改进细菌觅食算法在高维优化问题中的应用

李 珺 党建武
(兰州交通大学电子与信息工程学院 兰州 730070)

摘 要 针对以往细菌觅食优化算法自适应步长公式经验性参数过多、无法真正实现自适应的缺点,提出了改进的步长公式,使步长仅与细菌个体当前的进化代数和所求解问题的寻优范围有关,真正实现步长的自适应;其次,将混沌思想和差分进化思想与细菌觅食算法结合,对算法初始化过程和寻优过程进行改进,增加群体多样性,避免算法因为早熟而陷入局部最优值;在高维问题的优化过程中,采用逐维更新细菌位置的方法,将整体问题分维处理,极大地提高了算法效率和精度。通过对多个标准测试函数在多维空间进行测试,表明改进算法在高维空间中寻优时速度快、精度高、求解过程简单可行,在寻得最优解的精度上比其他改进方案有显著提高。

关键词 细菌觅食优化算法,自适应步长,混沌理论,差分进化,高维优化
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.056

Application of Bacteria Foraging Algorithm in High-dimensional Optimization Problems

LI Jun DANG Jian-wu
(School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract Excessive empirical parameters in the former self-adaptation step formula caused the defect in terms of failing to achieve self-adaptation in bacteria foraging optimization algorithm. Therefore, a revised step formula has been proposed, which enables step length to be relevant to the present evolution generation of individual bacteria as well as the optimal range of the problem to be solved, in order to achieve the step length self-adaption. Besides, the combination of chaotic thoughts and differential evolution thoughts with bacteria foraging algorithm can improve both the initial process and optimal process of the algorithm. This method increased the diversity of groups, preventing the algorithm from falling into the local optima due to the precocious. In the optimal process of high-dimensional problem, fractal dimension optimization is used to replace the former method. The fractal dimension optimization means that the information of every dimension will be updated one by one on the basis of whether the new position of every dimension changes comparing to the fitness value. Dealing with the problems in different dimensions can boost the precision and efficiency of the algorithm obviously. Experiments show that through the testing of multiple standard test functions in the hyperspace, the revised algorithm optimizing in the hyperspace has several benefits, such as fast speed, high precision and the simple process of solving. It has improved manifestly in terms of precision when comparing to other modified programs.

Keywords Bacteria foraging optimization algorithm (BFO), Adaptive step size, Chaos theory, Differential evolution (DE), High-dimensional optimization

1 引言

细菌觅食优化算法[1](Bacteria Foraging Optimization, BFO)是Passino于2002年提出的一种模拟大肠杆菌觅食行为的全局优化进化算法。算法从一组随机初始化的细菌位置开始,用每个细菌的位置来表示一个可能的解,通过三大操作(趋向性操作、复制操作和迁徙操作)以及群体感应机制来模拟细菌在觅食过程中位置的改变,从而找到问题的最优解。该算法一经提出,就引起了国内外学者的广泛关注。有研究显示,该算法在寻优过程中收敛速度快、精度高[2-3],且能有效地跳出局部最优解,已成为群体优化算法中的一个研究热点。

为了进一步提高算法的效率,国内外学者不断地对其进行研究与改进。标准BFO算法在趋向性操作过程中虽然提出了可以自适应地调整步长,却并未给出具体的步长调整方案,因此很多学者给出了自适应步长公式。陈建超等在步长中引入基于距离的聚类思想,得到了一种自适应变步长菌群

到稿日期:2016-03-03 返修日期:2016-07-01 本文受甘肃省科技计划项目:细菌觅食优化算法在多目标优化中的应用研究(1506RJZA084),甘肃省教育厅科研项目:菌群优化算法的融合、改进及应用(1204-13),甘肃省教育科学'十二五'规划课题:细菌觅食优化算法在高维优化问题中的应用(GS[2015]GHB0907),兰州市科技计划项目:细菌觅食优化算法在组合优化中的应用研究(2015-2-74)资助。

李 珺(1974—),女,博士生,副教授,主要研究方向为智能计算与智能信息处理、图形图像处理,E-mail:lijane@mail.lzjtu.cn;党建武(1962—),男,博士,教授,博士生导师,主要研究方向为人工智能、神经网络。

优化算法[4]；Das 等人提出了随适应值变化的趋化步长策略[5]；Dasgupta 将细菌当前的适应度引入步长，让当前的步长随着当前适应度值的好坏而变化[6]；许鑫在 Dasgupta 研究的基础上对步长进行了进一步的改进，提出了基于场的自适应细菌优化算法[7]。还有大量的研究集中在算法的融合、改进及应用上。Verma 等人将其应用在图像处理中，并取得了较好的效果[8]；章国勇等人将量子理论同菌群算法结合，提出了一种具有量子行为的细菌觅食优化算法[9]；刘小龙等人将菌群算法与免疫理论结合，提出了基于免疫算法的细菌觅食优化算法[10]；Saber 等人将算法应用于电力工程控制中，效果显著[11]。

