一种权重递增的粒子群算法

刘建华[1] 张永晖[1] 周 理[1] 贺文武[2]
(福建工程学院信息科学与工程学院 福州 350108)[1] (福建工程学院数理系 福州 350108)[2]

摘 要 粒子群算法(Particle Swarm Optimization,PSO)是仿真生物群体的社会行为的一种智能优化算法,现在已广泛应用到各种优化计算中。PSO算法的权重参数采用随迭代而递减的时变策略,权重时变值一般是根据试验结果来确定的,很少通过理论分析来选择权重。利用PSO算法的理论模型,分析权重值对算法的影响,并说明PSO算法采用时变权重的合理性。进一步根据分析模型,提出一种权重可以随迭代而递增的PSO算法模型。通过利用经典的基准函数,经仿真试验验证,这种权重递增的PSO算法优于传统权重递减的PSO算法,并且其性能与标准PSO算法相当。

关键词 粒子群算法,权重递增,群体智能,进化计算
中图法分类号 TP301.6 **文献标识码** A

Particle Swarm Optimization with Weight Increasing

LIU Jian-hua[1] ZHANG Yong-hui[1] ZHOU Li[1] HE Wen-wu[2]
(School of Information Science and Engineering,Fujian University of Technology,Fuzhou 350108,China)[1]
(Department of Mathematics and Physics,Fujian University of Technology,Fuzhou 350108,China)[2]

Abstract Particle swarm optimization (PSO) is an intelligent algorithm which simulates the social behavior of bird swarm or fish group and has been applied widely in all kinds of fields on optimization computation. The inertia weight of PSO has employed the policy of decreasing progressively with iteration,but the variable value of inertia weight is decided in term of the experiment and rarely analyzed with theory. This paper analyzed the inertia weight of PSO with the theoretical modal. And then a kind of PSO modal with inertia weight increasing progressively with iteration was provided. The benchmark functions were used to conduct the experiment. The test results show that PSO with weight increasing is superior to the traditional PSO with weight decreasing and can match with standard PSO.

Keywords Particle swarm optimization,Progressive weight increase,Swarm intelligence,Evolutionary computation

1 引言

粒子群算法(PSO)是Kennedy J和Eberhart R于1995年为模拟生物世界群体行为而提出的一种智能进化算法[1,2]。PSO算法主要用于非线性的连续优化问题,现广泛应用于大量的实际应用问题之中,显示了其良好的性能。特别是近十年来,PSO算法已受到广泛的关注与应用。其原因是算法的概念简单,易于实现,调整参数较少。

Shi Y和Eberhart R引入一个惯性权重参数,用来控制算法的全局搜索能力和局部搜索能力。惯性权重大小决定当前速度对粒子位置的影响大小,一个大的惯性权重有利于全局探测能力(搜索新的区域),而一个小的惯性倾向于局部探索,精细搜索目前的小区域。优化搜索算法早期应该具有更强的探测能力(全局搜能力)去发现好的解种,晚期需要更强的探索能力(局部搜索能力)去搜索好的局部解区域[3]。几乎所有的进化算法都利用一些参数策略去达到这个目的,例如,进化策略中的高斯变异步长参数[4]、模拟退算法的温度参数[5]。因此,PSO算法的权重 w 就是一个用于控制全局搜索与局部搜索的参数,其是一个随迭代递减而不是一个固定不变的值。权重 w 采用时变的方法,即权重随着算法的迭代运行,从一个最大值变化到最小值[6,7]。

自权重参数提出以来,研究者对权重参数做了大量研究和改进,文献[8]对权重做了综述,把PSO权重的取值大致分为3类:一类是取固定的值,例如Shi Y与Eberhart R在文献[6]中原始引入权重时采用固定权重的方法;第二类是取时变的权重,最经典的是线性递减的方式,还有Chatterjee与Siarry[9]提出的一种非线性递减的权重递减方式;第三类是自适用的权重值,例如Ismail A和Engelbrecht A提出一种自适用权重方法[10],以及J. Liu和X. Fan等提出一种动态权重方法[11]。权重改进变种虽然很多,但基本PSO算法还是采用权重线性递减的全局模型,或采用约束PSO的局部模型。

