学习因子随权重调整的混合粒子群算法

曹晓月,张旭秀
(大连交通大学 电气信息工程学院,辽宁 大连 116021)

摘 要:首先使惯性权重随迭代次数和粒子状态非线性改变平衡算法的全局探测和局部开采的能力,为了解决惯性权重与学习因子独立调整削弱了粒子群算法的统一性和智能性等问题,通过分析惯性权重与学习因子的变化关系,将学习因子表示为惯性权重的logistic回归分析型函数。由于非线性因子的加入会降低粒子的多样性,结合差分进化算法的交叉算子和变异策略,利用交叉算子来提高算法的全局探索能力,保持种群多样性;利用差分进化算法的变异策略产生候选解来更新位置公式,给出了学习因子随权重调整的混合粒子群算法,并对新提出算法的收敛性进行理论分析。将此改进算法与相关算法在四个测试函数上进行对比实验,证明该算法在寻优精度、迭代速度和收敛成功率上有明显改进。

关键词:粒子群算法;学习因子;惯性权重;混合算法;收敛性分析
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2020)11-0030-07
doi:10.3969/j.issn.1673-629X.2020.11.006

Research of Simplified Particle Swarm Optimization Algorithm with Weight and Learning Factor

CAO Xiao-yue, ZHANG Xu-xiu
(School of Electrical Information Engineering, Dalian Jiaotong University, Dalian 116021, China)

Abstract: Firstly, the inertia weight changes the global detection and local mining capability of the equilibrium algorithm nonlinearly with iteration times and particle state. In order to solve the problem that the independent adjustment of inertia weight and learning factor weakens the unity and intelligence of particle swarm optimization (PSO) algorithm, the learning factor is expressed as a logistic regression analysis function of inertia weight by analyzing the relationship between inertia weight and learning factor. Since the addition of nonlinear factors will reduce the diversity of particles, combining the crossover operator and mutation strategy of the differential evolution algorithm, the crossover operator is used to improve the global exploration ability of the algorithm, which keeps the diversity of the population. By the variation strategy of the differential evolutionary algorithm, the candidate solutions can be generated to update the position formula. A hybrid particle swarm optimization algorithm with learning factors adjusted with weights is proposed and its convergence is analyzed theoretically. Finally, the improved algorithm is compared with the existing algorithm on four test functions, and it is proved that it has obvious improvement on the optimization accuracy, iteration speed and convergence success rate.

Key words: particle swarm optimization; learning factor; inertia weight; hybrid algorithm; convergence analysis

0 引 言

粒子群算法(particle swarm optimization, PSO)是Kennedy和Eberhart教授于1995年联合提出的一种基于群体智能的随机进化优化算法[1]。PSO算法因其需要调整的参数较少、收敛速度快、精度高等优点被快速接受并运用在电磁优化[2-3]、参数优化[4-5]和电力系统[6]等领域。对粒子群算法的改进也层出不穷。其中对惯性权重和学习因子的改进最为简单快捷。Shi等人引入随进化次数而线性递减的惯性权重w[7],这使得PSO算法的运行速度和搜索能力得到了大幅度提升。文献[8]中通过随机分布的方式获取惯性权重,文献[9-10]将sigmoid函数引入到惯性权重中构造动态调整因子,这些都提高了算法的全局和局部的搜索能力。有的学者将学习因子进行改进[11-13]。

但是惯性权重和学习因子等参数的独立调整削弱了算法的智能性。该文结合对上述文献中对粒子群算法改进的分析,使学习因子随惯性权重非线性变化,并且加入差分进化算法中的交叉算子来增加种群多样

收稿日期:2019-11-22 **修回日期**:2020-03-25
基金项目:国家科技支撑计划资助项目(2015BAF20B02);国家自然科学基金(61471080, 61201419);辽宁省自然基金指导计划(1553737612631);国家留学基金委资助计划(201608210308)
作者简介:曹晓月(1994-),女,硕士,研究方向为智能控制理论与应用;张旭秀,博士,教授,研究方向为智能控制理论与应用、电信工业。

性,并对算法进行复杂性分析。最后同相关算法进行对比,证明算法的有效性。

1　基本粒子群算法

PSO 算法首先初始化一群粒子 $X_i = (x_{i1}, x_{i2}, \cdots, x_{in})$,在每一次迭代过程中,粒子通过个体极值 pbest 和全局极值 gbest 更新自身的速度和位置,更新公式如下:

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * \mathrm{rand}_1 * (\mathrm{pbest}_{ij}^{t} - x_{ij}^{t}) + c_2 * \mathrm{rand}_2 * (\mathrm{gbest}_{i}^{t} - x_{ij}^{t}) \quad (1)$$

