软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(9):2167−2190 [doi: 10.13328/j.cnki.jos.004800]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于正交实验设计的人工蜂群算法*

周新宇[1], 吴志健[2], 王明文[1]

[1](江西师范大学 计算机信息工程学院,江西 南昌 330022)

[2](软件工程国家重点实验室(武汉大学),湖北 武汉 430072)

通讯作者: 周新宇, E-mail: xyzhou@whu.edu.cn

摘 要: 人工蜂群算法是近年来提出的较为新颖的全局优化算法,已成功地应用于解决不同类型的实际优化问题.然而在该算法及相关的改进算法中,侦察蜂通常采用随机初始化的方法来生成新食物源.虽然这种方法较为简单,但易造成侦察蜂搜索经验的丢失.从算法搜索过程的内在机制出发,提出采用正交实验设计的方式来生成新的食物源,使得侦察蜂能够同时保存被放弃的食物源和全局最优解在不同维度上的有益信息,提高算法的搜索效率.在16 个典型的测试函数上进行了一系列实验验证,实验结果表明:1) 该方法能够在基本不增加算法运行时间的情况下,显著地提高人工蜂群算法的求解精度和收敛速度;2) 与 3 种典型的变异方法相比,有更好的整体性能;3) 可作为提高其他改进人工蜂群算法性能的通用框架,具备有良好的普适性.

关键词: 人工蜂群;侦察蜂;搜索经验;正交实验设计;通用框架

中图法分类号: TP301

中文引用格式: 周新宇,吴志健,王明文.基于正交实验设计的人工蜂群算法.软件学报,2015,26(9):2167−2190. http://www.jos.org.cn/1000-9825/4800.htm

英文引用格式: Zhou XY, Wu ZJ, Wang MW. Artificial bee colony algorithm based on orthogonal experimental design. Ruan Jian Xue Bao/Journal of Software, 2015,26(9):2167−2190 (in Chinese). http://www.jos.org.cn/1000-9825/4800.htm

Artificial Bee Colony Algorithm Based on Orthogonal Experimental Design

ZHOU Xin-Yu[1], WU Zhi-Jian[2], WANG Ming-Wen[1]

[1](School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China)

[2](State Key Laboratory of Software Engineering (Wuhan University), Wuhan 430072, China)

Abstract: Developed in recent years, artificial bee colony (ABC) algorithm is a relatively new global optimization algorithm that has been successfully used to solve various real-world optimization problems. However, in the algorithm, including its improved versions, the scout bee usually employs the random initialization method to generate a new food source. Although this method is relatively straightforward, it tends to result in the loss of the scout bee's search experience. Based on the intrinsic mechanism of ABC's search process, this paper proposes a new scheme that employs the orthogonal experimental design (OED) to generate a new food source for the scout bee so that the scout bee can preserve useful information of the abandoned food source and the global optimal solution in different dimensions simultaneously, and therefore enhancing the search efficiency of ABC. A series of experiments on the 16 well-known benchmark functions has been conducted with the experimental results showing the following advantages of the presented approach: 1) it can significantly improve the solution accuracy and convergence speed of ABC almost without increasing the running time; 2) it has better performance than other three typical mutation methods; and 3) it can be used as a general framework to enhance the performance of other improved ABCs with good applicability.

Key words: artificial bee colony; scout bee; search experience; orthogonal experimental design; general framework

* 基金项目: 国家自然科学基金(61305150, 61364025, 61462045); 教育部人文社科基金(13YJCZH174); 软件工程国家重点实验室开放基金(SKLSE2014-10-04); 江西省自然科学基金(20151BAB217007); 江西省教育厅科学技术项目(GJJ13729, GJJ14747)

收稿时间: 2014-05-19; 修改时间: 2014-10-20; 定稿时间: 2014-12-06

人工蜂群(artificial bee colony,简称 ABC)算法是近年来提出的一种较为新颖的群体智能算法[1,2],具有结构简单、控制参数少、不依赖梯度信息、性能优良等特点.该算法与粒子群优化(particle swarm optimization,简称 PSO)算法[3,4]等类似,都是通过模拟自然界生物的群体智能行为而设计的全局优化算法.但不同于 PSO 的是:ABC 模拟的是蜂群的智能采蜜行为,它根据不同种类蜜蜂的分工协作来实现整个蜂群采蜜量的最大化,即,找到优化问题的最优解[5].Karaboga 和 Akay 的研究工作[6]指出,ABC 的性能与差分演化(differential evolution,简称 DE)算法、PSO 算法以及遗传算法相比是有竞争力的.目前,ABC 已成功地应用于解决多种不同类型的实际优化问题,如参数优化[7]、符号回归[8]、神经网络[9,10]以及车辆路径问题[11]等.

ABC 的搜索过程可分为 3 个阶段:雇佣蜂(employed bee)阶段、观察蜂(onlooker bee)阶段以及侦察蜂(scout bee)阶段.在雇佣蜂和观察蜂两个阶段中,采用同一种解搜索方程(solution search equation)来寻找新的食物源;而在侦察蜂阶段,对被放弃食物源进行随机初始化.事实上,就整个搜索过程而言,解搜索方程在很大程度上决定了 ABC 的性能[12].因此,在 ABC 的相关研究工作中,有相当一部分是集中在如何改进解搜索方程或提出新的解搜索方程.这方面的一个代表性研究工作是 Zhu 和 Kwong 在 2010 年提出的 *gbest* 引导的 ABC 算法(GABC)[13],他们在解搜索方程中融入了全局最优解 *gbest* 的信息来提高算法的开采能力.值得说明的是,这种改进的动机是受到了 PSO 的启发.ABC 的另一个研究趋势是如何与其他搜索算子相结合来提高算法的搜索能力,比如,Gao 和 Liu 等人在文献[14]中将 ABC 与局部搜索的 Powell 方法结合起来.这两类研究工作有效地提高了 ABC 的性能,但遗憾的是,它们中还普遍存在着一个缺陷,即:侦察蜂采用随机初始化的方法来生成新的食物源,用于替代被放弃食物源.虽然这种机制较为简单,但需要指出的是:被放弃的食物源往往具有较好的适应值(在求解多峰函数时,可能是某一局部极值点),在搜索空间中也一般处于较好的位置,包含了有助于指导算法进一步搜索的有益信息.如果直接采用随机初始化的方法,易导致侦察蜂的搜索经验丢失,制约了 ABC 性能的进一步提高.

本文从 ABC 搜索过程的内在机制出发,注意到侦察蜂阶段的作用是避免局部极值,防止被放弃食物源占用过多的计算资源.为此,提出一种新的方法用于侦察蜂生成新食物源,即:采用正交实验设计(orthogonal experimental design,简称 OED)来帮助侦察蜂系统地探测被放弃食物源和全局最优解之间的超长方体搜索空间,使得侦察蜂能够同时保存被放弃食物源和全局最优解在不同维度上的有益信息,达到避免局部极值又能保存搜索经验的目的.为验证本文方法的有效性,在 16 个典型的测试函数(包含了偏移和旋转类型)上进行一系列实验.实验结果表明,本文方法具有如下 3 个方面的优势:

- 与随机初始化方法相比,本文方法能够在基本不增加算法运行时间的情况下,显著地提高 ABC 的求解精度和收敛速度;
- 与 3 种典型的变异方法相比,本文方法的性能更加优秀;
- 本文方法具备良好的普适性,可作为一个通用框架用于提高其他改进 ABC 算法的性能.

本文第 1 节简要介绍 ABC 算法.第 2 节介绍 ABC 算法的相关研究工作.第 3 节介绍 OED,并详细叙述本文提出的方法.第 4 节给出一系列相关实验及结果分析.最后一节对本文工作进行总结.

1 人工蜂群算法

在 ABC 中,人工蜂群包含了 3 种不同类型的蜜蜂:雇佣蜂、观察蜂以及侦察蜂.雇佣蜂负责勘探食物源,并在蜂巢的舞蹈区通过摇摆舞的形式将食物源的相关信息分享给观察蜂[15],这些信息包括食物源的位置和蜂蜜量的多少;之后,观察蜂根据收到的信息,以一定的概率选择某一食物源继续开采,若该食物源的蜂蜜量越多,则被选中的概率就越高;如果一个食物源的适应值连续 *limit* 次未更新,那么说明它已被开采殆尽,需要放弃该位置,在此,采蜜的雇佣蜂就转变为侦察蜂,再重新随机搜索一个新的食物源.需要说明的是:一个食物源的位置对应于优化问题的一个候选解,食物源的蜂蜜量即指适应值.雇佣蜂与食物源是一一对应的,同时,雇佣蜂的数量和观察蜂相同,而侦察蜂只有一只.

类似于其他演化算法,ABC 同样采用一个随机生成的群体开始迭代搜索.设食物源的规模为 SN,其中,食物源 $X_i=(x_{i,1},x_{i,2},\ldots,x_{i,D})$代表一个候选解,$i\in\{1,2,\ldots,SN\}$,$D$ 为优化问题的维度.算法开始迭代之后,ABC 根据蜜蜂的

类型将搜索过程分为 3 个阶段,如下所示:

- 雇佣蜂阶段

在该阶段,每一只雇佣蜂在对应的食物源 X_i 处,采用解搜索方程生成一个新的食物源 $V_i=(v_{i,1},v_{i,2},\ldots,v_{i,D})$.如果 V_i 的蜂蜜量更多即适应值更优,那就替换 X_i.该方程如下:

$$v_{i,j}=x_{i,j}+\phi_{i,j}\cdot(x_{i,j}-x_{k,j}) \tag{1}$$

其中,$\phi_{i,j}$ 是[−1,1]间的随机数;$k\in\{1,2,\ldots,SN\}$是随机选择的一个食物源,且 $k\neq i;j\in\{1,2,\ldots,D\}$是随机选择的一个维度.

- 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进一步开采.这种选择方式通常采用基于适应值的方法,如轮盘赌机制.若观察蜂采用轮盘赌机制进行选择,则一个食物源 X_i 被选中的概率 p_i 为

$$p_i=\frac{f_i}{\sum_{j=1}^{SN}f_j} \tag{2}$$

其中,f_i 为食物源 X_i 的适应值.从该式可以看出:食物源的适应值越优,它被观察蜂选中的概率越高.

- 侦察蜂阶段

当雇佣蜂对应的食物源的适应值连续 *limit* 次未更新,说明该食物源已被开采耗尽.这种情况下,对应的雇佣蜂转变成侦察蜂,采用如下初始化方法重新获得一个新的食物源:

$$x_{i,j}=a_j+rand\cdot(b_j-a_j) \tag{3}$$

其中,*rand* 是[0,1]间的随机数,$[a_j,b_j]$是第 j 维变量的上下界.值得说明的是:除群体规模等公共参数外,*limit* 是 ABC 中唯一的参数.

2 相关工作

ABC 的优良性能吸引了众多研究人员的注意,随之多种不同的改进版本也相继被提出.这些研究工作有效地提高了 ABC 的性能,它们大致可分为两个方面:1) 如何改进解搜索方程或提出新的解搜索方程;2) 如何与其他搜索算子相结合来提高算法的搜索能力.下面简要介绍这两类研究工作.

(1) 如何改进解搜索方程或提出新的解搜索方程

受 PSO 的启发,Zhu 和 Kwong[13]提出了一种 *gbest* 引导的 ABC 算法(GABC),将全局最优解 *gbest* 的信息融入到解搜索方程中,用于提高算法的开采能力.相应地,在解搜索方程中增加了一个新的项,即 *gbest* 项.报道的实验结果表明:在多个测试函数上,GABC 比 ABC 更加优秀.Akay 和 Karaboga[16]研究了影响 ABC 性能的两个因素:解搜索方程中的扰动频率和扰动尺度.为此,他们引入了两个新的参数来控制这两个因素,构造了改进的 ABC 算法.值得说明的,控制扰动频率的参数本质上与其他演化算法中的交叉概率是相同的.基于目前最优解(best-so-far),Banharnsakun 和 Achalakul 等人[17]提出了改进的 ABC,他们将算法的搜索方向偏移至目前最优解,从而加快收敛速度.在文献[18]中,Das 和 Biswas 等人将食物源之间的距离因素加入到解搜索方程中,避免在食物源的选择过程中全局最优解引起早熟现象.受 DE 算法中变异策略的启发,在文献[19−21]中,Gao 和 Liu 等人提出了多个不同版本的改进解搜索方程.在文献[19]中,受 DE/best/1 和 DE/rand/1 两种变异策略的启发,他们提出了 ABC/best/1 和 ABC/rand/1 两种改进的解搜索方程;进一步,为平衡算法的勘探和开采能力,引入概率选择参数来控制这两种改进解搜索方程的使用频率.在文献[20]中,他们提出两种不同的改进 ABC 版本,分别采用了 ABC/best/1 和 ABC/best/2 两种解搜索方程.实验结果表明,ABC/best/1 的性能要显著地优于 ABC/best/2.在文献[21]中,他们同样采用了 ABC/best/1 解搜索方程构造了改进的 ABC(MABC),但不同于标准 ABC 的是,MABC 剔除了轮盘赌选择和侦察蜂阶段.

