文章编号: 1001-0920 (2016) 07-1161-08 **DOI:** 10.13195/j.kzyjc.2015.0711

一种改进的全局粒子群优化算法

王 皓[1,2], 欧阳海滨[1], 高立群[1]

(1. 东北大学 信息科学与工程学院，沈阳 110004；2. 辽宁省交通高等专科学校 信息工程系，沈阳 110122)

摘 要: 为了改善粒子群优化算法的优化性能, 提出一种改进的全局粒子群优化 (IGPSO) 算法. 该算法基于开采能力和搜索能力相均衡的思想提出全局邻域搜索策略和扰动策略, 使算法减少陷入局部极值的可能性, 同时以一定概率对全局最优粒子进行摄动操作, 加快算法收敛. 与其他智能算法相比较, 测试结果从寻优精度、收敛速度和非参数统计显著性方面验证了 IGPSO 算法的有效性.

关键词: 粒子群优化算法；开采能力；搜索能力；收敛速度；显著性

中图分类号: TP301.6 **文献标志码:** A

An improved global particle swarm optimization

WANG Hao[1,2], *OUYANG Hai-bin*[1], *GAO Li-qun*[1]

(1. College of Information Science and Technology，Northeastern University，Shenyang 110004，China；2. Information Engineering Department，Liaoning Provincial College of Communications，Shenyang 110122，China. Correspondent: OUYANG Hai-bin，E-mail：oyhb1987@163.com)

Abstract: In order to improve the performance of particle swarm optimization(PSO) algorithm, an improved global particle swarm optimization(IGPSO) is presented. Based on a balance between exploitation and exploration ability, the global neighborhood search strategy and disturbance strategy are proposed to reduce the possibility of falling into local minima. Meanwhile, a perturbation operation with probabilities is implemented in the global best particle, which aims at accelerating the convergence speed. The test results demonstrate the effectiveness of the IGPSO algorithm in terms of accuracy, convergence speed, and nonparametric statistical significance when compared with other state-of-the-art intelligent algorithms.

Keywords: particle swarm optimization；exploitation ability；exploration ability；convergence speed；significance

0 引 言

粒子群优化算法 (PSO) 是 Kennedy 等[1]受到鸟群飞行及觅食行为的启发提出的一种智能优化算法. 该算法结构简单, 可操作性强, 便于实现, 得到了许多学者的关注和研究. 粒子群算法至今已成功地应用到许多复杂的实际优化问题中, 如随机作业车间调度问题[2]、太阳能光伏系统[3]、经济调度问题[4]、大规模社交网络聚类[5]. 然而, 粒子群算法存在易陷入局部最优的不足, 为了克服这一缺点, 学者们提出了各种改进的粒子群优化算法: 一是关于参数调整的研究, 惯性权重的调整[6]包括动态策略和自适应方法, 学习因子和社会因子的调整[7]包括经验调试、理论设计和自适应设计; 二是改进搜索策略或增加辅助操作, 如改进速度更新和位置更新、引入局部搜索操作、选择操作、交叉操作等[8]; 三是与其他算法相融合, 如粒子群优化算法与差分进化算法、分布估计算法、人工蜂群算法的结合[9].

为了较好地平衡 PSO 算法的开采能力和搜索能力, 本文提出一种改进的全局粒子群优化算法 (IGPSO). IGPSO 算法主要对 PSO 算法作了 3 方面改进: 一是提出全局邻域搜索策略, 加强当前种群邻域的勘探; 二是对粒子历史最优种群进行全局邻域扰动, 改善算法跳出局部最优能力; 三是基于优秀个体重点培养的思想, 以动态概率对全局最优解进行摄动操作, 有效地防止算法早熟, 提高算法对解空间的开发能力. 与其他智能算法相比较, 测试结果从寻优精度、收敛速度和非参数统计显著性方面验证了 IGPSO 算法的有效性.

收稿日期: 2015-06-04；**修回日期:** 2015-10-29.

基金项目: 国家自然科学基金项目(61403174)；辽宁省博士科研启动基金项目(201205118).

作者简介: 王皓(1981−), 女, 博士生, 从事智能优化算法及其应用的研究；高立群(1949−), 男, 教授, 博士生导师, 从事智能优化算法及图像处理等研究.

1 粒子群优化算法

PSO 算法包含速度更新和位置更新两个重要的操作, 分别为

$$v_{i,j}^{g+1}=\omega(g)v_{i,j}^{g}+c_1r_1(p_{i_{\text{best}},j}-x_{i,j}^{g})+ c_2r_2(p_{g_{\text{best}},j}-x_{i,j}^{g}),\tag{1}$$

$$x_{i,j}^{g+1}=x_{i,j}^{g}+v_{i,j}^{g+1}.\tag{2}$$

其中: $v_{i,j}^{g}$ 为第 g 次迭代中第 i 个粒子的第 j 个变量, $v_{i,j}^{g+1}$ 为第 $g+1$ 次迭代中第 i 个粒子的第 j 个变量, c_1 和 c_2 为认知因子和社会学习因子, $p_{i_{\text{best}},j}$ 为第 i 个个体最优粒子的第 j 个变量, $p_{g_{\text{best}},j}$ 为全局最优粒子的第 j 个变量, r_1 和 r_2 都是 0～1 之间均匀分布的随机数.

2 改进的全局粒子群优化算法

针对粒子群算法容易陷入局部最优和收敛速度慢的不足, 本文提出一种改进的全局粒子群优化(IGPSO) 算法. 下面具体介绍 3 种改进策略, 分别是全局邻域搜索策略、全局邻域扰动策略和最优粒子摄动策略.