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

其中,w 为惯性权重,表示粒子继承了当前速度的程度,w 越大,全局的搜索能力越强,w 越小,局部的搜索能力越强;c_1,c_2 为速度因子,其中 c_1 是“自我认知”,表示粒子向自我的学习能力强弱,其中 c_2 是“社会认知”,表示粒子向社会的学习能力强弱;rand_1 和 rand_2 是分布在[0,1]之间的随机数;t 为当前迭代次数;v_{id} 为粒子的运行速度。粒子群算法存在的主要问题是随着迭代次数的增加,搜索速度变慢和易陷入局部寻优。而 w 则是控制速度和全局搜索的重要因素,学习因子反映了微粒之间的交流情况,平衡了算法的全局探测和局部开采的能力。因此,对惯性权重和学习因子的改进使具有重要意义。

2　改进的粒子群算法

2.1　状态因子的提出

粒子在每次迭代都是将当前值与最优值比较,更新出全局最优值。但并不是所有粒子都具有这种趋向最优的特性。在搜索过程中,总会有一些粒子会离最优值较远或者集中于一个“自认为”最优的区域进行搜索。这时,就需要统筹兼顾所有粒子,用类似数学上的方差来表示粒子的当前质量。则提出粒子状态因子如下:

$$\sigma^2 = \sum_{i=1}^{n} (f_i \bullet f_{\mathrm{avg}}^{-1} - f_{\mathrm{best}} \bullet f_{\mathrm{avg}}^{-1}) \quad (3)$$

其中,f_i 为粒子的当前适应度值,f_{avg} 为粒子的平均适应度值,其数学公式定义为 $f_{\mathrm{avg}} = n^{-1} \sum_{i=1}^{n} f_i$,$f_{\mathrm{best}}$ 为粒子的最优适应度值,即全局最优值 。

全局最优值的取值决定于个体极值的变化,全局和个体的关系也反映了粒子的当前状态。在求极小值问题时,个体极值要小于全局最优值,当然也小于平均值。平均值反映了所有粒子的平均质量。

2.2　惯性权重的改进

Shi 等人[7]提出线性递减 w 的方法,实验表明虽然 LDW 在函数优化速度和精度方面有了提升,但是随迭代次数改变的惯性权重的方法对于最优解附近才有效。如果搜索初期没有找到最优解,随着搜索次数的增加,惯性权重逐渐减小,局部的搜索能力增强,粒子就会跳过最优解而陷入局部最优。该文提出了一种新的惯性权重更新公式,使惯性权重随粒子的迭代次数和搜索状态进行更新。公式如下:

$$w = w_{\max}(1 - \sqrt{t/t_{\max}}) \bullet \sigma^2/\sigma_{\max}^2 \quad (4)$$

其中,$\sigma_{\max}^2 = \max_{1 \leqslant i \leqslant n} \{\sigma^2\}$,$t$ 是当前进化次数,$t_{\max}$ 是最大进化次数,$w_{\max}$ 为最大惯性权重。

由式可知,$0 \leqslant \sigma^2/\sigma_{\max}^2 \leqslant 1$。$\sigma^2/\sigma_{\max}^2$ 在一定程度上反映了粒子的聚合情况,进而也反映了粒子的多样性状况。$\sigma^2/\sigma_{\max}^2$ 越大,表示粒子离最优值越远,比较分散,反之 $\sigma^2/\sigma_{\max}^2$ 越小,表示粒子的越集中。

在此惯性权重更新公式中,w 由迭代次数和粒子搜索状态共同决定。随着搜索次数的增加,w 随 $\sqrt{t/t_{\max}}$ 呈非线性递减变化;但随着搜索阶段的进行,粒子会出现部分分散,此时需要一个较大的惯性权重控制全局的快速搜索,此时状态因子起了重要的决定作用,它统筹了粒子的全部质量,对于分散的粒子有了一定的“拉回”作用;在后期粒子出现的部分集中现象,状态因子又起到了一定的“打散”作用。有了迭代次数和状态因子的双重保障,使惯性权重更加实时、动态的更新,更好地适应复杂的搜索过程。

2.3　学习因子的改进

粒子在搜索过程中通过跟踪个体最优值和全局最优值进行更新,所以加强粒子个体之间和全局之间的交流对提高算法速度有着重要意义。通过借鉴对学习因子已有的研究发现,w 和 c_1,c_2 的关系变化如图 1 和图 2 所示。

