doi:10.11959/j.issn.1000-436x.2016224

带自适应精英扰动及惯性权重的反向粒子群优化算法

董文永[1]，康岚兰[1,2]，刘宇航[1]，李康顺[3]
(1. 武汉大学计算机学院，湖北 武汉 430072；2. 江西理工大学应用科学学院，江西 赣州 341000;
3. 华南农业大学信息学院，广东 广州 510642)

摘 要：针对反向粒子群优化算法存在的易陷入局部最优、计算开销大等问题，提出了一种带自适应精英粒子变异及非线性惯性权重的反向粒子群优化算法（OPSO-AEM&NIW），来克服该算法的不足。OPSO-AEM&NIW 算法在一般性反向学习方法的基础上，利用粒子适应度比重等信息，引入了非线性的自适应惯性权重（NIW）调整各个粒子的活跃程度，继而加速算法的收敛过程。为避免粒子陷入局部最优解而导致搜索停滞现象的发生，提出了自适应精英变异策略（AEM）来增大搜索范围，结合精英粒子的反向搜索能力，达到跳出局部最优解的目的。上述 2 种机制的结合，可以有效克服反向粒子群算法的探索与开发的矛盾。实验结果表明，与主流反向粒子群优化算法相比，OPSO-AEM&NIW 算法无论是在计算精度还是计算开销上均具有较强的竞争能力。
关键词：一般性反向学习；粒子群优化；自适应精英变异；非线性惯性权重
中图分类号：TP181 **文献标识码**：A

Opposition-based particle swarm optimization with adaptive elite mutation and nonlinear inertia weight

DONG Wen-yong[1], KANG Lan-lan[1, 2], LIU Yu-hang[1], LI Kang-shun[3]
(1. Computer School, Wuhan University, Wuhan 430072, China;
2. Faculty of Applied Science, Jiangxi University of Science and Technology, Ganzhou 341000, China;
3. College of Information, South China Agricultural University, Guangzhou 510642, China)

Abstract: An opposition-based particle swarm optimization with adaptive elite mutation and nonlinear inertia weight (OPSO-AEM&NIW) was proposed to overcome the drawbacks, such as falling into local optimization, slow convergence speed of opposition-based particle swarm optimization. Two strategies were introduced to balance the contradiction between exploration and exploitation during its iterations process. The first one was nonlinear adaptive inertia weight (NIW), which aim to accelerate the process of convergence of the algorithm by adjusting the active degree of each particle using relative information such as particle fitness proportion. The second one was adaptive elite mutation strategy (AEM), which aim to avoid algorithm trap into local optimum by trigging particle's activity. Experimental results show OPSO-AEM&NIW algorithm has stronger competitive ability compared with opposition-based particle swarm optimizations and its varieties in both calculation accuracy and computation cost.
Key words: generalized opposition-based learning, particle swarm optimization, adaptive elite mutation, nonlinear inertia weight

1 引言

粒子群优化算法(PSO, particle swarm optimization)是一种基于群体进化的随机仿生优化算法，由 Kennedy 和 Eberhart 等[1]于 1995 年提出，其思想源于对鱼群及鸟类等群体觅食行为的模拟。算法自提出以来，由于其概念简单且易于理解和实现，在解决复杂优化问题，如非线性、多峰等问题性能表现

收稿日期：2015-09-22；**修回日期**：2016-09-09
通信作者：康岚兰，victorykll@163.com
基金项目：国家自然科学基金资助项目（No.61170305，No.61672024）
Foundation Item: The National Natural Science Foundation of China (No.61170305，No.61672024)

突出，从而吸引了大批科研人员对其展开研究。目前，PSO 已成功应用于交通、制造、通信工程、国防等多个不同学科及工程领域[2~4]，并取得了良好的效果。

传统 PSO 自提出以来，存在着几个天然缺陷，如参数依赖过大、局部搜索能力不足且易陷入局部最优、搜索精度低等。因此，大量科研人员从参数设置、自适应调整、算法混合或融合等多个角度对传统 PSO 进行改良。PSO 中每个粒子通过一个简单的运动向量公式来控制其飞行方向，该运动向量公式主要由 3 个关键参数控制：惯性权重(w)、认知学习因子(c_1)与社会学习因子(c_2)。因此，如何控制惯性大小、设定参数，如何更好地获取搜索环境的有效信息，避免粒子陷入局部最优的同时加快算法收敛速度是改进传统 PSO 的关键。

本文提出了一种带自适应精英扰动及惯性权重的反向粒子群优化算法(OPSO-AEM&NIW, opposition-based PSO with adaptive elite mutation and nonlinear inertia weight)。OPSO-AEM&NIW 算法取当前全局最优位(*gbest*)为精英粒子，将针对精英粒子的自适应变异策略(AEM)融入到反向学习中，使各粒子的自身历史最优位(*pbest*)趋向收敛于精英粒子(*gbest*)时，依据粒子的聚集程度，自适应产生变异位，从而帮助粒子跳出局部最优。同时，为加快算法的收敛，一种非线性动态惯性权重策略(NIW)被引入到新算法中，该策略使惯性权重随着粒子适应度值的变化而自适应改变。本文将新算法与多种基于反向学习 PSO 算法在 14 个典型测试函数上进行数值实验，实验结果表明 OPSO-AEM&NIW 算法在大部分测试函数上取得最优解的同时，收敛速度得到了显著的提高。

2 相关工作

2.1 基本 PSO

PSO 算法是仿生算法家族中的又一种群智能随机搜索算法，群体中个体被看作是搜索空间中没有质量和体积的粒子，每个粒子表征问题的一个候选解，具有速度与位置 2 个特征。在解空间中，粒子以一定的速度向自身历史最优位与全局最优位运动，不断进化候选解。

设群体在 D 维空间中包含 N 个粒子，第 i 个粒子 $x_i(t)=(x_{i,1},x_{i,2},\cdots,x_{i,D})$ 在第 j 维的速度分量 $v_{i,j}$ 和位置分量 $x_{i,j}$ 在 t+1 时刻的更新式如下 [5]

$$v_{i,j}(t+1)=\omega v_{i,j}(t)+c_1 rand_1(pbest_{i,j}-x_{i,j}(t))+c_2 rand_2(gbest_j-x_{i,j}(t)) \quad (1)$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1) \quad (2)$$