PSO算法的权重取值更多来源于直观分析与实验验证,

到稿日期:2013-05-15 返修日期:2013-09-02 本文受福建省科技厅重点项目(2012H0002),福建省自然科学基金(2012J01246,2012J01247),福建工程学院启动基金(E0600100)资助。
刘建华(1967—),男,博士,副教授,主要研究方向为智能计算、数据挖掘、智能交通等,E-mail:jhliu@fjnu.edu.cn(通信作者);张永晖(1973—),男,博士,副教授,主要研究方向为智能计算、智能仓储等;周 理(1977—)男,博士,讲师,主要研究方向为物联网、有限元计算等;贺文武(1972—),男,博士,副教授,主要研究方向为机器学习、数据挖掘等;

其权重随时变递减主要是通过实践验证其可行性与必要性。权重取值的理论分析研究并不多，也不太完善。van den Bergh F 利用差分方程对 PSO 算法做了比较系统的理论分析和大量的实验验证，并由此改进 PSO 算法[12,13]。J. Liu 等根据原有的理论，进一步分析 PSO 算法的随机性与交互性对 PSO 算法的影响[14]。但其都没有进一步分析权重效果、权重时变性选择的依据。本文根据文献提出的 PSO 算法理论模型分析说明权重的时变选值的原因，并且进一步根据 PSO 算法的分析理论模型，提出一种权重递增的 PSO 算法模型。本文第 2 节介绍粒子群算法及其理论分析模型；第 3 节利用 PSO 算法的理论分析模型分析说明 PSO 算法权重递减的理论依据，并提出一种权重递增的 PSO 算法模型；第 4 节利用标准的基本函数，通过实验分析验证提出的权重递增 PSO 算法总体要优于权重递减的基本 PSO 算法；最后总结全文。

2 粒子群算法及分析模型

2.1 基本粒子群算法

粒子群算法(PSO)是一种基于迭代模式的优化算法，其数学描述如下[3]：

一个由 n 个粒子(particle)组成的群体在 D 维搜索空间中求解，粒子群的第 i 个粒子是由 3 个 D 维向量组成，其包括 3 部分，分别为：

目前位置：$x_i=(x_{i1},x_{i2},\cdots,x_{iD})$；

历史最优位置：$p_i=(p_{i1},p_{i2},\cdots,p_{iD})$；

速度：$v_i=(v_{i1},v_{i2},\cdots,v_{iD})$；

这里 $i=1,2,\cdots,n$。目前位置被看作描述空间点的一套坐标，在算法每一次迭代中，目前位置 x_i 作为问题解被评价。如果目前位置好于历史最优位置 p_i，那么目标位置的坐标就存在第二个向量 p_i。另外，整个粒子群中迄今为止搜索到的最好位置记为：

$$p_g=(p_{g1},p_{g2},\cdots,p_{gD})$$

对于每一个粒子，其第 d 维($1\leqslant d\leqslant D$)根据如下等式变化：

$$v_{id}=w\cdot v_{id}+c_1\cdot rand()\cdot(p_{id}-x_{id})+c_2\cdot rand()\cdot(p_{gd}-x_{id}) \tag{1}$$

$$x_{id}=x_{id}+v_{id} \tag{2}$$

其中，w 是一权重；加速常数 c_1 和 c_2 是两个非负值，这两个常数使粒子向自己的历史最优点以及群体内或领域内的全局最优点靠近，c_1 和 c_2 通常等于 2。$rand()$ 是在范围[0,1]内取值的随机函数。V_{max} 是常数，限制了速度的最大值，由用户设定。粒子的速度被限制在一个范围$[-V_{max},V_{max}]$内。

w 为惯性权重，它起到平衡算法全局搜索和局部搜索能力的作用。文献[5]进一步提出自适应调整的策略，即随着迭代的进行，线性减少 w 的值。这使得算法在迭代初期探索能力较强，可以不断搜索新的区域，然后收敛能力逐渐增强，使算法在可能的最优解周围精细搜索。这是使用最广泛的基本 PSO 算法，其权重计算形式如下：

$$w=w_{min}+(w_{max}-w_{min})*\frac{t_{max}-t}{t_{max}} \tag{3}$$

其中，w_{max} 是开始运行时初始的权重，w_{min} 是最后运行的权重，t_{max} 是运行的最大迭代次数，t 是当前的迭代的次数，w 是当前的权重。

2.2 粒子群算法的分析模型

Fan van den Bergh 在标准 PSO 系统的简化模型中，利用线性系统理论分析了粒子的收敛性，并得到粒子轨迹的显式表达式，下面简述其过程[12,13]。