(2) 如何与其他搜索算子相结合来提高算法的搜索能力

Kang 和 Li 等人[22]将 Nelder-Mead 单形搜索机制(NMSS)和 ABC 结合起来,提出了一种混合的单形 ABC

算法.因为 NMSS 是一种局部下降算法,所以该机制在一定程度上可视为局部开采算子.类似地,沿着这一思路,他们又继续提出了 Rosenbrock ABC 算法[23],将 Rosenbrock 方法同样作为局部开采算子与 ABC 相结合;同时,为解决多峰问题,改进了 Rosenbrock 方法.Bansal 和 Sharma 等人[24]提出了一种 Memetic ABC,在 ABC 的 3 个搜索阶段基础上,增加了一种局部搜索算子,专门用于搜索全局最优解邻域内的个体.在他们的方法中,还采用了黄金分割搜索方式来控制参数 Φ.El-Abd[25]将一般反向学习机制应用到 ABC 中,提出了一般反向学习的 ABC 算法(GOABC),在群体初始化和算法迭代过程中均采用了反向学习机制.为提高算法的搜索能力,Gao 和 Liu 等人[26]在雇佣蜂阶段中结合了正交实验设计的方法;进一步在文献[14]中将传统的 Powell 方法作为局部搜索算子与 ABC 相结合,互补 Powell 方法的开采能力和 ABC 的勘探能力,从而提高了 ABC 算法的整体性能.

上述两类工作有效地提高了 ABC 的性能,为后续的研究提供了参考.严格来说,本文提出的方法属于第 2 类,但与第 2 类中其他方法又有区别的是:本文的目的在于揭示 ABC 中侦察蜂阶段存在的不足,试图从优化算法自身结构的角度着手提出相应的解决方法,即,在侦察蜂阶段采用正交实验设计的方法来生成新食物源.该方法不仅能够提高标准 ABC 的性能,且能作为一个通用的框架,用于提高其他改进 ABC 的性能.

3 基于正交实验设计的 ABC 算法(ABC-OED)

3.1 正交实验设计(OED)

正交实验设计是一种解决多因素、多水平的实验设计方法,能以较少的实验次数找到最好或较好的实验条件[27].该方法首先选择一个合适的正交表 $L_M(Q^N)$来安排实验,再通过因素分析来推测最好的水平组合方式.$L_M(Q^N)$表示具有 N 个因素和 Q 种水平的正交表,M 是水平组合的数量,也等于该正交表的行数.对一个具有 N 因素、Q 水平的实验问题,采用全面实验的方法需完成 Q^N 组实验.如果 N 和 Q 的值都较大,进行全面实验显然不是明智的选择.但采用 OED 方法,则只需安排 M 组实验,而 M 通常是远小于 Q^N 的.为更清楚地说明 OED 的使用方法,下面以化学实验中的转化率问题为例[26,28].

影响转化率的因素有 3 个,分别是温度(°C)、时间(M)以及碱含量(%).每个因素均有 3 种不同的水平,即 3 种值可供选择,见表 1.若进行全面实验,则需 3^3=27 组实验.但采用正交表 $L_9(3^4)$,OED 方法只需 9 组实验.

Table 1 Factors and levels in the conversion ratio problem

表 1 转化率问题中的因素及水平

Levels	A: Temperature (°C)	B: Time (M)	C: Alkali (%)
L_1	80	90	5
L_2	85	120	6
L_3	90	150	7

正交表 $L_9(3^4)$如公式(4)所示,每一行即为一种水平组合,也就是一次实验.例如该表的最后一行中,因素 1 和因素 2 同为第 3 种水平,因素 3 为第 2 种水平,因素 4 为第 1 种水平.正交表的正交性体现在[29]:1) 每列中每个因素的每种水平出现的次数相同;2) 任意两因素的每种水平的组合出现次数相同.

$$L_9(3^4)=\begin{bmatrix}1&1&1&1\\1&2&2&2\\1&3&3&3\\2&1&2&3\\2&2&3&1\\2&3&1&2\\3&1&3&2\\3&2&1&3\\3&3&2&1\end{bmatrix} \tag{4}$$

算法 1 给出了构建正交表 $L_M(Q^N)=(a_{i,j})_{M\times N}$ 的伪代码[30,31],其中,Q 为素数,且 $M=Q^J$;而 J 需满足公式(5):

$$N \leqslant \frac{Q^J-1}{Q-1} \tag{5}$$

算法 1. Construction of orthogonal array $L_M(Q^N)=(a_{i,j})_{M\times N}$.

1. **for** $\{k=1:J\}$ **do**
2. $j=\dfrac{Q^{k-1}-1}{Q-1}+1$
3. **for** $\{i=1:Q^J\}$ **do**
4. $a_{i,j}=\left\lfloor \dfrac{i-1}{Q^{J-k}} \right\rfloor \bmod Q$
5. **end**
6. **end**
7. **for** $\{k=2:J\}$ **do**
8. $j=\dfrac{Q^{k-1}-1}{Q-1}+1$
9. **for** $\{s=1:j-1\}$ **do**
10. **for** $\{t=1:Q-1\}$ **do**
11. $a_{j+(s-1)(Q-1)+t}=(a_s\times t+a_j) \bmod Q$
12. **end**
13. **end**
14. **end**
15. $a_{i,j}=a_{i,j}+1, 1\leqslant i\leqslant M$, and $1\leqslant j\leqslant N$

在使用 OED 方法安排实验之后,可采用因素分析计算出每个因素的每种水平对实验结果的影响,从而推测出最佳的水平组合方式.设 F_m 为正交表中第 m 组实验的结果,$1\leqslant m\leqslant M$;S_{nq} 为因素 n 的第 q 种水平对结果的影响,$1\leqslant q\leqslant Q$.S_{nq} 可按公式(6)计算得出[26,28]:

$$S_{nq}=\frac{\sum_{m=1}^{M}F_m\times z_{mnq}}{\sum_{m=1}^{M}z_{mnq}} \tag{6}$$

其中,若第 m 组实验中包含了因素 n 的第 q 种水平,则 $z_{mnq}=1$;否则,$z_{mnq}=0$.表 2 给出上述转化率问题的因素分析结果,从中可看出,最佳的水平组合方案是(A3,B2,C2).

Table 2 Results of factor analysis on the conversion ratio problem

表 2 转化率问题的因素分析结果

Combination	A: Temperature (°C)	B: Time (M)	C: Alkali (%)	Results
C_1	(1)80	(1)90	(1)5	F_1=31
C_2	(1)80	(2)120	(2)6	F_2=54
C_3	(1)80	(3)150	(3)7	F_3=38
C_4	(2)85	(1)90	(2)6	F_4=53
C_5	(2)85	(2)120	(3)7	F_5=49
C_6	(2)85	(3)150	(1)5	F_6=42
C_7	(3)90	(1)90	(3)7	F_7=57
C_8	(3)90	(2)120	(1)5	F_8=62
C_9	(3)90	(3)150	(2)6	F_9=64
Levels	Factor analysis			
L_1	$S_{A1}=(F_1+F_2+F_3)/3=41$	$S_{B1}=(F_1+F_4+F_7)/3=47$	$S_{C1}=(F_1+F_6+F_8)/3=45$	
L_2	$S_{A2}=(F_4+F_5+F_6)/3=48$	$S_{B2}=(F_2+F_5+F_8)/3=55$	$S_{C2}=(F_2+F_4+F_9)/3=57$	
L_3	$S_{A3}=(F_7+F_8+F_9)/3=61$	$S_{B3}=(F_3+F_6+F_9)/3=48$	$S_{C3}=(F_3+F_5+F_7)/3=48$	
OED results	A3	B2	C2	

3.2 ABC-OED

在 ABC 的侦察蜂阶段,若某一食物源的适应值超过 *limit* 次仍未更新,说明该食物源已被开采殆尽,需放弃该位置.相应地,在该处采蜜的雇佣蜂转变为侦察蜂,按公式(3)产生一个新的位置.从 ABC 搜索过程的内在机制来看,该过程的主要作用是避免局部极值,防止被放弃食物源占用过多的计算资源,从而能够将有限的计算资源用于搜索解空间中的其他可行区域,提高算法的搜索效率.虽然按公式(3)进行随机初始化可快速地找到一个新的位置,但该方式具有一定的局限性,易导致侦察蜂的搜索经验丢失.通常,被放弃食物源往往具有较好的适应值,在搜索空间中也处于较好的位置.若优化问题是多峰类型,则该食物源很可能位于某一局部极值点的邻域内,在一定程度上,比普通随机点的位置更好,包含了有利于指导算法进一步搜索的有益信息.为更好地说明该现象,以优化二维的 Shekel's Foxholes 函数问题为例,该函数的 3-D 图如图 1 所示.该函数是多峰类型,有 24 个局部极值点和 1 个全局最优点 $f(-32,-32)=0.998$,搜索区间范围是$[-65.536,65.536]^2$,其具体定义可参考 Yao 和 Liu 等人的文献[32].

此例中,假设食物源的规模为 30,在某一代时,食物源在搜索空间中的分布情况如图 2 所示.图 2 中,空心矩形为 25 个极值点所处位置,空心圆点为食物源所处位置,实心圆点为全局最优解 $f(0,-32)=2.982$,实心三角形为连续 *limit* 次仍未改进的食物源 $f(-32,16)=15.503$.可以看出:如果被放弃食物源位于局部极值点上,直接采用随机初始化的方法来生成替代个体,则显然会丢失该局部极值点的位置信息.事实上,若侦察蜂能够同时保存被放弃食物源在第 1 维的信息和全局最优解在第 2 维的信息,即能找到全局极值点(−32,−32).

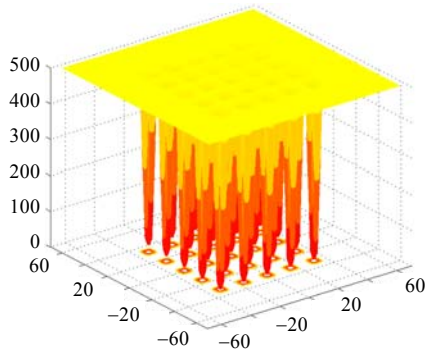


Fig.1 3-D plot of the Shekel's Foxholes function

图 1 Shekel's Foxholes 函数的 3-D 图

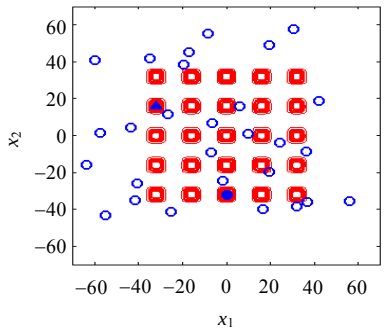


Fig.2 Distribution of food sources in the search space

图 2 食物源在搜索空间中的分布情况

为此,本文提出采用 OED 方法来克服该不足.在此方法中,为保存被放弃食物源在不同维度上的有益信息,并从全局最优解 *gbest* 中学习到有益信息,我们将侦察蜂搜索新食物源的行为视为一次实验,而被放弃食物源和 *gbest* 的每个维度视为影响实验结果的因素,采用 OED 方法来帮助侦察蜂探测两者所确定的超长方体空间,找出最好或较好的水平组合方式,使得构造出的新食物源达到既能跳出局部极值,又能保存搜索经验的目的.设当前被放弃的食物源为 $X=(x_1,x_2,\ldots,x_D)$,全局最优解 $gbest=(gbest_1,gbest_2,\ldots,gbest_D)$.首先,由 X 和 *gbest* 可以确定每个维度的搜索区域范围:$[\min(x_i,gbest_i),\max(x_i,gbest_i)]$;之后,再对每个维度的搜索区域范围进行量化[33,34],划分为 Q 个不同的水平,见公式(7):

$$l_{i,j}=\min(x_i,gbest_i)+\frac{j-1}{Q-1}(\max(x_i,gbest_i)-\min(x_i,gbest_i)),j=1,2,...,Q \tag{7}$$

因此,对所有维度而言,共有 Q^D 种不同的水平组合方式.若在两维空间中 $X=(4.5,9.0)$,$gbest=(0.5,1.0)$,水平 Q 为 5,那么它们确定的搜索空间的范围是$[0.5,4.5]\times[1.0,9.0]$,经过量化后,总共包含了 $Q^D=5^2=25$ 个点,如图 3 所示.

