文章编号: 1001-0920(2011)08-1233-06

基于高斯分布估计的细菌觅食优化算法

刘小龙, 李荣钧, 杨 萍

(华南理工大学 工商管理学院，广州 510640)

摘 要: 针对细菌觅食算法在优化过程中存在步长一致、速度较慢的缺陷, 赋予细菌以灵敏度的概念来调节趋化步长; 将分布估计算法的思想引入繁殖算子, 对细菌能量较好的半数细菌进行分布估计再生以增加群体的多样性, 提高收敛速度; 根据细菌的能量情况, 赋予细菌自适应迁移概率, 对较差的细菌进行随机或指定迁移, 以提高算法的全局寻优能力. 采用多峰高维标准测试函数对改进算法进行了测试, 结果表明, 所提出算法有效地提高了搜索速度和精度, 改造后可用于多维、约束等实际工程问题的优化.

关键词: 分布估计算法；细菌觅食；全局优化算法

中图分类号: TP18 **文献标识码:** A

Bacterial foraging optimization algorithm based on estimation of distribution

LIU Xiao-long, *LI Rong-jun*, *YANG Ping*

(School of Business Administration，South China University of Technology，Guangzhou 510640，China. Correspondent: LIU Xiao-long，E-mail：810963@qq.com)

Abstract: In view of the defects of the same swim step and slow velocity in the bacterial foraging algorithm, the conception of sensitivity is given to bacteria in order to regulate the swim step. The thoughts of estimation of distribution algorithms are introduced to the reproduction, and the half of bacteria population with the best values split based on the estimation of distribution. This approach increases the population diversity and improves the convergence speed. According to the energy of the bacteria, the probability of elimination-dispersion is computed, poor bacteria are randomly or assignably dispersed, which improves the global searching ability. The algorithm is tested by the high-dimensional and multimodal function. The results show that, the algorithm can effectively improve the searching speed and the accuracy, and can be applied to multidimensional and constrained practical engineering problems.

Key words: estimation of distribution；bacterial foraging；global optimization

1 引 言

自 20 世纪 40 年代以来,针对许多工程实际问题的高维度、多峰值、非线性且不可微分的特征, 人们一直在利用源于生物系统的灵感来解决实际问题, 进而构造并设计出了许多仿生优化算法. Holland[1]于 1975 年提出了遗传算法 (GA); Dorigo 等人[2]提出了蚁群优化算法 (ACO); Eberhart 和 Kennedy[3]提出了粒子群优化算法 (PSO); 李晓磊[4]提出了鱼群算法 (AFSA). 另外, Miiller 等人[5]提出了细菌趋药性算法 (BC); Passino 等人[6]提出了细菌觅食算法 (BFO).

上述 GA, ACO, PSO 和 AFSA 等算法, 都是基于高等生物作为启发对象, 形成的一种“生成 + 检验”为特征的自适应人工智能技术. 而 BC 和 BFO 算法, 则是从微生物的行为机制出发, 通过模拟细菌对环境感知的变化而形成的一种优化方法. BC 算法的基础是基于单个细菌的运动行为, 其算法本质是一种基于梯度原理的搜索技术, 算法性能与 GA 相当. BFO 算法通过细菌群体的竞争与协作来实现优化, 是一种基于群体的搜索技术. 相关研究表明, BFO 搜索性能优于 GA, 但没有 GA 成熟. 目前, BFO 算法因具有群体并行、易跳出局部极小等优点, 正在成为仿生算法研究领域的又一热点.

为了提高 BFO 算法的性能, 人们从算法参数和

收稿日期: 2010-05-24；**修回日期:** 2010-09-26.

基金项目: 国家自然科学基金项目(71071057).

作者简介: 刘小龙(1977−), 男, 讲师, 博士生, 从事智能计算及其在经济管理中的应用等研究；李荣钧(1946−), 男, 教授, 博士生导师, 从事智能计算等研究.

