文章编号:1009-3087(2016)05-0124-11 DOI:10.15961/j.jsuese.2016.05.018

精英区域学习的转轴人工蜂群算法

熊小峰[1],尹雅丽[1],郭肇禄[1*],吴志健[2]
(1.江西理工大学 理学院,江西 赣州,341000
2.武汉大学 计算机学院 软件工程国家重点实验室,湖北 武汉,430072)

摘 要:针对人工蜂群(ABC)算法在解决复杂优化问题时容易出现收敛速度慢、开采能力不足的问题,提出了一种精英区域学习的转轴人工蜂群(ERABC)算法。在ERABC算法中,通过执行区域学习方法构建精英池,并利用精英池改进其搜索策略,同时在每一代中以一定的频率对最优解执行转轴法(RM)局部搜索。在20个包含单峰、多峰和偏移函数的基准测试函数上,分析了ERABC算法中改进策略的有效性,并与多种新近的改进ABC算法和演化算法进行了比较实验。实验结果表明,提出的算法在保证精英池中个体多样性的同时加快了算法的收敛速度,RM有效地提高了算法的开采能力。

关键词:人工蜂群算法;精英区域学习;搜索策略;转轴法
中图分类号:TP391 **文献标志码**:A

Rosenbrock Artificial Bee Colony with Elite Region Learning

XIONG Xiaofeng[1], *YIN Yali*[1], *GUO Zhaolu*[1*], *WU Zhijian*[2]
(1. School of Sci., Jiangxi Univ. of Sci. and Technol., Ganzhou 341000, China;
2. State Key Lab. of Software Eng., Computer School, Wuhan Univ., Wuhan 430072, China)

Abstract: In order to enhance the exploitation ability of the basic artificial bee colony (ABC), a rosenbrock ABC with elite region learning (ERABC) was proposed. The proposed ERABC utilized an enhanced search strategy with elite region learning to maintain the population diversity. Moreover, the rosenbrock's rotational direction method was employed to improve the exploitation ability. The proposed ERABC was tested on 20 benchmark functions including unimodal, multimodal, and shifted functions. The effects of the improved strategy in ERABC were experimentally investigated. Furthermore, ERABC was compared with some state-of-the-art ABC variants and several related evolutionary algorithms. The experimental results indicated that ERABC enhances the convergence speed and exploitation ability.

Key words: artificial bee colony; elite region learning; search strategy; rosenbrock method

人工蜂群(artificial bee colony, ABC)算法[1]是由土耳其学者Karaboga于2005年提出的一种群智能优化算法,它通过模拟蜜蜂的智能觅食机制来寻找最优解。自从ABC算法提出以来,许多研究人员对其进行了研究,并与PSO、DE等演化算法进行了比较,实验表明ABC算法具有较好的全局寻优能力,是求解高维、多峰问题的一种有效方法[2]。由于ABC算法具有操作简单、易于实现、控制参数少等优点,因此ABC及其改进算法已广泛应用于多个领域[3]。在ABC算法中,雇佣蜂和观察蜂共用一个解搜索方程,算法的性能主要依赖该搜索方程,这种解搜索方程侧重于算法的勘探能力,往往开采能力不足[3]。与文献[4-6]演化算法类似,在解决复杂多峰问题时,ABC算法容易出现局部搜索能力不足、收敛速度慢等缺点。针对这些缺点,有学者利用全局最优解、精英个体所携带的丰富信息对算法进行了改进[7-12],这些改进策略提高了算法的收敛速度和开采能力,但求解复杂多峰问题时容易使算

收稿日期:2015-10-27
基金项目:国家自然科学基金资助项目(61662029;11401267;11461032)
作者简介:熊小峰(1965—),男,教授. 研究方向:建模与应用数理统计等. E-mail:xxf_gz@163.com
* 通信联系人 E-mail:gzl@whu.edu.cn
网络出版时间:2016-9-19 9:43:59 **网络出版地址**:http://www.cnki.net/kcms/detail/51.1596.T.20160919.0943.004.html

法陷入局部最优。

为了有效利用精英信息,在提高 ABC 算法的收敛速度和开采能力的同时尽可能避免算法陷入局部最优,提出了一种精英区域学习的转轴人工蜂群(rosenbrock artificial bee colony with elite region learning algorithm,ERABC)算法。该算法提出了一种精英区域学习搜索策略,该策略首先利用正弦函数进行区域学习构建精英池,这种精英池产生方法保证了个体的多样性,可以在一定程度上避免陷入局部最优,同时又为解搜索方程提供精英信息,然后在每一代中以一定的频率利用转轴法[13](rosenbrock method,RM)进行局部搜索。

1　人工蜂群算法(ABC)

ABC 算法通过模拟蜜蜂的采蜜群体行为来求解优化问题,它将蜂群划分为雇佣蜂、观察蜂、侦查蜂 3 类进行进化,且雇佣蜂、观察蜂、蜜源数量相等,蜜源对应优化问题的候选解,蜜源质量即对应优化问题的适应度值[1]。

设种群规模为 NP,待优化问题维度为 D,每个蜜源表示为 $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$,其中,$i = 1,2,\cdots,NP$。在算法的初始阶段,通过式(1)产生含 NP 个蜜源的初始种群[1]:

$$x_{ij} = x_j^{\min} + rand \cdot (x_j^{\max} - x_j^{\min}) \tag{1}$$

其中,j 为 $\{1,2,\cdots,D\}$ 中的随机数,$x_j^{\min}$、$x_j^{\max}$ 分别表示解空间在第 j 维的下界和上界,$rand \in [0,1]$ 为 0 ~ 1 之间服从均匀分布的随机数。

在解决最小值优化问题时,产生初始种群后,通过式(2)计算蜜源质量[1]:

$$fit_i = \begin{cases} \dfrac{1}{1+f_i}, f_i \geq 0; \\ 1 + |f_i|, f_i < 0 \end{cases} \tag{2}$$

其中:fit_i 为蜜源 i 的适应值,适应值越大表示该蜜源质量越优;f_i 为蜜源 i 的目标函数值。

种群初始化后,依次执行雇佣蜂、观察蜂、侦查蜂操作算子使种群不断向最优解逼近,各操作算子描述如下:

1)雇佣蜂操作算子。该阶段雇佣蜂根据式(3)对每个蜜源进行一次邻域搜索,产生新蜜源 $\boldsymbol{V}_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$[1]:

$$v_{ij} = x_{ij} + \varphi_{ij} \cdot (x_{ij} - x_{kj}) \tag{3}$$

其中,$\varphi_{ij} \in [-1,1]$,$k \in \{1,2,\cdots,NP\}$ 且 $k \neq i$,j 为 $\{1,2,\cdots,D\}$ 中的随机数。在每次邻域搜索过程中随机更新一个维度上的数值,然后计算新蜜源 $\boldsymbol{V}_i$ 的质量,若新蜜源 $\boldsymbol{V}_i$ 质量更优,则用其替换蜜源 $\boldsymbol{X}_i$。

