doi:10. 16055/j. issn. 1672-058X. 2022. 0003. 002

基于平均位置学习的改进粒子群算法研究

杨妹兰[1], 刘衍民[2], 张 倩[1], 舒小丽[3]
(1. 贵州大学 数学与统计学院,贵阳 550025;2. 遵义师范学院 数学学院, 贵州 遵义 563006;
3. 贵州民族大学 数据科学与信息工程学院, 贵阳 550025)

摘 要:针对粒子群算法在求解复杂的多维多峰问题时,存在着局部搜索精度不高和易陷入局部最优等不足,提出了一种基于平均位置学习的改进粒子群算法。该算法在学习策略上采用比粒子自身适应值更好的邻近粒子为学习对象,将该算法分两个阶段用不同更新速度公式,阶段一在更新速度公式中引入整个种群所有粒子位置的平均位置;阶段二在速度更新公式中引入新平均位置,采用贪心策略选择,通过粒子每次更新后选择的个体比种群历史最优适应值更优,且储存对应个体历史最优位置,在阶段一结束后开始求它们的平均位置。将平均位置作为学习对象,可增强粒子间的信息交流,同时可平衡算法的局部开发性能与全局搜索能力。在 CEC2017 测试函数实验中,实验结果显示所提改进算法与另外 4 个算法相比有一定的优势。

关键词:粒子群算法;平均位置;信息交流
中图分类号:TP18 **文献标志码**:A **文章编号**:1672-058X(2022)03-0009-11

0 引 言

粒子群算法(Particle Swarm Optimization, PSO)[1]是 1995 年由 Eberhart 和 Kennedy 提出的一种基于模拟鸟类觅食行为过程的随机智能优化算法。在 PSO 中,种群的每个粒子表示所研究问题的候选解,每个粒子通过向个体历史最优位置和种群历史最优位置学习来更新当前速度和位置。PSO 由于具有结构简单、寻优能力强、容易实现和收敛速度快等优点,近几年被广泛应用于各个领域,如煤炭[2]、工业生产[3]和水电调度[4]等。但是,PSO 仍然存在一些问题,比如容易出现早熟问题、维数灾难问题和易陷入局部最优解等。

为了提高 PSO 的综合性能,许多研究者对 PSO 存在的问题进行相了关的改进研究,主要的改进分为:

(1) 邻居拓扑结构的改进。研究者们已经设计了许多不同类型的拓扑结构,从而以很多不同方式来改进粒子的学习策略。如 Kennedy 等[5-6]提出了几类基本的领域拓扑结构,如环形结构、星型结构和齿形结构等,还分析了几种拓扑结构对优化问题结果的作用。文献[7]利用全局版本和局部版本拓扑结构相结合,从而改进了粒子的学习策略,有效提高算法的寻最优解能力和收敛性能。

(2) 优化粒子速度的更新过程。粒子的速度更

收稿日期:2021-04-31;**修回日期**:2021-06-29.
基金项目:国家自然科学基金(71461027);贵州省科技创新人才团队(黔科合平台人才[2016]5619).
作者简介:杨妹兰(1996—),女,贵州黔东南人,硕士研究生,从事优化理论及智能算法研究.
通讯作者:刘衍民(1978—),男,山东临沂人,教授,博士生导师,从事优化理论和智能算法研究. Email:yanmin7813@ 163. com.

新公式由学习样本的选择进行决定,如文献[8]利用个体间的欧式距离大小来选择每个个体的学习样本,有效增强了种群的多样性;文献[9]在粒子速度更新公式中引入全局最优位置和综合最优位置作为学习对象,提高了粒子的搜索能力;文献[10]引入免疫算法的粒子群优化方法,既增强了每个粒子的搜索能力,还提高了算法的收敛性能;文献[11]提出一种基于贝叶斯迭代法的综合学习粒子群算法,增强了算法的局部最优规避能力。这些改进方法对提高算法的运行效率,跳出局部最优解具有一定作用。但是,在遇到复杂的多维多峰问题时,收敛效果不理想、全局搜索能力和局部开发能力不协调等问题依然存在。

因此,本文针 PSO 的特性和不足,提出了一种结合适应度距离比值和平均位置的改进粒子群算法 MLFDR。通过与 4 个不同算法进行实验对比与分析,证明 MLFDR 能够更好地平衡局部开发和全局搜索,同时算法在求解精度和收敛性能上得到了显著提高。

1 基本粒子群算法

标准粒子群算法(PSO)是一种生物进化算法,它是受到鸟类觅食行为的启发。在研究 PSO 时,把种群中每个粒子看作 D 维搜索空间中的一个搜索个体,个体的当前位置视为优化问题的一个可能解,即粒子的飞行过程可视为对应个体的搜索过程。种群中粒子通过向个体历史最优位置和种群历史最优位置学习来更新当前速度和位置。

假设搜索空间为 D 维,搜索空间中粒子数有 ps 个,其中第 $i(i\leqslant ps)$ 个粒子的第 t 代位置和速度可以分别用两个指标来描述:位置可用一个 D 维向量 $x_i^t=(x_{i1}^t,x_{i2}^t,\cdots,x_{ij}^t,\cdots,x_{iD}^t)$ 来表示,飞行速度可用一个 D 维向量 $v_i^t=(v_{i1}^t,v_{i2}^t,\cdots,v_{ij}^t,\cdots,v_{iD}^t)$ 来表示。如果第 i 个粒子搜索到第 t 代时的个体历史最优位置为一个 D 维向量,即 $pbest_i=(pbest_{i1},pbest_{i2},\cdots,pbest_{ij},\cdots,pbest_{iD})$,搜索到第 t 代时的种群历史最优位置为一个 D 维向量,即 $gbest=(gbest_1,gbest_2,\cdots,gbest_j,\cdots,gbest_D)$。因此,到第 $t+1$ 代时,粒子的速度和位置迭代更新公式分别如下:

$$v_{ij}^{t+1}=w\cdot v_{ij}^t+c_1\cdot r_1\cdot(pbest_{ij}-x_{ij}^t)+c_2\cdot r_2\cdot(gbest_j-x_{ij}^t) \quad (1)$$

$$x_{ij}^{t+1}=x_{ij}^t+v_{ij}^{t+1} \quad (2)$$

其中:w 表示惯性权重因子,是用来衡量上一时刻的飞行速度对当前时刻飞行移动速度的影响;c_1 和 c_2 表示加速因子,通常取值为 2;r_1 和 r_2 表示[0,1]范围内满足均匀分布的随机数;v_{ij}^t、x_{ij}^t 分别表示第 i 个粒子在第 t 次迭代中第 j 维的速度和位置;$pbest_{ij}$ 表示第 i 个粒子的个体历史最优位置的第 j 维位置,$gbest_j$ 表示种群历史最优位置的第 j 维位置。