2.1 全局邻域搜索

原始的 PSO 算法被认为是全局版本的粒子群优化算法, 它单纯地依靠整个种群的全局最优个体的指导进行寻优, 往往由于全局最优个体极易陷入某一邻域而导致算法过早地收敛或早熟[10]. 为了降低 PSO 算法陷入局部极值的可能性, 本文提出一种新的速度更新公式, 命名为全局邻域搜索, 即加强全局最优个体附近邻域中随机个体的指导作用. 全局最优个体的邻域中随机个体相对于确定的全局最优个体更有益于 PSO 算法跳出局部极值. 全局邻域搜索策略为

$$v_{i,j}^{g+1}=\omega_g v_{i,j}^{g}+c_1r_1(p_{i_{\text{best}},j}(1\pm\delta U(0,1))-x_{i,j}^{g})+ c_2r_2(p_{g_{\text{best}},j}(1\pm\delta U(0,1))-x_{i,j}^{g}),\tag{3}$$

$$\omega_g=a\exp(bg^2)\cdot\text{rand},\tag{4}$$

$$b=\frac{1}{G^2-1}\ln\left(\frac{\omega_{\max}}{\omega_{\min}}\right),\tag{5}$$

$$a=\omega_{\max}\exp(-b).\tag{6}$$

其中: c_1 和 c_2 为认知因子和社会学习因子, δ 为扰动因子, r_1、r_2、rand 和 $U(0,1)$ 均为区间 (0,1) 上的均匀分布随机数, g 为当前迭代次数, G 为最大迭代次数. 设 $\omega_{\max}$ 和 $\omega_{\min}$ 分别为 0.9 和 0.4, 迭代 1 000 次, 得到动态惯性权重曲线如图 1 所示.

结合算法的搜索进程可知, 在搜索早期, ω 取得较大值的概率应适当增加, 以加强算法的全局搜索, 到后期, 则适当减小 ω 值, 使 ω 取得较小值的概率大一些, 在整体上使 ω 值的变化呈现下降趋势, 这样算法的全局搜索和局部搜索能够达到一定的平衡, 有益于提高算法的寻优能力. 基于以上原因, 本文设计了一种动态惯性权重 (式 (4)～(6)). 图 1 表明惯性权重呈一种下降趋势, 但并不表示下次迭代的惯性权重肯定比上次迭代小, 这有利于平衡算法的全局搜索和局部搜索, 增加算法种群的多样性.

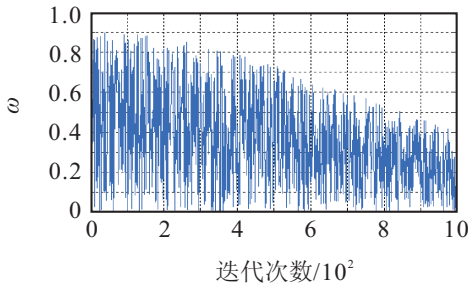

图 1 动态惯性权重

针对全局邻域搜索策略作进一步解释: 原始 PSO 算法的位置更新中, 新的粒子只由确定的个体最优位置和当前全局最优位置进行引导, 忽略了邻域区间可能存在的较优位置. 而全局邻域搜索加入了一定的邻域空间, 分别以个体最优位置和当前全局最优位置为中心, 原有值的 δ 倍为步长, 形成两个邻域区间 $[p_{i_{\text{best}},j}(1-\delta U(0,1))p_{i_{\text{best}},j}(1+\delta U(0,1))]$ 和 $[p_{g_{\text{best}},j}(1-\delta U(0,1))p_{g_{\text{best}},j}(1+\delta U(0,1))]$, 使算法在设定的区间进行邻域随机搜索, 有效增加了寻优范围.

2.2 全局邻域扰动

在 PSO 算法搜索中, 无论子代粒子是否优于父代粒子, 每个粒子的子代粒子始终都会取代对应的父代粒子, 而依靠粒子历史最优和全局最优粒子的邻域产生的子代, 在一定程度上会聚集于粒子历史最优和全局最优粒子的邻域, 这样整个种群的多样性仍然会迅速随着迭代的进行而降低, 导致搜索空间变小. 因此, 需要对全局邻域进行独立调整, 本质上就是针对粒子历史最优种群进行扰动, 拓宽搜索的空间, 增加种群的多样性.

大多数粒子群算法的改进都是针对当前种群进行的, 本文考虑到全局邻域搜索策略中粒子历史最优和全局最优粒子邻域的引导作用, 为了增强粒子历史最优种群的指导作用, 提出一种全局邻域扰动策略, 具体如下:

$$y_{i_{\text{best}},j}^{g+1}=\begin{cases}p_{i_{\text{best}},j}^{g}+GN(0,1)(p_{m_1\text{best},j}^{g}-p_{m_2\text{best},j}^{g}),\\ \quad f(p_{m_1\text{best}}^{g})\leqslant f(p_{m_2\text{best}}^{g});\\ p_{i_{\text{best}},j}^{g}+GN(0,1)(p_{m_2\text{best},j}^{g}-p_{m_1\text{best},j}^{g}),\\ \quad f(p_{m_2\text{best}}^{g})<f(p_{m_1\text{best}}^{g}).\end{cases}$$

其中: $f(x)$ 为 x 的目标函数值; m_1、$m_2\in 1,2,\cdots,$

NP, NP为种群数目; $y_{i_{\rm best},j}^{g+1}$ 为第 $g+1$ 代第 i 个候选的历史最优粒子的第 j 维, $y_{i_{\rm best},j}^{g}$ 为第 g 代第 i 个历史最优粒子的第 j 维, GN $(0,1)$ 为均值是0、方差是1的高斯分布随机数. 如果 $y_{i_{\rm best}}^{g}$ 的目标函数值优于 $p_{i_{\rm best}}^{g}$, $y_{i_{\rm best}}^{g}$ 则取代 $p_{i_{\rm best}}^{g}$, 否则保持 $p_{i_{\rm best}}^{g}$ 不变.