算法融合上进行了研究. Liu等人[7]改进了大肠杆菌间的相互作用机制, 并对BFO算法的收敛性进行了初步分析; Mishra等人[8]基于TS模糊规则, 系统地提出了改进的MBFO算法; Datta等人[9]根据自适应增量调制原理, 提出了具有自适应趋化步长的BFO算法; Majhi等人[10]设计了自动趋化步长调节的数学模型, 并将其应用于神经网络的训练; Chen等人[11]根据生物自适应搜索策略, 提出了自适应菌群觅食算法. 实验结果表明, 文献[9-11]从趋化步长的视角, 根据细菌在觅食生命周期内的获取能量, 赋予细菌自适应调节趋化步长的能力, 在不增加算法复杂性的前提下, 有效提高了算法寻优的速度.

Biswas等人[12]将BFO算法与粒子群算法相结合, 提出一种混合优化算法并应用于多峰函数优化; Dasgupta等人[13]将差分进化算法中的交叉与变异操作引入BFO算法, 提出了一种混合全局优化算法; Kim等人[14]在BFO算法中引入遗传算法的交叉、变异算子, 提出了GABFO算法; Luh等人[15]从细菌进化的角度, 提出了细菌进化算法(BEA). 在现行细菌觅食算法的相关研究中, 文献[12]在细菌的趋化算子内嵌入粒子群方法, 赋予细菌对全局极值的感知能力, 所有细菌通过与全局极值的对比, 根据粒子群迭代公式更新细菌位置, 此时算法中的趋化算子并没有起到作用. 文献[13]在细菌进化的过程中, 引入交叉和变异算子, 虽增加了种群的多样性, 但同时也容易导致群体中优秀个体的缺失.

针对上述问题, 本文通过赋予细菌以灵敏度, 对优秀个体进行分布估计再生, 进而对细菌群体进行自适应概率迁移, 实现了提高算法精度和速度的目的. 通过标准测试函数验证了所提出算法的有效性.

2 基于分布估计的细菌觅食优化设计

2.1 分布估计算法

分布估计算法(EDAs)[16]是一种基于变量概率分布的随机搜索算法. 与其他进化算法不同的是, 分布估计不采用交叉变异等方式产生下一代, 而是通过对优秀个体的采样并进行空间的统计分布分析, 进而建立相应的概率分布模型, 并以此概率模型产生下一代个体. 如此反复迭代, 实现群体的进化. 在EDA领域的相关研究中, 大都是针对二进制变量的; 实数编码的变量由于连续空间概率模型的复杂性, 给设计有效的分布估计算法增加了难度, 因此连续EDA的发展相对缓慢.

目前, 基于高斯分布的单变量实数编码分布估计算法(UMDAc)[17]和基于高斯分布的变量无关型分布估计算法(PBILc)[18]是比较有代表性的实数变量无关分布估计算法, 它们都采用高斯分布作为描述连续解空间的概率模型. 二者的不同在于采用了不同的构造方法更新高斯分布模型. 此外, 基于变量正态分布的随机爬山法(SHCLVND)[19]也是研究较早的基于高斯分布的随机优化算法. 除了高斯分布外, 直方图分布是日本学者Tsutsui等人[20]提出的另外一种描述连续解空间概率模型的有效方法. Rudlof等人[21]应用高斯分布模型来产生新的个体.

分布估计算法借助于样本分布的概率模型, 能很好地描述变量间的相互关系, 极大丰富了智能优化算法的研究内容, 为解决复杂的优化问题提供了新的方法, 对解决一些非线性、变量耦合的优化问题提供了更有效的工具.

2.2 细菌觅食优化算法

BFO算法求解优化问题的一般过程为: 产生初始解群体, 计算评价函数的值, 利用群体的相互影响和作用机制进行迭代优化. 该影响机制包括趋化、繁殖和迁移3个主要算子, 评价函数的适应值(优化问题的解)对应于搜索空间中的细菌状态(能量情况).

在BFO算法中, 趋化算子是模拟细菌实现最优觅食的一种行为, 这种行为表现在细菌通过翻转和游动, 向营养丰富的地方进行集中, 从而为不断寻找出路所体现的一种有目的的运动. 在细菌的趋化过程中, 细菌的运动行为包括翻转和游动两种. 该算法首先定义细菌向任意方向移动的单位步长为翻转; 如果细菌完成一次翻转后的适应值得到改善, 则沿同一方向继续移动若干步, 直至适应值不再改善, 或达到预定的移动步数为止, 这一过程定义为游动.