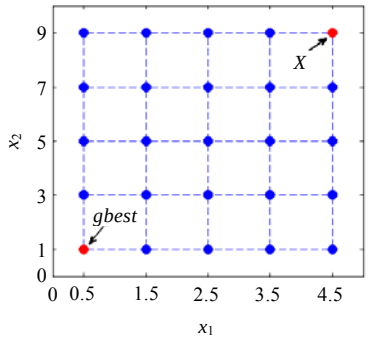


Fig.3 Illustration for quantization of the search space

图 3 搜索空间量化示意图

在确定了因素个数 N 及划分水平 Q 后,可应用正交表 $L_M(Q^N)$安排实验方案.但是需要指出的是:通常,优化问题的维度 D 较大,若直接应用正交表,则需要的实验次数 M 依然是相当大的,会导致 OED 占用过多的计算资源,降低了该方法的有效性.例如:若 D=50,Q=5,由公式(5)可计算出对应的实验次数 M=625 次,那么侦察蜂需要消耗 625 次适应度函数评估次数,显然,该代价是较大的.为此,在应用 OED 方法之前,需先对被放弃食物源和 *gbest* 的维度(决策变量)进行分组,将一组决策变量视为单个因素,从而减少总的因素个数,降低 OED 的计算负担.下面举例来说明分组的过程,将 D 个决策变量分为 F 组,见公式(8).

$$\begin{cases} G_1=(x_1,...,x_{k_1}) \\ G_2=(x_{k_1+1},...,x_{k_2}) \\ ... \\ G_F=(x_{k_{F-1}+1},...,x_D) \end{cases} \tag{8}$$

其中,$k_1,k_2,\ldots,k_{F-1}$ 为(1,D)之间随机生成的整数,且满足 $1<k_1<k_2<\ldots<k_{F-1}<D$.按公式(7)将每个因素划分为 Q 水平,则第 i 个分组的水平划分方式见公式(9).

$$\begin{cases} G_i(1)=(l_{k_{i-1}+1,1},l_{k_{i-1}+2,1},...,l_{k_i,1}) \\ G_i(2)=(l_{k_{i-1}+1,2},l_{k_{i-1}+2,2},...,l_{k_i,2}) \\ ... \\ G_i(Q)=(l_{k_{i-1}+1,Q},l_{k_{i-1}+2,Q},...,l_{k_i,Q}) \end{cases} \tag{9}$$

需要说明的是:若采用的正交表 $L_M(Q^N)$的列数大于分组个数,即 $N>F$,则可直接应用该表的前 F 列来安排实验[29,30].

若在 7 维空间中,X=(1.0,2.0,0.0,8.0,4.0,3.0,7.0),*gbest*=(3.0,4.0,2.0,6.0,6.0,1.0,5.0),水平 Q=3,分组数 F=4,随机生成的 k_1,k_2,k_3 分别为 2,5,6,那么按公式(8)可分组为公式(10)的形式:

$$\begin{cases} G_1=(x_1,x_2) \\ G_2=(x_3,x_4,x_5) \\ G_3=(x_6) \\ G_4=(x_7) \end{cases} \tag{10}$$

再按公式(9)将每组分为 3 种水平,则有:$G_1(1)$=(1.0,2.0),$G_1(2)$=(2.0,3.0),$G_1(3)$=(3.0,4.0);$G_2(1)$=(0.0,6.0,4.0),$G_2(2)$=(1.0,7.0,5.0),$G_2(3)$=(2.0,8.0,6.0);$G_3(1)$=(1.0),$G_3(2)$=(2.0),$G_3(3)$=(3.0);$G_4(1)$=(5.0),$G_4(2)$=(6.0),$G_4(3)$=(7.0).之后,按公式(4)所示的正交表 $L_9(3^4)$来安排实验,可构造公式(11)所示的 9 个新个体 $C_i(i=1,2,\ldots,9)$.进一步按公式(6)对每个因素进行因素分析,确定最佳的水平组合方式,得到一个预测个体 P.最后,从 C_i 和 P 中选择出最优的一个作为被放弃食物源 X 的替代个体.需要说明的是:如果被放弃的食物源恰好是全局最优个体,则从群体中随机选

择出一个个体,按 OED 方法的基本步骤来执行操作.

$$\begin{cases} C_1=(1.0,\ 2.0,\ 0.0,\ 6.0,\ 4.0,\ 1.0,\ 5.0) \\ C_2=(1.0,\ 2.0,\ 1.0,\ 7.0,\ 5.0,\ 2.0,\ 6.0) \\ C_3=(1.0,\ 2.0,\ 2.0,\ 8.0,\ 6.0,\ 3.0,\ 7.0) \\ C_4=(2.0,\ 3.0,\ 0.0,\ 6.0,\ 4.0,\ 2.0,\ 7.0) \\ C_5=(2.0,\ 3.0,\ 1.0,\ 7.0,\ 5.0,\ 3.0,\ 5.0) \\ C_6=(2.0,\ 3.0,\ 2.0,\ 8.0,\ 6.0,\ 1.0,\ 6.0) \\ C_7=(3.0,\ 4.0,\ 0.0,\ 6.0,\ 4.0,\ 3.0,\ 6.0) \\ C_8=(3.0,\ 4.0,\ 1.0,\ 7.0,\ 5.0,\ 1.0,\ 7.0) \\ C_9=(3.0,\ 4.0,\ 2.0,\ 8.0,\ 6.0,\ 2.0,\ 5.0) \end{cases} \tag{11}$$

3.3 算法框架

算法 2 给出了基于正交实验设计的 ABC 算法的伪代码,其中,*SN* 为食物源的数量,*FEs* 表示适应度函数的评估次数,*MaxFEs* 为适应度函数的最大评估次数,$trial_i$ 记录了食物源 X_i 的适应值未更新的次数.从中可看出:第 31 行和第 32 行是应用 OED 方法所在部分,而其他部分与标准 ABC 完全相同.

算法 2. ABC-OED.

```
1.  Randomly generate SN food sources {X_i|i=1,2,…,SN};
2.  FEs=SN;
3.  while FEs≤MaxFEs do
4.  /*Employed bee phase*/
5.    for i=1 to SN do
6.      Generate a new candidate solution V_i according to Eq.(1);
7.      if f(V_i)<f(X_i) then
8.        Replace X_i with V_i;
9.        trial_i=0;
10.     else
11.       trial_i=trial_i+1;
12.     end
13.     FEs=FEs+1;
14.   end
15. /*Onlooker bee phase*/
16.   Calculate the probability p_i according to Eq.(2);
17.   for i=1 to SN do
18.     Choose a food source X_j from the current population P by the roulette wheel selection mechanism;
19.     Generate a new candidate solution V_j according to Eq.(1);
20.     if f(V_j)<f(X_j) then
21.       Replace X_j with V_j;
22.       trial_j=0;
23.     else
24.       trial_j=trial_j+1;
25.     end
26.     FEs=FEs+1;
27.   end
```

```
/*Scout bee phase*/
  if max(trial_i)>limit then
    trial_j=0;
    Employ OED to generate M new individuals C_i and a predict individual P;
    Pick out the best one from C_i and P as the new food source for the scout bee;
    FEs=FEs+M+1;
  end
end
```

4 数值实验

为验证本文方法,在 16 个典型的测试函数上进行了一系列实验,包括:

- OED 方法有效性分析:
  1) 结果精度比较;
  2) 收敛速度比较;
  3) 维度扩展性比较;
  4) 运行时间比较;
- *limit* 参数分析;
- 与变异方法比较;
- 应用 OED 方法改进其他 ABC 算法.

4.1 测试函数

为验证 OED 方法的有效性,我们采用 16 个典型的测试函数进行实验,包括单峰、多峰以及偏移加旋转类型的函数.表 3 给出了这些函数的简要介绍,包括函数名称、搜索空间范围以及最优值.F01~F08 是单峰类型的函数,其中,F06 和 F07 是带偏移的单峰函数;F09~F16 是多峰类型的函数,其中,F13~F16 是带偏移和旋转的多峰函数.这些函数的具体定义见文献[35−37].

Table 3 16 test functions used in the experiments

表 3 实验中使用的 16 个测试函数

Function	Name	Search range	Global optimum $f(X^*)$
F01	Schwefel 2.21	$[-100,100]^D$	0
F02	Step	$[-100,100]^D$	0
F03	Elliptic	$[-100,100]^D$	0
F04	SumPower	$[-1,1]^D$	0
F05	Exponential	$[-1.28,1.28]^D$	0
F06	Shifted sphere function	$[-100,100]^D$	−450
F07	Schwefel's problem 2.6 with Global optimum on bounds	$[-100,100]^D$	−310
F08	Quartic with noise	$[-1.28,1.28]^D$	0
F09	Rastrigin	$[-5.12,5.12]^D$	0
F10	Griewank	$[-600,600]^D$	0
F11	NCRastrigin	$[-5.12,5.12]^D$	0
F12	Bohachevsky 2	$[-100,100]^D$	0
F13	Shifted rotated Ackley's function with Global optimum on bounds	$[-32,32]^D$	−140
F14	Shifted Rastrigin's Function	$[-5,5]^D$	−330
F15	Shifted expanded Griewank's plus Rosenbrock's function	$[-3,1]^D$	−130
F16	Shifted rotated expanded Scaffer's F6 function	$[-100,100]^D$	−300

4.2 OED方法有效性分析

为验证 OED 方法是否有助于提升 ABC 的性能,将其与标准 ABC 算法进行多方面的比较.为便于区别,将标准 ABC 记为 ABC-rand,表示侦察蜂采用随机初始化的方法.从结果精度、收敛速度、维度扩展性以及运行时间

这 4 个方面来全面比较 ABC-rand 和 ABC-OED.为公平比较,本文实验中对两者的食物源数量 *SN* 和 *limit* 这两种公共参数采用相同的设置.对 *SN* 而言,不同的研究工作给出的参考值也是不同的,例如:文献[1,38,39]给出的值为 10,文献[19,23,24]给出的值为 25,文献[16]给出的值为 30,文献[13,40,41]给出的值为 40,而文献[12,17,20,26,42]给出的值为 50.显然,对于 *SN* 的设置,研究人员没有达成共识,但可得出的是,*SN* 的取值范围为[10,50].因此,本文采用该范围的中间值,即,设置 *SN* 为 30.而对于 *limit* 的取值,因为该参数直接影响到 OED 方法的执行频率,在第 4.3 节中对该参数进行了详细分析,在此先设置其值为 100.测试函数的维度设为 D=30,适应度函数的最大评估次数 Max*FEs* 设为 100 000[26].对 ABC-OED,使用正交表 $L_{25}(5^6)$.每种算法在所有函数上独立运行 30 次,记录结果的均值(mean)与标准差(SD),其中,结果是指算法得到的最终值 $f(X)$和函数的最优值 $f(X^*)$之差,即 $f(X)-f(X^*)$.

4.2.1 结果精度比较

这一节验证 OED 方法对 ABC 的结果精度的影响.在测试函数为 D=30 的情况下,表 4 给出了 ABC-rand 和 ABC-OED 的实验结果,其中,Best, Median,Worst 分别表示 30 次结果中的最优值、中间值、最差值.Wilcoxon 表示应用显著性水平 α=0.05 的 Wilcoxon 轶和检验的结果,+,–,≈分别表示 ABC-OED 的性能要优于、劣于、相当于 ABC-rand,使得实验结果更具统计意义[43,44].

Table 4 Experimental results of ABC-rand and ABC-OED at D=30

表 4 在 D=30 时,ABC-rand 和 ABC-OED 的实验结果

Function	Algorithm	Mean	SD	Best	Median	Worst	Wilcoxon
F01	ABC-rand	3.17E+01	4.91E+00	1.95E+01	3.30E+01	3.89E+01	+
	ABC-OED	7.38E+00	1.90E+00	2.92E+00	8.57E+00	1.09E+01	
F02	ABC-rand	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	≈
	ABC-OED	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	
F03	ABC-rand	5.31E−10	8.48E−10	3.92E−12	2.45E−10	3.21E−09	+
	ABC-OED	2.18E−11	3.55E−11	2.83E−13	1.14E−11	1.77E−10	
F04	ABC-rand	1.12E−10	2.30E−10	4.90E−14	2.49E−10	1.11E−09	+
	ABC-OED	2.12E−20	3.37E−20	1.57E−22	2.15E−21	1.34E−19	
F05	ABC-rand	1.05E−06	9.08E−07	1.25E−11	1.75E−06	3.34E−06	+
	ABC-OED	1.00E−09	1.85E−09	1.41E−11	4.37E−10	6.44E−09	
F06	ABC-rand	1.84E−13	4.57E−14	1.14E−13	1.71E−13	2.84E−13	+
	ABC-OED	5.49E−14	1.02E−14	0.00E+00	5.68E−14	5.68E−14	
F07	ABC-rand	1.04E+04	1.43E+03	6.67E+03	1.17E+04	1.29E+04	+
	ABC-OED	3.45E+03	6.24E+02	1.88E+03	3.62E+03	4.79E+03	
F08	ABC-rand	1.86E−01	4.50E−02	8.48E−02	1.26E−01	2.67E−01	+
	ABC-OED	5.97E−03	2.82E−03	1.70E−03	6.13E−03	1.26E−02	
F09	ABC-rand	1.04E−14	1.57E−14	0.00E+00	1.42E−14	7.46E−14	+
	ABC-OED	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	
F10	ABC-rand	2.47E−04	1.33E−03	0.00E+00	1.82E−13	7.40E−03	+
	ABC-OED	9.33E−16	3.66E−15	0.00E+00	0.00E+00	1.87E−14	
F11	ABC-rand	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	≈
	ABC-OED	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	
F12	ABC-rand	7.33E−16	9.89E−16	0.00E+00	6.66E−16	5.00E−15	≈
	ABC-OED	6.96E−16	8.65E−16	0.00E+00	2.78E−16	3.44E−15	
F13	ABC-rand	2.10E+01	6.22E−02	2.08E+01	2.10E+01	2.11E+01	+
	ABC-OED	2.06E+01	7.34E−02	2.04E+01	2.05E+01	2.07E+01	
F14	ABC-rand	2.52E−07	1.33E−06	3.41E−13	7.82E−11	7.43E−06	+
	ABC-OED	2.95E−08	1.57E−07	1.14E−13	4.90E−11	8.75E−07	
F15	ABC-rand	2.03E+00	2.37E−01	1.30E+00	1.70E+00	2.34E+00	+
	ABC-OED	1.22E+00	1.75E−01	7.77E−01	1.42E+00	1.58E+00	
F16	ABC-rand	1.32E+01	1.87E−01	1.25E+01	1.34E+01	1.35E+01	+
	ABC-OED	1.24E+01	3.97E−01	1.16E+01	1.27E+01	1.31E+01	

从表 4 可以看出:ABC-OED 在 13 个函数上的性能要显著地优于 ABC-rand;而在其余的 3 个函数(F02,F11,F12)上,两者性能相当.具体而言,在 F02 和 F11 上,ABC-rand 和 ABC-OED 均取得了全局最优值 0;而在 F09 上,仅有 ABC-OED 取得了全局最优值.对 F04 和 F10,ABC-OED 的精度比 ABC-rand 有大幅度提高.F13 是带偏移

和旋转的多峰类型函数,其全局最优解位于搜索空间的边界上,求解难度非常大,而 ABC-OED 的最差值要优于 ABC-rand 的最优值,这也表明 OED 方法有效地提高了 ABC 的搜索能力.为更好地分析 ABC-rand 和 ABC-OED 的性能稳定性,在图 4 中用盒图(box plot)的形式给出了结果的分布情况.盒图是经济学领域统计分析的重要工具,它可以很好地反映数据的统计分布情况[45].本文用该工具来分析 ABC-rand 和 ABC-OED 独立运行 30 次得到的结果的统计特性.从图 4 中可看出,ABC-OED 在大部分函数(如 F03~F10)上的结果比 ABC-rand 分布更集中,表明 ABC-OED 的鲁棒性更好.