2)观察蜂操作算子。执行完雇佣蜂操作算子后,ABC 算法开始执行观察蜂操作算子。在观察蜂阶段,观察蜂根据雇佣蜂共享的蜜源信息按贪婪法选择蜜源进行开采,每个蜜源的选择概率计算方式如下[1]:

$$P_i = \frac{fit_i}{\sum_{i=1}^{NP} fit_i} \tag{4}$$

其中,P_i 为第 i 个蜜源的选择概率。当蜜源被选中后,将按式(3)对所选中的蜜源进行邻域搜索。

3)侦查蜂操作算子。在侦查蜂操作过程中,选择出未被改进次数最大的蜜源,当该蜜源未被改进次数大于事先给定的限制次数(*limit*),此蜜源将按式(1)重新随机产生。

2　ERABC 算法

为了提高 ABC 算法的性能,ERABC 算法首先针对种群中的个体进行区域学习构建精英池。一方面,精英池相比单一精英个体更具多样性;另一方面,利用区域学习的方式比将历史最优个体直接推入精英池的构建方法更能保障个体的多样性。然后在每一代中以一定的频率利用 RM 进行局部搜索,提高算法的开采能力。这些机制使得 ERABC 算法在保证精英池信息多样性的同时又为解搜索方程提供精英信息,可以有效加快算法的收敛速度,引入 RM 局部搜索可以有效提高算法的开采能力。

2.1　精英区域学习搜索策略

在 ABC 算法中,利用最优个体或精英信息可以加快收敛速度,但在解决复杂多峰问题时,最优解或单一精英信息容易使算法陷入局部最优,而利用精英池策略能在一定程度上避免这一问题。精英池的产生一般为将历史最优个体推入其中,而当前代最优个体往往是在上一代最优个体基础上更新得到的,会导致精英池中个体相近,降低精英池中个体多样性。为提高精英池中个体的多样性,采用反向学习(opposition-based learning,OBL)策略[14]来构建精英池。OBL 在搜索过程中,反向向量所对应的反向搜索区域为 $[\boldsymbol{L},\boldsymbol{U}]$,$\boldsymbol{L}$、$\boldsymbol{U}$ 分别表示当前种群中所有个体在各维度上的最小值和最大值,根据种群中的个体动态调整,虽然用动态搜索边界代替固定搜索边界保存搜索经验可以在一定程度上增强个体的多样性,但反向搜索只是在区域 $[\boldsymbol{L},\boldsymbol{U}]$ 内进行,使当前向量 $\boldsymbol{X}$ 至反向向量 $\boldsymbol{X}^*$ 的区域 $[\boldsymbol{X},\boldsymbol{X}^*]$ 被忽

略[15]。引入三角函数进行变异的区域学习策略[15]可以使搜索区域从$[\boldsymbol{L},\boldsymbol{U}]$扩展至$[\boldsymbol{X}+\boldsymbol{L},\boldsymbol{X}^*+\boldsymbol{U}]$。相比OBL,区域学习策略可以更好保证精英池中个体多样性,为解搜索方程提供精英信息的同时在一定程度上增强算法避免陷入局部最优的能力。

受彭虎等[15]提出的ELDDE算法中区域学习变异解启发,提出了精英区域学习搜索策略。策略中首先对种群中个体进行区域学习,学习方法如式(5)所示:

$$\boldsymbol{X}^* = \sin(\eta)\cdot\boldsymbol{X} + rand\cdot(\boldsymbol{U}+\boldsymbol{L}) \quad (5)$$

其中,$\boldsymbol{X}$为种群中个体,$\boldsymbol{X}^*$为个体对应的区域学习变异解,$\eta\in[-\pi,\pi]$为随机数,$\boldsymbol{L}$、$\boldsymbol{U}$分别为动态调整边界的下界和上界。然后根据种群中个体$\boldsymbol{X}$及其对应的区域学习变异解$\boldsymbol{X}^*$构建精英池$\boldsymbol{X}^e$。精英池的产生方式如算法1所示。

算法1 精英池产生算法

1)输入种群$\boldsymbol{X}$,精英池大小NE;

2)根据种群$\boldsymbol{X}$计算上下界$\boldsymbol{U}$和$\boldsymbol{L}$;

3)根据式(5)产生种群$\boldsymbol{X}$的变异解$\boldsymbol{X}^*$;

4)从$\boldsymbol{X}$和$\boldsymbol{X}^*$中根据适应值选取前NE个个体作为精英池$\boldsymbol{X}^e$中个体。

采用正弦函数对种群个体进行区域学习,由种群个体和区域学习变异解构建精英池,保证了精英池中个体的多样性,可以在一定程度上避免算法陷入局部最优,为了利用这种多样性的精英信息加快算法收敛速度,提出精英区域学习解搜索方程,如式(6)所示:

$$v_{ij} = x_{ij} + \varphi_{ij}\cdot(x_{ij}-x_{kj}) + \psi_{ij}\cdot(x^e_{r1j}-x_{ij}) \quad (6)$$

其中:$\varphi_{ij}\in[-1,1]$;$\psi_{ij}\in[0,1.5]$为随机数;i、k为$\{1,2,\cdots,NP\}$中的任意数,且$k\neq i$;x^e_{r1}为精英池$\boldsymbol{X}^e$中的随机个体,$r1$为$\{1,2,\cdots,NE\}$中的随机数,NE为精英池大小。

2.2 转轴法局部搜索算子

虽然利用精英区域学习策略可以加快ABC算法的收敛速度及在一定程度上提高算法的开采能力,但在求解一些复杂工程优化问题时,改进的ABC算法仍表现出开采能力不足的问题。为了进一步提高其开采能力,利用RM[13]进行局部搜索是一种有效方法。在RM[13]中,以一定的精度和更新次数限定搜索方向上个体的更新,当达不到更新条件时将改变搜索方向继续进行搜索,随着更新次数的增加,最小步长$\delta_{\min}$的限制条件也将作出动态调整。RM[13]算法流程如算法2所示。其中,d为搜索方向,δ_1、δ_2、$\cdots$、δ_D为每个维度上的搜索步长,α、β为步长调整因子,终止条件a、n_l、ε_1、ε_2均为常数,n_{RM}表示调用RM的次数[13]。

算法2 RM[13]算法

1. 输入初始点x^0,初始搜索方向d_j,初始更新步长δ_j,步长调整因子α、β,终止条件a、n_l、ε_1、ε_2,方向更新次数$k=0$、$k_2=0$;

2. while $k_2<2n_l$ and $\delta_{\min}\geq 1.0\mathrm{E}-(a+n_{\mathrm{RM}})$ do

3. 设置$x=x^k$,$k_1=0$,$z=x$;

4. while $k_1<n_l$ do

5. for $j=1$ to D do

6. $y=x+\delta_j d_j$;

7. if y 优于 x then 设置$x=y$,$\delta_j=\alpha\cdot\delta_j$; else $\delta_j=\beta\cdot\delta_j$;