以上改进方案对细菌觅食算法的寻优结果有了一定的改善，但改进效果有限。首先，自适应步长的各种改进方案都不同程度地引入了更多的参数，这些以经验确定的参数值在针对不同问题求解时并不能真正地实现步长的自适应；其次，标准 BFO 中的复制策略的目的在于提高算法的收敛速度，但同时也降低了群体多样性，使算法更容易陷入局部最优值；最后，在求解多维特别是高维问题时，标准趋向性操作中的位置更新方式忽略了各维度值的新位置对适应度的不同影响，有时新位置的适应度值虽未得到改善，但其中的某些维度信息其实已经得到了改进，而这些改进信息却未得到利用，降低了算法的寻优效率。

针对以上分析，结合混沌与差分思想，面向高维优化问题，提出改进的 BFO 算法。首先，重新定义自适应步长公式，大大减少了公式中的参数，使步长的设定仅与细菌个体的当前情况有关，提高了步长的自适应性；其次，结合混沌思想，在菌群个体初始化时，利用混沌变量引导优化变量的设置，尽可能使菌群个体在寻优空间中均匀分布，提高了全局最优解的搜索效率；然后，结合差分思想，对位置更新方式进行改进，以保证群体内个体的多样性；最后，在高维优化问题求解中，考虑每一次位置更新后各维度对适应度值的不同影响，充分利用各维度带来的有利信息，大大提高了算法逃脱局部最优值的能力，加快了算法寻找到全局最优解的效率。

2 细菌觅食优化算法的改进

2.1 自适应步长

BFO 算法中，步长值的选定对算法有着至关重要的影响[12]。步长值选择较大，在寻优的初始阶段，细菌个体会大步向最优值迈进，收敛速度快，然而当个体接近最优位置时，往往由于较大的步长使得细菌个体一步跨过最优位置，在最优值左右徘徊而无法达到最优位置，导致算法精度不足；步长值选择较小，细菌个体向最优值迈进的速度也较小，使得求解最优值的速度过慢，算法效率不高。

在已有的自适应步长的改进中，往往引入了大量的参数，而参数的确定凭借的是经验值，统一的参数无法达到真正的自适应步长。

为了使步长真正地达到自适应要求，在改进的步长机制中首先考虑了细菌当前的寻优范围大小，使步长的大小不受寻优范围的影响，自适应地随寻优空间的大小而调整；另外，考虑到随着进化代数的增加，细菌个体逐渐接近最优位置，步长应逐渐减小。改进的自适应步长公式如式(1)所示：

$$C(d)=\frac{MAX_d-MIN_d}{2\times generation} \tag{1}$$

其中，$C(d)$为细菌个体第 d 维的步长，$generation$ 为目前细菌的代数，$[MIN_d,MAX_d]$为第 d 维的寻优范围。此时，步长 $C(d)$的值会根据进化代数和寻优范围动态调整，实现了在寻优初期，细菌个体以较大的步伐充分完成全局搜索；在算法进行到一定程度，全局最优解的所属区域已定时，再采用较小的步长加强局部开采。这种动态步长机制并未引入经验性参数，步长的确定仅与被求解问题本身的范围及算法进行的程度有关，自适应步长容易求得，既保证了算法的全局寻优能力，又保证了寻得最优解的精度。

2.2 混沌思想初始化细菌位置

混沌理论(Chaos Theory)是系统中确定性与随机性两种成分都明显存在时所合成的一种运动体制[13]。混沌具有遍历性的特征，即混沌运动在其混沌吸引域内是各态历经的，在有限时间内混沌轨道经过混沌区内的每一个状态点。利用混沌思想进行优化求解，将求解过程对应为混沌空间中的遍历过程，尽可能地避免了搜索过程陷入局部极值。

利用混沌的遍历性特征，在菌群算法初始化时，使用混沌变量来指导菌群个体初始位置的确定，使菌群个体在寻优空间中分布更加均匀，增加种群的多样性，有效提高了算法的全局搜索能力。

这里采用立方映射[14]生成混沌变量，迭代方程如式(2)所示：

$$x(i+1)=4x(i)^3-3x(i)$$
$$-1<x(i)<1,i=0,1,2,\cdots,S-1 \tag{2}$$

立方映射处于完全混沌状态，并遍历整个[0,1]区间。$x(0)$初始化为[0,1]区间中的一个随机数，S 为菌群的规模，根据菌群的大小，产生 S 个混沌变量。将产生的混沌变量转化为寻优空间中的优化变量，采用式(3)所示的线性映射：

$$\theta_d^i(0,0,0)=MIN_d+(MAX_d-MIN_d)\times x(i) \tag{3}$$

根据式(3)可求出第 i 个细菌个体在第 d 维的初始位置 $\theta_d^i(0,0,0)$。在细菌个体初始化过程中，根据在[0,1]区间遍历的混沌变量，产生遍布整个优化区间的细菌个体。这种初始化方法，令细菌个体在寻优空间中尽可能地均匀分布，最大限度地提高寻得最优解的效率。