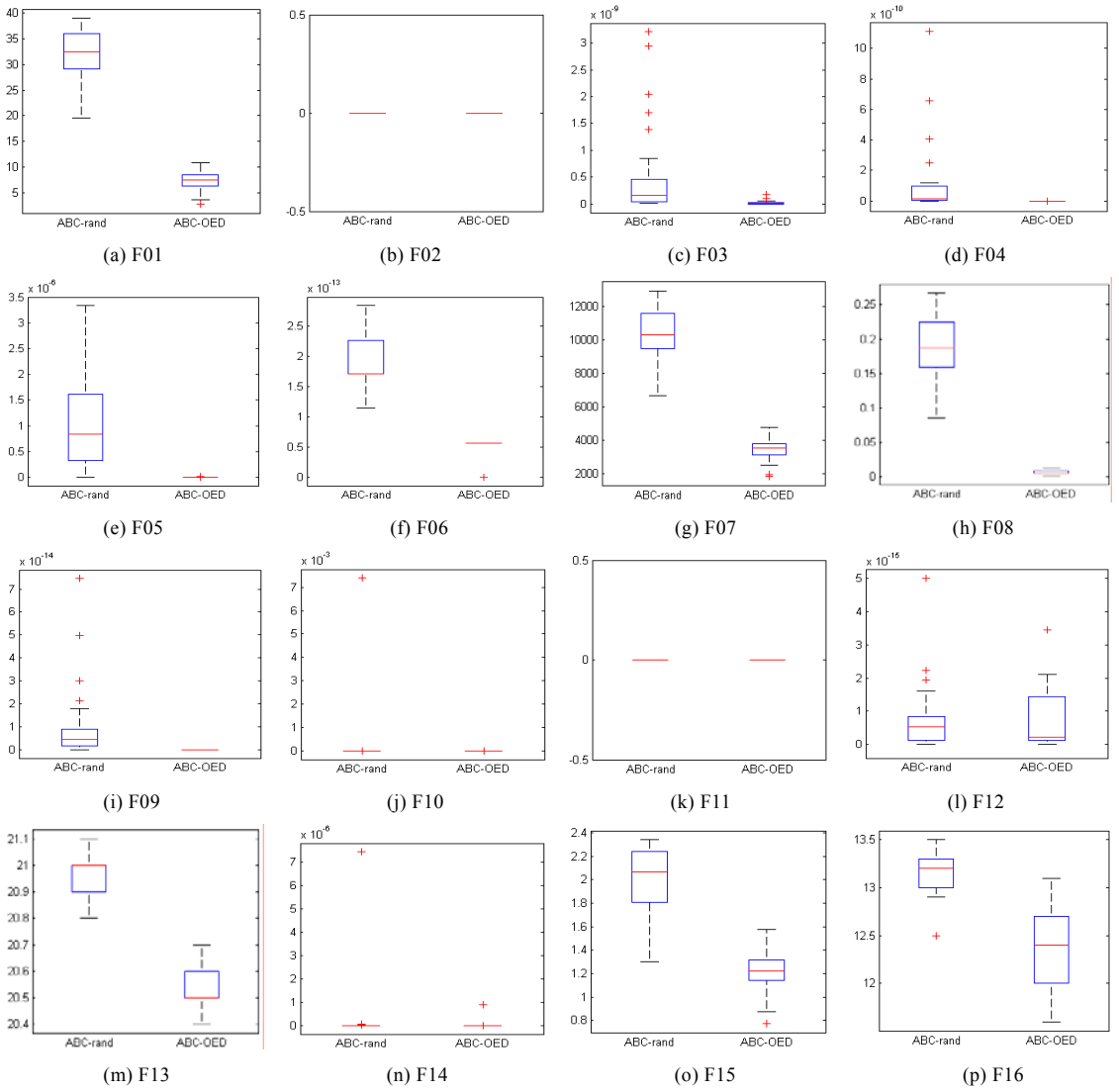


Fig.4 Box plots of the results obtained by ABC-rand and ABC-OED over 30 independent runs

图 4 ABC-rand 和 ABC-OED 独立运行 30 次结果的盒图

4.2.2 收敛速度比较

这一节验证 OED 方法对 ABC 的收敛速度的影响.为量化算法的收敛速度,我们对每个测试函数设定了一个对应的精度,见表 5 中的 Threshold,测试函数的维度 D 和其他参数设置与上一节保持不变.若算法在 Max*FEs* 内达到了设定的精度,则认为算法此次运行是成功的,记录此时消耗的适应度函数评估次数;否则,则认为此次

运行不成功.ABC-rand 和 ABC-OED 在所有函数上独立运行 30 次,记录消耗的适应度函数评估次数的平均值(mean FEs)、标准差(SD FEs)以及算法的成功率(SR).

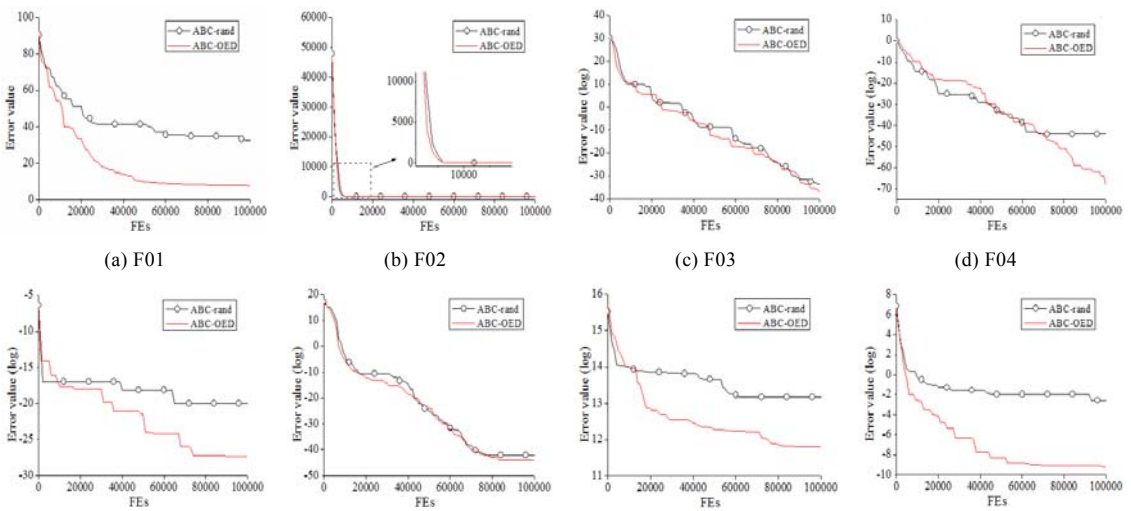
表 5 给出了最终结果,较优的结果用粗体突显,"NA"表示算法在对应的函数上未运行成功.

Table 5 Given predefined accuracy level, the algorithms' results of mean number of FEs and successful rate

表 5 在给定精度的情况下,算法的适应度函数评估次数的平均值及成功率

Function	Threshold	ABC-rand		ABC-OED	
		Mean FEs±SD FEs	SR (%)	Mean FEs±SD FEs	SR (%)
F01	5.00E+01	1.60E+04±3.83E+03	100.00	**5.33E+03±2.04E+03**	100.00
F02	1.00E−06	1.57E+04±6.43E+03	100.00	**8.21E+03±1.06E+03**	100.00
F03	1.00E−06	7.65E+04±4.51E+03	100.00	**6.90E+04±4.67E+03**	100.00
F04	1.00E−06	**2.38E+04±3.18E+03**	100.00	2.39E+04±4.19E+03	100.00
F05	1.00E−06	4.48E+04±3.54E+04	56.70	**2.70E+04±9.21E+03**	100.00
F06	1.00E−06	4.62E+04±2.92E+03	100.00	**4.46E+04±3.05E+03**	100.00
F07	1.00E+04	7.05E+04±5.42E+04	43.30	**2.09E+04±3.60E+03**	100.00
F08	1.00E−02	NA	0.00	**4.15E+04±1.88E+04**	83.30
F09	1.00E−06	**5.79E+04±6.98E+03**	100.00	5.86E+04±7.63E+03	100.00
F10	1.00E−06	**5.08E+04±1.11E+04**	96.70	5.16E+04±6.06E+03	100.00
F11	1.00E−06	1.55E+04±3.37E+03	100.00	**1.25E+04±3.06E+03**	100.00
F12	1.00E−06	5.35E+04±3.29E+03	100.00	**5.15E+04±4.08E+03**	100.00
F13	5.00E+01	NA	0.00	NA	0.00
F14	1.00E−06	8.25E+04±1.76E+04	96.70	**8.00E+04±8.97E+03**	100.00
F15	1.00E+01	1.62E+03±4.41E+02	100.00	**1.53E+03±3.62E+02**	100.00
F16	2.00E+01	NA	0.00	NA	0.00

从表 5 中可看出:ABC-OED 在大部分函数上比 ABC-rand 消耗的适应度函数评估次数更少,即收敛速度更快.同时,就算法的成功率 SR 而言,ABC-OED 更高,如在函数 F05,F08,F10 以及 F14 上.OED 方法有助于提高收敛速度的原因在于:随着演化代数的增加,整个群体的搜索空间范围不断缩小,算法的搜索行为也从勘探(exploration)逐步地转向为开采(exploitation),而 ABC-rand 的随机初始化方法具有较大的盲目性,使得新食物源的位置可能处于已经缩小的搜索空间之外,导致一定程度上的无效搜索,从而降低收敛速度.相比之下,ABC-OED 生成的新食物源能够保留被放弃食物源和全局最优个体 *gbest* 的有益信息,在搜索空间中可能处于较好的位置,有助于加快算法收敛.为更直观地对比 ABC-rand 和 ABC-OED 的收敛情况,图 5 描述了两种算法在所有函数上的收敛曲线.从这些收敛曲线图中也可看出,ABC-OED 在大部分函数上的收敛速度明显快于 ABC-rand.

