DOI: 10.7641/CTA.2017.60905

目标空间聚类的差分头脑风暴优化算法

吴亚丽†, 付玉龙, 王鑫睿, 刘 庆

(西安理工大学 自动化与信息工程学院, 陕西 西安 710048;

陕西省复杂系统控制与智能信息处理重点实验室, 陕西 西安 710048)

摘要: 作为一种新型的群体智能优化算法, 头脑风暴优化(brain storm optimization, BSO)算法一经提出便引起了众多研究者的关注. 本文在对原始头脑风暴算法的聚类操作和变异操作改进的基础上, 提出了基于目标空间聚类的差分头脑风暴(difference brain storm optimization based on clustering in objective space, DBSO–OS)算法. 算法通过对目标空间的聚类替代对决策空间的聚类, 减小了算法的运算复杂度; 采用差分变异代替高斯变异来增加种群的多样性. 多个测试函数的仿真结果表明, 目标空间聚类的差分头脑风暴算法不仅提高了算法的寻优速度, 而且提高了算法的寻优精度. 文中进一步分析了参数对算法性能的影响, 设计了最佳参数选择方案, 并用于对实际热电联供经济调度问题的求解, 验证了算法的实用性.

关键词: 头脑风暴算法; 聚类; 差分变异; 目标空间

中图分类号: TP18 **文献标识码**: A

Difference brain storm optimization algorithm based on clustering in objective space

WU Ya-li†, FU Yu-long, WANG Xin-rui, LIU Qing

(School of Automation and Information Engineering, Xi'an University of Technology, Xi'an Shannxi 710048, China;
Shaanxi Province Key Laboratory of Complex System Control and Intelligent Information Processing, Xi'an Shannxi 710048, China)

Abstract: As a new kind of swarm intelligence optimization algorithm, brain storm optimization (BSO) has paid more attention of more researchers in different fields. Based on the cluster operation and mutation of original BSO, a novel BSO algorithm named difference brain storm optimization based on clustering in objective space (DBSO–OS) is proposed in this paper to improve the performance of the original BSO algorithm. The clustering operation is designed in objective space which can decrease the computation complexity comparing with clustering in decision space in the proposed algorithm. The difference mutation operation is adopted to increase the diversity of the population. The simulation results of many benchmark functions of different dimensions demonstrate that the proposed algorithm can not only improve the time performance but also the precision. Moreover, the suitable parameter selection strategy is provided on the basis of the parameter analysis of the proposed algorithm. And the combined heat and power economic dispatch (CHPED) are implemented to evaluate the effectiveness of the proposed algorithm.

Key words: brain storm optimization (BSO); cluster; difference mutation; objective space

1 引言(Introduction)

群体智能是一个复杂的、高度交互的过程. 基于群体智能行为的群智能优化算法, 如蚁群算法、粒子群算法等, 虽然都是随机搜索方法, 但均具有灵活应对内部和外部变化的能力, 即当自身的某些个体失败或者搜索条件多变时, 能够自组织逃离并发现更优的位置[1]. 正是由于这些特有的优势, 群体智能算法目前已成功解决了众多的实际问题, 也同时引发了各个领域研究者的广泛关注.

头脑风暴优化算法(brain storm optimization, BSO)是史玉回[2]于2011年在第2次群体智能国际会议上提出的一种新型的群体智能优化算法. 该算法是受20世纪30年代美国创造学家A·F·奥斯本的人类社会头脑风暴过程的行为启发而提出, 重点模拟一种人类创造性解决问题的思路及过程. 该算法一经提出便被认为是一种很有潜力的方法.

收稿日期: 2016−11−29; 录用日期: 2017−09−14.

†通信作者. E-mail: yliwu@xaut.edu.cn; Tel.: +86 13119121204.

本文责任编委: 胡德文.

国家自然科学基金青年基金项目(61503299, 61502385)资助.

Supported by National Youth Foundation of China (61503299, 61502385).

原始的BSO算法将头脑风暴的寻优过程抽象为“聚”和“散”两个过程. 在初始化种群后, 通过聚类的方式将种群分类, 通过类内、类间及类中心的各类交互操作来实现个体的更新, 完成“聚”的操作; 在此基础上, 通过变异操作来实现一定范围的“散”, 以增强群体的多样性. 根据“聚”“散”实施的方式不同, 不同的研究者设计了不同版本的BSO 算法. Shi等人[2]的原始BSO采用k-means聚类方式和高斯变异实现群体中个体的更新; 在文献[3]中作者提出了基于k-medians聚类的头脑风暴优化算法, 以通过k-medians聚类代替k-means聚类来提高算法的时间效率; 杨玉婷等人[4]则提出了基于差分步长的头脑风暴优化算法; 张军等人[5]用一种简单的聚类方式(simple group method, SGM)代替k-means聚类, 并用差分变异代替原BSO中的高斯变异, 提出了一种改进的头脑风暴算法(modified brain storm optimization, MBSO), 改进算法缩短了运行时间, 同时大幅度提高了算法的寻优精度; 杨玉婷等[6]在分析头脑风暴过程的基础上, 提出了一种基于讨论机制的头脑风暴算法 (discussion mechanism based brain storm optimization, DMBSO), 提高了原BSO的优化效果和算法稳定性; 文献[7]则尝试用BSO解决多目标问题(multi-objective brain storm optimization, MOBSO), 用非支配排序来对由非劣解构成归档集进行更新, 表明了BSO在解决多目标问题上的有效性; 借鉴多目标优化的理念, Shi[8]提出通过对目标空间的分类(即精英类和普通类)来代替复杂的决策空间聚类方法, 提出了基于目标空间的头脑风暴算法(BSO–OS), 极大地加快了算法的运行速度, 降低了算法的复杂度.

从上述的研究进展可以看出, 头脑风暴理论及算法改进研究的核心, 就是“聚”“散”行为的实施方法. 文献[8]虽然提出了目标空间聚类的基本理念, 但聚类操作较为简单, 即为简单的分类; 此外, 高斯变异的局限性对BSO算法的收敛性能也有一定的劣势. 为此, 本文在以前的研究基础上, 通过对“聚”“散”行为的分析, 提出了通用的基于目标空间聚类的差分头脑风暴算法(DBSO–OS), 并通过测试函数对算法的性能进行分析. 在此基础上, 对算法中的参数, 特别是聚类个数对算法性能的影响程度进行分析, 给出了基于目标空间聚类的差分BSO算法的最佳参数选择方案, 并与多个算法进行对比分析, 结果表明算法的高效性.

