文章编号:1000－2367(2016)02－0148－09 **DOI**:10.16366/j.cnki.1000－2367.2016.02.027

基于云模型的细菌觅食优化算法

崔金玲[1],吴 迪[2]

(1.安阳师范学院 计算机与信息工程学院,河南 安阳 455000;2.河南师范大学 计算机与信息工程学院,河南 新乡 453007)

摘 要:针对细菌觅食优化算法收敛速度慢、容易陷入局部极值点出现早熟的问题,提出一种新的基于云模型优化的细菌觅食优化算法.首先给出了细菌灵敏度的概念,结合云模型随机性和稳定倾向性的特点,运用了 X 条件云发生器来调整细菌灵敏度,控制游动步长,进行了趋向性操作和复制操作,改进了标准的细菌觅食优化算法,提高了算法的收敛速度.然后利用正向正态云发生器,修正非线性自适应的迁移概率,进行了迁移操作,增强了算法的全局寻优能力.将该算法应用于自动组卷系统中,与遗传算法进行实验比较分析,结果表明:该算法的收敛速度与优化质量均优于遗传算法.

关键词:细菌觅食优化算法;云模型;自动组卷

中图分类号:TP18 **文献标志码**:A

2002 年,Passino 模拟人类大肠杆菌觅食行为提出了细菌觅食优化算法(Bacteria Foraging Optimization Algorithm,BFOA),这是一种智能仿生算法[1].BFOA 通过细菌种群之间的竞争与合作完成问题空间的优化求解,是一种基于整体种群的搜索优化技术.由于 BFOA 构造的直观性与易理解的自然机理,吸引了来自不同领域的学者对其进行研究.文献[2]采用余弦函数改进了趋向性操作的游动步长,提高了算法的局部搜索能力;文献[3]引入免疫算法中克隆选择的思想,改进了复制操作,提高了算法的搜索精度;文献[4]采用启发式准则改进了迁移操作,提高了收敛速度.文献[5]设计了多重趋向性操作,从而充分搜索局部最优解,并应用于解决柔性车间调度问题;文献[6]结合 BFOA 与归一化准则分割彩色图像,提高了算法的稳定性;文献[7]采用粒子群优化算法,更新 BFOA 中细菌个体的位置,提高了算法的收敛速度;文献[8]采用 BFOA 优化支持向量机的参数.总之,对 BFOA 进行了一系列改进,讨论了它在各个领域的应用,改进后的 BFOA 广泛应用于车间调度、图像处理、PID 控制器设计、故障诊断等中.但是,BFOA 还没有被应用到自动组卷系统中,而细菌在觅食过程中容易产生模糊信息,且文献[2－8]研究时没有充分考虑自然环境的随机性和模糊信息的处理.

李德毅院士于 1995 年提出了云模型[9],云模型采用正态分布表示自然语言概念,将定性概念的模糊性和随机性有机地结合在一起,从而实现了定性概念与定量数值之间的自然转换.正态云模型是最重要的一种模型,在文献[10]中,李德毅院士论证了正态云模型的普适性,文献[11]讨论了正态云模型在知识表达时模糊中带有确定性、稳定中带有变化的规律,体现了自然界物种生存进化的基本原理.

上面分析了细菌觅食优化算法和云模型的研究现状,本文在细菌觅食优化算法的研究基础上,结合云模型的特点,提出一种新的基于云模型优化的细菌觅食优化算法.

1 细菌觅食优化算法

在文献[12－16]的基础上,给出细菌觅食优化算法,其基本思想为:在大自然中,大肠杆菌通过其表面鞭

收稿日期:2015-05-06;**修回日期**:2016-01-21.

基金项目:河南省基础与前沿技术研究(122300410353;142300410084);教师教育精品资源共享课程(河南省教育厅教师〔2013〕136 号).

第 1 作者简介(通信作者):崔金玲(1965－),女,河南安阳人,安阳师范学院教授,研究方向为软件技术与课程教育论,E-mail:cjlrzp@sina.com.

毛的摆动不断地选择区域；进入区域搜寻食物；并且在消耗一定食物之后，决定是否迁移到另一个区域中；然后通过对大肠杆菌在觅食过程中的行为进行建模，并通过迭代求解的一种智能仿生算法. 该算法主要是通过趋向性操作、复制操作、迁移操作进行建模、迭代优化、求解的，下面主要介绍 3 种操作，然后给出算法的一般步骤.

