基于自适应搜索中心的骨干粒子群算法

王东风 孟 丽 赵文杰

(华北电力大学 控制与计算机工程学院 河北 保定 071003)

摘 要 该文在对标准粒子群算法(Particle Swarm Optimization,PSO)和骨干粒子群算法(Bare Bones Particle Swarm Optimization,BBPSO)中粒子位置的概率密度函数进行分析比较的基础上,对 BBPSO 进行了改进,并证明了改进算法以概率 1 收敛于全局最优解.在改进算法中,主要包括如下策略:(1)基于粒子间适应值的差异,提出一种对粒子位置高斯采样均值的自适应调整策略,分析了其作用机理,提出的搜索中心自适应调整策略增加了粒子分布中心的分散度,减缓粒子在中心的聚集趋势;(2)提出了一种"镜像墙"的越界粒子处理方法,该方法能够大幅度地提高算法找到最优解的概率;(3)粒子在不同的进化时期按不同的拓扑结构选取榜样粒子:算法前期主要采用随机结构以增加群体的多样性,算法后期主要采用全局结构以使得搜索更加精细.将该文提出的算法与多种形式的改进 PSO,如 GPSO(Global PSO)、LPSO(Local PSO)、FIPS(Fully Informed Particle Swarm)、CLPSO(Comprehensive Learning PSO)、HPSO-TVAC(Hierarchical PSO with Time-Varying Acceleration Coefficients)、APSO(Adaptive PSO)、DMS-PSO(Dynamic Multi-Swarm PSO)、OPSO(Orthogonal PSO)、OLPSO(Orthogonal Learning PSO)、ALC-PSO(PSO with an Aging Leader and Challengers)等,以及 BBPSO 的标准版本和改进版本,如 BBJ2(BBPSO with Jumps)、ABPSO(Adaptive BBPSO)、SMA-BBPSO(BBPSO with Scale Matrix Adaptation)等,对 CEC2013 标准函数进行测试,对实验数据进行非参数检验,结果表明该文改进算法的综合表现要优于其他算法.

关键词 粒子群算法;骨干粒子群算法;概率密度;搜索中心;全局收敛

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2016.02652

Improved Bare Bones Particle Swarm Optimization with Adaptive Search Center

WANG Dong-Feng MENG Li ZHAO Wen-Jie

(*School of Control and Computer Engineering, North China Electric Power University, Baoding, Hebei* 071003)

Abstract In particle swarm optimization (PSO) and bare bones particle swarms optimization (BBPSO), particle positions at every generation can be regarded as random variables. Based on the comparison of probability density distribution functions of the position variables in the two algorithms, this paper proposes an improved bare bones particle swarm optimization algorithm, which is guaranteed to converge to global optimum solution with probability one. There are following strategies in improved algorithm: (1) According to the diversity of individual fitness, the mean of Gaussian distribution of each dimension is controlled adaptively. The action mechanism of which was analyzed. This strategy can increase the dispersal of distribution center and reduce the concentration of particle around the center; (2) A new boundary condition, which works like a mirror wall, is applied. The method can greatly improve the algorithm to find the optimal solution in probability; (3) In order to balance the exploration and exploitation ability for different periods, particle chooses its exemplar according to different population topologies during evolu-

收稿日期:2015-01-18;在线出版日期:2015-07-23.本课题得到高等学校博士学科点专项科研基金(20120036120013)、河北省自然科学基金(F2014502059)和中央高校基本科研业务费专项资金(2014MS139)资助.**王东风**,男,1971 年生,博士,教授,主要研究领域为群智能优化算法和智能控制.E-mail:wangdongfeng@ncepubd.edu.cn.**孟 丽**,女,1985 年生,博士研究生,主要研究方向为群智能优化算法及其应用.**赵文杰**,男,1968 年生,博士,副教授,主要研究方向为优化理论和优化控制及其工业应用.

tionary process. At the beginning of the process, random topology is adopted to increase population diversity. At the later stage of the process, global topology is employed for a more accurate search. The algorithm is compared with other improved variations of PSO, e. g., GPSO(Global PSO), LPSO(Local PSO), FIPS(Fully Informed Particle Swarm), CLPSO(Comprehensive Learning PSO), HPSO-TVAC(Hierarchical PSO with Time-Varying Acceleration Coefficients), APSO(Adaptive PSO), DMS-PSO(Dynamic Multi-Swarm PSO), OPSO(Orthogonal PSO), OLPSO(Orthogonal Learning PSO), ALC-PSO(PSO with an Aging Leader and Challengers), and it is compared with the standard version as well as the improved versions of BBPSO, e. g., BBJ2(BBPSO with Jumps), ABPSO(Adaptive BBPSO), SMA-BBPSO(BBPSO with Scale Matrix Adaptation), and so on. The comparison was done on CEC2013 benchmark functions. Non-parametric test is carried on experimental data. The results show that the proposed algorithm has a better overall performance on the test functions.

Keywords　particle swarm optimization; bare bones particle swarm optimization; probability density; search center; global convergence

1　引　言

随着工程优化问题的日益复杂，传统的优化方法已经不能够满足实际的需求，智能优化算法在这种背景下产生并迅速发展. 智能优化算法是建立在生物智能或物理现象基础上的随机搜索算法，包括模拟退火算法、遗传算法、人工神经网络技术、人工免疫算法和群智能算法等. 其中，群智能算法通过模拟社会性动物的各种群体行为，利用群体中个体之间的信息交互和合作来实现寻优的目的，如蚁群算法、粒子群算法、混合蛙跳算法、人工蜂群算法、萤火虫算法等. 在众多的群智能算法中，粒子群优化算法(Particle Swarm Optimization，PSO)自 1995 年由 Kennedy 和 Eberhart[1] 提出就受到广泛的关注，其基本思想来源于鸟群和鱼群等寻找食物的行为. PSO 概念简单，易于实现，已被成功应用于参数辨识、电力系统优化、神经网络训练、数据挖掘等多种领域.

Shi 等人[2] 提出了带有惯性权重的 PSO，一般将其默认为标准 PSO(Standard PSO，SPSO)，随后，Kennedy 等人[3] 研究了不同的拓扑结构对 SPSO 性能的影响. SPSO 存在易早熟收敛，寻优精度不高的缺点，学者们从参数调节、种群拓扑结构以及与其他算法结合等多个方面对其进行了改进. 在众多对 SPSO的改进工作中，有一些被广泛熟知的方法，如 FIPS(Fully Informed Particle Swarm)[4]、CLPSO(Comprehensive Learning PSO)[5]、HPSO-TVAC(Hierarchical PSO with Time-Varying Acceleration Coefficients)[6]、APSO(Adaptive PSO)[7]、DMS-PSO(Dynamic Multi-Swarm PSO)[8]、OPSO(Orthogonal PSO)[9]、OLPSO(Orthogonal Learning PSO)[10]、ALC-PSO(PSO with an Aging Leader and Challengers)[11] 等. FIPS 和 CLPSO 主要从群体的拓扑结构方面对算法进行改进. FIPS 中，粒子的更新并不是仅仅利用其邻域内的最优粒子，而是使用邻域内所有成员的历史最优的加权平均值来指导更新；CLPSO 中粒子的每一维随机地选择自身或其他粒子为榜样粒子，当某个粒子的停滞代数超过预设值时，重新为其选择榜样粒子. HPSO-TVAC 和 APSO 都对算法的参数进行了调节. HPSO-TVAC 中学习因子 c_1 和 c_2 随着算法的迭代线性地变化；APSO 利用粒子之间的距离提供的信息确定种群所处的状态，根据不同的状态采用不同的参数调整策略. DMS-PSO 将群体分为若干个子群，子群体每经过数代进化后进行重新分组，使得信息能够在整个种群中进行流通. OPSO 和 OLPSO 都是将正交实验法引入 PSO，OPSO 利用正交实验法对群体的初始化进行改进，使得初始种群均匀分布在解空间上；OLPSO 通过对粒子的个体极值和邻域内最优粒子进行正交试验，获得更有价值的榜样粒子. ALC-PSO 在 PSO 中结合了衰老进化理论的思想，对群体的首领粒子设置寿命和年龄，根据首领粒子的领导能力调整其寿命，当首领粒子的年龄到达其寿命后，生成新的粒子对首领粒子进行挑战. 以上这些改进使得算法的性能有了很大的提高，但同时也在一

定程度上增加了算法的应用难度.

虽然 SPSO 的概念比较简单,但我们对粒子的位置生成仍然没有一个直观的印象,随着对 SPSO 算法的改进越来越复杂,了解改进算法中的粒子的运行方式也越来越困难. Kennedy[12] 于 2003 年提出了一种更为明晰的粒子群算法的形式:骨干粒子群算法(Bare Bones PSO, BBPSO). BBPSO 是在对 SPSO 中粒子的运行轨迹进行分析的基础上提出的,算法取消了速度项,粒子的位置由服从高斯分布的随机采样直接获得. BBPSO 的简单协作式的概率搜索方式能够提高算法的搜索效率和精度,并且避免了 SPSO 算法复杂的参数调节,已被成功应用于约束优化问题[13-14]、数据挖掘[15]、电力系统调控[16]等多种领域. BBPSO 在处理单峰问题上,表现出了很好的高效性,然而在一些多峰函数上,BBPSO 的效果不甚理想. 为了增加种群的多样性,Krohling 和 Mendel[17] 通过高斯分布或柯西分布产生扰动,帮助群体跳出局部最优,但不同的测试函数需要设置不同的扰动幅度. Blackwell 和 Majid [18-19] 提出了两种带有均匀变异的 BBPSO(BBPSO with Jumps, BBJ):BBJ1 和 BBJ2,两种算法中都是通过设置选择概率使粒子可以通过均匀分布产生变异点进行扰动,来减缓种群多样性的丧失,但是过多的变异会造成资源的浪费,过少则不利于群体跳出局部最优. Zhang 等人[20] 提出了 ABPSO(Adaptive BBPSO),算法根据粒子的聚散程度和种群的多样性自适应地调节高斯采样的标准差,同时采用变异算子进一步增加群体的多样性. 除以上对 BBPSO 的改进方案外,使用重尾分布代替高斯分布产生粒子的新位置也可以增大算法跳出局部极值的机会,Campos 等人[21] 将 t 分布应用在 BBPSO 的框架中,提出了 SMA-BB(BBPSO with Scale Matrix Adaptation).

BBPSO 中粒子的更新具有非常直观的物理意义,使得我们能够很容易地去调整粒子重点搜索的范围,因此,BBPSO 是一种具有潜力的算法. 本文首先将某一代进化过程中粒子的可能分布位置视为随机变量,对其概率密度函数进行求取,从一个崭新的角度分析了 SPSO 和 BBPSO 中粒子的生成方式,进而说明了 BBPSO 在多峰函数上不足的原因. 在此基础上,提出了自适应调整粒子搜索中心的策略,对 BBSPO 算法进行改进,并在改进算法中使用了新的边界策略,使得计算资源得到充分的利用. 仿真实验表明,经过改进后的算法在保留了原 BBPSO 的物理意义明确和搜索高效性的基础上,具有良好的全局搜索能力. 由于没有引入任何需要调节的参数,改进后的算法依然易于在实际中进行应用.