2.3 差分思想实现细菌位置更新

差分进化算法(Differential Evolution，DE)是由 Rainer Storn 和 Kenneth Price 为求解 Chebyshev 多项式提出来的[15]。该算法是模拟生物进化机制的一种仿生智能算法，通过变异、交叉、选择 3 个操作来求解最优解。

在 DE 算法中，差分策略用来实现个体变异，即随机地选取种群中两个不同的个体，将其向量差进行缩放之后与待变异个体的向量进行合成，如式(4)所示：

$$D_i(g+1)=X_{r1}(g)+F\times(X_{r2}(g)-X_{r3}(g)) \tag{4}$$

其中，F 为缩放因子，用来扩展搜索空间，其取值范围为[0.5，1]；X_{r1}，X_{r2}，X_{r3}是从群体中随机选取的 3 个不相同的个体；g 表示种群的代数；$r1$，$r2$，$r3$ 代表的是群体中个体的编号。

为了提升细菌觅食算法的全局搜索能力，增加种群多样

性，避免算法过早地陷入局部极值，在趋向性操作和复制操作中引入差分思想来完成相应的过程。

在趋向性操作中，将当前所有细菌的适应度进行排序，定义排序后前 40% 的个体为优秀，按照标准算法中的趋向性操作对位置进行更新，并采用改进的步长公式；剩下 60% 的个体则按照差分的方式更新位置，如式(5)所示：

$$\theta_d^i(j+1,k,l)=\theta_d^i(j,k,l)+(\theta_d^{r1}(j,k,l)-\theta_d^{r2}(j,k,l)) \tag{5}$$

其中，$\theta_d^i(j,k,l)$表示细菌 i 在第 j 次趋向性操作、第 k 次复制操作和第 l 次迁徙操作之后第 d 维的位置，那么第 $j+1$ 次趋向性操作后，此维的位置根据随机找到的两个细菌个体 $r1$ 和 $r2$ 以及它自身的当前位置并利用式(5)求得。

首先，根据适应度值排序后的优秀个体，采用标准趋向性操作进行位置更新，最大限度地对自身周围的环境进行探索；其次，针对适应度值较差的个体，采用差分方式进行位置更新，不受自身当前所处位置的限制，提高了细菌个体对未访问到的位置的探索概率和种群多样性；然后，寻优过程初始时，随机分布的细菌个体散布在寻优空间中，以差分方式更新个体的位置，这时得到的新位置广阔地分布在整个寻优空间中，最大程度地增加了种群的多样性，完成全局搜索过程；最后，随着算法的推进，越来越多的个体聚集在最优解周围，此时进行趋向性操作的个体以及随机选择完成差分计算的两个个体在最优解附近的概率逐渐增大，经过差分过程得到的新位置也逐渐集中在最优解周围，充分进行局部搜索。

在复制操作时，对适应度较差即排在后 50% 的细菌个体，并不是简单地用适应度好的一半个体来取代它，而是将这些细菌个体分成两部分，较好的一半依据式(5)采用差分方式更新位置，较差的一半则依次使用适应度值最优的细菌来取代它们的位置。

标准 BFO 算法中的复制策略，简单地使用适应度值好的一半细菌个体去取代适应度差的另一半细菌个体，并未使当前菌群中的最优位置得到改进；同时，这种复制方法以折半的方式减少了种群的多样性，通过牺牲群体的多样性来使算法得到快速的收敛，高维多峰函数的求解极易导致算法陷入局部最优值。

差分思想的引入增加了种群的多样性，降低了菌群算法寻优过程中陷入局部最优值的风险，很好地平衡了算法的全局搜索和局部寻优能力，提高了算法的效率。

3　高维空间寻优

3.1　改进的高维寻优方式

高维空间寻优一直是优化问题中的难点，随着问题维度的增加，寻找最优解的难度呈指数上升。虽然群体智能算法在优化问题中表现出了较强的优势，但面对高维问题，算法的求解速度和精度仍旧大大下降。如何提高群体智能算法在高维问题中的求解效率，是目前研究的热点及难点[16]。

仔细分析菌群优化算法中细菌个体在高维空间中的寻优过程不难发现，细菌个体在高维空间中的每一次移动，不一定使其所处环境整体得到提升，即个体的适应度值并未更优，但其新的位置信息中总是有部分维度信息已经得到改善。根据标准的 BFO 算法，在一次趋向性操作中，如果新位置的适应度值并未更优，则新位置被抛弃，细菌个体仍旧在原位置不动，因此新位置中那些改善的维度信息没有被利用。如果能充分利用每一维度位置的改善，就会大大提高算法的求解能力。

根据以上分析，针对高维问题的寻优过程，提出改进的分维寻优方式。在分维寻优过程中，不再将细菌个体找到的新位置作为整体来一次性更新细菌所有维度的位置信息，而是逐维更新位置信息。细菌个体搜索到一个新位置时，逐维判断每一维的新位置对整体适应度的影响，如果此维的新位置使得整体适应度值更优，则更新此维的位置信息，否则此维的位置信息不变，依次判断所有维度。