1.1　趋向性操作

趋向性操作有两种基本运动：游动和翻转. 细菌向任意一个方向游动一个单位的游动步长为翻转；如果细菌在完成一次翻转后的适应度有所提高，则沿同一方向继续游动若干步，直至其适应度不再有所改善，或已达到最大游动步数 N_c，这一过程为游动. 趋向性操作可以确保细菌种群的局部搜索能力，提高算法的收敛速度和计算精度.

1.2　复制操作

经过一个周期的趋向性操作，细菌开始复制操作，这个操作模拟了细菌种群优胜劣汰的繁殖过程. 假定细菌种群的规模大小为 S，淘汰觅食能力较差的 $S/2$ 细菌，剩下觅食能力较强的 $S/2$ 个细菌进行自我复制，从而保证细菌种群的规模.

1.3　迁移操作

细菌种群在复制操作结束后，开始迁移操作. 迁移操作可以描述为：给定一个迁徙概率 P_{e_d}，对某个个体产生一个[0, 1]的随机数 r，若 $r<P_{e_d}$，则该个体消亡，并随机产生一个新个体取代当前个体；否则，保持当前个体不变，转向下一个个体，直到遍历完种群中所有的个体为止. 迁移操作可以保证算法的全局寻优和跳出局部极值点的能力.

1.4　算法的一般步骤

细菌觅食优化算法求解问题的一般步骤为：

步骤 1　对问题编码.

步骤 2　初始化参数.

步骤 3　设计适应度函数.

步骤 4　产生初始种群.

步骤 5　对细菌种群分别进行趋向性操作，复制操作和迁移操作.

2　云模型

定义 1[17]　设 U 是一个定量论域，C 是 U 上的定性概念，如果定量值 $x \in U$，且 x 是定性概念 C 的一次随机实现，x 对 C 的确定度是 $\mu(x) \in [0,1]$ 是具有稳定倾向的随机数

$$\mu: U \to [0,1] \quad \forall x \in U \quad x \to \mu(x),$$

则 x 在论域区间 U 上的分布称为云，每一个 x 称为云滴.

云模型用 3 个数字特征表示概念：期望 E_x(Expected value)、熵 E_n(Entropy) 和超熵 H_e(Hyper entropy)，记作：$C(E_x, E_n, H_e)$[17].

期望 E_x：表示云滴在论域区间分布中的期望值，能够代表定性概念的量值，或者说是这个概念量化的最典型样本.

熵 E_n：表示定性概念的不确定性的度量. 一方面，熵 E_n 是定性概念亦此亦彼性的度量，反映了在论域区间能够被概念接受的云滴的取值范围；另一方面，熵 E_n 也是定性概念随机性的度量，反映了代表定性概念的云滴的离散程度.

超熵 H_e：表示熵 E_n 的不确定性的度量，是熵的熵. 它由熵的随机性和模糊性共同决定.

2.1　正向正态云发生器

定义 2[17]　假定 U 是一个定量论域，C 是 U 上的定性概念，如果定量值 $x \in U$，且 x 是定性概念 C 的一次随机实现，如果 x 满足：$x \sim N(E_x, E_n'^2)$，其中 $E_n' \sim N(E_n, H_e^2)$，且 x 对 C 的确定度满足：

$$\mu = e^{-\frac{(x-E_x)^2}{2(E_n')^2}},$$

则 x 在论域区间 U 上的分布称为正态云.

正向正态云发生器是定性概念到定量数值的一种映射,它根据云的数字特征 $C(E_x, E_n, H_e)$ 生成云滴.

正向正态云发生器的具体算法为:

输入 数字特征 $C(E_x, E_n, H_e)$,生成云滴的个数 n.

输出 n 个云滴 x 及其确定度 μ(也可表示为 $d(x_i,\mu_i)$, $i=1,2,\cdots,n$).

步骤 1 以 E_n 为期望值,H_e^2 为方差,生成一个正态随机数 $E_{n_i}^{'} = N(E_n, H_e^2)$.

步骤 2 以 E_x 为期望值,$E_{n_i}^{'2}$ 为方差,生成一个正态随机数 $x_i^{'} = N(E_x, E_{n_i}^{'2})$.

步骤 3 计算 $\mu_i = e^{-\frac{(x_i-E_x)^2}{2(E_{n_i}^{'})^2}}$.

步骤 4 具有确定度 μ_i 的 x_i 成为数域中的一个云滴.

步骤 5 重复步骤 1 到 4,直至生成 n 个云滴.

2.2 X 条件云发生器[17]

X 条件云发生器属于正向正态云发生器,在 X 条件云发生器中,云滴 x 是确定的,而确定度 μ 是随机的.

X 条件云发生器的具体算法为:

输入 数字特征 $C(E_x, E_n, H_e)$,论域空间的定量 x_0,生成云滴的个数 n.