2 改进的粒子群算法

2.1 贪心策略

贪心策略总是在当前条件下为最优的选择,换句话说,该策略不是从整体上进行选择,而它的选择只是在某种意义下的局部最优解,因此,很多问题自身的特点就决定着该问题采用贪心策略就能获得最优解或者较优解。

定义 1 贪心策略是指从问题的初始状态出发,通过若干次的贪心选择而得出最优值(或较优解)的一种解决问题方法。

本文改进的 PSO 应用贪心策略,粒子群在每次更新位置后,对局部搜索的结果个体历史最优位置($pbest$)采用贪心策略选择,即

$$pbest'_j=\begin{cases}pbest_j, & (pbestval_j>gbestval)\\ gbest, & \text{其他}\end{cases} \quad (4)$$

式(4)中,$pbest'_j$ 表示该策略选择的第 j 个最佳对象;$pbest_j$ 表示第 j 个粒子的个体历史最优位置;$pbestval_j$ 表示第 j 次粒子的个体历史最优适应值;$gbestval$ 表示种群历史最优适应值。

2.2 MLFDR 算法

PSO 具有参数比较少、易实现以及操作方便等优点,但算法也具有寻优精度较低、前期收敛速度偏慢以及容易陷入局部最优解等问题,种群中的每个

粒子仅由它的个体历史最优位置和种群历史最优位置决定。针对 PSO 的上述情况,本文引入比粒子自身适应值更好的邻近粒子和平均位置[12]作为学习对象。在学习策略中引入比粒子自身适应值更好的邻近粒子,使种群避免陷入局部最优解,还增强粒子的搜索能力。

在选择过程中,考虑最简单和最稳定的变化来选择一个适应值之差与一维位置距离的比值最大对应粒子 *nbest*,用粒子 *nbest* 来更新当前粒子速度的每个分量,可提高算法的寻优能力和收敛能力。选择最大比值公式为

$$B_i=\max\left[\frac{F(\tilde{x}_j)-F(x_i)}{|\tilde{x}_{jd}-x_{id}|}\right] \tag{5}$$

其中,B_i 表示最大比值;$\tilde{x}_i$ 表示第 i 个粒子的邻近粒子位置;$F(\tilde{x}_j)$ 表示 $\tilde{x}_j$ 的适应值;x_i 表示第 i 个粒子;$F(x_i)$ 表示为 x_i 的适应值;$\tilde{x}_{jd}$ 表示为第 j 个粒子的第 d 维个体历史最优位置;x_{id} 表示为第 i 个粒子的第 d 维位置;在|…|为绝对值,最大比值 B_i 解得比粒子 x_i 自身适应值更好的邻近粒子记为 $nbest_{id}$。

在学习策略中引入平均位置作为学习对象,可提升种群的多样性和改善算法的学习策略,同时还能提高算法的运行效率。该算法分为两个阶段更新粒子[13],具体改进策略如下。

2.2.1　前期引入种群的平均位置的改进

阶段一是指算法前期更新至第 $2\times ps$ 次的部分(ps 表示种群规模),改进之处是在速度更新公式中引入种群的平均位置作为学习对象,该平均位置综合了种群所有粒子的飞行行为,使算法在搜索过程具有一定的普遍性,可以保证算法的收敛性,同时提升了种群的多样性,还提高了每个粒子飞行过程的稳定性,还能更好地平衡局部开发能力和全局搜索能力,从而有效增强了算法的寻优能力。在每一次更新前,先计算出当前所有粒子位置的平均位置,具体平均位置公式如下:

$$mp(1,j)=\frac{\sum_{i=1}^{ps}x_{ij}^{t}}{ps} \tag{6}$$

$$h=\beta\times\frac{d}{ps} \tag{7}$$

其中,$mp(1,j)$ 表示第 j 维的平均位置;h 参数是一个社会影响因素;d 表示维数变量($d\leqslant D$),取 $\beta=0.01$。

一般情况下,种群的收敛要求每个粒子位置向量的每个维度收敛,所以算法的收敛难度与搜索维数成正比。在搜索维数的基础上建立社会影响因素 h,有利于收敛性与多样性之间有良好的平衡状态。根据观察,参数 h 与优化问题维数 d 成正比,可以控制种群平均位置的影响;如果该参数的值取得很大,可能会出现过早收敛到平均位置的情况,从而不能到达最佳行为。

在阶段一采用贪心策略,每次在当前粒子的速度和位置更新后,将更新后的个体历史最优位置适应值(*pbestval*)与种群历史最优适应值(*gbestval*)比较,若更新后的个体历史最优适应值(*pbestval*)更优,则将对应的个体历史最优位置(*pbest*)储存起来,还将个体历史最优位置和适应值分别赋值给种群历史最优位置和适应值;反之就不用储存且种群历史最优位置和适应值保留原值。具体公式如下:

$$y=\begin{cases}1, pbest_j=pbest'_j\\ 0, pbest_j\neq pbest'\end{cases} \tag{8}$$

$$pbest_{(k)}=pbest'_j\times sgn(y) \tag{9}$$

其中,*sgn* 表示符号函数;$pbest_{(k)}$ 表示算法储存的第 k 个个体历史最优位置。

2.2.2　后期引入最佳位置的平均位置改进

阶段二是指算法从第 $2\times ps$ 次评价之后的部分,粒子个体的收敛速度较快,但容易出现局部最优停滞的现象。为了使算法效果更好,在算法后期引入更新后储存的所有个体历史最优位置的平均位置,去替换阶段一的平均位置,既可以使算法跳出停滞现象,还提升了局部搜索精度,而且阶段一的优点依然保持,还能够很好地协调算法的局部开发能力与全局搜索能力。在本阶段更新后继续使用贪心策略,若与种群历史最优位置比较后得个体历史最优适应值更优,则储存对应个体历史最优位置。在阶段一结束后,开始计算储存中所有个体历史最优位