其中，$i=1,2,\cdots,N$, $j=1,2,\cdots,D$，ω 是惯性权值，$\omega\in[0,1]$，c_1 和 c_2 分别称为认知学习因子和社会学习因子，通常，$c_1,c_2\in[0,2]$，$rand_1$ 和 $rand_2$ 是在[0, 1]上服从均匀分布的 2 个随机数。

2.2 一般化的反向学习策略

2005 年，Tizhoosh[6]首次提出了反向学习(OBL, opposition-based learning)的概念。其主要思想是同时评估一个可行解与其反向解，将其中较优解保留，进入下一代进化。设 $\breve{x}_j$ 是粒子 x_j 的反向解，则 $\breve{x}_j$ 为

$$\breve{x}_j=a_j+b_j-x_j \quad (3)$$

其中，x_j 的范围为 $[a_j,b_j]$。依据概率论，解空间中个体反向解等概率优于原粒子，如对原粒子解与其反向解所构造的解空间同时进行搜索，将大大提高最优解的搜索效率。

2007 年，Wang 等[7]将 OBL 应用到 PSO 中，提出了一种基于 OBL 的 PSO 算法(OPSO)。2011 年，Wang 等在 OBL 的基础上进一步提出了一般性反向学习(GOBL, generalized OBL)的概念，在 GOBL 基础上提出了一种改进的反向学习粒子群算法 GOPSO[8]。2013 年，周新宇等[9]将精英思想引入到反向学习中，提出了精英反向粒子群算法(EOPSO)，该算法取群体中每个粒子的 *pbest* 作为精英粒子群，求其每一位的反向解，在其与原粒子群合并空间中选取适应值最优的前 N 个粒子进入下一次迭代中。同时，对当前全局最优位采用了差分变异(DEM, differential evolutionary mutation)进行扰动。为防止粒子越出搜索区域，文献[10]采用一种速度阀(velocity clamping)来控制粒子速度大小，从而进一步平衡算法的全局搜索与局部探测能力。Kaucic[11]针对边界约束问题提出了一种自适应速度的多始发反向粒子群优化算法(MSOPSOAV)，算法设计了一个基于差分操作的自适应速度更新公式来加强粒子优化能力，并采用一个超反向原则(super-opposition paradigm)对出现早熟的粒子重新初始化。Pehlivanoglu 在文献[12]中提出了一种多频振动 PSO 算法。该算法通过一种周期性变异策略，

结合人工神经网络方法保持了种群多样性，从而有效防止早熟现象的发生，使算法收敛速度得到大大的提高。基于参数的控制对进化算法的重要性，文献[13]在 2015 年对进化算法中参数的控制方法、趋势及挑战做出了综合性的讨论。接下来，给出一般反向学习的定义。

定义 1　一般反向学习策略(GOBL, generalized opposition-based learning)[8,14]。设 $\vec{x}_i=(x_{i,1},x_{i,2},\cdots,x_{i,D})$ 是 D 维空间中的一个普通粒子，其对应的反向解为 $\breve{x}_i=(\breve{x}_{i,1},\breve{x}_{i,2},\cdots\breve{x}_{i,D})$，每一维 $\breve{x}_{i,j}$ $(j=1,2,\cdots,D)$ 定义为

$$\breve{x}_{i,j}=k(da_j+db_j)-x_{i,j} \tag{4}$$

其中，$k\in$ U(0,1) 的随机数，该设定使粒子获得更好的反向解[9]，$x_{i,j}\in[a_j,b_j]$，$[da_j,db_j]$为第 j 维搜索空间的动态范围，取值为

$$da_j=\min(x_{i,j}),\quad db_j=\max(x_{i,j}) \tag{5}$$

若 $\breve{x}_{i,j}$ 取值在某代进化中跳出可行解边界，将对其在$[da_j,db_j]$内采用随机生成的方法进行重置。

$$\breve{x}_{i,j}=\begin{cases}randn(da_j,db_j), & \breve{x}_{i,j}<da_j \text{ 或 } \breve{x}_{i,j}>db_j\\ \breve{x}_{i,j}, & \text{其他}\end{cases} \tag{6}$$

依据演化算法收敛性理论，为增强算法的顽健性，$randn(da_j,db_j)$取在动态边界$[da_j,db_j]$上服从高斯分布的随机数。这一设定在实验过程已被验证取得了较好的算法稳定性。高斯分布函数定义为

$$F(x;\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x}\exp(-\frac{(x-\mu)^2}{2\sigma^2})\mathrm{d}x \tag{7}$$

其中，均值 $\mu=\dfrac{da_j+db_j}{2}$，方差 $\sigma=1$。

3　OPSO-AEM&NIW 算法

种群在进化过程中，每个粒子搜索潜力不尽相同，为充分发挥优势粒子潜力，进一步提高算法效率，本文采取“精英策略(elite strategy)”，给予群体中适应值占优的粒子(称为“精英粒子(elite particle)”)更多的进化机会。若精英粒子在试探过程中探索到新的优势点，则将其替代群体中被选择的劣势个体。然而，过度采取精英策略可能导致易陷入局部最优及最优值振荡等问题。为了克服上述问题，本文引入一种非线性自适应惯性策略来自适应调节各个粒子的活跃程度，增大算法跳出局部最优解的能力，和精英策略相互配合来调节算法全局探索与局部开采之间的平衡，提升反向 PSO 的性能。

3.1　自适应精英变异策略(AEM)

在精英策略中，精英粒子入选比例的确立对问题的求解起到关键的作用。一方面，比例过大，种群多样性随着进化过程的深入而大幅降低，从而易陷入局部最优；另一方面，一些理论分析表明，在未收敛前，粒子群搜寻的全局最优位总是在先前已知的几个候选解之间来回振荡[13, 15]。因此，本文算法仅将种群中 *gbest* 位作为精英粒子，在每一代进化过程中，根据式(11)和式(14)对 *gbest* 做自适应变异操作，由其产生的新个体对环境做进一步试探。若新个体适应值优于原 *gbest* 适应值，则取代之成为新的 *gbest* 位参与到新一轮的进化过程中。本文将这种融入精英思想的自适应变异方法称为自适应精英变异策略(AEM, adaptive elite mutation strategy)。

