控制与决策
Control and Decision

基于局部搜索的反向学习竞争粒子群优化算法

钱晓宇， 方伟

引用本文:
钱晓宇, 方伟. 基于局部搜索的反向学习竞争粒子群优化算法[J]. 控制与决策, 2021, 36(4): 779−789.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2019.1150

基于局部搜索的反向学习竞争粒子群优化算法

钱晓宇, 方 伟†

(江南大学 物联网工程学院，江苏 无锡 214122)

摘 要: 为提升粒子群优化算法在复杂优化问题,特别是高维优化问题上的优化性能,提出一种基于Solis & Wets局部搜索的反向学习竞争粒子群优化算法(solis and wets-opposition based learning competitive particle swarm optimizer with local search, SW-OBLCSO). SW-OBLCSO算法采用竞争学习和反向学习两种学习机制,并设计了基于个体的局部搜索算子. 利用10个常用基准测试函数和12个带有偏移旋转的复杂测试函数,在不同维度情况下将SW-OBLCSO算法与多种优化算法进行对比. 实验结果表明,所提出算法在收敛速度和全局搜索能力上表现出突出的性能. 对模糊认知图(fuzzy cognitive maps)学习问题的测试表明,SW-OBLCSO算法在处理实际问题时同样具有出色的性能.

关键词: 粒子群优化算法；反向学习；大规模优化问题；竞争学习；局部搜索；高维优化

中图分类号: TP18 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2019.1150 **开放科学(资源服务)标识码(OSID):**

引用格式: 钱晓宇,方伟. 基于局部搜索的反向学习竞争粒子群优化算法[J]. 控制与决策, 2021, 36(4): 779-789.

Opposition-based learning competitive particle swarm optimizer with local search

QIAN Xiao-yu, FANG Wei†

(School of IoT Engineering，Jiangnan University，Wuxi 214122，China)

Abstract: In order to improve the optimization ability of particle swarm optimization (PSO) algorithms in complex optimization problems, especially in high dimensional problems, an opposition-based learning competitive particle swarm optimization algorithm based on Solis & Wets (SW-OBLCSO) local search is proposed. The SW-OBLCSO algorithm adopts two learning mechanisms, namely competitive learning and opposition-based learning. An individual-based local search operator is also introduced. The SW-OBLCSO algorithm is compared with various optimization algorithms on the 10 benchmark functions and 12 complex test functions in different dimensions. The experimental results show that the proposed algorithm exhibits outstanding performance in convergence speed and global search ability. Performance comparison on the fuzzy cognitive map (FCM) learning problems shows that the SW-OBLCSO algorithm also has excellent performance when dealing with practical problems.

Keywords: particle swarm optimization；opposition-based learning；large-scale optimization problem；competitive learning；local search；high-dimensional optimization

0 引 言

粒子群优化(particle swarm optimization, PSO)算法[1-2]是一种群体智能优化算法,该算法源自对鸟类捕食问题的研究. 在标准PSO算法中,种群由若干个个体组成,每个个体用速度和位置表示,个体的速度决定了它的迭代方向和距离,位置代表其中一个解向量,并通过个体的速度来更新自己的位置,进而获取最优解.

PSO算法自提出以来便受到各领域学者的关注,并取得了丰硕的研究成果[3-5]. 然而,PSO算法易出现早熟收敛而导致优化性能降低[6],为此,研究人员在算法的性能改进方面展开了较多的研究. 其中一类改进方法是通过设计粒子不同的邻域结构来改善群体的信息交流机制,以提高群体的搜索能力. 如文献

收稿日期: 2019-08-11；**修回日期:** 2019-10-29.

基金项目: 国家重点研发计划项目(2017YFC1601800, 2017YFC1601000)；江苏省重点研发计划项目(BE2017630)；国家自然科学基金项目(61673194, 62073155)；江苏省自然科学基金项目(BK20131106)；中国博士后科学基金项目(2014M560390)；江苏省高校“青蓝工程”项目.

责任编委: 林崇.

†通讯作者. E-mail: fangwei@jiangnan.edu.cn.

[7]中研究的环形、冯诺依曼拓扑方式、全连接方式等结构;文献[8]设计了一种基于混合拓扑结构的PSO算法;文献[9]提出了具有自适应逃逸的环状全互联结构PSO算法;文献[10]将具有信息定向流动的闭环拓扑结构与星型拓扑结构相结合,促使粒子在前期寻优过程中具有较高的多样性. 在PSO算法中改进或引入更为智能的学习机制来影响群体的学习方法也是一直以来的研究热点. 文献[11]提出了自适应学习的PSO算法;文献[12]结合金属导体中自由电子的定向漂移运动和随机无规则热运动方式,提出了随机漂移粒子群(RDPSO)算法. 反向学习(opposition-based learning, OBL)作为一种有效拓宽搜索空间的机制[13],已被有效应用到PSO算法中. 文献[14]提出了一种反向学习粒子群优化算法,其中OBL不仅用于粒子的位置更新,也用于选择种群初始位置;文献[15]提出了一种邻域重心反向学习粒子群优化算法,以邻域重心为参考点计算反向点,从而在充分利用群体搜索经验的同时保持种群多样性. 上述两种改进没有关注算法的高维扩展性能. 文献[16-17]将反向学习引入竞争粒子群优化算法,在竞争过程中部分粒子通过反向学习更新其位置,该算法在高维问题上有较好的表现.

为了进一步改善PSO算法的求解效果,使算法的全局搜索能力与局部搜索能力达到更好的平衡,本文在改进竞争粒子群算法的基础上融合了反向学习机制和局部搜索策略,提出一种基于局部搜索的反向学习竞争粒子群优化算法(SW-OBLCSO). 其中改进的竞争机制结合反向学习机制提高了算法的收敛速度,在搜索后期结合基于个体的SW (Solis & Wets)[18]局部启发式搜索算子,帮助算法跳出局部最优解.

1 粒子群优化(PSO)算法

在PSO算法中,种群中每个粒子代表问题的潜在解,位置向量表示为$X_i = [X_{i1}, X_{i2}, \ldots, X_{iD}]$,速度向量表示为$V_i = [V_{i1}, V_{i2}, \ldots, V_{iD}]$,其中$D$是问题的维度. 在进化过程中记录和更新第$i$个粒子的个体历史最佳位置,并表示为$P_i = [P_{i1}, P_{i2}, \ldots, P_{iD}]$. 具有最佳适应度的粒子被认为是全局最佳粒子,表示为$G = [G_1, G_2, \ldots, G_D]$. 在初始化阶段,首先在有限搜索空间中以随机位置和速度生成粒子;然后,粒子根据个体经验和群体经验不断更新速度和位置,更新公式如下所示:

$$V_{id}(t+1) = \omega \cdot V_{id}(t) + c_1 \cdot r_{1d}(t) \cdot (P_{id}(t) - X_{id}(t)) + c_2 \cdot r_{2d}(t) \cdot (G_d(t) - X_{id}(t)), \tag{1}$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1). \tag{2}$$

其中:t是迭代次数;ω是用于平衡全局与局部搜索能力的惯性权重;c_1和c_2是两个常数,分别称为自学习因子和社会学习因子;r_{1d}和r_{2d}是两个均匀分布在[0, 1]范围内的随机数. 从速度和位置更新公式可以看出,粒子的位置由其自身的先前速度、从粒子到其个体历史最佳位置的距离以及粒子与每次迭代中的全局最佳位置之间的距离确定.

