广义菌群优化算法

陈建超[1] 胡桂武[1,2,3] 杜小勇[2,3]
(广东商学院数学与计算科学学院 广州 510320)[1]
(教育部数据工程与知识工程重点实验室 北京 100872)[2] (中国人民大学信息学院 北京 100872)[3]

摘 要 为提高菌群优化算法的性能,将群体聚集机制和自适应策略集成到趋药性操作中,取消聚集操作,构造出新的趋化操作,在趋化循环中引入自适应扩散机制,提高其克服"早熟"的能力,重新定义健康度,减少计算复杂性,得到了一种新的群体智能优化方法——广义菌群优化算法(GBFO,Generalized Bacterial Foraging Optimization)。通过10个复杂Benchmark函数的计算进行算法性能测试,并与几个典型的算法进行了实验比较,结果表明,GBFO算法在搜索能力和稳定性、求解质量和效率等方面优于其他典型算法的比率分别达到80%~90%,70%~80%,验证了该算法的优越性能。

关键词 菌群优化算法,聚集,趋化操作,扩散
中图法分类号 TP301.06 **文献标识码** A

Generalized Bacterial Foraging Optimization

CHEN Jian-chao[1] HU Gui-wu[1,2,3] DU Xiao-yong[2,3]
(School of Mathematics & Computational Science, Guangdong University of Business Studies, Guangzhou 510320, China)[1]
(Key Laboratory of Data Engineer and Knowledge Engineer for the Ministry of Education, Beijing 100872, China)[2]
(School of Information, Rennin University of China, Beijing 100872, China)[3]

Abstract In order to improve the performance of Bacterial Foraging Optimization (BFO), a new swarm intelligence optimization algorithm, called the generalized bacterial foraging optimization (GBFO) was proposed, which has new chemotactic operation, is only composed of chemotaxis with group aggregation mechanism and adaptive strategy, and swarming is cancelled. The chemotactic loop with adaptive diffusion mechanism can improve ability of overcoming the "premature", and healthiness is redefined to reduce the computational complexity. 10 complex Benchmark functions were tested. The simulation shows that the GBFO has better search ability and stability, solution quality and efficiency than other typical algorithm up to 80%~90%, 70%~80% among test functions. The comparisons also show GBFO has excellent performance.

Keywords Bacterial foraging optimization, Aggregation, Chemotactic operation, Diffusion

1 引言

菌群优化算法(Bacteria Foraging Optimization, BFO)是Kevin M. Passino[1]基于Ecoli大肠杆菌在人体肠道内吞噬食物的觅食行为而提出的一种新型群体智能优化算法,该算法具有良好的优化性能,吸引了众多研究者对该算法进行探讨[2-5]。相关研究表明,BFO算法的复制操作和迁徙操作在算法设计方面没有新意,其性能主要由趋化操作(包含趋药性操作和聚集操作)决定,趋药性操作中的步长是一个核心的问题,明显存在缺乏自适应性和无差异性的不足,不能很好地体现群体之间的协同和智能性。聚集操作带来的明显问题是参数多,在求解不同优化问题时,不同问题对其中的函数一致性设计存在挑战。总体来说,到现在为止还没有从算法结构上进行改进的研究。

鉴于以上情况,将群体聚集机制和自适应策略集成到趋药性操作中,在趋化循环中引入自适应扩散机制,重新定义健康度,减少计算的复杂性,得到了一种新的群体智能优化方法——广义菌群优化算法(GBFO,Generalized Bacterial Foraging Optimization)。相关实验表明GBFO算法具有优良的性能。

本文第2节是基本菌群优化算法和分析;第3节提出了广义菌群优化算法;第4节是实验;最后总结全文。

2 基本菌群优化算法

简单地说,BFO算法可以通过趋药性操作(Chemotaxis)、聚集操作(Swarming)(趋药性操作和聚集操作组合成为趋化操作)、复制操作(Reproduction)和迁徙操作(Elimination and Dispersal)[1]4个优化过程来实现。

到稿日期:2012-05-11 返修日期:2012-08-03 本文受国家自然科学基金(60873017),国家自然科学基金(中德合作)(61111130183)资助。
陈建超(1979-),男,博士,主要研究方向为智能计算、数据挖掘等,E-mail:cjcat126@126.com;**胡桂武**(1970-),男,博士后,教授,CCF高级会员,主要研究方向为数据挖掘、优化计算等;**杜小勇** 男,教授,博士生导师。