细菌觅食优化的趋化算子主要是细菌在解空间对可行解的邻域进行搜索, 并通过适应度进行判定和对行进方向进行调整. 经过一定步数的趋化搜索后, 为了提高搜索的精度和缩短搜索时间, 以趋化中各细菌的适应值累加(能量)和为标准, 能量高的半数细菌分裂出子细菌, 子细菌继承父代细菌所有的生物特性, 具有与父代细菌相同的位置及能量.

为了提高算法的全局寻优能力, BFO算法通过设定一个固定的迁移概率, 对算法迭代中的每个细菌进行随机性迁移. 若满足条件, 则随机生成一个新个体并替换原个体, 等同于将原细菌个体迁移到了一个新位置.

2.3 基于分布估计的细菌优化改进算法流程

由BFO算法的基本原理可以看出, 趋化算子确保了细菌的局部搜索能力, 繁殖算子能加快细菌的搜索速度和精度. 但由于细菌移动步长过小等因素, 高精度同样带来了早熟、不熟的问题, 使算法可能陷入

局部极小区域. 同时, 在趋化算子的游动和翻转中, 能量不同的细菌能够翻转同样的步长, 因而无法体现出能量高细菌和能量低细菌的步长特性, 在一定程度上降低了细菌趋化算子的寻优精度. 另外, 觅食算法中的迁移操作针对每个细菌给定相同的迁移概率 P_{ed}, 如果随机数小于 P_{ed}, 则对该细菌进行迁移. 这对于那些获得较好位置和能量 (比如全局极值附近) 的细菌而言, 相当于丢失了精英个体, 迁移相当于解的退化.

针对上述问题, 本文在细菌觅食的繁殖算子中, 嵌入分布估计算法, 通过对优秀群体的统计分析, 建立概率分布模型, 直接描述优秀群体的进化趋势, 这是对微生物进化进行"宏观"层面上的数学建模. 改进后的细菌觅食优化算法流程如图 1 所示.

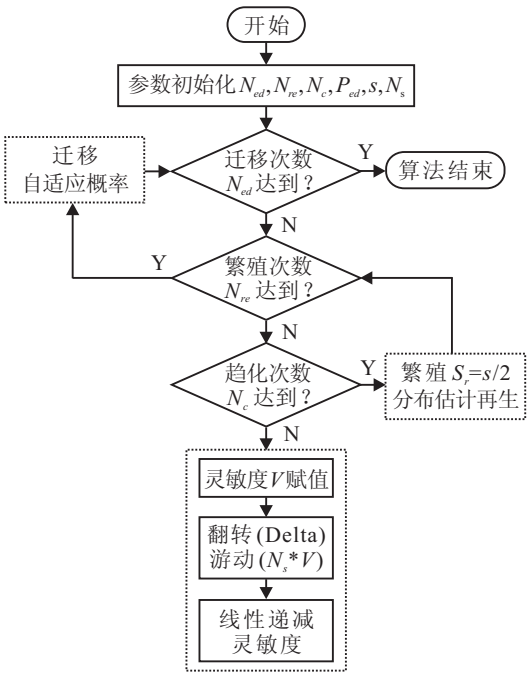

图 1　改进细菌觅食优化流程

另外, 为了提高细菌趋化行为的效率和增加种群多样性, 但又不扼杀种群中的优秀个体, 本文算法赋予细菌以灵敏度的概念, 从而加快了算法的收敛速度. 最后, 在迁移算子中, 所有细菌按照能量获取的大小进行自适应概率迁移. 新算法的主要步骤如下:

Step 1　参数初始化. S 为细菌规模数, N_{ed} 为迁移次数, N_{re} 为繁殖次数, N_c 为趋化次数, N_s 为游动次数, P_{ed} 为基本迁移概率.