8. end for

9. if $|f(z)-f(x)|/|f(x)+\varepsilon_1|<\varepsilon_2$ then 设置$k_2=k_2+1$; else $k_2=0$;

10. $k_1=k_1+1$;

11. end while

12. if $f(x)<f(x^k)$ and $\delta_{\min}\geq 1.0\mathrm{E}-(a+n_{\mathrm{RM}})$ then 设置$k=k+1$,$x^k=x$,更新搜索方向d;

13. end while

2.3 ERABC算法描述

将精英区域学习策略引入基本ABC算法的解搜索方程中,得到了精英区域学习人工蜂群(artificial bee colony with elite region learning algorithm, EABC)算法,该算法通过精英池产生算法1构建精英池,再利用式(6)进行解更新。相比ABC算法,基于精英区域学习策略的EABC算法利用了精英信息,可以在一定程度上提高算法的收敛速度和开采能力。为进一步提高算法的开采能力,在EABC算法的3个操作阶段完成后,针对最优个体引入RM[13]用于局部搜索,由此得到ERABC算法,其流程如算法3所示,其中,n_c为常数,一般设置为$n_c=k\cdot D$[13]。

算法3 提出的ERABC算法

1. 初始化种群$\boldsymbol{X}_i$,$i=1,2,\cdots,NP$,$iter=1$;

2. 计算各个体的函数值及适应值,保存最优个体x_{best};

3. 计算算法2中的初始步长δ_j;

4. 将x_{best}作为算法2的初始点调用算法2,获得最优个体x_{best}';

5. if $f(x_{\mathrm{best}}')<f(x_{\mathrm{best}})$ then 用x_{best}'替换种群中间位置的个体,$x_{\mathrm{best}}=x_{\mathrm{best}}'$;

6. while 终止条件不满足时 do

7. 调用算法 1 产生精英池 $\boldsymbol{X}^e$;
8. 雇佣蜂阶段:按式(6)对蜜源进行更新、计算函数值和适应值;
9. 按式(4)计算选择概率 P_i;
10. 观察蜂阶段:按贪婪选择机制选择蜜源并按式(6)进行更新;
11. 侦查蜂阶段:对于要丢弃的蜜源,按式(1)重新产生;
12. 保存当前最优个体 x_{best};
13. if mod($iter$,n_c) == 0 then
14. 计算步长 δ_j,并将 x_{best} 作为初始点调用算法 2;
15. if $f(x_{\text{best}}')$ < $f(x_{\text{best}})$ then 用 x_{best}' 替换种群中间位置的个体,x_{best} = x_{best}';
16. $iter$ = $iter$ + 1;
17. end while

2.4 算法复杂度分析

文献[16]中分析了 ABC 算法每一代的时间复杂度,为$O(NP\cdot D)$,采用与文献[16]相同的方法分析 ERABC 算法每一代的时间复杂度。设种群规模为NP,待优化问题维度为D。ERABC中,种群初始化、适应值计算及计算 RM 中初始步长的复杂度都为$O(NP\cdot D)$,RM[13] 算法的复杂度为$O(D^2)$,算法1的复杂度为$O(NP\cdot D)$,雇佣蜂操作和观察蜂操作的复杂度都为$O(NP\cdot D)$,计算选择概率的复杂度为$O(NP)$,侦查蜂操作的复杂度为$O(D)$。综上所述,ERABC 算法总的时间复杂度为$O(NP\cdot D)+O(D^2)$,与NP和D相关。相比 ABC 算法的时间复杂度,当$D\approx NP$或$D\ll NP$时 ERABC 算法的总时间复杂度与 ABC 的时间复杂度相当,当$D>NP$时,ERABC 算法的时间复杂度相比 ABC 稍高。

3 数值实验及分析

为了验证 ERABC 算法的有效性和适用性,选择函数优化领域中广泛采用的 20 个基准测试函数进行实验分析。实验在 Pentium(R) Dual-Core CPU:E5400、4 GB 内存、2.70 GHz 主频计算机上用 Matlab7.10.0 语言实现。

3.1 实验函数及参数设置

实验中采用的各函数如表 1 所示。其中,f_1 ~ f_4 为单峰函数,f_5 ~ f_{13} 为多峰函数,f_{14} ~ f_{15} 为单峰偏移函数,f_{16} ~ f_{20} 为多峰偏移函数,各函数的具体定义见参考文献[17]。

表 1 基准测试函数

Tab. 1 Benchmark functions used in experiments

编号	函数名	取值范围	最优值	精度设置	编号	函数名	取值范围	最优值	精度设置
f_1	Sphere	$[-100,100]^D$	0	1E-08	f_{11}	Levy	$[-10,10]^D$	0	1E-08
f_2	SumSquare	$[-10,10]^D$	0	1E-08	f_{12}	Bohachevsky	$[-100,100]^D$	0	1E-08
f_3	SumPower	$[-1,1]^D$	0	1E-08	f_{13}	Rosenbrock	$[-5,10]^D$	0	5E+00
f_4	Step	$[-100,100]^D$	0	1E-08	f_{14}	Shifted Sphere	$[-100,100]^D$	0	1E-08
f_5	Schwefel2.22	$[-10,10]^D$	0	1E-08	f_{15}	Shifted Schwefel1.2	$[-100,100]^D$	0	1E-08
f_6	Griewank	$[-600,600]^D$	0	1E-08	f_{16}	Shifted Zakharov	$[-5,10]^D$	0	1E-08
f_7	Ackley	$[-32,32]^D$	0	1E-08	f_{17}	Shifted Griewank	$[-600,600]^D$	0	1E-08
f_8	Penalized1	$[-50,50]^D$	0	1E-08	f_{18}	Shifted Ackley	$[-32,32]^D$	0	1E-08
f_9	Penalized2	$[-50,50]^D$	0	1E-08	f_{19}	Shifted Penalized1	$[-50,50]^D$	0	1E-08
f_{10}	Alpine	$[-10,10]^D$	0	1E-08	f_{20}	Shifted Penalized2	$[-50,50]^D$	0	1E-08

为了公平地比较各算法的性能,实验的公共参数作如下统一设定:问题的维度分别为 D = 30、60,对应的最大评价次数分别为 max FES = 100 000、200 000,种群规模 NP = 50,控制参数 $limit$ = 200。其中,RM 部分参数采用文献[13]推荐的设置。

3.2 参数敏感性分析

EABC 算法和 ERABC 算法采用了精英区域学习解搜索方程进行个体更新,而精英池规模 NE 往往会影响算法的有效性,为分析其对算法的影响,各参数设置如第 3.1 节所述,D = 30,最大评价次数为 max FES = 100 000,对包含单峰、多峰、单峰偏移和多峰偏移的 5 个测试函数进行实验。将精英池规模 NE 分别设置为 1、$NP/5$、$2NP/5$、$NP/2$、$3NP/5$、$4NP/5$、NP、$6NP/5$ 时,进行测试,以各函数独立运行 50 次的均值作为统计测试函数性能的结果,解的精度随精英池大小变化结果如图 1 所示。