输出 n 个确定度 μ (也可表示为 $d(x_i,\mu_i)$, $i=1,2,\cdots,n$).

步骤 1 以 E_n 为期望值,H_e^2 为方差,生成一个正态随机数 $E_{n_i}^{'} = N(E_n, H_e^2)$.

步骤 2 计算 $\mu_i = e^{-\frac{(x_0-E_x)^2}{2(E_{n_i}^{'})^2}}$.

步骤 3 具有确定度的 x_0 成为数域中的一个云滴.

步骤 4 重复步骤 1 到 3,直至生成 n 个云滴.

对于正向正态云,一方面通过对熵 E_n 和超熵 H_e 的调整,调整正向正态云生成的云的形状,使它具有随机性;另一方面,正向正态云具有正态分布的趋势性,云滴呈现两头小,中间大的特点,较好地保持了细菌在觅食过程中的趋势性,因此,正向正态云具有良好的数学建模能力,可以应用于具体实际问题中.

3 基于云模型的细菌觅食优化算法

在 BFOA 的算法中,趋向性操作是 BFOA 的核心操作,决定算法的局部搜索能力.在标准 BFOA 中一般采用固定步长,无法体现出不同细菌的差异以及在自然环境下觅食的随机性,使得收敛速度较慢.另外,在迁移操作中,对所有细菌采用相同的迁移概率 P_{e_d},造成部分精英个体消亡,降低了细菌种群多样性和全局寻优的能力,在一定程度造成了解的退化.

为了克服上述不足,在云模型的基础上,因此本文提出一种新的基于云模型的细菌觅食优化算法,针对趋向性操作,给出细菌灵敏度的概念,采用 X 条件云发生器调整细菌灵敏度,控制游动步长;引入正向正态云发生器修正迁移概率,改进迁移操作.满足算法的终止条件时,结束算法.

算法的步骤:

步骤 1 编码.问题空间向解空间的映射称为编码,编码的目的是为了 BFOA 中 3 个操作的实现以及对问题进行建模求解.一般使用的编码方案为:二进制编码,整数编码[18].

步骤 2 初始化参数.BFOA 的参数有:S, N_c, N_s, N_{r_e}, N_{e_d}, P_{e_d}, j, k, l.其中,S 为种群规模大小,N_c 为趋向性操作执行次数,N_s 为游动次数,N_{r_e} 为复制操作执行次数,N_{e_d} 为迁移操作执行次数,P_{e_d} 为迁移概率,j 为趋向性操作计数参数,k 为复制操作计数参数,l 为迁移操作计数参数.

步骤 3 设计适应度函数.适应度函数反映细菌获取食物和避开有毒物质的能力,适应度函数值越大表示细菌个体觅食能力越好.通常,适应度函是由目标函数变换得到,常用的变换方法有线性变换法,幂函数变换法,指数变换法.

步骤 4 生成初始种群.一般通过随机方法产生初始种群.

步骤 5 迁移循环:$l = l+1$.

步骤 6　复制循环：$k = k + 1$.

步骤 7　趋向性循环：$j = j + 1$，进行趋向性操作. 文献[19]中采用线性递减的方法调整细菌灵敏度，本文在此基础上，采用 X 条件云发生器对细菌灵敏度自适应递减. 下面给出趋向性操作的游动和翻转运动的描述.

(1) 灵敏度初值

$$V_0 = \frac{f_i}{f_{\max}}(X_{\max} - X_{\min}) \times r\ , \tag{1}$$

其中，V_0 为细菌灵敏度初值，$X_{\max}$，$X_{\min}$ 为变量的边界，f_i 为细菌 i 的适应度，$f_{\max}$ 为种群中适应度最大值，r 为 [0, 1] 的随机数.

(2) 翻转. 细菌根据(2)式进行翻转，

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\Phi(j), \tag{2}$$

其中，$\theta^i(j,k,l)$ 表示第 i 个细菌在进行第 j 次趋向性操作，第 k 次复制操作以及第 l 次迁移操作后的位置，$C(i)$ 表示向前的游动步长，$\Phi(j)$ 表示翻转后选择的随机方向.

(3) 游动. 细菌灵敏度改进如下：

$$E_x = S_a\ , \tag{3}$$

$$E_n = (S_{\max} - S_{\min})/m_1\ ,m_1 \text{ 表示控制参数}, \tag{4}$$

$$H_e = E_n/n_1\ ,n_1 \text{表示控制参数}, \tag{5}$$

$$E_n' = N(E_n, H_e^2), \tag{6}$$

$$V = \begin{cases} e^{-\frac{(S_i - E_x)^2}{2(E_n')^2}}, & S_i > S_a, \\ V_0, & S_i \le S_a, \end{cases}\ , \tag{7}$$

其中 S_a 表示游动次数的平均值，S_i 表示当前细菌游动的次数，V 为游动步长.

