一种改进惯性权重的 PSO 算法

刘 伟,周育人
LIU Wei,ZHOU Yu-ren

华南理工大学 计算机科学与工程学院,广州 510006
South China University of Technology,Guangzhou 510006,China
E-mail:liuw_d@163.com

LIU Wei,ZHOU Yu-ren.Modifed inertia weight particle swarm optimizer.Computer Engineering and Applications,2009,45(7):46-48.

Abstract: For complex functions with high dimensions,standard particle swarm optimization methods are slow speed on convergence and easy to be trapped in local optimum.This paper proposes an inertia weight function,which can balance global and local search ability,fasten convergence speed,and by adding the mutation operation to the algorithm in the later phase,this algorithm improves the ability to break away from the local optimum solutions effectively.Experimental results on three typical complex functions with high dimensions show that the modified algorithm can rapidly converge at high quality solutions.

Key words: particle swarm optimization;inertia weight;mutation

摘 要:针对高维复杂函数优化,标准 PSO 算法收敛速度慢,易陷入局部最优点的缺点,提出一个惯性权重函数使算法的全局与局部搜索能力得到良好平衡,以达到快速收敛;并且该算法通过在后期进行变异操作,有效地增强了算法跳出局部最优解的能力。通过对三个典型的测试函数的优化所做的对比实验,表明改进的算法在求解质量和求解速度两方面都得到了好的结果。

关键词:粒子群优化;惯性权重;变异
DOI:10.3778/j.issn.1002-8331.2009.07.015 **文章编号**:1002-8331(2009)07-0046-03 **文献标识码**:A **中图分类号**:TP301.6

1 引言

粒子群算法 PSO(Particle Swarm Optimization)是一种基于集群智能的随机优化算法,最早由 Kennedy 和 Eberhart 于 20 世纪 90 年代中期提出[1-2]。该算法源于对鸟群的捕食行为的研究,通过对简单鸟和鱼群居社会系统的模拟,在多维解空间中构造所谓的"粒子群",粒子群中每个粒子通过跟踪自己的和群体所发现的最优值,修正自己的前进方向和速度,从而实现寻优。但是算法也存在易陷入局部最优和收敛精度不高等特点。

在标准粒子群算法的基础上,提出一个惯性权重函数来平衡全局和局部搜索能力,并根据粒子适应度值变化率对粒子位置进行变异操作,加快了算法的收敛速度,同时有利于摆脱局部极值的干扰,使目标函数逼近最优解。

2 标准粒子群算法及一些改进方法

标准 PSO 算法采用速度-位置搜索模型。每个微粒代表解空间中的一个解,解的优劣程度由适应函数决定。其中,适应函数由优化目标来决定。每个微粒在 n 维搜索空间中以一定的速度飞行,并根据自己的飞行经验和群体的飞行经验来调整自己的飞行。

设 $X_i=(x_{i1},x_{i2},\cdots,x_{in})$为微粒 i 当前位置;$V_i=(v_{i1},v_{i2},\cdots,v_{in})$为微粒 i 的当前飞行速度。

进化过程中,记录微粒到当前为止的历史最好位置 $P_i=(p_{i1},p_{i2},\cdots,p_{in})$和所有微粒的全局最好位置 $P_g=(p_{g1},p_{g2},\cdots,p_{gn})$。在每次迭代中,粒子跟踪个体最优值、全局最优值和自己前一时刻的状态来调整当前时刻的位置和速度,迭代公式如下:

$$v_{ij}(t+1)=wv_{ij}(t)+c_1r_1(p_{ij}(t)-x_{ij}(t))+c_2r_2(p_{gj}(t)-x_{ij}(t)) \quad (1)$$

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

式(1)表示微粒 i 的第 j 维的速度变化方程,式(2)表示位置变化方程,其中,t 表示第 t 代,c_1、c_2 是加速系数(或称学习因子),通常 $c_1=c_2=2.05$,分别调节向个体最好位置和全局最好位置方向飞行的最大步长。$r_1\sim U(0,1)$,$r_2\sim U(0,1)$为两个相互独立的随机函数。w 为惯性权重,一般在 0.1 到 0.9 之间取值。此外,粒子速度 v_{ij} 由最大速度 v_{max} 所限制,若 $v_{ij}>v_{max}$,则令 $v_{ij}=v_{max}$。

粒子群算法操作简单,使用方便,收敛速度快。由式(1)可知,通过不断调整 w 的大小,可以改变粒子的局部搜索性能与全局搜索性能。为此 Shi 和 Eberhart 在文献[3]提出线性递减权重(Linearly Decreasing Weight,LDW)策略,是指如果让 w 随算法迭代的进行而线性减少,将会显著改善算法的收敛性能。