通过分维寻优，对问题进行精细化分割，充分利用局部维度的变化信息，针对复杂的高维优化问题，有效地提高了算法的效率和精度。

3.2　改进算法的流程

求解时，所有细菌的位置采用数组 $Bac[S][D]$保存，S 表示细菌的规模，D 表示整个寻优空间的维度。用 $C(D)$记录当前代趋向性操作中细菌个体在每一维度上的移动步长。存储结构如表 1 所列。

表 1　最优解的存储结构

细菌个体 \ 存储数组：每维当前位置	第 1 维位置	第 2 维位置	…	第 D 维位置
第 1 个细菌个体	$Bac[0][0]$	$Bac[0][1]$	…	$Bac[0][D-1]$
第 2 个细菌个体	$Bac[1][0]$	$Bac[1][1]$	…	$Bac[1][D-1]$
第 3 个细菌个体	$Bac[2][0]$	$Bac[2][1]$	…	$Bac[2][D-1]$
…				…
第 S 个细菌个体	$Bac[S-1][0]$	$Bac[S-1][1]$	…	$Bac[S-1][D-1]$
自适应步长	$C[0]$	$C[1]$	…	$C[D-1]$

使用类 C 语言对算法进行描述，其中，$newsite[r]$为细菌 i 前进一步后新位置 r 维的值；$adaptation(Bac[i])$ 函数用于计算细菌当前位置的适应度值；用 $fitness[S]$数组来保存细菌个体的适应度值；$fitness[i]=adaptation(Bac[i])$；$sort(Bac,fitness,S)$函数按照适应度值的优劣对菌群进行排序；$N_c$ 为趋向操作次数；N_s 为趋向操作中单向泳动的最大步数；$random(0,S)$生成[0,S]的随机数。

算法整体流程如下：

(1)初始化算法参数：$S=100$(种群规模)，D(寻优空间的维度)，$N_c=50$(趋向性操作次数)，$N_s=4$(同一个方向上泳动的最大步数)，$N_{re}=2$，P_{ed}(迁徙概率)，$d_{attractant}$(引力深度)，$w_{attractant}$(引力宽度)，$w_{repellant}$(斥力宽度)，$h_{repellant}$(斥力高度)，$[MIN_d,MAX_d]$(每个维度上的寻优范围)，进化代数 $generation=0$。

(2)初始化细菌个体位置

```
for (j=1;j<=D;j++)
{
  x0=random(0,1);
  依据式(2)、式(3)生成 S 个细菌第 j 维的初始位置；依此构造 Bac
  [S][D]数组；
  }
  for(i=0;i<S;i++)
  fitness[i]=adaptation (Bac[i]);//计算当前细菌个体的适应度值，
```

```
并将它们记录在 fitness[S]数组中
    (3)趋向性操作
for(j=0;j<Nc;j++)
{
  generation++;
  依据式(1)计算 C(d);
  sort(Bac,fitness,S);
  //按数组适应度降序排列所有细菌
for(i=0;i<S;i++) //对每一个细菌进行位置更新
{
  if(i<S*0.4) //针对排在前 40%的细菌获得新位置
{
  产生一个 D 维空间随机方向向量;
  细菌 i 在此方向上前进一单位步长得到 newsite;
}
else
//对剩下 60%的细菌采用差分方式获得新位置
{
  for(r=0;r<D;r++)
  {
    //按照差分的方式(式(5))生成新位置
    r1=random(0,S);
    r2=random(0,S);//随机生成细菌编号
    newsite[r]=Bac[i][r]+(Bac[r1][r]-Bac[r2][r]);
}
  }
  //逐维更新细菌 i 的每个维度值
t=0;
while(t<Ns)
{
  adapt_old=fitness[i];
  for(r=0;r<D;r++)
  {
    temp_r=Bac[i][r];
    Bac[i][r]=newsite[r];
    adapt_new=adaptation(Bac[i]);
    if( fitness[i] 优于 adapat_new)
      Bac[i][r]=temp_r;//还原
    }
    //如果位置得到改善,则沿以上方向继续前进,直到位置信息不
      再改善或达到最大次数 Ns 为止
    if( fitness[i] 不优于 adapat_old)
      break
    else
        t++;
    }
  }
}
    (4)复制操作
每隔 50 代进行一次复制操作;
sort(Bac,fitness,S);//按数组适应度降序排列细菌
for(i=0;i<S;i++)
{
  if(i>S*0.5) //针对排在后 50%的细菌获得新位置
  {
    if(i>S*0.75)//对排在最后 75%的细菌个体,按照差分方式
    生成新位置代码同趋向性操作中差分方式获得新位置
    else
    其余细菌采用传统方法,用排在前 25%的细菌个体取代它们
    的位置;
    fitness[i]=adaptation(Bac[i]);
    }
}
```

(5)迁徙操作

①每隔 100 代进行一次迁徙操作;

②为避免迁徙中产生“逃逸”现象,引用文献[17]中的迁徙操作方案,采取精英保留策略,避免“逃逸”现象发生。

(6)若满足终止条件(达到预定进化代数或适应度值达标),则输出最优解位置,算法结束;否则转向步骤(3)。

4 算法性能测试

为了验证新算法的性能,选取了 8 个标准测试函数[18-19]对改进后的算法进行测试。函数说明如表 2 所列,实验参数如表 3 所列。实验测试的软件平台为 Windows7, Visual C++。