因粒子轨迹的分析对不同粒子在不同维数里是一样的，所以把两个下标 d 和 i 去掉，不考虑粒子下标 i 和维数下标 d，并且在离散时间步骤中考虑粒子的轨迹，因此 v_{id} 和 x_{id} 可以表示为 $v(t)$ 和 $x(t)$，分别代表粒子在时间步 t 时的速度与位置。

假设 $\phi_1=c_1\cdot rand()$，$\phi_2=c_2\cdot rand()$，$\phi=\phi_1+\phi_2$，如果 p，p_g，ϕ_1 和 ϕ_2 保持固定的值，则由式(1)和式(2)消去速度相关项可以得到式(4)：

$$x(t+1)=(1+w-\phi)\cdot x(t)-w\cdot x(t-1)+\phi_1\cdot p+\phi_2\cdot p_g \tag{4}$$

文献[7,8]经过推导，式(4)可以解得：

$$x(t)=k_1+k_2\alpha^t+k_3\beta^t \tag{5}$$

其中，k_1，k_2 和 k_3 是常数，由系统的初始条件决定。

由式(5)可知，$x(t)$要收敛，α，β 的绝对值必须小于 1，即 $\max(\|\alpha\|,\|\beta\|)<1$，则粒子的位置 $x(t)$按下式收敛：

$$\lim_{t\to\infty}x(t)=(1-a)p+ap_g \tag{6}$$

这里 $a=\dfrac{\phi_1}{\phi_1+\phi_2}$，所以 $a\in[0,1]$。

由于 $\max(\|\alpha\|,\|\beta\|)<1$ 时才能保证式(6)成立，因此可以计算参数 w，ϕ_1，ϕ_2 需要满足什么关系才使粒子的轨迹收敛。经过推导，当不考虑随机性且个体最优点和群体全局点为常数时，PSO 算法的收敛条件为：

$$\begin{cases}0<w<1\\0<\phi<4\\w>\dfrac{1}{2}\phi-1\end{cases} \tag{7}$$

式(7)刻画 PSO 算法的参数在保证算法收敛条件下的取值范围，显然在此之外算法不能收敛，这可以根据 $\max(\|\alpha\|,\|\beta\|)$的值来判断。

3 权重递增模型

3.1 收敛条件

根据式(7)，图 1 演示 $\max(\|\alpha\|,\|\beta\|)$随 ϕ 和权重变化的三维空间。基本 PSO 的权重最大值小于 2，而 $\phi=\phi_1+\phi_2$ 通常小于 4。因此，图 1 中的 ϕ 值范围为[0,4]，而权重范围为[0,2]。根据式(5)，当参数取值能保证 $\max(\|\alpha\|,\|\beta\|)<1$ 时，算法才能收敛，否则算法发散，而其收敛的参数范围由式(7)决定。

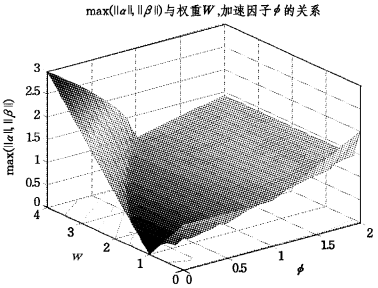

图 1 $\max(\|\alpha\|,\|\beta\|)$与权重 w、加速因子的关系

根据式(7)，当参数需要满足公式 $w>0.5(\phi_1+\phi_2)-1$ 时，PSO 算法才能收敛。显然，w 值越大，使算法收敛的 $\phi_1+\phi_2$ 值的范围也越大，收敛条件越容易满足。但根据基本 PSO 算法的思想，其权重 w 值随迭代运行而线性递减，即说明 w 越小，越有助于算法收敛，有助于算法晚期增强局部搜索能力，这似乎有矛盾。但其实收敛与收敛速率是两个不同的概念。收敛并不意味着收敛速度快，选择的参数有利于收敛，但收敛速率不一定要快。

图 2 演示 $\max(\|\alpha\|,\|\beta\|)$ 随权重 w 变化的过程。这里 $\phi=\phi_1+\phi_2$ 且 ϕ 取固定值 2。因为 $\phi_1=c_1\cdot\text{rand}()$，$\phi_2=c_2\cdot\text{rand}()$，而 c_1，c_2 通常为 2，所以 ϕ 取 2，实际为其随机变化的期望值。