PSO算法的流程如下.

step 1: 设定群体大小,初始化粒子位置和速度.

step 2: 计算目标函数值,更新全局最优位置和粒子的局部最优位置.

step 3: 根据式(1)和(2)更新粒子的速度和位置.

step 4: 判断是否满足结束条件, 若满足, 则输出结果并结束;否则转入step 2.

2 基于局部搜索的反向学习竞争粒子群优化(SW-OBLCSO)算法

2.1 反向学习机制

反向学习机制(opposition-based learning, OBL)是一种可以有效地拓宽搜索空间、覆盖可行解区域的机制,并且已被有效应用在差分进化算法[19]中. 反向学习机制可以在单次迭代中计算候选解决方案及其相应的反向解决方案. 反向算子是当前候选解决方案的扩展,有较大概率接近最优解. 考虑区间$[a, b]$中的实数x,其反向数$\tilde{x}$计算如下:

$$\tilde{x} = a + b - x. \tag{3}$$

将一维空间的反向数概念扩展到多维空间,设$P(x_1, x_2, \ldots, x_D)$为D维空间中的数,$x_1, x_2, \ldots, x_D$均为实数,$x_i \in [a_i, b_i]$,则其反向数$\tilde{P}$由其坐标$\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_D$定义,其中

$$\tilde{x}_i = a_i + b_i - x_i. \tag{4}$$

引入反向学习的优点是,它可以同时搜索当前点和反向点,如果反向点的适应值优于当前点的适应值,则粒子可以直接跳转到反向点或其邻域继续搜索. 这可以快速扩大解决方案空间,增加找到全局最优解的可能性.

2.2 Solis & Wets局部搜索(SW)算法

局部搜索算法是解决优化问题常用的一种元启发式算法. 局部搜索从一个初始解出发,然后探索其邻域. 如果发现较优的解,则移动到该解并继续执行搜索,否则保留当前解.

SW 算法是一个强有力的局部搜索算法,属于带有自适应步长的随机爬山算法,它的性能在很多大规模优化算法中得到了验证. 首先通过高斯采样得到偏差向量d, d的维度与问题维数相同. 在初始搜索点x的基础上产生两个新解$x+d$和$x-d$. 计算两个新解的适应度值,并选择一个较优解作为当前新解x_{new}. 比较x和x_{new}的适应度值,如果当前新解较优,则x被x_{new}替换,成功搜索计数器递增一次且失败搜索计数器清零. 如果原来的解较优,则保持原来的解不变,失败搜索计数器递增一次且成功搜索计数器清零. 根据计数器的数值动态调整偏差向量的大小. SW 算法的流程如下.

step 1: 初始化迭代次数、各计数器阈值,获取初始解x和val.

step 2: 生成高斯分布随机向量d,产生新解$x_1=x+d, x_2=x-d$并评估,取较优解记为x_3,其适应度值为$\text{val}(x_3)$.

step 3: 比较较优解与当前解的适应度大小,更新当前解和相关计数器.

step 4: 判断计数器值是否超过阈值,调整搜索步长并更新计数器.

step 5: 判断是否满足结束条件, 若满足, 则输出结果并结束;否则转入step 2.

2.3　竞争学习机制和改进策略分析

在PSO算法中,随着进化过程的进行,粒子的个体历史最佳位置可能与全局最佳粒子具有相同的值,导致群体的多样性降低. 为了削弱或消除全局最佳粒子和个体历史最佳粒子的影响,文献[17]提出了竞争群体优化器(CSO),其中粒子通过连续随机成对竞争来更新. 在成对竞争之后,优胜粒子直接传递给下一次迭代,失败粒子向优胜粒子学习. 文献[16]在此基础上引入反向学习机制,提出了OBLCPSO 算法,将粒子的成对竞争改进为3组粒子竞争,具有中间适应度值的粒子通过反向学习更新位置,最终使算法获得更强的优化性能并具备一定的高维扩展能力.

竞争粒子群算法中粒子不具有记忆功能,算法在寻优过程中不容易陷入局部最优,而反向学习也是一种有效扩宽搜索空间的机制. 为了充分利用两种机制的优势,进一步提升算法的寻优性能,将竞争机制作用在4组粒子之间,使其效率高于CSO和OBLCPSO. 对于竞争过程中粒子的处理模式选择上,有以下几种可行策略:

策略1.1: 优胜者位置不变,第2名进行反向学习,第3名向优胜者学习,失败者直接获取优胜者的位置并增加一个较小的偏置,加快收敛速度的同时保持种群多样性.

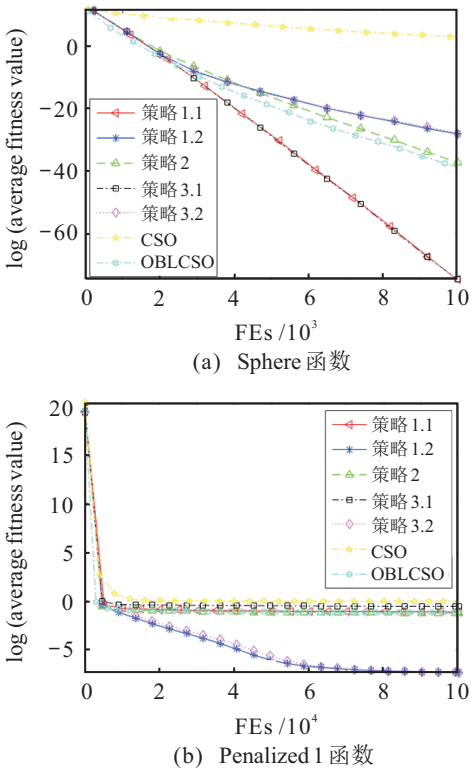
策略1.2: 与策略1.1不同的是,失败者获取优胜者位置后增加一个较大的偏置.

策略2: 与策略1.1不同的是,失败者获取优胜者位置后增加一个单调递减的偏置.

策略3.1: 优胜者位置不变,第2名进行反向学习,失败者向优胜者学习,第3名直接获取优胜者的位置并增加一个较小的偏置.

策略3.2: 与策略3.1不同的是,第3名获取优胜者位置后增加一个较大的偏置.

为了探讨各策略的优化效果,选用Sphere函数和Penalized 1函数作为测试函数. Sphere函数是连续凸单峰函数,函数曲面较为简单,在低维情况时较易获得全局最优解. Penalized 1函数是多峰函数,函数中带有的惩罚项以及三角函数使得函数曲面上局部最优点较多.将各策略以及CSO算法、OBLCPSO算法应用在这两个测试函数上. 测试函数维度设置为$D=100$, 由于函数复杂性不同,分别设置最大评估次数max FEs为10 000和100 000. 改进策略中较小的偏置设置为$\text{bias}\,1=0.005$,较大的偏置设置为$\text{bias}\,2=0.02$, 单调递减的偏置设置为$\text{bias}\,3=0.02-0.005$. 对比算法的参数与相关文献一致,各算法均单独运行25次. 图1是各算法优化过程收敛曲线.



(a)　Sphere函数

(b)　Penalized 1函数

图 1　各算法优化2个测试函数的收敛曲线

分析收敛曲线可知,对于Sphere函数,策略1.1和策略3.1的优化性能最好. CSO算法的收敛速度较慢,其余采用反向学习的算法都有较快的收敛速度,表明反向学习机制是有效的. 其中OBLCPSO算法在迭代初期收敛速度最快,随着种群多样性降低,收敛速度明显有下降的趋势. 策略1.1和策略3.1对失败粒子添加了一个较小的偏置,在种群更新过程中保持了一定的多样性,使得收敛速度并没有下降太多. 而策略2和策略3由于添加了较大的偏置,使失败粒子远离优胜粒子较多,反而使得粒子的更新没有往好的方向发展. 对于Penalized 1函数而言,策略1.2取得了最好的效果,策略3.2与之相差不多,其余的算法都很快陷入了局部最优. 这是由于Penalized 1函数有大量局部最优点,策略1.2和策略3.2对失败者添加了较大的偏置,使得粒子发生较大的位置变化而跳出了当前局部最优点.