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60673062);广东省自然科学基金(the Natural Science Foundation of Guangdong Province of China under Grant No.06025686)。

作者简介:刘伟(1980-),男,硕士研究生,研究方向:演化计算、遗传算法、粒子群算法等;周育人,男,工学博士,副教授,研究方向:演化计算、数据挖掘、智能计算等。

收稿日期:2008-10-22 **修回日期**:2008-12-25

惯性权重 w 按下式进行调节:

$$w=w_{max}-(w_{max}-w_{min})\times\frac{iter}{iter_{max}} \quad (3)$$

式中 w_{max} 为最大惯性因子一般取 0.9,w_{min} 为最小惯性因子一般取 0.1。$iter_{max}$ 为最大进化代数,$iter$ 为当前进化代数。实验表明 LDWPSO 算法在优化方程方面有明显的效果,但是这种算法中的惯性因子 w 的变化只与迭代次数线性相关,不能更好地适应于那些具有复杂、非线性变化特性的优化问题。

3　一种改进惯性权重的 PSO 算法

3.1　基于惯性权重的改进

对于不同的优化问题,如何确定局部搜索能力与全局搜索能的比例关系,对其求解过程非常重要。在 PSO 算法中,调节局部搜索能力与全局搜索力的关键在于惯性权重 w 的设计。

惯性权重 w 描述了粒子上一代速度对当前代速度的影响。控制其取值大小可调节 PSO 算法的全局与局部寻优能力。w 值较大,全局搜索能力越强,局部搜索能力越弱,反之,则局部搜索能力增强,而全局搜索能力减弱。对于惯性权重 w 来说,线性递减惯性权重只对某些问题有效,当待解问题很复杂时,该法使得 PSO 算法在迭代后期局部搜索能力不足,易陷入局部最优,导致不能找到要求的最优解。

因此,对于难优化的复杂高维函数,提出一个惯性权重函数来平衡全局和局部搜索能力:

$$w=w_{min}+(w_{max}-w_{min})\times\exp(-20\times(iter/iter_{max})^6) \quad (4)$$

取 w_{max}=0.9,w_{min}=0,惯性权重函数 w 的变化曲线如图 1 所示。

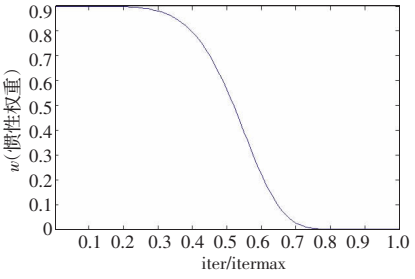

图 1　惯性权重变化曲线

对于难优化的复杂高维函数,在迭代初期需要加强全局搜索能力,增加全局搜索时间,一但定位到最优解的大致位置,就需要加强局部搜索能力,增加局部搜索时间来进行精细的局部搜索。在式(3)的线性递减权重策略中,w 从 w_{max} 线性减小到 w_{min},全局搜索和局部搜索的比例没有变化,全局搜索和精细的局部搜索时间不够,用式(3)作 PSO 的惯性权重去优化复杂的高维函数很难找到要求的最优解。

从图 1 的惯性权重变化曲线可以看出,本文提出的惯性权重函数使惯性权重 w 在迭代初期长时间保持了一个较大的值,在迭代后期长时间保持了一个较小的值,从而增加了迭代初期的全局搜索时间和迭代后期的局部搜索时间,加强了在迭代初期的全局搜索强度,以及在迭代后期的局部搜索强度,使全局搜索能力与局部搜索能力良好平衡。通过调整 w_{max} 和 w_{min} 的取值,可进一步对惯性权重函数进行调整。

在复杂高维函数的优化问题中,PSO 算法的求解结果为局部最优的概率远大于为全局最优的概率。运用提出的惯性权重函数,能有效地提高算法的运行效率,也同样容易陷入局部最优。为了解决这个问题,采用变异机制使其跳出局部最优。

3.2　基于变异的改进

3.2.1　变异时机的选择

(1)通过适应度变化率来确定变异时机

适应度变化率(FSlope):适应度变化率是本次迭代与本次迭代之前 M 次迭代之间,历史最佳粒子位置 P_g 的适应度值的变化率,定义如下:

$$FSlope=\frac{f(p_g(t))-f(p_g(t-M))}{f(p_g(t))}(t>M)$$

当适应度变化率(FSlope)小于某一阈值时,开始变异。这一变异时机是取全局最好位置 P_g 的适应度的变化率作为变异时机的参数选择,是针对整个粒子群进行的变异。如果是针对单个粒子,则以此粒子适应度的变化率作为变异时机的参数选择。