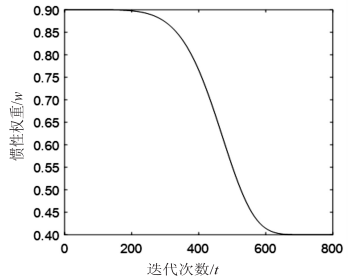

图 1　$w - t$ 变化曲线

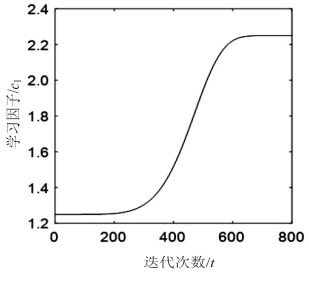

图 2　$c_1 - t$ 变化曲线

由图分析可知，c_1 与 w 的关系类似于 logistic 回归分析，图形是一个 s 型变化曲线，二者呈正相关变化。并且满足：

$$\begin{cases} w(t \to 0) = 0.9 \\ w(t \to \infty) = 0.4 \end{cases}$$
$$\Downarrow$$
$$\begin{cases} c_1(w \to 0.9) = 2.5 \\ c(w \to 0.4) = 1.25 \end{cases}$$

综上，将 w 表示为 c_1 的函数，并且满足 $c_1 + c_2 = 2.5$。c_1 和 c_2 的更新规则如下：

$$\begin{cases} c_1 = \dfrac{2.5}{1 + e^{-w}} \\ c_2 = 2.5 - c_1 \end{cases} \tag{5}$$

其中，c_1 采用 logistic 回归曲线方程，这个函数满足了中期随 w 快速变化，加快算法的速度；后期缓慢变化，配合 w 进行精细搜索，平衡算法的全局和局部的搜索能力。$c_1 + c_2 = 2.5$，是此消彼长设置。前期 c_1 较大而 c_2 较小，此时粒子的自我学习能力较强而社会学习能力较弱，有利于加强算法的全局搜索；后期 c_1 较小而 c_2 较大，此时粒子的社会学习能力较强而自我学习能力较弱，有利于算法局部的精细搜索。

3 种群多样性的保持

粒子群算法的缺陷之一是随着搜索的进行多样性会降低，并且随着上述提出的非线性因子的加入，粒子多样性缺失会更加严重，导致陷入局部寻优。为了解决这一现象，加入差分进化算法中的交叉算子来提高算法的全局探索能力，保持种群多样性，利用 DE 算法的变异策略产生候选解来更新位置公式。

3.1 变异策略的加入

差分进化算法是基于种群的随机优化算法[14-15]，此算法在同一种群中将父代个体和其他个体之间通过交叉得到差异信息，从而获得候选解。

设当前进化代数为 t。当前种群是规模为 NP 的 D 维向量 $X(t) = \{x_1^t, x_2^t, \cdots, x_{NP}^t\}$，其中，$x_i^t = (x_{i1}^t, x_{i2}^t, \cdots, x_{iD}^t)^T$ 为种群中的第 i 个个体。

变异操作的主要目的是生成变异个体。该文采用策略一来产生变异个体。变异策略如下：

$$v_{ij}^t = x_{r_1 j}^t + F(x_{r_2 j}^t - x_{r_3 j}^t), j = 1, 2, \cdots, D$$

其中，$x_{r_1}^t = (x_{r_1 1}^t, x_{r_1 2}^t, \cdots, x_{r_1 D}^t)^T$，$x_{r_2}^t = (x_{r_2 1}^t, x_{r_2 2}^t, \cdots, x_{r_2 D}^t)^T$，$x_{r_3}^t = (x_{r_3 1}^t, x_{r_3 2}^t, \cdots, x_{r_3 D}^t)^T$ 是群体中的三个随机个体，并且 $r_1 \neq r_2 \neq r_3 \neq i$，$x_{r_1 j}^t$，$x_{r_2 j}^t$，$x_{r_3 j}^t$ 分别为个体 r_1，r_2 和 r_3 的第 j 列分量。F 为变异因子，控制差分进化搜索的步长，其值在 [0,1]，这里取 0.5。F 对控制种群多样性具有重要作用。当 F 较大时，它可以保证种群的多样性；当 F 较小时，局部开采能力曾强，加快算法收敛。

3.2 交叉算子的加入

在变异策略完成产生变异个体之后，再将变异个体与目标个体通过交叉操作产生试验个体 $u_i^t = (u_{i1}^t, u_{i2}^t, \cdots, u_{iD}^t)^T$。二项式交叉操作如下：

$$u_{ij}^t = \begin{cases} v_{ij}^t & \text{if rand}[0,1] \leqslant \text{CR or } j = j\_\text{rand} \\ x_{ij}^t & \text{if rand}[0,1] > \text{CR and } j \neq j\_\text{rand} \end{cases}$$