2 头脑风暴优化算法的基本原理(The principle of brain storm optimization)

2.1 头脑风暴法(Brainstorming)

头脑风暴优化算法来源于头脑风暴法, 又称为智力激励法, 是由美国创造学家A·F·奥斯本于1939年首次提出, 1953年正式发表的一种激发性的思维方法, 其具体步骤如下表示:

头脑风暴过程:

Step 1 尽可能多的不同背景的人聚在一起;

Step 2 根据奥斯本头脑风暴过程中遵循的规则生成尽可能多的想法;

Step 3 有3–5个人作为问题的推动者提出解决问题的更好想法;

Step 4 利用Step 3提出的想法比其他想法更高概率作为线索, 根据奥斯本头脑风暴过程中遵循的规则产生更多的想法;

Step 5 让问题持有者挑出Step 4产生的几个好的想法;

Step 6 随机选中的对象作为线索根据奥斯本头脑风暴过程中遵循的规则产生更多想法;

Step 7 让问题持有者挑出几个好的想法;

Step 8 希望通过考虑或者合并已经产生的想法得到一个好的解. 否则, 进行下一次头脑风暴过程.

上述步骤中所提到的奥斯本头脑风暴过程中遵循的规则是指“庭外判决; 各抒己见; 取长补短; 追求数量”. 从头脑风暴的过程来看, 头脑风暴是一个集思广益的过程, 在这个过程中有3个比较重要的操作, 即:

1) 生成, 指用规则生成尽可能多的想法;

2) 挑选, 指从众多想法中挑出好的想法;

3) 探索, 即通过考虑或合并产生好的解. 因此头脑风暴过程通过不断“生成”新想法, 不断“选择”好想法, 和不断“探索”优想法实现思维的演化和优化.

2.2 头脑风暴优化算法(Brain storm optimization)

BSO是基于头脑风暴这种创造性解决问题的思想, 由shi[2]在2011年第2次国际群体智能会议上提出的. 在算法中, 每一个个体都代表一个潜在的问题的解, 通过个体的演化和融合进行个体的更新, 这一过程与人类头脑风暴的过程相似. BSO算法的实现过程相对比较简单, 其过程如下所示:

头脑风暴优化算法过程:

Step 1 种群初始化;

Step 2 评价个体, 聚类;

Step 3 产生新个体;

Step 4 更新种群和聚类中心;

Step 5 若到最大达迭代次数则输出最优个体, 否则转Step 2.

3 目标空间聚类的差分头脑风暴算法(Difference brain storm optimization algorithm based on clustering in objective space)

3.1 差分头脑风暴算法(Difference brain storm optimization algorithm)

传统的BSO采用高斯随机值来实现当前解的探索. 即新一代个体表示如式(1):

$$X_{\text{new}}^{\text{d}} = X_{\text{selected}}^{\text{d}} + \xi(\mu, \sigma), \tag{1}$$

其中: $X^{\mathrm{d}}_{\mathrm{selected}}$表示选择个体的第$d$维, $X^{\mathrm{d}}_{\mathrm{new}}$是产生的新个体的第$d$维, $\xi(\mu,\sigma)$是以μ为均值、以σ为方差的高斯随机函数, 是一个衡量高斯随机值权重的系数, 表示为

$$\xi = \mathrm{logsig}((\mathrm{max\_iteration}/2 - \mathrm{iteration})/K) * \mathrm{rand}, \tag{2}$$

其中: logsig()是一个S型对数传递函数, max_iteration为最大迭代次数, iteration为当前迭代次数, K用来改变logsig()的斜率, 可调节算法由全局搜索到局部搜索的速度, rand产生$(0,1)$之间的随机值.

高斯变异是一种常用的变异方式, 在得到系数后, 与高斯随机函数相乘便得到了变异量. logsig()函数值域为(0, 1), 显然$\xi \in (0,1)$. 而高斯分布服从3σ原则, 即当μ和σ一定时, 高斯函数产生的绝大多数值在$(\mu-3\sigma, \mu+3\sigma)$之间, 如图1所示. 因此采用高斯变异, 变异量的范围在一个固定的范围之内. 图2为$\mu=0$, $\sigma=1$, max_iteration= 1000时变异量取值, 可见变异量的范围为$(-3,3)$, 甚至在$(-2.5,2.5)$之间.

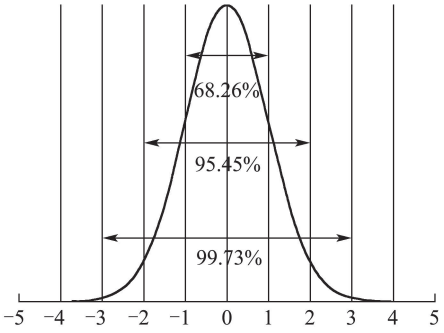

图 1 高斯分布图

Fig. 1 Gaussian distribution

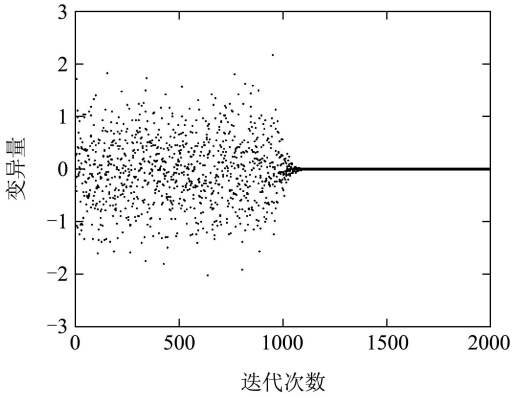

图 2 $\mu=0, \sigma=1$时的变异量

Fig. 2 Variance in $\mu=0, \sigma=1$

在种群内进行最优解的搜索时, 初始在定义空间内产生均匀的一组解, 应进行全局搜索; 到后期由于解质量的整体上升, 应重点进行局部寻优, 因此初始的变异量应比较大, 后期则较小. 高斯变异正是符合这一特点, 但由其产生的变异量范围基本固定, 没有充分利用当前种群中个体的信息. 此外, 高斯变异运行过程中需要完成S型对数传递函数、高斯分布函数、随机函数和四则混合运算, 其运算复杂度也相对较高.