Step 2　细菌位置初始化. 设细菌个体代表解空间的一个解, $P(i,j,k,l)$ 表示细菌 i 的空间位置向量. 其中: j 为第 j 代趋化循环, k 为第 k 代繁殖循环, l 为第 l 代迁移循环. 为每个细菌随机产生任意方向的单位步长向量, 初始化位置采用 $X = X_{\min} + \mathrm{rand}(X_{\max} - X_{\min})$ 产生, rand 为均匀分布在 [0 ∼ 1] 区间的随机数, 并计算细菌的初始适应度 J 值.

Step 3　迁移循环 $l = 1 : N_{ed}$; 繁殖循环 $K = 1 : N_{re}$; 趋化循环 $j = 1 : N_c$.

Step 4　细菌趋化算子, 在一个趋化步骤内, 细菌具有灵敏度记忆功能, 每个细菌按照如下步骤进行:

1) 灵敏度赋值

$$V = \frac{J_i}{J_{\max}}(X_{\max} - X_{\min}) * \mathrm{rand}. \tag{1}$$

其中: V 为灵敏度; $X_{\max}, X_{\min}$ 为变量的边界; J 为适应值.

2) 翻转. 产生随机向量 $\Delta(i)$, 进行方向调整, 按下式更新细菌位置和适应值:

$$P(i, j+1, k, l) = P(i, j, k, l) + C(i)\phi(i), \tag{2}$$

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta^{\mathrm{T}}(i)\Delta(i)}}. \tag{3}$$

其中: $C(i)$ 为按选定的方向游动的单位步长, $\Delta(i)$ 为变向中生成的随机向量, $\phi(i)$ 为进行方向调整后选定的方向.

3) 游动. 如果翻转的适应值得到改善, 则按照翻转的方向进行游动, 直至适应值不再改善. 游动步长采用 CV 来表示, C 为单位步长, V 为灵敏度.

Step 5　按下式线性递减细菌灵敏度:

$$V = \frac{\mathrm{step}_{\max} - \mathrm{step}_i}{\mathrm{step}_{\max}} V. \tag{4}$$

Step 6　分布估计繁殖. 经过一个完整的趋化循环后, 对每个细菌按照能量 (适应度的累加和) 进行排序, 能量较差的半数细菌死亡, 对能量较好的半数细菌进行分布估计再生. 假设待优化变量的每一维度相互独立, 且各维度之间服从高斯分布, 繁殖方法如下:

$$X_{\mu,\sigma} = r_{\mathrm{norm}}\sigma + \mu, \tag{5}$$

$$r_{\mathrm{norm}} = \sqrt{-2\ln r_1}\sin(2\pi r_2). \tag{6}$$

其中: r_1 和 r_2 为区间 [0, 1] 均匀分布的随机数; r_{norm} 产生的公式称为 BOX-Muller 公式; μ, σ 为细菌较优位置的分维度均值和标准差向量; 乘积采用点乘.

Step 7　细菌迁移. 所有细菌按照下式进行自适应概率迁移:

$$S_{el}(B_{ac_i}) = \frac{J_{\max}(B_{ac}) - J_i(B_{ac})}{J_{\max}(B_{ac}) - J_{\min}(B_{ac})} P_{ed}. \tag{7}$$

其中: J 为能量函数, P_{ed} 为基本迁移概率. 为了提高算法后期的细菌群体多样性, 本文按照细菌群体在生命周期内已获得的能量大小进行概率迁移, 能量大 (适应值小) 的迁移概率小, 能量小 (适应值大) 的迁移概率大, 迁移概率按照遗传算法中的轮盘赌方法[1]作为选择机制. 因采用了轮盘赌方法进行选择, J 最小的细菌肯定被迁移, 故在式 (7) 中乘以了基本迁移概率.

Step 8　循环结束条件判断. 若满足, 则结束, 输出结果.

3 实验与分析

3.1 检验函数

实验函数采用Rosenbrock (f_1), Rastrigin (f_2), Griewank (f_3) 和 Ackley (f_4) 这4个标准测试函数进行分析, 详细描述见表1. 这几个典型的标准函数都是NP难问题, 都具有全局极小点, 用他们来比较各种算法的性能是合适的. 其中: f_1 和 f_3 是单峰函数, 解空间有非常多狭窄的局部极小通道; f_2 和 f_4 是多峰函数, 算法很容易陷入通向全局最优点的局部最优上.