其中，rand 是 [0,1] 间的随机数；CR 是范围在 [0,1] 间的常数，称为交叉因子，CR 值越大，发生交叉的可能性就越大，就越能保持种群的多样性，而 CR 较小的话可以保证算法快速收敛；$j\_$ rand 是在 $[1, D]$ 随机选择的一个整数，它保证了对于试验个体 u_i^t 至少要从变异个体 v_i^t 中获得一个元素，保证了试验个体不同于目标个体。图 3 给出了 5 维向量交叉原理示意图。

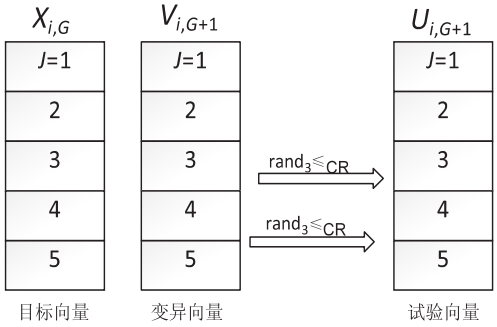

图 3 5 维向量交叉原理示意图

综上提出新的位置更新公式：

$$x_{i,j} = \begin{cases} x_{r_1,j} + F(x_{r_2,j} - x_{r_3,j}), \text{rand} < \text{CR} \\ x_{r_1,j}, \text{rand} \geqslant \text{CR} \end{cases} \tag{6}$$

$r_1, r_2, r_3 \in \{1, 2, \cdots, N\}$ 是随机的三个个体且互不相同。这里缩放因子 F 取 0.5。CR 为交叉算子，这里取 0.1。当 rand 产生的随机数小于 CR 时，采用差分交叉算子来更新位置公式；当 rand 产生的随机数大于 CR 时，采用经典粒子群算法的更新位置公式。

3.3 算法流程

学习因子随权重调整的混合粒子群算法的流程如下：

Step1：初始化粒子，令其值约束在规定的范围内，并令每个粒子都具有位置向量 X_i，速度向量 V_i；

Step2：计算粒子的适应度值，将 f_i 设置为粒子当前位置的适应度，计算出 f_{avg} 为平均适应度值；

Step3：将粒子的适应度值与自身经过的粒子的所有位置进行比较，将较好的一个作为当前的最好位置，记为 pbest；将粒子的适应度值与所有粒子经过的位置进行比较，将较好的一个作为全局的最好位置，记为 gbest；

Step4：如果 rand < 变异率，采用差分交叉算子更新粒子位置，否则根据标准粒子群算法来调整微粒速

度和位置;

Step5:对所有粒子相继按式(6)更新位置公式;按式(4)、式(5)计算惯性权重 w , c ;计算适应度值;

Step6:如果算法达到最大迭代次数,执行步骤 8,否则执行步骤 3;

Step7:将迭代次数加一,执行步骤 3;

Step8:达到最大迭代次数,输出 gbest 。

4　收敛性分析

4.1　收敛准则

在分析 DSPSO 算法时,需要用到一些假设和定理,下面简要给出它们的内容。

定义:给出一个函数 f ,其解空间为 $D^n \sim D$, S 为 D^n 的一个子集;在 S 中存在一个点 z ,它能够使函数 f 的值最小或能够使函数 f 的值在 S 上能够有较小的下确值。

假设 1:$f(H(z,\delta)) \leqslant f(z)$,若 $\delta \in s$,则 $f(H(z,\delta)) \leqslant f(\delta)$ 。H 是在待求解空间产生的一个解的函数,并且应保证 H 所产生的所有新个体优于当前个体。

4.2　DSPSO 算法的全局收敛证明

引理 1:DSPSO 算法满足假设 1。

证明 由式(1)、式(6)可知,DSPSO 算法的可描述为:

$$H(\text{gbest},X_i)=\begin{cases}\text{gbest}, f(\text{gbest}) \leqslant f(X_i)\\ X_i, f(X_i) \leqslant f(\text{gbest})\end{cases}$$