2.3 全局最优粒子学习

全局最优粒子引导整个种群向最好的方向迁徙, 但是由于最优粒子自身缺乏先验知识和自我学习能力, 导致最优粒子自身难以得到有效提升, 从而陷入局部极值的邻域, 难以进一步搜索到更优的方向. 最优粒子得不到有效更新而使得整个种群的搜索能力降低, 所以迫切需要独立对全局最优粒子进行挖掘. 本文考虑满足一定概率条件下, 对全局最优粒子进行探索, 并提出随机学习策略和反向学习策略两种方式对全局最优粒子进行有效的学习, 以提高全局最优粒子的自我学习能力.

1) 随机学习策略.

全局最优粒子是当前种群中最优的粒子, 如果仍然向当前种群学习, 则难以得到提高. 全局最优粒子需要拓展到当前种群以外的搜索空间, 所以提出一种随机学习策略, 具体表达如下:

$$u_{g_{\rm best},j}=p_{g_{\rm best},j}+\varphi(p_{g_{\rm best},j}-x_{a,j}). \tag{7}$$

其中: $x_{a,j}$ 为随机产生粒子的第 j 维变量, φ 为$-1\sim 1$之间的均匀分布随机数, $u_{g_{\rm best},j}$ 为候选全局最优粒子的第 j 维变量. 通过随机产生的粒子促进最优粒子的广泛学习, 增加搜索的空间.

2) 反向学习策略.

反向学习技术是一种新的数值智能计算结构, 最初由 Tizhoosh[11]提出. 该技术寻找一个解及其反方向的解, 将两个解进行比较, 选出较优的解. 这样反复地跳跃性进化, 增强解空间的开发, 并逐渐寻找到最优解. 反向学习具体定义如下:

设 $P=(x_1,x_2,\cdots,x_D)$ 为 D 维空间的一个点, $x_1,x_2,\cdots,x_n\in \boldsymbol{R}$, $x_i\in[a_i,b_i]$, 则全局反向点 $\tilde{P}=(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_D)$ 定义为

$$\tilde{x}_i=a_i+b_i-x_i. \tag{8}$$

基于这种思想, 本文针对全局最优粒子应用反向学习策略, 表达式如下:

$$\begin{aligned}&u_{g_{\rm best},j}=\alpha b_1+\beta b_2,\\&b_1=x_{j,L}+\max(x_j)-p_{g_{\rm best},j},\\&b_2=x_{j,U}+\min(x_j)-p_{g_{\rm best},j}.\end{aligned} \tag{9}$$

其中: $\alpha+\beta=1$, $x_{j,L}$ 和 $x_{j,U}$ 分别为粒子第 j 维变量的上下限, $\max(x_j)$ 和 $\min(x_j)$ 分别为所有粒子第 j 维的最大值和最小值. 反向学习原理如图2所示.

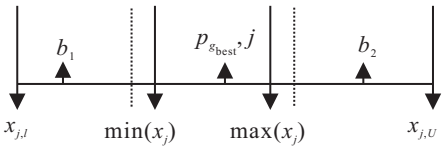

图 2　反向学习原理

由式(9)可知, b_1 和 $p_{g_{\rm best},j}$ 关于 $x_{j,L}$ 和 $\max(x_j)$ 的中点对称, b_2 和 $p_{g_{\rm best},j}$ 关于 $x_{j,U}$ 和 $\min(x_j)$的中点对称, 因此在图2中, b_1 和 $p_{g_{\rm best},j}$ 关于虚线1对称, b_2 和 $p_{g_{\rm best},j}$ 关于虚线2对称, 候选全局最优粒子 $u_{g_{\rm best},j}$ 通过协同两种不同区间的反向学习粒子得到, 有效增加搜索空间, 提高跳出局部极值的可能性. 在本文算法中, 利用参数 τ 确定全局最优粒子学习的进行, 即当随机数小于 τ 时, 本文算法进行全局最优粒子学习操作, 这样不仅保证当前粒子种群和历史最优粒子种群的更新, 而且在一定概率上调整全局最优粒子, 以便更好地引导整个种群向更优的方向进行搜索.

2.4 算法流程

本文算法流程如下.

Step 1: 初始化算法和问题参数. 认知因子 $c_1=2$, 社会学习因子 $c_2=2$, 扰动因子 δ, 惯性权重的最大值 $\omega_{\max}=0.9$, 惯性权重的最小值 $\omega_{\min}=0.4$, 全局最优粒子调整概率 τ, 参数 α、β 问题维数 D, 每一维变量的上下限 $x_{j,U}$ 和 $x_{j,L}$, 最大迭代次数 G, 当前迭代次数 g.

Step 2: 种群初始化. 根据每一维变量的上下限初始化每个粒子, 得到初始化种群.

Step 3: 针对每个粒子进行全局邻域搜索, 基于速度更新和位置更新, 最终得到一个新的种群.

Step 4: 针对每个粒子的历史最优粒子进行全局邻域扰动, 产生一个新的历史最优粒子种群.