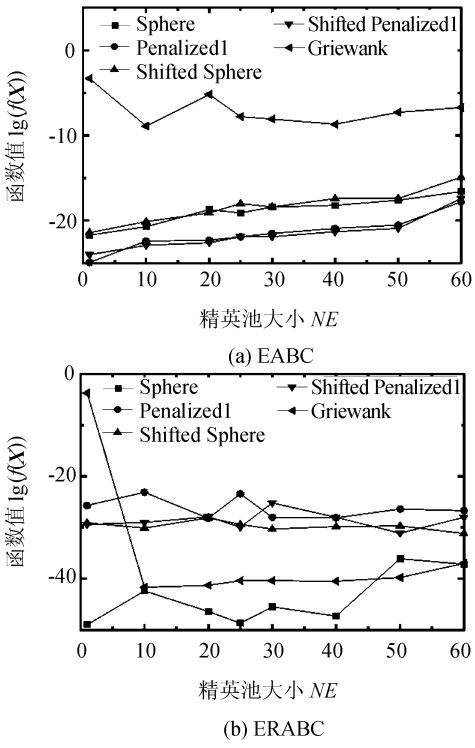

图1 $D=30$ 时 EABC 和 ERABC 算法的解的精度随精英池大小变化图

Fig. 1 Accuracy of EABC and ERABC algorithms varies with the elite pool size when $D=30$

从图1(a)中可以看出:EABC 算法在求解除 Griewank 函数外的其他4个函数时总体趋势为当精英池规模 NE 越小,其解的精度越高。对于函数 Griewank,当精英池规模 $NE=1$ 时,其解的精度最低,当 $NE=NP/5$ 时,解的精度最高,此后当精英池规模不断增加时,解的精度变化不大。

从图1(b)中可以看出:ERABC 算法在求解5个函数时各函数的精度相比 EABC 都得到了很大提高,对于 Penalized1、Shifted Sphere 及 Shifted Penalized1 函数,解的精度都在一定范围内变化,精英池规模对解的精度没有明显影响;求解 Sphere 函数时,$NE\in[1,4NP/5]$ 为解的精度较高区域,当 $NE>4NP/5$ 时精度有所降低;求解 Griewank 函数时,$NE=1$ 与 $NE>1$ 时解的精度相差很大,从 $NE=NP/5$ 开始,随着精英池规模增大,ERABC 的解的精度逐渐降低,但降幅不明显。

ERABC 采用 RM[13]进行局部搜索有效地提高了算法的开采能力,此时的精英区域学习策略更应该注重避免算法容易陷入局部最优的问题,所以此时精英池规模不可太小,而太大又不能有效利用精英信息。综上所述,当EABC和ERABC算法精英池规模 $NE\in[NP/5,3NP/5]$ 时算法表现较佳,综合分析选取精英池规模 $NE=2NP/5$ 用于算法性能测试。

3.3 策略有效性分析

为分析 ERABC 算法的性能,现将 ERABC 与 ABC[1]、EABC 和 RABC[13]算法进行比较。其中,EABC 算法为将精英区域学习策略引入 ABC 算法所提出的改进算法,RABC[13]算法是在 ABC 的基础上针对当前最优个体引入 RM 局部搜索所提出的改进算法。对比结果如表2所示,其中,括号内外数值分别为独立实验50次的标准差(*Std*)和均值(*Mean*)。为了从统计学意义上分析 ERABC 算法的有效性,对结果进行显著性水平为 $p=0.05$ 的非参数 Wilcoxon 假设检验。其中,符号"+"表示 ERABC 显著优于对应算法,"-"表示对应算法显著优于 ERABC,"≈"表示 ERABC 与对应算法无显著性差异。

从表2可以看出:当 $D=30$ 时,ERABC 算法在19个测试函数上得到了优于 ABC 的结果,解的精度提高了6~34个数量级,在 Step 函数上取得了与 ABC 算法相当的结果;ERABC 算法在19个测试函数上取得了优于 EABC 算法的结果,解的精度提高了1~34个数量级,在 Step 函数上取得了与 EABC 算法相当的结果;ERABC 算法在16个测试函数上取得了优于 RABC 算法的结果,解的精度提高了2~38个数量级,在 Shifted Sphere 函数上表现稍逊于 RABC 算法,在 Step、Alpine、Shifted Zakharov 3个函数上取得了与 RABC 算法相当的结果。

当 $D=60$ 时,ERABC 算法在19个测试函数上得到了优于 ABC 的结果,解的精度提高了7~36个数量级,在 Step 函数上取得了与 ABC 算法相当的结果;ERABC 算法在19个测试函数上取得了优于 EABC 算法的结果,解的精度提高了2~33个数量级,在 Step 函数上取得了与 EABC 算法相当的结果;ERABC 算法在17个测试函数上取得了优于 RABC 算法的结果,解的精度提高了2~34个数量级,在 Shifted Zakharov 函数上表现稍逊于 RABC 算法,在 Step、Alpine 2个函数上取得了与 RABC 算法相当的结果。从统计意义上表明了 ERABC 算法的性能得到了提高。

为从统计意义上比较4种算法的性能,采用非参数的 Friedman 检验[18]统计4种算法在 $D=30$ 和 $D=60$ 两种维度下的 Friedman 排名。4种算法的 Friedman 排名情况如表3所示。