2.1 趋药性操作

BFO 中的趋药性操作可描述如下：

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

式中，$\theta^i(j,k,l)$即个体 i 的位置，j,k,l 分别表示趋药性循环、复制循环、迁移循环的代数，$\phi(j)$表示单位方向向量，$C(i)$表示前进的步长。

2.2 聚集操作

对于最小值问题，设第 i 个细菌个体的信息为 $\theta^i=(\theta_1^i,\theta_2^i,\cdots,\theta_p^i)$，则 BFO 的聚集操作的数学表达式为：

$$\begin{aligned}&J_{cc}(\theta^i,P(j,k,l))\\&=\sum_{x=1}^{S}J_{cc}(\theta,\theta^x(j,k,l))\\&=\sum_{x=1}^{S}\left[-d_{attractant}\exp\left(-w_{attractant}\sum_{m=1}^{p}(\theta_m^i-\theta_m^x)^2\right)\right]+\sum_{x=1}^{S}\\&\left[h_{repellant}\exp\left(-w_{repellant}\sum_{m=1}^{p}(\theta_m^i-\theta_m^x)^2\right)\right]\end{aligned} \tag{2}$$

式中，$d_{attractant}$，$w_{attractant}$ 为引力的深度和宽度，$h_{repellant}$，$w_{repellant}$ 为斥力的高度和宽度，θ_m^i 为个体 i 的第 m 个分量。此公式的含义为整个细菌群体在细菌 i 所处位置 θ^i 处所产生的作用力之和，此时第 i 个个体的适应度的计算公式变为：

$$J(\theta^i)=J(\theta^i)+J_{cc}(\theta^i,P(j,k,l)) \tag{3}$$

因此聚集操作就是对适应值 $J(\theta^i)$进行修正，以达到使细菌聚集的目的。

2.3 复制操作

在执行完指定步数的趋化操作之后，根据各个细菌的健康度进行排序。健康度根据相应细菌在它的生命过程中所吸收到的营养值的多少来计算。健康度定义为：

$$Heath(i)=\sum_{i=1}^{num}J_i \tag{4}$$

式中，$Heath(i)$表示第 i 个细菌的当前健康度，J_i 表示第 i 个细菌的适应度函数值，num 表示到当前位置时，第 i 个细菌所完成的趋化操作的数目。先对整个菌群按照健康度进行排序，健康度较高的细菌有权利进行繁殖行为，生成子代。子代的位置、步长以及其它信息都与父代完全相同。健康度不高的细菌将死亡，以此来维持菌群的规模不变。算法通常的操作是：

群体中要被淘汰的个体个数设为 $S_r=S/2$（S 为群体的规模），首先对群体中的个体按其优劣进行排序，然后将排在较靠后的 S_r 个个体删除，剩下的 S_r 个个体进行自身的复制。

2.4 迁徙操作

在执行完指定次数的复制操作之后，对整个群体中的各个个体进行有选择的迁徙操作。迁移操作是按照给定的概率 P_{ed} 发生的，如果某个体满足迁移操作的条件，那么就将此个体删除，重新生成一个新的个体，相当于将原来的个体移到了一个新的位置。

3 广义菌群优化算法

3.1 菌群优化算法分析

趋药性操作是对大肠杆菌趋药性行为的两个基本动作前进（Swim）和翻转（Tumble）的模仿，其数学模型是式(1)，步长 $C(i)$不能自动反映个体在不同时候的差别以及群体的协同性和智能性，针对不同问题的初始值也无理论依据。

聚集操作是模拟菌群寻取食物的过程中，细菌个体之间通过相互间的作用而实现群体的聚集行为，式(2)是其数学模型。模型中的明显问题是参数多，模型中的函数选择无任何理论支撑，类似的函数选择空间很大，群体间的相互作用是基于类距离的，在求解不同优化问题时，其中的函数一致性设计存在挑战。

复制操作是基于健康度的行为，健康度是根据式(3)和式(4)计算的，在算法执行的晚期明显会引起不正常的震荡，容易误判真实的局部最优解，式(3)中的简单加法在度量不一致时会影响算法的效能，简单地使用健康度高的生存、低的淘汰，从自然真实情况或统计理论上来分析不一定科学。