Step 5: 判定随机数是否小于 τ, 如果小于 τ 则进行全局最优粒子学习操作, 分别进行随机学习和反向学习.

Step 6: 判断迭代次数是否满足设定的最大值, 如果满足, 则迭代终止, 否则循环进行 Step 3 ~ Step 5.

Step 7: 统计搜索结果, 保存优化结果.

3 实验分析

为测试IGPSO算法的优化性能, 考虑以下几方面: 1) 算法的寻优精度, 即在有限的目标函数评价次数条件下, 算法寻找到的最优值; 2) 算法的收敛速度, 即在给定足够大的目标函数评价次数条件下, 搜索到给定精度所需要的评价次数, 评价次数越小收敛速度越快; 3) 算法的非参数统计显著性, 如 t-测试等.

实验选用12个经典的测试函数如表1所示. 其中: Rosenbrock是多变量相关的函数, 最优值往往难

表 1 12 个无约束标准函数

函数名	函数模型	初始范围	$f_{\min}$
Sphere	$f_1=\sum_{i=1}^{D}x_i^2$	$[-100, 100]$	0
Schwefel 2.22	$f_2=\sum_{i=1}^{D}\|x_i\|+\prod_{i=1}^{D}\|x_i\|$	$[-100, 100]$	0
Schwefel 1.2	$f_3=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	$[-100, 100]$	0
Schwefel 2.21	$f_4=\max\{\|x_i\|,\ 1\leqslant i\leqslant D\}$	$[-100, 100]$	0
Rosenbrock	$f_5=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i)^2+(x_i-1)^2]$	$[-30, 30]$	0
Step	$f_6=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	$[-100, 100]$	0
Quartic	$f_7=\sum_{i=1}^{D}ix_i^4+\mathrm{random}[0,1)$	$[-1.28, 1.28]$	0
Schwefel	$f_8=\sum_{i=1}^{D}-x_i\times\sin\sqrt{\|x_i\|}$	$[-500, 500]$	$-12\,569$
Rastrigin	$f_9=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12, 5.12]$	0
Ackley	$f_{10}=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)+20+e$	$[-32, 32]$	0
Griewank	$f_{11}=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\frac{x_i}{\sqrt{i}}+1$	$[-600, 600]$	0
Penalized	$f_{12}=\frac{\pi}{D}\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{D}u(x_i,10,100,4)y_i=1+\frac{1}{4}(x_i+1)$	$[-50, 50]$	0

以搜索到; Quartic 是带有噪声的函数, 其最优值随着随机均匀分布数 random 的改变而改变; Penalized 是多峰难测函数, 随维数的增加, 局部极小值也不断增加. 因此, 这些函数能够较好地测试算法的优化性能.

3.1 算法的寻优精度比较

通过 12 个常用的测试函数, 对 IGPSO 算法的寻优性能进行评价, 并与一些经典的优化算法, 如粒子群优化 (PSO) 算法[1]、自适应粒子群优化 (APSO) 算法[7]、学习粒子群优化 (CLPSO) 算法[12]、差分进化 (DE) 算法[13]、带有外部存档的差分进化算法 (JADE)[14]、全局最好和声搜索 (GHS) 算法[15]、最优引导的人工蜂群 (GABC) 算法[16]、布谷鸟搜索算法 (CS)[17]进行比较. 为了公平比较, 所有算法都设置相同的最大目标函数评价次数 $\mathrm{MAX_{FEs}}=1.2\times10^5$, 进行比较的算法参数均按照文献提供的最优参数设置. 本文算法中, $c_1=2$, $c_2=2$, 种群大小 NP $=20$, $\omega_{\max}=0.9$, $\omega_{\min}=0.4$, 扰动因子 $\delta=0.01$, $\tau=0.75$, α 为 0 ∼ 1 之间的随机数, $\beta=1-\alpha$. 函数的维数设置为 $N=30$, 各个算法对每个函数都独立运行 40 次, 优化结果如表 2 所示.

由表 2 可见, PSO 算法在优化 12 个函数时几乎都陷入了局部极值, 优化结果是所有比较算法中最差的. CLPSO算法引入了一种学习策略, 有效改善了 PSO 算法的性能. APSO 算法设计一种模糊隶属度函数评估算法的搜索状态, 主要分为探索、开采、收敛、跳出局部极值, 在不同的状态设置不同的参数调整方式, 提高了 PSO 逃离局部极值的能力. 但是, 这两种算法的优化精度并不令人满意. DE 算法和 JADE 算法在寻优能力上具有较强的勘探能力, 尤其 JADE 算法通过参数的自学习策略自适应地提高搜索的有效性, 是一种优秀的进化算法. GHS 引入全局最优粒子指导和声搜索算法的即兴创作, 文献 [15] 结果表明, GHS 算法优于和声搜索 (HS) 算法和改进的和声搜索 (IHS) 算法, 但是 GHS 算法也仅仅依靠全局最优粒子指导, 难以有效跳出局部极值. GABC 算法同样加入了全局最优粒子指导的操作, 虽然相对于 ABC 算法, 文献 [16] 显示 GABC 优化的结果是较优的, 但由于全局最优粒子的自学习能力差, 最终优化的结果也并不令人满意. CS 算法是新提出的一种智能算法, 其优化结果依赖于所设计的 Lévy 概率模型. 本文所提出的算法 IGPSO 结合了全局最优粒子的指导作用和其本身的自学习, 结果表明, 除了 f_5 函数 IGPSO 所取得的结果没有 JADE 算法好, f_{12} 函数 IGPSO 没有 ICLPSO 算法好外, IGPSO 算法无论在最好值、平均值、方差, 还是在最差值、中间值方面都要优于 PSO、APSO、CLPSO、DE、JADE、GHS、