置的平均位置,具体平均公式如下:

$$mp(1,j)=\frac{\sum_{k=1}^{u} pbest_{(k)j}}{u} \tag{10}$$

其中,u 表示储存个体历史最优位置的个数;$pbest_{(k)j}$ 表示储存中被标记的第 k 个粒子第 j 维的个体历史最优位置;mp 表示平均位置。

MLFDR 的学习对象除了个体历史最优位置和种群历史最优位置外,还添加了两个学习对象,即比粒子自身适应值更好的邻近粒子($nbest$)和平均位置(mp),可保证算法改进的优点,还能使整个算法达到更佳的效果。粒子的迭代更新速度公式如下:

$$v_{ij}^{t+1}=w\cdot v_{ij}^{t}+c_1\cdot r_1\cdot(pbest_{ij}-x_{ij}^{t})+c_2\cdot r_2\cdot(gbest_j-x_{ij}^{t})+c_3\cdot r_3\cdot(nbest_{ij}-x_{ij}^{t})+h\cdot c_4\cdot r_4\cdot(mp-x_{ij}^{t}) \tag{11}$$

其中,$nbest_{ij}$ 表示最佳适应度距离比值对应的粒子位置;mp 表示平均位置,阶段一用式(6)计算平均位置,阶段二用式(10)计算平均位置;h 表示社会影响因素,控制种群平均位置的影响,还可以更好地平衡种群多样性与收敛性之间的状态。

2.3 改进粒子群算法的基本步骤

(1) 种群初始化。种群初始化的粒子是以随机方式来生成初始位置和速度。

(2) 计算适应值。计算每个粒子的适应度值,然后,确定当前的个体历史最优位置和种群历史最优位置,将个体最优的适应度值存储在 *pbestval* 中,将种群最优的适应度值存储在 *gbestval* 中。

(3) 按照式(5)计算最大值比,求得 *nbest*;用式(7)计算 h;在前 *ps* 次更新时,需按照式(6)计算种群所有粒子位置的平均位置 *mp*。

(4) 更新粒子位置。按照式(11)和式(2)来更新当前粒子的速度和位置。

(5) 更新个体最优。将粒子更新后的适应度值与其未更新前的适应度进行比较,取适应值较好的粒子作为当前迭代后的粒子。

(6) 更新种群最优。通过比较个体历史最优适应值和种群历史最优适应值,若个体历史最优适应值较好,将对应的个体历史最优位置储存起来,且将个体历史最优适应值赋值给种群历史最优适应值的方式更新种群历史最优位置。

(7) 判断算法是否评价至第 2×*ps* 次,若是则按照式(10)计算 *mp* 后,将 *mp* 赋值给步骤(3)的 *mp*。

(8) 算法终止。如果满足最大迭代数结束条件,算法搜索停止;否则返回到步骤(3)继续搜索。

MLFDR 流程如图 1 所示。

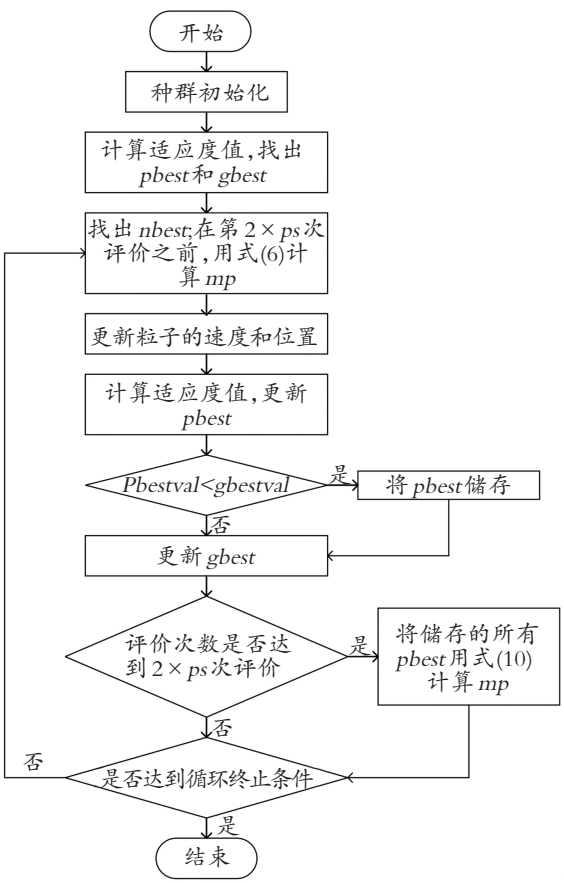


图 1 MLFDR 算法流程

Fig. 1 MLFDR algorithm flow

3 MLFDR 算法性能测试与分析

3.1 检测函数选取

为了测试 MLFDR 的收敛速度和全局搜索能力,利用 CEC2017 测试函数中的单峰和多峰共 8 个测试函数进行函数优化的对比实验。其中,f_4 和 f_5 为单峰函数,f_1、f_2、f_3、f_6、f_7 和 f_8 为多峰函数,它们分别用于检验不同算法的搜索速度、寻优能力和全局搜索能力。具体测试函数见表 1。