一般地，在细菌觅食寻优开始时，种群中大部分细菌个体与全局最优点的距离较远，游动步长 C 应较大，使得细菌种群快速向目标区域移动，以此增加算法的全局搜索能力；但是随着觅食过程的进行，越来越多的细菌个体接近全局最优点，游动步长 C 应较小，避免陷入局部极值点，以此增加算法的局部搜索能力.

根据对游动步长 C 的分析，本文利用 X 条件云发生器调整细菌灵敏度，使细菌灵敏度以整个游动步长为搜索范围，以正向正态云的形状进行递减，从而控制游动步长，较好地满足了对游动步长的要求. 改进后的游动步长不仅可以保证细菌种群的趋势性，而且可以较好地反应细菌种群在生存和自然进化过程中的不确定性.

步骤 8　若 $j < N_c$，则转向步骤 7.

步骤 9　复制操作. 保存适应度高的细菌个体并进行自我复制.

步骤 10　若 $k < N_{r_e}$，则转向步骤 6.

步骤 11　迁移操作. 所有细菌按照以下方式自适应调整 P_{e_d}.

$$E_x = f_a\ , \tag{8}$$

$$E_n = \frac{(f_{\max} - f_{\min})}{m_2}\ ,m_2 \text{ 表示控制参数}\ , \tag{9}$$

$$H_e = \frac{E_n}{n_2}\ ,n_2 \text{ 表示控制参数}, \tag{10}$$

$$E_n' = N(E_n, H_e)\ , \tag{11}$$

$$P_{e_d} = \begin{cases} t_1 e^{-\frac{(f_a - E_x)^2}{2(E_n')^2}}, f_a \ge f_{a_v}, \\ t_2, f_a < f_{a_v}, \end{cases} \tag{12}$$

其中 f_a 为迁移个体的适应度，$f_{\max}$，$f_{\min}$ 和 f_{a_v} 分别为当前种群中适应度的最大值、最小值和平均值. 引入正向正态云发生器，以随机选择的迁移个体的适应度为期望，以最大和最小适应度的差为基数作为搜索范围更新迁移概率 P_{e_d}.

改进的迁移概率 P_{e_d} 可以保护高于平均适应度的较好的细菌个体,生成较多集中的云滴;对那些低于平均适应度的细菌个体,生成较少离散的云滴.因此,迁移概率 P_{e_d} 即具有趋势性,满足快速寻优的能力,又具有随机性,较好地满足了全局搜索的能力.

迁移操作需要随机生成新个体时,按照以下方式随机生成新个体

$$E_x = X_a \tag{13}$$

$$E_n = Q/m_3, Q = t_3 \times (X_{f_{\max}} - X_{f_{\min}}), m_3, t_3 \text{ 表示控制参数,} \tag{14}$$

$$H_e = \frac{E_n}{n_3}, n_3 \text{ 表示控制参数,} \tag{15}$$

$$E_n' = N(E_n, H_e), \tag{16}$$

$$X_a' = N(E_x, E_n'^2), \tag{17}$$

$$\mu = e^{-\frac{(X_a' - E_x)^2}{2(E_n')^2}}, \tag{18}$$

若 $\mu < P_{e_d}$,则更新个体.其中,X_a 为迁移前的个体,X_a' 为迁移后的新个体,$X_{f_{\max}}$,$X_{f_{\min}}$ 为适应度最大、最小个体.采用正向正态云发生器改进迁移操作,以适应度最高的个体和适应度最低的个体的差值为搜索范围,有效地提高算法性能,较好地反映了自然界中细菌生存进化的随机性.

步骤 12 若 $l < N_{e_d}$,则转向步骤 5;否则,输出结果,算法结束.

算法的流程图如图 1 所示.

