文章编号: 1005−8152(2011)02−0223−06

细菌菌落优化算法

李　明[1], 杨成梧[2]

(1. 西南林业大学 交通机械与土木工程学院, 云南 昆明 650224; 2. 南京理工大学 动力工程学院, 江苏 南京 210094)

摘要: 根据细菌菌落生长演化的基本规律, 提出一种新的细菌菌落优化算法. 首先, 依据细菌生长繁殖规律, 制定符合算法需要的个体进化机制. 其次根据细菌在培养液中的觅食行为, 建立算法中个体泳动、翻滚、停留等运动方式. 最后, 借鉴菌落中细菌信息交互方式, 建立个体信息共享机制. 另外, 该算法提供了一种新的结束方式, 即在没有任何迭代次数或精度条件的前提下, 算法会随着菌落的消失而自然结束, 并且可以保持一定的精度. 通过与两类PSO算法比较的仿真实验验证了细菌菌落优化算法的效果, 通过仿真实验验证了细菌菌落优化算法自然结束过程.

关键词: 菌落; 群集智能; 优化算法; 进化机制

中图分类号: TP273　　**文献标识码**: A

Bacterial colony optimization algorithm

LI Ming[1], YANG Cheng-wu[2]

(1. College of Communication, Machinery and Civil Engineering, Southwest Forestry University, Kunming Yunnan 650224, China;
2. College of Power Engineering, Nanjing University of Science & Technology, Nanjing Jiangsu 210094, China)

Abstract: A bacterial colony optimization(BCO) algorithm based on the basic growth law of bacterial colony is presented. Firstly, an evolutionary mechanism for the individual of the BCO algorithm is designed by the reproduction law of bacterium. Secondly, swimming, tumbling and dwelling moving modes are established for the individual by the basic foraging theory of bacterium. Finally an information sharing method is built for the colony. This BCO algorithm provides a new type of termination: it will terminate the iteration when the colony vanishes, regardless of the iteration number or the precision value. The performance of the BCO algorithm is verified by some comparative simulations with two particle swarm optimization algorithms. Other simulations are employed to test the new termination type.

Key words: bacterial colony; swarm intelligence; optimization algorithm; evolutionary mechanism

1　引言(Introduction)

群集智能(swarm intelligence)作为一种新兴的智能计算技术已成为人工智能研究者关注的焦点. 2002年Passino和Müller等分别根据细菌的觅食和趋化行为提出了用于解决实数优化问题的细菌觅食优化(bacterial foraging optimization, BFO)算法[1]及细菌趋化性优化(bacterial chemotaxis optimization, BChO)算法[2]. 随后由Abraham, Biswas, Kim等人组成的研究团队对BFO算法进行了一系列研究, 论证了BFO算法中的繁殖操作有利于提高算法收敛速度[3], 分别利用基因算法和微分进化算法改善BFO算法搜索全局最优的能力[4,5], 并将BFO算法用于PID控制器参数优化中[6]. 另外, BFO算法还被应用于设计模糊PID控制器参数[7]、矩形微带天线响应频率计算[8]、金融期权参数估计[9]、股市预测[10]、自适应信道均衡器设计[11]、PI控制器参数优化设计[12]、电力系统无功电力调度[13]等方面.

国内在这方面的研究不是很多, 主要是李威武等人根据Müller等人提出的单个细菌趋化模型, 借鉴PSO算法提出了一种细菌群体趋化性(bacterial colony chemotaxis, BCC)算法[14]; 刘文霞等人用微分进化和混沌迁移的方法对BCC算法加以改进, 以提高算法的全局搜索能力[15]; 陈瀚宁等人通过改进个体交流方式以增强BFO算法的寻优能力[16]; 储颖等人则参考PSO算法, 使个体可以感知种群最优位置, 以改善BFO算法性能, 并将其用于图像压缩处理中[17,18].

现在对细菌算法的研究主要集中在算法改进方面, 对算法机理的研究尚不多见. 本文将根据细菌菌落的生长演化过程, 在分析和比较BFO及BChO算法的基础上, 建立新的细菌运动模型及繁殖和死亡机制, 提出一种新的细菌菌落优化算法.