综上所述,策略1总体表现出了最好的性能. 对于具有较少局部最优点的问题,策略1.1的方法对解决问题更有帮助. 对于具有多个局部最优点的复杂问题,策略1.3中添加了较大的偏置能帮助算法跳出局部最优点. 策略1总体具有较快的收敛速度,消耗很少的评估次数代价就能较快地接近全局最优点,这十分契合局部搜索策略高度依赖初始解的质量的特点,且能为局部搜索预留充足的评估次数.

2.4 基于局部搜索的反向学习竞争粒子群优化(SW-OBLCSO)算法

结合上一节的分析,本文提出一种基于局部搜索的反向学习竞争粒子群算法(SW-OBLCSO). 算法在改进竞争机制的基础上结合了反向学习和局部搜索,相比CSO算法而言同样简单却更加有效. 算法迭代过程中,每次把种群随机分成4组,每组依次选取一个粒子进行适应度评估,然后根据适应度值进行排序. 排名首位的粒子直接进入下一代,称为优胜者;排名第2的粒子进行反向学习;排名第3的粒子向优胜者粒子竞争学习;排名最后的粒子获得优胜粒子的位置并添加一个偏置量,以提高收敛速度并保持种群多样性. 4组中的所有粒子都参加竞争后,找出群体中此时的最小适应度值,如果连续多代最小适应度值都没有发生变化,则以后的迭代中对具有最小适应度的粒子进行SW搜索,其中SW偏差向量d随迭代次数增加而变小. SW-OBLCSO 算法中第2(second)、第3(third)、第4(loser)适应度值的粒子通过以下公式更新位置和速度:

$$V_{td}^k(t+1)=R_{1d}^k(t)\cdot V_{td}^k(t)+R_{2d}^k(t)\cdot(X_{wd}^k(t)-X_{td}^k(t))+\psi\cdot R_{3d}^k(t)\cdot(\tilde{X}_d^k(t)-X_{td}^k(t)),\tag{5}$$

$$X_{td}^k(t+1)=X_{td}^k(t)+V_{td}^k(t+1),\tag{6}$$

$$X_{sd}^k(t+1)=ub_d+lb_d-X_{sd}^k(t)+R_{4d}^k(t)\cdot X_{sd}^k(t),\tag{7}$$

$$X_{ld}^k(t+1)=X_{wd}^k(t)\pm\tanh'i\cdot p_d(t).\tag{8}$$

其中: $X_{sd}^k(t)$、 $X_{td}^k(t)$和$X_{ld}^k(t)$分别代表第t次迭代第k轮竞争中的第2名、第3名和失败者位置的第d维; $V_{td}^k(t)$是第t次迭代第k轮竞争中第3名的第d维的速度; $R_{1d}^k(t)$、$R_{2d}^k(t)$、$R_{3d}^k(t)$和$R_{4d}^k(t)$是[0, 1]内的4个随机数; ψ是人工设置的参数; $\tilde{X}_d^k(t)$是所有粒子的平均位置; ub_d和lb_d是搜索空间的第d维的上界和下界; $p_d(t)$是与问题维数相同的高斯采样偏差向量在第t次迭代中的第d维; i为常数.

在优胜者的基础上产生两个新解$x+\tanh'i\cdot p$和$x-\tanh'i\cdot p$. 计算两个新解的适应度值,并选择一个较优解赋给失败者. SW-OBLCSO算法流程(见图2)如下.

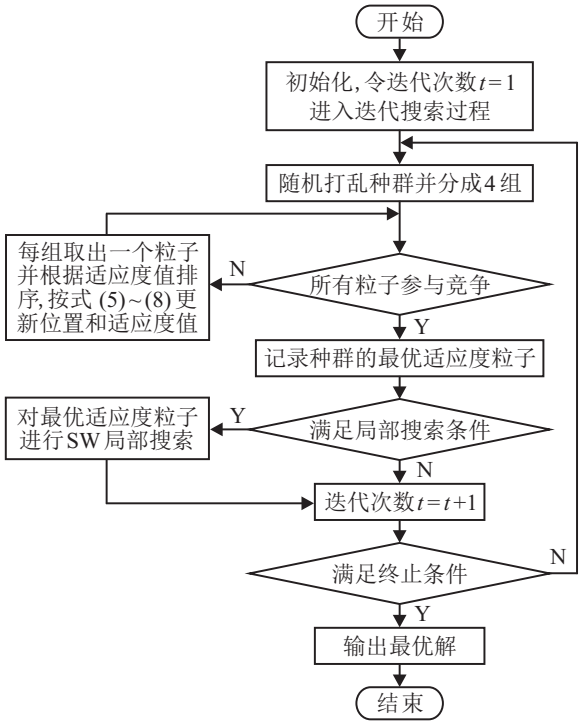


图2 SW-OBLCSO算法流程

step 1: 初始化种群P、粒子的位置和速度;

step 2: while FEs<max_FEs;

step 3: 随机打乱种群并分成4组;

step 4: for $k=1:N/4$ do;

step 5: 每组中取出一个粒子,记为(r_1,r_2,r_3,r_4);

step 6: $(w,s,t,l)=\mathrm{compete}(r_1,r_2,r_3,r_4)$; //根据适应度值排序;

step 7: 根据式(5)～(8)更新粒子的位置和速度;

step 8: 更新第2、第3、第4名的适应度值;

Step 9: end for;

step 10: 找出种群的最优适应度粒子,记录[min_index, min_fit];

step 11: 若 min_fit 的值连续 N 次不变,则对 $P(\text{min\_index})$ 进行SW局部搜索;

step 12: end while.

3 实验结果与分析

3.1 10个基准测试实验

3.1.1 测试函数

为了检验本文所提出的SW-OBLCSO算法的寻优能力和收敛效率,首先选择表1中的10个常用的基准测试函数作为算法的测试函数,这些测试函数被广泛使用在计算智能领域的算法性能检测中. 其中: F_1 是Sphere函数,特征为单峰、完全可分;函数 F_2 是典型病态香蕉型的单峰函数,算法能找到全局最优的机会很小; F_7、F_8 统称为Schwefel问题,特征为单峰不可分; F_9 为带有大量噪声的单峰函数; F_3 为step函数,该函数在定义域内趋近于无穷时,会在给定的间隔上出现不同的阶跃现象,并且在每个阶跃之间会产生大量局部极值;函数 F_4 为Rastrigin函数,特征是多峰不对称; F_5、F_{10} 是较为复杂的不可分多峰函数,且局部最优与全局最优相距很远,导致算法难以获得全局最优;函数 F_6 是多峰、不可分的可旋转函数,函数曲面存在大量的波峰波谷,在通往全局最优的路径上优化算法很容易陷入局部最优点.