迁徙操作是模仿实际环境中的细菌被外力杀死或者被驱散到新的区域中去的一种现象，显然具有随机搜索的优点和不足。

3.2 广义菌群优化算法

鉴于以上分析，本文从以下几个方面进行了尝试，首先从总体结构上进行了设计：删除聚集操作，结构上简化趋化操作。其次是按照式(5)重新设计步长，将群体聚集机制和自适应策略集成到趋药性操作中，同时在趋化循环中引入邻域自适应扩散机制，以提高算法的搜索性能，最后是重新定义健康度为适应值之和，以减少计算复杂性。

定义 1（协同步长） 每个细菌的步长 $C(i)$由其周围邻居与它的距离（聚集程度）决定，不再依赖于初始值，实现群体的聚集和自适应。具体设计如式(5)所示。

$$C(i)=f\left(\sum_{j=1}^{N(i)}\|\theta_i-\theta_{i\_j}\|/N(i)\right) \tag{5}$$

式中，$N(i)$是个体 θ_i 最近的邻居数目参数，$f(.)$是一个单调递减非负函数（最小值可以设计足够小），$\|\theta_i-\theta_{i_j}\|$是个体 θ_i 与第 j 个最近邻居 θ_{i_j}之间的距离。

定义 2（邻居平均距离） 个体 θ_i 的邻居平均距离定义如式(6)所示。

$$mean\_dis_i=\sum_{j=1}^{N(i)}\|\theta_i-\theta_{i\_j}\|/N(i) \tag{6}$$

式中，$N(i)$是个体 θ_i 最近的邻居数目。

定义 3（局部超邻域） 称式(7)描述的集合为高维超空间中位置 o_i 的局部超邻域。

$$\Omega_i=\{|\theta|\ \|o_i-\theta\|\leqslant r_n\} \tag{7}$$

式中，r_n 是局部超邻域的半径，为可选参数。

定义 4（自适应扩散） 在趋化操作中，群体会聚集，聚集到一定程度时，邻居平均距离会足够小，导致停滞，为了改变这种局面，当所有个体的邻居平均距离都小于 $d_{diffusion}$（称为“聚集距离最小值”，可初始化为 20）时，进行如下操作：

步骤 1 按照式(8)重新修正 $d_{diffusion}$。

$$d_{diffusion}=\lambda_d\times\max_{i=1}^{S}\{mean\_dis_i\} \tag{8}$$

式中，λ_d 是一个选自区间(0,1)的系数，例如 0.15，S 是种群规模。

步骤 2 依次对每个个体 θ_i，找出离 θ_i 最近的 $N(i)$ 个最近的邻居，按式(9)计算此 $N(i)$ 个个体的虚拟中心，记为 o_i。

$$o_i=\sum_{j=1}^{N(i)}\theta_{i\_j}/N(i) \tag{9}$$

步骤 3 对于个体 θ_i，在 o_i 的局部超邻域 Ω_i 内重新生成 θ_i。

基于以上的定义和设计，广义菌群优化算法完整描述如下：

步骤 1 初始化种群和参数。

步骤 2 令 $l=0$，开始迭代次数为 N_{ed} 的迁徙循环，在循环的每次迭代中，l 递增 1，循环体包含了步骤 3 至步骤 8。

步骤 3　令 $k=0$，开始迭代次数为 N_{re} 的复制循环，在循环的每次迭代中，先让 k 递增 1，健康度都初始化为 0，循环体包含了步骤 4 至步骤 7。

步骤 4　令 $j=0$，开始迭代次数为 N_c 的趋化循环，在循环的每次迭代中，j 递增 1，循环体为步骤 5 至步骤 6。

步骤 5　依次对每个个体 $\theta^i(i=1,2,\cdots,S)$，执行从步骤 5.1 到步骤 5.5 的趋化操作：

步骤 5.1　计算 θ^i 的当前适应值 $J(i,j,k,l)$。

步骤 5.2　为 θ^i 选择一个随机游动方向 Δ^i，并按式(5)计算 θ^i 的游动步长。

步骤 5.3　让 θ^i 按选定的方向和步长，进行最多 N_s 次的游动：在每次游动的时候，先尝试走出一步，得到 θ^i 的临时位置，计算临时适应值 $J'(i,j,k,l)$。如果 $J'(i,j,k,l)<J(i,j,k,l)$，则确认临时位置为 θ^i 的新位置，更新 $J(i,j,k,l)$；否则放弃这个临时位置，并结束 θ^i 的本次趋化操作。