表 2 测试函数说明表

函 数	取值范围,理论最优值
$f_1(\vec{x})=\sum_{i=1}^{D}x_i^2$	$-100<x_i<100, f_1(\vec{0})=0$
$f_2(\vec{x})=\sum_{i=1}^{D-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	$-100<x_i<100, f_2(\vec{1})=0$
$f_3(\vec{x})=\sum_{i=1}^{D}(x_i^2+10\cos(2\pi x_i)+10)$	$-10<x_i<10, f_3(\vec{0})=0$
$f_4(\vec{x})=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}+1)$	$-600<x_i<600, f_4(\vec{0})=0$
$f_5(\vec{x})=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{I=1}^{D}x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i)+20+e$	$-32<x_i<32, f_5(\vec{0})=0$
$f_6(\vec{x})=\sum_{i=1}^{D}-x_i\cdot\sin(\sqrt{\lvert x_i\rvert})$	$-500\leqslant x_i\leqslant 500$, $f_6(420.9687)=-D\cdot 418.9829$
$f_7(\vec{x})=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	$-65.536\leqslant x_i\leqslant 65.536$, $f_7(\vec{0})=0$
$f_8(\vec{x})=\sum_{i=1}^{D}\lvert x_i\rvert^{(i+1)}$	$-1\leqslant x_i\leqslant 1, f_8(\vec{0})=0$

表 3 各种 BFO 版本的通用参数设置

S	N_c	N_s	N_{ed}	N_{re}	P_{ed}	$d_{attractant}$	$w_{attractant}$	$w_{repellant}$	$h_{repellant}$	ψ	λ
100	100	4	4	16	0.25	0.1	0.2	10	0.1	0.8	400

4.1 算法收敛性测试

首先,将 f_1-f_5 的测试结果与标准细菌觅食优化算法 BFOA[6]、演化算法 EA[20]、粒子群改进算法 HPSO-TVAC[21]、细菌觅食优化算法与粒子群优化算法相结合的细菌群 BSO[22]共 4 种算法进行比较,以测试一般维度下算法的收敛性。

实验中每个函数运行 50 次,分别计算寻优均值与标准差,算法的终止条件为达到最大迭代次数 500 或与理论最优值的

误差小于 10^{-5};并将数据进行对比,文献中算法迭代 $10^4 \sim 10^6$　　次,对比结果如表 4 所列。

表 4　标准测试函数 30 维收敛精度

测试函数 (本文算法迭代次数)	维度	优化结果均值(标准差)				
		本文算法	BFOA	EA	HPSO-TVAC	BSO
f_1 (500)	30	0.000000 (0.000000)	0.0840 (0.0025)	0.0360 (0.0010)	0.0650 (0.0534)	0.0560 (0.0112)
f_2 (500)	30	0.000001 (0.000001)	58.2160 (14.3254)	31.7380 (3.6452)	706.2630 (951.9533)	15.4710 (2.6550)
f_3 (500)	30	0.000000 (0.000000)	17.5248 (9.8962)	3.7970 (0.8241)	34.8370 (10.1280)	13.7731 (3.9453)
f_4 (500)	30	0.000000 (0.000000)	0.3729 (0.0346)	0.2684 (0.3616)	0.2175 (0.1953)	0.2565 (0.1431)
f_5 (500)	30	0.000000 (0.000000)	2.3243 (1.8833)	0.6059 (0.3372)	0.5684 (0.1927)	0.5954 (0.1246)

从表 4 中可以看出,将改进算法应用在 30 维的优化问题中,寻优结果均优于其他几种对比方法,由此证明改进的寻优方式非常有效。

将本文算法和文献[7]中算法进行对比,以验证改进方法在高维优化问题中的性能。文献[7]提出了基于场的自适应细菌优化算法,一方面给出了基于场的自适应步长公式,另一方面对多维函数进行分场寻优。在基于场的自适应步长公式中,对文献[6]中提出的自适应步长公式进行了改进,改进的步长公式虽然对算法的寻优效率有了一定提升,但公式中含有大量参数,使得公式的适用性受到了限制,统一的参数设置无法适用于种类繁多的问题;在对多维函数寻优时,基于场的寻优方式仅简单地将问题固定地分解为 2 维场和 3 维场进行寻优,未考虑问题自身的特性,使得寻优效率受到很大影响。文献[7]给出了对函数 f_1-f_5 在 100,200,300 维上的寻优结果,本文同样对 f_1-f_5 函数在以上维度上寻优,每个函数运行 50 次,分别计算寻优均值与标准差,并将数据进行对比,文献[7]中算法的迭代次数为 $2\times10^6 \sim 5\times10^6$。对比结果如表 5 所列。