为此, 本文采用差分变异来替代原BSO中的高斯变异来实现BSO解的探索作用, 具体如式(3)所示:

$$x^{\mathrm{d}}_{\mathrm{new}} = \begin{cases} \mathrm{rand}(L_{\mathrm{d}}, H_{\mathrm{d}}), & \mathrm{rand}() < P_{\mathrm{r}}, \\ x^{\mathrm{d}}_{\mathrm{selected}} + \mathrm{rand}() * (x_1 - x_2), & \text{其他}, \end{cases} \tag{3}$$

式中: $x^{\mathrm{d}}_{\mathrm{selected}}$是选择的个体的第$d$维, $x^{\mathrm{d}}_{\mathrm{new}}$是生成个体的第$d$维, L_{d}, H_{d}为第d维的上下界, x_1, x_2为在当代全局中选择的两个不同的个体. 为了保持种群的多样性, 防止陷入局部最优, 设置开放性概率P_{r}. 文献[5]表明参数设置在[0.001, 0.01]时算法性能较好, 本文取P_{r} =0.05.

从式(3)可以看出, 与高斯变异相比, 差分变异中仅有随机函数和四则混合运算, 运算量会显著减小, 其运行速度会显著提高. 而且变异量的产生是基于当代种群内其他个体而来, 可根据种群内个体的离散程度来自适应调节, 能够更好地与种群内其他个体共享信息, 其搜索效率会更高. 因此, 差分变异能够在搜索过程中更好地平衡局部搜索和全局搜索, 有效地提高算法性能.

3.2 目标空间聚类的差分头脑风暴优化算法(Difference brain storm optimization algorithm based on clustering in objective space)

聚类是BSO的特点, 也是众多BSO研究者一直以来的研究热点. 在众多的研究中, 主要是基于聚类方式与聚类算法的研究, 如初始BSO中的聚类算法为k-means聚类, 而对于聚类空间的选择及聚类个数的分析却鲜有成果. 在文[7]等在采用BSO解决多目标优化问题时, 首次采用对目标空间聚类的思想, 以减小多目标优化非劣比较的复杂程度, 文[8]提出了基于目标空间聚类的BSO思路, 并通过适应度排序将解空间分为精英解和普通解, 省去了聚类的极大工作量. 但聚类操作在目标空间的可行性, 聚类算法的有效性及聚类个数的选择等问题有待进一步的研究. 因此, 本节将聚类算法用于单目标优化问题的目标空间, 通过个体在目标空间的欧式距离来对个体进行分类, 通过聚类个数的改变来分析基于目标空间聚类的差分头脑风暴优化算法的性能. k-means聚类过程如下:

k-means聚类过程:

Step 1　从当前n个个体中选出m个作为聚类中心;

Step 2　计算当前个体到每个聚类中心的欧氏距离, 并将其聚到与其欧氏距离最小的类中;

Step 3　计算每个类中所有点的坐标平均值, 并将

这些平均值作为新的聚类中心;

Step 4 重复Step 2和Step 3, 直到聚类中心不再进行大范围移动或者聚类次数达到要求为止.

基于目标空间欧式距离聚类的过程如下:

基于目标空间聚类过程:

Step 1 从当前n个个体中选出m个作为聚类中心;

Step 2 对个体进行评价, 计算当前个体的适应度值到每个聚类中心的适应度值的欧氏距离;

Step 3 将这个个体分配到与其欧氏距离最小的一个聚类中心的类中;

Step 4 重复Step 2和Step 3, 直到所有个体聚类完成为止.

图3则综合第3.1节和第3.2节内容, 给出了基于目标空间聚类的差分头脑风暴优化算法流程.

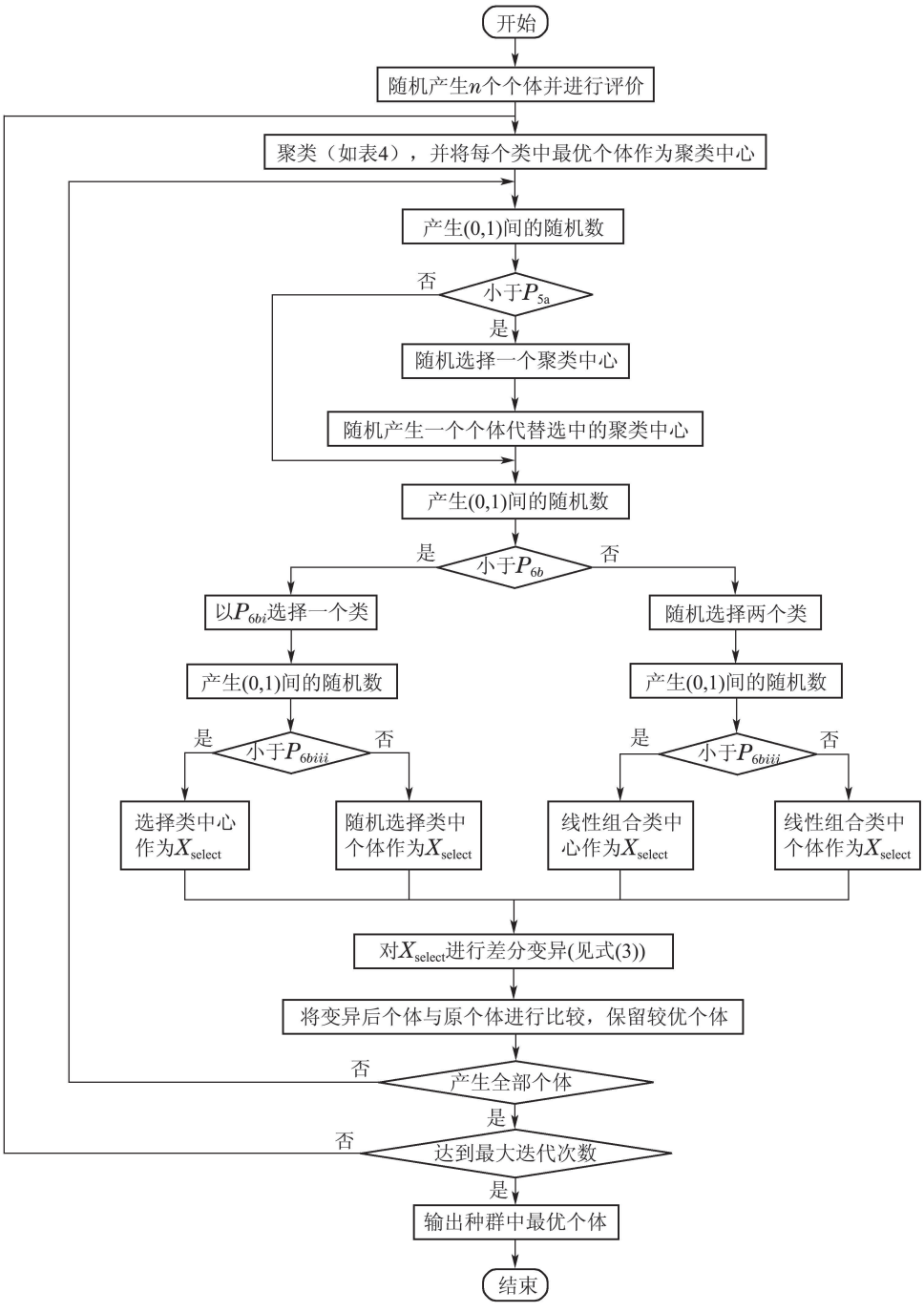

图 3 DEBSO–OS算法流程

Fig. 3 The process of DEBSO–OS

4 仿真结果及参数分析(The simulation results and the parameter analysis)