AEM 变异操作中，将通过自适应生成变异种子 *xm* 及自适应选取变异常量 *C*，产生真实变异量 *F*(*xm*)代入到式(14)中，实现对精英粒子 *gbest* 自适应扰动的目的。具体地，*xm* 的生成综合考虑个体 *pbest* 与精英粒子 *gbest* 的距离 *r* 以及进化代数 *t* 的关系，依据式(9)分别计算 *pbest* 到 *gbest* 在各维上的距离，距离越大，表示群体间相似性越小，再依据适应度标准差 *st_d* 的取值不同(见式(13))，自适应取得较大的变异常量 *C*，生成较大变异量 *F*(*xm*)代入式(14)，使原 *gbest* 位自适应获得较大的变异位 $gbest^*$。若变异后的 $f(gbest^*)$ 优于 $f(gbest)$ ($f(\bullet)$ 为问题适应度函数)，*gbest* 将被 $gbest^*$ 取代。新全局最优个体*gbest*在后续的进化过程中将吸引群体粒子从而帮助粒子跳出局部最优位。由此可见，AEM 在算法初期能够自适地使 *gbest* 获得足够的扰动，达到扩大搜索空间，增强算法的全局搜索能力的目的；反之，在迭代后期，逐渐减小的变异率使最优解在避免振荡的同时加速了算法的收敛。

在 AEM 变异中，首先根据式(8)计算变异种子 *xm* 在各维上的值。

$$xm(i)=\exp\left(-\lambda\frac{t}{t_{\max}}\right)\left(1-\frac{r(i)}{r_{\max}}\right) \tag{8}$$

其中，$i=1,2,\cdots,D$，λ 是常数，根据 4.2.3 节中对参数 λ 的实验分析结果，在后续算法性能分析实验中取

建议值 $\lambda=30$；t 为进化代数，$t_{\max}$ 是最大进化代数；$r(i)$ 是 N 个个体的 $pbest$ 在各维上的平均值 $avg\_pbest(i)$ 到 $gbest$ 的距离；$r_{\max}$ 是各维间最大距离。

$$r(i)=\left|gbest(i)-avg\_pbest(i)\right| \tag{9}$$

$$avg\_pbest(i)=\frac{\sum_{j=1}^{N}pbest[j][i]}{N} \tag{10}$$

其中，$pbest[j][i]$ 是第 j 个粒子在第 i 维上的位置。

将 xm 代入式(11)，自适应产生变异量。

$$F(xm)=\frac{1}{\pi}\arctan(xm)+C \tag{11}$$

其中，C 是一个待定常量，它将根据适应度标准差 st_d 的不同而自主选取不同的值，具体取值如下

$$C=\begin{cases}1.5, & st\_d<10^{-2}\\ 1.0, & 10^{-2}\leqslant st\_d<10^{-1}\\ 0.5, & \text{其他}\end{cases} \tag{12}$$

$$st\_d=\sum_{i=1}^{N}\left|\frac{f_i-f_{gbest}}{f_{gbest}}\right| \tag{13}$$

st_d 将根据式(13)来判断种群聚集度，其值被分成 $(0,10^{-2}]$、$(10^{-2},10^{-1}]$ 及 $(10^{-1},+\infty)$ 3 段(不同分段点内取值分布情况在本文 4.2.3 节中做出统计分析)。其中，f_i 为第 i 个个体的适应值，f_{gbest} 为 $gbest$ 的适应值。当个体越相似，即 $f_i\to f_{gbest}$，则 st_d 取值越小，表示群体越聚集，个体之间的距离越小，C 取得的值将越大，对应式(11)产生的变异量越大。图 1 展示了当 C=0.5 时变异量 $F(xm)$ 与 xm 中变量之间的对应关系。

如图 1 所示，一方面，当 r 较小，即群体极值趋于一致时，当前全局最优位将获得较大的变异值(F 较大)，算法的搜索能力获得提高；反之，当 r 逐渐增大，即群体较分散时，变异值的减少(F 较小)可避免最优值搜寻过程的回荡，从而加速算法的收敛。另一方面，在迭代初期(t 较小)，由于先验认知的不足，种群获得较大的变异值(F 较大)，对解空间进行充分的搜索；反之，随着进化的深入(t 逐渐增大)，变异值 F 随之减小，从而确保问题能够平滑地收敛到最优值。

结合上述分析结论，得到自适应精英变异操作如下

$$gbest^{*}=gbest+F(xm) \tag{14}$$

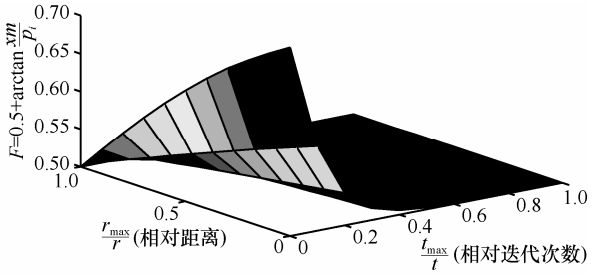


图 1 自适应精英变异三维空间

AEM 在每一次迭代中，式(14)根据 $F(xm)$ 对 $gbest$ 进行扰动，产生变异位 $gbest^{*}$，若 $f(gbest^{*})$ 优于 $f(gbest)$，则 $gbest=gbest^{*}$。

3.2 非线性自适应惯性权重更新策略

在速度更新中，本文采用一种如式(15)所示的非线性的自适应策略对粒子惯性权值进行动态调整[16]为

$$w=\begin{cases}w_{\min}+\dfrac{(w_{\max}-w_{\min})(f_i-f_{\min})}{(f_{\text{avg}}-f_{\min})}, & f_i\leqslant f_{\text{avg}}\\ w_{\max}, & f_i>f_{\text{avg}}\end{cases} \tag{15}$$

其中，$w_{\min}$、$w_{\max}$ 分别为 w 的最小值与最大值；f_i 为粒子 i 的适应值；$f_{\min}$、f_{avg} 分别为当前粒子群的最小适应值和平均适应值。由式(15)可知，当粒子适应值优于平均适应值时，取得最大惯性权重，从而增加粒子的活跃度；反之，取得较小惯性权重，使粒子更多地被精英粒子吸引，从而向优势搜索空间靠拢。同时，当所有粒子趋于一致(此时 $\left|f_{\text{avg}}-f_{\min}\right|$ 较小)，w 随之增加；较分散时(此时 $\left|f_{\text{avg}}-f_{\min}\right|$ 较大)，w 随之减小。以上非线性自适应惯性权重更新策略将进一步平衡算法的全局搜索与局部探测能力。