表 1　文中所用测试函数

函数表达式	搜索空间
$F_1=\sum_{i=1}^{D}x_i^2$	$[-100,100]$
$F_2=\sum_{i=1}^{D-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$	$[-30,30]$
$F_3=\sum_{i=1}^{D}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]$
$F_4=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]$
$F_5=-20\exp\Big(-0.2\sqrt{\sum_{i=1}^{D}x_i^2/D}\Big)-\exp\Big(\sum_{i=1}^{D}\cos(2\pi x_i)/D\Big)+20+\mathrm{e}$	$[-32,32]$
$F_6=\sum_{i=1}^{D}x_i^2/4\,000-\prod_{i=1}^{D}\cos(x_i/\sqrt{i})+1$	$[-600,600]$
$F_7=\sum_{i=1}^{D}\Big(\sum_{j=1}^{i}x_j\Big)^2$	$[-100,100]$
$F_8=\max_{i=1,\dots,D}\lvert x_i\rvert$	$[-100,100]$
$F_9=\sum_{i=1}^{D}i\times x_i^4+\mathrm{random}[0,1)$	$[-1.28,1.28]$
$F_{10}=\frac{\pi}{D}\Big\{10(\sin(\pi y_1))^2+\sum_{i=1}^{D-1}(y_i-1)^2[1+10(\sin(\pi y_{i+1}))^2]+(y_D-1)^2\Big\}+\sum_{i=1}^{D}u(x_i,10,100,4)$ where $y_i=1+(x_i+1)/4$, $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\leqslant x_i\leqslant a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	$[-50,50]$

3.1.2 参数设置

在对比算法方面,选择PSO算法、QPSO算法[20]以及RDPSO算法、SLPSO算法[21]、OBLCPSO算法、CSO算法这4个最新的改进算法作为对比算法. 为了检验SW-OBLCSO算法在低维问题上的优化性能以及对高维问题的扩展性能,分别将10个基本测试函数的维度设置为 $D=100,500,1\,000$. 由于问题维度的改变,各算法的种群规模设置为 $N=M+D/10$,按照文献[22]中的方法,取 $M=100$. 根据相应参考文献,采用的算法参数设置如表2所示. 为了减少实验的随机性,各个算法在每个测试函数上均随机独立运行25次. 每次实验都以最大评估次数(max FEs)作为终止条件, max FEs = 200 000. 本文实验所用的计算机配置如下: Intel Xeon e3-1230-v5CPU, 3.4 GHz, 8 G内存, Windows 10专业版, 64位操作系统.

表 2　各对比算法参数设置

算法	参数设置
SW-OBLCSO	$\psi=0.7, i=20, N=50, \max S=5, \max F=3, \mathrm{ad}S=2, \mathrm{ad}F=0.5, \mathrm{dev}=\tanh'(t/120)$
CSO	$\psi=0.7$
PSO	$\omega=0.9-0.4, c_1=c_2=2.0$
OBL-CPSO	$\psi=0.7-0.2$
SLPSO	$\alpha=0.5, \beta=0.01$
RDPSO	$\omega=0.9-0.3, c_1=c_2=1.5$
QPSO	$\alpha=1.0-0.5$

3.1.3 实验结果及分析

本文提出的SW-OBLCSO算法以及对比算法在10个标准测试函数上的实验结果由表3给出. 每个测试函数的实验结果由3行数据构成,分别代表100, 500, 1 000维下的优化结果. 每个算法对应有两列,第1列表示算法在此测试函数上此维度下独立

运行25次所得的目标函数的平均值,第2列表示25个目标函数值的标准差. 表4给出了表3数据的wilcoxon秩和检验结果,符号“+”表示两种算法对比,第1种比第2种有明显的优势;符号“=”表示两种算法优化效果差不多,无明显差别;符号“−”表示第1种算法优化结果比第2种算法有明显的劣势;gm表示符号“+”的总数与符号“−”的总数的差值,表明了第1种算法比第2种算法优化性能强的一个量值. 图3给出了100维情况下各算法优化各个测试函数的收敛曲线,横坐标为评估次数,对于适应度值变化较大的函数,纵坐标为平均适应度值(取自然对数),其余函数纵坐标为平均适应度值.