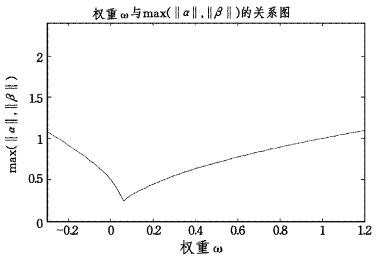

图 2　$\max(\|\alpha\|,\|\beta\|)$ 和权重 w 的关系

观察图 2 中权重 w 的取值，当 $w<1$ 时，$\max(\|\alpha\|,\|\beta\|)<1$。根据式(5)，PSO 算法收敛。但 $\max(\|\alpha\|,\|\beta\|)$ 的值有一个折线变化，当它的值大于 1 时，算法发散。而当它的值小于 1 时，算法收敛，并且其值越小，收敛越快。随着 w 变小到约为 0.2 时，$\max(\|\alpha\|,\|\beta\|)$ 变成最小值，即算法收敛速度为最快，但随后 $\max(\|\alpha\|,\|\beta\|)$ 越变越大。因此，这个结论与一些文献的试验要求相符[12,13]，即对权重 w 的取值要求线性递减，但不能低于 0.2。所以，图 2 说明 $\max(\|\alpha\|,\|\beta\|)$ 取值的确需要权重线性递减。算法初期需要全局开拓性，不需要收敛，这样有利于全局开拓性；而后期算法需要局部探测性，希望算法收敛速率更快，因此权重的值要变小。图 2 说明了权重动态递减的根本原因。但是，权重不能小于 0.2，而且实际应用中未必要求收敛过快，所以，在基本 PSO 算法中，权重线性递减一般要求为 0.9 到 0.4，不能太小。

3.2　一种权重递增的方法

时变权重方法普遍采用的是递减的方法，其原因是如上节所述，为了使 $\max(\|\alpha\|,\|\beta\|)$ 的值也是从大到小变化，算法能够在早期具有全局探索性、晚期具有局部探测性。因此，基本 PSO 算法的权重是从大到小通常采用线性递减的方法。最大值一般是在[1,2]之间，而最小值一般在[0.2,0.6]之间，这与图 2 的体现形式基本一致。

但从图 2 中可以看到，当权重从 −0.22 递增变化到 0.2 时，其 $\max(\|\alpha\|,\|\beta\|)$ 的值也是从大到小变化。其变化与权重从 1.2 到 0.2 的变化造成 $\max(\|\alpha\|,\|\beta\|)$ 变化的效果一样，都使 $\max(\|\alpha\|,\|\beta\|)$ 的值也是从大到小变化。因此，当其它参数不变化时，PSO 算法的权重从 −0.2 递增变化到[0,0.2]某值的效果，与从 2 递减变化到[1,0.2]某权重值的效果是相当的，也可以使算法在早期具有全局探索性、晚期具有局部探测性。为此，这里设计一种权重递增的 PSO 算法，此算法与原权重递减的 PSO 算法基本类似，只是算法的权重采用从某个负值递增到不大于 0.2 的某值的变化策略，即权重计算公式如下：

$$w=w_{\min}+(w_{\max}-w_{\min})*\frac{t}{t_{\max}} \tag{8}$$

其中，$w_{\max}$ 是开始运行时初始的权重，$w_{\min}$ 是最后运行的权重，$t_{\max}$ 是运行的最大迭代次数，t 是当前的迭代次数，w 是当前的权重。式(8)与式(3)对比，式(8)是权重 w 随 t 递增而递增，而前式的权重 w 是随 t 的递增而减小。当然，两个公式中的 $w_{\max}$ 和 $w_{\min}$ 两量取值不同，其具体取值通过实验测定。两个算法的流程是一样的，只是计算权重公式不同。

4　实验分析

4.1　实验设置

为了测试本文提出的权重递增 PSO 算法(记为：PSOIncr)的性能，这里主要与权重递减的全局基本 PSO 算法(记为：PSO)模型进行比较。同时为了进一步比较本文算法的性能，与 2007 年总结出的标准 PSO(记为：SPSO)的性能进行比较。标准 PSO 是总结多种 PSO 算法变种后提出的一种性能比较好的 PSO 算法，其采用约束型 PSO 算法的局部模型[15]。