所以假设 1 很容易证明。

引理 2:收敛性证明。

证明:不考虑式(1)中的随机分量 rand($i=1,2$),并假设 p_d 为一常量且定义如下:

$$p_d=\frac{c_1 * p_i + c_2 * p_g}{c_1 + c_2}$$

将上式带入到式(1)中

$$v_{t+1}=w * v_t + c(p_d - x_t)$$

其中, $c=c_1+c_2$ 。

设 $y_t=p_d-x_t$,则速度和位置更新公式可简写为:

$$v_{t+1}=\omega * v_t + c * y_t$$

$$y_{t+1}=-\omega * v_t + (1-c) * y_t$$

设 $\boldsymbol{P}_t=\begin{bmatrix}v_t\\ y_t\end{bmatrix}$, $\boldsymbol{M}=\begin{bmatrix}\omega & c\\ -\omega & 1-c\end{bmatrix}$,则 DSPSO 算法可以用矩阵方程 $\boldsymbol{P}_t=\boldsymbol{M}^t\boldsymbol{P}_0$ 表示且 $\boldsymbol{M}$ 的特征值经计算为:

$$\begin{cases}\lambda_1=\dfrac{\omega+1-c+\sqrt{(\omega+1-c)^2-4\omega}}{2}\\ \lambda_2=\dfrac{\omega+1-c-\sqrt{(\omega+1-c)^2-4\omega}}{2}\end{cases}$$

当 $(\omega+1-c)^2 \neq 4\omega$ 时,可以定义矩阵 $\boldsymbol{N}$ 满足下式:

$$\boldsymbol{NMN}^{-1}=\boldsymbol{L}=\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}$$

其中,

$$\boldsymbol{N}=\begin{bmatrix}2\omega & \omega-1+c-\sqrt{(\omega+1-c)^2-4\omega}\\ 2\omega & \omega-1+c+\sqrt{(\omega+1-c)^2-4\omega}\end{bmatrix}。$$

假设 $\boldsymbol{Q}_t=\boldsymbol{NP}_t$,则 $\boldsymbol{Q}_t=\boldsymbol{L}^t\boldsymbol{Q}_0$ 。对于 $\boldsymbol{L}$ 为对角矩阵,可得下式:

$$\boldsymbol{L}'=\begin{bmatrix}\lambda_1' & 0\\ 0 & \lambda_2'\end{bmatrix} \tag{7}$$

由动力系统的稳定性理论,可得如下定理:

定理:要使 DSPSO 算法模型稳定,即收敛于 p_d ,当且仅当 $\max\{|\lambda_1|,|\lambda_2|\}<1$ 。

推论 1:当 $(\omega+1-c)^2 \neq 4\omega$ 且 $0<\omega<1$ 时要使 DSPSO 算法模型稳定,即收敛于 p_g ,当且仅当 $0<c<2(\omega+1)$ 。

推论 2:当 $(\omega+1-c)^2=4\omega$,且 $0<\omega<1$ 时要使 DSPSO 算法模型稳定,即收敛于 p_g ,当且仅当 $\omega-1<c<w+3$ 。

推论 3:当 $0<\omega<1$ 时要使 IDWPSO 算法模型稳定,即收敛于 p_g ,当且仅当 $0<c\leqslant 2(\omega+1)$ 。

综上,学习因子和惯性权重之间的收敛关系如图 4 所示。

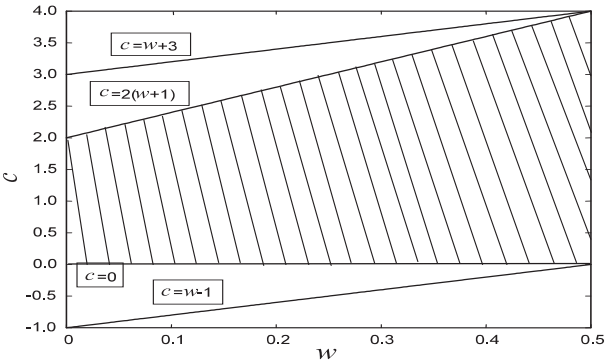

图 4　c 与 w 的收敛关系

DSPSO 算法中学习因子表示为惯性权重的函数,既学习因子随惯性权重而决定。在实际算法中,学习因子往往乘以 [0,1] 随机数 $\text{rand}_i(i=1,2)$,所以只需要确定算法收敛模型学习因子的上限值,而乘以随机数后必然收敛于此条件,在此算法中惯性权重取值在 [0.4,0.9] ,满足上述条件。

5　仿真测试与分析

5.1　实验设计

为了验证改进算法的有效性,用 SPSO 和 LPSO 与提出的 DSPSO 进行对比。用四个测试函数进行测

试。在收敛精度、收敛速度和维度方面进行对比。

在实验中,种群规模为 30,迭代次数为 1 000,LPSO 惯性权重从 0.9 线性递减到 0.4;DSPSO 的参数设置:惯性权重 $w_{\max}=0.9$, $w_{\min}=0.4$。