表 5　标准测试函数高维收敛精度

测试函数 (本文算法迭代次数)	寻优均值(标准差)					
	100 维		200 维		300 维	
	ABFOF	本文算法	ABFOF	本文算法	ABFOF	本文算法
f_1 ($2*10^3$)	0.020536 (0.000886)	0.000000 (0.000000)	0.187451 (0.005662)	0.000000 (0.000000)	0.661638 (0.012676)	0.000002 (0.000001)
f_2 ($2*10^3$)	0.983173 (0.056000)	0.000061 (0.000027)	2.163371 (0.105935)	0.000729 (0.000413)	3.247034 (0.140152)	0.000962 (0.000315)
f_3 ($2*10^3$)	0.788590 (0.578410)	0.000090 (0.000003)	1.727793 (0.252216)	0.000139 (0.000915)	7.515161 (1.36952)	0.000471 (0.000233)
f_4 ($2*10^3$)	0.005105 (0.005575)	0.000950 (0.000053)	0.018616 (0.027167)	0.000191 (0.000098)	0.032759 (0.032494)	0.000372 (0.000121)
f_5 ($2*10^3$)	0.007543 (0.000242)	0.000172 (0.000064)	0.008512 (0.000167)	0.000571 (0.000192)	0.009214 (0.000213)	0.000583 (0.000107)

为了更好地验证算法在高维优化问题中的性能,对表 2 中的 8 个标准测试函数进行了实验。分别测试其维度为 500 维、600 维、700 维情况下的寻优效果,并将结果与基本 BFO 算法在同维情况下的寻优效果进行了对比,结果如表 6 所列。

表 6　标准测试函数超高维收敛精度

测试函数 (迭代次数)	寻优均值(标准差)					
	500 维		600 维		700 维	
	BFO	本文算法	BFO	本文算法	BFO	本文算法
f_1 ($1*10^4$)	6.732324 (2.412591)	0.000003 (0.000002)	10.309565 (7.369871)	0.000003 (0.000002)	16.841256 (10.254778)	0.000003 (0.000002)
f_2 ($1*10^4$)	146.705328 (40.173134)	0.002208 (0.004431)	190.328754 (62.173134)	0.005889 (0.004988)	210.247034 (93.140152)	0.010901 (0.008649)
f_3 ($1*10^4$)	33.482314 (17.664745)	0.000512 (0.000327)	69.457123 (42.012413)	0.000662 (0.000399)	101.513161 (70.836952)	0.000989 (0.000564)
f_4 ($1*10^4$)	13.832162 (6.126334)	0.002768 (0.005638)	70.157412 (45.222143)	0.005312 (0.005189)	124.024759 (83.032494)	0.010931 (0.008243)
f_5 ($1*10^4$)	28.324712 (10.562764)	0.003921 (0.004464)	87.332578 (62.012478)	0.005212 (0.061431)	189.214514 (99.004513)	0.009935 (0.006315)
f_6 ($1*10^4$)	9.958626 (5.224893)	0.000008 (0.000021)	20.395213 (16.325871)	0.000013 (0.000031)	41.332547 (25.387454)	0.000017 (0.000024)
f_7 ($1*10^4$)	84.235231 (39.224987)	0.004612 (0.002269)	193.224577 (96.214587)	0.005173 (0.005215)	445.221548 (223.659871)	0.009929 (0.033286)
f_8 ($1*10^4$)	3.221598 (1.269875)	0.000000 (0.000000)	4.221457 (2.911472)	0.000000 (0.000000)	6.522145 (2.985566)	0.000003 (0.000002)

从表 5 和表 6 的对比结果可知,通过对基本 BFO 算法的　　改进,有效提高了算法的收敛精度。

4.2 算法效率分析

在高维寻优过程中，采用分维寻优方式有效提高了算法的收敛精度。分维寻优时，对细菌个体找到的每一个新位置，需要依次取新位置中的每一个维度值，分别计算此维的新位置是否使适应度值更优。如果问题的维度是 D 维，那么对于一个新位置，就需要分 D 次计算每一维度对适应度值的影响。与标准 BFO 算法相比，在其余部分算法时间复杂度相同的情况下，分维寻优部分的时间复杂度将增加 $O(D)$。

然而通过实验发现，虽然分维寻优部分的时间复杂度增加了，但是改进算法总的迭代次数却少于对比算法。这里假设算法其他部分的时间复杂度相同，仅讨论分维寻优部分对总的时间复杂度的影响。以上 3 组实验中的时间复杂度的分析结果见表 7。

表 7 分维寻优时间复杂度对比

维度	30 维		100,200,300 维				500,600,700 维			
	迭代次数	适应度值计算次数	迭代次数	适应度值计算次数			迭代次数	适应度值计算次数		
对比算法	$10^4\sim10^6$	$10^4\sim10^6$	$2\times10^5\sim5\times10^6$	$2\times10^5\sim5\times10^6$			1×10^4	1×10^4		
本文算法	500	1.5×10^4 (30×500)	2×10^3	100 维	200 维	300 维	1×10^4	500 维	600 维	700 维
				2×10^5 ($100\times2\times10^3$)	4×10^5 ($200\times2\times10^3$)	6×10^5 ($300\times2\times10^3$)		5×10^6 (500×10^4)	6×10^6 (600×10^4)	7×10^6 (700×10^4)