为了评价权重递增的 PSO 算法的性能，这里采用由 CEC2005 提供的 14 个测试函数来进行实验分析[16]。这些函数包括 5 个单模函数(f_1，f_2，f_3，f_4，f_5)、9 个多模函数(f_6，f_7，f_8，f_9，f_{10}，f_{11}，f_{12}，f_{13}，f_{14})。表 1 列出这些函数各自的名称、范围和最优值。函数最优值一般不为 0，一般添加了偏置值。这些函数的最优位置从零点平移到其解空间的不同位置，这种最优点的偏置值可避免算法对最优点的偏好，能更公平地评价算法。关于这些函数细节可参考文献[14]。

表 1　测试函数

No.	函数名	范围	最优值
f_1	Shifted Sphere Function	$[-100,100]^D$	−450
f_2	Shifted Schwefel's Problem 1.2	$[-100,100]^D$	−450
f_3	Shifted Rotated High Conditioned Elliptic Function	$[-100,100]^D$	−450
f_4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	$[-100,100]^D$	−450
f_5	Schwefel's Problem 2.6 with Global Optimum on Bounds	$[-100,100]^D$	−310
f_6	Shifted Rosenbrock's Function	$[-100,100]^D$	390
f_7	Shifted Rotated Griewank's Function without Bounds	$[0,600]^D$	−180
f_8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	$[-32,32]^D$	−140
f_9	Shifted Rastrigin's Function	$[-5,5]^D$	−330
f_{10}	Shifted Rotated Rastrigin's Function	$[-5,5]^D$	−330
f_{11}	Shifted Rotated Weierstrass Function	$[-0.5,0.5]^D$	90
f_{12}	Schwefel's Problem 2.13	$[-\pi,\pi]^D$	−460
f_{13}	Expanded Extended Griewank's plus Rosenbrock's Function (F8F2)	$[-3,1]^D$	−130
f_{14}	Shifted Rotated Expanded Scaffer's F6	$[-100,100]^D$	−300

4.2　参数的设置

PSOIncr 算法与 PSO 算法的每个粒子的邻居为所有的粒子，粒子群设为 50 个，递增型与递减型 PSO 算法的参数 c_1 和 c_2 设置为 2。SPSO 算法的参数根据文献[15]提出的参数

设置，其详细内容可以参考文献[15]。实验采用 matlab12b 语言环境，并在 window7 下执行实现。

全局 PSO 采用由文献[7]提出的从 0.9 到 0.4 递减的权重，而本文提出的权重递增 PSO 算法模型采用最小权重到最大权重递增方法，其最小权重与最大权重的值通过实验来确定。根据收敛条件式(7)，权重范围是 $0<w<1$；实际计算时，早期权重参数最好取值在收敛范围之外，而晚期权重取值在权重收敛范围之内。所以传统权重递减 PSO 算法考虑权重从 1.2→0.4 递减。同样本文的权重递增 PSO 算法要求权重从−0.2→0.2 递增，其都是从不收敛的值变化到收敛的值。

本实验利用递增型 PSO 算法对两个基准函数进行计算，并使权重最小值从−0.2 变化到−0.1，而最大值从 0.1 变化到0.2，步长分为 0.01，各种值组合迭代计算。本实验对两个基准函数(f_6，f_{13})在 30 维下运行 25 次，计算其函数值与其真正最优值的差值($f(x)-f(x^*)$，$f(x^*)$是基准函数真正的最优值)平均值，结果见表 2 和表 3。观察表 2 和表 3 的计算结果，在权重递增模型中，两个表显示，最小值为−0.14、−0.15 及−0.16 比较好，而最大值为 0.15、0.16 及 0.17 较好。根据这个计算结果，本文采用权重从−0.15 到 0.16 的递增变化，即最小值为−0.15，而最大值为 0.16 。