为了对DEBSO–OS的性能, 本文对常用的4个单目标测试函数在不同维数下(即低维5, 中维50和高维500)对算法的性能进行分析和测试. 测试函数见表1.

在表1中, $f_1(x)$ 为最简单的单峰值函数 Sphere, 易于找到最优解, 在$(0,0,\cdots,0)$取最小值0; $f_2(x)$为自变量之间独立的多模态函数Rastrigin, 在解空间内有$10N$个局部极值点, $(0,0,\cdots,0)$有最小值0; $f_3(x)$为多模态函数Minima, 其局部极值点随维数成指数增加, 易陷入局部最优, 最小值为-78.3323; $f_4(x)$为多峰函数Griewank, 局部极值点随维数不断增加, 但由于乘积项的影响随维数不断降低, 所以高维较易于优化.

为详细分析算法的性能, 本节的仿真对比分为两部分, 一是聚类个数对算法性能的影响, 二是本文算法与其它BSO算法的对比研究.

表 1 测试函数

Table 1 Benchmark function

测试函数	搜索空间	最小值
$f_1(x)=\sum_{i=1}^{N}x_i^2$	$(-100,100)^{\mathrm{T}}$	0
$f_2(x)=\sum_{i=1}^{N}(x_i^2-10\cos(2\pi x_i)+10)$	$(-5.12,5.12)^{\mathrm{T}}$	0
$f_3(x)=\frac{1}{N}\sum_{i=1}^{N}(x_i^4-16x_i^2+5x_i)$	$(-5,5)^{\mathrm{T}}$	-78.3323
$f_4(x)=\frac{1}{4000}\sum_{i=1}^{N}(x_i^2)+\prod_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}})+1$	$(-600,600)^{\mathrm{T}}$	0

4.1 聚类个数对DEBSO–OS算法性能的影响(The impact of clustering number for DEBSO–OS)

为方便比较, 在本部分的仿真中的参数设置统一为表2所示.

表 2 DEBSO–OS仿真参数

Table 2 The parameters for DEBSO–OS

N	N_{c}	T	$P_{5\mathrm{a}}$	$P_{6\mathrm{b}}$	$P_{6\mathrm{biii}}$	$P_{6\mathrm{c}}$	P_{r}
80	$2-16$	1000	0.2	0.8	0.4	0.5	0.005

其中N为种群规模, T为最大迭代次数, N_{c}为聚类个数. 考虑到算法的随机性, 对于每一个测试函数每一维的结果, 均运行30次得到平均值, 表7–10给出了各函数在各维度和不同聚类个数下的寻优精度和寻优时间的值. 其中下划线数字表示最优值, 即最高精度和最短寻优时间.

从表 3 中可以看出, 对于Sphere函数, 在5 dim, 50 dim, 200 dim中, 聚2类的比其他聚类个数的寻优精度都要好; 在50 dim时聚3类的寻优时间最短, 但是在5 dim, 200 dim时, 聚2类的寻优时间最短, 可以看出DEBSO–OS算法在Sphere函数中聚2类时的性能是比较好. 聚类个数对寻优结果影响很大, 甚至处于不同的数量级.

从表4可看出, 对于Rastrigin函数, 在5 dim时, 聚类的个数和寻优精度没有关系, 但在50 dim, 200 dim中, 聚2类的比其他聚类个数的寻优精度都要好; 在5 dim, 50 dim, 200 dim时, 聚2类的寻优时间最短.

从表5中可以看出, 对于Greiwank函数, 在5 dim时, 聚类的个数为12时, 寻优精度较好, 在50 dim时, 聚类的个数为4时, 寻优精度较好, 但在200 dim中, 聚2类的比其他聚类个数的寻优精度都要好; 在5 dim时, 聚3类的寻优时间较好, 但在50 dim, 200 dim时, 聚2类的寻优时间比其他聚类个数的寻优时间要短.

从表6中可以看出, 对于Minima函数, 在5 dim, 50 dim时, 聚类的个数和寻优精度没有关系, 在200 dim中, 聚2类的比其他聚类个数的寻优精度都要好; 在5 dim, 50 dim, 200 dim时, 聚2类的寻优时间最短.

由表3–6中对各个聚类个数从2–16变化时算法的寻优结果的统计, 可以发现, 对于绝大多数的测试函数, 在聚类个数为2时算法不仅有着最好的寻优精度, 而且有最好的时间性能. 从直观上来看, 就本文采用的测试函数, 虽然表达形式不同, 决策空间的特点也各异, 但都属于单目标优化问题. 本文算法通过对目标空间聚类来实现头脑风暴过程, 实际上是在一维空间聚类, 对一维空间而言, 分优与劣两类就可以将目标的特征进行完全的区分, 聚类个数的增多不仅使得算法复杂度增加, 而且使得较差和较优个体的更新过程更慢, 造成整个种群向群体最优学习的速度变慢. 因此, 在一维空间上选择2类, 不仅能简化算法的复杂程度, 而且种群中较好的类提供了寻优方向, 较差的类保证了算法的多样性.

当聚类个数为2时, 聚类其实是一种相对简单的过程, 只需要计算每个点到每个聚类中心的距离, 但这种方式并不是最合理的聚类方式, 因为有的个体总体来说较第1类近但却距其聚类中心稍远. 为

此在实现过程中可以采用另外的一种简单聚类方式: 根据适应度值进行优劣排序, 然后在优劣差距最大的两个个体之间分开, 分为两个类, 其步骤如下. 简单聚类过程:

Step 1 评价n个个体, 并按优至劣进行排序;

Step 2 计算各相邻个体间的距离Δx_i;

Step 3 在最大距离$\max(\Delta x_i)$处, 将种群分开, 前者记为精英, 后者记为普通.