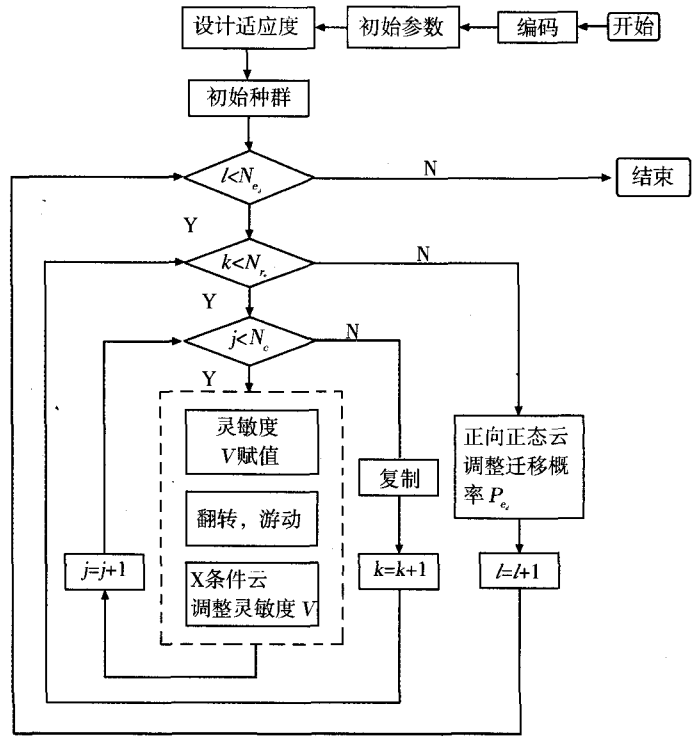

图 1 基于云模型的细菌觅食优化算法流程图

4 实验及分析

将基于云模型的细菌觅食优化算法应用于自动组卷系统中,与遗传算法进行实验比较分析,验证算法的有效性.

本文算法实验环境为 Microsoft Visual Studio 2013,语言为 C#,试题库为 SQL SERVER 2008,建立 1000 道题的试题库,其中单选题、多选题,填空题,判断题以及问答题各 200 道,每种题型的试题单独在一张

表内.

4.1 试卷的属性

为了更好地考核学生对知识的掌握程度，将试卷中的每道试题选择分为 6 个核心属性.

(1) 试题编号：表示试题在试题库中的唯一标识.

(2) 题型编号：表示题目所属的题目类型，本文指定 5 种题型，题型编号 1～5 分别为：1 表示单选题，2 表示多选题，3 表示判断题，4 表示填空题，5 表示问答题.

(3) 试题难度系数：是指参加测试的学生在该题上的失分率. 在自动组卷开始之前，试题的难度系数由用户依据经验设置.

(4) 试卷区分度：是指对于不同层次的学生，区分其水平能力的指标.

(5) 考试时间：表示考生完成试卷的时间，一般的在线考试系统采用倒计时的方法来实现.

(6) 题分：表示试卷中每题的分值.

4.2 试卷的数学模型

在试题核心属性的基础上，构建自动组卷系统的数学模型. 在本文提出的算法中，将一份试卷映射为一个细菌个体. 假定每份试卷中有 m 道题，则决定一道试题就要决定上述的 6 个约束变量(题号 a_1，题型 a_2，难度 a_3，区分度 a_4，时间 a_5，题分 a_6)，a_i 表示第 i 个约束变量. 这 6 个约束变量可以用一个向量(a_1，a_2，a_3，a_4，a_5，a_6) 来表示，称为属性向量. m 个属性向量作为行，构成一个 $m\times 6$ 的目标矩阵 A

$$A=\begin{bmatrix} a_{11},a_{12} & \cdots & a_{16} \\ a_{21},a_{22} & \cdots & a_{26} \\ \vdots\ \vdots & & \vdots \\ a_{m1},a_{m2} & \cdots & a_{m6} \end{bmatrix}$$

其中，a_{ij} 表示第 i 道试题的第 j 个属性. 决定一份试卷就需要决定目标矩阵 A，且 A 满足以下约束条件.

(1) 试卷的总分 T_S，$T_S=\sum_{i=1}^{m}a_{i6}$.

(2) 试卷的题量 N，$N=\sum_{K=1}^{5}Q_k$，$Q_k=\sum_{i=1}^{m}C_{2i}$，其中 $C_{2i}=\begin{cases}1, a_{i2}=k\\ 0, a_{i2}\neq k\end{cases}$，$k$ 为题型编号，Q_k 表示第 k 种题型的试题量.

(3) 试卷的难度 P，各个难度等级所占的分值比例为：$P=\sum_{i=1}^{m}C_{3i}a_{i3}/T_S$，其中 $C_{3i}=\begin{cases}1 & ,a_{i3}=k\\ 0 & ,a_{i3}\neq k\end{cases}$，$k$ 表示难度等级号. 根据难度 P 的不同，可以将试题分为易、较易、中等、较难和难 5 个难度等级，分别用[0，0.2]、(0.2，0.4]、(0.4，0.6]、(0.6，0.8] 和(0.8，1] 表示.