(2)通过粒子无进化的次数来确定变异时机

当粒子群的全局最好位置的适应度连续 M 次都没有得到改进,则认为粒子群已经聚集到某一局部最优位置,此时整个粒子群开始变异。这一变异时机与适应度变化率比较类似,也可以针对整个粒子群和单个粒子。对于单个粒子,如果它的适应值连续 M 次都没有得到改进,就可以对它进行变异。

以上两种变异时机都在一定程度上反映了历史最佳粒子位置 P_g 的变化过程,若 P_g 的适应度连续多次不变化或极其缓慢的变化,则认为算法可能处于停滞状态。对于高维复杂函数,粒子群的进化非常缓慢,单独以一种变异时机来判断变异,都有可能使粒子仍在进化时突然变异,破坏粒子群的结构。而将两者结合在一起则能更好的判断变异时机,因此提出一种将两者结合在一起,利用适应度变化率来确定粒子的无进化次数,并据此进行变异的改进。

3.2.2　基于适应度变化率和粒子无进化次数的变异

在提出的变异改进中,将适应度变化率(FSlope)定义为两次迭代运算之间,历史最佳粒子位置 P_g 的适应度值的变化率,定义如下:

$$FSlope=\frac{f(p_g(t))-f(p_g(t-1))}{f(p_g(t))} \quad (6)$$

粒子无进化次数用 *StopTime* 表示,适应度变化率阈值为 *Slopevalue*,粒子无进化次数阈值为 *MaxStep*。迭代开始时粒子无进化次数 *StopTime*=0。

在迭代过程中,粒子无进化次数根据适应度变化率来确定,如下所示:

$$\begin{cases}\text{if } FSlope\leqslant SlopeValue & StopTime=StopTime+1\\ \text{if } FSlope>SlopeValue & StopTime=0\end{cases} \quad (7)$$

若适应度变化率连续多次小于适应度变化率阈值 *Slopevalue*,从而使粒子无进化次数 *StopTime* 达到粒子无进化次数阈值 *MaxStep*,则认为算法可能处于停滞状态。此时按照变异率 ρ 对 P_g 进行变异操作,改变粒子群的前进方向,使粒子进入其他新的领域进行搜索。变异规则如下:

```
if StopTime>MaxStep
  按变异率ρ对 pgs(t)进行变异
  pgs(t)=pgs(t)+0.5×rand( )×pgs(t)   i=1,2,…,int(ρ×D)
  StopTime=0
end
```

其中 *rand*( )是在区间(0,1)上服从均匀分布的随机变量。

3.3 算法流程

综上所述，提出的改进惯性权重的 PSO 算法的基本流程如下：

步骤 1 初始化粒子群。给定种群规模 $popsize$、进化代数 $iter_{max}$、最大惯性因子 w_{max}、最小惯性因子 w_{min}、适应度变化率阈值 $Slopevalue$、变异率取 ρ、粒子无进化次数阈值 $MaxStep$。

步骤 2 初始化粒子的全局最优值 P_g 和个体最优值 P_i。

步骤 3 判断算法是否满足收敛准则，如果满足，则算法结束，否则执行步骤 4。

步骤 4 根据式(1)、式(2)、式(4)更新所有粒子的速度和位置，计算粒子的适应度，更新粒子的全局最优值 P_g 和个体最优值 P_i。

步骤 5 根据式(6)计算适应度值变化率，根据式(7)计算连续不变化次数 $Addtime$。

步骤 6 判断是否满足变异条件，若满足则根据 3.2 节的变异规则进行变异操作。

步骤 7 将迭代次数增加 1，并返回到步骤 3 执行。

4 仿真结果及分析

4.1 测试函数

本文测试函数为四个典型函数优化问题[4-5]，其中 Rosenbrock 函数是极难最小值非凸的病态函数，Griewank 函数和 Ackley 函数是多峰函数，极难找到全局最优解。

Rosenbrock 函数：

$$f_1(x)=\sum_{i=1}^{n}[100\times(x_{i+1}^2-x_i)+(1-x_i^2)],-30\leqslant x_i\leqslant 30$$

Griewank 函数：

$$f_2(x)=\frac{1}{4\,000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1,-600\leqslant x_i\leqslant 600$$

Ackley 函数：

$$f_3(x)=-20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e,30\leqslant x_i\leqslant 30$$