表 1 标准测试函数及其搜索范围

函数	数学表达式	变量边界
$f_1(x)$	$\sum_{i=1}^{n}(100(x_{i+1}-x_i)^2+(x_i-1)^2)$	$(-5,10)$
$f_2(x)$	$\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	$(-5.12,5.12)$
$f_3(x)$	$\frac{1}{4\,000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x-i}{\sqrt{i}}\right)+1$	$(-600,600)$
$f_4(x)$	$20+e-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)$	$(-15,30)$

以上4个测试函数, 即使采用传统的优化方法也很难实现优化, 且优化的难度会随着测试函数维数的增加而急剧上升. 对于低维单峰函数, 细菌觅食优化算法可以得到满意的优化结果. 对于高维且多峰函数问题, 细菌觅食算法的优化效果难以令人满意. 基于此, 4个测试函数的维度设置为30维和50维, 趋化内的迭代总次数 (用于算法性能追踪) 为 MaxIter $= N_{ed}*N_{re}*N_c*N_s$. 然而实际细菌位置较好的 N_s 并没有达到4, 经过测试整个迭代约为800次. 当然, 为了提高算法的精度, 还可以设置更高的迭代次数.

为了测试新算法的性能, 本文同时对细菌觅食算法BFO, 粒子群算法PSO, 遗传算法GA以及本文提出的基于高斯分布估计的细菌觅食算法EDA-BFO, 在上述4个高维多峰函数中进行了测试. 测试指标包括优化均值、标准差、最优值和成功率. 其中: 优化均值是指所有优化搜索结果中的最优平均值; 标准差是算法搜索50次最优值的标准差; 最优值是50次测试的最好结果; 成功率是对照阈值统计最优值落入阈值内的比率.

3.2 对比实验参数设置

对于细菌觅食算法, 主要参数设为: 细菌数 (粒子数) $s=50$, 迁移次数 $N_{ed}=2$, 繁殖次数 $N_{re}=4$, 趋化次数 $N_c=40$, 游动次数 $N_s=3$, 迁移概率 $P_{ed}=0.25$, 游动步长 $C=0.001R$, R 为优化区间的宽度. PSO算法中, $\omega=0.9$, C_1 从5线性递减至0.1, C_2 从0.5线性递减至0.1. GA算法采用谢菲尔德GATBX工具箱算法, 其中: 选择率为0.9, 交叉率为0.7, 变异率为0.08.

3.3 实验结果与讨论

4种算法在以上4个标准测试函数上的测试次数均为50次. 由于本文是比较现行算法在给定时间复杂度情况下的性能, 在迭代步数一致的情况下, 阈值设定根据维度而增加. 因本文的基本参照对象是粒子群算法, 故阈值设置以粒子群算法的平均值为参考值. 为了对比维度的下降对算法性能的影响, 本文还对4种算法在30维度的性能表现进行了测试, 因基本结果与50维度一致, 故此处省略了30维度的表格测试比较. 算法的性能测试结果如表2所示.

表 2 基于50维的算法性能测试结果

函数 (维度) ⟨阈值⟩	指标	比较的相关算法			
		GA	PSO	BFO	EDA-BFO
$f_1(50)$ ⟨200⟩	平均值	50.418	289.02	50.45	30.06
	标准差	34.58	79.63	0.23	23.11
	最优值	0.822	139.46	49.81	7.25e-04
	成功率/%	100	14	100	100
$f_2(50)$ ⟨120⟩	平均值	66.977	120.615	3.73	0.002 7
	标准差	11.75	20.79	0.36	0.003 5
	最优值	37.81	73.29	2.47	1.709e-07
	成功率/%	100	56	100	100
$f_3(50)$ ⟨8⟩	平均值	1.505	7.466	1.028	0.033 7
	标准差	1.15	2.67	0.008	0.044 9
	最优值	0.974	3.542	1.006	2.302e-07
	成功率/%	100	68	100	100
$f_4(50)$ ⟨4⟩	平均值	4.262	3.581	0.397	0.012 5
	标准差	0.877	0.426	0.034	0.010 6
	最优值	1.087	2.735	0.318	8.279e-05
	成功率/%	26	76	100	100

由表2可知, 在4种算法的整体性能中, GA性能最差; 改进后的EDA-BFO算法在 $f_1\sim f_4$ 中的最优值精度都高于其他算法, 除了 f_1 的测试结果, 新算法的标准差相对稳定, 在平均值、最优值和成功率上的总体性能均优于其他4种算法.