表 2　在 10 维空间递增型 PSO 算法计算函数 f_6 运行 25 次的均值的不同权重变化结果

	−0.20	−0.19	−0.18	−0.17	−0.16	−0.15	−0.14	−0.13	−0.12	−0.11	−0.10
0.10	93.20	141.09	165.27	323.48	199.81	380.37	341.60	209.73	249.60	215.04	141.66
0.11	357.84	133.28	215.51	334.98	272.44	391.93	176.38	330.57	238.12	182.28	74.24
0.12	294.62	229.38	116.44	330.65	214.63	376.32	237.82	92.94	199.27	126.96	99.66
0.13	149.48	**115.17**	227.57	361.06	82.90	82.63	147.43	136.74	256.87	169.18	90.38
0.14	138.87	148.75	215.35	138.92	151.27	175.06	162.75	**41.93**	202.96	123.33	330.43
0.15	236.75	218.32	145.06	167.53	73.615	68.7	87.53	47.41	168.15	322.55	157.54
0.16	172.60	164.17	314.38	274.48	174.39	71.66	53.78	66.81	305.22	190.14	174.36
0.17	237.18	306.18	125.91	205.04	165.67	98.10	95.22	192.05	260.89	172.54	332.44
0.18	118.04	317.56	188.03	131.93	163.76	240.51	203.13	105.86	213.47	162.01	90.49
0.19	307.47	153.44	254.02	145.96	172.34	132.42	188.08	161.22	207.49	281.75	120.75
0.20	286.45	204.99	111.87	148.37	107.92	122.50	254.11	109.64	107.77	122.49	290.27

表 3　在 10 维空间递增型 PSO 算法计算函数 f_{13} 运行 25 次的均值的不同权重变化结果

	−0.20	−0.19	−0.18	−0.17	−0.16	−0.15	−0.14	−0.13	−0.12	−0.11	−0.10
0.10	4.2130	4.6410	4.5624	4.6811	4.3593	4.9983	5.5034	3.8559	4.5378	4.9060	4.0025
0.11	4.0049	4.5098	4.0407	4.0038	3.8862	5.0820	3.9683	3.7932	4.5142	5.4488	3.7467
0.12	4.1383	4.7174	4.4930	4.4185	**3.5478**	3.9587	**3.5950**	4.1755	5.3159	4.4091	4.4234
0.13	4.1774	4.1646	4.3539	3.7471	4.0958	3.8353	3.6235	3.5624	3.7630	4.8272	4.3785
0.14	4.1780	4.3869	3.7662	4.1732	3.7344	3.5562	3.9504	3.9890	4.1721	3.9941	4.2317
0.15	3.8646	4.1203	3.8933	3.7606	3.9414	3.7146	**3.4354**	3.8864	3.6916	3.9972	4.2181
0.16	4.2004	4.2468	4.1370	4.3631	3.6301	**3.3239**	**3.3786**	**3.5199**	3.8866	4.2943	4.1852
0.17	3.9187	4.3373	4.3569	**3.6075**	3.6173	**3.5419**	3.6046	3.9518	4.2054	4.0158	4.0357
0.18	4.1651	**3.8022**	3.9162	4.0392	4.0023	3.6862	3.7064	4.6249	3.9768	**3.6197**	3.8162
0.19	4.4937	4.0933	4.2055	4.4702	4.0556	4.1138	4.5978	4.4917	4.4903	4.3622	4.3488
0.20	4.1586	3.8918	4.2992	3.8664	3.8704	4.4997	4.1690	4.1735	4.4019	3.9942	4.2388

4.3　实验结果与分析

实验对 14 个基准函数的每一个函数运行 25 次后，计算其函数值与其真正最优值的差值($f(x)-f(x*)$，$f(x*)$是基准函数真正的最优值)平均值、标准方差和其中最好值。每个算法分别在 10 维和 30 维两种情况下运行。在 10 维情况下，每个算法每次运行对函数值评估次数为 1000、10000 和 10000；而在 30 维情况下，每个算法每次运行对函数值评估次数为 10000、100000 和 300000。

表 4 是 14 个基准函数在 10 维空间运行 25 次的各自与其最优值的平均差值、标准方差和最优值，其评价次数分别为 1000、10000 和 100000 次。表 5 是 14 个基准函数在 30 维空间运行 25 次的各自与其最优值的平均差值、标准方差和最优值，其评价次数分别为 1000、100000 和 300000 次。