表 2　$D=30$、60 时,ABC、EABC、RABC 和 ERABC 算法结果

Tab. 2　Results of ABC, EABC, RABC and ERABC algorithms when $D=30$, 60

编号	$D=30$				$D=60$			
	ABC *Mean*(*Std*)	EABC *Mean*(*Std*)	RABC *Mean*(*Std*)	ERABC *Mean*(*Std*)	ABC *Mean*(*Std*)	EABC *Mean*(*Std*)	RABC *Mean*(*Std*)	ERABC *Mean*(*Std*)
f_1	4.82E-10 (2.75E-09)+	9.80E-18 (5.01E-17)+	1.14E-33 (2.00E-32)+	9.23E-41 (3.17E-39)	1.24E-09 (6.29E-09)+	9.81E-17 (4.88E-16)+	1.20E-33 (2.49E-32)+	3.13E-40 (1.05E-38)
f_2	4.42E-11 (3.54E-10)+	2.05E-19 (1.13E-18)+	1.30E-34 (1.92E-33)+	2.20E-41 (4.60E-40)	3.18E-10 (1.87E-09)+	9.21E-18 (4.28E-17)+	1.18E-33 (3.60E-32)+	6.27E-40 (2.25E-38)
f_3	5.45E-17 (5.68E-16)+	1.67E-35 (4.46E-34)+	2.25E-40 (4.97E-39)+	2.06E-47 (1.01E-45)	1.03E-13 (2.96E-12)+	1.01E-33 (4.08E-32)+	1.01E-37 (3.53E-36)+	7.88E-48 (3.90E-46)
f_4	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)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)
f_5	1.79E-06 (5.54E-06)+	1.46E-10 (2.80E-10)+	2.31E-18 (3.93E-17)+	9.99E-22 (1.08E-20)	5.84E-06 (1.35E-05)+	5.06E-10 (9.81E-10)+	8.89E-18 (1.85E-16)+	3.21E-21 (4.95E-20)
f_6	3.19E-08 (5.29E-07)+	6.28E-08 (3.08E-06)+	1.74E-04 (7.77E-03)+	2.43E-42 (7.64E-41)	4.68E-06 (2.31E-04)+	4.51E-09 (2.22E-07)+	1.95E-08 (6.49E-07)+	1.08E-42 (2.69E-41)
f_7	1.01E-05 (3.22E-05)+	2.38E-10 (4.97E-10)+	1.00E-05 (2.79E-05)+	5.82E-14 (7.28E-14)	1.84E-05 (3.85E-05)+	5.15E-10 (9.38E-10)+	1.94E-05 (5.08E-05)+	1.18E-13 (1.01E-13)
f_8	3.65E-12 (2.30E-11)+	6.75E-21 (2.72E-20)+	3.47E-12 (5.28E-11)+	7.19E-24 (2.04E-22)	6.62E-12 (4.05E-11)+	1.19E-20 (4.34E-20)+	3.59E-12 (3.25E-11)+	3.61E-23 (7.66E-22)
f_9	3.16E-10 (1.75E-09)+	7.33E-19 (4.17E-18)+	1.59E-10 (4.69E-09)+	5.52E-25 (2.32E-23)	1.42E-09 (6.95E-09)+	5.19E-18 (2.77E-17)+	3.35E-10 (4.69E-09)+	7.69E-23 (2.14E-21)
f_{10}	1.86E-04 (1.12E-03)+	5.29E-05 (1.59E-04)+	5.80E-14 (6.87E-13)≈	5.64E-14 (4.84E-13)	3.84E-03 (2.96E-02)+	1.86E-04 (3.34E-04)+	2.33E-13 (2.95E-12)≈	1.70E-13 (3.44E-12)
f_{11}	4.84E-09 (3.90E-08)+	2.75E-15 (3.64E-14)+	4.60E-09 (6.66E-08)+	2.22E-20 (7.08E-19)	6.20E-09 (4.75E-08)+	4.76E-15 (5.01E-14)+	6.93E-09 (8.94E-08)+	5.36E-21 (2.58E-19)
f_{12}	3.77E-07 (3.41E-06)+	3.82E-14 (2.73E-13)+	1.00E-05 (2.79E-05)+	1.11E-18 (5.50E-17)	2.91E-07 (1.43E-06)+	4.02E-13 (2.52E-12)+	3.82E-07 (4.09E-06)+	0.00E+00 (0.00E+00)
f_{13}	3.05E-01 (2.44E+00)+	2.97E-01 (1.32E+00)+	4.22E-02 (8.45E-01)+	1.31E-20 (4.23E-19)	3.94E-01 (2.79E+00)+	1.23E+00 (8.90E+00)+	7.95E-02 (1.39E+00)+	2.81E-21 (1.37E-20)
f_{14}	5.37E-10 (4.20E-09)+	9.63E-18 (5.92E-17)+	3.55E-32 (5.08E-31)-	2.03E-27 (5.81E-26)	1.25E-09 (6.00E-09)+	8.87E-17 (3.74E-16)+	2.23E-24 (1.08E-22)+	3.03E-26 (9.01E-25)
f_{15}	1.06E+04 (1.60E+04)+	1.07E+04 (1.58E+04)+	9.21E-21 (2.12E-19)+	6.00E-23 (1.47E-21)	4.60E+04 (3.96E+04)+	4.67E+04 (4.70E+04)+	2.34E-19 (2.69E-18)+	5.02E-21 (1.57E-19)
f_{16}	2.82E+02 (2.48E+02)+	2.84E+02 (2.63E+02)+	1.70E-21 (1.96E-20)≈	8.96E-20 (2.52E-18)	7.63E+02 (3.69E+02)+	7.60E+02 (2.97E+02)+	5.71E-19 (4.97E-18)-	3.52E-18 (6.83E-17)
f_{17}	1.38E-07 (3.23E-06)+	1.26E-09 (2.91E-08)+	5.68E-05 (2.78E-03)+	4.97E-29 (2.43E-27)	6.25E-09 (1.03E-07)+	1.34E-10 (6.50E-09)+	2.70E-09 (4.85E-08)+	8.61E-29 (3.26E-27)
f_{18}	1.20E-05 (3.22E-05)+	2.86E-10 (6.81E-10)+	1.11E-05 (3.96E-05)+	1.80E-11 (2.75E-10)	1.95E-05 (5.81E-05)+	6.44E-10 (1.02E-09)+	2.20E-05 (7.27E-05)+	8.42E-12 (2.53E-10)
f_{19}	3.75E-12 (2.16E-11)+	7.10E-21 (3.03E-20)+	4.13E-12 (4.94E-11)+	3.44E-24 (1.48E-22)	5.88E-12 (3.37E-11)+	1.63E-20 (6.00E-20)+	2.82E-12 (3.72E-11)+	2.36E-23 (8.19E-22)
f_{20}	3.29E-10 (2.64E-09)+	9.28E-19 (5.84E-18)+	2.17E-10 (4.05E-09)+	1.23E-25 (2.84E-24)	1.60E-09 (1.01E-08)+	5.83E-18 (3.48E-17)+	6.21E-10 (9.49E-09)+	2.80E-22 (8.27E-21)

表3 ABC、EABC、RABC 和 ERABC 算法在 $D=30$ 和 $D=60$ 时的 Friedman 排名

Tab. 3 Friedman average rankings of ABC, EABC, RABC and ERABC algorithms when $D=30$ and $D=60$

算法	Friedman 排名	
	$D=30$	$D=60$
ABC	3.63	3.63
EABC	2.65	2.68
RABC	2.65	2.58
ERABC	1.08	1.13

从表3中可以看出,ERABC 算法的性能最优,其他3种算法的性能从优到劣依次为 RABC、EABC、ABC。

综上所述,ERABC 算法在大部分函数上要优于 ABC、EABC 和 RABC 算法,特别是50次独立实验能在算法前期快速收敛至精度较高的解。这是因为 ERABC 算法在求解过程中很好地结合了精英区域学习策略和 RM 局部搜索方法的优势,在保持精英池中个体多样性的同时尽可能地利用精英信息加快算法的收敛速度,提高开采能力,增强了 ABC 算法的寻优能力。从非参数 Friedman 检验[18]的排名结果看,ERABC 算法取得了最优排名,其中,采用了区域学习精英池策略的 EABC 算法与 RABC 算法在处理 $D=30$ 和 $D=60$ 维的优化问题时,结果相差不大,因为精英区域学习策略在处理多峰函数问题时能在一定程度上避免算法陷入局部极值而又能尽可能地利用精英信息提高算法的收敛速度和开采能力。