实验将从以下三方面进行测试分析:

(1)对四个函数在三种算法下的 3 维、10 维和 30 维运行 50 次求平均值,对其各维度收敛精度进行分析;

(2)通过多次实验的平均适应度进化曲线来比较四种算法的收敛速度;

(3)固定收敛精度,规定迭代次数,比较四个函数在三种算法下的成功率。

测试函数如表 1 所示。

表 1 测试函数信息

函数名	函数表达式	解空间	最优值位置	最优值
Schwefel2.22	$f_1(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]^D$	$(0)^D$	0
Ackley	$f_2=-20\exp\left(-20\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32,32]^D$	$(0)^D$	0
Griewank	$f_3(x)=\frac{1}{400}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^D$	$(0)^D$	0
Penalized1.1	$f_4(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	$[-50,50]^D$	$(0)^D$	0

5.2 测试结果分析

5.2.1 算法收敛精度比较

对四个函数分别在 3 维、10 维和 30 维进行 50 次运行取平均值,最优值和平均值如表 2 所示。

表 2 函数测试结果

函数	维数	理论值		LPSO	SPSO	DSPSO
f_1	3	0	a	1.189 1e-69	3.038 4e-40	1.867e-217
			b	3.586 27e-67	4.429 2e-32	1.424e-209
	10	0	a	0.033 706	3.098 8e-17	1.337 6e-26
			b	0.095 693 2	2.470 8e-15	2.916 7e-26
	30	0	a	4.574 5	2.458 5e-05	4.367 5e-07
			b	2.435 554	0.602 849 5	4.657 2e-06
f_2	3	0	a	8.881 8e-16	8.881 8e-16	8.881 6e-16
			b	5.856 5e-06	2.928 2e-06	8.881 8e-16
	10	0	a	8.881 8e-16	8.881 8e-16	8.881 8e-16
			b	0.000 294 1	0.000 147 5	8.881 8e-16
	30	0	a	8.881 8e-16	8.881 8e-16	8.881 8e-16
			b	5.735 5e-06	2.867 7e-06	8.881 8e-16
f_3	3	0	a	0	0	0
			b	0	0	0
	10	0	a	0.543 6	0.358 57	0.024 737
			b	1.617 722	0.639 57	0.085 173
	30	0	a	1.384	0.643 22	1.906 9e-07
			b	5.319 14	0.825 33	0.019 103
f_4	3	0	a	2.335 8e-31	2.335 8e-31	3.412 6e-52
			b	2.335 8e-31	3.072 8e-30	2.335 8e-46
	10	0	a	5.195 7e-32	6.099 4e-28	4.711 6e-32
			b	6.712 2e-09	0.124 42	4.755 1e-32
	30	0	a	7.399 2e-09	0.735 19	1.186 9e-08
			b	0.020 73	2.680 078	4.793 0e-06

a 是平均值,b 是最优值

由表 2 可知,宏观来看,对于函数 f_1, f_3, f_4,DSPSO 不管在 3 维、10 维或是 30 维,平均值都优与 SPSO 和 LPSO。对于函数 f_2,三种算法的最优值都一样,但平均值却不同,SPSO 和 LPSO 在 3 维和 30 维的精度在 10^{-6}左右,而 DSPSO 不管在 3 维、10 维还是 30 维,其最优值和平均值都是 8.881 8E-16,这也就说明了 DSPSO 对处理 f_2 这种用于许多局部最优值并且自变量之间相互独立的函数具有很好的稳定性。

随着 4 个函数的维数增加,函数变得逐渐复杂。SPSO 和 LPSO 的搜索精度渐渐变低,说明 SPSO 和 LPSO 对于处理高维度的函数的搜索能力逐渐减弱;而对于 DSPSO,虽然随着四个函数的维度增加,精度在逐渐减低,但对于 f_1,DSPSO 的精度逐渐由 10^{-29}到 10^{-26}最后到 10^{-6};在函数 f_2 中,DSPSO 的值一直是 8.881 8e-16,体现了 DSPSO 的稳定性;在函数 f_3 中,DSPSO 的算法平均值由 0 到 0.085 173 再到 0.019 103,由此表明,虽然在 3 维时搜索结果最优,但

随着维数增加,30 维的结果优于 10 维,但二者精度是一样的;在函数 f_4 中,DSPSO 的精度由 10^{-46} 到 10^{-32} 再到 10^{-6},但都是远远优于同维度的其他两种算法。综上表明 DSPSO 算法对于搜索精度和解决高维度的函数问题有很好的效果。