根据表 4，权重递增型 PSO 算法总体要好于权重递减型 PSO 算法。在 10 维的空间中，当计算评价次数为 1000 时，递增型 PSO 算法除第 8 个函数外，其它性能都要高于原递减型的 PSO 算法；在评价次数为 10000 和 100000 时，虽然 4 个函数与最优值的平均差值不如递减型模型，但其它函数的效果都要好于递减模型，并且不好的效果也不会太差于递减模型，而好的效果却比递减模型要好很多。尤其对函数 f_1，f_2，f_3 及 f_6 的效果有较大的提高。表 5 展示在 30 维空间中相同的效果，当评价次数为 10000 时，递增模型都要好于递减模型，而评价次数为 100000 和 300000 次时，除了几个函数稍微略差于递减模型外，其它函数的性能都要优于递减模型，甚至出现对几个函数的效果有较大提高。与标准 PSO 算法相比，递增型 PSO 算法在评价次数较低、维数不高时，其性能要好于标准 PSO 算法，当随着评价次数增加，维数由 10 维变化到 30 维时，其性能会略低于标准 PSO 算法。

图 3−图 16 展示了 14 个基准函数在 30 维空间、评价次数为 100000 次(每代 50 个粒子，共运行 2000 代)时各函数值与最优值差的对数值的迭代变化过程，很形象地演示了两种算法的效果，说明递增 PSO 模型对计算大部分的函数优值，其效果都要好于原来的递减 PSO 模型。只有几个函数的效果略低于递减的 PSO 模型，而与标准 PSO 算法相比，其性能会大致相平。