基于 f_1 的时间测试结果如表3所示. 从时间复杂度上看, 本文算法的耗时与BFO相当.

表 3 基于 f_1 的算法时间测试

f_1 维度	GA	PSO	BFO	EDA-BFO
30维	21.563	1.203	2.484	2.453
50维	32.938	1.360	2.719	2.703

另外, 算法的收敛性是检测算法性能的重要指标. 基于 f_1 的收敛曲线如图2所示.

由图2可知, 4种算法中, EDA-BFO算法由于引入了高斯分布的繁殖算子, 使得收敛速度加快. 即使是在算法的后期, 因细菌之间保持了种群的多样性, 使平均值与最优值之间保持了适当的距离, 故具有较好的寻优能力.

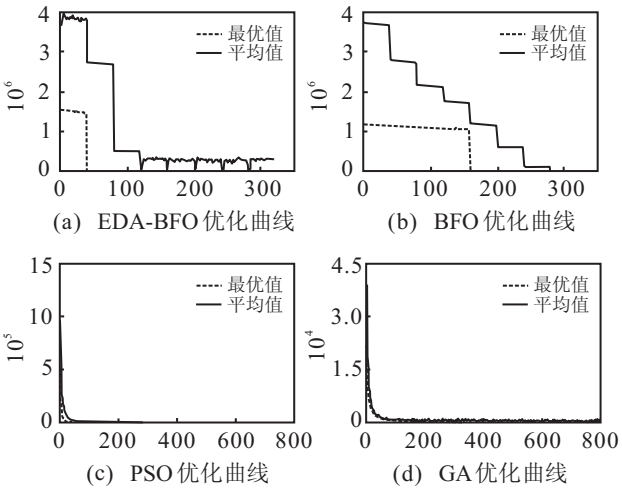

图 2　基于 50 维 f_1 的算法收敛性能测试

3.4　重要参数的敏感性分析

鉴于本文算法在 f_4 中的寻优标准差最小, 多次测试的最优值和平均最优值较为接近, 本文在现有参数赋值的基础上, 通过改变赋值来观察参数在 50 维度 f_4 中寻优精度 (20 次测试的最优平均值) 的改变情况, 从而获得该算法在参数设置上的一般规律. 文中涉及的主要参数有: $s, N_{ed}, N_{re}, N_c, N_s, P_{ed}, C$. 现有参数在 20 次函数测试中的最优平均值为 0.002 4, 基于 f_4 的单个参数改变对算法寻优精度的影响见表 4 和表 5.

表 4　基于 f_4 的参数变化敏感性分析

参数改变量/%	s	N_c	P_{ed}	C
+120	0.002 0	0.004 4	0.005 4	0.011 5
+100	0.002 5	0.005 6	0.004 1	0.008 2
+80	0.003 1	0.004 7	0.004 1	0.008 9
+60	0.003 5	0.003 6	0.006 5	0.005 2
+40	0.002 9	0.003 6	0.003 2	0.004 3
+20	0.002 8	0.003 4	0.006 1	0.004 2
−20	0.006 3	0.003 3	0.008 2	0.002 7
−40	0.006 2	0.003 5	0.006 3	0.001 9

表 5　基于 f_4 的参数赋值敏感性分析 (量纲: e-02)

参数赋值	1	2	3	4	5	6	7
N_{ed}	141	0.24	0.29	0.40	0.39	0.36	0.46
N_s	0.38	0.37	0.24	0.44	0.38	0.35	0.20
N_{re}	8.61	0.61	0.38	0.24	0.45	0.43	0.45

由表 4 可知, 大量的细菌数 (s) 可以增加寻优精度, 但时间也是成倍放大. 现有测试结果表明: 等同于维度的细菌数可以基本实现测试函数所需要达到的精度; 搜索精度随着趋化数 N_c 的增加而呈现一种V型趋势, 说明适中的 N_c 会导致搜索精度的提高; P_{ed} 对搜索精度的影响不太明显, 没有呈现特定的规律; 游动步长 C 的增加明显降低了搜索精度, 但较小的移动步长所花费的搜索时间将大量增加.