表 1　本文选用的 8 个测试函数

Table 1　Eight test functions selected in this paper

类　型	函数名称	测试函数	维　数
单峰函数	High Conditioned Elliptic 函数	$f_4 = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} \times x_i^2$	20
	Discus 函数	$f_5 = 10^6 \times x_1^2 + \sum_{i=2}^{D} x_i^2$	20
多峰函数	Rastrigin 函数	$f_1 = \sum_{i=1}^{D} (x_i^2 - 10 \times \cos(2\pi x_i) + 10)$	20
	Rosenbrock 函数	$f_2 = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	20
	Non−continuous Rotated Rastrigin 函数	$f_3 = \sum_{i=1}^{D} (z_i^2 - 10 \times \cos(2\pi z_i) + 10) + f_{13}^*$ $y_1 = \begin{cases} \hat{x}_i & \text{if } \lvert \hat{x}_i \rvert \leqslant 0.5 \\ round(2\hat{x}_i) & \text{if } \lvert \hat{x}_i \rvert > 0.5 \end{cases}$ for $i = 1,2\cdots,D$ $\hat{x} = M_1 \times \frac{5.12(x-o)}{100}$, $z = M_1 \wedge^{10} M_2 T_{asy}^{0.2}(T_{osz}(y))$,其中 $\wedge^{\alpha}$ 表示对角线的第 i 元素为 $\lambda_{ii} = \alpha^{\frac{i-1}{2(D-1)}}$ 的 D 维对角矩阵($i = 1,2,\cdots,D$)	20
	Ackley 函数	$f_6 = -20 \times \exp(-2 \times \sqrt{\frac{1}{D} \times \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \times \sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + e$	20
	Happy Cat 函数	$f_7 = \lvert \sum_{i=1}^{D} x_i^2 - D \rvert^{\frac{1}{4}} + (0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$	20
	HGBat 函数	$f_8 = \lvert (\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2 \rvert^{\frac{1}{2}} + (0.5 \times \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$	20

为了增加这些检测函数的优化难度,将部分函数都进行旋转,用“ * ”表示对应测试函数进行旋转 。选择 4 个不同算法与本文提出的 MLFDR 相比较,4 个不同算法分别是:PSO[1]、FDR[8]、wFIPS[14] 和 UPSO[7]算法。

3.2　算法参数设置

选取 FDR、PSO、wFIPS 和 UPSO 算法进行对比,为了体现不同算法之间的可比性,实验中所有算法都取参数保持一致,种群规模是 10,每个测试函数单独运行了 50 次,对于非旋转的测试函数则每次运行进行 3×10^4 函数评价,旋转的测试函数则每次运行进行 2×10^4 函数评价,其他参数设置见表 2。

表 2　对比 5 个算法的具体参数设置

Table 2　Comparison of the specific parameter settings of five algorithms

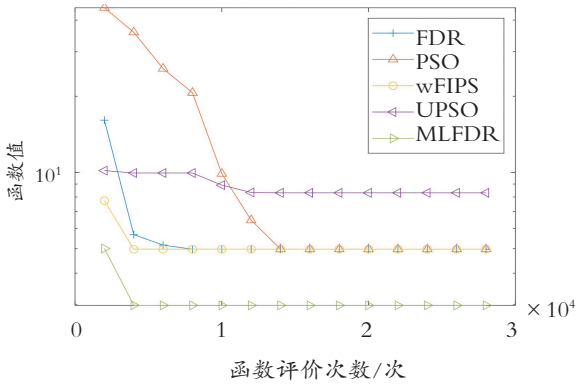
算　法	参数设置
FDR	w:[0.4,0.9],$c_1=c_2=1$,$c_3=2$
PSO	w:[0.4,0.9],$c_1=c_2=2$
w FIPS	$w=0.729$, $\chi=0.729$, $c_1=c_2=2$
UPSO	$w=0.729$,$c_1=c_2=1.494\,45$,$u=0.1$
MLFDR	w:[0.4,0.9],$c_1=1$,$c_2=0.5$,$c_3=1.5$,$c_4=1$

3.3　实验结果和分析

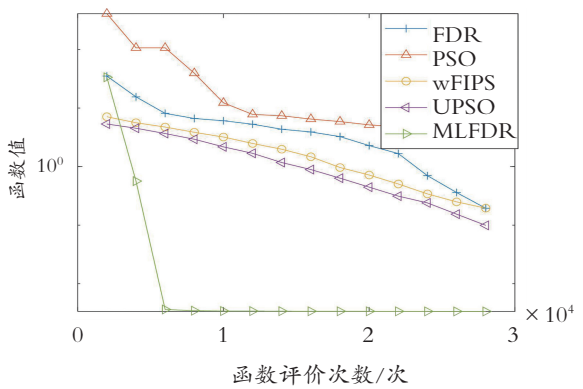
3.3.1　收敛特性仿真实验

为了检测不同算法的收敛特性,给出在确定的函数评价次数下不同算法的运行特性。不同算法以 3×10^4 和 2×10^4 函数评估后的收敛曲线图分别有图 2 和图 3。

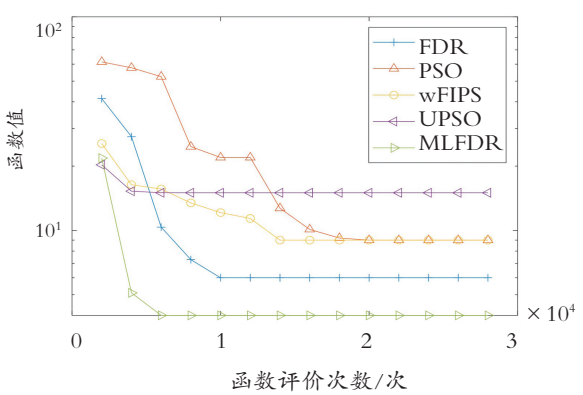
从图 2 可以看出,采用非旋转测试函数检测算法的情况,对于 Rastrigin’s 、Rosenbrock’s、Non−continuous Rotated Rostrigin’s 、Ackley’s、Happy Cat 和 HGBat 函数检测时,MLFDR 算法取得了最好的运行效果,且表现出更加明显的优势。而 FDR 和 wFIPS 算法在多峰问题上也表现出求解能力,主要是它们的学习策略不同于 PSO 和 UPSO 算法,可以有效地避免早熟收敛。从图 3 可以看出,采用旋转测试函数,对于 Rastrigin’s、Non−continuous Rotated Rostrigin’s、High Conditioned Elliptic、Rosenbrock’s、Discus 、Ackley’s、Happy Cat 和 HGBat 函数检测时,MLFDR 表现出较好的收敛性,主要是引入平均位置来构建广泛学习策略,于是提高了种群的多样性,增强了种群跳出局部最优解的能力。