5.2.2　算法进化速度比较

四个函数的适应度进化曲线如图 5 和图 6 所示。

在图 5 左中,由于 DSPSO 算法在 700 次左右完成迭代,而 SPSO 和 LPSO 没有完成迭代,所以附上程序运行后的迭代最优值,如图 7 所示。

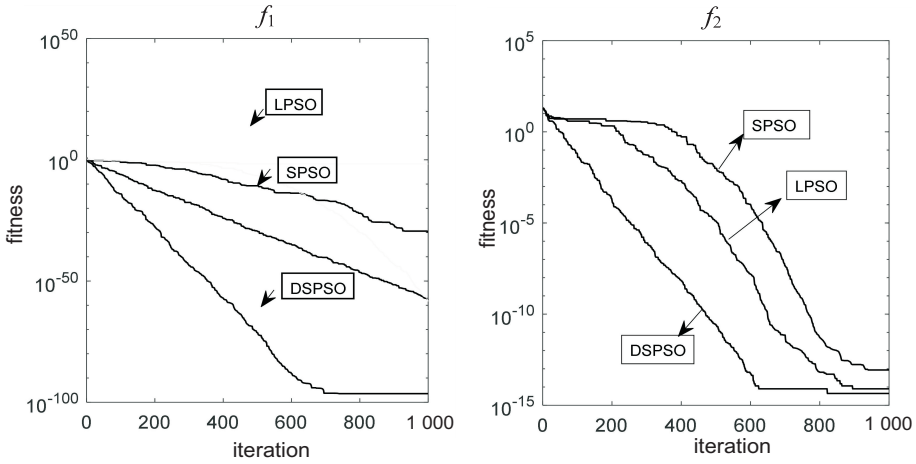

图 5　函数 f_1、f_2 适应度进化曲线

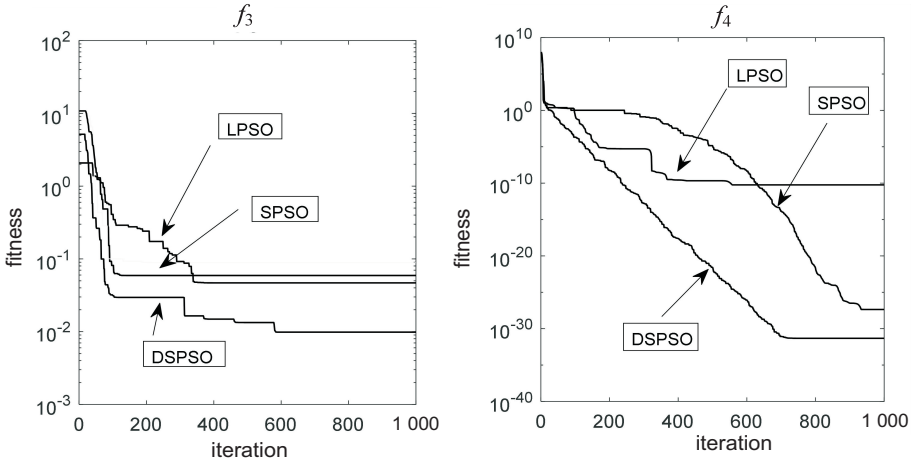

图 6　函数 f_3、f_4 适应度进化曲线

```
found by DSPSO is : 4.4033e-81
found by SPSO is : 4.0006e-34
found by LPSO is : 6.4185e-58
```

图 7　函数 f_1 的最优值

由图 5 左和图 7 可知,DSPSO 在 600 完成迭代,而且精度是 10^{-100},远远高于 SPSO 和 LPSO 的精度;SPSO 和 LPSO 在 1 000 次内并没有完成迭代,说明 DSPSO 的搜索速度远高于其他两种算法;在图 5 右中,虽然三种算法最后搜索精度和迭代次数相近,但可以发现,DSPSO 还是比较优中取精的,不管是速度还是精度都要优于其他两种算法;在图 6 左中,SPSO 和 LPSO 早早陷入局部寻优,DSPSO 在 300 次时陷入局部最优,但是随后跳出局部最优,在 600 次左右完成迭代,增加了搜索的精度;在图 6 右中,LPSO 早早陷入局部寻优,DSPSO 的最后收敛精度优于 LPSO10^{-3}个精度,并且 DSPSO 的搜索速度快于 SPSO,并且 DSPSO 在 700 次迭代完成,而 SPSO 要 900 次迭代完成。