由表 5 可知, N_{ed} 的增加并没有带来精度的提高, 却导致搜索速度变慢; N_s 的改变对搜索精度的影响不太大, 但较小的 N_s 有利于提高搜索速度; N_{re} 表示分布估计算法的导入次数, 太大的 N_{re} 对提高搜索精度没有作用.

另外, $s, N_c, N_{ed}, N_{re}, N_s$ 的改变, 对时间具有一种放大效应; P_{ed} 和 C 的改变对时间的响应较为平缓. 综合搜索精度和搜索时间, 本文选择的参数赋值是一种基于文献的相对折中, 其他比较算法的参数赋值则多基于现有文献的建议.

4　结　论

本文通过赋予细菌灵敏度的概念来改变细菌的游动步长, 同时将分布估计思想嵌入 BFO 算法的繁殖算子, 进而提出了一种改进的细菌觅食优化方法. 该方法尝试对待优化变量的搜索空间在开发能力和探索能力上取得均衡. 实验结果表明, 该方法性能优于简单 BFO 方法, 有效避免了 GA, PSO, PSO-BFO 等其他算法的部分缺陷, 编码后适用于解决高维复杂工程的相应优化问题. 后续研究的方向是对该方法的参数设置进行讨论, 以及探讨多变量相关情况下的收敛情况, 并将该方法应用于工程实际.

参考文献(References)

[1] Holland J. Adaptation in natural and artificial systems[M]. Ann Arbor: The University of Michigan Press, 1975.

[2] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[C]. Proc of ECAL'91. Paris: Elsevier Publishing, 1991: 134-142.

[3] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory[C]. Proc of the 6th Int Symposium on Micro Machine and Human Science. Nagoya: IEEE Robotics and Automation Society, 1995: 39-43.

[4] 李晓磊, 冯少辉, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法[J]. 系统工程理论与实践, 2002, 22(11): 32-38.
(Li X L, Feng S H, Qian J X. Artificial fish school algorithm based on animal self-contained optimization model[J]. Systems Engineering Theory and Practice, 2002, 22(11): 32-38.)

[5] Miiller S, Airaghi S, Marchelo J, et al. Optimization algorithms based on a model of bacterial chemotaxis[C]. Proc of the 6th Int Conf on Simulation of Adaptive Behavior. Paris: MIT Press, 2000: 375-384.

[6] Passino K M. Biomimicry of bacterial foraging for distributed optimization and control[J]. IEEE Control Systems Magazine, 2002, 22(3): 52-67.

[7] Liu Y, Passino K M, Polycarpou M. Stability analysis of m-dimensional asynchronous swarms with a fixed communication topology[J]. IEEE Trans on Automatic Control, 2003, 48(1): 76-95.

[8] Mishra S. A hybrid least square-fuzzy bacteria foraging strategy for harmonic estimation[J]. IEEE Trans on Evolutionary Computation, 2005, 9(1): 61-73.

[9] Datta T, Misra I S, Mangaraj B B, et al. Improved adaptive bacteria foraging algorithm in optimization of antenna array for faster convergence[J]. Progress in Electromagnetics Research C, 2008, 16(1): 143-157.

[10] Majhi R, Panda G, Sahoo G. Efficient prediction of stock market Indices using adaptive bacterial foraging optimization(ABFO) and BFO based techniques[J]. Expert Systems with Applications, 2009, 36(1): 10097-10104.