3.3 OPSO-AEM&NIW

本文将 AEM 与 NIW 策略同时应用于反向学习 PSO 算法中，给出了一种带自适应精英扰动及惯性权重的反向粒子群优化算法，算法 1 用伪代码描述了 OPSO-AEM&NIW 的基本步骤。其中，OP 为当前种群 P 的反向种群，jr 为使用 GOBL 策略的概率。

3.4 算法时间复杂度分析

对算法 1 分析可知，OPSO-AEM&NIW 算法主要由初始化种群、GOBL 策略、速度与位置更新操作、NIW 策略以及 AEM 策略 5 个部分组成。对于

初始化种群和 GOBL 策略 2 个部分，除前者包含随机化种群，后者包含边界动态更新外，两者还包含共同部分：反向种群生成及群体选择机制，显然，随机化种群与反向种群的生成，以及边界动态更新部分的时间复杂度均为 $O(ND)$，对于群体选择机制，排序是其主要操作，计算复杂度为 $O(N^2)$。特别注意，在低维空间中，通常种群规模 N 取近似于维度 D 的数，即 $N \approx D$，若维度较高，如 $D \geqslant 100$，则通常可取 $N<D$，计算复杂度 $O(N^2)$ 可计为 $O(ND)$。因此，初始化种群和 GOBL 策略 2 个部分的计算复杂度为 $O(ND)$；在 OPSO-AEM&NIW 中，不难得出，NIW 策略、AEM 策略以及速度与位置更新操作的计算复杂度均为 $O(ND)$。综上所述，OPSO-AEM&NIW 的计算复杂度为 $O(ND)$。

算法 1　OPSO-AEM&NIW 算法

1) 随机初始化含 N 个粒子的种群 P;

2) 根据式(4)和式(6)获得 N 个粒子的反向粒子，构成其反向种群 OP;

3) 计算种群 $P \cup OP$ 中每个粒子的适应值，选择其中较优的 N 个粒子作为初始种群 P;

4) while 未达到终止条件时 do

5) if $rand(0,1)<jr$ then

6) 根据式(5)更新动态范围 $[da_j, db_j]$;

7) 根据式(4)和式(6)更新 OP，在解空间 $P \cup OP$ 中选取 N 个适应值占优的粒子;

8) 更新 $pbest$ 和 $gbest$;

else

9) for i=1 to N do

10) 根据式(15)求自适应惯性系数 w，并代入式(1)和式(2)中，获得第 i 个粒子的速度与位置向量，并更该粒子适应值;

11) 更新 $pbest$ 和 $gbest$;

12) end for

13) end if

14) for j=1 to D do

15) 对当前获得的 $gbest$ 采用 AEM 策略，即对式(14)进行变异操作;

16) end for

17) if $fitness(gbest^*)< fitness(gbest)$ then

18) 用 $gbest^*$位更新当前 $gbest$ 位;

19) end if

20) end while

4　数值实验及分析

本文实验将被分成 3 个部分。1) 算法性能测试，除基本 PSO 外，OPSO-AEM&NIW 还与其他 4 种基于反向学习的 PSO 算法(OBL-based PSO)进行比较，分别是 OPSO[8]、GOPSO[8]、OVCPSO[10]和 EOPSO[9]。实验中，参与对比算法的种群规模 N 均取 40，其他参数保持与原文献一致的设置。2) 策略分析，分析 AEM 和 NIW 这 2 种策略为提高 OPSO-AEM&NIW 算法性能各自产生的影响，以及两者共同作用下为避免陷入局部最优策略所发挥的作用。3) 参数敏感性分析，观察参数 λ (AEM)取不同值时对算法 OPSO-AEM&NIW 性能的变化情况；并分析 AEM 策略中待定参数 C 取不同常量值的平均概率情况；同时，对 NIW 策略中参数 w 自适应取值分布进行分析。依据文献[7, 17]中对参数 c_1、c_2、jr 取值分析，以及本文对参数 w、λ 的敏感性分析，在后续实验中，若无特别说明，OPSO-AEM&NIW 默认参数设置如表 1 所示。

表 1　OPSO-AEM&NIW 参数设置

c_1	c_2	$w_{\max}$	$w_{\min}$	jr	λ
1.496 18	1.496 18	0.5	0.2	0.3	30

4.1　实验设置

所有实验针对表 2 所列 14 个测试函数展开。按其属性分为单峰(f_1~f_5)和多峰(f_6~f_{14})两大类，其中，多峰函数又被细分为 3 类：简单多峰函数(f_6~f_8)、带旋转的多峰函数(f_9~f_{11})和移位多峰函数(f_{12}~f_{14})。为使所有测试函数在零点位，均取全局最优值 0，本文参考 CEC 2010[18]中的函数定义，将移位多峰函数中的参数 f_bias 统一设定为 0。具体测试函数如表 2 所示。

4.2　实验结果与分析

4.2.1　算法对比分析

在 6 种 PSO 算法的对比实验中，算法最大迭代次数为 10 000，每个测试函数分别运行 30 次，记录全局最优值的均值。实验采用双尾 t 检验对各算法结果进行显著性差异统计分析，显著水平为 0.05。表 3 给出了 6 种对比算法的实验结果，最好结果用黑体显示，下同，表中用“－”、“+”、“~”这 3 种符号表示 OPSO-AEM&NIW 算法的性能“劣于”、“优于”、“相当于”对应比较算法。