为了描述方便, 通过这种简单聚类方式的DEBSO–OS算法简记为BSO1.

表 3 Sphere函数在各维度和各聚类个数下的寻优精度和寻优时间

Table 3 The optimization precision and elapsed time of Sphere under each dimension and the clustering number

聚类个数	Sphere					
	5 dim		50 dim		200 dim	
	寻优精度	寻优时间	寻优精度	寻优时间	寻优精度	寻优时间
2	8.81e − 145	3.27936	4.07e − 14	4.69532	0.143518	12.8855
3	3.33e − 119	3.28492	2.46e − 11	4.65862	0.947435	12.9001
4	2.71e − 105	3.29630	1.38e − 09	4.69054	3.91381	12.9155
5	4.53e − 94	3.29564	2.07e − 08	4.78894	9.55552	12.9245
6	6.25e − 87	3.31422	2.05e − 07	4.74248	18.3328	12.9438
7	4.09e − 82	3.31497	8.43e − 07	4.75153	29.9046	12.9593
8	7.11e − 77	3.33196	4.92e − 06	4.75978	42.1880	13.0502
9	5.34e − 73	3.34044	1.37e − 05	4.79061	64.2403	13.0863
10	7.19e − 69	3.42123	4.90e − 05	4.78274	97.4012	13.0283
11	8.97e − 67	3.37583	0.000193	4.79910	134.948	13.0473
12	9.84e − 64	3.37637	0.000469	4.81987	218.887	13.0626
13	1.50e − 61	3.38236	0.001286	4.84280	300.709	13.1382
14	1.94e − 58	3.40445	0.003230	4.86499	412.958	13.1549
15	9.45e − 56	3.41022	0.005556	4.86888	546.412	13.1735
16	8.78e − 54	3.41266	0.011339	4.87371	679.018	13.2014

表 4 Rastrigin函数在各维度和各聚类个数下的寻优精度和寻优时间

Table 4 The optimization precision and elapsed time of Rastrigin under each dimension and the clustering number

聚类个数	Rastrigin					
	5 dim		50 dim		200 dim	
	寻优精度	寻优时间	寻优精度	寻优时间	寻优精度	寻优时间
2	0	3.29038	0.00154	5.05116	6.043e + 03	15.2033
3	0	3.30007	7.47588	5.22628	7.225e + 03	15.2975
4	0	3.31152	25.2375	5.21736	7.749e + 03	15.4725
5	0	3.32114	35.0162	5.24939	8.427e + 03	15.6088
6	0	3.33844	59.1543	5.30026	9.083e + 03	15.6193
7	0	3.35115	72.9152	5.30367	9.909e + 03	15.6828
8	0	3.36687	93.8745	5.40787	9.906e + 03	15.7114
9	0	3.37616	1.080e + 03	5.36591	1.084e + 04	15.6904
10	0	3.39621	1.163e + 03	5.40248	1.099e + 04	15.7453
11	0	3.40896	1.427e + 03	5.41782	1.161e + 04	15.8500
12	0	3.41881	1.670e + 03	5.43931	1.251e + 04	15.9046
13	0	3.44482	1.728e + 03	5.45698	1.322e + 04	15.9106
14	0	3.44820	1.972e + 03	5.48408	1.306e + 04	15.9735
15	0	3.47676	1.938e + 03	5.51378	1.314e + 04	16.0149
16	0	3.47403	1.884e + 03	5.54470	1.407e + 04	16.0253

表 5 Greiwank函数在各维度和各聚类个数下的寻优精度和寻优时间

Table 5 The optimization precision and elapsed time of Greiwank under each dimension and the clustering number

聚类个数	Greiwank					
	5 dim		50 dim		200 dim	
	寻优精度	寻优时间	寻优精度	寻优时间	寻优精度	寻优时间
2	0.014208	4.16371	0.000903	6.12826	0.049103	16.3120
3	0.018969	4.14733	0.001642	6.13457	0.301738	16.4704
4	0.017162	4.18358	0.000247	6.16926	0.698246	16.6633
5	0.013467	4.19773	0.003608	6.29648	1.03027	16.7634
6	0.018724	4.21526	0.000493	6.29046	1.15617	16.8602
7	0.012480	4.25513	0.001233	6.29628	1.28718	16.8937
8	0.014450	4.29888	0.001402	6.38467	1.44222	17.0224
9	0.015599	4.29131	0.002400	6.40441	1.62008	17.2200
10	0.015272	4.32575	0.001048	6.45704	1.87846	17.3133
11	0.013219	4.34791	0.001905	6.50996	2.32496	17.4060
12	0.011414	4.37580	0.002673	6.55663	2.88933	17.5324
13	0.019872	4.40034	0.002646	6.70290	3.77841	17.6375
14	0.013793	4.42272	0.005559	6.64157	4.85809	17.7310
15	0.023773	4.43656	0.009357	6.66062	6.06170	17.8369
16	0.017733	4.47553	0.016509	6.71076	7.39601	17.9440

表 6 Minima函数在各维度和各聚类个数下的寻优精度和寻优时间

Table 6 The optimization precision and elapsed time of Minima under each dimension and the clustering number

聚类个数	Minima					
	5 dim		50 dim		200 dim	
	寻优精度	寻优时间	寻优精度	寻优时间	寻优精度	寻优时间
2	−78.3323	3.40415	−78.3323	6.19760	−76.3139	19.0165
3	−78.3323	3.44428	−78.3323	6.23811	−73.2504	19.1701
4	−78.3323	3.58257	−78.3323	6.29090	−69.4609	19.2653
5	−78.3323	3.46028	−78.3323	6.30427	−67.5982	19.4082
6	−78.3323	3.48343	−78.3323	6.35569	−65.0596	19.5425
7	−78.3323	3.48786	−78.3323	6.37087	−65.3063	19.6472
8	−78.3323	3.51281	−78.3321	6.55309	−63.2086	19.7334
9	−78.3323	3.52584	−78.3318	6.50598	−61.4462	19.8204
10	−78.3323	3.55163	−78.3317	6.53066	−60.4829	19.8715
11	−78.3323	3.60423	−78.3306	6.57872	−59.7747	19.9174
12	−78.3323	3.58804	−78.3297	6.59561	−59.1137	20.0138
13	−78.3323	3.57658	−78.3156	6.63598	−58.6150	20.1154
14	−78.3323	3.60389	−78.2995	6.67300	−56.9772	20.1997
15	−78.3323	3.59445	−78.2905	6.67469	−57.0165	20.2766
16	−78.3323	3.61131	−78.1948	6.72769	−56.0952	20.3846

4.2 BSO1算法与其它BSO算法的对比(The contrast of BSO1 and other BSO algorithms)

下面采用上述4个测试函数在30, 100, 500维时, 对本文提出的通过简单的聚类方式的BSO1与文献[5]的MBSO, 文献[8]的BSO–OS进行对比分析和比较. 各算法的参数设置如表7所示.