2 BBPSO 与 SPSO 的粒子分布比较

2.1 两种算法的基本形式

BBPSO 是在 SPSO 的基础上衍生出来的,首先给出 SPSO 的形式,在 SPSO 中,粒子代表 D 维寻优空间中的候选解,具有速度和位置两个属性,$k+1$ 时刻粒子 i 的第 d 维的速度 v_{id} 和位置 x_{id} 按式(1)和(2)进行更新:

$$v_{id}(k+1) = wv_{id}(k) + c_1 r_1 (p_{id}(k) - x_{id}(k)) + c_2 r_2 (p_{gd}(k) - x_{id}(k)) \quad (1)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (2)$$

式中:p_{id} 为个体 i 的历史最优解,即个体极值;p_{gd} 为整个群体的历史最优解,即全局极值. w 为惯性权重,c_1 和 c_2 为学习因子,r_1 和 r_2 为相互独立的服从(0,1)间均匀分布的随机数.

Clerc 和 Kennedy[22] 对 SPSO 中粒子的运行轨迹进行了分析,指出粒子的运行轨迹以 $\boldsymbol{p}_i$ 和 $\boldsymbol{p}_g$ 的加权平均值为中心进行振荡. Kennedy[12] 在此基础上对粒子轨迹振荡的中心和幅值进行了进一步研究,提出了 BBPSO,粒子位置的更新直接通过高斯分布采样得到,如式(3)

$$X_{id}(k+1) \sim N(\mu_{id}, \delta_{id}^2) \quad (3)$$

其中:$N(\mu, \delta^2)$ 表示均值为 μ、标准差为 δ 的高斯分布,$\mu_{id} = (p_{id}(k) + p_{gd}(k))/2$,$\delta_{id} = |p_{id}(k) - p_{gd}(k)|$;$X_{id}$ 为表示粒子 i 第 d 维位置的随机变量,服从高斯分布,新位置 $x_{id}(k+1)$ 为 $X_{id}(k+1)$ 的一个采样点. 由高斯采样对粒子 i 的所有维更新得到 $\boldsymbol{x}_i(k+1)$ 后,对个体极值和全局极值进行更新,以便进行下一代的进化.

Kennedy 同时提出了探索型 BBPSO,粒子的每一维以 50% 的概率选择保留个体极值,以 50% 的概率进行高斯分布的采样. 对以上两种形式的 BBPSO 进行试验时发现,当算法采用相同的拓扑结构时,探索型 BBPSO 要相对稳定,因此本文基于探索型 BBPSO进行改进.

2.2 两种算法粒子位置的分布情况

在 SPSO 中,由于式(1)中存在随机数 r_1 和 r_2,得到的 x_{id} 也为某一随机变量的采样点. 一般分析粒子轨迹的工作都是考虑粒子进化数代的过程中作为随机数的 x_{id} 的变化. Kennedy 同样从该角度对粒子的运行轨迹进行了总结,在假定 $\boldsymbol{p}_i$ 和 $\boldsymbol{p}_g$ 不变的前提

下，SPSO 中粒子进化 10^6 代，记录每一代的位置并绘制频率直方图进行分析[12]，指出："速度项是不必要的"，因而在 BBPSO 中直接取消了速度项. 现只考虑在某一 $k+1$ 时刻的情形，粒子 i 的第 d 维位置为一随机变量 X_{id}. 不同的进化策略使得 X_{id} 具有不同的分布规律，决定了粒子对某个区域的搜索能力，进而决定了不同的算法的优化性能的不同. 本节对 BBPSO 和 SPSO 两种算法中粒子位置随机变量 X_{id} 的分布情况进行比较，分析简单去掉速度项对 BBPSO 的影响，进而在下一节提出 BBPSO 的改进策略.

描述随机变量的统计规律性一般要用到概率密度函数. 我们首先对 SPSO 中位置变量 X_{id} 的概率密度函数进行求取. 将 SPSO 用随机变量的形式来表示，如式(4)～(6)：

$$a = wv_{id}(k) + x_{id}(k) \tag{4}$$

$$Z = Y_1 + Y_2 \tag{5}$$

$$X_{id}(k+1) = a + Z \tag{6}$$

式(4)中的 a 由 k 时刻的速度和位置决定，因此在特定的 $k+1$ 时刻，a 为常数. 式(5)中，Y_1 为服从 $(0,b_1)$ 间均匀分布的随机变量，$b_1 = c_1(p_{id}(k) - x_{id}(k))$；$Y_2$ 为服从 $(0,b_2)$ 间均匀分布的随机变量，$b_2 = c_2(p_{gd}(k) - x_{id}(k))$. Y_1 和 Y_2 是相互独立的. 设 Y_1，Y_2 和 Z 的概率密度函数分别为 $f_{Y_1}(y_1)$，$f_{Y_2}(y_2)$ 和 $f_Z(z)$. 由卷积公式可得 $f_Z(z)$ 的计算如式(7)

$$f_Z(z) = \int_{-\infty}^{\infty} f_{Y_1}(y_1) f_{Y_2}(z - y_1)\mathrm{d}y_1 \tag{7}$$

计算得 $f_Z(z)$ 为分段函数，如图 1 所示，分段点 z_1，z_2，z_3 和 z_4 由 b_1，b_2 计算得到，与 b_1 和 b_2 的大小有关，即与 x_{id}，p_{id} 和 p_{gd} 之间相互的位置有关，将 $[0, b_1, b_2, b_1+b_2]$ 按照从小到大的顺序进行排列，对应即得到 $[z_1, z_2, z_3, z_4]$.

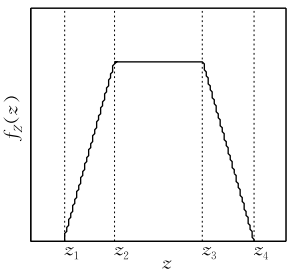


图 1　变量 Z 的概率密度函数图形

由式(6)可知，SPSO 中，X_{id} 的概率密度函数的形状与 $f_Z(z)$ 相同，只是沿着变量轴进行了大小为 a 的平移. 也就是说，在 SPSO 中 $k+1$ 时刻的位置更新时，k 时刻的位置与当前的 p_{id} 和 p_{gd} 共同决定了新位置的可能分布区间大小和分布的概率，k 时刻的速度则影响整个分布覆盖的位置.

容易得到 BBPSO 中 X_{id} 的概率密度函数曲线是以 μ_{id} 为对称中心的钟形曲线. μ_{id} 为位置参数，决定了曲线的位置，δ_{id} 为尺度参数，决定了 X_{id} 落在 μ_{id} 附近概率的大小. 由此可见，粒子的搜索区域的中心定位在 p_{id} 和 p_{gd} 的中间位置 μ_{id} 上. 在进化初期，群体较为分散，δ_{id} 较大，产生新位置的分布相对分散，便于进行全局搜索；在进化后期，群体较为集中，粒子则围绕 μ_{id} 进行精细的搜索.

由两种算法中粒子位置随机变量 X_{id} 的概率密度可以看出，BBPSO 中粒子的搜索范围更广，能够产生远离局部极值的变异点. 但是，粒子的位置分布中心 μ_{id} 的位置不够灵活，限制了粒子重点搜索的区域. 而在 SPSO 中速度项的存在则使得粒子分布沿着寻优空间可以进行一定的平移，避免群体多样性的过快丧失. 虽然 BBPSO 本身就能够产生变异点增加群体的多样性，但是由于我们事先并不知道函数的局部最优点和全局最优点的相对位置，当群体陷入局部最优时，依靠随机产生变异点来跳出局部极值，是有一定难度的，因此，可以通过增加粒子的重点搜索区域的灵活性来进一步增加种群的多样性，进而增强算法的全局搜索能力，避免群体陷入局部极值.

3　基于搜索中心 μ 自适应调节的改进 BBPSO 算法：Aμ-BB

我们对 BBPSO 的改进主要包括两个方面：一是对粒子的搜索中心 μ 进行自适应调节，以增加单个粒子重点搜索区域的多样性，进而增强整个种群的全局搜索能力；二是引入新的边界控制策略. 本节首先详细介绍 μ 的自适应调节的实现方法及其作用机理，并将其应用于拓扑结构不同的两种 BBPSO 上，观察这一策略给算法带来的收益；接着提出了新的边界控制策略，举例说明了对算法的影响；最后基于以上两个方面，给出了基于 μ 自适应调节的改进 BBPSO 算法(BBPSO with Adaptive μ，Aμ-BB).

3.1　搜索中心自适应调节策略

自适应的思想已被应用在遗传算法[23-24]、模拟退火算法[25-26]、差分进化算法[27-28]、粒子群算法[7,29-31]等多种优化算法中，使得算法能够在运行的过程中根据种群提供的信息动态地控制参数或选择个体进化的方式. 自适应策略充分利用了进化过

程本身的信息,增强了个体的"智能",减少了算法的性能对参数和函数的依赖,增强了算法的鲁棒性.在对 BBPSO 的改进中也有基于自适应思想的工作.ABPSO[20]在每个粒子的高斯分布的标准差 δ 上增加不同程度的正值扰动,扰动的大小取决于种群的聚散程度和粒子与全局最优粒子之间适应值的差异.ABPSO 对 δ 的调节可以增加种群的多样性,提高 BBPSO 的优化性能,但在 ABPSO 中,并未对高斯分布的均值 μ 做任何改进.而从 2.2 节的分析中我们得知,BBPSO 中高斯分布均值 μ 的生成方式限制了种群的多样性和算法的优化性能,因此,在本文对 μ 进行自适应的调整,通过对其作用机理的分析可知,本文的 μ 值自适应调节策略能够更大程度地增加群体的多样性.