表3 各算法在10个标准测试函数上的实验结果

函数	维度	SW-OBLCSO		CSO		PSO		OBL-CPSO		SLPSO		RDPSO		QPSO	
F_1	100	**0.00e+00**	0.00e+00	9.92e-01	9.66e-01	4.44e+02	4.42e+02	6.34e-67	3.12e-66	1.57e-31	1.35e-31	2.31e-01	1.67e-01	1.65e-03	1.48e-03
	500	**1.69e-262**	0.00e+00	8.59e+03	1.04e+03	1.85e+05	2.13e+04	2.48e-55	7.31e-55	3.32e+03	4.07e+02	4.60e+04	5.38e+03	2.52e+04	4.34e+03
	1 000	**3.44e-194**	0.00e+00	3.13e+04	2.04e+03	6.15e+05	2.10e+04	9.74e-44	1.80e-43	2.67e+05	1.09e+04	2.40e+05	1.36e+04	1.77e+05	1.59e+04
F_2	100	**9.22e+01**	1.21e+00	9.00e+02	2.44e+02	1.96e+05	4.61e+05	9.74e+01	4.98e-01	1.36e+02	5.97e+01	5.47e+02	1.28e+02	2.80e+02	8.34e+01
	500	**4.95e+02**	1.30e+00	1.24e+06	2.19e+05	2.85e+08	3.74e+07	4.98e+02	2.21e-01	1.25e+07	2.45e+06	4.47e+07	6.80e+06	2.64e+07	4.45e+06
	1 000	**9.94e+02**	2.80e+00	5.17e+06	4.46e+05	1.13e+09	1.07e+08	9.97e+02	2.04e-01	6.99e+08	2.65e+07	2.94e+08	2.46e+07	2.65e+08	2.64e+07
F_3	100	**0.00e+00**	0.00e+00	2.22e+01	1.10e+01	3.54e+03	2.00e+03	**0.00e+00**	0.00e+00	5.20e-01	9.18e-01	1.55e+01	2.16e+01	1.23e+01	7.82e+00
	500	**0.00e+00**	0.00e+00	9.54e+03	1.12e+03	2.02e+05	1.36e+04	**0.00e+00**	0.00e+00	3.40e+03	4.02e+02	5.04e+04	6.87e+03	2.87e+04	4.57e+03
	1 000	**0.00e+00**	0.00e+00	3.30e+04	2.04e+03	6.36e+05	2.48e+04	**0.00e+00**	0.00e+00	2.70e+05	9.05e+03	2.37e+05	1.45e+04	1.84e+05	1.52e+04
F_4	100	**0.00e+00**	0.00e+00	4.89e+01	7.81e+00	2.74e+02	2.98e+01	**0.00e+00**	0.00e+00	2.81e+02	2.25e+02	1.15e+02	1.60e+01	1.16e+02	1.46e+01
	500	**0.00e+00**	0.00e+00	8.83e+02	7.03e+01	3.59e+03	1.11e+02	**0.00e+00**	0.00e+00	5.64e+03	4.78e+01	2.22e+03	1.04e+02	1.83e+03	9.77e+01
	1 000	**0.00e+00**	0.00e+00	2.79e+03	1.03e+02	8.97e+03	2.05e+02	**0.00e+00**	0.00e+00	1.18e+04	8.75e+01	6.38e+03	2.11e+02	5.83e+03	2.25e+02
F_5	100	**8.88e-16**	2.01e-31	1.05e+00	4.86e-01	8.74e+00	1.16e+00	**8.88e-16**	2.01e-31	1.61e-14	2.62e-15	8.34e-01	5.00e-01	1.20e-02	9.44e-03
	500	**8.88e-16**	2.01e-31	6.12e+00	2.74e-01	1.74e+01	2.05e-01	**8.88e-16**	2.01e-31	5.00e+00	1.33e-01	1.16e+01	4.26e-01	1.04e+01	5.05e-01
	1 000	**8.88e-16**	2.01e-31	7.54e+00	1.91e-01	1.83e+01	1.44e-01	**8.88e-16**	2.01e-31	1.48e+01	1.51e-01	1.45e+01	2.28e-01	1.38e+01	2.39e-01
F_6	100	**0.00e+00**	0.00e+00	4.52e-01	3.29e-01	5.02e+00	3.09e+00	**0.00e+00**	0.00e+00	1.97e-03	5.45e-03	1.77e-01	2.21e-01	2.03e-02	3.47e-02
	500	**0.00e+00**	0.00e+00	7.95e+01	1.21e+01	1.67e+03	1.59e+02	**0.00e+00**	0.00e+00	3.14e+01	4.85e+00	4.26e+02	6.32e+01	2.19e+02	2.41e+01
	1 000	**0.00e+00**	0.00e+00	2.80e+02	1.72e+01	5.48e+03	2.40e+02	**0.00e+00**	0.00e+00	2.41e+03	9.38e+01	2.12e+03	1.64e+02	1.58e+03	1.32e+02
F_7	100	**0.00e+00**	0.00e+00	8.13e+03	1.47e+03	2.22e+04	4.90e+03	9.14e-65	4.52e-64	7.51e+04	1.33e+04	1.04e+04	2.24e+03	1.23e+04	2.35e+03
	500	**1.71e-257**	0.00e+00	1.65e+05	1.60e+04	1.52e+06	2.72e+05	2.57e-51	8.06e-51	3.88e+06	3.58e+05	5.09e+05	9.81e+04	5.64e+05	5.30e+04
	1 000	**5.21e-189**	0.00e+00	4.94e+05	5.33e+04	5.86e+06	1.15e+06	4.45e-42	1.35e-41	1.57e+07	1.08e+06	2.28e+06	4.46e+05	2.37e+06	2.76e+05
F_8	100	**5.28e-187**	0.00e+00	9.93e+00	1.44e+00	5.13e+01	3.98e+00	1.28e-35	2.81e-35	1.44e-01	3.95e-02	2.22e+01	2.67e+00	1.62e+01	2.71e+00
	500	**9.77e-130**	4.80e-129	2.40e+01	1.25e+00	7.47e+01	2.59e+00	9.31e-28	2.10e-27	8.72e+01	1.26e+00	5.91e+01	1.68e+00	6.23e+01	1.84e+00
	1 000	**5.81e-96**	2.05e-95	2.57e+01	8.54e-01	7.80e+01	1.86e+00	4.47e-22	1.19e-21	9.09e+01	7.35e-01	6.38e+01	1.12e+00	6.92e+01	1.57e+00
F_9	100	**4.58e-03**	4.95e-03	1.78e-01	4.87e-02	3.09e+00	1.68e+00	1.18e-02	1.26e-02	1.07e-01	1.69e-02	1.50e-01	3.72e-02	1.39e-01	3.09e-02
	500	6.73e-03	6.38e-03	1.27e+01	2.32e+00	1.88e+03	2.83e+02	**5.82e-03**	4.56e-03	5.95e+01	1.11e+01	2.71e+02	6.51e+01	1.79e+02	3.89e+01
	1 000	5.27e-03	5.56e-03	7.35e+01	9.36e+00	1.73e+04	1.70e+03	**3.63e-03**	3.69e-03	8.34e+03	5.01e+02	4.00e+03	4.61e+02	3.35e+03	3.43e+02
F_{10}	100	**5.08e-07**	9.46e-07	7.12e-01	3.08e-01	1.54e+01	1.25e+01	2.97e-01	3.18e-02	4.98e-03	1.47e-02	1.99e+00	6.09e-01	2.19e-02	4.31e-02
	500	**3.67e-02**	2.75e-02	1.07e+01	1.51e+00	2.65e+08	7.31e+07	6.83e-01	3.10e-02	7.58e+07	2.00e+07	1.19e+07	5.70e+06	6.74e+06	3.25e+06
	1 000	**3.62e-01**	3.19e-01	1.83e+01	4.22e+00	1.35e+09	1.70e+08	8.23e-01	2.08e-02	1.38e+09	1.03e+08	1.61e+08	2.56e+07	1.98e+08	3.82e+07

表4　各算法仿真实验的wilcoxon检验结果

wilcoxon rank-sum-test	D	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	+/−/=/gm
SW-OBLCSO vs CSO	100	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	500	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	1 000	+	+	+	+	+	+	+	+	+	+	10/0/0/10
SW-OBLCSO vs PSO	100	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	500	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	1 000	+	+	+	+	+	+	+	+	+	+	10/0/0/10
SW-OBLCSO vs OBLCPSO	100	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	500	+	+	+	+	+	+	+	+	=	+	9/0/1/9
	1 000	+	+	+	+	+	+	+	+	=	+	9/0/1/9
SW-OBLCSO vs SLPSO	100	+	+	+	+	+	+	+	+	+	−	9/1/0/8
	500	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	1 000	+	+	+	+	+	+	+	+	+	+	10/0/0/10
SW-OBLCSO vs RDPSO	100	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	500	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	1 000	+	+	+	+	+	+	+	+	+	+	10/0/0/10
SW-OBLCSO vs QPSO	100	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	500	+	+	+	+	+	+	+	+	+	+	10/0/0/10
	1 000	+	+	+	+	+	+	+	+	+	+	10/0/0/10