从表 7 中可以看出，在维度为 30,100,200,300 的情况下，本文分维寻优方法的适应度值的计算次数并不大于对比算法，即算法总的时间复杂度与对比算法相当，但从表 4、表 5 中可以看到，分维寻优算法精度远远优于对比算法；在维度为 500,600,700 时，测试时选取的迭代次数均为 1×10^4，此时分维寻优适应度值的计算次数分别增加了 500,600,700，即算法时间复杂度增加了 $O(D)$，但从表 6 中可以看到，分维寻优算法的收敛精度平均优于对比算法 10^5 倍。

为了进一步证明算法的效率，特别对第三组实验（见表 6）进行了测试。将对比算法的迭代次数分别增加为 500×10^4（500 维）、600×10^4（600 维）和 700×10^4（700 维），此时对比算法与分维寻优算法的时间复杂度相当。每一维度下，每个函数对比算法分别运行 50 次，在维度为 500 时，对 f_2，f_3，f_5，f_7 函数，最好的情况下精度仅能达到 10^1，对其余函数，最好时精度也仅能达到 10^0。当维度上升到 600 和 700 维时，所有函数的运算精度都变得更糟。在此，仅记录所有函数在以上 3 个维度下运行 50 次的最优结果，对以上实验数据进行了统计，在分维寻优算法和对比算法时间复杂度相当时，同一维度相同的函数，分维寻优算法的精度要优于对比算法 $10^3\sim10^5$。

综上所述，对改进的分维寻优方式，虽然算法的时间复杂度会随维度的增加而增大，但是在相同时间复杂度的情况下，改进算法的精度有了很大的提高，算法简单可行。实验证明，改进算法是有效的。

结束语 本文首先对基本 BFO 算法进行改进，克服了以往自适应步长公式中经验性参数过多、无法真正实现步长自适应的缺点，使步长的取值仅与其求解的问题和细菌个体的当前位置有关，真正实现了步长自适应；将混沌思想和差分进化算法与 BFO 算法相结合，在细菌个体初始化阶段和寻优过程中，最大程度地增加群体多样性，避免算法因为早熟而陷入局部最优值。其次，在高维函数的优化过程中，采用分维寻优的方法，充分利用细菌个体每次位置改变带来的有利信息，极大地提高了算法的精度。通过对 8 个标准测试函数在多维空间寻优的实验证明，改进算法在高维空间中寻优时速度快、精度高、求解过程简单可行，是一种有效解决高维优化问题的新方法。

参 考 文 献

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

[2] ZHOU Y L. Research and application on bacteria foraging optimization algorithm[J]. Computer Engineering and Applications, 2010,46(20):16-21. (in Chinese)
周雅兰. 细菌觅食优化算法的研究与应用[J]. 计算机工程与应用,2010,46(20):16-21.

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

[4] CHEN J C,HU G W,DU X Y. Adaptive variable step size bacterial foraging optimization[J]. Commputer Engineering and Applications,2012,48(33):29-33. (in Chinese)
陈建超,胡桂武,杜小勇. 自适应变步长菌群优化算法[J]. 计算机工程与应用,2012,48(33):29-33.

[5] DAS S,BISWAS A,DASGUPTA S,et al. Bacterial foraging optimization algorithm:theoretical foundations,analysis,and applications[J]. Foundations of Computational Intelligence, 2009, 203:23-55.

[6] DASGUPTA S,DAS S,ABRAHAM A. Adaptive computational chemotaxis in bacterial foraging optimization:an analysis[J]. IEEE Transactions on Evolutionary Compution, 2009, 13(4): 919-941.

[7] XU X. Research on the Bacterial Foraging Optimization Algorithm[D]. Jilin:Jilin University,2012. (in Chinese)
许鑫. 细菌觅食优化算法研究[D]. 吉林:吉林大学,2012.

[8] VERMA P O,HANMANDLU M,KUMAR P,et al. A novel bacterial foraging technique for edge detection[J]. Pattern Recognition Letters,2011,32(8):1187-1196

[9] ZHANG G Y,WU Y G,TAN Y X. Bacterial Foraging Optimization Algorithm with Quantum Behavior[J]. Journal of Electronics & Information Technology, 2013, 35(3): 614-621. (in Chinese)

（下转第 311 页）

证，不过算法也存在一定的不足，例如高维数的特征就会带来较大的计算量；目标识别算法本身的复杂度会影响到整个算法的实时性问题。总而言之，本文算法为光照变化场景下的目标跟踪提供了一个新的方向，而算法的不足也会在以后的研究中进行克服。

参 考 文 献

[1] Ning J F, ZHANG L, ZHANG D, et al. Scale and Orientation Adaptive Mean Shift Tracking [J]. IET Computer Vision, 2012, 6 (1): 52-61.

[2] KHAN Z H, GU Y H, BACKHOUSE A. Robust Visual Object Tracking Using Multi-Mode Anisotropic Mean Shift and Particle Filters [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2011, 21(1): 74-87.