收稿日期: 2009−07−07, 收修改稿日期: 2009−12−13.
基金项目: 云南省应用基础研究面上资助项目(2009CD070); 云南省教育厅资助项目(08Y0204).

2 BFO和BChO算法(BFO and BChO algorithms)

现在基于细菌的群集智能算法的研究几乎都是采用BFO或BChO算法模型, 为此, 首先简单介绍这两种算法, 然后通过与PSO算法的比较, 分析他们的优化能力.

在BFO算法中, 问题的解空间对应着培养基, 算法中的个体对应着单个细菌, 解空间中某位置的适应值(fitness value)和相应位置培养基的浓度成正比, 即适应值越高意味着浓度越高. 个体有翻滚(tumbling)和泳动(swimming)两种运动方式, 翻滚就是个体在当前位置附近沿任意方向移动, 泳动是个体沿某个特定的方向移动. 为了保证算法的顺利进行, 还需要设定算法中个体的数量, 个体的生命周期等参数. 具体的算法过程如下:

每次迭代开始时, 个体首先通过翻滚转移到新的位置, 如果新的位置具有较好的适应值, 则沿着前次翻滚的方向泳动, 直到适应值产生下降或到达规定的最大泳动步数, 否则进入下一次迭代. 达到规定的生命周期($N_{\rm C}$)后, 依据个体在所有$N_{\rm C}$次迭代过程中适应值的和, 将较差的一半个体消灭, 另一半则进行复制, 整个过程保持种群数量不变. 个体在移动过程中还会释放一定的化学物质, 向同伴传达自身所处位置的营养物浓度, 以便于对方决定是否向该位置靠拢. 在算法执行过程中, 个体会依一定的概率突然消失, 类似于细菌遭遇偶然事件(如有害物质等)死亡, 在个体消失的同时, 会在解空间中任意产生一个新的个体, 以保持个体总量的不变. 当达到一定的迭代次数或精度则算法结束.

BChO算法是利用Dahlquist等人提出的实际细菌趋化模型建立的一种优化算法. 解空间中某个位置的适应值正比于溶液中相应位置的引诱剂浓度, 并假设个体移动路线是由一系列的直线段组成, 每段轨迹有速度、方向、移动时间3个参数, 其中速度设定为常数. 个体改变运动方向时, 向左或向右拐的概率相同; 个体在各段轨迹上的移动时间和相邻轨迹的夹角都由概率分布决定; 个体每次的移动时间服从指数分布; 细菌在各段轨迹上的移动时间和相邻轨迹的夹角是和以前移动轨迹相互独立的两个参数.

算法执行时, 首先将个体放置在某个位置, 然后产生一个随机数t, 并将其代入规定的移动时间概率分布函数, 计算出该次移动的时间. 接着确定移动方向, 即计算下一移动轨迹与当前移动轨迹的夹角, 它服从高斯分布, 与移动时间的计算类似. 最后, 根据计算的移动时间及移动方向, 确定个体的新位置. 算法也是以迭代次数与精度作为结束条件.

上述两种算法与PSO算法类似, 都是求解实数优化问题, 但是在个体模型和群体中信息交换方式上存在差异. PSO算法是根据鸟群觅食过程建立的一种群集智能优化算法, 从算法角度讲, 个体具有较强的“记忆”能力和“感知”能力, 即个体能够记住自身经历的最优位置和感知整个群体所经历过的最优位置. 并即根据上述两个信息, 结合自身的搜索经验, 自动调节搜索的速度及方向, 向这两个最优位置移动.