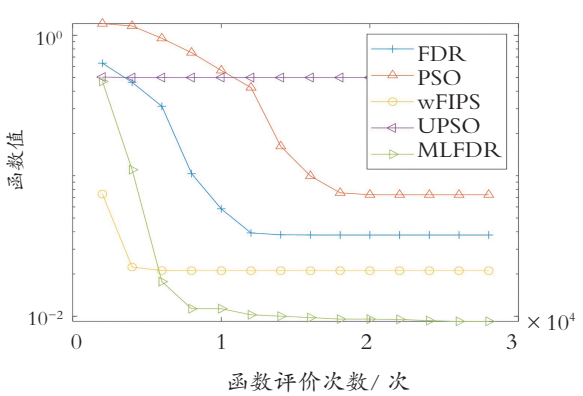
(a) Rastrigin's 函数



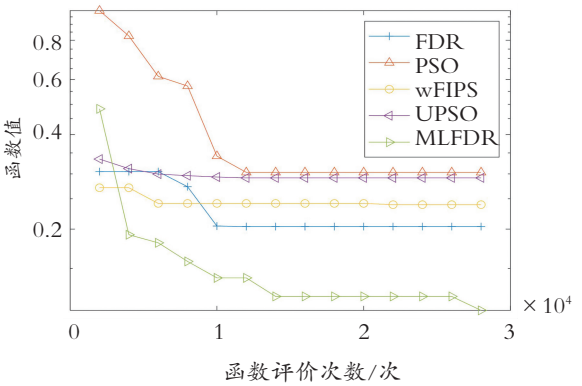
(b) Rosenbrock's 函数



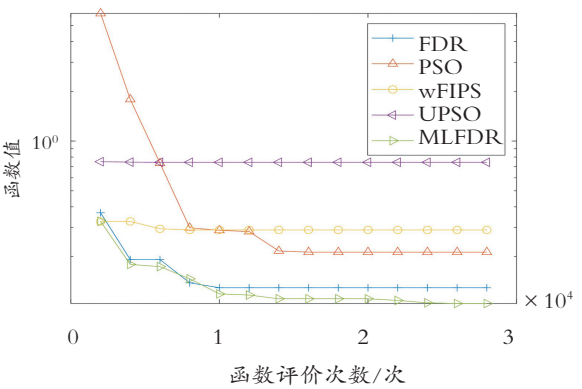
(c) Non-continuous Rotated Rostrigin's 函数



(d) Ackley's 函数



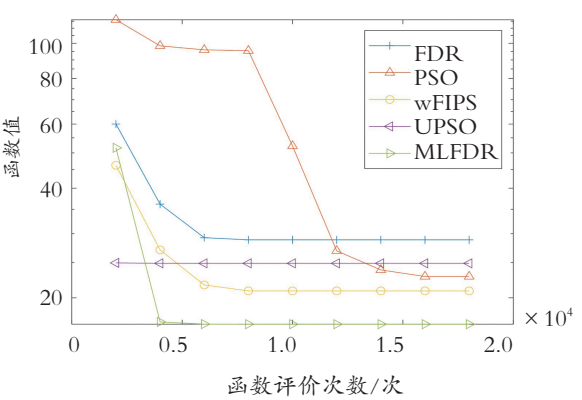
(e) Happy Cat 函数



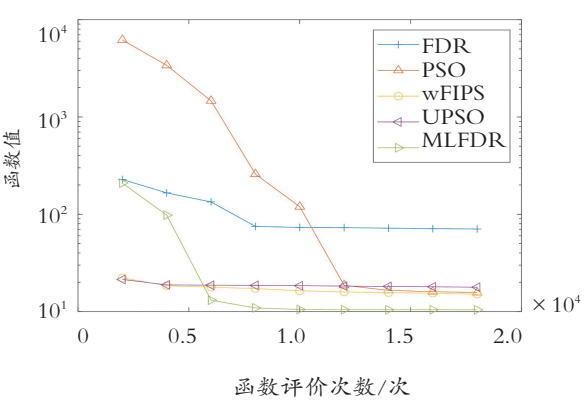
(f) HGBat 函数

图 2 CEC2017 非旋转测试函数收敛性图

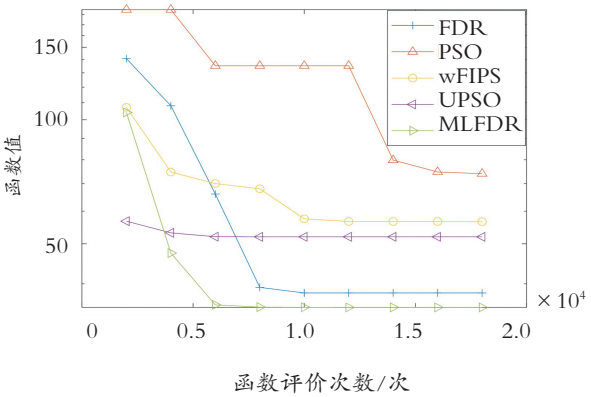
Fig. 2 CEC2017 non-rotating test function convergence graph