所有算法在MATLAB2015a上实现, 实验在相同的机器上运行, 为2.40 GHz Intel core i3 CPU, 2 GB RAM和Windows 7操作系统. 各算法在各维数均运行30次, 记录其平均值、最优值、最劣值、方差和运行时间, 其统计结果见表8, 下划线表示最优值.

表 7 各对比算法的参数设置

Table 7 The parameters of comparison algorithms

算法	参数
BSO–OS	$\mathrm{perc_e}=0.1,\ p_{\mathrm{e}}=0.2,\ p_{\mathrm{one}}=0.8,\ k=25,\ \mu=0,\ \sigma=1$
MBSO	$p_{\mathrm{r}}=0.005, p_{5\mathrm{a}}=0.2,\ p_{6\mathrm{b}}=0.8, M=5,$ population$=100$
BSO1	$p_{6\mathrm{biii}}=0.4, p_{6\mathrm{c}}=0.5,$ max_iteration$=2000$

表 8 不同算法在不同函数的平均值、最优值、最劣值、方差、运行时间

Table 8 The mean, optimal, worst, variance and elapsed time of different algorithms for different functions

Benchmark函数	维度	算法	平均值	最优值	最劣值	方差	运行时间/s
Sphere	30	BSO–OS	9.356e − 34	6.731e − 34	1.246e − 33	1.975e − 68	21.0504
		MBSO	5.177e − 57	1.924e − 61	8.229e − 56	3.10e − 112	20.5688
		BSO1	1.705e − 73	3.350e − 77	1.728e − 72	1.37e − 145	13.5724
	100	BSO–OS	1.048918	0.454271	1.689933	0.0719228	24.0608
		MBSO	1.607e − 17	7.733e − 19	1.519e − 16	1.216e − 33	30.5712
		BSO1	1.010e − 21	1.712e − 22	3.368e − 21	7.849e − 43	16.8860
	500	BSO–OS	1.448e + 03	1.244e + 03	1.762e + 03	1.792e + 04	40.5200
		MBSO	0.044262	0.024633	0.115148	2.708e − 04	85.9082
		BSO1	0.008548	0.004704	0.014437	7.398e − 06	33.5656
Rastrigin	30	BSO–OS	32.5291	18.3108	60.6644	90.2303	16.8342
		MBSO	0	0	0	0	15.4702
		BSO1	0	0	0	0	15.6650
	100	BSO–OS	2.728e + 02	1.633e + 02	3.328e + 02	1.459e + 03	19.5869
		MBSO	6.790e − 08	4.215e − 12	8.423e − 07	3.508e − 14	25.6967
		BSO1	2.741e − 04	7.604e − 11	0.008198	2.239e − 06	14.2531
	500	BSO–OS	2.977e + 03	2.830e + 03	3.203e + 03	8.214e + 03	43.2115
		MBSO	1.948e + 03	1.733e + 03	2.319e + 03	2.000e + 04	86.7294
		BSO1	2.075e + 03	1.769e + 03	2.360e + 03	1.490e + 04	34.6316
Griewank	30	BSO–OS	0.005912	0	0.029552	6.760e − 05	25.2396
		MBSO	0	0	0	0	30.3147
		BSO1	0	0	0	0	20.1390
	100	BSO–OS	0.116911	0.070669	0.175699	6.067e − 04	35.2640
		MBSO	4.105e − 04	0	0.012316	5.056e − 06	51.5386
		BSO1	0	0	0	0	31.5623
	500	BSO–OS	4.452e + 02	2.071e + 02	8.576e + 02	2.602e + 04	84.4789
		MBSO	0.008962	0.003908	0.030793	2.780e − 05	124.748
		BSO1	0.004210	9.352e − 04	0.080467	2.076e − 04	42.5401
Minima	30	BSO–OS	−73.4630	−77.3898	−69.8502	3.006635	21.6899
		MBSO	−78.3323	−78.3323	−78.3323	0	17.1195
		BSO1	−78.3323	−78.3323	−78.3323	0	11.8680
	100	BSO–OS	−67.5347	−69.4524	−64.1852	1.806768	23.7691
		MBSO	−78.3323	−78.3323	−78.3323	1.671e − 27	29.8191
		BSO1	−78.3323	−78.3323	−78.3323	4.526e − 28	17.6412
	500	BSO–OS	−52.8975	−55.1170	−50.8101	1.032188	55.2891
		MBSO	−73.7509	−78.1819	−67.3678	10.05406	86.9617
		BSO1	−71.2136	−77.9252	−64.9661	12.67040	44.5234

由表8的结果可以看出, 差分变异无疑能达到比高斯变异更高的精度. 在维数较低时, 由于差分变异中仅有加减运算, 相比高斯变异中的既有加减乘除运算, 又调用logsig()函数来说, 总体上运算时间更小. 但随着维数的增加, SGM聚类耗时也倍增, 对于所有测试函数在100维时, MBSO运行时间均大于BSO–OS运行时间, 500维时时间性能差距又会增加. 这是因为尽管SGM聚类为一种简单的聚类方式, 但仍为基于解空间聚类, 函数维数增加, 计算距离所需时间也随之倍增, 相比之下基于目标空间聚类仍只有一维, 因此算法的快速性更好. 为了更进一步描述算法的收敛过程, 图4给出了不同维度下3种不同算法的收敛曲线.