[11] Chen H, Zhu Y, Hu K. Self-adaptation in bacterial foraging optimization algorithm[C]. Proc of the 3rd Int Conf on Intelligent System and Knowledge Engineering. Xiamen, 2008: 1026-1031.

[12] Biswas A, Dasgupta S, Das S, et al. Synergy of PSO and bacterial foraging optimization: A comparative study on numerical benchmarks[C]. Proc of the 2nd Int Symposium on Hybrid Artificial Intelligent Systems. Salamanca, 2007: 255-263.

[13] Dasgupta A, Dasgupta S, Das S, et al. A synergy of differential evolution and bacterial foraging optimization for global optimization[J]. Neural Network Word, 2007, 17(6): 607-626.

[14] Kim D H, Abraham A, Cho J H. A hybrid genetic algorithm and bacterial foraging approach[J]. Information Sciences, 2007, 177(18): 3918-3937.

[15] Luh G C, Lee S W. A bacterial evolutionary algorithm for the job shop scheduling problem[J]. J of Chinese Institute of Industrial Engineers, 2006, 23(3): 185-191.

[16] Larra Naga P, Lozano J A. Estimation of distribution algorithms — A new tool for evolutionary computation[M]. Boston: Kluwer Academic Publishers, 2002.

[17] Larra Naga P, Etxeberria R, Lozano J A, et al. Optimization in continuous domains by learning and simulation of Gaussian networks[C]. Proc of the 2000 Genetic and Evolutionary Computation Conf Workshop Program. Las Vegas, 2000: 201-204.

[18] Sebag M, Ducoulombier A. Extending population-based in cremental learning to continuous search spaces[C]. Proc of the 5th Conf on Parallel Problem Solving from Nature — PPSN V. Amsterdam: Springer-Verlag, 1998: 418-427.

[19] Rudlof S, Koppen M. Stochastic hill climbing by vectors of normal distributions[C]. Proc of the 1st Online Workshop on Soft Computing. Nagoya, 1996: 60-70.

[20] Tsutsui S, Pelikan M, Goldberg D E. Evolutionary algorithm using marginal histogram models in continuous domain[R]. University of Illinois at Urbana-Champaign, Urbana, 2001.

[21] Rudlof S, Koppen M. Stochastic hill climbing with learning by vectors of normal distributions[M]. Berlin: Institute for Production Systems and Design Technology, 1996: 1-11.

(上接第1232页)

[3] Liu H, Yu L. Towards integrating feature selection algorithms for classification and clustering[J]. IEEE Trans on Knowledge and Data Engineering, 2005, 17(4): 491-502.

[4] Yu L, Liu H. Efficient feature selection via analysis of relevance and redundancy[J]. J of Machine Learning Research, 2004, 5(10): 1205-1224.

[5] Marko R S, Igor K. Theoretical and empirical analysis of Relief and ReliefF[J]. Machine Learning J, 2003, 53(1/2): 23-69.

[6] Crammer K, Gilad-Bachrach R, Navot A, et al. Margin analysis of the LVQ algorithm[C]. Proc of the 17th Conf on Neural Information Processing Systems. Banff, 2002: 462-469.

[7] Kononenko I. Estimating attributes: Analysis and extensions of relief[C]. Proc of the 6th European Conf on Machine Learning. Berlin, 1994: 171-182.

[8] Schollkopf B, Smola A, Williamson R C, et al. New support vector algorithms[J]. Neural Computation, 2000, 12(5): 1207-1245.

[9] 姜慧研, 崔晓亮, 周晓杰, 等. 基于双快速行进法的图像分割方法的研究[J]. 系统仿真学报, 2008, 20(3): 803-806.
(Jiang H Y, Cui X L, Zhou X J. et al. Image segmentation based on improved dual fast marching method[J]. J of System Simulation, 2008, 20(3): 803-806.)

[10] 蒋玉娇, 王晓丹, 王文军, 等. 一种基于PCA和ReliefF的特征选择方法[J]. 计算机工程与应用, 2010, 46(26): 170-172.
(Jiang Y J, Wang X D, Wang W J, et al. New feature selection approach by PCA and ReliefF[J]. Computer Engineering and Applications, 2010, 46(26): 170-172.)