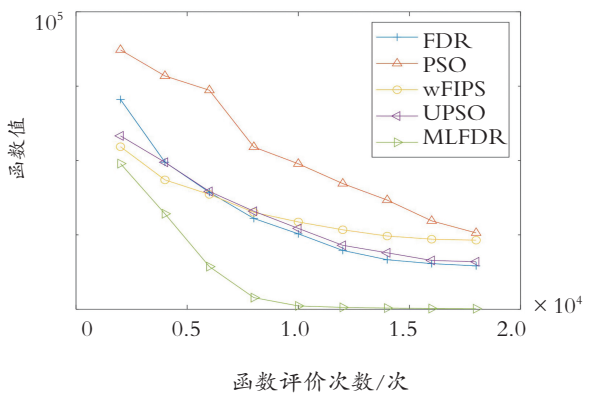
(a) Rastrigin's 函数



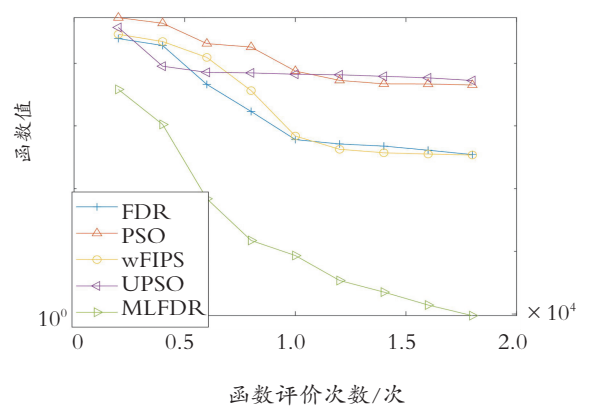
(b) Rosenbrock's 函数



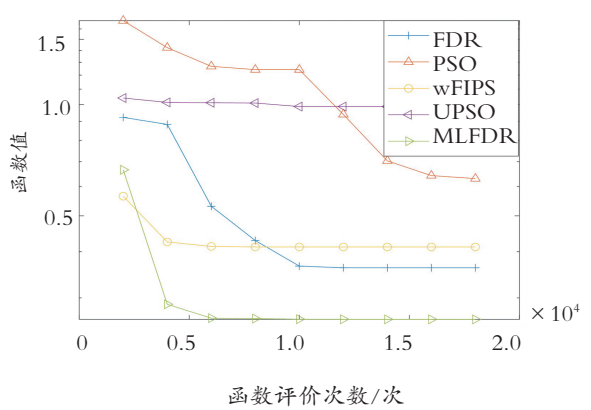
(c) Non-continuous Rotated Rostrigin's 函数



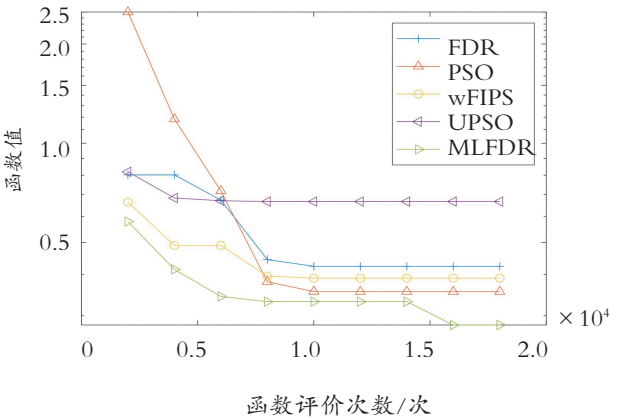
(d) High Conditioned Elliptic 函数



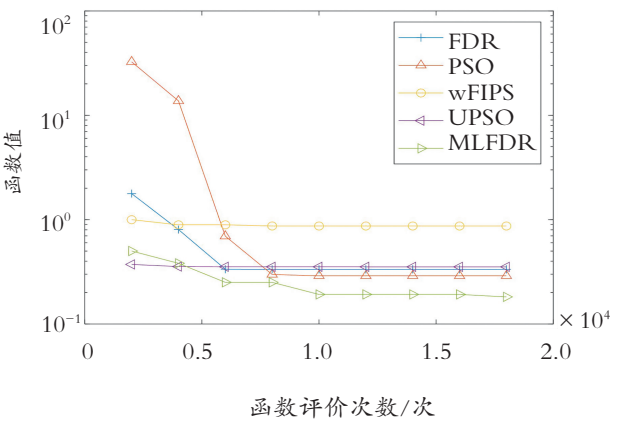
(e) Discus 函数



(f) Ackley's 函数



(g) Happy Cat 函数



(h) HGBat 函数

图 3　CEC2017 旋转测试函数收敛性图

Fig. 3　CEC2017 rotation test function convergence graph

3.3.2　鲁棒性分析

为了检测 5 个不同算法在相同测试函数的不同环境(函数旋转和不旋转)下算法运行是否具有稳定性,选择了 3 个 Benchmark 函数(Ackley、Happy Cat 和 HGBat),通过实验研究算法在这些测试函数的稳定性。鲁棒性分析结果在表 3 中给出,本文"鲁棒性"是在 50 次单独运行过程中算法成功取到指定阈值的比例,*FEs* 为取到阈值时的函数评价次数,*S* 为取到指导阈值的比例(单位是%)。其中 3 个函数的阈值分别是 0.4、0.3 和 0.5。

在表 3 可以看出,PSO、wFIPS、FDR 和 MLFDR 算法在 Ackley's 函数非旋转情况下 100%取到了指定阈值,但在旋转情况下只有 MLFDR 成功取得阈值。Happy Cat 函数在非旋转和旋转情况下都只有 MLFDR 算法成功取得阈值,但 FDR 在旋转和非旋转的情况下表现较为稳定,wFIPS 在旋转情况下也

表现较为稳定。HGBat 函数在非旋转和旋转情况下都仅有 PSO、FDR 和 MLFDR 算法成功取到了阈值，但 MLFDR 算法比 PSO 和 FDR 算法都用了较少的函数评价。

表 3　5 个算法的鲁棒性分析

Table 3　Robust analysis of five algorithms

算　法	f_6		f_7		f_8		f_6^*		f_7^*		f_8^*	
	FES	*S*/%	*FES*	*S*/%	*FES*	*S*/%	*FES*	*S*/%	*FES*	*S*/%	*FES*	*S*/%
PSO	11 590	100	17 612	76	7 370	100	11 966	80	11 866	78	5 484	100
FIPS	940	100	5 009	94	2 646	94	1 825	96	3 265	98	1 993	94
UPSO	1 130	90	8 065	84	8 362	76	12 024	44	6 698	76	4 039	82
FDR	5 940	100	6 213	98	1 745	100	5 297	94	5 154	98	1 626	100
MLFDR	879	100	4 197	100	950	100	2 187	100	3 128	100	954	100

3.3.3　测试函数寻优结果分析

为了排除偶然性引起的误差，分别对 6 个非旋转和 8 个旋转测试函数进行 50 次独立实验。在多个标准测试函数上，对 5 个不同算法进行 50 次独立运行后所得实验结果如下。

