaN)	0(NaN)	72(419 400)	20(443 790)	**100(240 074)**
f_{10}	Mean(std)	**1.08e-01(9.79e-02)**	1.71e+01(3.19e+01)	7.15e-01(1.77e+00)	1.54e-01(1.44e-01)	3.69e+00(1.44e+01)
	SR(AVEN)	**60**(352 657)	36(388 883)	28(298 079)	40(343 890)	48(**225 692**)
f_{11}	Mean(std)	3.04e-12(1.52e-11)	**0.00e+00(0.00e+00)**	7.96e-02(2.75e-01)	**0.00e+00(0.00e+00)**	5.08e-02(2.04e-01)
	SR(AVEN)	**100**(399 922)	**100**(282 398)	92(131 863)	**100**(157 098)	92(**120 276**)
f_{12}	Mean(std)	5.13e-02(2.05e-01)	**0.00e+00(0.00e+00)**	1.20e-01(3.32e-01)	4.00e-02(2.00e-01)	**0.00e+00(0.00e+00)**
	SR(AVEN)	80(449 100)	**100**(314 842)	84(184 984)	96(183 075)	**100(131 998)**
f_{13}	Mean(std)	1.15e-16(3.12e-16)	**0.00e+00(0.00e+00)**	2.25e-13(1.12e-12)	2.66e-17(1.33e-16)	**0.00e+00(0.00e+00)**
	SR(AVEN)	**100**(291 538)	**100**(195 134)	**100**(109 406)	**100**(150 818)	**100(91 046)**
f_{14}	Mean(std)	6.69e-11(5.94e-12)	4.66e-11(6.30e-12)	3.79e+01(7.43e+01)	**3.41e-11(4.05e-12)**	1.42e+01(3.93e+01)
	SR(AVEN)	**100**(307 226)	**100**(266 386)	76(**154 418**)	**100**(167 930)	88(172 932)
f_{15}	Mean(std)	1.28e-08(4.23e-09)	1.11e-13(1.24e-14)	4.80e-14(1.11e-14)	1.41e-14(7.11e-16)	**1.36e-14(1.68e-15)**
	SR(AVEN)	36(496 128)	**100**(326 006)	**100**(185 154)	**100**(211 510)	**100(128 914)**
f_{16}	Mean(std)	2.66e-17(3.13e-17)	4.76e-32(1.90e-32)	**4.71e-33(6.98e-49)**	**4.71e-33(6.98e-49)**	**4.71e-33(6.98e-49)**
	SR(AVEN)	**100**(253 770)	**100**(155 050)	**100**(89 338)	**100**(128 090)	**100(63 042)**
f_{17}	Mean(std)	4.52e-15(3.06e-15)	5.53e-30(2.60e-30)	**1.50e-33(0.00e+00)**	**1.50e-33(0.00e+00)**	**1.50e-33(0.00e+00)**
	SR(AVEN)	**100**(319 730)	**100**(185 790)	**100**(104 702)	**100**(151 442)	**100(73 522)**
f_{18}	Mean(std)	4.29e-04(5.98e-04)	5.17e-05(6.74e-05)	3.95e-16(5.48e-16)	**7.78e-27(5.67e-27)**	1.18e-16(2.41e-16)
	SR(AVEN)	0(NaN)	0(NaN)	**100(162 714)**	**100**(191 214)	**100**(177 430)
f_{19}	Mean(std)	6.75e-14(1.03e-13)	5.55e-31(2.46e-31)	**1.35e-31(2.23e-47)**	**1.35e-31(2.23e-47)**	**1.35e-31(2.23e-47)**
	SR(AVEN)	**100**(305 010)	**100**(179 426)	**100**(97 646)	**100**(134 782)	**100(78 346)**
f_{20}	Mean(std)	5.40e-01(1.49e-01)	1.80e-01(8.96e-02)	7.23e-03(1.65e-02)	**0.00e+00(0.00e+00)**	**0.00e+00(0.00e+00)**
	SR(AVEN)	0(NaN)	0(NaN)	36(**254 872**)	**100**(291 370)	**100**(274 950)
f_{21}	Mean(std)	−78.332(2.15e-14)	−78.332(8.70e-15)	−78.332(2.30e-14)	**−78.332(0.00e+00)**	**−78.332(0.00e+00)**
	SR(AVEN)	**100**(113 934)	**100**(69 566)	**100**(32 870)	**100**(34 250)	**100(27 386)**
f_{22}	Mean(std)	−99.964(6.32e-03)	−99.962(8.95e-03)	−100.000(2.61e-05)	**−100.000(1.08e-08)**	−100.000(1.45e-04)
	SR(AVEN)	**100**(156 814)	**100**(143 646)	**100**(77 918)	**100**(86 826)	**100(31 934)**