从表 3 中可得出 OPSO-AEM&NIW 在 6 种算法

表 2　数值实验中使用的 14 个测试函数

属性	测试函数	维度 D	搜索区间	全局最优值	函数名
单峰函数	$f_1(x)=\sum_{i=1}^{D}x_i^2$	30	$[-100, 100]^D$	0	Sphere
	$f_2(x)=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	30	$[-100, 100]^D$	0	Step
	$f_3(x)=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(1-x_i)^2]$	30	$[-30, 30]^D$	0	Rosenbrock
	$f_4(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	30	$[-100, 100]^D$	0	Quadric
	$f_5(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30	$[-10, 10]^D$	0	S chwefel2.22
多峰函数	$f_6(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	30	$[-5.12, 5.12]^D$	0	Rastrigin
	$f_7(x)=-20\exp(-0.2\sqrt{\frac{\sum_{i=1}^{D}x_i^2}{D}}-\exp\frac{\sum_{i=1}^{D}\cos(2\pi x_i)}{D}+20+\mathrm{e}$	30	$[-32, 32]^D$	0	Ackley
	$f_8(x)=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	30	$[-600, 600]^D$	0	Griewank
	$f_9(x)=f_6(z),\quad z=x\boldsymbol{M}$	30	$[-5.12, 5.12]^D$	0	Rotated Rastrigin
	$f_{10}(x)=f_7(z),\quad z=x\boldsymbol{M}$	30	$[-32, 32]^D$	0	Rotated Ackley
	$f_{11}(x)=f_8(z),\quad z=x\boldsymbol{M}$	30	$[-600, 600]^D$	0	Rotated Griewank
	$f_{12}(x)=f_6(z),\quad z=x-\boldsymbol{o}$	30	$[-5.12, 5.12]^D$	0	Shifted Rastrigin
	$f_{13}(x)=f_7(z),\quad z=x-\boldsymbol{o}$	30	$[-32, 32]^D$	0	Shifted Ackley
	$f_{14}(x)=f_8(z),\quad z=x-\boldsymbol{o}$	30	$[-600, 600]^D$	0	Shifted Griewank

注：$\boldsymbol{M}$ 为 DXD 正交矩阵，$\boldsymbol{o}$ 为移位向量。

表 3　PSO、OPSO、GOPSO、OVCPSO、EOPSO 和 OPSO−AEM&NIW 算法在 14 个测试函数上的平均适应值

测试函数	PSO		OPSO		GOPSO		OVCPSO		EOPSO		OPSO−AEM&NIW
f_1	1.56×10^{-3}	+	4.59×10^{-36}	+	**0**	~	5.00×10^{-1}	+	**0**	~	**0**
f_2	7.18×10^{-2}	+	2.09×10^{-35}	+	3.02×10^{-321}	+	6.67×10^{-2}	+	1.97×10^{-323}	+	**0**
f_3	1.26	~	7.18	~	2.82×10^{1}	+	8.70×10^{1}	+	$\mathbf{1.57\times10^{-24}}$	−	6.94
f_4	5.51×10^{-2}	+	4.13×10^{4}	+	1.32×10^{4}	+	2.94×10^{1}	+	5.01×10^{-3}	+	**0**
f_5	-1.48×10^{-3}	+	-6.51×10^{-12}	+	-6.98×10^{-162}	+	−2.49	+	3.01×10^{-162}	+	$\mathbf{-9.88\times10^{-324}}$
f_6	2.06	+	1.51×10^{1}	+	**0**	~	7.57×10^{1}	+	2.09	+	**0**
f_7	4.45×10^{-2}	+	1.85	+	**0**	~	9.99×10^{-1}	+	1.86×10^{-1}	+	**0**
f_8	1.14×10^{-3}	+	3.83×10^{-1}	+	**0**	~	2.05×10^{-2}	+	1.26×10^{-2}	+	**0**
f_9	2.99	+	1.51×10^{1}	+	**0**	~	4.14×10^{1}	+	4.11	+	**0**
f_{10}	2.81×10^{-2}	+	2.98	+	**0**	−	1.97	+	3.41×10^{-1}	+	6.63×10^{-15}
f_{11}	7.95×10^{-4}	+	2.33×10^{-2}	+	**0**	~	6.14×10^{-1}	+	1.22×10^{-2}	+	**0**
f_{12}	4.68	+	1.35×10^{1}	+	**0**	~	4.07	+	1.76	+	**0**
f_{13}	4.25×10^{-1}	+	2.98	+	**0**	−	2.06×10^{1}	+	1.34×10^{-1}	+	3.55×10^{-15}
f_{14}	7.69	+	1.95×10^{-2}	+	**0**	~	1.02×10^{3}	+	1.44×10^{-2}	+	**0**
总计 +	13		13		4		14		12		—
总计 ~	1		1		8		0		1		—
总计 −	0		0		2		0		1		—

中都取得了较好的结果。实验结果显示，OPSO-AEM&NIW 在所有测试函数上均优于 OVCPSO；与 PSO 和 OPSO 比较，除在 f_3 上处取得相当结果，OPSO-AEM&NIW 在其他函数上均取得较优解；EOPSO 虽然在 f_3 上取最优解，但在其他 12 个函数上均劣于 OPSO-AEM&NIW。特别地，OPSO-AEM&NIW 与 GOPSO 在大多数函数中取得了相当的最优解，但在 4 个测试函数 f_2~f_5 上，OPSO-AEM&NIW 优于 GOPSO。

为进一步比较 OPSO-AEM&NIW 与 GOPSO 这 2 种算法性能上的差异，2 类实验分别展开。首先除去平均适应值(表 3 中已记录)，将 2 种算法运行 30 次后的另外 3 个性能指标：标准方差、最佳及最差值分别记录在表 4 中，结果显示，OPSO-AEM&NIW 在 12 个测试函数中的最差值均优于 GOPSO，说明新算法相对 GOPSO 性能整体较为稳定。另外，2 种算法在求解精度为 10^{-16} 以及最大迭代次数为 10 000 次的条件下再次进行实验，获得的适应值(*fval*)及适应值函数评估次数(*NFC*)如表 5 所示。通过对 *NFC* 值的比较可见，OPSO-AEM&NIW (除 f_{10} 和 f_{13})明显小于 GOPSO，实验结果表明 OPSO-AEM&NIW 大大提升了反向 PSO 算法的收敛速度。

表 5　GOPSO 和 OPSO-AEM&NIW 算法在 14 个测试函数上函数评估次数比较

测试函数	GOPSO		OPSO-AEM&NIW	
	适应值	评估次数	适应值	评估次数
f_1	9.31×10^{-17}	47 556	4.27×10^{-17}	**10 377**
f_2	0	11 466	0	**2 846**
f_3	2.38×10^{1}	400 080	1.74	400 080
f_3	9.97×10^{-17}	254 778	3.18×10^{-17}	**14 201**
f_5	9.87×10^{-17}	85 230	5.96×10^{-17}	**18 634**
f_6	0	98 481	0	**10 195**
f_7	0	93 746	0	**15 019**
f_8	0	43 388	0	**10 827**
f_9	0	125 458	0	**9 850**
f_{10}	0	79 714	3.55×10^{-15}	817 901
f_{11}	0	47 401	0	**10 965**
f_{12}	0	51 078	0	**8 993**
f_{13}	0	89 212	3.55×10^{-15}	822 750
f_{14}	0	47 886	0	**10 083**