在BFO算法中, 个体在整个迭代过程中惟一能记住的就是每次迭代时的适应值, 同时只能感知群体当前的最优位置, 由于不了解自身及种群过去的经历, 所以个体搜索没有任何经验可供参考, 与PSO算法中的个体相比, 个体像是在“漫无目的”地搜索. 另外, 个体通过释放类似ACO算法中“信息素”的化学物质吸引或排斥其他个体, 仅利于个体向当前群体的最优位置聚拢, 而在PSO算法中, 个体通过感知聚向群体经历的最优位置. BChO算法中个体同样缺乏“记忆”和“感知”能力, 其个体每次的移动完全由概率决定, 事实上, 解空间每一维都有“左”和“右”两个方向, 对于低维的解空间, 个体选择正确的移动方向的概率更大一些. 例如, 对于二维解空间, 个体每次选择到正确移动方向的概率是1/4, 但对于高维或多极值的情况, 这个概率明显要低得多. 这就意味着个体很难选择到正确的移动方向, 必然导致个体在某个位置“徘徊”很久, 文献[2]中对多峰函数的仿真实例也证明了这一点. 因此, 与PSO算法相比, BFO和BChO算法无论是个体还是种群的搜索能力都存在明显不足.

现有细菌算法虽然是模拟细菌的觅食或趋化行为, 但实际上无论在个体和种群上都没有体现细菌觅食或趋化过程作为优化算法模型的优点.

3 细菌菌落优化算法(bacterial colony optimization algorithm)

从微生物学角度讲, 细菌通常具有生命周期短、繁殖快、对环境敏感等特点, 细菌在培养基中的觅食过程, 必然伴随着细菌生长、繁殖、死亡等一系列过程[19]. 微生物学研究表明, 细菌群体在培养基中遵循一定的生长规律, 即细菌群体的生长将经历延滞期(lag phase)、指数期(exponential phase)、稳定期(stationary phase)和衰亡期(death phase)4个阶段, 延滞期是指单个或少量细菌接种到培养基中后适应环境的过程, 在这个阶段细菌数量基本保持不变; 一旦细菌吸收到足够的营养物质, 个体将进行繁殖, 由于细菌多以二分裂方式繁殖且代时较短, 所以此时细菌群体数量将呈指数增长, 并由此形成菌落(colony), 这就是指数期; 由于培养基中营养物质是有限的, 指数期将很快过渡到稳定期, 在这个阶段群体数量将保持稳定, 也就是新繁殖的细菌数与

衰亡的细菌数相等; 随着营养物质的不断消耗, 个体死亡速度超过新生速度, 群体数量不断减少, 直至消失[19]. 细菌在培养基中的生长过程就是菌落形成、发展直到消失的演化过程.

现有细菌优化算法虽然受启发于细菌的觅食, 但在算法中并没有体现出前述细菌觅食的特点. 首先在这些算法中, 群体的数量总是保持不变, 这显然不符合细菌的生长规律. PSO算法模拟了鸟群或鱼群的一次觅食过程, 在他们的一次觅食过程中完全可以不考虑个体的繁殖和死亡, 因此种群数量被认为保持不变, 但由于细菌具有生命周期短、繁殖快等特点, 显然在细菌觅食过程中必然会伴随群体数量的不断变化. 从这一点上讲, 现有细菌算法与PSO算法没有区别, 更形象地说这些算法中的个体只是一些缩小了的“鸟”.

其次, 现有细菌算法中个体具有相同的生命周期, 换句话说, 所有细菌经历相同的时间后, 将同时面临繁殖或死亡的选择. 事实上, 营养物质的摄取将直接决定细菌存亡, 一旦细菌获得充足的营养物质, 则可进行分裂繁殖, 相反地, 一旦缺乏营养物质或遭遇有害物质, 则将面临死亡.

最后, 现有细菌算法中个体运动方式略显单调, 虽然也有翻滚和泳动两种形式, 但他们并没有充分体现细菌对环境非常敏感的特性. 由于营养物质对细菌生死存亡起着至关重要的作用, 所以细菌对环境始终保持高度敏感, 并且通过翻滚、停留、泳动等多种运动方式应对环境变化.