3.1.1 搜索中心自适应调节的实现

为了使粒子的搜索中心多样化,赋予 μ_{id} 一个取值范围$[\mu_{\min},\mu_{\max}]$,μ_{id} 在其间按均匀分布取值.假设粒子 i 更新第 d 维变量时所用的榜样为粒子 m,$f(\boldsymbol{p}_i)$和 $f(\boldsymbol{p}_m)$分别是粒子 i 和 m 的历史最优函数值.以最小化问题为例,按照式(8)~(11)可以得到 μ_{id} 的取值范围$[\mu_{\min},\mu_{\max}]$:

$$K=\frac{f(\boldsymbol{p}_m)-f(\boldsymbol{p}_i)}{f_{\max}-f_{\min}+\varepsilon} \tag{8}$$

$$B_1=\begin{cases}p_{md}, & f(\boldsymbol{p}_i)<f(\boldsymbol{p}_m)\\ p_{id}, & f(\boldsymbol{p}_i)\geq f(\boldsymbol{p}_m)\end{cases} \tag{9}$$

$$B_2=\begin{cases}p_{id}+K(p_{id}-p_{md}), & f(\boldsymbol{p}_i)<f(\boldsymbol{p}_m)\\ p_{md}+K(p_{id}-p_{md}), & f(\boldsymbol{p}_i)\geq f(\boldsymbol{p}_m)\end{cases} \tag{10}$$

$$\mu_{\min}=\min(B_1,B_2),\ \mu_{\max}=\max(B_1,B_2) \tag{11}$$

式(8)中的 $f_{\max}$ 和 $f_{\min}$ 分别表示群体中所有粒子的个体极值的最大和最小值,ε 是为了避免当 $f_{\max}$ 和 $f_{\min}$ 相等导致算式被零除而引入的非常小的正数,因而 K 的取值范围是$(-1,1)$.当 $f(\boldsymbol{p}_i)$小于 $f(\boldsymbol{p}_m)$时,即 $\boldsymbol{p}_i$ 优于 $\boldsymbol{p}_m$ 时,K 为正值,μ_{id} 的取值范围在$|p_{id}-p_{md}|$的基础上向着 p_{id} 的方向扩展 K 倍;当 $f(\boldsymbol{p}_i)$大于 $f(\boldsymbol{p}_m)$时,即 $\boldsymbol{p}_i$ 劣于 $\boldsymbol{p}_m$ 时,K 为负值,μ_{id} 的取值范围在$|p_{id}-p_{md}|$的基础上向着 p_{md} 的方向扩展$|K|$倍.当 $f(\boldsymbol{p}_i)$和 $f(\boldsymbol{p}_m)$的差异较大时,$|K|$值也较大,使得较优粒子附近比较多的区域包含在了 μ_{id} 的取值范围中,对较优的粒子给予了更多的关注;当 $f(\boldsymbol{p}_i)$和 $f(\boldsymbol{p}_m)$的差异较小时,$|K|$值较小,对 $\boldsymbol{p}_i$ 和 $\boldsymbol{p}_m$ 周围的区域给予的关注并没有很大的差别.在算法迭代初期,群体较为分散,$|p_{id}-p_{md}|$较大,随着迭代的进行,群体趋于集中,$|p_{id}-p_{md}|$也逐渐减小,对于同一个 K 值,可以理解为随着算法的进行扩展的绝对范围逐渐减少,满足算法对前期的勘探能力和后期的开采能力的要求.式(11)中的 $\mu_{\min}$ 或 $\mu_{\max}$ 越界时,进行限制如下:

$$\mu'_{\min}=p_{\min}-\mathrm{mod}(x_{\min}-\mu_{\min},p_{\min}-x_{\min}) \tag{12}$$

$$\mu'_{\max}=p_{\max}-\mathrm{mod}(\mu_{\max}-x_{\max},x_{\max}-p_{\max}) \tag{13}$$

其中:$x_{\min}$ 和 $x_{\max}$ 分别为寻优空间的下界和上界;$p_{\min}$ 和 $p_{\max}$ 分别为 p_{id} 和 p_{md} 中的较小值和较大值.mod()为求余函数.

3.1.2 搜索中心自适应调节的作用机理

将 BBPSO 中粒子的搜索中心进行自适应调节后,粒子的位置 X_{id} 依然为一随机变量,为了观察这一策略对粒子搜索行为的影响,下面求取 X_{id} 服从的概率密度函数.为了使表达简洁且与概率论相统一,省略下标,并且将问题重新描述如下:

随机变量 Y 在$[\mu_{\min},\mu_{\max}]$上均匀取值,当观察到 $Y=y$ 时,$X\sim N(y,\delta^2)$,求取 X 的概率密度函数 $f_X(x)$.其中 δ 为一给定值,而 y 则对应为高斯分布的均值 μ,为$[\mu_{\min},\mu_{\max}]$上的服从均匀分布的采样点.

由问题描述可得到 Y 的概率密度函数 $f_Y(y)$如式(14)所示:

$$f_Y(y)=\begin{cases}\dfrac{1}{\mu_{\max}-\mu_{\min}}, & \mu_{\min}\leq y\leq\mu_{\max}\\ 0, & \text{其他}\end{cases} \tag{14}$$

在 $Y=y$ 的条件下,X 的条件概率密度 $f_{X|Y}(x|y)$如式(15):

$$f_{X|Y}(x|y)=\frac{1}{\sqrt{2\pi}\delta}\mathrm{e}^{-\frac{(x-y)^2}{2\delta^2}},\ -\infty<x<\infty \tag{15}$$

由 $f_Y(y)$和 $f_{X|Y}(x|y)$可求得 Y 和 X 的联合概率密度函数 $f(y,x)$,如式(16):

$$f(y,x)=f_{X|Y}(x|y)f_Y(y)=$$

$$\begin{cases}\dfrac{\mathrm{e}^{-\frac{(x-y)^2}{2\delta^2}}}{(\mu_{\max}-\mu_{\min})\sqrt{2\pi}\delta}, & \mu_{\min}\leq y\leq\mu_{\max},\ -\infty<x<\infty\\ 0, & \text{其他}\end{cases} \tag{16}$$

求得 $f(y,x)$后,可由式(17)求得关于 X 的边缘概率密度函数 $f_X(x)$:

$$f_X(x)=\int_{-\infty}^{\infty}f(y,x)\mathrm{d}y$$

$$=\frac{1}{2(\mu_{\max}-\mu_{\min})}\left[erf\left(\frac{\mu_{\max}-x}{\sqrt{2}\delta}\right)+erf\left(\frac{x-\mu_{\min}}{\sqrt{2}\delta}\right)\right],\quad -\infty<x<\infty \tag{17}$$

其中，$erf(x)$为误差函数，如式(18)：

$$erf(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt \tag{18}$$

将上述问题还原到我们的算法中，式(17)中的$f_X(x)$即为对μ进行调节后得到的粒子位置X的概率密度函数，μ取值的上下限影响$f_X(x)$. 图 2 直观的展示了对μ进行调节后的粒子的分布与原高斯分布的不同. 图中的 GD 表示原 BBPSO 中以$(p_{id}+p_{md})/2$为中心的高斯分布. 图中的K是 3.1.1 节中的扩展系数. 对μ的调节改变了粒子位置随机变量的概率密度函数，扩大了重点搜索区域. 当μ的取值区间向p_{id}的方向扩展时，K为正值；向p_{md}的方向扩展时，K为负值，$|K|$的大小则反映了对较优粒子周围区域关注的程度，这与3.1.1节的分析是一致的.

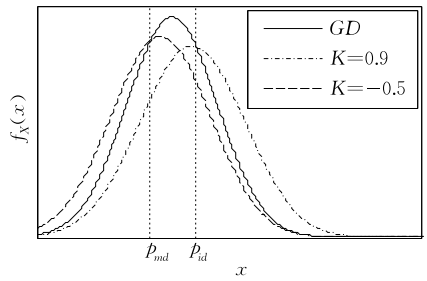


图 2　μ的调节对X的概率密度的影响

3.1.3　搜索中心自适应策略在 BBPSO 上的应用

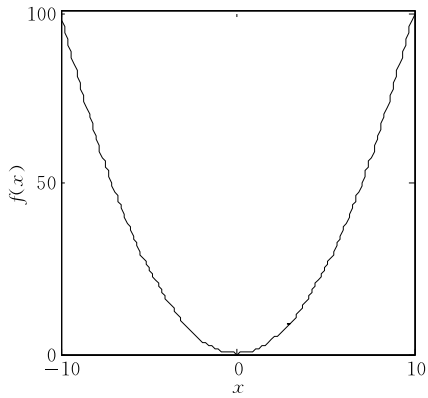
我们将搜索中心自适应调节策略分别应用于两个版本的 BBPSO 上. 当应用于采用全局拓扑结构的 BBPSO，即 GBB(Global BBPSO)，得到 Aμ-GBB (GBB with Adaptive μ)；当应用于采用随机拓扑结构的 BBPSO，即 RBB(Rand BBPSO)，得到 Aμ-RBB (RBB with Adaptive μ). 将这 4 种算法应用于以下 3 个典型函数上进行性能测试：sphere 函数、rastrigin 函数和 schwefel 函数，3 个函数的表达式分别如式(19)、(20)和式(21)所示

$$f(x)=\sum_{i=1}^{D}x_i^2 \tag{19}$$

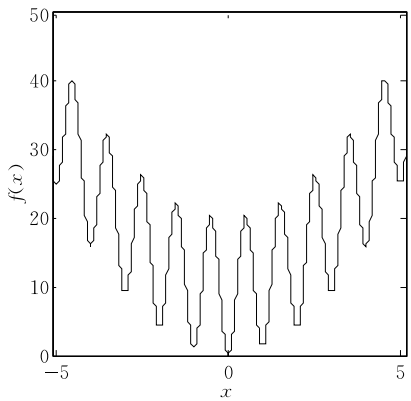
$$f(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10) \tag{20}$$

$$f(x)=418.9828\times D-\sum_{i=1}^{D}x_i\sin(|x_i|^{\frac{1}{2}}) \tag{21}$$

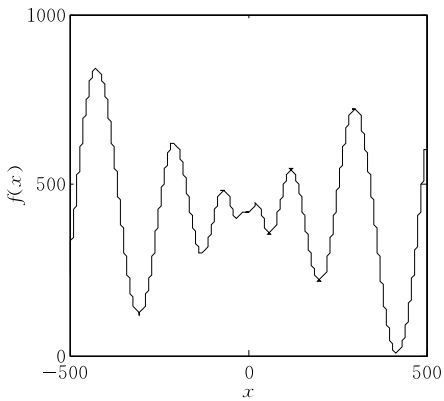
图 3 给出了 3 个函数的一维图像. sphere 为简单的对称单峰函数；rastrigin 具有大量局部最优点，最好的局部最优离全局最优点很近；schwefel 为不可分离的函数，全局最优与最好的局部最优相距很远，算法往往朝着错误的方向收敛. 3 个函数的函数



(a) sphere



(b) rastrigin



(c) schwefel

图 3　3 个典型测试函数一维图像

最优解值f^*均为0.函数取$D=10$,种群大小$N=10$,函数值最大计算次数FEs(Function Evaluations)设置为5×10^4.每个函数独立运行30次,记录每次运行过程中群体最佳函数值与函数全局最优值的差值$(f-f^*)$的变化曲线,取30次的平均值,如图4所示,每个图的横轴为函数值计算次数,纵轴是对$(f-f^*)$取以10为底的对数,从中可以看出对μ进行调节给算法带来的影响.