表 2　优化结果的比较 ($N = 30$) (一)

函数	指标	PSO	APSO	CLPSO	DE	JADE	GHS	GABC	CS	IGPSO
f_1	时间	5.38e+00	9.25e+00	6.64e+00	6.75e+00	6.84e+00	1.85e+01	2.38e+00	**2.27e+00**	3.63e+00
	最好值	3.27e+04	5.80e-21	3.31e-82	2.40e-21	2.57e-49	2.20e-05	5.11e-09	3.83e-26	**4.01e-178**
	中间值	3.29e+04	7.12e-21	2.38e-45	1.58e-20	3.49e-47	5.37e-04	1.52e+02	9.03e-24	**1.61e-173**
	最差值	3.39e+04	1.42e-20	5.48e-24	1.07e-19	1.15e-46	1.18e-03	3.87e+02	1.80e-23	**7.87e-154**
	平均值	3.31e+04	8.55e-21	1.37e-24	3.54e-20	4.63e-47	5.70e-04	1.73e+02	9.03e-24	**1.97e-154**
	方差	5.27e+02	3.87e-21	2.74e-24	4.85e-20	5.62e-47	5.00e-04	2.02e+02	1.27e-23	**3.93e-154**
f_2	时间	5.09e+00	8.97e+00	6.68e+00	7.01e+00	6.19e+00	6.82e+00	**2.54e+00**	2.59e+00	4.20e+00
	最好值	4.50e+33	3.77e-11	1.46e-11	1.64e-11	5.00e-06	1.03e-01	1.10e+01	1.15e-10	**3.13e-86**
	中间值	6.94e+34	8.79e-11	9.21e-03	4.64e-11	5.70e-06	1.94e-01	4.12e+01	4.24e-10	**1.30e-85**
	最差值	4.61e+38	1.07e-10	7.44e-02	5.57e-11	7.92e-06	3.71e-01	4.54e+01	7.34e-10	**3.70e-82**
	平均值	1.15e+38	8.01e-11	2.32e-02	4.12e-11	6.08e-06	2.16e-01	3.47e+01	4.24e-10	**9.26e-83**
	方差	2.30e+38	3.06e-11	3.50e-02	1.87e-11	1.28e-06	1.12e-01	1.60e+01	4.38e-10	1.85e-82
f_3	时间	7.64e+00	9.49e+00	7.59e+00	1.51e+01	7.30e+00	1.47e+01	6.76e+00	**4.02e+00**	6.39e+00
	最好值	5.34e+04	5.58e+02	1.75e-05	3.64e-01	3.51e-19	7.16e+00	3.95e+02	2.98e-26	**1.33e-190**
	中间值	5.54e+04	7.46e+02	1.79e-04	7.16e-01	8.76e-18	2.86e+01	1.45e+04	3.52e-23	**2.57e-175**
	最差值	5.90e+04	9.24e+02	5.20e-04	1.83e+00	1.16e-15	9.58e+02	1.85e+04	7.04e-23	**3.64e-163**
	平均值	5.58e+04	7.43e+02	2.24e-04	9.06e-01	2.94e-16	2.56e+02	1.20e+04	3.52e-23	**9.10e-164**
	方差	2.33e+03	1.83e+02	2.12e-04	6.73e-01	5.77e-16	4.68e+02	8.24e+03	4.97e-23	**0.00e+00**
f_4	时间	8.30e+00	8.94e+00	6.62e+00	6.76e+00	6.57e+00	6.48e+00	**2.38e+00**	4.22e+00	6.45e+00
	最好值	3.02e+04	3.02e+00	1.65e-02	5.96e+00	1.16e-15	1.02e+01	2.57e+01	1.55e+00	**0.00e+00**
	中间值	3.51e+04	3.27e+00	2.59e-02	1.81e+01	3.22e-14	1.69e+01	2.73e+01	3.20e+01	**0.00e+00**
	最差值	3.91e+04	3.66e+00	8.19e-02	2.49e+01	1.12e-13	2.71e+01	3.36e+01	6.24e+01	**0.00e+00**
	平均值	3.49e+04	3.30e+00	3.75e-02	1.68e+01	4.43e-14	1.77e+01	2.85e+01	3.20e+01	**0.00e+00**
	方差	4.21e+03	2.66e-01	3.06e-02	9.22e+00	4.79e-14	7.72e+00	3.56e+00	4.30e+01	**0.00e+00**
f_5	时间	4.84e+00	9.48e+00	6.81e+00	7.55e+00	6.33e+00	7.24e+00	2.77e+00	**2.40e+00**	3.88e+00
	最好值	8.35e+07	2.28e+01	1.82e+00	2.15e+01	**4.86e-12**	2.93e-01	6.19e+01	1.89e+01	6.05e-04
	中间值	9.14e+07	7.73e+01	1.63e+01	2.42e+01	**3.39e-11**	1.49e+01	6.48e+03	1.95e+01	1.10e-03
	最差值	1.11e+08	8.20e+01	7.67e+01	2.85e+01	**8.40e-10**	2.86e+01	4.39e+04	2.00e+01	1.98e+01
	平均值	9.44e+07	6.49e+01	2.78e+01	2.46e+01	**2.28e-10**	1.47e+01	1.42e+04	1.95e+01	4.95e+00
	方差	1.29e+07	2.81e+01	3.35e+01	3.17e+00	**4.08e-10**	1.57e+01	2.07e+04	7.46e-01	9.89e+00
f_6	时间	5.00e+00	9.60e+00	6.67e+00	6.87e+00	6.19e+00	6.55e+00	2.51e+00	**2.46e+00**	3.97e+00
	最好值	2.79e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**0.00e+00**
	中间值	3.54e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**0.00e+00**
	最差值	4.25e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**0.00e+00**
	平均值	3.53e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**0.00e+00**
	方差	6.10e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**0.00e+00**

GABC、CS 算法. 这表明 IGPSO 算法寻优精度高, 具有良好的优化潜力. 此外, 本文算法的运行时间与其他智能算法相差不大.