步骤 5.4　把步骤 5.3 得到的 θ^i 的 $J(i,j,k,l)$ 累加给 $health_i$。

步骤 5.5　记录当前个体 θ^i 与其的邻居平均距离。

步骤 6　如果满足自适应扩散的条件，则对每个个体进行自适应扩散。

步骤 7　执行一次复制操作。

步骤 8　执行一次迁徙操作。

4　算法性能测试

4.1　测试函数和参数

为了验证新算法的有效性，对来自文献[3]的 10 个 Benchmark 函数($f1-f10$)进行优化实验。

为了与文献[3]中的算法做比较，与其相同的参数选择了文献[3]中给出的数值，本算法独有的参数则选择恰当值。具体如下：种群规模 $S=50$，函数 $f1-f7$ 的维数为 45，$N_c=100$，$N_s=12$，$N_{ed}=4$，$N_{re}=16$，$P_{ed}=0.25$，$N(i)=6$，$\lambda_d=0.15$，$r_n=0.25$，$d_{diffusion}$ 的初始值为 20。式(5)中的函数选为：$f(x)=\mathrm{rand}(0.2,1)\times\frac{x}{3}$。

10 个测试函数的表达式、各维取值范围和最优解说明如下：

(1)Sphere function：

$$f_1(\vec{x})=\sum_{i=1}^{D}x_i^2$$

$$-100<x_i<100,f_1(\vec{0})=0$$

(2)Rosenbrock's function：

$$f_2(\vec{x})=\sum_{i=1}^{D-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$$

$$-100<x_i<100,f_2(\vec{1})=0$$

(3)Rastrigin's function：

$$f_3(\vec{x})=10D+\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i))$$

$$-10<x_i<10,f_3(\vec{0})=0$$

(4)Griewank's function：

$$f_4(\vec{x})=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$$

$$-600<x_i<600,f_4(\vec{0})=0$$

(5)Ackley's function：

$$f_5(\vec{x})=-20\times\exp[-0.2\sqrt{\frac{1}{D}\times\sum_{i=1}^{D}x_i^2}]-\exp[\frac{1}{D}\times\sum_{i=1}^{D}\cos(2\pi x_i)]+20+e$$

$$-32<x_i<32,f_5(\vec{0})=0$$

(6) Step function：

$$f_6(\vec{x})=\sum_{i=1}^{D}([x_i+0.5])^2$$

$$-100<x_i<100,f_6(\vec{p})=0,-\frac{1}{2}<p_i<\frac{1}{2}$$

(7) Schwefel's problem2.22：

$$f_7(\vec{x})=\sum_{i=1}^{D}|x_i|+\prod_{i=1}^{D}|x_i|$$

$$-500<x_i<500,f_7(\vec{0})=0$$

(8)Shekel's Foxholes：

$$f_8(\vec{x})=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$$

$$-65.536<x_i<65.536,f_8(-32,32)=0.998$$

(9)Six-Hump Camel-Back function：

$$f_9(\vec{x})=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^6$$

$$-5<x_i<5,f_9(0.08983,-0.7126)=f_9(-0.08983,0.7126)=-1.0316285$$

(10)Goldstein-Price function：

$$f_{10}(\vec{x})=\{1+(x_0+x_1+1)^2(19-14x_0+3x_0^2-14x_1-6x_0x_1+3x_1^2)\}\{30+(2x_0-3x_1)^2\times(18-32x_0+12x_0^2+48x_1-36x_0x_1+27x_1^2)\}$$

$$-2<x_i<2,f_{10}(0,-1)=3$$

4.2　实验结果

在 45 或 2 维的 $f1-f10$ 函数优化实验中，GBFO 将与 BFOA[3]，HPSO-TVAC[6]，EA[7] 和 BSO[8] 这 4 种在函数优化方面表现出良好性能的算法进行比较。终止条件是函数评估次数大于 5×10^5 或者优化值低于预先设定的($f1-f7$：0.001，$f8$：0.998，$f9$：−1.0316，$f10$：3.00)，表 1、表 2 为 GBFO、BFOA、HPSO-TVAC、EA 和 BSO 这 5 种算法分别对 $f1-f10$ 的连续 50 次运算的统计结果，它们相同的参数值一样，其它参数见文献[3]。表 1、表 2 中的数据，除了 GBFO 之外，其余都引自于文献[3]。从表 1 可知：