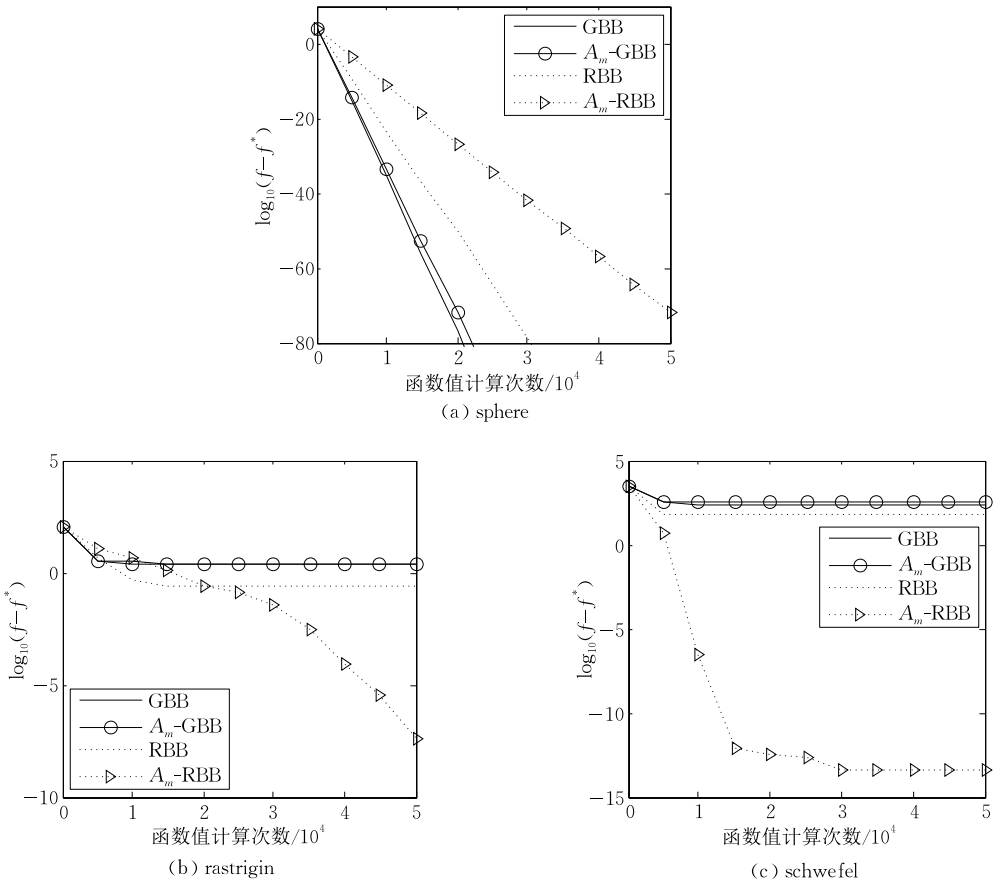


图4 3个基本测试函数的算法收敛曲线

由于sphere的最优解的方向非常容易找到,GBB以最快的速度向最优解收敛,对μ的调节分散了粒子的搜索空间,减缓了GBB和RBB的收敛速度;对rastrigin函数,GBB、RBB和Aμ-GBB都陷入了局部最优,只有Aμ-RBB跳出了局部最优,朝着最优解的方向发展,但收敛速度比较慢;对schwefel函数,也是只有Aμ-RBB能够将搜索区域迅速定位在最优解的附近,而且收敛速度相对较快.

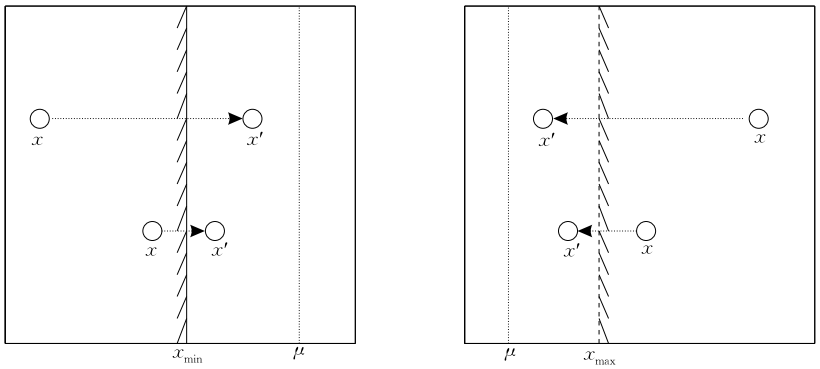
3.2 越界粒子的处理

在群体进化的初期,不同粒子的个体极值之间距离较远,用于更新位置的高斯分布的标准差δ相对较大,导致产生的新位置越过寻优空间边界的机会也较大.PSO中,除了直接将越界粒子的位置和速度均置于边界上,主要还有以下几种方法:吸收墙、反射墙、衰减墙和隐匿墙.前3种方法都是只将越界粒子的位置置于边界上,而对速度的处理不同:吸收墙将粒子的速度置为零;反射墙中粒子的速度大小不变,方向反向;衰减墙将粒子的速度大小减小,方向反向.而隐匿墙对粒子不做处理,不参与极值的更新.在BBPSO中,粒子本身不具有"速度"这一属性,对于越界粒子,常用的处理方法是将其置于边界上.由于在BBPSO中,$k+1$时刻粒子的位置与k的位置没有直接的关系,同时待优化目标函数的最优解往往不在边界上,因此,简单地将越界粒子置于边界上,会造成资源的浪费.这里考虑粒子越过边界的程度,以高斯分布的期望μ为中心将越界的粒子x拉回寻优空间中得到x',对越界粒子按照式(22)进行处理.

$$x' = \mu + \frac{(x_{\text{border}} - \mu)^2}{x - \mu} \tag{22}$$

其中:x_{border}为x越过的边界,即当$x>x_{\max}$时,$x_{\text{border}}=x_{\max}$;当$x<x_{\min}$时,$x_{\text{border}}=x_{\min}$.分析式(22)可知,当$x$越过上界时,即$x>x_{\max}$时,粒子被拉回时落在区间$(\mu,x_{\max})$上.$x$越过$x_{\max}$的程度越小,被拉回时就

越靠近 $x_{\max}$；x 越过 $x_{\max}$ 的程度越大，被拉回时就越靠近中心 μ. 当 x 越过下界时，即 $x<x_{\min}$ 时，粒子被拉回时落在区间 $(x_{\min},\mu)$ 上. 同样的，粒子被拉回的力度与其越过边界的程度成正比. 图 5 展示了对越界粒子的处理，由图中可以看到式(22)所体现的类似“镜像墙”式的越界惩罚机制.



(a) 粒子越过下界的情况　　(b) 粒子越过上界的情况

图 5　对越界粒子的处理

我们仍然以 sphere 函数、rastrigin 函数和 schwefel 函数为例来说明式(22)的边界策略对算法性能的影响. BBPSO 分别采用下列两种边界策略：(1)将越界粒子直接置于边界上；(2)按式(22)处理越界粒子. 函数取 $D=10$，种群大小 $N=10$，函数值最大计算次数 $FEs=5\times10^4$，每个函数独立运行 100 次. 由于 sphere 函数非常简单，策略(1)和(2)的运行结果差别不大，以几乎相同的精度逼近最优解值；优化 rastrigin 函数时，我们发现群体中粒子越界的次数非常大，策略(1)和(2)找到最优解的成功率分别为 51%和 70%；schwefel 函数的最优解靠近寻优空间的边界，策略(1)和(2)找到最优解的成功率分别为 16%和 57%. 由此可见，边界处理的方式会对算法的性能带来一定的影响，本文提出的越界处理方式较大幅度地提高了找到最优解的概率.

3.3　Aμ-BB 算法描述

通过 3.1.3 节可以看到，如果不要求算法的收敛速度，Aμ-RBB 是个很好的选择. 但在实际应用中，我们尽可能的在算法的收敛速度和寻优精度之间进行平衡，因此，我们将 Aμ-GBB 和 Aμ-RBB 结合起来得到 Aμ-BB.

在算法迭代初期，随机结构在群体中占主体，以增大群体在整个寻优空间的搜索力度，减小陷入局部极值的可能；随着迭代的进行，全局结构逐渐占主导地位，进行精细的搜索. 拓扑结构的变化通过粒子对榜样粒子的选择来实现，设置选择概率：$P_c=(k/k_{\max})^{1/2}$，k 为当前的代数，$k_{\max}$ 为算法最大迭代次数. 对每一个粒子的每一维设置一个服从(0,1)间均匀分布的随机数 r_c，当 $r_c>P_c$ 时，粒子随机从群体中选择一个粒子作为自己的榜样粒子；$r_c\le P_c$ 时，榜样粒子为整个群体的最优粒子. 第 $k+1$ 代，Aμ-BB 中粒子 i 的第 d 维位置更新的过程可概括为以下步骤：

1. 生成服从(0,1)间均匀分布的随机数 r_1，若 $r_1<0.5$，执行步 2，否则执行步 7；

2. 计算 P_c，生成服从(0,1)间均匀分布的随机数 r_c. 若 $r_c>P_c$，榜样粒子 m 从种群中随机选取，否则，榜样粒子 m 为全局最优粒子；

3. 由式(8)～(11)计算得到 $\mu_{\min}$ 和 $\mu_{\max}$，并根据式(12)或(13)进行越界处理；

4. 生成服从(0,1)间均匀分布的随机数 r_2，$\mu_{id}(k+1)=\mu_{\min}+r_2(\mu_{\max}-\mu_{\min})$；

5. 若 $p_{id}(k)=p_{md}(k)$，生成服从(0,1)间均匀分布的随机数 r_3，$\delta_{id}(k+1)=r_3(x_{\max}-x_{\min})$；否则，$\delta_{id}(k+1)=|p_{id}(k)-p_{md}(k)|$；

6. 由高斯采样 $N(\mu_{id}(k+1),\delta_{id}(k+1))$ 得到 $x_{id}(k+1)$，执行步 8；

7. $x_{id}(k+1)=p_{id}(k)$；

8. 根据式(22)进行越界粒子的处理.

4　Aμ-BB 的全局收敛性分析

Solis 和 Wets[32] 对随机优化算法的收敛性进行了深入的研究，给出了一般随机优化算法收敛性的判定标准，其主要结论如下.

问题描述. 给定一目标函数 $f(x)$：$R^D\to R$，$S\subset R^D$，$\boldsymbol{x}=(x_1,x_2,\cdots,x_D)\in S$. 在 Lebesgue 测度空间定义搜索的下确界：$\psi=\inf(t:v[\boldsymbol{x}\in S|f(\boldsymbol{x})<t]>0)$，其中，$v(A)$ 表示在集合 A 上的 Lebesgue 测度. 在此基础上定义函数 $f(\boldsymbol{x})$ 的最优区域 $R_{\varepsilon,M}$：

$$R_{\varepsilon,M}=\begin{cases}\{\boldsymbol{x}\in S\mid f(\boldsymbol{x})<\psi+\varepsilon\}, & \psi\text{有限}\\ \{\boldsymbol{x}\in S\mid f(\boldsymbol{x})<-M\}, & \psi=-\infty\end{cases},$$

其中,$\varepsilon>0$,M 为充分大的正值.如果算法找到了 $R_{\varepsilon,M}$ 中的一个点,称算法找到了误差为 ε 的可接受点.

假设 1[32]. 若 $f(H(z,\boldsymbol{\xi}))\leq f(z)$,$\boldsymbol{\xi}\in S$,则

$$f(H(z,\boldsymbol{\xi}))\leq f(\boldsymbol{\xi}).$$

其中:H 为产生问题解的函数;$\boldsymbol{\xi}$ 为从概率空间 $(R^D,\boldsymbol{J},\Omega)$ 产生的随机向量.$\boldsymbol{J}$ 为 R^D 上的一个 σ 代数,Ω 为 $\boldsymbol{J}$ 上的概率测度.