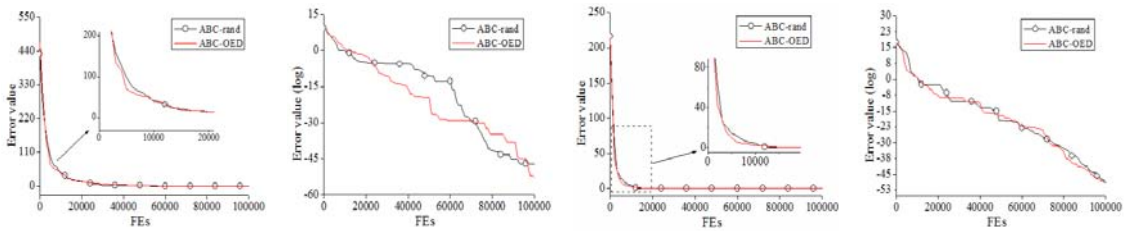


(a) F01 (b) F02 (c) F03 (d) F04

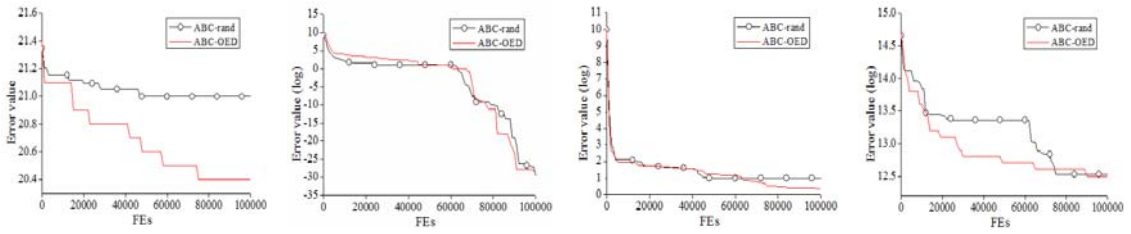
(e) F05 (f) F06 (g) F07 (h) F08

Fig.5 Convergence curves of ABC-rand and ABC-OED on all the test functions

图 5 ABC-rand 和 ABC-OED 在所有测试函数上的收敛曲线



(i) F09 (j) F10 (k) F11 (l) F12



(m) F13 (n) F14 (o) F15 (p) F16

Fig.5 Convergence curves of ABC-rand and ABC-OED on all the test functions (Continued)

图 5 ABC-rand 和 ABC-OED 在所有测试函数上的收敛曲线(续)

4.2.3 维度扩展性比较

对演化算法而言,测试函数维度的增加会使得问题的复杂度也急剧增加,从而导致算法的性能下降.为此,这一节验证测试函数的维度对 OED 方法性能的影响.我们将函数从 30 维扩展到 50 维,ABC-rand 和 ABC-OED 的参数设置与前面保持相同,而适应度函数的最大评估次数 Max*FEs* 设为 200 000.每个算法在所有函数上同样独立运行 30 次,在表 6 中给出实验结果.可以看出:在所有函数上 ABC-OED 的性能均要显著地优于 ABC-rand.特别地,在 F02 和 F11 上,ABC-rand 在 30 维时能获得全局最优值 0,而当 50 维时,ABC-rand 却未能得到全局最优值;但对 ABC-OED 而言,它依然能够获得全局最优值,并且在 F09,F10 以及 F12 上,ABC-OED 也能够获得全局最优值.总之,虽然测试函数的维度增加了,但 ABC-OED 的性能却依然高效,比 ABC-rand 更加优秀.

Table 6 Experimental results of ABC-rand and ABC-OED at *D*=50

表 6 *D*=50 时,ABC-rand 和 ABC-OED 的实验结果

Function	Algorithm	Mean	SD	Best	Mean	Worst	Wilcoxon
F01	ABC-rand	5.79E+01	3.75E+00	4.99E+01	5.15E+01	6.51E+01	+
	ABC-OED	2.01E+01	2.36E+00	1.59E+01	1.94E+01	2.52E+01	
F02	ABC-rand	2.33E−01	4.23E−01	0.00E+00	0.00E+00	1.00E+00	+
	ABC-OED	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	
F03	ABC-rand	2.97E−12	5.09E−12	1.15E−15	5.87E−14	1.90E−11	+
	ABC-OED	2.32E−15	2.25E−15	4.77E−17	8.60E−16	8.88E−15	
F04	ABC-rand	4.63E−08	4.78E−08	2.44E−09	1.01E−08	1.82E−07	+
	ABC-OED	2.76E−29	1.46E−28	4.31E−34	5.14E−30	8.14E−28	
F05	ABC-rand	6.11E−07	6.68E−07	1.25E−09	3.85E−07	3.26E−06	+
	ABC-OED	7.07E−13	2.71E−12	6.66E−16	1.38E−14	1.51E−11	
F06	ABC-rand	4.36E−13	7.95E−14	3.41E−13	5.12E−13	6.82E−13	+
	ABC-OED	9.09E−14	2.78E−14	5.68E−14	5.68E−14	1.14E−13	
F07	ABC-rand	2.38E+04	1.65E+03	1.91E+04	2.62E+04	2.68E+04	+
	ABC-OED	6.90E+03	1.01E+03	4.08E+03	7.42E+03	8.50E+03	
F08	ABC-rand	4.70E−01	7.82E−02	3.61E−01	3.73E−01	6.90E−01	+
	ABC-OED	1.50E−02	3.72E−03	9.43E−03	1.83E−02	2.59E−02	
F09	ABC-rand	7.84E−13	3.83E−12	1.78E−15	1.24E−14	2.14E−11	+
	ABC-OED	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	
F10	ABC-rand	4.76E−11	2.44E−10	0.00E+00	1.03E−11	1.36E−09	+
	ABC-OED	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	

Table 6 Experimental results of ABC-rand and ABC-OED at D=50 (continued)

表 6 D=50 时,ABC-rand 和 ABC-OED 的实验结果(续)

F11	ABC-rand	3.17E−11	1.67E−10	0.00E+00	0.00E+00	9.34E−10	+
	ABC-OED	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	
F12	ABC-rand	6.29E−17	7.41E−17	0.00E+00	0.00E+00	2.22E−16	+
	ABC-OED	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	
F13	ABC-rand	2.11E+01	4.23E−02	2.10E+01	2.11E+01	2.12E+01	+
	ABC-OED	2.05E+01	6.91E−02	2.03E+01	2.04E+01	2.06E+01	
F14	ABC-rand	1.35E−06	7.12E−06	4.55E−13	1.01E−11	3.97E−05	+
	ABC-OED	1.10E−13	2.51E−14	5.68E−14	1.14E−13	1.71E−13	
F15	ABC-rand	3.99E+00	3.57E−01	3.04E+00	4.18E+00	4.62E+00	+
	ABC-OED	1.94E+00	4.21E−01	1.13E+00	1.96E+00	2.64E+00	
F16	ABC-rand	2.28E+01	1.98E−01	2.22E+01	2.30E+01	2.32E+01	+
	ABC-OED	2.14E+01	6.90E−01	2.00E+01	2.07E+01	2.27E+01	

4.2.4 运行时间比较

本节验证 OED 方法对 ABC 运行时间的影响.与 ABC-rand 相比,ABC-OED 的唯一不同在于侦察蜂采用 OED 方法来生成食物源.OED 的执行过程主要包含两个部分:1) 按公式(8)和公式(9)将被放弃食物源和全局最优个体进行因素分组和水平划分;2) 按正交表来安排实验,即:构造对应的 M 个新个体,并用因素分析来构造预测个体 P.通常,演化算法的运行时间包含两个方面:1) 算法的操作算子的运行时间;2) 评估适应度函数的时间[36,46].因此,可以将 OED 的执行过程视为算法的一个操作算子来分析其对 ABC 运行时间的影响.设评估一次适应度函数的时间为σ个单位时间,ABC 的雇佣蜂和观察蜂阶段执行一次的时间为$(\mu\cdot\sigma)$个单位时间,OED 执行一次的时间为$(\lambda\cdot\sigma)$个单位时间.需要指出的是:μ和λ的值是和优化问题的类型相关的,复杂度越高的问题,它们的值越小,即,算法的操作算子占整个运行时间的比例越小;反之亦然.因此,以适应度函数的最大评价次数 $MaxFEs$ 为停止条件时,若 ABC-rand 的运行时间是 $T_{\text{ABC-rand}}=MaxFEs\cdot\sigma+\alpha\cdot\mu\cdot\sigma$,那么 ABC-OED 是:

$$T_{\text{ABC-OED}}=\text{Max}FEs\cdot\sigma+\beta\cdot\mu\cdot\sigma+\omega\cdot\lambda\cdot\sigma.$$

α和β分别是 ABC-rand 和 ABC-OED 的雇佣蜂与观察蜂阶段的执行次数,ω是 OED 的执行次数.最后,可以通过计算两者运行时间的比例 $Ratio=T_{\text{ABC-OED}}/T_{\text{ABC-rand}}$ 来评估 OED 对 ABC 运行时间的影响.

下面以实验中使用的 16 个测试函数为例,量化 $Ratio$ 的值来评估 OED 对 ABC 运行时间的影响.ABC-rand 和 ABC-OED 的参数设置与前面保持相同,分别是 SN 设为 30,$limit$ 设为 100.D 为 30 维时,MaxFEs 设为 100 000;D 为 50 维时,MaxFEs 设为 200 000.表 7 给出了 ABC-rand 和 ABC-OED 在每个函数上的平均 CPU 时间.

Table 7 Average CPU time (in seconds) of ABC-rand and ABC-OED on the test functions (s)

表 7 ABC-rand 和 ABC-OED 在测试函数上的平均 CPU 时间(单位:s)

Function	D=30			D=50		
	ABC-rand	ABC-OED	*Ratio*	ABC-rand	ABC-OED	*Ratio*
F01	0.103	0.123	1.194	0.214	0.242	1.131
F02	0.184	0.172	0.935	0.532	0.451	0.848
F03	1.020	1.070	1.049	3.520	3.560	1.011
F04	0.973	1.010	1.038	3.350	3.430	1.024
F05	0.118	0.129	1.093	0.262	0.292	1.115
F06	0.100	0.110	1.106	0.210	0.221	1.052
F07	0.224	0.226	1.009	0.894	0.929	1.039
F08	1.060	1.090	1.028	3.540	3.580	1.011
F09	0.267	0.289	1.082	0.822	0.830	1.010
F10	0.382	0.404	1.058	1.100	1.080	0.982
F11	0.230	0.197	0.857	0.717	0.546	0.762
F12	0.432	0.440	1.019	1.450	1.340	0.924
F13	0.619	0.626	1.011	2.280	2.400	1.053
F14	0.271	0.286	1.055	0.805	0.804	0.999
F15	0.373	0.369	0.989	1.100	1.080	0.982
F16	0.600	0.588	0.980	2.220	2.290	1.032
Overall	6.956	7.129	1.025	23.016	23.075	1.003

每种算法在所有函数上独立运行 30 次,记录使用的平均 CPU 时间.算法的运行平台如下:操作系统:Windows 7(×64);CPU:Intel Core 2 Quad CPU Q8200(2.33GHz);内存:4G;编程语言:Java;开发工具:Eclipse SDK 4.2.0.

从表 7 可以看出:当 D=30 时,ABC-OED 和 ABC-rand 在所有函数上的 CPU 时间之比的变化范围是[0.857, 1.194],总的 CPU 时间之比是 1.025.值得注意的是:在 4 个函数上(F02,F11,F15,F16)的 CPU 时间之比低于 1,说明 ABC-OED 在这些函数的运行时间反而低于 ABC-rand.出现这种现象的可能原因有两个方面:1) 函数的复杂度较低,用较少的适应度函数评估次数就能求出全局最优解,使得食物源在之后的算法迭代过程中适应值不再更新,从而成为被放弃的食物源;2) 函数的复杂度较高,即使适应度函数的评估次数达到了最大值也难求出全局最优解,使得食物源在算法迭代过程中难以更新其适应值,从而成为被放弃的食物源.这两种原因的共同特点就是被放弃食物源的数量增多,使得 OED 方法被执行的频率也相应增加,导致需要消耗更多的适应度函数评估次数.在这种情况下,雇佣蜂和观察蜂阶段被执行的次数就会相应减少,意味着算法在这两个阶段的时间开销会降低,使得 ABC-OED 的运行时间可能会小于 ABC-rand.从图 5 的收敛曲线图中可以看到:在 F02.F11.F15 以及 F16 这 4 个函数上,算法在迭代初期的收敛曲线下降速度非常快,但中后期的变化幅度却小很多.这表明算法在中后期的迭代过程中,食物源的适应值未能一直持续更新,导致被放弃食物源的数量增加,验证了前面给出的原因.

当维度从 30 增至 50 时,函数的复杂度也随之增加,从而使得算法的运行时间也相应增加.此时,总的 CPU 时间之比是 1.003,说明随着函数的复杂度增加,OED 方法对 ABC 运行时间的影响反而降低了,这也验证了本节开始部分的分析.总之,OED 方法对 ABC 运行时间的影响是微小的,在可接受的范围内.

4.3 *limit*参数分析

在 ABC 中,被放弃食物源的认定和 *limit* 参数是紧密相关的,因而 OED 方法的性能也是与 *limit* 参数相关的.*limit* 参数的值越小,食物源成为被放弃食物源的概率越大,那么 OED 方法的执行频率就越高.通常,*limit* 参数有 3 种典型的设置,分别是:1) *limit*=100[8,39,41];2) *limit*=200[14,16,18,26,40,47];3) *limit*=$SN\times D$[1,6,12,23].本节中,为验证 *limit* 参数的影响,对 4 种不同的设置进行实验,分别是:1) *limit*=50;2) *limit*=100;3) *limit*=200;4) *limit*=$SN\times D$.实验中,ABC-OED 的参数设置与第 4.2 节保持相同.函数的维度 D 设为 30,适应度函数的最大评估次数 *MaxFEs* 设为 100 000.表 8 给出了使用不同的 *limit* 参数值,ABC-OED 在所有测试函数上的结果.每种参数值在所有函数上独立测试 30 次,记录结果的均值和标准差,最好的结果用粗体突显.从中可以看出:当 *limit*=100 时,OED 方法在 11 个函数上的结果是最优的,取得了最佳的整体性能.