实验结果如表 4 和表 5 可知，实验在相同约束条件下，MLFDR 分别在非旋转测试函数和旋转测试函数的统计结果中明显优于另 4 个对比算法。对于测试函数 f_6，几种算法寻优效果差别不大，MLFDR 寻优结果略大于其他 4 个算法；对于 f_1、f_2、f_3、f_7、f_8 和旋转的所有测试函数，MLFDR 寻优效果都优于其他 4 个算法。不论是非旋转测试函数还是旋转测试函数，MLFDR 寻优结果优于其他 4 个算法，平均值和标准差都比其他 4 个算法小，说明 MLFDR 的稳定性和鲁棒性优于其他 4 个算法，证明改进后的粒子群算法具有一定优势。

表 4　非旋转测试函数寻优结果

Table 4　Non-rotation test function optimization result

统计量	算　法	f_1	f_2	f_3	f_6	f_7	f_8
最优值	PSO	1.99E+00	4.10E-02	1.00E+00	3.37E-03	4.03E-02	8.37E-02
	FDR	2.98E+00	3.11E-02	1.00E+00	8.40E-03	4.58E-02	4.77E-02
	wFIPS	1.99E+00	7.28E-02	2.00E+00	1.18E-02	6.64E-02	1.07E-01
	UPSO	2.98E+00	2.52E-02	1.00E+00	9.80E-03	7.18E-02	1.26E-01
	MLFDR	**9.95E-01**	**3.74E-05**	**4.13E-07**	**2.90E-03**	**3.76E-02**	**4.01E-02**
平均值	PSO	5.49E+00	2.84E+00	5.76E+00	2.21E-02	2.18E-01	2.26E-01
	FDR	7.70E+00	1.36E+00	7.24E+00	1.42E-01	1.55E-01	1.96E-01
	wFIPS	5.87E+00	1,45E+00	6.86E+00	7.56E-02	1.72E-01	3.72E-01
	UPSO	7.40E+00	2.64E+00	1.07E+01	2.35E-01	2.12E-01	3.71E-01
	MLFDR	**3.96E+00**	**2.58E+00**	**5.51E+00**	**2.98E-02**	**1.51E-01**	**1.66E-01**
标准差	PSO	1.56E+00	1.78E+00	3.21E+00	2.25E-02	1.03E-01	8.55E-02
	FDR	2.60E+00	2.17E+00	3.52E+00	8.38E-01	6.50E-02	6.89E-02
	wFIPS	2.26E+00	1.91E+00	4.34E+00	4.85E-02	7.60E-02	1.08E-01
	UPSO	2.61E+00	1.95E+00	4.96E+00	1.47E-01	8.90E-02	1.96E-01
	MLFDR	**1.36E+00**	**1.46E+00**	**2.39E+00**	**3.35E-02**	**6.01E-02**	**6.73E-02**

表 5　旋转测试函数寻优结果

Table 5　Rotation test function to find the optimal result

统计量	算　法	f_1^*	f_2^*	f_3^*	f_4^*	f_5^*	f_6^*	f_7^*	f_8^*
最优值	PSO	4.98E+00	6.20E−01	4.00E+00	2.20E−03	1.33E−02	4.12E−02	1.14E−01	1.01E−01
	FDR	2.99E+00	4.98E−02	3.00E+00	3.15E−02	2.32E−01	4.01E−02	6.16E−02	7.65E−02
	wFIPS	3.98E+00	2.87E−01	2.00E+00	3.60E−02	2.57E+01	2.23E−02	8.44E−02	1.34E−01
	UPSO	4.98E+00	1.79E−02	7.00E+00	1.80E−03	3.78E+00	1.04E−01	1.14E−01	1.04E−01
	MLFDR	**1.99E+00**	**5.30E−03**	**1.00E+00**	**4.26E−07**	**6.30E−03**	**1.93E−02**	**4.99E−02**	**4.43E−02**
平均值	PSO	1.20E+01	6.50E+00	1.44E+01	4.70E+01	4.02E+03	2.45E−01	2.48E−01	2.45E−01
	FDR	1.29E+01	2.73E+00	1.21E+01	1.03E+01	1.59E+03	2.15E−01	1.60E−01	2.21E−01
	wFIPS	1.07E+01	2.59E+00	8.19E+00	1.30E+01	4.68E+03	1.52E−01	1.73E−01	3.97E−01
	UPSO	1.31E+01	2.63E+00	1.59E+01	2.64E+01	7.78E+03	4.21E−01	2.48E−01	3.60E−01
	MLFDR	**8.36E+00**	**2.58E+00**	**7.31E+00**	**8.71E+00**	**1.54E+03**	**1.41E−01**	**1.48E−01**	**1.86E−01**
标准差	PSO	5.07E+00	1.55E+01	6.72E+00	6.58E+01	3.99E+03	1.49E−01	1.02E−01	7.57E−02
	FDR	5.69E+00	2.15E+00	5.17E+00	1.94E+01	2.05E+03	9.55E−02	6.32E−02	7.75E−02
	wFIPS	3.37E+00	1.82E+00	3.48E+00	1.44E+01	5.55E+03	1.14E−01	5.21E−02	1.25E−01
	UPSO	5.26E+00	2.09E+00	6.04E+00	4.22E+01	8.39E+03	1.95E−01	1.02E−01	1.54E−01
	MLFDR	**2.49E+00**	**1.69E+00**	**3.44E+00**	**1.19E+01**	**1.72E+03**	**6.94E−02**	**4.39E−02**	**6.53E−02**

3.3.4　Wilcoxon 秩和检验

由表 6 可知,为了进一步评估 MLFDR 的性能,在 a = 5% 显著性水平下,对 30 次独立运行下的 MLFDR 于其他 4 个算法的最佳结果进行 Wilconxon 秩和检验。以验证 MLFDR 所得结果与其他 4 个算法中最好结果的差别是否有统计意义。表 6 显示检验结果所示,小于等于 0.05 表示结果有差(符号“1”),大于 0.05 表示结果没有差异(符号“0”),在 8 个基准测试函数中,MLFDR 的性能在 8 个测试函数上与 PSO、FDR、wFIPS 和 UPSO 算法有差异,而且 MLFDR 的 p 值基本都小于等于 0.05,表明 MLFDR 的差异性在统计上显著的。