依据微生物生长规律, 本文提出了一种新的细菌菌落优化算法(bacterial colony optimization, BCO), 算法的主要思路是: 首先在解空间中放置单个或少量个体, 并仿照微生物生长规律, 制定新的个体生存繁殖和死亡机制, 当细菌连续沿正的浓度梯度方向移动的次数达到某个设定值($N_{\rm H}$)后, 意味着细菌吸收了足够的营养, 从而达到繁殖条件; 反之, 当细菌连续沿负的浓度梯度(从高适应值向低适应值)方向移动的次数达到某个设定值($N_{\rm L}$)后, 意味着细菌一直忙于奔波, 吸收的营养很难维持生存, 从而达到死亡条件. 由于细菌繁殖速度快, 因此算法中需要设定种群的最大规模S.

其次, 对于那些时而沿正方向时而沿负方向移动的细菌, 当他们这种反复的移动次数达到最长寿命(N)后, 将会自然死亡, 显然N在数值上大于$N_{\rm H}$和$N_{\rm L}$. 因此, 细菌繁殖的条件是惟一的, 但是死亡的情况却有两种——一种是“营养不够”的突然死亡, 另一种是“年龄过大”的自然死亡.

最后设定算法中个体(即细菌)具有3种基本运行方式: 泳动、翻滚、停留, 当个体第k次迭代的适应值优于第$k-1$次的适应值, 则个体在$k+1$次首先作短暂的停留, 再选择泳动方式; 一旦个体第k次的适应值不如第$k-1$次的适应值, 则个体在$k+1$次直接选择翻滚方式, 并且每个个体都能感知整个菌落曾经经历过的最优位置. 具体地讲, 在泳动时, 细菌将沿着前一次的移动方向向群体经历的最优位置移动, 其算法模型为

$$\begin{aligned}\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + C_1 \cdot r_1 \cdot (\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) + \\ C_2 \cdot r_2 \cdot (\boldsymbol{g} - \boldsymbol{x}_k).\end{aligned} \tag{1}$$

翻滚时个体将沿着与前次移动方向相反的方向向群体经历的最优位置移动. 其算法模型为

$$\begin{aligned}\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - C_1 \cdot r_1 \cdot (\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) + \\ C_2 \cdot r_2 \cdot (\boldsymbol{g} - \boldsymbol{x}_k).\end{aligned} \tag{2}$$

细菌每次移动到浓度更高的区域, 都要作短暂的停留, 体现在算法上就是, 当个体移动到适应值更高的位置后, 会在该位置停留并在其附近作随机搜索, 这就相当于细菌进入更高浓度区域后, 将作短暂停留以吸收一定的营养物质, 满足自身成长需求. 其算法模型为

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + R \cdot \boldsymbol{r}, \tag{3}$$

其中: $\boldsymbol{x} = (x_{1,k}, x_{2,k}, \cdots, x_{d,k})$是第$k$次迭代时的位置; $\boldsymbol{g} = (g_1, g_2, \cdots, g_d)$是菌落所经历的最优位置, d是解空间维数; r_1和r_2是区间[0, 1]上的随机数; R, c_1, c_2是常数; $\boldsymbol{r} = (r_1, r_2, \cdots, r_d)$, r_i是区间[0, 1]上的随机数.

与其他群集智能算法类似, BCO算法可以以迭代次数或精度作为结束条件, 另外, 由于BCO模拟了细菌菌落演化的全过程, 因此当种群中个体全部死亡后, 算法自然结束.

4　仿真实例(Simulation)

为了说明BCO算法的性能, 本文将它与线性惯性权[20]及带收缩因子[21]的两种经典PSO算法进行比较实验, 首先选取3个标准的非线性测试函数, 他们分别是:

1) Sphere函数

$$f_1(x) = \sum_{i=1}^{n} x_i^2. \tag{4}$$

2) Rastrigrin函数

$$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10). \tag{5}$$

3) Griewank函数

$$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1. \tag{6}$$

其中$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$是$n$维实向量.