3.4 与其他改进 ABC 算法的比较

为进一步分析 ERABC 算法性能,将 ERABC 算法与其他3种改进 ABC 算法(PABC[19]、OGABC[20]和 BCABC[16])进行对比分析。其中:PABC[19]算法在观察蜂阶段的解搜索方程中引入最优个体信息,同时采用 Powell's 方法进行局部搜索;OGABC[20]算法在雇佣蜂阶段根据 GABC 的搜索方程或根据正交学习策略更新个体;BCABC[16]算法在雇佣蜂阶段依适应值排序后以轮盘赌方式选择个体进行 CABC 搜索方程更新,同时在观察蜂阶段通过高斯分布产生新个体。不同于上述算法的是,ERABC 算法在提出的解搜索方程中引入了精英池信息,并在精英池的产生过程中采用区域学习方式扩大反向搜索区域以保证精英池中个体多样性,同时针对算法的开采能力引入 RM 进行局部搜索。

在比较实验中,4种算法的公共参数设置为 $NP=50$,$D=30$,$limit=200$。每个算法的其他参数按原文献进行设置,对比结果如表4所示。

表4 $D=30$ 时 ERABC 与 PABC、OGABC、BCABC 算法结果

Tab. 4 Results of PABC, OGABC, BCABC and ERABC algorithms when $D=30$

函数	最大评价次数	PABC[19] *Mean*(*Std*)	OGABC[20] *Mean*(*Std*)	BCABC[16] *Mean*(*Std*)	ERABC *Mean*(*Std*)
Sphere	150 000	2.63E-67(2.23E-67)	6.35E-60(8.85E-60)	2.84E-67(9.77E-68)	1.31E-67(1.04E-66)
Schwefel2.22	200 000	7.97E-35(4.51E-35)	3.58E-41(7.86E-41)	1.22E-47(7.18E-48)	5.82E-49(2.82E-48)
Rosebrock	300 000	3.85E-03(6.65E-03)	9.85E-01(8.36E-01)	2.28E-02(1.06E-02)	1.25E-21(3.19E-20)
Step	10 000	1.48E+01(7.78E+00)	9.83E+00(4.57E+00)	4.33E+00(1.73E-01)	2.40E-01(3.33E+00)
Ackley	50 000	6.07E-08(8.65E-08)	6.45E-08(2.86E-08)	2.97E-10(1.16E-10)	5.07E-12(8.70E-11)
Griewank	50 000	6.32E-12(1.41E-11)	8.36E-10(1.54E-10)	1.99E-15(2.40E-15)	4.07E-33(1.50E-31)
Penalized1	50 000	6.46E-21(4.92E-21)	5.89E-18(6.85E-19)	2.21E-21(7.28E-21)	6.37E-24(2.89E-22)
Penalized2	50 000	3.35E-20(1.58E-19)	4.78E-17(2.06E-17)	6.60E-21(4.41E-21)	2.93E-26(5.39E-25)
Alpine	300 000	3.14E-15(1.74E-15)	6.35E-17(7.23E-17)	6.90E-76(1.16E-75)	4.03E-18(2.55E-17)

从表4中可以看出:ERABC 在9个函数上表现出了优于 PABC、OGABC 算法的性能。求解 Sphere、Step 函数时与 BCABC 结果相当;求解 Alpine 函数时,BCABC 比 ERABC 算法更优,是因为 BCABC 算法采用高斯分布策略对个体进行更新,比 ERABC 算法采用的精英区域学习策略对该函数的处理更有效;在其余5个多峰测试函数上 ERABC 表现出了优于 BCABC 算法的性能,表明求解大部分多峰函数时 ERABC 有效提高算法开采能力的同时在一定程度上避免了算法容易陷入局部最优的问题。

表5给出了这4种算法在 $D=30$ 时的 Friedman 排名[18]。

从表5可以看出 ERABC 算法在4种算法中取得了最好的排名。

表 5　PABC、OGABC、BCABC 和 ERABC 算法在 $D=30$ 时的 Friedman 排名

Tab. 5　Friedman average rankings of PABC, OGABC, BCABC and ERABC algorithms when $D=30$

算法	Friedman 排名
PABC	3.13
OGABC	3.63
BCABC	2.13
ERABC	1.13

3.5　与其他相关演化算法的比较

为进一步验证 ERABC 算法的性能,将其与 4 种改进的 PSO 算法(FIPS[21]、HPSO-TVAC[22]、CLPSO[23]和 OLPSO-G[24])进行对比分析。其中:FIPS[21]算法根据每个个体邻域中的最优个体对当前个体进行更新;HPSO-TVAC[22]算法通过引入时变策略调整不同优化问题的合适突变步长,同时对个体停滞的速度向量进行重新初始化;CLPSO[23]算法依据个体的历史最优位置对个体的速度向量进行更新;OLPSO-G[24]算法通过正交学习策略来指导 PSO 算法进行全局搜索。在与 4 种改进 PSO 算法的比较实验中,所有函数的最大评价次数设置为 max FES = 200 000,公共参数同第 3.1 节,其他参数采用原文献推荐设置,对比结果如表 6 所示。

表 6　$D=30$ 时 ERABC 与 FIPS、HPSO-TVAC、CLPSO、OLPSO-G 算法结果

Tab. 6　Results of FIPS, HPSO-TVAC, CLPSO, OLPSO-G and ERABC algorithms when $D=30$

函数	FIPS[21] *Mean*(*Std*)	HPSO-TVAC[22] *Mean*(*Std*)	CLPSO[23] *Mean*(*Std*)	OLPSO-G[24] *Mean*(*Std*)	ERABC *Mean*(*Std*)
Sphere	2.42E-13(1.73E-13)	2.83E-33(3.19E-33)	1.58E-12(7.70E-13)	4.12E-54(6.34E-54)	1.51E-71(7.47E-70)
Schwefel2.22	2.76E-08(9.04E-09)	9.03E-20(9.58E-20)	2.51E-08(5.84E-09)	9.85E-30(1.01E-29)	5.82E-49(2.82E-48)
Rosebrock	2.51E+01(5.10E-01)	2.39E+01(2.65E+01)	1.13E+01(9.85E+00)	2.15E+01(2.99E+01)	3.56E-21(9.16E-20)
Step	0(0)	0(0)	0(0)	0(0)	0(0)
Ackley	2.33E-07(7.19E-08)	7.29E-14(3.00E-14)	3.66E-07(7.57E-08)	7.98E-15(2.03E-15)	5.86E-15(3.22E-15)
Griewank	9.01E-12(1.84E-11)	9.75E-03(8.33E-03)	9.02E-09(8.57E-09)	4.83E-03(8.63E-03)	3.69E-44(5.32E-44)
Penalized1	1.96E-15(1.11E-15)	2.71E-29(1.88E-28)	6.45E-14(3.70E-14)	1.59E-32(1.03E-33)	1.57E-32(3.87E-47)
Penalized2	2.70E-14(1.57E-14)	2.79E-28(2.18E-28)	1.25E-12(9.45E-12)	4.39E-04(2.20E-03)	1.50E-33(0.00E+00)