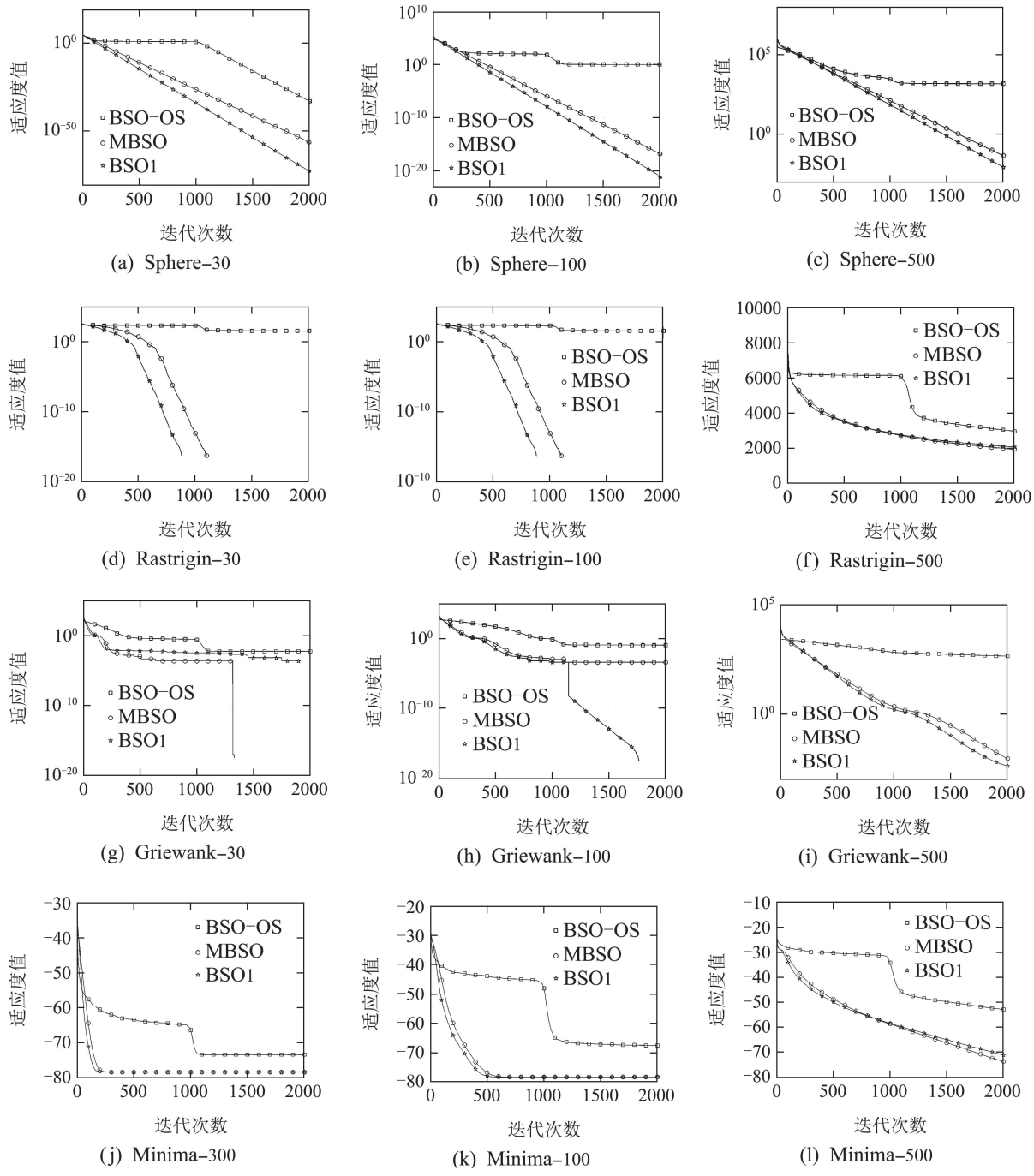

图 4　不同算法在不同函数不同维度的寻优曲线

Fig. 4　Different algorithms in different function optimization curve of different dimensions

比较上述图4中的3条曲线, MBSO与BSO1寻优精度相仿, 均领先于BSO–OS. 在时间性能方面, BSO1略优于BSO–OS, 远优于MBSO. 可见, 总体来说BSO1结合了MBSO的寻优精度高和BSO–OS寻

优速度快的特性, 且基于之前的讨论, 得知在聚类个数为2时算法有着最好的性能, 所以在对聚类方式作出改进之后, 算法在时间性能方面领先于BSO–OS, 而在寻优精度方面仍然有与MBSO相仿的特性.

5 实际应用(The practical application)

为了更加清楚地体现算法性能, 本文以电力系统热电联供经济调度(combined heat and power economic dispatch, CHPED)问题来进行仿真. 热电联供又叫热电联产, 是指热力发电厂在通过一定的方法, 在向广大用电用户供电的同时, 也向用户供暖, 极大地提高热电厂的经济效率. 在CHPED问题中, 包含了热机组、电机组和相互耦合的热电机组, 每一个机组有着输出火力的上下限, 同时热和电存在着双平衡, 即总发电量和总发热量应分别满足电负荷和热负荷的需求, 而且在热电联产机组中, 耦合关系使得热电出力不能分离开来, 即每一组热电都有着各级的限定区域. 总之, CHPED问题属于多约束优化问题.

本文采用7机组和48机组两种情况下的CHPED系统, 问题模型和具体参数详见文[9–10]. 考虑到算法的运行时间, 采用与参考文献相同的平台, 在7机组(情况1)和48机组(情况2)下分别设置最大迭代次数为300和800, 算法的其余参数设置如表9.

表 9 BSO1算法参数设置

Table 9 The parameters of BSO1

$P_{\rm r}$	P_{5a}	P_{6b}	P_{6biii}	P_{6c}	种群规模
0.005	0.1	0.8	0.4	0.5	100

算法运行30次, 得到其中的最优值和平均值, 及运行时间, 并绘制平均值和最优值的收敛曲线. 寻优结果与参考文献的对比如表10.

表 10 各算法在情况1下的最优花费对比

Table 10 Comparison of optimal costs obtained by different algorithms for test Case 1

算法	最少花费/$	运行时间/s
PSO[11]	10613	5.3844
EP[11]	10390	5.2750
DE[11]	10317	5.2563
RCGA[12]	10667	6.4723
BCO[12]	10317	5.1563
CPSO[9]	10325.3339	3.29
TVAC-PSO[9]	10100.3164	3.25
TLBO[10]	10094.8384	2.86
OTLBO[10]	10094.3529	3.06
BSO1	10093.6666	3.7420

由表10和表11可见, 对于CHPED问题, 无论是对于7机组还是48机组, BSO1在精度方面有着明显的优势, 甚至优于当前较好的TLBO和OTLBO. 在时间性能方面, 对48 机组而言, BSO1算法也明显优于基本的PSO, EP, DE, RCGA和BCO, 与CPSO, TLBO, TVAC–PSO和OTLBO相比也有优越性. 7机组的运行时间比TLBO稍长.

图5和图6分别给出了两种规模下算法的收敛曲线. 由图5可见, 本文算法仅在50代以后就收敛到了最优解, BSO1在情况1上统计的花费时间稍长, 但是为了和其它算法相比较, 设置了相同的迭代次数, 实际算法的性能要好得多.