GBFO 对 $f1-f7,f9-f10$ 函数的优化结果从均值和标准差方面来分析，远远优于算法 BFOA，占测试函数总数的 90%。

GBFO 对 $f1-f7,f9-f10$ 函数的优化结果从均值和标准差方面来分析，远远优于算法 HPSO-TVAC，占测试函数总数的 90%。

GBFO 对 $f1-f7,f9-f10$ 函数的优化结果从均值和标准差方面来分析，远远优于算法 EA，占测试函数总数的 90%。

GBFO 对 $f1,f2-f7,f9-f10$ 函数的优化结果从均值和标准差方面来分析，远远优于算法 BSO，占测试函数总数的 80%。

从平均最优值(标准差)的统计结果来分析，GBFO 对 $f1,f4,f6,f8-f10$ 函数优化结果从均值和标准差方面来分析，优于算法 BFOA，HPSO-TVAC，EA 和 BSO 4 种算法的比率占测试函数总数的 80%，总体分析显示了算法具有良好的计算能力和稳定性。

表 1　GBFO、BFOA、HPSO-TVAC、EA 和 BSO 的平均最优值(标准差)的统计结果

Function	Dim	Maximum no. of FES	Mean best value (standard deviation)				
			GBFO	BFOA	HPSO-TVAC	EA	BSO
f_1	45	5×10^5	**0.001(0)**	0.776(0.1563)	0.383(0.05564)	0.257(0.0323)	0.354(0.2239)
f_2	45	5×10^5	40.265(0.4212)	96.873(26.136)	935.2601(1102.352)	67.473(16.3526)	**30.986(4.3438)**
f_3	45	5×10^5	**0.001(0)**	32.9517(10.0034)	46.332(22.4518)	8.536(2.7281)	18.9461(7.7075)
f_4	45	5×10^5	**0.001(0)**	0.6351(0.0522)	0.4748(0.4561)	0.3732(0.0971)	0.5678(0.236)
f_5	45	5×10^5	**0.001(0)**	3.4564(3.4394)	0.9782(0.2029)	0.9298(0.7631)	1.0383(0.2542)
f_6	45	5×10^5	**0.001(0)**	14.7328(3.2827)	13.8478(2.5673)	6.8825(0.6471)	4.2832(0.6476)
f_7	45	5×10^5	**0.0049(0.0052)**	9.4563(10.2425)	5.5674(0.3526)	6.2452(2.3724)	1.7828(0.4652)
f_8	2	5×10^5	1.0735(0.2818)	1.056433(0.01217)	0.9998323(0.00537)	**0.9998329(0.00382)**	0.9998017(0.00825)
f_9	2	1×10^5	**−1.031628(0)**	−0.925837(0.000827)	−1.029922(1.382)	−1.031149(2.527)	−1.031242(0.00759)
f_{10}	2	1×10^5	**3.0000(0)**	3.656285(0.109365)	3.1834435(0.2645)	3.146090(0.06237)	3.443712(0.007326)

注：Function，Dim 分别表示函数和维数，Maximum no. of FES 表示适应值函数评价的最大次数，Mean best value (standard deviation)表示平均最优值(标准差)。

不同算法的机制和参数不一样，单从平均最优值(标准差)来比较不是很全面，从获得预设最优值的次数和适应值评价次数来分析更能体现算法的优越性，总体来说，得到的预设最优值比率越高，适应值评价次数越低，算法性能越好。从表 2 可知：

GBFO 对 $f1, f3-f6, f9-f10$ 函数的优化结果从获得预设值的次数和适应值函数评估平均次数方面来分析，远远优于算法 BFOA，占测试函数总数的 70%。