5.2.3　算法成功率比较

对每个函数进行 50 次仿真,其中对 f_1、f_2、f_3、f_4 设置 1 000 次迭代次数。当每个函数达到设定迭代次数时仍未收敛的则认定收敛失败。收敛成功率分析如表 3 所示。

表 3　函数成功率

函数	算法	成功收敛次数	成功率/%	函数	算法	成功收敛次数	成功率/%
	SPSO	25	50		SPSO	22	44
f_1	LPSO	33	66	f_3	LPSO	33	66
	DSPSO	30	60		DSPSO	47	94
	SPSO	7	14		SPSO	9	18
f_2	LPSO	19	38	f_4	LPSO	48	96
	DSPSO	50	100		DSPSO	50	100

其中 f_1 是单峰函数，f_2、f_3、f_4 为双峰函数。

由表 3 可知，在处理 f_1 之类的单峰函数问题时，DSPSO 算法并不比 SPSO 和 LPSO 具有很大优势；但在 f_2、f_3、f_4 多峰问题上，DSPSO 算法成功率明显高于其他两种算法。这表明，DSPSO 算法的改进对处理多峰复杂函数问题有明显优势。

6 结束语

经过实验分析，提出了学习因子随权重调整的混合粒子群算法。首先对惯性权重进行改进，使其随迭代次数和粒子状态改变；将学习因子表示为惯性权重的函数，加强了粒子的统一性和智能性；引入差分进化算法保持种群多样性，利于算法跳出局部收敛；最后对算法的收敛性进行分析，证明算法可行性。仿真结果表明该算法的寻优性能明显提升。在今后的算法研究中，应加强群算法的实际工程应用，可以将其应用到电机控制参数调整中，减小超调。

参考文献：

[1] EBERHART R C，KENNEDY J. A new optimizer using particle swarm theory[C]//Proceedings of the sixth international symposium on micro machine and human science. Nagoya：IEEE，1995：39-43.

[2] LIU Z H，WEI H L，ZHONG Q C，et al. GPU implementation of DPSO-RE algorithm for parameters identification of surface PMSM considering VSI nonlinearity[J]. IEEE Journal of Emerging and Selected Topics in Power Electronics，2017，5(3)：1334-1345.

[3] LIU Z H，ZHANG J，ZHOU S W，et al. Coevolutionary particle swarm optimization using AIS and its application in multiparameter estimation of PMSM[J]. IEEE Transactions on Cybernetics，2013，43(6)：1921-1935.

[4] LIU Z H，WEI H L，ZHONG Q C，et al. Parameter estimation for VSI-fed PMSM based on a dynamic PSO with learning strategies[J]. IEEE Transactions on Power Electronics，2016，32(4)：3154-3165.

[5] LIU Z H，LI X H，ZHANG H Q，et al. An enhanced approach for parameter estimation：using immune dynamic learning swarm optimization based on multicore architecture[J]. IEEE Systems，Man，and Cybernetics Magazine，2016，2(1)：26-33.

[6] 陈 雁，孙海顺，文劲宇，等. 改进粒子群算法在船舶电力系统网络重构中的应用[J]. 电力自动化设备，2011，31(3)：29-34.

[7] SHI Y，EBERHART R. A modified particle swarm optimizer[C]//1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360). Anchorage，Alaska，USA：IEEE，1998：69-73.

[8] 赵志刚，黄树运，王伟倩. 基于随机惯性权重的简化粒子群优化算法[J]. 计算机应用研究，2014，31(2)：361-363.

[9] 郑春颖，郑全弟，王晓丹，等. 基于试探的变步长自适应粒子群算法[J]. 计算机科学，2009，36(11)：193-195.

[10] 黄 洋，鲁海燕，许凯波，等. 基于 S 型函数的自适应粒子群优化算法[J]. 计算机科学，2019，46(1)：245-250.

[11] SUGANTHAN P N. Particle swarm optimiser with neighbourhood operator[C]//Proceedings of the 1999 congress on evolutionary computation. Piscataway：IEEE，1999：1958-1962.

[12] RATNAWEERA A，HALGAMUGE S. Self-organization hierarchical particle swarm optimizer with varying acceleration coefficients[J]. Evolutionary Computation，2004，8(3)：240-255.

[13] 毛开富，包广清，徐 驰. 基于非对称学习因子调节的粒子群优化算法[J]. 计算机工程，2010，36(19)：182-184.

[14] STORN R，PRICE K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization，1997，11(4)：341-359.

[15] DAS S，SUGANTHAN P N. Differential evolution：a survey of the state-of-the-art[J]. IEEE Transactions on Evolutionary Computation，2011，15(1)：4-31.