表 4　GOPSO 和 OPSO-AEM&NIW 算法在 14 个测试函数上的运行结果

测试函数	GOPSO			OPSO-AEM&NIW		
	标准方差	最佳值	最差值	标准方差	最佳值	最差值
f_1	0	0	7.41×10^{-323}	0	0	**0**
f_2	0	0	9.06×10^{-320}	0	0	**0**
f_3	**1.36**	2.46×10^{1}	2.90×10^{1}	1.10×10^{1}	$\mathbf{2.92\times10^{-3}}$	2.89×10^{1}
f_3	2.02×10^{4}	1.37×10^{-100}	6.01×10^{4}	0	0	$\mathbf{9.88\times10^{-324}}$
f_5	2.91×10^{-161}	2.38×10^{-175}	1.56×10^{-160}	**0**	**0**	$\mathbf{8.89\times10^{-323}}$
f_6	0	0	0	0	0	0
f_7	0	0	0	0	0	0
f_8	0	0	0	0	0	0
f_9	0	0	0	0	0	0
f_{10}	**0**	0	**0**	1.09×10^{-14}	0	4.62×10^{-14}
f_{11}	0	0	0	0	0	0
f_{12}	0	0	0	0	0	0
f_{13}	**0**	0	**0**	1.83×10^{-15}	0	7.11×10^{-15}
f_{14}	0	0	0	0	0	0

4.2.2 2 种策略有效性分析

为研究自适应精英变异策略与非线性自适应惯性策略在融入到反向学习(GOBL)中后对反向 PSO 算法性能所产生的影响，以测试函数 f_2、f_9 与 f_{14} 为例，将 AEM 与 NIW 这 2 种策略分别与 GOBL 结合，形成 GOBL+AEM 和 GOBL+NIW 两类算法，与新算法 OPSO-AEM&NIW 展开性能对比实验。特别指出，在 GOBL+ AEM 实验中，w 取固定值 0.729 84[19]。实验结果如图 2 所示，NIW 策略能够充分调动粒子在不同阶段的活跃性，有效平衡算法的全局探索与局部开采能力，加快了算法的收敛速度；AEM 策略则通过对精英粒子 *gbest* 的扰动，有效避免粒子陷入局部最优的可能性，确保算法得到快速平滑收敛。

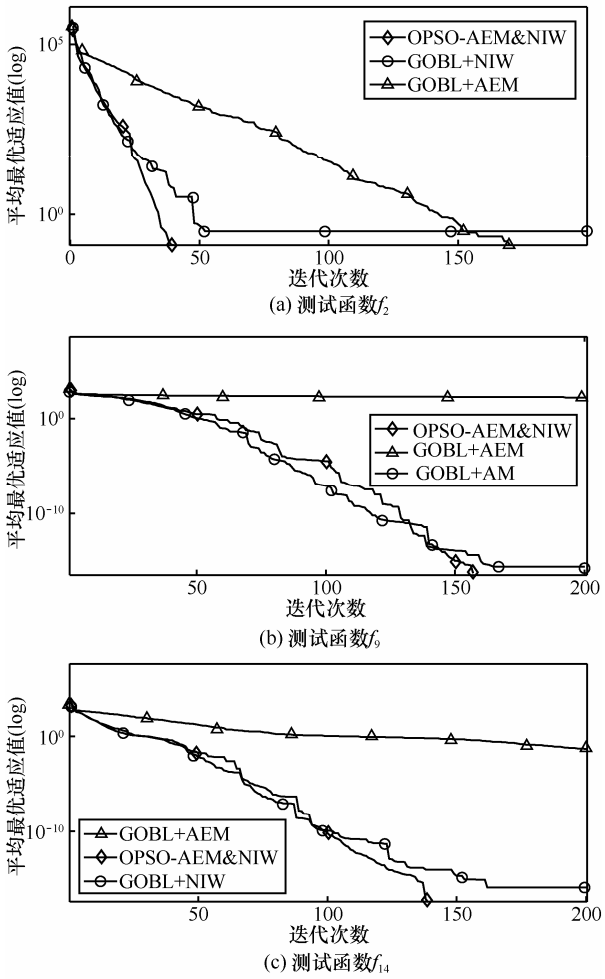


图 2 AEM 与 NIW 2 种策略对算法的影响

为从另一侧面了解上述 2 种策略引入到反向粒子群中为避免算法陷入局部最优所起到的作用，表 6 分别记录了 OPSO 算法[7]与 OPSO-AEM&NIW 算法在 30 次运行中陷入局部最优的平均次数(MNLO, mean number of local optimum)，以及对应全局最优结果的均值(*mean*)。实验结果表明，除 f_4 以外，OPSO-AEM&NIW 在其他测试函数上的 *MNLO* 值均少于 OPSO 的 *MNLO* 值，表明 AEM 与 NIW 策略能使粒子在进化过程中有效地避免陷入局部最优的次数，一旦发生，也能采用自适应精英变异策略更成功地跳出局部最优，从而快速地收敛到全局最优值，并表现出较好的稳定性。同时注意到，虽然 f_4 的 *MNLO* 在 OPSO 中更小，但它并没有收敛到全局最优，原因是 OPSO 算法在处于某个局部最优时无法通过自身的变异操作跳出，进而出现早熟收敛现象。