GBFO 对 $f1, f3-f6, f8-f10$ 函数的优化结果从获得预设值的次数和适应值函数评估平均次数方面来分析，远远优于算法 HPSO-TVAC，占测试函数总数的 80%。

GBFO 对 $f1, f3-f6, f9-f10$ 函数的优化结果从获得预设值的次数和适应值函数评估平均次数方面来分析，远远优于算法 EA，占测试函数总数的 70%。

GBFO 对 $f1, f3-f6, f9-f10$ 函数的优化结果从从获得预设值的次数和适应值函数评估平均次数方面来分析，远远优于算法 BSO，占测试函数总数的 70%。

另外 GBFO 对 $f3-f6, f9-f10$ 函数进行优化获得预设值的次数为 50，并且函数评估次数低于 BFOA，HPSO-TVAC，EA 和 BSO 4 种算法，占测试函数总数的 60%。

从获得预设值的次数(适应值函数评估平均次数)的统计结果来看，GBFO 具有好的最优解次数的比率高、适应值函数评价次数少的优点，显示了算法良好的求解质量和效率。

表 2　GBFO、BFOA、HPSO-TVAC、EA 和 BSO 获得预设值的次数(适应值函数评估平均次数)的统计结果

Function	Dim	No. of runs converging to the pre-defined objective function value(mean no. of FEs required)				
		GBFO	BFOA	HPSO-TVAC	EA	BSO
f_1	45	**50**(96710.72)	24(172228.73)	45(84712.34)	41(36523.46)	39(74782.63)
f_2	45	0	0	0	0	1(127687)
f_3	45	**50**(233643.48)	0	0	1(372833)	0
f_4	45	**50**(105809.56)	0	7(292643.54)	4(394852.75)	6(475832.65)
f_5	45	**50**(239478.16)	0	文献中数据无效	18(264723.57)	12(472631.67)
f_6	45	**50**(122849.12)	0	0	34(265732.58)	25(748237.40)
f_7	45	0	0	0	0	**23**(126377.65)
f_8	2	34(2431.44)	38(32928.14)	23(26843.92)	**50(30272.74)**	**50**(19823.70)
f_9	2	**50**(3879.38)	46(25374.87)	40(31928.70)	50(28372.74)	50(66290.80)
f_{10}	2	**50**(2450.54)	35(139584.44)	30(347285.80)	42(129372.87)	50(126574.64)

注：Function，Dim 分别表示函数和维数，No. of runs converging to the pre-defined objective function value(mean no. of FEs required)表示预设值的次数(适应值函数评估平均次数)。

结束语　广义菌群优化算法建立在一种新体系结构层面上，可以根据群体之间的信息自适应调控趋化步长，消除其对初始步长的依赖，实现种群之间的协调性和智能性，有效地提高算法的搜索能力。为检验算法解决复杂优化问题的能力，10 个 45 或 2 维 Benchmark 函数被用于测试，实验表明 GBFO 的优化结果明显要好于文献[3]中记录，证明了 GBFO 不仅是可行的，而且是有效的算法。同时，这种机制也为遗传算法、粒子群优化算法、差分进化算法等群体智能算法的改进提供了一种新思路，下一步的工作是研究将这种思想应用于其它智能算法以及理论方面。

参 考 文 献

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

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

[3] Dasgupta S, Das S, Abraham A. Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis[J]. IEEE Trans. Evol. Comput., 2009, 13(4): 919-941

[4] Chatzisa S P, Koukas S. Numerical optimization using synergetic swarms of foraging bacterial populations[J]. Expert Systems with Applications, 2011, 38(12): 15332-15343

[5] Majhi R, Panda G, Majhi B, et al. Efficient prediction of stock market indices using Adaptive Bacterial Foraging Optimization (ABFO) and BFO based techniques[J]. Expert Systems with Applications, 2009, 36(6): 10097-10104

[6] Tang M S, Tang W J, Wu W H, et al. Bacterial foraging algorithm with varying population for optimal power flow[J]. Lecture Notes in Computer Science, 2007, 4448: 32-41

[7] Ratnaweera A, Halgamuge K S. Self organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients [J]. IEEE Trans. Evol. Comput., 2004: 8(3): 240-254

[8] Bäck T. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms [C]// Proc. Genetic Algorithms. London, U. K.: Oxford Univ. Press, 1996