3.2　算法的收敛速度比较

本节测试 IGPSO 算法的收敛速度. 设定各个函数的优化精度, 函数的维数取 30, 设定目标函数评价次数最大值为 3×10^5, 独立运行 50 次, 比较搜索到给定精度需要的最大评价次数 Maxi_FEs、最小评价次数 Mini_FEs、中间评价次数 Medi_FEs、平均评价次数 Mean_FEs、方差 SD_FEs、达到优化精度的成功率 SR. 成功率指的是寻优达到所给定精度的运行次数与总运行次数的比值, 测试结果如表 3 所示.

由表 3 可见, IGPSO 算法搜索到给定精度所需要的评价次数的最大值、最小值、中间值、平均值在大多数函数测试中都是最小的, 在同样的精度条件下, 评价次数越小, 表明算法的搜索速度越快. 从平均评价次数来看, 本文算法的搜索速度比 PSO、APSO、CLPSO、DE、JADE、GHS、GABC、CS 算法要快. 尽管 GHS 算法在处理函数 f_6 和 f_8 时体现出较高的搜索效率, 但在其他 10 个函数上都劣于本文算法. JADE 算法在 f_5 和 f_{12} 表现最好, 但优化其他函数时本文算法的搜索速度要快于 JADE. 值得一提的是, 方

差为0表示算法多次测试都运行到最大评价次数才截止. 如果算法的平均评价次数达到了给定的最大迭代次数, 往往表明算法搜索不到满足精度要求的解. 从另一个方面也表明本文算法能用最短的时间搜索到所要求的优化精度, 显示出良好的全局搜索能力. 此外, 成功率为0表明, 对于50次独立运行, 在规定的迭代次数下, 算法不能达到给定的优化精度; 成功率为1表明, 对于所有50次独立运行, 在规定的迭代次数下, 算法都能达到给定的优化精度. 表3结果表明, 本文算法相对于其他算法具有更好的寻优性(收敛性)和稳定性.

3.3 算法的非参数统计显著性比较

本节进行t测试实验, 目的是检验本文算法的优化性能是否显著.

假设t测试实验中显著性水平为0.005, 测试结果小于0.005表示显著, 记为“+”; 大于0.005表示不显著, 记为“−”. 独立测试算法50次, 利用50次优化的结果进行两个数据的t测试, 分别比较IGPSO与PSO、APSO、CLPSO、DE、JADE、IGHS、GABC、CS的显著性, 结果如表4所示.

表3 算法评价次数比较

函数	精度	指标	算法								
			PSO	APSO	CLPSO	DE	JADE	GHS	GABC	CS	IGPSO
f_1	1.00e-15	Mini_FEs	120 000	99 500	24 400	92 300	46 300	120 000	86 902	114 798	**7 826**
		Medi_FEs	120 000	100 720	24 880	93 950	47 300	120 000	90 558	115 042	**9 620**
		Maxi_FEs	120 000	102 180	25 080	95 800	48 600	120 000	120 000	120 000	**10 288**
		Mean_FEs	120 000	100 780	24 810	94 000	47 375	120 000	97 004.5	115 035	**9 338.5**
		SD_FEs	0	1 123.328	**300.444 1**	1 555.635	1 027.538	0	15 427.75	2 189.557 4	1 069.412
		SR	0	**1**	**1**	**1**	**1**	0	0.75	0.5	**1**
f_2	1.00e-15	Mini_FEs	120 000	120 000	120 000	120 000	120 000	120 000	120 000	120 000	**17 372**
		Medi_FEs	120 000	120 000	120 000	120 000	120 000	120 000	120 000	120 000	**19 063**
		Maxi_FEs	120 000	120 000	120 000	120 000	120 000	120 000	120 000	120 000	**21 704**
		Mean_FEs	120 000	120 000	120 000	120 000	120 000	120 000	120 000	120 000	**19 300.5**
		SD_FEs	0	0	0	0	0	0	0	0	**1 930.654**
		SR	0	0	0	0	0	0	0	0	**1**
f_3	1.00e-15	Mini_FEs	120 000	120 000	120 000	120 000	103 300	120 000	120 000	114 890	**8 294**
		Medi_FEs	120 000	120 000	120 000	120 000	107 850	120 000	120 000	115 011	**8 828**
		Maxi_FEs	120 000	120 000	120 000	120 000	115 400	120 000	120 000	120 000	**9 084**
		Mean_FEs	120 000	120 000	120 000	120 000	108 600	120 000	120 000	115 041.5	**8 758.5**
		SD_FEs	0	0	0	0	5 430.163	0	0	1 155.369 5	**335.481 2**
		SR	0	0	0	0	**1**	0	0	0.85	**1**
f_4	1.00e-04	Mini_FEs	120 000	120 000	120 000	120 000	44 400	120 000	120 000	120 000	**5 874**
		Medi_FEs	120 000	120 000	120 000	120 000	46 850	120 000	120 000	120 000	**13 033**
		Maxi_FEs	120 000	120 000	120 000	120 000	50 400	120 000	120 000	120 000	**42 868**
		Mean_FEs	120 000	120 000	120 000	120 000	47 125	120 000	120 000	120 000	**18 702**
		SD_FEs	0	0	0	0	**2 497.165**	0	0	0	16 768.16
		SR	0	0	0	0	**1**	0	0	0	**1**
f_5	1.00e-02	Mini_FEs	120 000	120 000	120 000	120 000	94 400	120 000	120 000	120 000	**44 604**
		Medi_FEs	120 000	120 000	120 000	120 000	96 200	120 000	120 000	120 000	**85 955**
		Maxi_FEs	120 000	120 000	120 000	120 000	**104 600**	120 000	120 000	120 000	124 496
		Mean_FEs	120 000	120 000	120 000	120 000	97 850	120 000	120 000	120 000	**85 252.5**
		SD_FEs	0	0	0	0	**4 740.253**	0	0	0	35 988.35
		SR	0	0	0	0	**1**	0	0	0	0.75
f_6	1.00e-15	Mini_FEs	120 000	37 180	7 240	21 400	11 200	**841**	25 558	114 797	1 926
		Medi_FEs	120 000	38 660	12 780	24 100	12 350	**1 746.5**	33 312	114 928.5	2 734
		Maxi_FEs	120 000	40 080	23 020	24 300	12 500	5 385	113 694	115 131	**3 576**
		Mean_FEs	120 000	38 645	13 955	23 475	12 100	**2 429.75**	51 469	114 946.3	2 742.5
		SD_FEs	0	1 185.06	7 896.368	1 396.126	616.441 4	2 082.403	41 657.96	1 138.999 7	**871.953 2**
		SR	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