(4) 试卷的区分度 D，各个区分度等级所占的分值比例为：$D=\sum_{i=1}^{m}C_{4i}a_{i4}/T_S$，其中 $C_{4i}=\begin{cases}1 & ,a_{i4}=k\\ 0 & ,a_{i4}\neq k\end{cases}$，$k$ 表示区分度等级号. 根据区分度 D 的不同，可以将试题分为优良、合格、修改、淘汰 4 个区分度等级，分别用[0，0.2]、(0.2，0.3]、(0.3，0.4]、(0.4，1] 表示.

(5) 试卷的时间 T，$T=\sum_{i=1}^{m}a_{i5}$.

4.3 目标函数的确定

在实际组卷中，用户提供期望的试卷总分、试题量、难度、区分度以及时间. 假定 e_i 表示各个约束条件实际值与期望值的误差，F 表示目标函数. 因为各个约束条件的重要程度不同，所以 F 为各个约束条件的误差 e_i 的加权和. 同时，指定 e_i 取其绝对值，以免各个约束条件之间的误差相互抵消. 根据试卷数学模型，确定了目标矩阵 A 有 5 个约束条件，则目标函数为 $F=\sum_{i=1}^{5}e_i\omega_i$. 表示目标矩阵 A 的第 i 个约束条件的权重，由专家根据约束条件重要程度的不同给出相应的权值且 $\sum_{i=1}^{5}w_i=1$. 目标函数 F 的值越小表示实际生成的试卷与用

户期望的试卷之间误差越小,生成的实际试卷越符合用户的要求.

4.4 自动组卷系统中算法的具体步骤

步骤 1 编码. 本文采用整数编码,将一份试卷映射为一个细菌个体,将试题编号作为细菌个体基因,按照不同题型分段编码. 表 1 为第 i 份试卷,即第 i 个细菌个体.

表 1 第 i 个细菌个体

11	23	56	39	…	06	58	26	65	…	30	15	28	07	…	23	16	42	31	…
单选题					多选题					填空题					判断题				

步骤 2 初始化参数. 控制参数 $m_1 = m_2 = m_3 = 3S, n_1 = n_2 = n_3 = 6$, $S = 100, N_c = 40, N_s = 4$, $N_{r_e} = 4$, $N_{e_d} = 16, t_2 = 0.3, j = 0, k = 0, l = 0, T_S = 100$ 分, $T = 120$ 分钟, $Q_1 = 10$ 道,每题 1 分, $Q_2 = 10$ 道,每题 2 分, $Q_3 = 5$ 道,每题 2 分, $Q_4 = 10$ 道,每题 1 分, $Q_5 = 5$ 道,每题 10 分.

步骤 3 设计适应度函数. 采用指数型目标函数,则适应度函数为: $f = e^{-\beta F}$, β 取 0.03.

步骤 4 初始种群的产生. 初始种群满足 T_S, N 和 $T3$ 个约束条件.

步骤 5 迁移循环: $l = l + 1$.

步骤 6 复制循环: $k = k + 1$.

步骤 7 趋向性循环: $j = j + 1$,进行趋向性操作.

(1) 灵敏度初值. 计算 $f_i, f_{\max}$,根据公式(1) 求得 V_0.

(2) 翻转. 对细菌 i,随机产生题型编号 Q_1,确定题型区间. ($1 \leq Q_2 \leq 5$). 对第 Q_1 种题型,随机产生试题数量 Q_2,确定区间 $[Q_2, Q_k]$. ($1 \leq Q_2 \leq Q_k$, Q_k 表示第 K 种题型的题量),从相应的题型的试题库中随机挑选 $|Q_k - Q_2|$ 道试题替换,得到新个体.

(3) 游动. 若新个体的适应度 f 较大,则沿该方向继续游动. 比较 S_i 与 S_a,根据公式(3) ~ (7) 计算游动步长 C.

步骤 8 若 $j < N_c$,则转向步骤 7.

步骤 9 复制操作. 所有细菌根据适应度 f 降序排列,淘汰后 $S/2$ 个细菌个体,复制前 $S/2$ 个细菌个体.

步骤 10 若 $k < N_{r_e}$,则转向步骤 6.

步骤 11 迁移操作. 比较 f_a 与 f_{a_v},根据公式(8) ~ (12) 计算迁移概率 P_{e_d}. 若需随机生成新个体,则根据公式(13) ~ (17) 更新个体,否则,转向步骤

步骤 12 若 $l < N_{e_d}$,则转向步骤 5;否则,输出结果,算法结束.