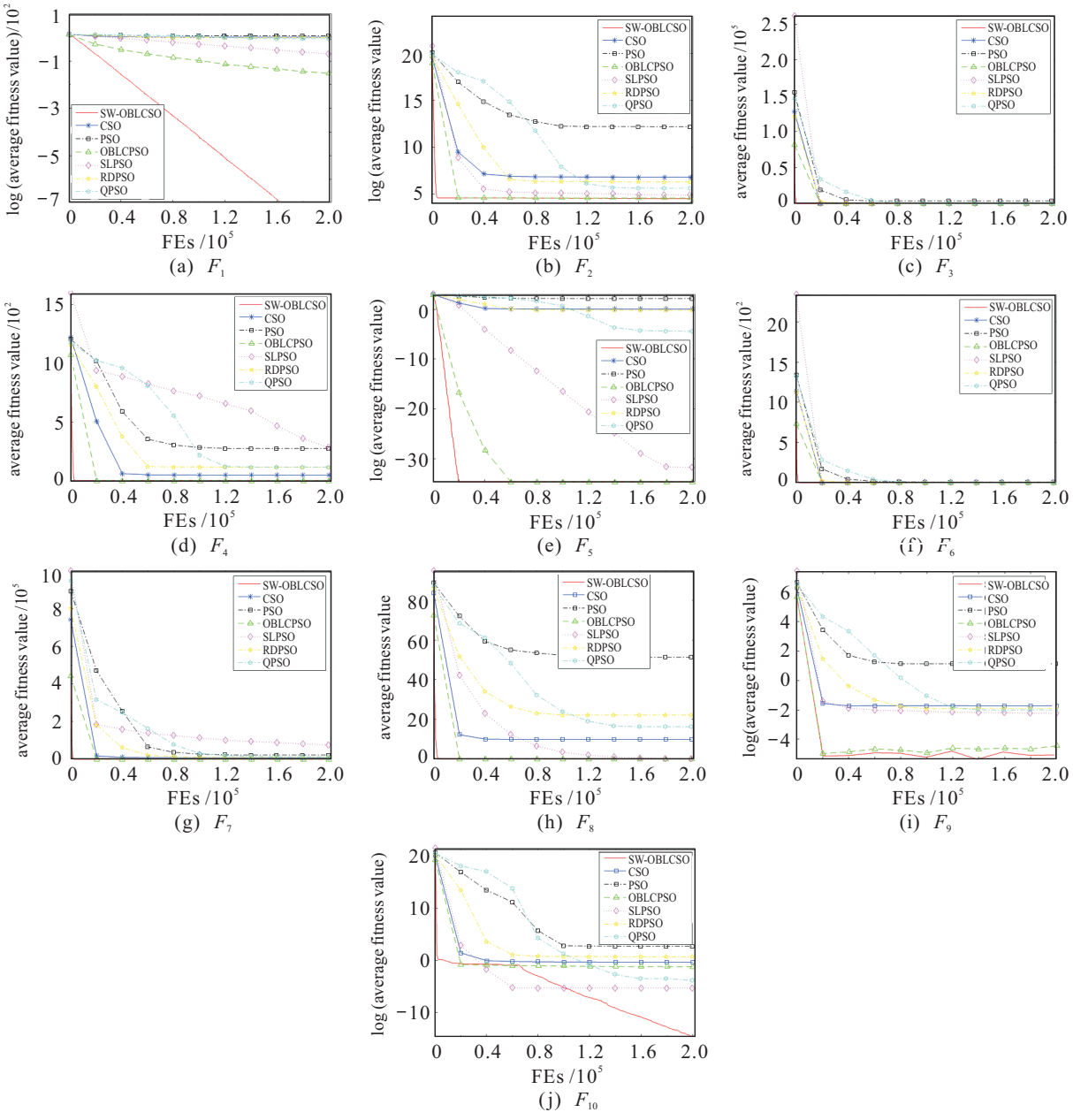


图 3　各算法优化10个测试函数(100维)的收敛曲线

从表3的实验数据可以看出,本文提出的SW-OBLCSO算法在10个标准测试函数上的优化性能总体上优于其他6个算法,且优化结果标准差较小. 与OBLCPSO算法相比: 在100维下,SW-OBLCSO算法在6个测试函数的性能更优越,在剩余4个测试函数上,两个算法的优化性能相当;在500维和1 000维下,SW-OBLCSO算法在5个测试函数的性能更优越,在1个测试函数上的性能不如OBLCPSO算法,在剩余4个测试函数上两者优化性能相当. 与其他5个算法相比,在各个维度下SW-OBLCSO算法在10个函数上的优化性能都有一定优势. 另外可以看出,随着维度增加,SW-OBLCSO算法依然可以保持良好的优化性能,具有良好的高维可扩展性. 从收敛曲线可以看出,采用反向学习的两种算法在大部分测试函数上有着最快的收敛速度,特别是在函数$F_3 \sim F_6$上,都能在很少的评估次数内寻找到全局最优点,也验证了反向学习机制与PSO算法结合是有效的. 对于函数F_{10},SLPSO算法的优化性能较好,在搜索过程的前中期始终保持着较快的收敛速度. CSO、OBLCPSO和SW-OBLCSO算法很快陷入了局部最优点,随着局部搜索策略的介入,SW-OBLCSO算法得以跳出局部最优,收敛速度得到了明显提升. 综上可知: 得益于特殊的竞争机制和反向学习机制,SW-OBLCSO算法拥有极高的搜索效率;由于结合了SW局部搜索算法,使得算法同时拥有很高的优化精度.

3.2 12个变形测试函数上的实验

在实际工程应用中,所求问题解的各维一般不相等. 因此,本节选用CEC'2005RPO测试集中12个带旋转、偏移的测试函数进行测试. 函数的维度为$D = 100$,相关信息如表5所示. 选用的对比算法以及参数设置如表2所示. 所有算法在每个测试函数上均独立运行25次,最大评估次数为max FEs = 200 000. 表6给出了各算法优化12个测试函数的数值结果. 部分测试函数上各算法的收敛曲线如图4所示.

从表6的实验数据可以看出: 对于函数F_1,SW-OBLCSO、 SLPSO和QPSO算法搜索到了全局最优值,RDPSO算法效果相差不多;对于函数F_2、F_6、F_8、F_{12},SW-OBLCSO算法的收敛精度最好;对于函数F_5、F_9、F_{11},SLPSO算法的收敛精度最好;对于函数F_3和F_7,SLPSO算法与SW-OBLCSO算法的收敛精度一致;对于函数F_4,QPSO算法得到了最好的优化结果;CSO算法在函数F_{10}上收敛精度最好;OBLCPSO算法在大部分测试函数上的收敛精度最差;基于原点的反向学习策略对经过偏移、旋转后的测试函数适用度不高.

分析收敛曲线: 对于函数F_1、F_{12},SW-OBLCSO算法陷入了局部最优点,由于改进的竞争机制花费很少的代价为局部搜索策略提供了一个可行的初始解,使得算法在迭代结束时也获得了最好的收敛精度;对于带有噪声的函数F_4,由于噪声的影响使得局部搜索策略没有介入到搜索过程;对于函数F_{10},SW搜索在迭代后期并不能继续改进解的质量,体现了SW-OBLCSO算法的局限性.

综上所述,在处理带噪声的函数以及局部搜索策略不适用的函数时,SW-OBLCSO算法表现出了一定的局限性,在总体上优化效果好于其他对比算法.

表5 带有偏移、旋转的测试函数

函数	名称	类型	搜索范围	最小值
F_1	Shifted Sphere function	单峰	$[-100, 100]$	-450
F_2	Shifted Schwefel's problem 1.2	单峰	$[-100, 100]$	-450
F_3	Shifted rotated high conditioned elliptic function	单峰	$[-100, 100]$	-450
F_4	Shifted Schwefel's problem 1.2 with noise in fitness	单峰	$[-100, 100]$	-450
F_5	Schwefel's problem 2.6 with global optimum on bounds	单峰	$[-100, 100]$	-310
F_6	Shifted rosenbrock's function	多峰	$[-100, 100]$	390
F_7	Shifted rotated griewank's function without bounds	多峰	$[0, 600]$	-180
F_8	Shifted rotated ackley's with global optimum on bounds	多峰	$[-32, 32]$	-140
F_9	Shifted rastrigin's function	多峰	$[-5, 5]$	-330
F_{10}	Shifted rotated rastrigin's function	多峰	$[-5, 5]$	-330
F_{11}	Shifted rotated weierstrass function	多峰	$[-0.5, 0.5]$	90
F_{12}	Schwefel's problem 2.13	多峰	$[-100, 100]$	-460