[3] CHEN F, HUANG Y J, LI S Y, et al. Meanshift Target Tracking Algorithm based on Adaptive Window [J]. Electronic Technology, 2016, 45(5): 1-4. (in Chinese)
陈菲，黄勇坚，李淑彦，等. 自适应窗口的 Meanshift 目标跟踪算法[J]. 电子技术，2016，45(5)：1-4.

[4] MEHMOOD R, NAWAZ R, RAO N I. Occlusion handling in meanshift tracking using adaptive window Normalized Cross Correlation[C]// International Bhurban Conference on Applied Sciences & Technology. 2014: 126-129.

[5] HE J, YANG Y. Multi-iterative tracking method using meanshift based on kalman filter[C]// IEEE International Conference on Signal Processing, Communications & Computing. 2014: 22-27.

[6] WANG X, HU Z, FENG J, et al. Mean-Shift tracking algorithm based on Kalman filter using adaptive window and sub-blocking [C]// World Congress on Intelligent Control & Automation. 2015: 5438-5443.

[7] TAN X, TRIGGS B. Enhanced local texture feature sets for face recognition under difficult lighting conditions[C]// Analysis and Modeling of Faces and Gestures. 2007: 168-182.

[8] TOPI M, MATTI P, TIMO O. Texture Classification by Multi-Predicate Local Binary Pattern Operators[J]. International Conference on Pattern Recognition, 2000, 3(3): 3951.

[9] ROSIN P L. Thresholding for change detection [J]. Computer Vision and Image Understanding, 2002, 86(2): 79-85.

[10] TAN X, TRIGGS B. Fusing Gabor and LBP features sets for kernel based face recognition[J]. International Conference on Analysis & Modeling of Fce & Gestures, 2007, 28(9): 235-249.

[11] FERNÁNDEZ-CABALLERO A, CASTILLO J C, MARTÍNEZ-CANTOS J, et al. Optical flow or image subtraction in human detection from infrared camera on mobile robot[J]. Robotics and Autonomous Systems, 2010, 58(12): 1273-1281.

[12] JIAO B, YAN L L, LI W . Fast convergent Gaussian Mixture Model in moving objects detection [C] // IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2011. 2011: 422-425.

(上接第 274 页)

章国勇，伍永刚，谭宇翔. 一种具有量子行为的细菌觅食优化算法[J]. 电子与信息学报，2013，35(3)：614-621.

[10] LIU X L, ZHAO K L. Bacteria foraging optimization algorithm based on immune algorithm[J]. Journal of Computer Applications, 2012, 32(3): 634-637, 653. (in Chinese)
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)：634-637，653.

[11] SABER A Y. Economic dispatch using particle swarm optimization with bacterial foraging effect [J]. Electrical Power and Energy Systems, 2012, 34(1): 38-46.

[12] LI J, DANG J W, BU F. Study on Adaptive Step Length Bacterial Foraging Algorithm[J]. Journal of Lanzhou Jiaotong University, 2013, 32(6): 10-14. (in Chinese)
李珺，党建武，卜锋. 自适应步长的细菌觅食优化算法研究[J]. 兰州交通大学学报，2013，32(6)：10-14.

[13] CHEN G, WU X D, ZHU X Q, et al. Efficient string matching with wildcards and length constraints [J]. Knowledge and Information Systems, 2006, 10(4): 399-419.

[14] NAVARRO G. Regular expression searching over Ziv-Lempel compressed text[C]// Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching. Berlin: Springer, 2001: 1-17.

[15] STORN R, PDEE K. Differential Evolution-A Simple and Efficient Heuristic for Global optimization over Continuous Spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[16] LI J, DANG J W. Bacterial Foraging Algorithm for Solving High-Dimensional Optimization Problems [J]. Application research of Computers, 2016, 33(4): 1024-1027, 1033. (in Chinese)
李珺，党建武. 细菌觅食算法求解高维优化问题[J]. 计算机应用研究，2016，33(4)：1024-1027，1033.

[17] LI J, DANG J W, BU F. Reseach and Improve on Bacteria Foraging Optimization Algorithm[J]. Computer Simulation, 2013, 30 (4): 344-347, 415. (in Chinese)
李珺，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)：344-347，415.

[18] CEC2005[DB/OL]. http://www3.ntu.edu.sg/home/EPNSugan/index_files/ CEC-05/CEC05.htm.

[19] YAO X, LIU Y, LIN G. Evolutionary programming made faster [J]. IEEE Transactions on Evolutionary Computation, 1999, 3 (2): 82-102.

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

[21] BISWAS A, DASGUPTA S, DAS S, et al. Synergy of PSO and bacterial foraging optimization: a comparative study on numerical benchmarks [C] // Innovations in Hybrid Intelligent Systems. 2007: 255-263.

[22] KRINK T, VESTERSTROM J S, RIGET J. Particle swarm optimization with spatial particle extension [C] // Proceedings of IEEE Congress on Evolutionary Computation. Honolulu, Hawaii USA, 2002: 1474-1497.