假设 2[32]. 若对 S 的任意 Borel 子集 B,有 $\upsilon(B)>0$,则

$$\prod_{k=0}^{\infty}(1-\Omega_k[B])=0.$$

其中,$\Omega_k[B]$为第 k 代算法 H 在 B 上的概率测度.

定理 1[32]. 设 f 为一可测函数,S 为 R^D 的一可测子集,$\{z_k\}_0^{\infty}$ 为随机算法产生的解序列,则当满足假设 1 和假设 2 时,有 $\lim\limits_{k\to+\infty}P[z_k\in R_{\varepsilon,M}]=1$.

在此基础上,下面给出本文改进算法 Aμ-BB 的全局收敛性.

定理 2. 假设 Aμ-BB 求解的目标函数 f 是可测的,其解空间 S 为可测子集,则算法以概率 1 收敛于全局最优解.

证明. 在 Aμ-BB 中,函数 H 可以描述为

$$H(\boldsymbol{p}_g(k),\boldsymbol{x}_i(k))=\begin{cases}p_g(k), & f(p_g(k))\leq f(x_i(k))\\ x_i(k), & f(p_g(k))>f(x_i(k))\end{cases}\tag{23}$$

容易证明其满足假设 1.

下面证明 Aμ-BB 满足假设 2.

设种群的大小为 N,在 $k+1$ 时刻,粒子 i 的第 d 维 x_{id} 以 0.5 的概率选择 $p_{id}(k)$;以 0.5 的概率从随机变量 X_{id} 中采样,X_{id} 的概率密度函数如式(17)所示.计算可得 $\Omega_k[B]$:

$$\Omega_{k+1}[B]=P(X_{id}(k+1)\in B)\times 0.5+P(p_{id}(k)\in B)\times 0.5\tag{24}$$

由式(17)和(22)可得,只要 $\upsilon(B)>0$,便有 $P(X_{id}(k+1)\in B)>0$ 成立;$P(p_{id}(k)\in B)$的值为 0 或 1,因此,$\Omega_{k+1}[B]>0$ 成立,从而有 $\prod\limits_{k=0}^{\infty}(1-\Omega_k[B])=0$.因此 A$\mu$-BB 满足假设 2.

由定理 1,可得:Aμ-BB 算法以概率 1 收敛于全局最优解. 证毕.

5 仿真实验

5.1 基准函数及其他用于比较的算法

本文用 CEC2013 标准库函数[33]来测试算法的性能,CEC2013 标准测试函数共 28 个,分为单峰函数、基本多峰函数和复合函数三类.由于篇幅所限,本文选取用于比较的各种优化算法在性能上表现出差异较大的 16 个函数(如表 1 所示),将其结果在文中详细展示.另外 12 个函数的运行结果列于附表 1.

文中所用的比较算法如表 2 所示,其中各个算法的参数设置均为原文献的推荐值,群体的拓扑结构也与原文献保持一致.表 2 中的 GPSO 和 LPSO 分别表示采用全局拓扑结构(Global PSO, GPSO)和局部拓扑结构(Local PSO, LPSO)的标准粒子群算法,GPSO 和 LPSO 中的参数设置参考 Shi 等人[34]的工作.表中,$v_{\max}$为粒子速度的边界值,Range 表示搜索区域的范围,其他各参数与原文献保持一致.

表 1 本文用于详细比较的 CEC2013 测试函数

	函数序号	函数名称	最优值 f^*
单峰函数	f_2	Rotated high conditioned elliptic function	−1300
	f_3	Rotated bent cigar function	−1200
	f_4	Rotated discus function	−1100
	f_5	Different powers function	−1000
多峰函数	f_6	Rotated rosenbrock's function	−900
	f_7	Rotated schaffers F7 function	−800
	f_{10}	Rotated griewank's function	−500
	f_{11}	Rastrigin's function	−400
	f_{14}	Schwefel's function	−100
	f_{15}	Rotated Schwefel's function	100
	f_{17}	Lunacek bi_rastrigin function	300
	f_{19}	Expanded griewank's plus rosenbrock's function	500
复合函数	f_{22}	Composition function 2	800
	f_{23}	Composition function 3	900
	f_{27}	Composition function 7	1300
	f_{28}	Composition function 8	1400

表 2　用于比较的其他优化算法及参数

算法名称缩写	发表年份	参数设置	拓扑结构
GPSO[2]	1998	ω:0.9～0.4, $c_1=c_2=2.0$, $v_{\max}=0.5\times Range$	全局结构
LPSO[3]	2002	ω:0.9～0.4, $c_1=c_2=2.0$, $v_{\max}=0.5\times Range$	环形结构
FIPS[4]	2004	$\chi=0.729$, $\sum c_i=4.1$, $v_{\max}=0.2\times Range$	环形结构
CLPSO[5]	2006	ω:0.9～0.4, $c=1.49445$, $m=7$, $v_{\max}=0.2\times Range$	随机结构*
HPSO-TVAC[6]	2004	c_1:2.5～0.5, c_2:0.5～2.5, $v_{\max}=0.5\times Range$, v: $v_{\max}$～$0.1\times v_{\max}$	全局结构
APSO[7]	2009	ω:0.9, $c_1=c_2=2.0$, $v_{\max}=0.2\times Range$	全局结构
DMS-PSO[8]	2005	ω:0.9～0.2, $c_1=c_2=2.0$, $m=3$, $R=5$, $v_{\max}=0.2\times Range$	全局结构
OLPSO[10]	2011	ω:0.9～0.4, $c=2$, $G=5$	环形结构*
ALC-PSO[11]	2013	$\omega=0.4$, $c_1=c_2=2.0$, $\Theta_0=60$, $T=2$, $v_{\max}=0.5\times Range$	全局结构
BBPSO[12]	2003	—	随机结构
BBJ2[19]	2012	$\alpha=0.75$, $p_J=0.001$	全局结构
ABPSO[20]	2014	$prob=0.7$	全局结构
SMA-BBPSO[21]	2014	$m_{\max}=5$, β取值随函数而不同	环形结构

注：随机结构*：CLPSO 中榜样粒子为从随机选取的两个粒子中选择优胜的一个.
环形结构*：OLPSO 中的榜样粒子由全局最优和个体最优进行正交试验得到.

5.2　实验结果及分析

本文对测试函数分别取维数 $D=10$ 和 $D=30$ 进行测试. $D=10$ 时，除 DMS-PSO 外，种群大小 N 均设置为 20，$D=30$ 时，除 DMS-PSO 外，$N=40$. 对于不同的 D，DMS-PSO 选取相同的群体规模，设置 3 个子群，每个子群的大小均为 10. 计算函数值的最大次数 FEs 设置为 $D\times 10^4$[33]. 每个函数的搜索范围和初始化范围均为$[-100,100]^D$[33]. 每个算法对每个函数都独立运行 51 次[33]，记录每次运行迭代过程中的群体最佳函数值与函数全局最优值 f^* 的偏差.

由于篇幅有限，并且考虑到算法的改进方向和在 12 个函数上的综合性能，从表 2 中所示的 13 种算法中选取 10 种算法，将其运算结果详细展示在图 6 和表 3 中. 除了篇幅有限，其他 3 种算法的结果未详细列出的原因还有：GPSO 和 LPSO 都可以做为 PSO 的标准形式，并且在 12 个测试函数上的综合排名比较接近，而 GPSO 较为常见，因此 LPSO 的结果未详细列出；FIPS 和 CLPSO 都是综合运用了邻域内所有粒子的个体极值的信息来对粒子进行更新，在单峰函数上，性能较差，在多峰函数上，性能较好，CLPSO 的综合性能要优于 FIPS，因此 FIPS 的运行结果未详细给出；HPSO-TVAC 在所有的测试函数上都没有突出的表现，综合性能劣于同类型的 APSO，HPSO-TVAC 的结果未详细给出.

图 6 为 $D=10$ 时（$D=30$ 的情况与此相近），运行过程中，算法找到的最佳函数值的平均值与 f^* 的偏差的变化情况. 每个图的横轴为函数值计算次数，纵轴是对每代的运行结果取以 10 为底的对数. 表 3 为 $D=30$ 时（$D=10$ 的情况与此相近）的运行结果，表中的 $Mean$ 和 SD 分别表示独立运行函数 51 次得到的结果的平均值和标准差，$Rank$ 表示算法在某一个函数上的性能排名，如果两个算法的 $Mean$ 值相同，则认为 SD 值较小的优于 SD 值较大的. 位于表格最后面的 Num 表示算法在 12 个测试函数中取得最优的次数，$Ave.R$ 表示算法的平均排名，$T.R$ 表示算法总的排名，若两种算法的 $Ave.R$ 值相同，则认为 Num 值大的算法较优.

由图 6 可以看到，$D=10$ 时，CEC2013 f_5 和 f_{11} 是相对比较容易的，CLPSO，DMS-PSO，ABPSO，SMA-BB 和 Aμ-BB 都找了 f_5 的最优解，CLPSO，ALC-PSO，BBJ2 和 Aμ-BB 都找了 f_{11} 的最优解. 在其他的函数上，所有的算法都没有收敛到函数的全局极值. Aμ-BB 在 f_3，f_6，f_7，f_{10}，f_{14}，f_{27}，f_{28} 上具有略微的优势；OLPSO 在 f_{17} 上取得了最好的结果，在 f_{22} 上的收敛速度较快，但最后到达的解的精度与 CLPSO 基本相同，略微优于其他算法；ABPSO在 f_{15} 和 f_{23} 上表现出了较好的性能；在 f_2 和 f_4 取得最好的结果的算法分别为 SMA-BB 和APSO；各算法在 f_{19} 上的性能差别是很小的.