表6　各算法在12个变形函数上的实验结果

函数	指标	SW-OBLCSO	CSO	PSO	OBLCPSO	SLPSO	RDPSO	QPSO
F_1	mean	**−4.50e+02**	6.30e+01	5.29e+04	1.01e+05	**−4.50e+02**	−449.999 6	**−4.50e+02**
	std.	0.00e+00	2.51e+02	8.55e+03	9.85e+03	0.00e+00	6.45e-04	0.00e+00
F_2	mean	**3.98e+03**	9.49e+04	1.80e+05	2.82e+05	7.60e+04	4.87e+04	4.89e+04
	std.	5.63e+03	8.87e+03	2.19e+04	2.90e+04	1.41e+04	8.65e+03	7.70e+03
F_3	mean	**1.243e+07**	1.24e+08	9.81e+08	1.75e+09	**1.241e+07**	5.79e+07	4.47e+07
	std.	8.03e+06	3.41e+07	3.81e+08	4.52e+08	2.90e+06	2.17e+07	1.00e+07
F_4	mean	2.85e+05	1.76e+05	3.28e+05	3.90e+05	1.60e+05	1.52e+05	**1.35e+05**
	std.	3.05e+04	2.08e+04	8.24e+04	5.86e+04	3.04e+04	3.59e+04	3.34e+04
F_5	mean	1.56e+04	3.39e+04	5.52e+04	6.06e+04	**4.59e+03**	2.53e+04	2.12e+04
	std.	2.37e+03	2.07e+03	5.25e+03	2.71e+03	8.09e+02	2.37e+03	1.86e+03
F_6	mean	**4.86e+02**	5.14e+07	4.94e+09	1.95e+10	5.03e+02	1.25e+03	7.18e+02
	std.	3.87e+00	3.40e+07	1.65e+09	3.61e+09	3.39e+01	1.07e+03	7.26e+01
F_7	mean	**−179.997 9**	2.66e+02	1.61e+03	3.72e+04	**−179.991 6**	−1.25e+02	−1.72e+02
	std.	4.00e-03	1.19e+02	4.14e+02	3.62e+03	9.93e-03	3.87e+01	4.49e+00
F_8	mean	**−119.946 4**	−118.940 6	−118.725 9	−118.885 2	−118.658 6	−118.656 8	−118.650 8
	std.	2.48e-02	2.76e-01	5.12e-02	8.97e-02	2.78e-02	1.93e-02	2.76e-02
F_9	mean	3.12e+02	−4.95e+01	5.78e+02	8.66e+02	**−2.36e+02**	−9.03e+01	−1.57e+02
	std.	7.23e+01	2.81e+01	5.72e+01	4.27e+01	1.42e+01	2.62e+01	1.94e+01
F_{10}	mean	6.65e+02	**4.47e+01**	1.10e+03	1.22e+03	5.32e+02	3.48e+02	3.55e+02
	std.	1.27e+02	2.62e+01	1.21e+02	8.65e+01	1.91e+01	2.72e+02	2.15e+02
F_{11}	mean	2.16e+02	1.58e+02	2.32e+02	2.32e+02	**1.19e+02**	2.45e+02	2.34e+02
	std.	1.22e+01	5.18e+00	7.64e+00	5.88e+00	5.41e+00	8.77e+00	2.10e+01
F_{12}	mean	**2.92e+04**	1.49e+06	8.69e+06	1.79e+07	2.64e+07	3.16e+07	3.15e+07
	std.	2.05e+04	3.91e+05	2.04e+06	1.43e+06	4.81e+06	1.63e+06	1.44e+06

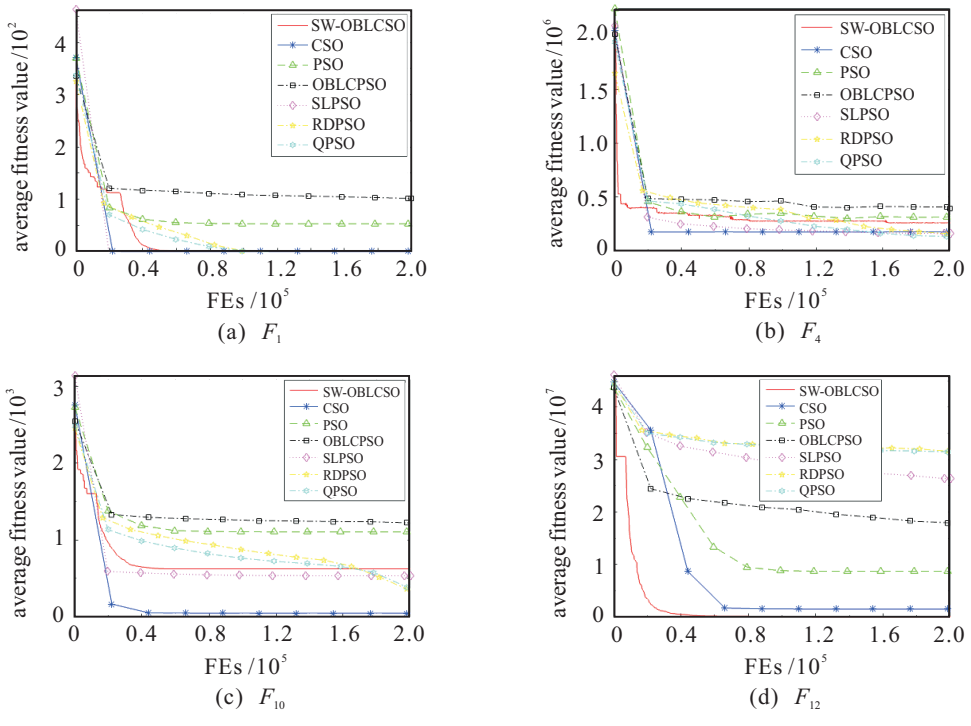


图4　各算法优化12个变形函数的部分收敛曲线

3.3 模糊认知图学习问题上的应用

模糊认知图(FCM)是用于构建复杂系统模型的有用工具. FCM是一个有向模糊图,由一组节点和节点间的有向加权边构成. 节点表示诸如值、目标等的真实世界概念,加权有向边代表概念(节点)之间的因果关系. 本节实验分别使用人工数据以及标准测试集DREAM 4[23]来测试SW-OBLCSO算法的性能. 人工数据按照文献[24]中的方法生成,包含20个节点,5个响应序列,每个序列有11个时间节点,边密度为20%. 对于DREAM 4,取基因规模为10的5个响应序列,每个序列取后11个不包含扰动的时间节点.

FCM学习问题的目标函数是评估给定响应序列与生成的响应序列之间的差异,可以定义如下:

$$\mathrm{Data\_Error}(W)=\frac{\sum\limits_{s=1}^{N_s}\sum\limits_{t=1}^{N_t-1}\sum\limits_{n=1}^{N_n}(C_n^t(s)-\hat{C}_n^t(s))^2}{N_nN_s(N_t-1)},\tag{9}$$

$$C_n^{t+1}=g\Big(\sum_{j=1}^{N_n}w_{ji}C_j^t\Big),\ i=1,2,\ldots,N_n,\tag{10}$$

$$g(x)=\frac{1}{1+\exp^{-5x}}.\tag{11}$$

其中: $N_n=10$是节点(概念)的数量, $N_t=10$是响应序列中使用的时间点的数量, $N_s=5$是响应序列的数量, $C_n^t(s)$和$\hat{C}_n^t(s)$分别是给定的和生成的第s个响应序列中第t个时间点下第n个概念的值. 概念的值在[0, 1]范围内,边的权重在[−1, 1]之间.

Out_of_Sample_Error用于评估算法在处理过拟合问题时的能力,具体计算公式如下:

$$\mathrm{Out\_of\_Sample\_Error}(W)=\frac{1}{N_nN_s(N_t-1)}\cdot\sum_{s=1}^{N_s}\sum_{t=1}^{N_t-1}\sum_{n=1}^{N_n}|C_n^t(s)-\hat{C}_n^t(s)|.\tag{12}$$

为了把粒子群算法应用到FCM学习问题上,将FCM的权重矩阵转换为向量并表示为一个粒子. 粒子是实数编码的,作为候选解决方案处理. 由于真实数据与人工数据节点数的不同,这里粒子的维度分别为100和400. 种群大小为$100+D/10$,最大评估次数为200 000. 上一节中使用的其他6种算法也用于解决FCM学习问题以进行比较,同时由于实际问题更为复杂,局部最优点较多,根据2.3节的策略分析,对SW-OBLCSO算法中失败粒子更新公式设置一个较大的偏置作为对比,令偏置量系数$i=10$. 算法的参数设置与表2中的相同. 所有算法均独立运行25次, Data_Error 和Out_of_Sample_Error的平均值和标准差记录在表7中. 根据表7中的结果, SW-OBLCSO算法在增大偏置量以后其优化性能得到了提升,在Data_Error上和Out_of_Sample_Error上都有最好的收敛精度. 在使用较小偏置量时,算法在陷入局部最优点之后,通过局部搜索策略同样得到了比对比算法更好的优化结果.