续表 3

函数	精度	指标	算法								
			PSO	APSO	CLPSO	DE	JADE	GHS	GABC	CS	IGPSO
f_7	1.00e-15	Mini_FEs	120 000	120 000	120 000	120 000	120 000	20 202	120 000	120 000	**10 518**
		Medi_FEs	120 000	120 000	120 000	120 000	120 000	**29 210.5**	120 000	120 000	29 338
		Maxi_FEs	120 000	120 000	120 000	120 000	120 000	93 047	120 000	120 000	**58 226**
		Mean_FEs	120 000	120 000	120 000	120 000	120 000	42 917.5	120 000	120 000	**31 855**
		SD_FEs	0	0	0	0	0	33 981.29	0	0	**21 007.96**
		SR	0	0	0	0	0	**1**	0	0	**1**
f_8	−12 569	Mini_FEs	120 000	53 020	46 217	120 000	55 700	**620**	120 000	120 000	12 640
		Medi_FEs	120 000	57 910	49 720.5	120 000	56 300	**921.5**	120 000	120 000	25 925
		Maxi_FEs	120 000	120 000	67 251	120 000	59 600	**2 522**	120 000	120 000	48 956
		Mean_FEs	120 000	72 210	53 227.25	120 000	56 975	**1 246.25**	120 000	120 000	28 361.5
		SD_FEs	0	31 954.79	9 515.507	0	1 819.112	**863.240 6**	0	0	17 684.6
		SR	0	0.75	**1**	0	**1**	**1**	0	0	**1**
f_9	1.00e-15	Mini_FEs	120 000	120 000	120 000	120 000	120 000	120 000	120 000	120 000	**19 410**
		Medi_FEs	120 000	120 000	120 000	120 000	120 000	120 000	120 000	120 000	**19 980**
		Maxi_FEs	120 000	120 000	120 000	120 000	120 000	120 000	120 000	120 000	**21 298**
		Mean_FEs	120 000	120 000	120 000	120 000	120 000	120 000	120 000	120 000	**20 167**
		SD_FEs	0	0	0	0	0	0	0	0	**802.370 7**
		SR	0	0	0	0	0	0	0	0	**1**
f_{10}	1.00e-10	Mini_FEs	120 000	68 120	67 820	54 700	27 100	120 000	59 982	120 000	**6 294**
		Medi_FEs	120 000	69 250	103 950	55 550	27 800	120 000	71 260.5	120 000	**6 463**
		Maxi_FEs	120 000	71460	120 000	57 500	28 900	120 000	120 000	120 000	**7 168**
		Mean_FEs	120 000	69 520	98 930	55 825	27 900	120 000	80 625.75	120 000	**6 597**
		SD_FEs	0	1 459.406	25 673.48	1 192.686	748.331 5	0	27 654.18	0	**407.498 1**
		SR	0	**1**	0.5	**1**	**1**	0	0.75	0	**1**
f_{11}	1.00e-06	Mini_FEs	120 000	67 340	11 140	48 700	24 900	120 000	52 969	24 853	**4 530**
		Medi_FEs	120 000	73 300	120 000	48 900	26 350	120 000	90 680	105 083	**11 586**
		Maxi_FEs	120 000	77 100	120 000	49 000	31 600	120 000	120 000	120 000	**20 018**
		Mean_FEs	120 000	72 760	92 785	48 875	27 300	120 000	88 582.25	105 068.3	**11 930**
		SD_FEs	0	4 032.997	54 430	**125.830 6**	2 947.315	0	36 439.47	5 741.962	8 198.946
		SR	0	**1**	0.25	**1**	**1**	0	0.5	0.65	**1**
f_{12}	1.00e-06	Mini_FEs	120 000	57 740	15 660	36 700	20 600	**5 945**	31 866	124 821	18 614
		Medi_FEs	120 000	58 850	70 030	40 450	20 950	63 446.5	57 232.5	125 099.5	**20 320**
		Maxi_FEs	120 000	60 320	120 000	41 500	**22 000**	120 000	82 450	125 243	39 918
		Mean_FEs	120 000	589 40	68 930	39 775	**21 125**	63 209.5	57 195.25	125 065.8	24 793
		SD_FEs	0	1 162.87	58 997.91	2 112.463	**670.199**	65 577.16	22 010.74	193.410 4	10 166.55
		SR	0	**1**	0.5	**1**	**1**	0.5	**1**	0	**1**