表 11 各算法在情况2下的最优花费对比

Table 11 Comparison of optimal costs obtained by different algorithms for test Case 2

算法	最少花费/$	运行时间/s
CPSO[9]	119708.8818	14.00
TVAC–PSO[9]	117824.8956	13.44
TLBO[10]	116739.3640	10.38
OTLBO[10]	116579.2390	10.93
BSO1	116362.4053	10.0896

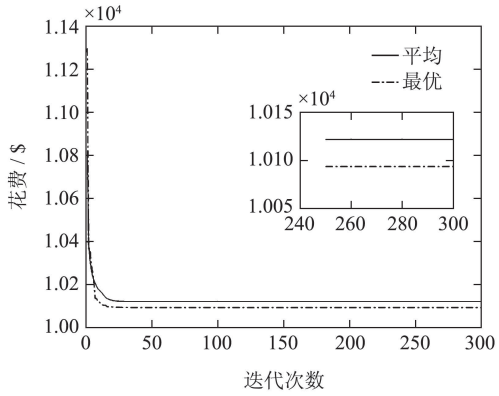

图 5 情况1下BSO1的收敛曲线

Fig. 5 The cost convergence curve of BSO1 for test Case 1

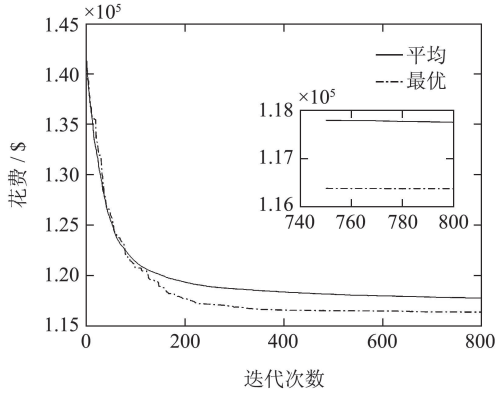

图 6 情况2下BSO1的收敛曲线

Fig. 6 The cost convergence curve of BSO2 for test Case 2

6 总结(Conclusions)

本文通过对头脑风暴优化算法的“聚”“散”行为及性能特征的分析, 提出了一种基于目标空间聚类的差分头脑风暴优化算法. 算法采用简单的差分变异取代原BSO算法中的高斯和柯西变异, 极大增加了种群的多样性; 通过目标空间聚类实现了聚类操作的简化, 降低了算法的复杂度. 通过对算法的参数和性能进行分析可以看出, 本文算法不但寻优精度高, 而且寻优速度快, 是一种性能良好的群智能优化算法. 通过对电力系统的环境经济调度问题中的7机组和48机组问题进行仿真分析, 结果表明本文算法对应用实际问题也有很好的性能. 算法的进一步理论分析和应用领域扩展是作者下一步的研究内容.

参考文献(References):

[1] YANG Yuting. *Research on brain storm optimization algorithm and video based non-contact motion quantitative analysis method* [D]. Hangzhou: Zhejiang University, 2015.
(杨玉婷. 头脑风暴优化算法与基于视频的非接触式运动定量分析方法研究 [D]. 杭州: 浙江大学, 2015.)

[2] SHI Y H. Brain storm optimization algorithm [M] //*Advances in Swarm Intelligence*. Berlin Heidelberg: Springer, 2011: 303 – 309.

[3] ZHU H Y, SHI Y H. Brain storm optimization algorithm swith k-medians clustering algorithms [C] //*The 7th International Conference on Advanced Computational Intelligence*. Wuyi, China: IEEE, 2015: 107 – 110.

[4] YANG Yuting, DUAN Dingna, ZHANG Huan, et al. Kinematic recognition of hidden markov model based oncimproved brain storm optimization algorithm [J]. *Space Medicine & Medical Engineering*, 2015, 28(6): 403 – 407.
(杨玉婷, 段丁娜, 张欢, 等. 基于改进头脑风暴优化算法的隐马尔可夫模型运动识别 [J]. 航天医学与医学工程, 2015, 28(6): 403 – 407.)

[5] ZHAN Z H, ZHANG J, SHI Y H, et al. A modified brain storm optimization [J]. *Evolutionary Computation*, 2012, 22(10): 1 – 8.

[6] YANG Yuting, SHI Yuhui, XIA Shunren. Discussion mechanism based on brain storm optimization algorithm [J]. *Journal of Zhejiang University*, 2013, 47(10): 1705 – 1711.
(杨玉婷, 史玉回, 夏顺仁. 基于讨论机制的头脑风暴优化算法 [J]. 浙江大学学报(工学版), 2013, 47(10): 1705 – 1711.)

[7] XUE J Q, WU Y L, SHI Y H, et al. Brain storm optimization algorithm for multi-objective optimization problems [J]. *Lecture Notes in Computer Science*, 2012, 7331(4): 513 – 519.

[8] SHI Y H. Brain storm optimization algorithm in objective space [C] //*IEEE Congress on Evolutionary Computation*. Sendai, Japan, 2015: 1227 – 1234.

[9] MOHAMMADI I B, MORADI D M, RABIEE A. combined heat and power economic dispatch problem solution using particle swarm optimization with time varying acceleration coefficients [J]. *Electric Power Systems Research*, 2013, 95(1): 9 – 18.

[10] ROY P K, PAUL C, SULTANA S. Oppositional teaching learning based optimization approach for combined heat and power dispatch [J]. *International Journal of Electrical Power & Energy Systems*, 2014, 57(5): 392 – 403.

[11] BASU M. Combined heat and power Economic dispatch by using differential evolution [J]. *Electric Power Components & Systems*, 2010, 38(8): 996 – 1004.

[12] BASU M M. Bee colony optimization for combined heat and power economic dispatch [J]. *Expert Systems with Applications*, 2011, 38(11): 13527 – 13531.

作者简介:

吴亚丽　(1975–), 女, 副教授, 主要研究方向为智能优化算法理论及应用研究、复杂系统建模与优化, E-mail: yliwu@xaut.edu.cn;

付玉龙　(1994–), 男, 硕士研究生, 主要研究方向为多目标群智能优化算法及其应用, E-mail: fylshadow@163.com;

王鑫睿　(1993–), 女, 硕士研究生, 主要研究方向为深度学习优化算法, E-mail: xinruiwang6@163.com;

刘　庆　(1983–), 男, 讲师, 主要从事群智能优化理论及应用、数据驱动建模等方面研究, E-mail:liuqing@xaut.edu.cn.