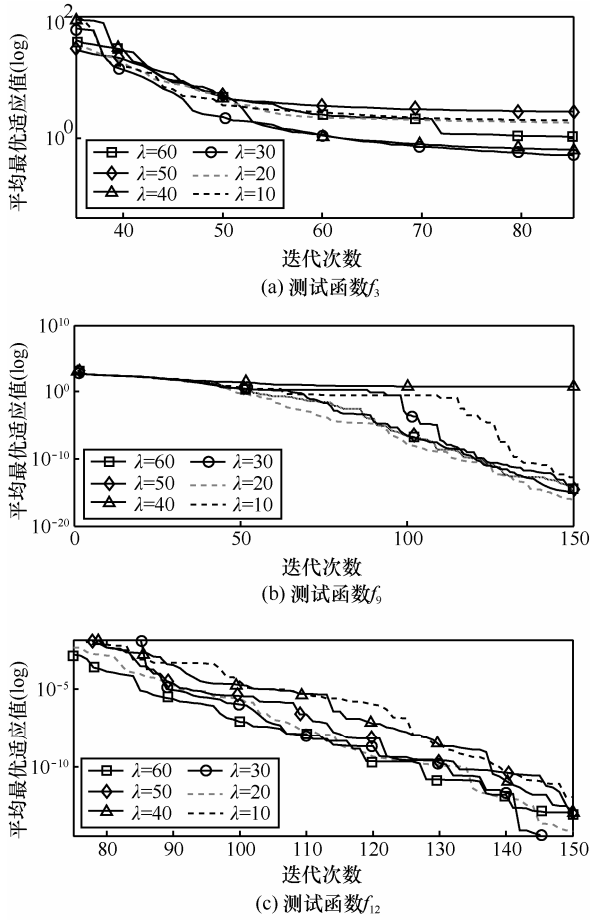
表 6 AEM 与 NIW 策略为避免陷入局部最优的有效性分析

测试函数	OPSO		OPSO-AEM&NIW	
	mean	*MNLO*	*mean*	*MNLO*
f_1	4.59×10^{-36}	9.09×10^{2}	0	**7.23**
f_2	2.09×10^{-35}	3.37×10^{1}	0	**7.43**
f_3	7.18	8.29×10^{2}	6.94	**5.97**
f_4	4.13×10^{4}	$\mathbf{2.67\times10^{-1}}$	0	1.07
f_5	-6.51×10^{-12}	7.41×10^{2}	-9.88×10^{-324}	**1.07**
f_6	1.51×10^{1}	3.86×10^{2}	0	$\mathbf{6.33\times10^{-1}}$
f_7	1.85	3.20×10^{2}	0	**3.60**
f_8	3.83×10^{-1}	3.91×10^{2}	0	**9.87**
f_9	1.51×10^{1}	3.91×10^{2}	0	$\mathbf{3.33\times10^{-1}}$
f_{10}	2.98	3.30×10^{2}	6.63×10^{-15}	**3.33**
f_{11}	2.33×10^{-2}	3.98×10^{2}	0	$\mathbf{1.21\times10^{1}}$
f_{12}	1.35×10^{1}	5.68×10^{2}	0	**2.12**
f_{13}	2.98	2.11×10^{2}	3.55×10^{-15}	**1.13**
f_{14}	1.95×10^{-2}	2.07×10^{2}	0	$\mathbf{3.41\times10^{1}}$

4.2.3 参数敏感性分析

通过策略分析，对 3 种策略(GOBL、AEM 与 NIW)在 OPSO-AEM&NIW 算法基本 PSO 性能改进中起到的作用有了清晰的了解。策略中各个参数的设置对算法的性能往往有着很大的影响，文献[7]中已通过实验得出结论：当 OBL 使用概率 *jr* 为 0.3 时，算法达到最佳性能，本文引用了这一结论。

为检验 AEM 策略中参数 λ 的取值对算法性能是否具有一定的影响，通过对 14 个测试函数上 λ 在 10~60 之间取不同的 6 个整数值时算法 OPSO-AEM&NIW 收敛过程的观测可知，当 λ=30 时，所有测试问题都能较平稳而快速地收敛到全局最优解。介于篇幅的限制，图 3 仅给出了在测试函数 f_3、f_9 和 f_{12} 中 OPSO-AEM&NIW 参数 λ 取不同值时算法收敛到全局最优值的进化过程。



(a) 测试函数f_3

(b) 测试函数f_9

(c) 测试函数f_{12}

图 3　λ 在不同取值下 OPSO-AEM&NIW 全局收敛过程

除此之外，表 7 统计了 AEM 策略的另一个待定参数 C 对 14 个测试函数在 30 次实验中自适应取不同常量值时各自的平均概率情况。其中，*Mut* 为变异位占优的次数。特别注意，当 C=0.5 时，变异式(11)将退化成柯西变异[7]。实验数据表明，相比于柯西变异，自适应精英变异策略将增加 25%的概率获得更优的变异位，虽然这看似远小于 C=0.5 的概率，但其对于算法是否能成功跳出局部最优位是非常关键的。

由 4.2.2 节分析可知，NIW 策略的引入相对于 w 取固定值时能显著加快算法的收敛速度，而对于其中参数 w 取值范围(w_{max} 与 w_{min})，通过反复实验可知，当 w 取值在 0.2~0.5 之间时，算法在所有测试函数上取得最好结果。图 4 以 f_6 为例，展示了算法进化前 1 000 代过程中 w 自适应取值的变化情况。由图 4 可知，随着进化的深入，w 取值将趋于 0.5，使算法以较大的惯性快速收敛到全局最优值。

表 7　AEM 变异策略在 14 个测试函数上取值概率

测试函数	C=0.5	C=1.0	C=1.5	*Mut*
f_1	0.80	0.13	0.07	6
f_2	0.84	0.10	0.06	7
f_3	0.00	0.00	1.00	4
f_4	0.92	0.05	0.03	2
f_5	0.60	0.13	0.27	6
f_6	0.94	0.04	0.02	4
f_7	0.78	0.10	0.11	3
f_8	0.76	0.20	0.04	9
f_9	0.82	0.04	0.14	2
f_{10}	0.71	0.17	0.12	5
f_{11}	0.85	0.09	0.06	8
f_{12}	0.89	0.06	0.06	1
f_{13}	0.65	0.24	0.11	5
f_{14}	0.90	0.06	0.04	9
总体平均概率	0.75	0.10	0.15	5

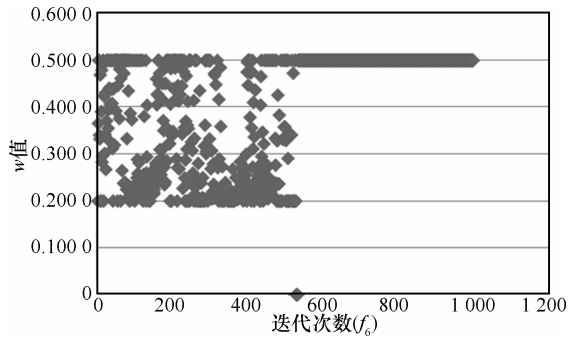


图 4　w 取值自适应变化情况

5　结束语

本文提出了一种带自适应精英扰动及惯性权重的反向粒子群优化算法。该算法将自适应精英变异和非线性自适应惯性权重 2 种策略融入到一般性反向学习策略中。为提高全局探索能力，GOBL 策略