对比表 3 和图 6 可以看到，当函数的维数由 10 增加到 30 时，所有算法在测试函数上的寻优精度都有所下降. $D=30$ 和 $D=10$ 时，大部分算法的优劣情况是相似的，受维数变化影响较大的算法有 ALC-PSO 和 OLPSO. ALC-PSO 在 $D=30$ 时的综合性能要优于 $D=10$ 时，这是由于 ALC-PSO 中的挑战机制使得挑战者在维数较高和群体规模较大的情况下更容易挑战成功，而在维数较低和群体规模较小的情况下，在生成成功的挑战者时会浪费大量的计算资源，使得算法会有很长时间的停滞. 在

表 3 $D=30$,各算法运行结果

		GPSO	CLPSO	APSO	DMS-PSO	OLPSO	ALC-PSO	BBPSO	BBJ2	ABPSO	SMA-BB	Aμ-BB
	Mean	2.6e+07	1.8e+07	2.3e+06	1.1e+07	2.9e+07	3.0e+06	1.1e+08	2.0e+06	7.8e+06	1.6e+07	1.3e+07
f_2	*SD*	1.1e+07	2.3e+06	2.2e+06	1.6e+06	9.4e+06	2.9e+06	3.0e+07	1.1e+06	6.2e+06	1.6e+07	1.1e+06
	Rank	9	8	2	5	10	3	11	1	4	7	6
	Mean	1.1e+08	1.4e+08	3.9e+09	1.6e+07	2.7e+09	3.3e+06	1.0e+09	3.0e+09	4.0e+06	1.0e+07	2.8e+06
f_3	*SD*	1.0e+08	6.3e+07	5.7e+09	1.3e+07	2.0e+09	4.2e+06	8.4e+08	3.9e+09	5.1e+06	1.3e+07	3.6e+06
	Rank	6	7	11	5	9	2	8	10	3	4	1
	Mean	1.8e+04	2.2e+04	5.4e+03	8.3e+03	9.9e+04	1.2e+03	4.7e+04	2.0e+04	2.4e+03	2.8e+03	9.1e+03
f_4	*SD*	4.9+03	7.6e+03	1.5e+03	8.5e+03	1.5e+04	4.3e+02	8.1e+03	1.4e+04	6.1e+02	5.5e+03	2.8e+03
	Rank	7	9	4	5	11	1	10	8	2	3	6
	Mean	6.0e−13	2.3e−13	9.1e−13	6.2e−13	1.8e−12	2.4e−13	2.2e−13	9.4e−13	2.8e−14	1.0e−13	7.9e−14
f_5	*SD*	9.3e−14	2.0e−14	1.7e−12	1.1e−12	1.3e−12	8.3e−14	2.0e−14	2.2e−13	4.8e−14	5.6e−14	1.6e−13
	Rank	7	5	9	8	11	6	4	10	1	3	2
	Mean	8.2e+01	2.7e+01	3.3e+01	5.7e+01	3.7e+01	7.5e+01	3.5e+01	4.1e+01	3.8e+01	2.2e+01	1.7e+01
f_6	*SD*	3.3e+01	5.2e+00	3.0e+01	2.7e+01	2.4e+01	6.9e+01	1.2e+01	3.2e+01	1.9e+01	2.4e+01	3.2e+00
	Rank	11	3	4	9	6	10	5	8	7	2	1
	Mean	4.4e+01	6.7e+01	1.2e+02	1.4e+01	1.1e+02	3.4e+01	8.5e+01	4.0e+02	1.6e+02	1.3e+02	1.1e+01
f_7	*SD*	2.2e+01	8.0e+00	2.8e+01	8.2e+00	1.4e+01	7.5e+00	9.9e+00	5.1e+02	7.0e+01	2.9e+01	3.7e+00
	Rank	4	5	8	2	7	3	6	11	10	9	1
	Mean	1.7e−01	1.9e+00	2.0e−01	1.5e−01	5.8e+00	4.9e+00	5.5e+01	3.6e−01	5.5e+00	8.4e−01	8.2e−02
f_{10}	*SD*	1.0e−01	5.2e−01	8.8e−02	4.5e−02	2.4e+00	9.23e+00	3.1e+00	2.4e−01	3.9e+00	2.0e−01	3.3e−02
	Rank	3	7	4	2	10	8	11	5	9	6	1
	Mean	1.8e+01	1.4e−13	1.7e+00	5.2e+00	4.3e−01	2.3e−13	7.7e+01	1.9e−13	3.1e+00	5.3e+00	4.9e−01
f_{11}	*SD*	7.9e+00	1.0e−13	6.2e−01	3.5e+00	7.2e−01	6.8e−14	7.6e+01	3.9e−14	1.9e+00	3.1e+00	9.6e−01
	Rank	10	1	6	8	4	3	11	2	7	9	5
	Mean	9.3e+02	2.1e+01	4.4e+02	7.0e+02	4.2e+01	4.7e+02	2.4e+03	8.7e−01	7.3e+02	7.4e+01	3.8e+01
f_{14}	*SD*	2.1e+02	7.1e+00	1.8e+02	2.3e+02	4.1e+01	2.5e+02	1.9e+02	7.3e−01	2.2e+02	2.1e+01	3.8e+01
	Rank	10	2	6	8	4	7	11	1	9	5	3
	Mean	7.1e+03	4.4e+03	6.3e+03	5.0e+03	6.9e+03	4.1e+03	7.5e+03	4.5e+03	3.8e+03	4.2e+03	6.6e+03
f_{15}	*SD*	4.7e+02	6.0e+02	4.8e+02	6.8e+02	3.9e+02	6.0e+02	1.6e+02	6.0e+02	5.3e+02	2.7e+02	7.1e+02
	Rank	10	4	7	6	9	2	11	5	1	3	8
	Mean	9.9e+01	3.7e+01	3.5e+01	1.1e+02	3.8e+01	5.0e+01	1.4e+02	3.0e+01	6.8e+01	4.0e+01	6.3e+01
f_{17}	*SD*	1.5e+01	3.4e+00	3.2e+00	1.3e+01	4.4e+00	1.6e+01	6.8e+00	7.8e−03	1.6e+01	9.8e+00	2.2e+01
	Rank	9	3	2	10	4	6	11	1	8	5	7
	Mean	1.0e+01	1.6e+00	2.5e+00	6.6e+00	2.6e+00	7.5e+01	1.2e+01	1.2e+00	3.1e+00	5.8e+00	2.8e+00
f_{19}	*SD*	1.6e+00	3.2e−01	4.8e−01	1.4e+00	9.6e−01	2.0e+02	8.8e−01	4.9e−01	7.9e−01	2.3e+00	5.2e−01
	Rank	9	2	3	8	4	11	10	1	6	7	5
	Mean	9.3e+02	1.4e+02	1.0e+03	3.5e+02	2.5e+01	3.2e+02	4.4e+03	1.3e+02	1.2e+02	7.2e+02	3.1e+02
f_{22}	*SD*	2.9e+02	8.8e+00	6.5e+02	9.1e+01	9.4e+01	8.8e+01	3.6e+02	7.5e+01	5.4e+01	2.4e+02	1.2e+02
	Rank	9	4	10	7	1	6	11	3	2	8	5
	Mean	7.2e+03	5.5e+03	7.3e+03	4.7e+03	6.6e+03	4.3e+03	7.6e+03	6.2e+03	4.5e+03	5.3e+03	6.0e+03
f_{23}	*SD*	4.3e+02	2.3e+03	4.2e+02	6.1e+02	9.5e+02	1.0e+03	2.9e+02	5.4e+02	6.8e+02	7.7e+02	8.2e+02
	Rank	9	5	10	3	8	1	11	7	2	4	6
	Mean	1.4e+03	8.5e+02	1.0e+03	8.8e+02	1.1e+03	5.1e+02	1.2e+03	1.7e+03	9.0e+02	9.4e+02	7.4e+02
f_{27}	*SD*	7.5e+01	7.6e+02	1.2e+02	4.2e+01	5.4e+01	1.7e+02	1.8e+01	5.7e+01	1.1e+02	1.4e+02	1.4e+02
	Rank	10	3	7	4	8	1	9	11	5	6	2
	Mean	3.4e+02	3.0e+02	3.8e+02	3.3e+02	2.6e+02	3.2e+02	2.9e+02	7.3e+02	7.9e+02	6.4e+02	2.4e+02
f_{28}	*SD*	2.2e+02	1.9e−02	1.9e+02	1.9e+02	8.9e+01	1.4e+02	8.5e+00	1.9e+02	6.4e+02	1.1e+02	8.7e+01
	Rank	7	4	8	6	2	5	3	10	11	9	1
Num		0	1	0	0	1	3	0	4	2	0	5
Ave.R		8.12	4.50	6.31	6.00	6.75	4.68	8.93	5.87	5.43	5.62	3.75
T.R		10	2	8	7	9	3	11	6	4	5	1
Alg		GPSO	CLPSO	APSO	DMS-PSO	OLPSO	ALC-PSO	BBPSO	BBJ2	ABPSO	SMA-BB	Aμ-BB