表7 各算法在FCM学习问题上的实验结果

指标	SW-OBLCSO ($i=20$)	SW-OBLCSO ($i=10$)	CSO	PSO	OBLCPSO	SLPSO	RDPSO	QPSO
Data_Error_Real(W)	3.32e-03	**3.29e-03**	5.09e-03	7.46e-03	7.26e-03	4.80e-03	4.68e-03	4.63e-03
	6.90e-05	8.53e-05	2.13e-04	1.58e-03	3.41e-04	6.90e-04	2.81e-04	4.79e-04
Out_of_Sample_Error_R(W)	4.69e-02	**4.67e-02**	5.58e-02	6.81e-02	6.39e-02	5.62e-02	5.47e-02	5.58e-02
	1.31e-03	1.10e-03	1.52e-03	9.15e-03	1.66e-03	4.62e-03	2.08e-03	2.16e-03
Data_Error_Synthetic(W)	1.13e-02	**9.85e-03**	1.78e-02	4.91e-02	4.09e-02	2.64e-02	2.45e-02	1.67e-02
	4.95e-03	6.31e-03	3.94e-03	4.86e-03	1.04e-03	1.66e-03	4.56e-03	3.70e-03
Out_of_Sample_Error_S(W)	8.37e-02	**7.74e-02**	9.47e-02	1.41e-01	1.42e-01	9.83e-02	9.69e-02	8.90e-02
	1.54e-02	1.47e-02	1.12e-02	1.00e-02	3.61e-03	4.79e-03	8.88e-03	1.23e-02

4 结 论

本文通过引入反向学习机制和局部搜索策略,提出了一种基于局部搜索的反向学习竞争粒子群优化算法. 竞争学习机制每次在群体中的4个粒子之间进行,通过比较它们的适合度值决定更新方式: 具有最差适应性的粒子获得最优适应度值的粒子位置并添加一个偏移量,让算法获得更快的收敛速度; 拥有第2适应度值的粒子通过反向学习来更新自身; 第3适应度值的粒子通过从具有最佳适应性的粒子中学习来更新其位置和速度. 竞争学习与算法中的反向学习、局部搜索相结合,有助于算法具有良好的探索和开发能力. 对多个测试函数的实验结果表明, SW-OBLCSO算法优于其他6个PSO算法变体. 在解决FCM的学习问题上, SW-OBLCSO算法也表现出良

好的性能.

参考文献(References)

[1] Eberhart R, Kennedy J. Particle swarm optimization[C]. IEEE International Conference on Neural Networks. Perth: IEEE, 1995: 1942-1948.

[2] Shi Y, Eberhart R. Modified particle swarmoptimizer[C]. 1998 IEEE World Congress on Computational Intelligence. Anchorage: IEEE Xplore, 1998: 69-73.

[3] Poli R. Analysis of the publications on the applications of particle swarm optimisation[J]. Journal of Artificial Evolution & Applications, 2008, 2008: 1-10.

[4] Banks A, Vincent J, Anyakoha C. A review of particle swarm optimization, Part I: Background and development[J]. Natural Computing, 2007, 6(4): 467-484.

[5] Banks A, Vincent J, Anyakoha C. A review of particle swarm optimization, Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications[J]. Natural Computing, 2008, 7(1): 109-124.

[6] 范培蕾, 张晓今, 杨涛. 克服早熟收敛现象的粒子群优化算法[J]. 计算机应用, 2009, 29(S1): 122-124.
(Fan P L, Zhang X J, Yang T. Particle swarm optimization algorithm for overcoming premature convergence[J]. Journal of Computer Applications, 2009, 29(S1): 122-124.)

[7] Kennedy J, Mendes R. Neighborhood topologies in fully informed and best-of-neighborhood particle swarms[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2006, 36(4): 515-519.

[8] 焦重阳, 周清雷, 张文宁. 混合拓扑结构的粒子群算法及其在测试数据生成中的应用研究[J]. 计算机科学, 2017, 44(12): 249-254.
(Jiao C Y, Zhou Q L, Zhang W N. Hybrid topology structured particle swarm optimization algorithm and its application in test data generation[J]. Computer Science, 2017, 44(12): 249-254.)

[9] Jin Y X, Zhang X, Xue D. Ring-shaped full-informed particle swarm optimization algorithm with adaptive escape for a ring full interconnection Structure[J]. Microelectronics & Computer, 2018, 35(2): 1-5.

[10] 许胜才, 蔡军, 程昀, 等. 基于拓扑结构与粒子变异改进的粒子群优化算法[J]. 控制与决策, 2019, 34(2): 419-428.
(Xu S C, Cai J, Cheng Y, et al. Improved particle swarm optimization algorithm based on topology and particle variation[J]. Control and Decision, 2019, 34(2): 419-428.)

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

[12] Fang W, Sun J, Wu X J, et al. Study on the compression-expansion coefficient in drift particle swarm optimization[C]. 2012 IEEE Congress on Evolutionary Computation. Brisbane: IEEE, 2012: 1-6.

[13] Tizhoosh H R. Opposition-based learning: A new scheme for machine intelligence[C]. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce. Vienna: IEEE, 2005, 1: 695-701.

[14] Wang H, Li H, Liu Y, et al. Opposition-based particle swarm algorithm with Cauchy mutation[C]. 2007 IEEE Congress on Evolutionary Computation. Singapore: IEEE, 2007: 4750-4756.

[15] Zhou L Y, Ding L X, Peng H, et al. A particle swarm optimization algorithm for neighborhood gravity reverse learning[J]. Acta Electronica Sinica, 2017, 45(11): 2815-2824.

[16] Zhou J, Fang W, Wu X, et al. An opposition-based learning competitive particle swarm optimizer[C]. 2016 IEEE Congress on Evolutionary Computation. Vancouver: IEEE , 2016: 515-521.

[17] Cheng R, Jin Y. A competitive swarm optimizer for large scale optimization[J]. IEEE Transactions on Cybernetics, 2014, 45(2): 191-204.

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

[19] Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition-based differential evolution[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 64-79.

[20] Fang W, Sun J, Ding Y R, et al. A review of quantum-behaved particle swarm optimization[J]. IETE Technical Review, 2010, 27(4): 336-348.

[21] Cheng R, Jin Y C. A social learning particle swarm optimization algorithm for scalable optimization[J]. Information Sciences, 2015, 291: 43-60.

[22] Auger A, Hansen N. A restart CMA evolution strategy with increasing population size[C]. 2005 IEEE Congress on Evolutionary Computation. Edinburgh: IEEE, 2005: 1769-1776.

[23] Greenfield A, Madar A, Ostrer H, et al. DREAM4: Combining genetic and dynamic information to identify biological networks and dynamical models[J]. PloS One, 2010, 5(10): e13397.

[24] Chen Y, Mazlack L, Lu L. Learning fuzzy cognitive maps from data by ant colony optimization[C]. Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. Philly: ACM, 2012: 9-16.

作者简介

钱晓宇(1995–), 男, 硕士, 从事粒子群优化算法的研究, E-mail: 6161910030@vip.jiangnan.edu.cn;

方伟(1980–), 男, 教授, 博士生导师, 从事计算智能等研究, E-mail: fangwei@jiangnan.edu.cn.

(责任编辑: 李君玲)