Table 8 Experimental results of ABC-OED with four different settings of parameter *limit*

表 8 4 种 *limit* 参数值的 ABC-OED 的实验结果

Function	*limit*=50 Mean±SD	*limit*=100 Mean±SD	*limit*=200 Mean±SD	*limit*=*SN*×*D* Mean±SD
F01	2.83E+01±6.26E+00	**7.38E+00±1.90E+00**	8.38E+00±1.51E+00	7.71E+00±2.26E+00
F02	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F03	2.98E−10±3.83E−10	2.18E−11±3.55E−11	**4.96E−12±5.61E−12**	5.12E−12±5.18E−12
F04	**7.79E−27±1.43E−26**	2.12E−20±3.37E−20	1.39E−17±2.89E−17	3.95E−17±1.21E−16
F05	9.51E−07±1.13E−06	**1.00E−09±1.85E−09**	8.47E−07±1.01E−06	7.88E−07±8.18E−07
F06	6.82E−14±2.27E−14	**5.49E−14±1.02E−14**	7.39E−14±2.60E−14	1.27E−13±2.40E−14
F07	4.46E+03±1.03E+03	**3.45E+03±6.24E+02**	6.65E+03±1.19E+03	1.06E+04±1.40E+03
F08	2.51E−02±8.79E−03	**5.97E−03±2.82E−03**	1.60E−02±5.39E−03	5.78E−02±1.11E−02
F09	5.27E−15±2.71E−14	**0.00E+00±0.00E+00**	2.56E−14±1.27E−13	3.13E−13±1.67E−12
F10	**0.00E+00±0.00E+00**	9.33E−16±3.66E−15	4.14E−15±2.00E−14	5.77E−04±2.18E−03
F11	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
F12	1.25E−15±1.76E−15	6.96E−16±8.65E−16	**2.00E−16±2.67E−16**	4.11E−16±5.50E−16
F13	2.09E+01±6.44E−02	**2.06E+01±7.34E−02**	2.09E+01±7.20E−02	2.09E+01±7.72E−02
F14	**3.01E−12±1.12E−11**	2.95E−08±1.57E−07	2.28E−09±8.96E−09	1.52E−08±6.82E−08
F15	1.42E+00±2.28E−01	**1.22E+00±1.75E−01**	1.51E+00±1.73E−01	1.54E+00±1.21E−01
F16	1.30E+01±2.35E−01	**1.24E+01±3.97E−01**	1.30E+01±2.07E−01	1.31E+01±2.37E−01

为在统计意义上比较多种算法的性能,按文献[43,44]的建议,在表 9 中采用 Friedman 检验对这 4 种不同参

数值的最终结果进行了排名,先后顺序依次是 100,50,200 以及 *SN*×*D*.

Table 9 Average rankings of four different settings of parameter *limit*

表 9 4 种 *limit* 参数值的平均排名

limit	Average rankings
50	1.88
100	**1.37**
200	3.34
SN×*D*	3.41

为进一步挖掘 *limit* 参数与 OED 方法的内在联系,表 10 给出了 OED 方法在每个函数上的平均执行次数.可以看出:总的变化趋势是 *limit* 的值越小,执行频率越高.当 *limit* 为 50 时,OED 在所有函数上的平均执行次数是 503.52 次,消耗的适应度函数评估次数占总次数的比例见公式(12).

$$\frac{503.52\times M}{\text{Max}FEs}=\frac{503.52\times 25}{100000}=12.58\% \tag{12}$$

而当 *limit* 增至 *SN*×*D* 时,平均执行次数是 4.13,所占比例为 0.03%.这说明:要保证 OED 的有效性,需要消耗一定的适应度函数评估次数,但其占用的比例也不宜过大,因而 *limit* 的值设为 100 是合适的.

Table 10 Average number of times of running OED

表 10 OED 方法的平均执行次数

Function	*limit*=50	*limit*=100	*limit*=200	*limit*=*SN*×*D*
F01	1010	318	49	0.03
F02	855	567	294	59
F03	250	83.3	19.4	0
F04	171	39.7	3.17	0
F05	1030	365	354	57.5
F06	277	144	57.1	0.03
F07	360	87.9	41.9	0.93
F08	789	430	174	4.43
F09	61.2	10.7	1.93	0
F10	150	41.2	7.33	0
F11	856	452	199	33.8
F12	103	30.5	4.3	0
F13	1010	149	321	31.7
F14	28.1	1.37	0.07	0
F15	241	47.6	18.3	0
F16	865	122	178	4.13
Average	503.52	180.58	107.66	11.97
Ratio	12.58%	4.51%	2.69%	0.03%

4.4 与变异方法比较

OED 方法的作用是帮助侦察蜂快速有效地找到新的食物源,保存搜索经验,避免局部极值的影响.然而对演化算法而言,避免局部极值的常用手段是采用变异方法.因此在本节中,将 OED 方法与 3 种典型的变异方法进行对比,分别是:

- 高斯变异[48,49]:

$$x_j^*=x_j+gaussian_j() \tag{13}$$

- 柯西变异[50−52]:

$$x_j^*=x_j+cauchy_j() \tag{14}$$

- 差分演化变异[53]:

$$v_j=x_i+F\cdot(x_{r1,j}-x_{r2,j})+F\cdot(x_{r3,j}-x_{r4,j}) \tag{15}$$

$$x_j^*=\begin{cases}v_j, \text{ if } rand\leqslant CR \text{ or } j_{rand}=j\\ x_j, \text{ otherwise}\end{cases} \tag{16}$$

公式(13)~公式(16)中:

- x_j 是被放弃食物源 $X=(x_1,x_2,\ldots,x_D)$在第 j 维的分量;
- x_j^* 是对 X 进行变异操作后生成的新食物源 $X^*=(x_1^*,x_2^*,...,x_D^*)$ 在第 j 维的分量;
- $gaussian_j()$是标准正太分布生成的随机数;
- $cauchy_j()$是尺度参数 t=1 的柯西分布生成的随机数;
- F 是尺度系数,$r1$,$r2$,$r3$ 和 $r4$ 是区间[1,SN]上互不相等的随机整数,代表不同食物源的下标;
- *rand* 是[0,1]区间的随机数;
- CR 为交叉概率;
- j_{rand} 是区间[1,D]上随机生成的整数,用于保证 X^*至少有一维不同于 X;
- F 和 CR 的取值按文献[53]的推荐,分别设置为 1 和 0.1.

在侦察蜂阶段,分别采用这 3 种变异方法的 ABC 算法简记为 ABC-Gaussian,ABC-Cauchy 以及 ABC-DE.实验中,测试函数的维度设为 30,适应度函数的最大评估次数为 100 000,*SN* 和 *limit* 的设置与第 4.2 节保持相同.因此,在本节中包含了 4 种算法的比较,分别是 ABC-Gaussian,ABC-Cauchy,ABC-DE 以及 ABC-OED.实验中,记录这 4 种算法独立运行 30 次后的均值和标准差.采用显著性水平α=0.05 的 Wilcoxon 轶和检验来统计算法的性能是否存在显著性差异.

表 11 给出了最终结果,可以看出:OED 方法在大部分函数上的结果要显著地优于其他 3 种变异方法.具体而言:与 ABC-Gaussian 相比,ABC-OED 仅在 1 个函数上(F14)的性能要差一些,但在其他的 10 个函数上,ABC-OED 的性能则更优;在 F14 上,ABC-OED 的性能也要劣于 ABC-DE.注意到:ABC-Gaussian 和 ABC-DE 的扰动尺度相对于 ABC-Cauchy 是略小的,而函数 F14 的搜索空间中遍布了局部极值点,且这些局部极值点之间的距离非常近,被放弃的食物源很可能位于某个局部极值点上,若采用变异的方法对其进行微小的扰动,可能恰好让生成的新食物源跳入另一个情况较好的局部极值点上,因此,ABC-Gaussian 和 ABC-DE 在该函数上的性能要好于 ABC-OED.图 6 给出了函数 F14 为二维时的 3-D 图.

Table 11 Compared results between OED and mutation-based approaches

表 11 OED 方法与变异方法的比较结果

Function	ABC-Gaussian Mean±SD	ABC-Cauchy Mean±SD	ABC-DE Mean±SD	ABC-OED Mean±SD
F01	7.61E+00±2.53E+00≈	1.22E+01±2.19E+00+	1.52E+01±2.08E+00+	7.38E+00±1.90E+00
F02	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
F03	2.50E−11±4.70E−11≈	4.67E−11±9.08E−11+	9.65E−13±1.21E−12−	2.18E−11±3.55E−11
F04	1.14E−10±1.84E−10+	1.11E−10±1.42E−10+	2.34E−17±4.82E−17+	2.12E−20±3.37E−20
F05	9.26E−07±8.21E−07+	1.02E−06±7.26E−07+	8.03E−07±6.56E−07+	1.00E−09±1.85E−09
F06	1.80E−13±4.17E−14+	1.82E−13±3.08E−14+	1.21E−13±1.93E−14+	5.49E−14±1.02E−14
F07	8.60E+03±1.28E+03+	7.93E+03±1.23E+03+	7.63E+03±1.20E+03+	3.45E+03±6.24E+02
F08	1.90E−01±4.46E−02+	1.96E−01±4.42E−02+	9.21E−02±1.67E−02+	5.97E−03±2.82E−03
F09	6.33E−14±2.05E−13+	2.21E−13±1.08E−12+	9.06E−15±1.62E−14+	0.00E+00±0.00E+00
F10	3.29E−04±1.77E−03+	5.80E−10±2.68E−09+	3.53E−04±1.90E−03+	9.33E−16±3.66E−15
F11	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
F12	2.22E−16±3.23E−16≈	3.39E−16±5.80E−16≈	9.62E−16±1.52E−16≈	6.96E−16±8.65E−16
F13	2.10E+01±6.87E−02+	2.10E+01±4.65E−02+	2.10E+01±4.16E−02+	2.06E+01±7.34E−02
F14	6.12E−09±2.12E−08−	9.48E−07±4.87E−06+	8.95E−10±4.73E−09−	2.95E−08±1.57E−07
F15	1.99E+00±2.08E−01+	1.91E+00±2.64E−01+	1.67E+00±1.85E−01+	1.22E+00±1.75E−01
F16	1.31E+01±2.06E−01+	1.31E+01±2.07E−01+	1.31E+01±2.27E−01+	1.24E+01±3.97E−01
+/≈/−	10/5/1	13/3/0	11/3/2	−/−

图 6 给出了函数 F14 为二维时的 3-D 图.

与 ABC-Cauchy 相比,ABC-OED 在所有函数上的性能均不劣于它,且在 13 个函数上更优.虽然与 ABC-DE 相比,ABC-OED 在 2 个函数上的性能更差,但在其他的 11 个函数上, ABC-OED 更优.

在表 12 中,采用 Friedman 检验给出了这 4 种算法的平均排名情况.可以看出,ABC-OED 的名次是最好的.

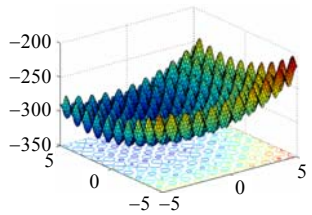


Fig.6 3-D plot of two-dimensional F14
图 6 函数 F14 为二维时的 3-D 图

Table 12 Average rankings of OED and mutation-based approaches
表 12 OED 方法与变异方法的平均排名

Algorithm	Average rankings
ABC-Gaussian	2.88
ABC-Cauchy	3.18
ABC-DE	2.44
ABC-OED	**1.50**

4.5 应用OED方法改进其他ABC算法

前面的实验已验证:OED 方法能够显著地提高标准 ABC 的性能,克服侦察蜂采用随机初始化方法的不足.在本节中,将验证 OED 方法是否有助于提高其他改进 ABC 算法的性能.为此,我们选择了 3 种代表性的算法,分别是:

- GABC[13]:*gbest* 引导的 ABC;
- ABC/best/1[20]:全局最优的 ABC;
- OCABC[26]:采用改进解搜索方程和正交学习的 ABC.

GABC 算法在解搜索方程中融入全局最优解 *gbest* 的信息;ABC/best/1 采用了类型于变异策略 DE/best/1 的解搜索方程 ABC/best/1;OCABC 采用了改进的解搜索方程,并在雇佣蜂阶段对随机选择的一个食物源进行正交学习.GABC 和 OCABC 的共同特点是在侦察蜂阶段均采用了随机初始化的方法,而 ABC/best/1 在侦察蜂阶段采用了混沌系统和反向学习的方法.为应用 OED 方法,我们对这 3 种改进的 ABC 算法进行了调整,在侦察蜂阶段均使用 OED 方法.相应地,使用了 OED 方法的这 3 种改进 ABC 算法分别简记为 GABC-OED,ABC/best/1-OED,OCABC-OED.为公平比较,GABC,ABC/best/1 以及 OCABC 的参数全部采用原文献的推荐设置,见表 13.而对 GABC-OED,ABC/best/1-OED 以及 OCABC-OED,除 *limit* 参数设为本文推荐的 100 之外,其他参数的设置与原文献保持相同.测试函数的维度设为 30,适应度函数的最大评估次数为 100 000.每种算法在所有函数上独立运行 30 次,记录结果的均值和标准差.