表 4　优化结果的比较 ($N = 30$) (二)

算法	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
IGPSO/PSO	+	+	+	+	+	+	+	+	+	+	+	+
IGPSO/APSO	+	+	+	+	+	+	+	+	+	+	+	+
IGPSO/CLPSO	+	+	+	+	+	+	+	+	+	+	+	+
IGPSO/DE	+	+	+	+	+	+	+	+	+	+	+	+
IGPSO/JADE	+	+	+	+	NA	+	+	+	+	+	+	+
IGPSO/ABC	+	+	+	+	+	+	+	+	+	+	+	+
IGPSO/IGHS	+	+	+	+	+	+	+	+	+	+	+	+
IGPSO/CS	+	+	+	+	+	+	+	+	+	+	+	NA

由表4可见, 在12个函数中, IGPSO算法优化出的结果相对于PSO、APSO、DE、IGHS、GABC、CS都是显著的, 表明在优化性能上, IGPSO算法优于PSO、APSO、DE、IGHS、GABC、CS算法, IGPSO算法具有较大的进步, 优化结果比较稳定, 是一种竞争力较强的启发式算法. 与CLPSO算法相比, 除了函数f_{12}本文算法的优化结果没有优势外, 余下的11个函数本文算法都具有明显的优势, 表明本文算法优于CLPSO算法. 通过JADE算法与本文算法的比较可以看出, 本文算法在大多数函数测试中占有明显的竞争优势, 因此, 本文算法在整体上具有更好的优化性能.

4 结 论

本文提出一种改进的全局粒子群优化(IGPSO)算法, 主要体现在3个方面的改进: 1) 考虑粒子历史最优和全局最优邻域的潜在解, 加强了它们邻域的搜索; 2) 增强了粒子历史最优种群的邻域扰动, 独立进行历史最优粒子的自我学习过程, 以增强局部搜索能力; 3) 为防止全局最优粒子得不到有效的更新和进化, 引入两种学习策略, 增强全局最优粒子的自我提升能力, 以便更好地引导整个群体向更优的优化方向移动. 实验测试表明, IGPSO算法具有显著的优化优势, 是一种具有竞争力的优化算法. 尽管IGPSO算法在许多函数测试中脱颖而出, 但其优化性能需要许多参数的合理设置. 相对于其他算法而言, IGPSO算法需要增加进一步参数自适应设置策略, 这是IGPSO算法的后续工作. 此外, 可以进一步将IGPSO算法应用于网络优化、图像处理、智能机械优化设计等方面.

参考文献(References)

[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neutral Networks. Perth, 1995: 1942-1948.

[2] Zhang R, Song S, Wu C. A two-stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem[J]. Knowledge-Based Systems, 2012, 27: 393-406.

[3] Khare A, Rangnekar S. A review of particle swarm optimization and its applications in Solar Photovoltaic system[J]. Applied Soft Computing, 2013, 13(5): 2997-3006.

[4] Mahor A, Prasad V, Rangnekar S. Economic dispatch using particle swarm optimization: A review[J]. Renewable and Sustainable Energy Reviews, 2009, 13(8): 2134-2141.

[5] Cai Q, Gong M, Ma L, et al. Greedy discrete particle swarm optimization for large-scale social network clustering[J]. Information Sciences, 2014, 316: 503-516.

[6] Han W, Yang P, Ren H, et al. Comparison study of several kinds of inertia weights for PSO[C]. Proc of 2010 IEEE Int Conf on Informatics and Computing. Shanghai, 2010, 1: 280-284.

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

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

[9] Thangaraj R, Pant M, Abraham A, et al. Particle swarm optimization: Hybridization perspectives and experimental illustrations[J]. Applied Mathematics and Computation, 2011, 217(12): 5208-5226.

[10] Li Y, Zhan Z H, Lin S, et al. Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems[J]. Information Sciences, 2015, 293: 370-382.

[11] Tizhoosh H R. Opposition-based learning: A new scheme for machine intelligence[C]. Proc of Int Conf on Computational Intelligence. Vienna, 2005: 695-701.

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

[13] Storn R, Price K. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces[J]. J of Global Optimization, 1995, 11(4): 341-359.

[14] Zhang J Q, Sanderson A C. JADE: Adaptive differential evolution with optional external archive[J]. IEEE Trans on Evolutionary Computation, 2009, 13(5): 945-958.

[15] Mohammed E A. An improved global-best harmony search algorithm[J]. Applied Mathematics and Computation, 2013, 222(1): 94-106.

[16] Zhu G P, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics and Computation, 2010, 217(7): 3166-3173.

[17] Yang X S, Deb S. Cuckoo search via Lévy flights[C]. Proc of IEEE World Congress on Nature & Biologically Inspired Computing. New York, 2009: 210-214.

(责任编辑：郑晓蕾)