上述3个标准测试函数中, Sphere函数是单峰函数, 其他两个函数是多峰函数, 并且3个函数的最小值均为0, 在所有实验中, 这3个函数均取30维. 惯性权PSO算法的参数设置为: $w = 0.9 - 0.5t/t_{\max}$,

$c_1 = c_2 = 2$, 其中: t为当前迭代次数, $t_{\max}$为最大迭代次数; 带收缩因子的PSO算法的参数设置为: $k = 0.729$, $\varphi = 4.1$; 两种PSO算法中微粒子数目均为30. 在BCO算法中, 初始个体为1, 最大种群数量为$S = 30$, 个体最长寿命$N = 4$, 繁殖条件$N_{\rm H} = 2$, 死亡条件$N_{\rm L} = 2$, 系数$c_1 = c_2 = 0.2$, $R = 0.1$.

对所有个体的初始值采用不对称的选择方式, 他们的初始值范围、目标精度、PSO算法中的个体速度和位置极限值如下表所示, 所有实验的最大迭代次数为5000次.

表 1 各测试函数的参数范围

Table 1 Parameter limits of three test functions

	初始值范围	目标精度	$x_{\max} = v_{\max}$
f_1	$(50, 100)^{30}$	10^{-5}	200
f_2	$(2.56, 5.12)^{30}$	100	10
f_3	$(300, 600)^{30}$	0.05	1200

为了确保实验客观可信, 本文对上述3种算法分别进行20次独立实验, 记录每次达到目标精度所需的迭代次数, 如表2～4所示, 其中$\rm PSO_{lw}$代表线性惯性权PSO算法, $\rm PSO_{cf}$表示带收缩因子的PSO算法.

根据表2可以算出两种PSO算法达到预期目标分别平均需要1883和614次, BCO算法平均需要迭代211次.

表3表明对于对于Rastrigrin函数, 带收缩因子的PSO算法在所有20次实验中, 均未能达到目标精度, 表中数据是每次达到的最小精度. 经计算, 线性惯性权PSO算法达到预期目标精度平均需要迭代1805次, BCO算法平均需要迭代仅9次.

根据表4可以算出, 线性惯性权PSO算法达到预期目标精度平均需要迭代3205次, 带收缩因子的PSO算法在20次实验中, 有5次在5000次迭代中不能达到目标精度, 其他15次能达到目标精度的平均迭代次数为386, BCO算法平均需要迭代仅10次.

表 2 Sphere函数的实验数据

Table 2 Experimental data of Sphere function

$\rm PSO_{lw}$					$\rm PSO_{cf}$					BCO				
1906	1945	1938	1839	1827	596	614	552	620	583	248	173	115	195	253
1855	1868	1853	1931	1904	642	590	620	618	546	137	295	215	77	95
1917	1831	1845	1967	1871	751	588	583	579	718	185	302	319	222	234
1913	1890	1828	1879	1852	694	573	510	574	622	223	304	223	224	181

表 3 Rastrigrin函数的实验数据

Table 3 Experimental data of Rastrigrin function

$\rm PSO_{lw}$					$\rm PSO_{cf}$					BCO				
1744	2205	1821	2143	1848	317.4	181.1	193.0	250.7	197.9	8	8	9	8	9
1440	1783	1982	1771	1408	345.2	113.4	284.6	270.1	234.8	9	11	8	8	8
1563	1798	1924	1825	1767	272.6	270.6	247.7	191.0	305.4	8	9	9	8	8
1800	1818	1870	1811	1784	220.9	203.9	268.6	232.8	157.2	9	8	9	12	8

表 4 Griewank函数的实验数据

Table 4 Experimental data of Griewank function

$\rm PSO_{lw}$					$\rm PSO_{cf}$					BCO				
3081	3246	3291	3075	3134	465	330	5000	514	5000	9	9	11	9	9
3075	3594	3135	3245	3185	414	323	298	467	337	11	8	12	11	8
3231	3149	3240	3197	3207	5000	418	352	5000	407	10	9	12	9	11
3208	3223	3166	3169	3249	373	431	5000	315	352	9	11	9	9	12

上述比较实验的结果说明了本文提出的BCO算法的有效性, 尤其对于Rastrigrin函数和Griewank函数两种多峰函数, BCO算法具有明显优势. 需要说明的是, BCO算法虽然收敛更快, 但该算法需要设定的参数也较多, 一些参数设置不当甚至会造成算法无法进行, 上述数据是在特定的参数条件下的实验结果, 至于参数对算法性能的影响还需要进一步地研究.