Table 13 Parameter settings for GABC, ABC/best/1, and OCABC
表 13 GABC,ABC/best/1 以及 OCABC 的参数设置

Algorithm	Parameter settings
GABC	SN=40, Ψ=1.5, $limit$=200
ABC/best/1	SN=50, $limit$=0.6×SN×D
OCABC	SN=50, $limit$=200

表 14 给出了这 6 种算法的最终结果,从中可看出:在应用 OED 方法之后,GABC,ABC/best/1 以及 OCABC 的性能均有提升.具体而言,GABC 和 ABC/best/1 在 11 个函数上的性能有显著提升,而 OCABC 在 12 个函数上的性能有显著提升.值得说明的是:这 3 种算法在应用 OED 方法之后,未出现在某一函数上的性能比原算法更差.这些实验结果表明:OED 方法具备较好普适性,可作为提高其他改进 ABC 算法性能的一个通用框架.

Table 14 Experimental results of the improved ABCs with OED

表 14 应用 OED 方法的改进 ABC 算法的结果

Function	GABC Mean±SD	ABC/best/1 Mean±SD	OCABC Mean±SD
F01	4.87E+01±4.89E+00+	1.43E+01±2.33E+00+	1.04E+01±1.36E+00+
F02	7.67E−01±8.44E−01+	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈
F03	3.16E−10±3.63E−10≈	2.10E−27±1.85E−27≈	2.99E−12±9.76E−12≈
F04	2.84E−09±4.49E−09+	8.14E−34±3.62E−33+	5.65E−17±1.85E−16+
F05	1.37E−05±1.52E−05+	4.74E−06±4.40E−06+	9.52E−06±8.65E−06+
F06	4.55E−13±2.08E−13≈	1.00E−13±2.40E−14+	7.58E−14±2.68E−14+
F07	9.67E+03±1.22E+03+	7.39E+03±1.03E+03+	8.97E+03±9.82E+02+
F08	2.72E−01±6.31E−02+	5.46E−02±1.19E−02+	3.91E−02±1.09E−02+
F09	1.61E−01±3.35E−01≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈
F10	1.11E−03±3.57E−03+	2.23E−07±1.20E−06+	8.00E−07±3.49E−06+
F11	3.76E−09±1.34E−08+	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈
F12	6.43E−11±4.70E−11≈	0.00E+00±0.00E+00≈	3.70E−18±1.99E−17+
F13	2.10E+01±6.45E−02+	2.10E+01±6.08E−02+	2.10E+01±5.94E−02+
F14	1.05E−01±2.58E−01≈	9.85E−14±2.91E−14+	8.53E−14±3.20E−14+
F15	3.93E+00±6.79E−01+	1.61E+00±2.90E−01+	1.84E+00±1.69E−01+
F16	1.33E+01±2.16E−01+	1.31E+01±2.51E−01+	1.33E+01±1.18E−01+
Function	GABC-OED Mean±SD	ABC/best/1-OED Mean±SD	OCABC-OED Mean±SD
F01	3.04E+00±1.03E+00	1.14E+01±2.42E+00	2.42E+00±5.71E−01
F02	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
F03	2.47E−10±2.18E−10	3.21E−27±2.48E−27	1.75E−12±6.61E−11
F04	1.12E−19±3.44E−19	2.13E−34±3.52E−34	1.82E−18±4.71E−18
F05	1.82E−09±2.55E−09	3.86E−09±4.14E−06	2.01E−07±2.21E−07
F06	3.71E−13±1.99E−13	5.49E−14±1.02E−14	5.31E−14±1.42E−14
F07	3.46E+03±5.77E+02	3.22E+03±5.29E+02	3.78E+03±5.92E+02
F08	5.11E−03±2.33E−03	2.45E−02±8.40E−03	3.14E−03±1.12E−03
F09	1.18E−01±3.03E−01	0.00E+00±0.00E+00	0.00E+00±0.00E+00
F10	3.32E−04±1.77E−03	2.23E−08±8.66E−08	7.22E−10±2.61E−09
F11	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
F12	7.50E−11±8.72E−11	0.00E+00±0.00E+00	0.00E+00±0.00E+00
F13	2.06E+01±1.04E−01	2.07E+01±8.35E−02	2.08E+01±8.33E−02
F14	1.13E−01±3.06E−01	5.31E−14±1.42E−14	4.55E−14±2.27E−14
F15	1.72E+00±2.43E−01	1.30E+00±3.37E−01	1.62E+00±1.82E−01
F16	1.23E+01±4.70E−01	1.23E+01±4.21E−01	1.22E+01±3.96E−01
+/≈/−	11/5/0	11/5/0	12/4/0

4.6 进一步讨论

通过上述一系列实验,验证了在侦察蜂阶段采用 OED 生成新的食物源可有效地改善 ABC 算法的性能,并且可作为提高其他改进 ABC 算法性能的通用框架.事实上,在其他类型的群体智能算法中,OED 同样也能够发挥出重要作用.在文献[28]中,Zhan 和 Zhang 等人针对 PSO 算法中存在的“两步前进,一步后退(two steps forward, one step back)”的现象,提出采用 OED 来构造一个位于 *pbest* 和 *gbest* 所在搜索区域之间的中间粒子,用该粒子来引导其他粒子的飞行,从而提高 PSO 算法的搜索效率.在文献[30]中,Leung 和 Wang 把 OED 方法应用到了遗传算法中,采用 OED 来生成初始群体,并设计了一种正交杂交算子,有效地提高了遗传算法的性能.在文献[31]中,蔡自兴等人提出了一种基于 OED 的约束优化进化算法,通过采用 OED 对多个父体进行交叉操作,增加了子代个体的采样范围,从而可提高算法的搜索能力.在文献[34]中,龚文引和蔡之华将 OED 应用到了多目标的 DE 算法中,他们在群体的初始化过程中引入 OED,使得个体可均匀地分布于搜索空间,提高了算法对初始群体的利用能力.这些研究工作有效地利用了 OED 能以较少计算开销而对搜索空间进行均匀局部搜索的能力,达到了改善相应算法性能的目的.类似于这些研究工作,本文采用 OED 的目的也是利用其具备的均匀局部搜索能力来提高 ABC 算法的性能,但又与它们有所区别的是:本文是从保存侦察蜂搜索经验的角度出发,提出采用 OED 来生成侦察蜂阶段的新食物源,使得新的食物源能在不同维度上继承被放弃食物源和全局最优解的有益信息,充分利用了 OED 能以较少计算开销达到均匀局部搜索的能力,从而避免了随机初始化方法易导致搜索经验丢失的

问题.

表 15 给出了 ABC-OED 与上述两种基于 OED 的改进算法的比较结果.这两种算法是:1) 正交学习的局部 PSO 算法(OLPSO-L)[28];2) 带量化技术的正交遗传算法(OGA/Q)[30].实验中,ABC-OED 的参数设置与第 4.2 节保持相同,测试函数的维度和适应度函数的最大评估次数与文献[28]保持相同,分别设为 30 和 200 000.为保持比较的公平性,OLPSO-L 和 OGA/Q 的实验结果直接摘自文献[28].ABC-OED 的实验结果为运行 30 次后的均值和标准差.从表 15 可看出:OGA/Q 和 OLPSO 在 5 个函数上取得了最好结果,表明了基于 OED 方法的有效性,特别是 OGA/Q 在前两个单峰函数上获得了全局最优解.与这两种算法相比,ABC-OED 在 7 个函数上取得了最好结果,并且可注意到,ABC-OED 在多峰函数上的性能是 3 种算法中最好的.可能的原因应该是:采用了 OED 方法的侦察蜂在一定程度上有助于算法跳出局部极值,从而可避免算法陷入局部最优.

Table 15 Comparison results between ABC-OED and other two OED-based algorithms

表 15 ABC-OED 与其他两种基于 OED 算法的比较结果

Function	Type	OLPSO-L Mean±SD	OGA/Q Mean±SD	ABC-OED Mean±SD
Sphere	Unimodal	1.11E−38±1.28E−38	**0.00E+00±0.00E+00**	2.70E−48±5.12E−48
Schwefel's P2.22	Unimodal	7.67E−22±5.63−22	**0.00E+00±0.00E+00**	2.52E−26±1.42E−26
Rosenbrock	Unimodal	1.26E+00±1.40E+00	7.50E−01±1.10E−01	**4.88E−01±1.10E+00**
Noise	Unimodal	1.64E−02±3.25E−03	6.30E−03±4.07E−04	**5.62E−03±2.18E−03**
Schwefel	Multimodal	**3.82E−04±0.00E+00**	3.03E−02±6.44E−04	**3.82E−04±9.09E−13**
Rastrigin	Multimodal	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
Ackley	Multimodal	4.14E−15±0.00E+00	**4.44E−16±3.98E−17**	1.52E−14±2.91E−15
Griewank	Multimodal	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
Generalized Penalized 1	Multimodal	**1.57E−32±2.79E−48**	6.01E−16±1.16E−06	**1.57E−32±5.47E−48**
Generalized Penalized 2	Multimodal	**1.35E−32±5.59E−48**	1.86E−04±2.61E−05	**1.35E−32±5.47E−48**

为进一步深入分析 OED 方法,将其与随机初始化方法同时用于侦察蜂生成新食物源,如果 OED 生成的食物源要优于随机初始化方法,则将该食物源用于替换被放弃的食物源;反之,则将随机初始化方法生成的食物源用于替换被放弃的食物源.实验中分别记录两种方法的替换次数,替换一次就视为相应方法成功执行了一次,以此来计算 OED 占总次数的成功比率(successful ratio,简称 SR).实验中对 D 为 30 维和 50 维的两种情况进行实验,参数设置与第 4.2 节保持相同,即:对应的适应度函数的最大评估次数分别为 100 000 和 200 000,SN 为 30,$limit$ 为 100.两种算法在每个测试函数上均运行 30 次,记录两种方法的平均成功执行次数和 OED 的平均成功比率.表 16 给出了实验结果,从该表中可以看出:相对于随机初始化方法,OED 在两种维度情况下的平均成功比率均超过了 99%,表明侦察蜂阶段的新食物源几乎全部由 OED 所生成,这也在一定程度上可说明 OED 方法要显著地优于随机初始化方法.

Table 16 Average successful times of running OED and random initialization approaches, and the average successful ratio of OED

表 16 OED 和随机初始化方法的平均成功执行次数以及 OED 的平均成功比率

Function	D=30			D=50		
	ABC-rand	ABC-OED	SR (%)	ABC-rand	ABC-OED	SR (%)
F01	0	317	100	0	965	100
F02	0	565	100	0	1 210	100
F03	0	85.9	100	0	149	100
F04	0	39.9	100	0	147	100
F05	2	364	99.45	3	668	99.55
F06	0	145	100	0	509	100
F07	0	86.6	100	0	175	100
F08	0	431	100	0	905	100
F09	0	8.57	100	0	139	100
F10	0	42.1	100	0	376	100

Table 16 Average successful times of running OED and random initialization approaches, and the average successful ratio of OED (continued)

表 16 OED 和随机初始化方法的平均成功执行次数以及 OED 的平均成功比率(续)

Function	*D*=30			*D*=50		
	ABC-rand	ABC-OED	SR (%)	ABC-rand	ABC-OED	SR (%)
F11	0	452	100	0	1030	100
F12	0	29.2	100	0	201	100
F13	6	155	96.27	7	185	96.35
F14	0	1.03	100	0	77.5	100
F15	0	47.8	100	0	65	100
F16	2	123	98.40	1	158	99.37
Average	0.63	180.82	99.63	0.69	434.97	99.70

综上所述,OED 方法的优越性主要体现在:

1) 能够以较低的计算开销对被放弃食物源和全局最优解形成的超立方体空间进行均匀局部搜索,使得新的食物源能在不同维度上保存两者的有益信息,达到保存侦察蜂的搜索经验的目的;
2) 与随机初始化方法相比,OED 方法对 ABC 算法的运行时间影响微小,在可接受的范围内;
3) 可作为一个通用框架用于提高其他改进 ABC 算法的性能.

5 结束语

在 ABC 算法中,侦察蜂采用随机初始化的方法来生成新的食物源,易导致搜索经验丢失.为此,本文提出了一种采用正交实验设计的方法用于侦察蜂生成新的食物源.该方法可使得侦察蜂能够系统地探测被放弃食物源和全局最优解之间的超长方体搜索空间,在不同的维度上保存被放弃食物源和全局最优解的有益信息,达到快速有效地找到新食物源的目的,从而提高 ABC 的搜索性能.在 16 个典型的测试函数上进行了一系列的实验验证,实验结果表明:本文方法能在基本不增加算法运行时间的条件下,显著地提高 ABC 的求解精度和收敛速度.与其他 3 种变异方法的比较结果也表明,本文方法的优势更加明显.同时,本文方法能够作为一个通用的框架,用于提高其他改进 ABC 算法的搜索性能.