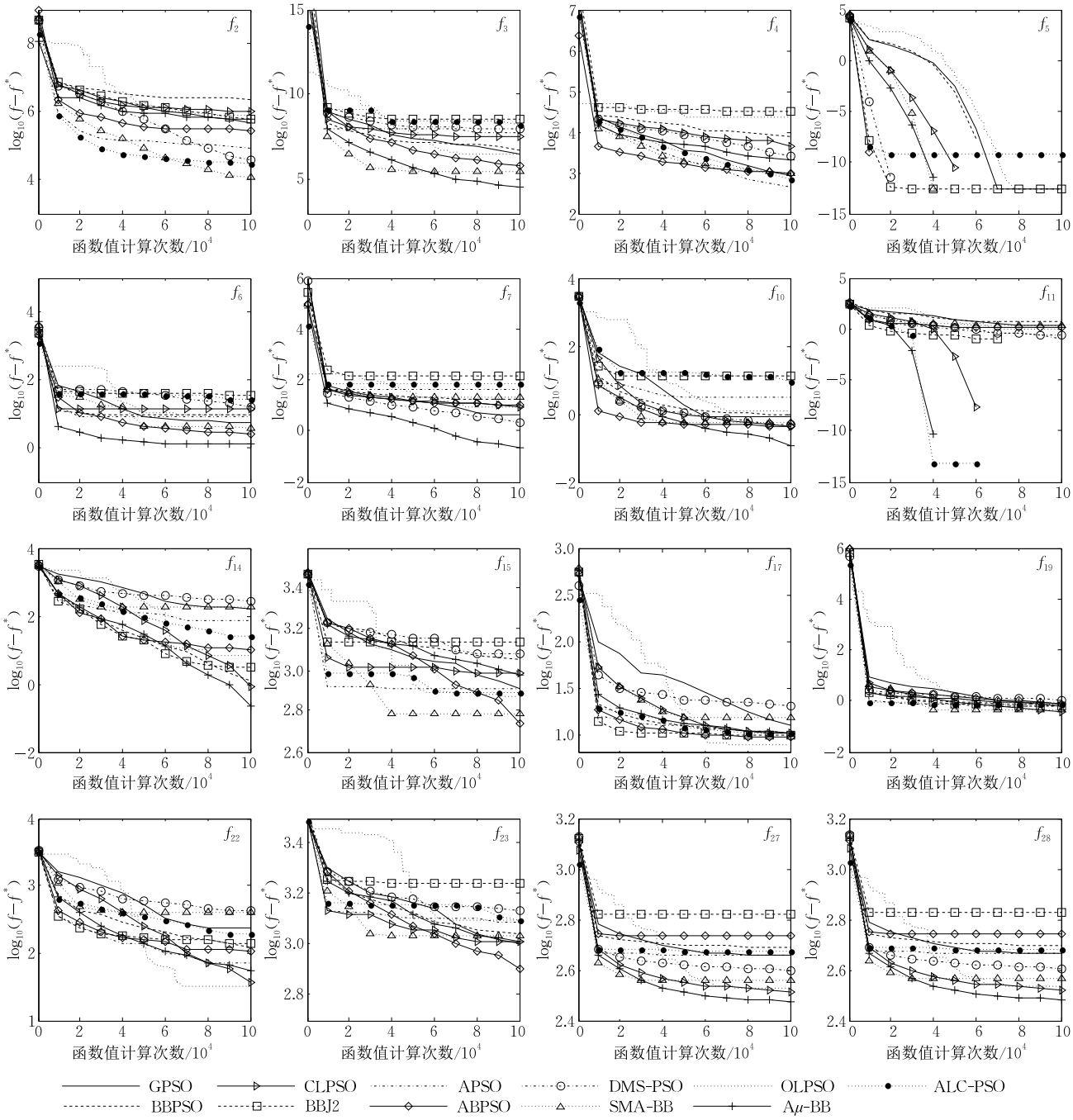


图 6　$D=10$，各算法的收敛曲线

OLPSO中，榜样粒子的生成需要进行正交试验，随着函数维数的增高，进行正交试验时需要计算函数值的次数也增加了，增大了计算量. 同样应用自适应思想对 BBPSO 进行改进的 ABPSO 通过扩大高斯分布的标准差 δ 来增加群体的多样性，减缓了粒子位置分布在 μ 附近的集中趋势，但并没有改变搜索中心，由图 2 可以看到，Aμ-BB 中对 μ 的调节不仅分散了粒子位置的分布，还使得搜索中心更加灵活. 综合 $D=10$ 和 $D=30$ 的运行结果，从总体来看，对单峰函数，ABPSO 的性能优于 Aμ-BB，而对多峰函数，Aμ-BB 的性能一般要优于 ABPSO. 在运行过程中，我们发现 SMA-BB 算法中的可调参数 β 对算法性能的影响很大，虽然 SMA-BB 在某些函数上的表现很好，但是，不同的函数需要设置不同的参数，限制了 SMA-BB 在实际中的应用. 由运算结果可以看到，没有哪一种算法在所有的测试函数上能够取得优胜，Aμ-BB 也并不例外，但是，Aμ-BB 在所有函数上的综合性能还是要优于其他比较算法的.

为了对运算结果有一个比较全面的了解，我们对 Aμ-BB 和其他算法在 $D=30$ 时得到的结果逐一

进行显著性检验.由于不能够确定 PSO 多次优化同一函数的结果服从分布的类型,因此我们采用非参数检验方法,这里选用 Mann-Whitney 检验方法,显著水平 $\alpha=0.05$.表 4 详细列出了表 3 中用于比较的 10 种算法的检验数据,这里以 GPSO 列为例说明表中各数据的意义:GPSO 列与 f_1 行交叉的数据为"0.000↑","0.000"为 Mann-Whitney 检验的 p 值,"↑"表示 GPSO 在 f_1 上的运行数据的秩均值大于 Aμ-BB 的运行数据的秩均值;若为"↓",则表示 GPSO 在此函数上的运行数据的秩均值小于 Aμ-BB 的运行数据的秩均值;若为"→",则表示两种算法的运行数据的秩均值相等;表中的 Better 行、Same 行和 Worse 行分别表示在显著水平为 0.05 下,12 个测试函数中 Aμ-BB 优于 GPSO、与 GPSO 无差和劣于 GPSO 的函数的个数.从表 4 中,我们可以清楚地看到,在 12 个测试函数上,Aμ-BB 和其他算法的综合性能的比较结果,表 4 中未列出的 3 个算法的比较结果如下:Aμ-BB 优于 LPSO、与 LPSO 相同和劣于 LPSO 的函数个数分别为 16,0,0;Aμ-BB 优于 FIPS、与 FIPS 相同和劣于 FIPS 的函数个数分别为 12,2,2;Aμ-BB 优于 HPSO-TVAC、与 HPSO-TVAC 相同和劣于 HPSO-TVAC 的函数个数分别为 11,3,2.以上均表现出 Aμ-BB 明显占优.

表 4 $D=30$,Mann-Whitney 检验数据

	GPSO	CLPSO	APSO	DMS-PSO	OLPSO	ALC-PSO	BBPSO	BBJ2	ABPSO	SMA-BB
f_2	0.001↑	0.608↓	0.255↓	0.180↓	0.000↑	0.181↓	0.000↑	0.061↓	0.255↓	0.041↑
f_3	0.000↑	0.000↑	0.000↑	0.018↑	0.000↑	0.197↑	0.000↑	0.000↑	0.002↑	0.788↓
f_4	0.000↑	0.000↑	0.000↓	0.000↓	0.000↑	0.000↓	0.000↑	0.654↑	0.000↓	0.000↓
f_5	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.062↓	0.000↑
f_6	0.000↑	0.000↑	0.000↑	0.008↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↓
f_7	0.000↑	0.000↑	0.000↑	0.788↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑
f_{10}	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑
f_{11}	0.000↑	0.888↓	0.000↑	0.000↑	0.000↑	0.010↑	0.000↑	0.010↑	0.000↑	0.000↑
f_{14}	0.000↑	0.033↓	0.000↑	0.000↑	0.790↓	0.000↑	0.000↑	0.000↓	0.000↑	0.000↑
f_{15}	0.012↑	0.000↓	0.010↓	0.000↓	0.121↑	0.000↓	0.000↑	0.000↓	0.000↓	0.000↓
f_{17}	0.000↑	0.000↓	0.000↓	0.000↑	0.000↓	0.021↓	0.000↑	0.000↓	0.007↑	0.000↓
f_{19}	0.000↑	0.000↓	0.040↓	0.000↑	0.005↓	0.000↑	0.000↑	0.000↓	0.407↑	0.000↑
f_{22}	0.000↑	0.000↓	0.000↑	0.083↑	0.062↓	0.504↑	0.000↑	0.000↓	0.000↓	0.000↑
f_{23}	0.000↑	0.154↓	0.000↑	0.000↓	0.061↑	0.000↓	0.000↑	0.327↑	0.000↓	0.002↓
f_{27}	0.000↑	0.008↑	0.000↑	0.000↑	0.000↑	0.000↓	0.000↑	0.000↑	0.000↑	0.000↑
f_{28}	0.000↑	0.000↑	0.000↑	0.000↑	0.000↑	0.036↑	0.161↑	0.000↑	0.000↑	0.000↑
Better	16	8	11	10	10	8	15	8	9	10
Same	0	3	1	3	4	3	1	3	3	1
Worse	0	5	4	3	2	5	0	5	4	5

6 结 论

BBPSO 是在对 SPSO 中粒子的运行轨迹进行研究的基础上提出的一种应用高斯分布进行位置更新的粒子群算法,其中粒子位置的进化形式更加清晰明了,局部搜索能力较强,搜索精度较高.通过与 SPSO 对比分析可知,BBPSO 在单个粒子的搜索位置的多样性上有所丧失.本文对 BBPSO 进行了改进,对高斯分布的位置参数进行自适应的调整,使粒子的搜索中心多样化.并且在进化的前期侧重于发展粒子的多样性,搜索更多的空间,减小陷入局部极值的可能,在后期则侧重精细的搜索,以提高解的精度.为了验证算法的有效性,将其与另几种发表在权威期刊的改进粒子群算法进行比较,对 CEC2013 标准函数测试的结果表明,本文提出的改进算法 Aμ-BB的综合性能是令人满意的.

本文主要是针对基于高斯分布的 BBPSO 进行讨论,对于基于其他钟形分布的 BBPSO 的变形,本文中对单个粒子搜索位置多样性和算法的全局收敛性的分析仍然适用,做出的改进思想同样可以扩展到这些算法中.另外,在本文对 BBPSO 进行改进的过程中,只是对高斯分布的均值 μ 做了讨论,将其进行了分布位置多样性的拓展,并未对标准差 δ 进行讨论.而 δ 取值的大小直接影响到高斯分布的形状,扩大 δ 可以增加粒子的多样性,减小 δ 可以加速粒子的收敛.

除了对算法本身的改进工作外,本文对 PSO 算法的分析角度,也为对 PSO 算法的改进工作提供了新的思路,粒子位置变量的概率分布情况决定了群

体的搜索能力，改变其概率密度函数的形式，便可以使算法具有不同的特性.

致　谢　特别感谢审稿专家和编辑对本文的认真审阅，正是由于他们在认真仔细审阅的基础上所给出的宝贵意见和建议，才使得本文的工作得到改进和提高！感谢新加坡南洋理工大学 P N Suganthan 教授提供 CLPSO，FIPS，DMS-PSO 和 FDR-PSO 算法的源程序！感谢中国矿业大学张勇教授提供 AB-PSO 算法的源程序！由于他们的无私奉献才使得我们的工作更加顺利.

参　考　文　献

[1] Wang Ling, Liu Bo. Particle Swarm Optimization and Scheduling Algorithms. Beijing: Tsinghua University Press, 2008(in Chinese)
(王凌，刘波. 微粒群优化与调度算法. 北京：清华大学出版社，2008)

[2] Shi Y, Eberhart R. A modified particle swarm optimizer//Proceedings of the IEEE Conference on Evolutionary Computation. Anchorage, USA, 1998: 69-73

[3] Kennedy J, Mendes R. Population structure and particle swarm performance//Proceedings of the Congress on Evolutionary Computation. Honolulu, USA, 2002: 1671-1676

[4] Mendes R, Kennedy J, Neves J. The fully informed particle swarm: Simpler, maybe better. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 204-210

[5] Liang J J, Qin A K, Suganthan P N, Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation, 2006, 10(3): 281-295

[6] Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 240-255

[7] Zhan Z H, Zhang J, Li Y, Chung H S H. Adaptive particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2009, 39(6): 1362-1381

[8] Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer//Proceedings of the IEEE Swarm Intelligence Symposium. Pasadena, USA, 2005: 124-129

[9] Xue Ming-Zhi, Zuo Xiu-Hui, Zhong Wei-Cai, Liu Jing. Orthogonal particle swarm optimization. Journal of System Simulation, 2005, 17(12): 2908-2911(in Chinese)
(薛明志，作秀会，钟伟才，刘静. 正交微粒群算法. 系统仿真学报，2005，17(12)：2908-2911)

[10] Zhan Z H, Zhang J, Li Y, Shi Y H. Orthogonal learning particle swarm optimization. IEEE Transactions on Evolutionary Computation, 2011, 15(6): 832-847

[11] Chen W N, Zhang J, Lin Y, Chen N, et al. Particle swarm optimization with an aging leader and challengers. IEEE Transactions on Evolutionary Computation, 2013, 17(2): 241-258

[12] Kennedy J. Bare bones particle swarms//Proceedings of the IEEE Swarm Intelligence Symposium. Indianapolis, IN, USA, 2003: 80-87

[13] Campos M, Krohling R A. Hierarchical bare bones particle swarm for solving constrained optimization problems//Proceedings of the IEEE Congress on Evolutionary Computation. Cancun, Mexico, 2013: 805-812

[14] Campos M, Krohling R A. Bare bones particle swarm with scale mixture of Gaussians for dynamic constrained optimization //Proceedings of the IEEE Congress on Evolutionary Computation. Beijing, China, 2014: 202-209