上述实验中, 3种算法均以精度和迭代次数作为结束条件. 接着本文仍以上述3个测试函数为例, 通过仿真实验说明BCO算法的自然结束方式. 在所有仿真实验中初始个体均为1, 个体最长寿

命、繁殖条件和死亡条件及各系数均保持不变. 在对Sphere函数仿真实验中, 最大种群数量S设定为22; 对Rastrigrin函数和Griewank函数仿真实验时S设定为12. 图1~3分别绘制了3种情况下适应值及种群规模的变化.

图1(a)显示精度在0.001~0.01之间. 种群规模变化符合菌落生长演化的基本规律, 迭代次数小于6时, 相当于延滞期; 第7~10次迭代相当于指数期, 此时种群数量明显增加; 第10~60次迭代相当于稳定期, 该阶段种群数量变化不大; 第60次迭代以后相当于衰亡期, 该阶段种群迅速消亡. 图1(b)表明算法在没有任何外加条件的情况下, 经过70次迭代后, BCO算法会自然结束.

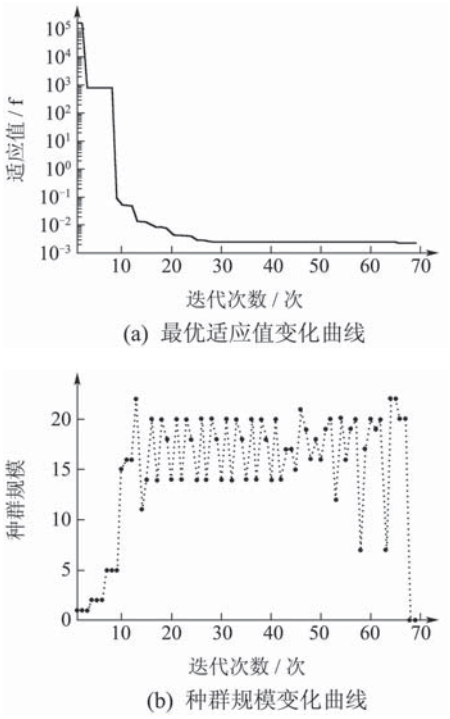
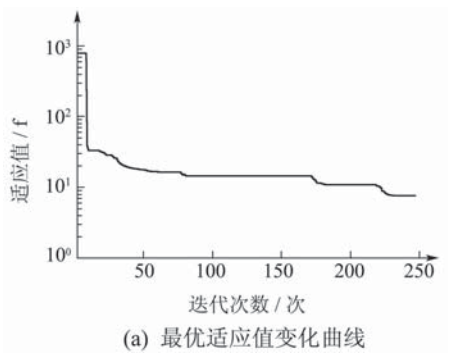


图 1 Sphere函数仿真曲线

Fig. 1 Simulation curve of Sphere function

图2表明对于Rastrigrin函数在采用上述设置时, 在经过250次左右迭代后, 算法就自然结束, 精度在1~10之间. 其种群规模变化曲也符合细菌菌落演化基本规律, 只是在稳定期种群规模变化波动较大.



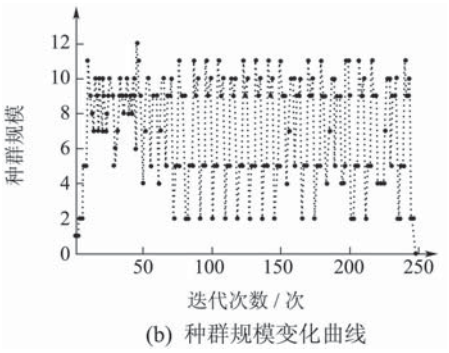


图 2 Rastrigrin函数仿真曲线

Fig. 2 Simulation curve of Rastrigrin function

图3表明对于Greiwank函数在采用上述设置时, 在经过60次左右迭代后, 算法就自然结束, 精度在0.001~0.01之间. 与图2类似, 在整个迭代过程中种群规模变化波动较大.