表 6 Wilcoxon 秩与检验 p 值

Table 6 Wilcoxon rank and test p values

函 数	MLFDR-PSO	MLFDR-wFIPS	MLFDR-FDR	MLFDR-UPSO
f_1	0.024 0	0.001 6	0.026 8	0.017 6
f_2	1.868 2E-05	2.831 4E-08	7.118 6E-09	3.368 1E-05
f_6	5.092 2E-08	8.870 1E-06	4.713 8E-04	7.380 3E-10
f_7	3.830 7E-05	0.018 4	0.015 0	1.584 6E-04
f_8	0.028 1	1.167 4E-05	0.007 0	0.001 0
f_3^*	0.031 4	8.510 0E-04	0.002 7	4.214 4E-04
f_4^*	2.388 5E-04	0.011 7	0.027 1	0.048 4
f_5^*	0.003 3	0.002 9	0.016 3	1.040 7E-04
1/0	8/0	8/0	8/0	8/0

4 结 论

本文为解决 PSO 存在寻优精度不高、易陷入局部最优解等不足,提出了一种基于平均位置学习的改进粒子群算法 MLFDR。该算法通过引入比粒子自身适应值更好的邻近粒子和平均位置作为学习对象,改善了算法的早熟收敛问题,提高了收敛速度。其主要研究粒子群算法在优化问题时综合所有粒子平均位置的有益信息,提高了算法对函数的优化,特别是多峰函数优化中的性能。通过用 8 个测试函数进行实验,与其他 4 个算法的比较得出 MLFDR 综合性能更好。

参考文献(References):

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

[2] CUI Z H, ZHANG J J, WU D, et al. Hybrid many-objective particle swarm optimization algorithm for green coal production problem[J]. Information Sciences, 2020 (518): 256—27.

[3] ZHAO F, QIN S, YANG G, et al. A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective[J]. Expert Systems with Application, 2019(126): 41—53.

[4] WU Y, WU Y, LIU X. Couple-based particle swarm optimization for short-term hydrothermal scheduling[J]. Applied Soft Computing, 2019(74): 440—450.

[5] KENNEDY J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance [C] //Congress on Evolutionary Computation. IEEE, 1999: 1931—1938.

[6] KENNEDY J, RUI M. Population structure and particle swarm performance[C] // Proceedings of the 2002 Congress on Evolutionary Computation. 2002: 1671—1676.

[7] PARSOPOULOS K E, VRAHATIS M N. A Unified Particle Swarm Optimization Scheme[J]. Lecture Series on Computational Sciences, 2004: 868—873.

[8] PERAM T, VEERAMACHANENI K, MOHAN C K. Fitness-distance-ratio based particle swarm optimization [C]//Proceedings of the Swarm Intelligence Symposium Indianapolis USA, 2003: 174—181.

[9] LYNN N, SUGANTHAN P N. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation[J]. Swarm and Evolutionary Computation, 2015(24): 11—24.

[10] HAYASHIDA T, NISHIZAKI L SEKIZAKI S, et al. Improvement of two-swarm cooperative particle dwarm optimization using immune slgorithms and swarm clustering [C]//IEEE 11th International Workshop on Computational Intelligence and Applications, 2019: 101—107.

[11] ZHANG X, SUN W, XUE M, et al. Probability-optimal

leader comprehensive learning particle swarm optimization with bayesian iteration [J]. Applied Soft Computing Journal, 2021,103(2): 107—132.

[12] CHENG R, JIN Y. A social learning particle swarm optimization algorithm for scalable optimization [J]. Information Sciences, 2015(291): 43—60.

[13] LIM H W, ISA N A M. Bidirectional teaching and peer-learning particle swarm optimization [J]. Information Sciences, 2014, 280(4): 111—134.

[14] MENDES R, KENNEDY J, NEVES J. The fully informed particle swarm: simpler, maybe better [J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 204—210.

Research on Improved Particle Swarm Algorithm Based on Average Position Learning

YANG Mei-lan[1], LIU Yan-min[2], ZHANG Qian[1], SHU Xiao-li[3]

(1. School of Mathematics and Statistics, Guizhou University, Guiyang 550025, China;
2. School of Mathematics, Zunyi Normal College, Guizhou Zunyi 563006, China;
3. School of Data Science and Information Engineering, Guizhou Minzu University, Guiyang 550025, China)

Abstract: Particle Swarm Optimization (PSO) has some shortcomings in solving complex multi-dimensional and multi-peak problems, such as low local search accuracy and easy to fall into local optimum. Therefore, this paper presents an improved particle swarm optimization algorithm based on average position learning. In this algorithm, neighboring particles with better adaptive values than the particles themselves are adopted as learning objects in the learning strategy, and the algorithm is divided into two stages with different updating speed formulas. In stage one, the average position of all particle positions in the whole population is introduced into the updating velocity formula. In the second stage, a new average position is introduced into the velocity updating formula, and the greedy strategy is adopted to select the individuals selected after each updating of the particles to be better than the historical optimal adaptive value of the population. In addition, the historical optimal positions of the corresponding individuals are stored, and their average positions are calculated after the end of the first stage. Taking the average position as the learning object can enhance the information exchange among particles, and balance the local development performance and global search ability of the algorithm. In the CEC2017 test function experiment, the experimental results show that the proposed algorithm has certain advantages compared with the other four algorithms.

Key words: particle swarm optimization; mean position; information exchange

责任编辑:罗姗姗

引用本文/Cite this paper:

杨妹兰,刘衍民,张倩,等. 基于平均位置学习的改进粒子群算法研究[J]. 重庆工商大学学报(自然科学版),2022,39(3):9—19.

YANG Mei-lan, LIU Yan-min, ZHANG Qian, et al. Research on improved particle swarm algorithm based on average position learning[J]. Journal of Chongqing Technology and Business University (Natural Science Edition),2022,39(3):9—19.