4.5 实验结果及分析

将本文算法应用于自动组卷系统中,与遗传算法进行实验比较分析. 使用本文算法与遗传算法分别进行 20 次的独立的组卷测试并对其收敛精度、收敛时间和组卷成功率依次进行比较分析,结果如表 2、表 3 和表 4 所示.

表 2 本文算法与遗传算法的收敛精度

实验次数	本文算法			遗传算法		
	最大值	平均值	标准差	最大值	平均值	标准差
1	9.5342E-01	9.5326E-01	2.0857E-04	9.4819E-01	9.4755E-01	7.6285E-04
2	9.5339E-01	9.5323E-01	2.0244E-04	9.4824E-01	9.4760E-01	7.9235E-04
3	9.5333E-01	9.5317E-01	1.8963E-04	9.4817E-01	9.4753E-01	7.0351E-04

表 3 本文算法与遗传算法的进化代数平均时间

实验次数	本文算法		遗传算法	
	进化代数	t/s	进化代数	t/s
1	29	40	38	81
2	31	41	34	79
3	28	38	33	80

表 4　本文算法与遗传算法的成功次数和成功率

实验次数	本文算法		遗传算法	
	成功次数	成功率	成功次数	成功率
1	19	0.95	10	0.5
2	20	1	12	0.6
3	20	1	10	0.5

由表 2 和表 3 可知，本文算法的收敛精度均优于遗传算法，同时本文算法可以较快收敛至最优解. 由表 4 可以看出，相对于遗传算法，本文算法的组卷成功率较高，其中，成功率＝组卷成功的次数/实验总次数. 这是因为新算法采用整数编码，无需解码，并且初始种群已经满足试卷总分、试卷题量、考试时间 3 个约束条件，所以，该算法有效地减少了组卷的冗余度，提高了算法的收敛速度. 同时，采用云模型改进后的游动步长和迁移概率提高算法的全局寻优能力和局部寻优能力，避免出现陷入局部极值点的问题. 图 2 为新算法与遗传算法收敛曲线的对比图，其中 GA 为遗传算法，CBFOA 为基于云模型的细菌觅食优化算法.

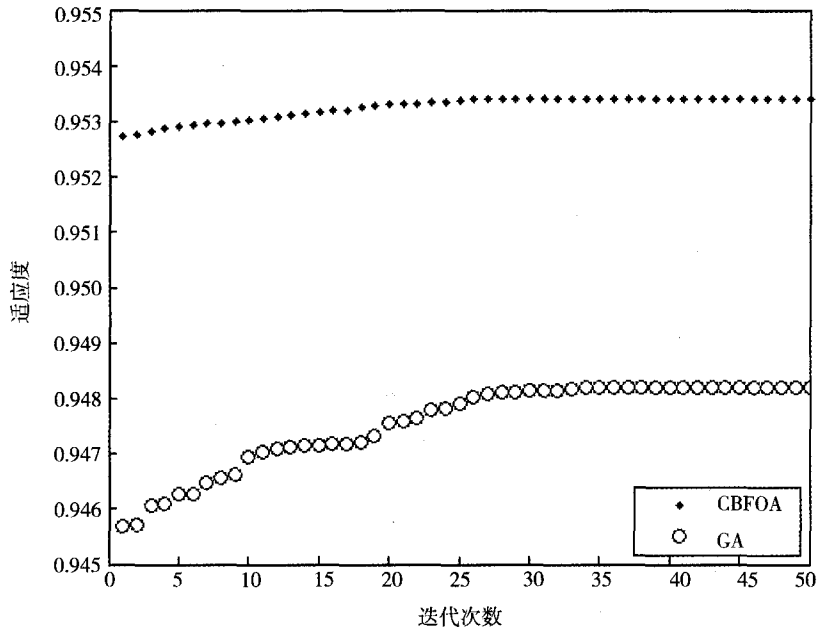

图2 实验结果对比图

由图 2 可知，相比遗传算法，新算法的适应度均大于遗传算法，说明新算法中的细菌个体的适应能力较强. 新算法采用 X 条件云发生器调整游动步长后，较快地收敛于全局最优解 0.009 540. 新算法中的适应度曲线变化比较平缓，这是因为在引入正向正态云发生器修正迁移概率，进行迁移操作后，有效地提高适应度较低个体的搜索能力，产生了更多新个体，增强了算法的随机性. 实验数据表明，本文提出的新算法在收敛速度和优化精度方面均优于遗传算法.