在每代中对解空间与其反向解空间同时进行搜索。AEM 策略将全局最优粒子选为精英粒子，根据当前群体聚集程度的不同，自适应选取变异量，使精英粒子获得足够的扰动，并吸引其他粒子向其运动，从而帮助粒子跳出局部最优。NIW 策略的引入可进一步克服算法全局探索与局部开采之间的矛盾。它通过自适应获取惯性权值，根据需求调节各个粒子在不同阶段的活跃程度。进化初期，w 在一定区间内自适应地变化，以扩大算法的全局搜索空间；随着迭代次数的增加，w 取值逐渐趋于平稳，从而有效避免算法最优值的振荡，提高算法的局部开采能力，确保算法平滑快速地收敛到全局最优值。

通过对多种函数的测试结果表明，新算法在获得高精度解的同时，显著提高了算法的收敛速度。在未来的工作中，如何使算法适应更多的问题还需展开更多的实验。另外，算法在高维问题中是否适用及可能存在的问题还需进一步的实验。如何将算法与实际问题相结合是未来研究工作的重点。

参考文献：

[1] KENNEDY J, EBERHART R C. Particle swarm optimization[C]// IEEE International Conference on Neural Networks. Perth, Australia, 1995: 1942-1948.

[2] 史霄波, 张引, 赵杉, 等. 基于离散多目标优化粒子群算法的多移动代理协作规划[J]. 通信学报, 2016, 37(6): 29-37.
SHI X B, ZHANG Y, ZHAO S, et al. Discrete multi-objective optimization of particle swarm optimizer algorithm for multi-agents collaborative planning [J]. Journal on Communications, 2016, 37(6): 29-37.

[3] INBARANI H H, AZAR A T, JOTHI G. Supervised hybrid feature selection based on PSO and rough sets for medical diagnosis [J]. Computer Methods and Programs in Biomedicine, 2014, 113(1): 175-185.

[4] ZAD B B, HASANVAND H, LOBRY J, et al. Optimal reactive power control of DGs for voltage regulation of MV distribution systems using sensitivity analysis method and PSO algorithm [J]. International Journal of Electrical Power and Energy System, 2015, 68: 52-60.

[5] SHI Y, EBERHART R C. A modified particle swarm optimizer[C]//The IEEE Congress on Evolutionary Computation (CEC 1998). 1998: 69-73.

[6] TIZHOOSH H R. Opposition-based learning: a new scheme for machine intelligence[C]//The IEEE International Conference of Intelligent for Modeling, Control and Automation. PiscatNIWay: Inst of Elec. and Elec Eng Computer Society. 2005: 695-701.

[7] WANG H, LI H, LIU Y, et al. Opposition-based particle swarm algorithm with Cauchy mutation [C]//The IEEE Congress on Evolutionary Computation. 2007: 356-360.

[8] WANG H, WU Z J, RAHNAMAYAN S, et al. Enhancing particle swarm optimization using generalized opposition-based learning [J]. Information Sciences, 2011, 181: 4699-4714.

[9] 周新宇, 吴志健, 王晖, 等. 一种精英反向学习的粒子群优化算法[J]. 电子学报, 2013, 41(8): 1647-1652.
ZHOU X Y, WU Z J, WANG H, et al. Elite opposition-based particle swarm optimization[J]. Acta Electronica Sinica, 2013, 41(8): 1647-1652.

[10] SHAHZAD F, BAIG A R, MASOOD S, et al. Opposition-based particle swarm optimization with velocity clamping (OVCPSO) [J]. Advances in Computational Intell, 2009: 339-348.

[11] KAUCIC M. A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization[J]. J Glob Optim, 2013, 55: 165-188

[12] PEHLIVANOGLU Y V. A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks [J]. IEEE Trans Evol Comput. 2013, 17: 436-452.

[13] KARAFOTIAS G, HOOGENDOORN M, EIBEN A E. Parameter control in evolutionary algorithms: trends and challenges[J]. IEEE Transactions on Evolutionary Computation, 2015, 19: 167-187.

[14] 汪慎文, 丁立新, 谢承旺, 等. 应用精英反向学习策略的混合差分演化算法[M]. 武汉大学学报(理学版), 2013, 59(2): 111-116.
WANG S W, DING L X, XIE C W, et al. A hybrid differential evolution with elite opposition-based learning [J]. Journal of Wuhan University, 2013, 59(2): 111-116.

[15] OZCAN E, MOHAN C K. particle swarm optimization: surfing and waves[C]//Congress on Evolutionary Computation (CEC1999). 1999: 1939-1944.

[16] 龚纯, 王正林. 精通 MATLAB 最优化计算[M]. 电子工业出版社, 2012: 283-285.
GONG C, WANG Z L. Proficient optimization calculation in MATLAB [M]. Electronic Industry Press, 2012: 283-285.

[17] FRANS V D B. An analysis of particle swarm optimizers[D]. Department of Computer Science, University of Pretoria, South Africa, 2002.

[18] TANG K, LI X D, SUGANTHAN P N, et al. Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization[R]. Nature Inspired Computation and Applications Laboratory, USTC, China, 20092, 1.

[19] BERGH F, ENGELBRECHT A P. Effect of swarm size on cooperative particle swarm optimizers[C]//Genetic and Evolutionary Computation Conference. 2001: 892-899.

作者简介：

董文永（1973-），男，河南南阳人，博士，武汉大学教授、博士生导师，主要研究方向为演化计算、智能仿真优化、系统控制、机器学习等。

康岚兰（1979-），女，江西赣州人，武汉大学博士生，江西理工大学讲师，主要研究方向为演化计算、机器学习等。

刘宇航（1979-），男，湖北孝感人，武汉大学博士生，主要研究方向为统计机器学习及相关应用。

李康顺（1962-），男，江西兴国人，博士，华南农业大学教授、博士生导师，主要研究方向为智能计算、多目标优化、视频流图像识别等。