从表 6 可知:由于 Step 为阶梯函数,较简单且最优值位于区间内,5 种算法在该函数上都获得了相当的结果;ERABC 算法与 OLPSO-G 算法在求解 Ackley 函数时结果相当,在其他函数的求解中 ERABC 都取得了更优的结果;ERABC 比其他 4 种改进 PSO 算法在所有函数上的表现更优。

表 7 给出了 5 种算法的 Friedman 排名[18]。从表 7 中可知,在这 5 种算法中 ERABC 算法取得了最优排名。

表 7　FIPS、HPSO-TVAC、CLPSO、OLPSO-G 和 ERABC 算法在 $D=30$ 时的 Friedman 排名

Tab. 7　Friedman average rankings of FIPS, HPSO-TVAC, CLPSO, OLPSO-G and ERABC algorithms when $D=30$

算法	Friedman 排名
FIPS	4.31
HPSO-TVAC	3.44
CLPSO	4.56
OLPSO-G	3.19
ERABC	1.31

另将 ERABC 算法与基本 DE[25]算法及其 3 种改进算法(jDE[26]、JADE[27]和 SaDE[28])进行比较。其中:jDE[26]算法通过自适应调整控制参数来优化函数;JADE[27]算法在突变操作中采用 DE/current-to-pbest/1 策略,并对控制参数进行自适应调整;SaDE[28]算法在 4 种解搜索方程中适应性地选择变异操作策略和调整控制参数。与上述算法的比较中,最大评价次数设置如表 8 所示,公共参数设置同第 3.1 节,其他参数采用原文献推荐设置,对比结果如表 8 所示。

从表 8 可以看出:ERABC 算法相比 DE、jDE 和 SaDE 算法在 9 个测试函数上都体现了更优的性能;ERABC 在 8 个测试函数上取得了优于 JADE 算法的结果,在 Step 函数上,两种算法取得了相当的结果。

表 9 给出了 5 种算法的 Friedman 排名[18]。从表 9 可知,ERABC 算法在 5 种算法中取得了最优排名,表明 ERABC 是一种有竞争力的算法。

表8 $D=30$ 时 ERABC 与 DE、jDE、JADE、SaDE 算法结果

Tab. 8 Results of DE, jDE, JADE, SaDE and ERABC algorithms when $D=30$

函数	最大评价次数	DE[25] *Mean*(*Std*)	jDE[26] *Mean*(*Std*)	JADE[27] *Mean*(*Std*)	SaDE[28] *Mean*(*Std*)	ERABC *Mean*(*Std*)
Sphere	150 000	9.8E-14(8.4E-14)	1.46E-28(1.78E-28)	2.69E-56(1.41E-55)	3.28E-20(3.63E-20)	1.31E-67(1.04E-66)
Schwefel2.22	200 000	1.6E-09(1.1E-09)	9.02E-24(6.01E-24)	3.18E-25(2.05E-24)	3.51E-25(2.74E-25)	5.82E-49(2.82E-48)
Rosebrock	300 000	2.1E+00(1.5E+00)	1.3E+01(1.4E+01)	3.20E-01(1.10E+00)	2.10E+01(7.80E+00)	1.25E-21(3.19E-20)
Step	10 000	4.7E+03(1.1E+03)	6.13E+02(1.72E+02)	5.62E+00(1.87E+00)	5.07E+01(1.34E+01)	2.40E-01(3.33E+00)
Ackley	50 000	1.1E-01(3.9E-02)	2.37E-04(7.10E-05)	3.35E-09(2.84E-09)	3.81E-06(8.26E-07)	5.07E-12(8.70E-11)
Griewank	50 000	2.0E-01(1.1E-01)	7.29E-06(1.05E-05)	1.57E-08(1.09E-07)	2.52E-09(1.24E-08)	4.07E-33(1.50E-31)
Penalized1	50 000	1.2E-02(1.0E-02)	7.03E-08(5.74E-08)	1.67E-15(1.02E-14)	8.25E-12(5.12E-12)	6.37E-24(2.89E-22)
Penalized2	50 000	7.5E-02(3.8E-02)	1.80E-05(1.42E-05)	1.87E-10(1.09E-09)	1.93E-09(1.53E-09)	2.93E-26(5.39E-25)
Alpine	300 000	2.3E-04(1.7E-04)	6.08E-10(8.36E-10)	2.78E-05(8.43E-06)	2.94E-06(3.47E-06)	4.03E-18(2.55E-17)

表9 DE、jDE、JADE、SaDE 和 ERABC 算法在 $D=30$ 时的 Friedman 排名

Tab. 9 Friedman average rankings of DE, jDE, JADE, SaDE and ERABC algorithms when $D=30$

算法	Friedman 排名
DE	4.78
jDE	3.67
JADE	2.33
SaDE	3.22
ERABC	1.00

3.6 运行时间比较

为进一步分析 ERABC 算法性能,将 ERABC 与 ABC 算法达到各测试函数预设精度所耗 CPU 时间进行比较。在 $D=30$、$D=60$ 时独立实验50次,记录两种算法所耗平均 CPU 时间。其他参数设置同第3.1节。所耗 CPU 时间结果见表10。

从表10可看出:当 $D=30$ 时,ERABC 与 ABC 算法所消耗 CPU 时间之比的范围为[0.04,1.24],总时间比为0.73。当维度从30增至60时,函数的复杂度随之增加,导致算法的运行时间相应增加。当 $D=60$ 时,ERABC 算法与 ABC 算法所耗 CPU 时间之比的范围为[0.05,1.48],总时间比增至0.93。在2种维度下 ERABC 在大部分函数上与 ABC 算法所耗 CPU 时间比小于1,因为 ERABC 算法在种群初始化后针对种群中的最优个体进行了 RM 局部搜索,这一策略加快了算法的收敛速度和开采能力,能使函数快速收敛至预设精度。在小部分函数上 ERABC 算法与 ABC 算法所耗 CPU 时间比大于1,但并未高出很多,验证了第2.4节中算法时间复杂度分析的理论结果。综上所述,由于 ERABC 算法较好地结合了精英区域学习策略和 RM 局部搜索策略,很好地提高了算法的收敛速度和开采能力。