4.2 算例仿真结果比较

为了验证本文算法的有效性，分别采用文献[3]中的 LDW-PSO 算法和本文 MIW-PSO 算法对本文所列函数进行实例计算并进行对比与分析。两种算法的参数设置分别为：

(1)在 LDW-PSO 算法中，学习因子 $c_1=c_2=2.05$，迭代总次数 $iter_{max}=4\,000$，粒子群总粒子数为 30，最大惯性因子 $w_{max}=0.9$，最小惯性因子 $w_{min}=0.1$。

(2)在本文的 MIW-PSO 算法中，学习因子 $c_1=c_2=2.05$，迭代总次数 $iter_{max}=4\,000$，粒子群总粒子数为 30，最大惯性因子 $w_{max}=0.9$，最小惯性因子 $w_{min}=0$，变化率阈值根据文献[6]选择 $SlopeValue=10E-10$，变异率取 $\rho=0.4$，连续不变化次数阈值 $MaxStep=8$。

(3)算法的评估参数。①平均迭代次数；②平均收敛率，算法运行达到收敛的次数与算法运行总次数的比值；③达优率，算法运行达到最优解的次数与算法运行总次数的比值。

在 Matlab 2007 的运行环境中，每个函数用 MIW-PSO 算法和 LDW-PSO 算法在 10 维、30 维、60 维的情况下分别运行 20 次。具体结果比较见表 1~表 3。

表 1 函数 $f_1(x)$ 的运行情况对比

	LDW-PSO			MIW-PSO		
	平均迭代次数	平均收敛率	达优率	平均迭代次数	平均收敛率	达优率
10 维	2 143	1.00	0.80	154	1.00	0.70
30 维	1 604	0.70	0.70	613	0.90	0.85
60 维	2 221	0.60	0.50	1 222	0.90	0.70

表 2 函数 $f_2(x)$ 的运行情况对比

	LDW-PSO			MIW-PSO		
	平均迭代次数	平均收敛率	达优率	平均迭代次数	平均收敛率	达优率
10 维	218	1.00	1.00	741	1.00	1.00
30 维	659	1.00	0.85	269	1.00	0.90
60 维	770	1.00	0.60	304	1.00	0.80

表 3 函数 $f_3(x)$ 的运行情况对比

	LDW-PSO			MIW-PSO		
	平均迭代次数	平均收敛率	达优率	平均迭代次数	平均收敛率	达优率
10 维	799	1.00	0.80	192	1.00	0.75
30 维	1 527	1.00	0.60	133	1.00	0.70
60 维	1 486	0.70	0.40	154	1.00	0.55

从表 1~表 3 可以得出：用式(4)作惯性权重的本文算法收敛速度以及达优率明显好于用式(3)作惯性权重的标准算法仿真结果；本文算法对 $f_1(x)$、$f_2(x)$、$f_3(x)$ 高维复杂函数优化效果较好。

4.3 结论

针对高维复杂函数的优化问题，提出的惯性权重函数加强了算法的搜索强度，使全局搜索能力与局部搜索能力良好平衡，收敛速度明显增快；并且在算法中根据适应度值变化率对 P_g 进行变异操作，使粒子进入其他新的领域进行搜索，跳出局部最优，算法的优化性能较好。另本算法也适合一般函数优化。如何对 PSO 算法进行收敛分析，以及与其它方法相结合，并用之求解实际问题，将是进一步的研究工作。

参考文献：

[1] Kennedy J,Eberhart R C.Particle swarm optimization[C]//Proc IEEE International Conferece on Neural Networks.USA:IEEE Press,1995,4:1942-1948.

[2] Eberhart R C,Kennedy J.A new optimizer using particle swarm theroy[C]//Proc of the 6th international Symposium on MicroMachine and Hunan Science,Nagoya,Japan,1995:39-43.

[3] Shi Y,Eberhart R.A modified particle swarm optimizer[C]//IEEE World Congress on Computational Intelligence,Anchorage,Alaska,1998:69-73.

[4] Lei Kai-you,Wang Fang,Qiu Yu-hui,et al.An adaptive inertia weight strategy for particle swarm optimizer[C]//The 3rd Intl Conf on Mechat Ronics and Information Technology,Chongqing,China,2005.

[5] 吕振肃,侯志荣.自适应变异的粒子群优化算法[J].电子学报,2004,32(3):416-420.

[6] van den Bergh F.An analysis of particle swarm optimizers[D].South Africa:University of Pretoria,2002.

[7] 杨维,李歧强.粒子群优化算法综述[J].中国工程科学,2004,6(5):87-93.

(下转 55 页)