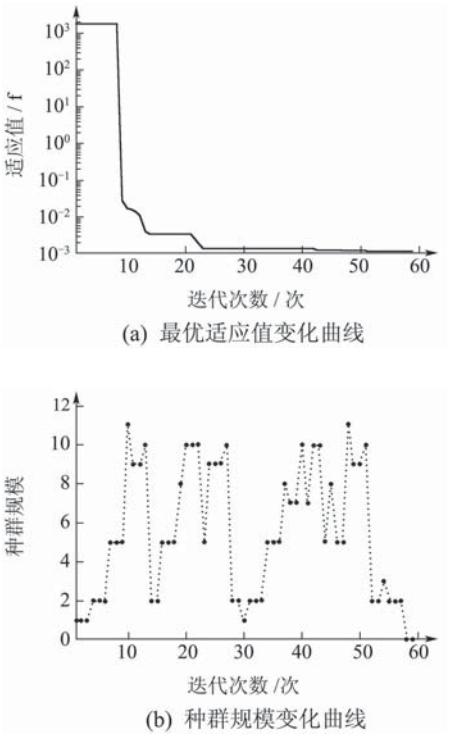


图 3 Griewank函数仿真曲线

Fig. 3 Simulation curve of Griewnak function

上述3组实验主要为了说明BCO算法自然结束的方式, 对于Sphere函数而言, 虽然算法在70次左右就可以结束, 但精度不很理想, 事实上, 通过提高种群规模可以适当提高算法精度, 但自然结束所需要的迭代次数将迅猛增加. 对于Rastrigrin函数和Griewank函数, 算法在自然结束时精度较高, 这也进一步表明该算法对多峰函数更有优势.

5 结论(Conclusion)

本文模拟细菌菌落生长演化过程, 设计了一种新的群集智能优化算法, 该算法主要用于实函数

优化问题. 问题解空间相当于细菌培养液, 算法中的个体相当于细菌, 根据细菌觅食和生长繁殖的基本规律, 设计了符合算法中个体的泳动、翻滚及停留等3种运动方式及运动规律, 同时制定了个体繁殖及死亡机制. 算法执行时, 将单个或少量个体放置到解空间中, 个体按照设定的运动规律搜索最优解, 在搜索过程中种群数量将按照细菌菌落生长的基本规律变化, 当细菌菌落消失后, 则算法可以自然结束. 仿真实验说明了BCO算法的有效性, 尤其对多峰函数, 其优势比较明显. 但是, BCO算法的收敛性及参数设置等一系列问题还需要作进一步研究和探讨.

参考文献 (References):

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

[2] MULLER S D, MARCHETTO J, AIRAQHI S, et al. Optimization based on bacterial chemotaxis[J]. *IEEE Transactions on Evolutionary Computation*, 2002, 6(1): 16 – 29.

[3] ABRAHAM A, BISWAS A, DASQUPTA S, et al. Analysis of reproduction operator in bacterial foraging optimization algorithm[C] //*Proceedings of 2008 IEEE Conference on Evolutionary Computation*. New York: IEEE, 2008: 1476 – 1483.

[4] KIM H D, ABRAHAM A, CHO H J. A hybrid genetic algorithm and bacterial foraging approach for global optimization[J]. *Information Sciences*, 2007, 17(18): 3918 – 3937.

[5] BISWAS A, DASQUPTA S, DAS S, et al. A synergy of differential evolution and bacterial foraging optimization for global optimization[J]. *Neural Network World*, 2007, 17(6): 607 – 626.

[6] KIM H D, CHO H J. Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization[C] //*Proceedings of 3rd International Atlantic Web Intelligence Conference on Advances in Web Intelligence*. New York: IEEE, 2005: 231 – 235.

[7] CHEN H C. Bacterial foraging based optimization design of fuzzy PID controllers[C] //*Proceedings of 4th International Conference on Intelligent Computing*. New York: IEEE, 2008: 841 – 849.

[8] GOLIPUDI S V R S, PATTNAIK S S, BAJPAI O P, et al. Bacterial foraging optimization technique to calculate resonant frequency of rectangular microstrip antenna[J]. *International Journal of RF and Microwave Computer-Aided Engineering*, 2008, 18(4): 383 – 388.