5　结　论

本文提出了一种基于云模型的细菌觅食优化算法. 首先给予细菌灵敏度的概念，结合正态云模型的随机性和稳定倾向性的特点，采用 X 条件云发生器调整细菌灵敏度，从而控制游动步长，进行趋向性操作和复制操作，提高了算法的收敛速度. 其次引入正向正态云发生器改进迁移概率，进行迁移操作，较快地搜索到全局最优解. 最后将改进后的算法应用于自动组卷系统中，通过与遗传算法的实验比较分析，证明了该算法的有效性. 还需要深入研究的：考虑其他操作对算法的影响，进一步改进基于云模型的细菌觅食优化算法；随着试题数量和自动组卷系统约束条件的增加，对算法的收敛情况和算法参数的设计需要深入研究.

参 考 文 献

[1] 李 珊,党建武,卜 锋.细菌觅食优化算法的研究与改进[J].计算机仿真,2013,30(4):344-347.

[2] 姜建国,周佳薇,郑迎春,等.一种自适应细菌觅食优化算法[J].西安电子科技大学学报,2015,42(1):75-81.

[3] 刘小龙,赵奎领.基于免疫算法的细菌觅食优化算法[J].计算机应用,2012,32(3):634-637.

[4] 梁艳春,吴春国,时小虎.群智能优化算法理论应用[M].北京:科学出版社,2009:157-159.

[5] 吴秀丽,张志强,杜彦华,等.改进细菌觅食优化算法求解柔性作业车间调度问题[J].计算机集成制造系统,2015,21(5):1262-1270.

[6] 周 逊,郭 敏,马 苗.细菌觅食优化算法优化归一化准则的彩色图像分割[J].西北大学学报(自然科学版),2015,45(1):40-44.

[7] 胡海波,黄友锐.混合粒子群算法优化分数阶 PID 控制参数研究[J].计算机应用,2009,29(9):2483-2486.

[8] 杨大炼,刘义伦,李学军,等.基于细菌觅食优化决策的齿轮箱故障诊断[J].中南大学学报(自然科学版),2015,46(4):1224-1230.

[9] 李德毅,孟海军,史雪梅.隶属云和隶属云发生器[J].计算机研究与发展,1995,32(6):15-20.

[10] 李德毅,刘常昱.论正态云模型的普适性[J].中国工程科学,2004,6(8):28-34.

[11] 张光卫,何 锐,刘 禹,等.基于云模型的进化算法[J].计算机学报,2008,31(7):1082-1091.

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

[13] Jain A K, Srivastava S C, Singh S N, et al, Bacteria Foraging Optimization Based Bidding Strategy Under Transmission Congestion[J]. IEEE Systems Journal,2015,9(1):141-151.

[14] Sathya P D, Kayalvizhi R. Optimal segmentation of brain MRI based on adaptive bacterial foraging algorithm[J]. Neurocomputing, 2011,74(14/15):2299-2313.

[15] 李丽娟.改进细菌觅食算法求解流水线调度问题[D].成都:西南交通大学,2014.

[16] 杨大炼,李学军,蒋玲莉.一种细菌觅食算法的改进及其应用[J].计算机工程与应用,2012,48(13):31-34.

[17] 苗夺谦,李德毅.不确定性与粒计算[M].北京:科学出版社,2011:4-7.

[18] 乌兰图雅,李瑞俊.基于量化模型的自动组卷算法研究[J].内蒙古大学学报(自然科学版),2014,45(2):202-207.

[19] 李 莹,简献忠,陈 青.基于免疫进化细菌觅食优化算法的无功优化[J].上海理工大学学报,2014,36(3):245-249.

The Bacteria Foraging Optimization Algorithm Based on the Cloud Model

CUI Jinling[1], WU Di[2]

(1. College of Computer and Information Technology, Anyang Normal University, Anyang 455000, China;
2. College of Computer and Information Technology, Henan Normal University, Xinxiang 453007, China)

Abstract: A new bacteria foraging optimization algorithm based on the cloud model is presented for solving the problems of slow convergence rate, partial optimum and premature convergence. Firstly, in the operation of chemotaxis and reproduction, the conception of sensitivity is given and adjusted by the *X*-conditional cloud generator for controlling swim steps, combined with the characters of randomness and stability of the cloud model. The convergence rate is improved by this method. Then, in the operation of elimination and dispersal, the adaptive and non-linear probability of elimination and dispersal is adopted by the forward normal cloud generator, which improves the global-optimization capability. Finally, this algorithm is used to the system of automatic test, compared and analyzed with the experiment of Genetic Algorithm. The results of experiment show that this algorithm is better than Genetic Algorithm both in convergence rate and quality of optimization.

Keywords: bacteria foraging optimization algorithm; cloud model; automatic test