同时由表3可见,随着维数的增加,各种算法都出现不同程度的问题,不过本文算法仍然在大部分函数上优于对比算法. 特别是对于f_6和f_9而言,当$D=100$时,对比算法f_6的成功率为0%,本文算法的成功率为100%;对比算法f_9的成功率较低,而本文算法的成功率达100%. f_9是一个具有噪声的4次函数,表明本文算法对f_9的鲁棒性更好. 不过本文算法对f_{14}不太适用,随着维数的增加,成功率在下降,由$D=30$时的100%下降到$D=100$时的88%. 对于f_{10},之前提到的特征导致其他算法以及本文算法的鲁棒性没有ABC算法好. 对于部分函数,由于最大函数评价次数取的较大,即迭代次数较大,本文算法与对比算法得到相同的平均值,到达一个稳定的状态,但是本文算法的收敛速度更快.

整体而言,从所有实验结果看,各种算法均有优缺点. 数值结果表明,本文算法在单模态函数上的性能具有显著优势,但在个别多模态函数上的性能不明显. 如f_{10}这样的特殊函数,因为GABC、EABC和本文算法的搜索方程都利用了当前最优解的信息,所以它们的结果没有ABC和CABC的结果好. 对于其他个别多模态函数,在25次独立实验中由于初始点的随机选取,迭代后进入一个希望不大的区域,最终结果可能会出现一两个异常值,从而导致平均最佳目标函数值变大.

4 结 论

本文针对ABC算法存在的问题,分别基于雇佣蜂和观察蜂提出两个搜索方程,从而改善了ABC算法的局部搜索能力和收敛速度. 实验结果表明,所提出的基于精英解和随机个体邻域信息的改进人工蜂群算法比对比算法的寻优性能有明显提高. 下一步将对ABC算法参数设置理论及文中的不足之处进行进一步的研究.

参考文献(References)

[1] Wei Y H, Xu C, Hu Q Y. Transformation of optimization problems in revenue management, queueing system, and supply chain management[J]. International Journal of Production Economics, 2015, 146(2): 588-597.

[2] Li G H, Lin Q Z, Gui L Z, et al. A novel hybrid differential evolution algorithm with modified CoDE and JADE[J]. Applied Soft Computing, 2016, 47: 577-599.

[3] Li M, Lu Y M. Hybrid multipopulation cellular geneticalgorithm and its performance[J]. Transaction of Nanjing University Astronautics, 2014, 31(4): 405-412.

[4] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Kayseri: Engineering Faculty Computer Engineering Department, Ereiyes University, 2005.

[5] Chen Y G, Li L X, Peng H P, et al. Particle swarm optimizer with two differential mutation[J]. Applied Soft Computing, 2017, 61: 314-330.

[6] Xiang Y, Zhou Y R, Liu H L. An elitism based multi-objective artificial bee colony algorithm[J]. European Journal of Operational Research, 2015, 245(1): 168-193.

[7] Ozturk C, Hancer E, Karaboga D. A novel binary artificial bee colony algorithm based on genetic operators[J]. Information Sciences, 2015, 297: 154-170.

[8] Yan X H, Zhu Y L, Zou W P, et al. A new approach for data clustering using hybrid artificial bee colony algorithm[J]. Neurocomputing, 2012, 97: 241-250.

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

[10] Gao W F, Chan F T S, Huang L L, et al. Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood[J]. Information Sciences, 2015, 316: 180-200.

[11] Cui L Z, Li G H, Lin Q Z, et al. A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation[J]. Information Sciences, 2016(367/368): 1012-1044.

[12] Lin Q Z, Zhu M M, Li G H, et al. A novel artificial bee colony algorithm with local and global information interaction[J]. Applied Soft Computing, 2018, 62: 702-735.

[13] Cai Y Q, Wang J H. Differential evolution with neighborhood and direction information for numerical optimization[J]. IEEE Transactions on Cybernetics, 2013, 43(6): 2202-2215.

[14] Gao W F, Huang L L, Liu S Y, et al. Artificial bee colony algorithm based on information learning[J]. IEEE Transactions on Cybernetics. 2015, 45(12): 2827-2839.

[15] Gao W F, Liu S Y, Huang L L. Enhancing artificial bee colony algorithm using more information-based search equations[J]. Information Sciences, 2014, 270: 112-133.

[16] Gao W F, Liu S Y, Huang L L. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning[J]. IEEE Transactions on Cybernetics, 2013, 43(3): 1011-1024.

[17] Shang Y W, Qiu Y H. A note on the extended Rosenbrock function[J]. Evolutionary Computation, 2006, 14: 119-126.

作者简介

孟红云(1975−),女,副教授,博士,从事最优化方法、智能信息处理等研究, E-mail: menghy@xidian.edu.cn;

位冰可(1994−),女,硕士生,从事进化计算、最优化方法的研究, E-mail: 2411369389@qq.com.

(责任编辑: 郑晓蕾)