表10 ABC 和 ERABC 算法消耗的平均 CPU 时间

Tab. 10 Average CPU time of ABC and ERABC algorithms

函数	$D=30$			$D=60$		
	t/s		时间比	t/s		时间比
	ABC	ERABC	(ERABC:ABC)	ABC	ERABC	(ERABC:ABC)
f_1	2.89	0.13	0.04	6.04	0.28	0.05
f_2	3.54	0.54	0.15	10.20	1.28	0.13
f_3	1.43	0.18	0.13	3.90	0.47	0.12
f_4	0.55	0.59	1.07	1.77	1.74	0.98
f_5	4.93	2.93	0.59	12.17	9.23	0.76
f_6	5.34	4.47	0.84	12.75	7.36	0.58
f_7	5.54	6.20	1.12	11.62	14.04	1.21
f_8	5.25	6.50	1.24	14.51	21.39	1.47
f_9	6.93	4.15	0.60	21.10	18.66	0.88
f_{10}	3.42	1.53	0.45	8.08	3.85	0.48
f_{11}	5.16	4.00	0.78	14.34	12.87	0.90
f_{12}	6.34	6.08	0.96	18.87	19.64	1.04
f_{13}	1.61	1.67	1.04	6.15	7.87	1.28
f_{14}	3.82	0.19	0.05	8.01	0.43	0.05
f_{15}	20.68	15.03	0.73	75.95	81.74	1.08
f_{16}	8.55	6.07	0.71	24.96	28.57	1.14
f_{17}	8.42	6.06	0.72	23.53	13.98	0.59
f_{18}	6.89	8.11	1.18	14.14	19.72	1.39
f_{19}	5.87	7.04	1.20	16.23	24.06	1.48
f_{20}	7.68	3.45	0.45	22.71	16.30	0.72
总计	114.84	83.68	0.73	327.03	303.48	0.93

4 结 论

针对 ABC 算法求解复杂多峰函数时容易出现收敛速度慢、开采能力不足的缺点,提出了 ERABC 算法。该算法将精英区域学习策略引入到解的搜索方程,通过区域学习策略增强精英池中个体的多样性,在一定程度上避免算法陷入局部最优,同时利用精英信息提高算法的收敛速度。为了进一步提高算法的开采能力,在 ERABC 算法中融合了 RM 局部搜索。在实验中,分析了精英池大小对 ERABC 算法性能的影响,并分析了改进策略的有效性,将其与 3 种改进 ABC 算法及多种改进 DE 算法和改进 PSO 算法进行了对比分析。结果表明,ERABC 算法有效加快了 ABC 的收敛速度,提高了算法的开采能力。

参考文献:

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

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

[3] Qin Quande, Cheng Shi, Li Li, et al. Artificial bee colony algorithm: A survey [J]. CAAI Transactions on Intelligent Systems, 2014, 9(2): 127 – 135. [秦全德,程适,李丽,等. 人工蜂群算法研究综述[J]. 智能系统学报, 2014, 9(2): 127 – 135.]

[4] Du Zhengcong, Feng Dahai, Niu Gaoyuan. Particle swarm optimization artificial immune particle filter [J]. Journal of Sichuan University (Engineering Science Edition), 2013, 45(1): 146 – 151. [杜正聪,冯大海,牛高远. 粒子群优化人工免疫粒子滤波器[J]. 四川大学学报(工程科学版), 2013, 45(1): 146 – 151.]

[5] Shen Jinan, Liang Fang, Zheng Minghui. New hybrid differential evolution and particle swarm optimization algorithm and its application [J]. Journal of Sichuan University (Engineering Science Edition), 2014, 46(6): 38 – 43. [沈济南,梁芳,郑明辉. 一种新的混合差分粒子群优化算法及其应用[J]. 四川大学学报(工程科学版), 2014, 46(6): 38 – 43.]

[6] Wu Tao, Chen Xi, Yan Yusong. Study of the binary correlation quantum-behaved PSO algorithm [J]. Journal of Sichuan University (Engineering Science Edition), 2014, 46(4): 103 – 110. [吴涛,陈曦,严余松. 2 元相关性量子行为粒子群优化算法研究[J]. 四川大学学报(工程科学版), 2014, 46(4): 103 – 110.]

[7] Gao Weifeng, Liu Sanyang, Huang Lingling. Enhancing artificial bee colony algorithm using more information-based search equations [J]. Information Sciences, 2014, 270(4): 112 – 133.

[8] Xiang Yi, Peng Yuming, Zhong Yubin, et al. A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization [J]. Computational Optimization and Applications, 2014, 57(2): 493 – 516.

[9] Zhao Hui, Li Mudong, Weng Xingwei. Improved artificial bee colony algorithm with self-adaptive global bestguided quick searching strategy [J]. Control and Decision, 2014, 10(10): 1 – 7. [赵辉,李牧东,翁兴伟. 具有自适应全局最优引导快速搜索策略的人工蜂群算法[J]. 控制与决策, 2014, 10(10): 1 – 7.]

[10] Ma Lianbo, Hu Kunyuan, Zhu Yunlong, et al. A hybrid artificial bee colony optimizer by combining with life-cycle, Powell's search and crossover [J]. Applied Mathematics and Computation, 2015, 252: 133 – 154.

[11] Tran D H, Cheng M Y, Cao M T. Hybrid multiple objective artificial bee colony with differential evolution for the time-cost-quality tradeoff problem [J]. Knowledge-Based Systems, 2015, 74: 176 – 186.

[12] Wang Zhigang. Hybrid optimization algorithm based on particle swarm optimization and artificial bee colony algorithm [J]. Science Technology and Engineering, 2012, 12(20): 4921 – 4925. [王志刚. 基于粒子群和人工蜂群算法的混合优化算法[J]. 科学技术与工程, 2012, 12(20): 4921 – 4925.]

[13] Kang Fei, Li Junjie, Ma Zhenyue. Rosenbrock artificial bee colony algorithm for accurate global optimization of

numerical functions[J]. Information Sciences, 2011, 181(16):3508 - 3531.

[14]Tizhoosh H R. Opposition-based learning: A new scheme for machine intelligence[C]//Proceedings of International Conference on the Computational Intelligence for Modeling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce. Piscataway: IEEE, 2005:695 - 701.

[15]Peng Hu, Wu Zhijian, Zhou Xinyu, et al. Dynamic differential evolution algorithm based on elite local learning [J]. Acta Electronica Sinica, 2014, 8(8):1522 - 1530. [彭虎,吴志健,周新宇,等. 基于精英区域学习的动态差分进化算法[J]. 电子学报,2014,8(8):1522 - 1530.]

[16]Gao Weifeng, Liu Sanyang, Huang Lingling, et al. Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood[J]. Information Sciences, 2015, 316:180 - 200.

[17]Wang Hui, Wu Zhijian, Liu Yong, et al. Space transformation search: A new evolutionary technique[C]//Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation. New York: ACM, 2009:537 - 544.

[18]Garcia S, Fernandez A, Luengo J, et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power[J]. Information Sciences, 2010, 180(10):2044 - 2064.

[19]Gao Weifeng, Liu Sanyang, Huang Lingling. A novel artificial bee colony algorithm with Powell's method[J]. Applied Soft Computing, 2013, 13(9):3763 - 3775.

[20]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.

[21]Rui M, Kennedy J, Neves J. The fully informed particle swarm: Simpler maybe better[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3):204 - 210.

[22]Ratnaweera A, Halgamuge S, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3):240 - 255.

[23]Liang Jing, Qin Akai, Suganthan P N, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(3):281 - 295.

[24]Zhan Zhihui, Zhang Jun, Li Yun, et al. Orthogonal learning particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 15(6):1763 - 1764.

[25]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.

[26]Brest J, Greiner S, Boskovic B, et al. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(6):646 - 657.

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

[28]Qin Akai, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(2):398 - 417.

(编辑 赵 婧)