[9] DANG J, BRABAZON A, ONEILL M, et al. Option model calibration using a bacterial foraging optimization algorithm[C] //*Proceedings of Applications of Evolutionary Computing*. New York: IEEE, 2008: 113 – 122.

[10] MAJHI R, PANDA G, SAHOO G, et al. Stock market prediction of S and P 500 and DJIA using Bacterial Foraging Optimization technique[C] //*Proceedings of 2007 IEEE Conference on Evolutionary Computation*. New York: IEEE, 2007: 2569 – 2575.

[11] MAJHI B, PANDA G, CHOUBEY A. On the development of a new adaptive channel equalizer using bacterial foraging optimization technique[C] //*Proceedings of 2006 Annual India Conference*. Piscataway, NJ, USA: IEEE, 2006: 65 – 70.

[12] MISHRA S, BHENDE C N, LAI L L. Optimization of a distribution static compensator by bacterial foraging technique[C] //*Proceedings of 2006 International Conference on Machine Learning and Cybernetics*. New York: IEEE, 2006: 4075 – 5082.

[13] LU Z, LI M S, TANG W J, et al. Multi-objective optimization of reactive power dispatch using a bacterial swarming algorithm[C] //*Proceedings of 26th Chinese Control Conference*. New York: IEEE, 2007: 460 – 464.

[14] 李威武, 王慧, 邹志君, 等. 基于细菌群体趋药性的函数优化方法[J]. 电路与系统学报, 2005, 10(1): 58 – 63.
(LI Weiwu, WANG Hui, ZOU Zhijun, et al. Function optimization method based on bacterial colony chemotaxis[J]. *Journal of Circutis and Systems*, 2005, 10(1): 58 – 63.)

[15] 刘文霞, 刘晓茹, 张建华, 等. 基于微分进化和混沌迁移的细菌群体趋药性算法[J]. 控制理论与应用, 2009, 26(4): 353 – 357.
(LIU Wenxia, LIU Xiaoru, ZHANG Jianhua, et al. Bacterial colony chemotaxis algorithm based on differential evolution and chaos migration[J]. *Control Theory & Applications*, 2009, 26(4): 353 – 357.)

[16] CHEN H N, ZHU Y L, HU K U, et al. Cooperative approaches to bacterial foraging optimization[C] //*Proceedings of 2008 IEEE Conference on Evolutionary Computation*. New York: IEEE, 2008: 541 – 548.

[17] CHU Y, MI H, LIAO H, et al. A fast bacterial swarming algorithm for high-dimensional function optimization[C] //*Proceedings of 4th International Conference on Intelligent Computing*. New York: IEEE, 2008: 3135 – 3140.

[18] 储颖, 邵子博, 糜华, 等. 细菌觅食算法在图像压缩中的应用[J]. 深圳大学学报(理工版), 2008, 25(2): 153 – 157.
(CHU Ying, SHAO Zibo, MI Hua, et al. An application of bacterial foraging algorithm in image compression[J]. *Journal of Shenzhen University(Science & Engineering)*, 2008, 25(2): 153 – 157.)

[19] 周德庆. 微生物学教程[M]. 北京: 高等教育出版社, 2002.
(ZHOU Deqing. *Essential Microbiology*[M]. Beijing: Higher Education Press. 2002.)

[20] SHI S Y, EBERHART R. A modified particle swarm optimizer[C] //*Proceedings of the IEEE International Conference on Evolutionary Computation*. New York: IEEE, 1998: 69 – 73.

[21] CLERC M, KENNEDY J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space[J]. *IEEE Transactions on Evolutionary Computation*, 2002, 6(1): 58 – 73.

作者简介:

李 明 (1977—), 男, 副教授, 博士, 硕士生导师, 主要研究方向为神经网络控制、智能群集算法等, E-mail: swfu_lm@swfu.edu.cn;

杨成梧 (1937—), 男, 教授, 博士生导师, 主要研究方向为非线性控制、迭代控制、智能优化算法等.