[15] Zhang Y, Gong D, Hu Y, Zhang W. Feature selection algorithm based on bare bones particle swarm optimization. Neurocomputing, 2015, 148: 150-157

[16] Koshti A, Arya L D, Choube S C. Voltage stability constrained distributed generation planning using modified bare bones particle swarm optimization. Journal of the Institution of Engineers(India) Series B(Electrical, Electronics & Telecommunication and Computer Engineering), 2013, 94(2): 123-133

[17] Krohling R A, Mendel E. Bare bones particle swarm optimization with Gaussian or Cauchy jumps//Proceedings of the IEEE Congress on Evolutionary Computation. Trondheim, Norway, 2009: 3285-3291

[18] Blackwell T. A study of collapse in bare bones particle swarm optimization. IEEE Transactions on Evolutionary Computation, 2012, 16(3): 354-372

[19] Majid al-Rifaie M, Blackwell T. Bare bones particle swarms with jumps//Proceedings of the International Conference on Swarm Intelligence. Brussels, Belgium, 2012: 49-60

[20] Zhang Y, Gong D W, Sun X Y, Geng N. Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis. Soft Computing, 2014, 18(7): 1337-1352

[21] Campos M, Krohling R A, Enriquez I. Bare bones particle swarm optimization with scale matrix adaptation. IEEE Transactions on Cybernetics, 2014, 44(9): 1567-1578

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

[23] Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, 1994, 24(4): 656-667

[24] Kamyab S, Eftekhari M. Using a self-adaptive neighborhood scheme with crowding replacement memory in genetic algorithm for multimodal optimization. Swarm and Evolutionary Computation, 2013, 12(5): 1-17

[25] e Oliveira Junior H A, Petraglia A. Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing. Applied Soft Computing, 2013, 13(11): 4349-4357

[26] Zhao X. Simulated annealing algorithm with adaptive neighborhood. Applied Soft Computing, 2011, 11(2): 1827-1836

[27] Brest J, Greiner S, Boskovic B, Mernik M, et al. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation, 2006, 10(6): 646-657

[28] Das S, Mandal A, Mukherjee R. An adaptive differential evolution algorithm for global optimization in dynamic environments. IEEE Transactions on Cybernetics, 2014, 44(6): 966-978

[29] Hu M, Wu T, Weir J D. An adaptive particle swarm optimization with multiple adaptive methods. IEEE Transactions on Evolutionary Computation, 2013, 17(5): 705-720

[30] Li C, Yang S, Nguyen T T. A self-learning particle swarm optimizer for global optimization problems. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2012, 42(3): 627-646

[31] Juang Y T, Tung S L, Chiu H C. Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions. Information Sciences, 2011, 181(20): 4539-4549

[32] Solis F J, Wets R J B. Minimization by random search techniques. Mathematics of Operations Research, 1981, 6(1): 19-30

[33] Liang J J, Qu B Y, Suganthan P N, Hernandez-Diaz A G. Problem definitions and evaluation criteria for the CEC2013 special session on real-parameter optimization. Zhengzhou University, Zhengzhou China and Nanyang Technological University, Singapore, Technical Report, 201211, 2013

[34] Shi Y, Eberhart R C. Empirical study of particle swarm optimization//Proceedings of the Congress on Evolutionary Computation. Washington, USA, 1999: 1945-1950

附表 1. 文中未列出的 CEC2013 函数，$D=30$ 时的运行结果

		GPSO	CLPSO	APSO	DMS-PSO	OLPSO	ALC-PSO	BBPSO	BBJ2	ABPSO	SMA-BB	Aμ-BB
f_1	*Mean*	8.6e−13	2.2e−13	**2.0e−13**	2.2e−13	2.2e−13	3.4e−13	2.5e−13	8.6e−13	5.9e−13	2.2e−13	2.2e−13
	SD	1.2e−13	0.0e+00	7.1e−14	0.0e+00	0.0e+00	1.1e−13	1.0e−13	2.0e−13	1.1e−13	6.5e−14	0.0e+00
f_8	*Mean*	2.0e+01	2.0e+01	2.0e+01	2.0e+01	2.1e+01	2.0e+01	2.0e+01	2.0e+01	2.0e+01	2.0e+01	2.0e+01
	SD	9.3e−02	3.2e−02	6.3e−02	5.4e−02	6.6e−02	5.7e−02	2.1e−02	9.9e−02	6.3e−02	3.5e−02	3.4e−02
f_9	*Mean*	3.7e+01	**2.6e+01**	2.7e+01	3.5e+01	3.4e+01	**2.6e+01**	3.8e+01	3.7e+01	**2.6e+01**	3.3e+01	3.3e+01
	SD	2.7e+00	1.4e+00	5.0e+00	4.0e+00	2.2e+00	3.7e+00	1.0e+00	3.6e+00	5.7e+00	1.8e+00	1.8e+00
f_{12}	*Mean*	1.1e+02	1.1e+02	**1.0e+02**	1.8e+02	1.3e+02	1.5e+02	2.1e+02	6.0e+02	1.5e+02	1.3e+02	**1.0e+02**
	SD	6.6e+01	1.5e+01	3.0e+01	4.7e+00	1.4e+01	8.5e+01	1.2e+01	1.4e+02	5.8e+01	2.5e+01	1.3e+01
f_{13}	*Mean*	2.2e+02	**1.6e+02**	1.9e+02	2.2e+02	1.9e+02	2.0e+02	2.0e+02	5.6e+02	1.9e+02	1.8e+02	1.7e+02
	SD	2.6e+01	1.9e+00	3.0e+01	1.3e+01	2.5e+01	4.1e+01	1.6e+01	1.3e+02	3.6e+01	2.4e+01	7.6+01
f_{16}	*Mean*	2.7e+00	2.4e+00	2.2e+00	1.7e+00	2.3e+00	1.6e+00	2.5e+00	1.6e+00	**1.5e+00**	1.7e+00	**1.5e+00**
	SD	4.5e−01	2.1e−01	2.7e−01	1.3e−01	4.2e−01	2.5e−01	2.6e−01	5.33e−01	6.2e−01	2.1e−01	1.4e−01
f_{18}	*Mean*	2.4e+02	2.0e+02	2.1e+02	1.9e+02	2.2e+02	**1.5e+02**	2.3e+02	2.0e+02	**1.5e+02**	1.9e+02	1.6e+02
	SD	2.9e+01	1.9e+01	1.3e+01	1.6e+01	1.4e+01	4.6e+01	1.5e+01	1.5e+01	3.3e+01	6.8e+01	1.0e+01
f_{20}	*Mean*	1.4e+01	1.3e+01	1.2e+01	**1.1e+01**	1.5e+01	1.2e+01	1.3e+01	1.4e+01	1.5e+01	1.4e+01	1.2e+01
	SD	1.2e+00	6.2e−01	9.2e−01	4.1e−01	2.1e−01	1.3e+00	2.7e−01	2.0e−01	0.0e+00	4.9e−01	3.8e−01
f_{21}	*Mean*	3.2e+02	2.8e+02	2.9e+02	2.8e+02	**2.1e+02**	3.1e+02	3.4e+02	3.3e+02	2.8e+02	2.8e+02	2.6e+02
	SD	1.1e+02	3.0e+01	1.0e+02	7.3e+01	3.1e+01	1.1e+02	9.8e+01	9.9e+01	7.3e+01	4.2e+01	5.1e+01
f_{24}	*Mean*	2.6e+02	2.6e+02	2.8e+02	2.5e+02	2.7e+02	2.5e+02	2.8e+02	3.0e+02	2.5e+02	2.4e+02	**2.1e+02**
	SD	8.5e+00	8.1e+00	1.3e+01	3.8e+00	1.1e+01	1.5e+01	4.7e+00	1.2e+01	9.1e+00	5.2e+00	7.7e+00
f_{25}	*Mean*	3.0e+02	2.9e+02	2.9e+02	2.6e+02	3.0e+02	3.0e+02	2.8e+02	3.2e+02	2.7e+02	2.9e+02	**2.5e+02**
	SD	1.0e+01	2.9e+00	1.4e+01	1.8e+01	7.8e+00	7.8e+00	3.4e+00	9.7e+00	1.0e+01	3.0e+00	1.1e+01
f_{26}	*Mean*	3.2e+02	2.1e+02	3.3e+02	2.5e+02	2.2e+02	3.3e+02	2.2e+02	3.6e+02	3.3e+02	3.3e+02	**2.0e+02**
	SD	6.3e+01	2.8e−01	7.1e+01	6.5e+01	5.5e+01	7.1e+01	7.4e+00	8.5e+01	2.5e+01	3.3e+01	4.9e−01

WANG Dong-Feng, born in 1971, Ph. D., professor. His main research interests include swarm intelligence-based optimization and intelligent control.

MENG Li, born in 1985, Ph. D. candidate. Her main research interests include swarm intelligence-based optimization and its application.

ZHAO Wen-Jie, born in 1968, Ph. D., associate professor. His main research interests include optimization theory, optimal control and their industrial application.

Background

The work is jointly supported by the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20120036120013), the Natural Science Foundation of Hebei Province (Grant No. F2014502059) and the Fundamental Research Funds for the Central Universities (Grant No. 2014MS139). The purpose of this paper is to improve the performance of BBPSO algorithm, thus to provide an optimization algorithm which is suitable for engineering application for the optimization issues in these projects.

Bare bones particle swarm optimization algorithm (BBPSO) is proposed by Kennedy in 2003. It is based on particle swarm optimizer (PSO) which is proposed by Kennedy and Eberhart in 1995. PSO is easy to implement, but there are parameters to be adjusted according to different kinds of objective functions. Furthermore, how the particle works is still mysterious and it is very difficult to grasp in one's mind how the particle changes its position. Based on the previous study on particle trajectory in PSO, Kennedy carries on a further study. He suggests that the whole velocity part of the particle swarm formula can be dropped. The new position of particle is a sampling point form Gaussian distribution. This is the update strategy of particle position in BBPSO.

BBPSO is a simpler algorithm with no parameter to be adjusted; moreover, BBPSO has shown potential for solving single objective optimization problems over continuous search space. However, it also suffers from premature convergence. The methods which are employed to improve the performance of PSO can also implement on BBPSO, such as changing the population topological structure, combining with other optimization algorithms and putting variation disturbance onto particles.

In this paper, we use a new point of view to observe BBPSO and PSO. As we know, there are random numbers in the update formula of particle position in PSO. So we can regard the particle position as a random variable, which is the same in BBPSO. It is precisely because the statistical regularities of position random variables in different algorithms are different, the algorithms can exhibit different properties. The probability density function of the position random variable shows the search mode of a particle: the search scope and probability to search a certain region. To the best of our knowledge, there is no analysis from this aspect to explain how the particle works. Viewed at this angle, the working mode of a particle in PSO is clearer, so as the effect of the velocity item: not to change the shape of the probability density function, but to shift it. In BBPSO, new position of a particle is a sampling point from Gauss distribution and the velocity item is removed. The centre of Gauss distribution is restricted. The centre can be shifted randomly in an appropriate range to compensate for the influence of dropping the velocity item.