References:

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

[2] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization, 2007,39(3):459−471. [doi: 10.1007/s10898-007-9149-x]

[3] Kennedy J, Eberhart R. Particle swarm optimization. In: Proc. of the IEEE Int'l Conf. on Neural Networks. Piscataway: IEEE, 1995. 1942−1948. [doi: 10.1109/ICNN.1995.488968]

[4] Hu W, Li ZS. A simpler and more effective particle swarm optimization algorithm. Ruan Jian Xue Bao/Journal of Software, 2007, 18(4):861−868 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/861.htm [doi: 10.1360/jos180861]

[5] Gao WF, Liu SY, Huang LL. Inspired artificial bee colony algorithm for global optimization problems. Acta Electronic Sinica, 2012,40(12):2396−2403 (in Chinese with English abstract). http://www.ejournal.org.cn/CN/abstract/abstract7004.shtml [doi: 10.3969/j.issn.0372-2112.2012.12.007]

[6] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation, 2009,214(1): 108−132. [doi: 10.1016/j.amc.2009.03.090]

[7] Jia ZS, Si XC, Wang T. Optimum method for sea clutter parameter based on artificial bee colony. Journal of Central South University (Science and Technology), 2012,43(9):3485−3489 (in Chinese with English abstract). http://new.zndxzk.com.cn/paper/paperView.aspx?id=paper_30216

[8] Karaboga D, Ozturk C, Karaboga N, Gorkemli B. Artificial bee colony programming for symbolic regression. Information Sciences, 2012,209(11):1−15. [doi: 10.1016/j.ins.2012.05.002]

[9] Garro BA, Sossa H, Vázquez RA. Artificial neural network synthesis by means of artificial bee colony (ABC) algorithm. In: Proc. of the IEEE Congress on Evolutionary Computation. New Orleans: IEEE, 2011. 331−338. [doi: 10.1109/CEC.2011.5949637]

[10] Yeh W, Hsieh T. Artificial bee colony algorithm-neural networks for S-system models of biochemical networks approximation. Neural Computing and Applications, 2012,21(2):365−375. [doi: 10.1007/s00521-010-0435-z]

[11] Szeto WY, Wu Y, Ho SC. An artificial bee colony algorithm for the capacitated vehicle routing problem. European Journal of Operational Research, 2011,215(1):126−135. [doi: 10.1016/j.ejor.2011.06.006]

[12] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing, 2008,8(1):687−697. [doi: 10.1016/j.asoc.2007.05.007]

[13] Zhu G, Kwong S. Gbest-Guided artificial bee colony algorithm for numerical function optimization. Applied Mathematics and Computation, 2010,217(7):3166−3173. [doi: 10.1016/j.amc.2010.08.049]

[14] Gao W, Liu S, Huang L. A novel artificial bee colony algorithm with Powell's method. Applied Soft Computing, 2013,13(9): 3763−3775. [doi: 10.1016/j.asoc.2013.05.012]

[15] Karaboga D, Gorkemli B, Ozturk C, Karaboga N. A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review, 2014,42(1):21−57. [doi: 10.1007/s10462-012-9328-0]

[16] Akay B, Karaboga D. A modified artificial bee colony algorithm for real-parameter optimization. Information Sciences, 2012, 192(6):120−142. [doi: 10.1016/j.ins.2010.07.015]

[17] Banharnsakun A, Achalakul T, Sirinaovakul B. The best-so-far selection in artificial bee colony algorithm. Applied Soft Computing, 2011,11(2):2888−2901 [doi: 10.1016/j.asoc.2010.11.025]

[18] Das S, Biswas S, Kundu S. Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization. Applied Soft Computing, 2013,13(12):4676−4694. [doi: 10.1016/j.asoc.2013.07.009]

[19] Gao W, Liu S. Improved artificial bee colony algorithm for global optimization. Information Processing Letters, 2011,111(17): 871−882. [doi: 10.1016/j.ipl.2011.06.002]

[20] Gao W, Liu S, Huang L. A global best artificial bee colony algorithm for global optimization. Journal of Computational and Applied Mathematics, 2012,236(11):2741−2753. [doi: 10.1016/j.cam.2012.01.013]

[21] Gao W, Liu S. A modified artificial bee colony algorithm. Computers & Operations Research, 2012,39(3):687−697. [doi: 10.1016/j.cor.2011.06.007]

[22] Kang F, Li J, Xu Q. Structural inverse analysis by hybrid simplex artificial bee colony algorithms. Computers & Structures, 2009, 87(13):861−870. [doi: 10.1016/j.compstruc.2009.03.001]

[23] Kang F, Li J, Ma Z. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. Information Sciences, 2011,181(16):3508−3531. [doi: 10.1016/j.ins.2011.04.024]

[24] Bansal JC, Sharma H, Arya KV, Nagar A. Memetic search in artificial bee colony algorithm. Soft Computing, 2013,17(10): 1911−1928. [doi: 10.1007/s00500-013-1032-8]

[25] El-Abd M. Generalized opposition-based artificial bee colony algorithm. In: Proc. of the IEEE Congress on Evolutionary Computation. Brisbane: IEEE, 2012. 1−4. [doi: 10.1109/CEC.2012.6252939]

[26] Gao W, Liu S, Huang L. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. IEEE Trans. on Cybernetics, 2013,43(3):1011−1024. [doi: 10.1109/TSMCB.2012.2222373]

[27] Zhao XM. Design of Experiments. Beijing: Science Press, 2006 (in Chinese).

[28] Zhan Z, Zhang J, Li Y, Shi Y. Orthogonal learning particle swarm optimization. IEEE Trans. on Evolutionary Computation, 2011, 15(6):832−847. [doi: 10.1109/TEVC.2010.2052054]

[29] Wang Y. Solving complex continuous optimization problems based on evolutionary algorithms [Ph.D. Thesis]. Changsha: Central South University, 2011 (in Chinese with English abstract). [doi: 10.7666/d.y1918388]

[30] Leung Y, Wang Y. An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Trans. on Evolutionary Computation, 2001,5(1):41−53. [doi: 10.1109/4235.910464]

[31] Cai ZX, Jiang ZY, Wang Y, Luo YD. A novel constrained optimization evolutionary algorithm based on orthogonal experimental design. Chinese Journal of Computers, 2010,33(5):855−864 (in Chinese with English abstract). http://cjc.ict.ac.cn/qwjs/view.asp?id=3091 [doi: 10.3724/SP.J.1016.2010.00855]

[32] Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Trans. on Evolutionary Computation, 1999,3(2):82−102. [doi: 10.110.9/4235.771163]

[33] Wang Y, Cai Z, Zhang Q. Enhancing the search ability of differential evolution through orthogonal crossover. Information Sciences, 2012,185(1):153−177. [doi: 10.1016/j.ins.2011.09.001]

[34] Gong WY, Cai ZH. Research on an ε-domination based orthogonal differential evolution algorithm for multi-objective optimization. Journal of Computer Research and Development, 2009,46(4):655−666 (in Chinese with English abstract). http://crad.ict.ac.cn/CN/Y2009/V46/I4/655

[35] Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report, Singapore: Nanyang Technological University, 2005.

[36] Zhou X, Wu Z, Wang H, Rahnamayan S. Enhancing differential evolution with role assignment scheme. Soft Computing, 2014, 18(11):2209−2225. [doi: 10.1007/s00500-013-1195-3]

[37] Xiong G, Shi D, Duan X. Enhancing the performance of biogeography-based optimization using polyphyletic migration operator and orthogonal learning. Computers & Operations Research, 2014,41:125−139. [doi: 10.1016/j.cor.2013.07.021]

[38] Alatas B. Chaotic bee colony algorithms for global numerical optimization. Expert Systems with Applications, 2010,37(8): 5682−5687. [doi: 10.1016/j.eswa.2010.02.042]

[39] Ren Y, Wu Y. An efficient algorithm for high-dimensional function optimization. Soft Computing, 2013,17(6):995−1004. [doi: 10.1007/s00500-013-0984-z]

[40] Sharma TK, Pant M. Enhancing the food locations in an artificial bee colony algorithm. Soft Computing, 2013,17(10):1939−1965. [doi: 10.1007/s00500-013-1029-3]

[41] Xiang W, An M. An efficient and robust artificial bee colony algorithm for numerical optimization. Computers & Operations Research, 2013,40(5):1256−1265. [doi: 10.1016/j.cor.2012.12.006]

[42] Luo J, Wang Q, Xiao X. A modified artificial bee colony algorithm based on converge-onlookers approach for global optimization. Applied Mathematics and Computation, 2013,219(20):10253−10262. [doi: 10.1016/j.amc.2013.04.001]

[43] Garc IAS, Molina D, Lozano M, Herrera F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC 2005 special session on real parameter optimization. Journal of Heuristics, 2009,15(6): 617−644. [doi: 10.1007/s10732-008-9080-4]

[44] Garc IAS, Fern A, Ndez A, Luengo J, Herrera F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences, 2010,180(10): 2044−2064. [doi: 10.1016/j.ins.2009.12.010]

[45] Gong MG, Jiao LC, Yang DD, Ma WP. Research on evolutionary multi-objective optimization algorithms. Ruan Jian Xue Bao/Journal of Software, 2009,20(2):271−289 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3483.htm [doi: 10.3724/SP.J.1001.2009.03483]

[46] Epitropakis MG, Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN. Enhancing differential evolution utilizing proximity-based mutation operators. IEEE Trans. on Evolutionary Computation, 2011,15(1):99−119. [doi: 10.1109/TEVC.2010.2083670]

[47] Li G, Niu P, Xiao X. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. Applied Soft Computing, 2012,12(1):320−332. [doi: 10.1016/j.asoc.2011.08.040]

[48] Higashi N, Iba H. Particle swarm optimization with Gaussian mutation. In: Proc. of the IEEE Swarm Intelligence Symp. Indianapolis: IEEE, 2003. 72−79. [doi: 10.1109/SIS.2003.1202250]

[49] Wang H, Wang W, Wu Z. Particle swarm optimization with adaptive mutation for multimodal optimization. Applied Mathematics and Computation, 2013,221(9):296−305. [doi: 10.1016/j.amc.2013.06.074]

[50] Stacey A, Jancic M, Grundy I. Particle swarm optimization with mutation. In: Proc. of the IEEE Congress on Evolutionary Computation. Canberra: IEEE, 2003. 1425−1430. [doi: 10.1109/CEC.2003.1299838]

[51] Wang H, Liu Y, Zeng S, Li H, Li C. Opposition-Based particle swarm algorithm with Cauchy mutation. In: Proc. of the IEEE Congress on Evolutionary Computation. Singapore: IEEE, 2007. 4750−4756. [doi: 10.1109/CEC.2007.4425095]

[52] Wang H, Wu Z, Rahnamayan S, Liu Y, Ventresca M. Enhancing particle swarm optimization using generalized opposition-based learning. Information Sciences, 2011,181(20):4699−4714. [doi: 10.1016/j.ins.2011.03.016]

[53] Zhou XY, Wu ZJ, Wang H, Zhang HY. Elite opposition-based particle swarm optimization. Acta Electronic Sinica, 2013,41(8): 1647−1652 (in Chinese with English abstract). http://www.ejournal.org.cn/CN/abstract/abstract7622.shtml [doi: 10.3969/J.ISSN. 0372-2112.2013.08.031]

附中文参考文献:

[4] 胡旺,李志蜀.一种更简化而高效的粒子群优化算法.软件学报,2007,18(4):861−868. http://www.jos.org.cn/1000-9825/18/861.htm [doi: 10.1360/jos180861]

[5] 高卫峰,刘三阳,黄玲玲.受启发的人工蜂群算法在全局优化问题中的应用.电子学报,2012,40(12):2396−2403. http://www.ejournal.org.cn/CN/abstract/abstract7004.shtml [doi: 10.3969/j.issn.0372-2112.2012.12.007]

[7] 贾宗圣,司锡才,王桐.基于人工蜂群技术的海杂波参数优化方法.中南大学学报(自然科学版),2012,43(9):3485−3489. http://new.zndxzk.com.cn/paper/paperView.aspx?id=paper_30216

[27] 赵选民.实验设计方法.北京:科学出版社,2006.

[29] 王勇.基于进化算法求解复杂连续优化问题的研究[博士学位论文].长沙:中南大学,2011.[doi: 10.7666/d.y1918388]

[31] 蔡自兴,江中央,王勇,罗一丹.一种新的基于正交实验设计的约束优化进化算法.计算机学报,2010(5):855−864. http://cjc.ict.ac.cn/qwjs/view.asp?id=3091 [doi: 10.3724/SP.J.1016.2010.00855]

[34] 龚文引,蔡之华.基于ε占优的正交多目标差分演化算法研究.计算机研究与发展,2009,46(4):655−666. http://crad.ict.ac.cn/CN/Y2009/V46/I4/655

[45] 公茂果,焦李成,杨咚咚,马文萍.进化多目标优化算法研究.软件学报,2009(2):271−289. http://www.jos.org.cn/1000-9825/3483.htm [doi: 10.3724/SP.J.1001.2009.03483]

[53] 周新宇,吴志健,王晖,李康顺,张浩宇.一种精英反向学习的粒子群优化算法.电子学报,2013,41(8):1647−1652. http://www.ejournal.org.cn/CN/abstract/abstract7622.shtml [doi: 10.3969/J.ISSN.0372-2112.2013.08.031]

周新宇(1987－),男,江西九江人,博士,讲师,CCF 会员,主要研究领域为智能计算,并行计算.

吴志健(1963－),男,博士,教授,博士生导师,主要研究领域为智能计算,并行计算,智能信息处理.

王明文(1964－),男,博士,教授,博士生导师,CCF 会员,主要研究领域为智能信息处理,信息检索.