表 4　在 10 维空间基准函数运行 25 次的均值、标准差和最好值的统计结果

FES		No.	PSO	SPSO	PSOIncr	No.	PSO	SPSO	PSOIncr
	Mean		3.32169e+003	1.44173e+003	**4.64890e+001**		**2.06546e+001**	2.07319e+001	2.07249e+001
	Std	f_1	1.44724e+003	4.74900e+002	**5.23449e+001**	f_8	**1.55495e−001**	1.17860e−001	1.21100e−001
	Best		5.17050e+002	6.85233e+002	**2.26927e+000**		**2.02769e+001**	2.04621e+001	2.03794e+001
	Mean		1.21264e+004	4.90779e+003	**1.21099e+003**		5.17435e+001	6.66501e+001	**3.62241e+001**
	Std	f_2	4.25339e+003	2.40834e+003	**6.80362e+002**	f_9	1.27111e+001	8.13338e+000	**1.00049e+001**
	Best		4.90087e+003	1.46840e+003	**2.10278e+002**		2.89063e+001	5.04306e+001	**1.44348e+001**
	Mean		4.38178e+007	1.56564e+007	**3.81703e+006**		7.99871e+001	7.88808e+001	**5.30912e+001**
	Std	f_3	3.11474e+007	1.15193e+007	**2.74724e+006**	f_{10}	1.67662e+001	1.13705e+001	**1.40562e+001**
	Best		6.85224e+006	1.48143e+006	**3.57034e+005**		4.40713e+001	5.49095e+001	**2.79219e+001**
	Mean		1.38614e+004	6.28269e+003	**3.03673e+003**		9.00448e+000	9.43028e+000	**8.92706e+000**
1e+3	Std	f_4	6.63180e+003	2.52400e+003	**1.52323e+003**	f_{11}	1.28687e+000	8.29105e−001	**1.09274e+000**
	Best		3.39549e+003	3.17476e+003	**3.92916e+002**		5.55298e+000	7.41120e+000	**6.73755e+000**
	Mean		1.09054e+004	2.59667e+003	**2.91042e+003**		3.67109e+004	2.25458e+004	**7.83531e+003**
	Std	f_5	2.62222e+003	1.19396e+003	**1.64785e+003**	f_{12}	1.78702e+004	8.53018e+003	**1.47795e+004**
	Best		5.89525e+003	7.04522e+002	**8.44740e+002**		1.18013e+004	6.17203e+003	**7.13019e+001**
	Mean		3.10368e+008	2.45129e+007	**5.78427e+004**		7.80544e+000	5.04655e+000	**3.79969e+000**
	Std	f_6	3.08828e+008	1.75679e+007	**7.54043e+004**	f_{13}	3.05791e+000	5.64936e−001	**1.05784e+000**
	Best		2.11168e+006	3.71757e+006	**7.04511e+002**		3.75601e+000	3.72060e+000	**1.02453e+000**
	Mean		1.39721e+003	1.26726e+003	**1.31394e+003**		4.04477e+000	4.17584e+000	**3.99422e+000**
	Std	f_7	1.17592e+002	1.11337e−001	**6.40996e+001**	f_{14}	2.78604e−001	1.24923e−001	**1.97890e−001**
	Best		1.26785e+003	1.26723e+003	**1.26706e+003**		3.28037e+000	3.87165e+000	**3.50271e+000**
	Mean		2.08911e+002	5.40211e−003	**4.16432e−009**		**2.04939e+001**	2.03722e+001	2.04980e+001
	Std	f_1	3.74706e+002	4.10998e−003	**1.12261e−008**	f_8	**1.05123e−001**	8.02741e−002	1.54509e−001
	Best		5.65075e−005	2.64259e−004	**1.81899e−012**		**2.02552e+001**	2.02060e+001	2.00911e+001
	Mean		1.30091e+002	3.68985e+001	**1.21803e+000**		8.35767e+000	1.14780e+001	**7.80582e+000**
	Std	f_2	1.30982e+002	2.69538e+001	**1.07919e+000**	f_9	3.27479e+000	2.98534e+000	**2.69126e+000**
	Best		1.93164e−001	7.44086e+000	**1.17998e−002**		2.75172e+000	6.01297e+000	**2.99230e+000**
	Mean		3.00115e+006	6.02057e+005	**6.53437e+005**		**2.04942e+001**	1.91263e+001	2.22162e+001
	Std	f_3	6.57477e+006	3.91037e+005	**3.61730e+005**	f_{10}	**7.10782e+000**	5.87262e+000	1.25050e+001
	Best		1.04099e+005	1.46400e+005	**1.13696e+005**		**8.40258e+000**	9.89936e+000	4.87049e+000
	Mean		1.42384e+002	1.50360e+002	**2.05292e+001**		**3.96348e+000**	5.58078e+000	5.04295e+000
1e+4	Std	f_4	1.42442e+002	1.01926e+002	**1.96849e+001**	f_{11}	**1.37680e+000**	9.21767e−001	1.29802e+000
	Best		1.68170e+000	8.09479e+000	**1.79247e+000**		**1.21034e+000**	3.46687e+000	2.17708e+000
	Mean		7.76493e−005	4.88749e−003	**1.30089e−001**		**3.65820e+003**	6.54312e+002	3.79700e+003
	Std	f_5	8.06349e−005	7.66164e−003	**1.76988e−001**	f_{12}	**5.64259e+003**	7.48974e+002	4.95912e+003
	Best		8.55801e−006	9.82516e−005	**1.36606e−009**		**1.08496e+001**	3.55868e+001	7.18545e+000
	Mean		7.74780e+006	9.50675e+001	**6.97139e+001**		1.70991e+000	9.97577e−001	**1.54588e+000**
	Std	f_6	1.79804e+007	1.11108e+002	**1.59732e+002**	f_{13}	8.58752e−001	2.37739e−001	**5.96607e−001**
	Best		4.39794e+000	1.32603e+001	**3.64786e−001**		6.32417e−001	4.91576e−001	**6.85748e−001**
	Mean		1.26735e+003	1.26721e+003	**1.26705e+003**		3.50302e+000	3.47444e+000	**3.40437e+000**
	Std	f_7	2.53993e−001	4.56755e−002	**2.27359e−008**	f_{14}	3.19986e−001	2.14309e−001	**4.08105e−001**
	Best		1.26723e+003	1.26705e+003	**1.26705e+003**		2.85002e+000	2.88019e+000	**2.48031e+000**
	Mean		4.13032e+001	0.00000e+000	**4.54747e−015**		**2.02938e+001**	2.02627e+001	2.03189e+001
	Std	f_1	5.65568e+001	0.00000e+000	**1.57392e−014**	f_8	**6.94421e−002**	6.72440e−002	7.14857e−002
	Best		0.00000e+000	0.00000e+000	**0.00000e+000**		**2.01566e+001**	2.01192e+001	2.00975e+001
	Mean		9.10537e+001	4.16852e−014	**6.59384e−014**		5.26899e+000	5.21789e+000	**1.87052e+000**
	Std	f_2	7.93139e+001	3.31563e−014	**3.14785e−014**	f_9	5.92626e+000	2.61457e+000	**7.77088e−001**
	Best		5.68434e−014	0.00000e+000	**0.00000e+000**		0.00000e+000	9.94959e−001	**0.00000e+000**
	Mean		1.83585e+006	7.46342e+004	**1.54662e+005**		2.02922e+001	1.19342e+001	**1.26514e+001**
	Std	f_3	3.51889e+006	5.32645e+004	**1.15530e+005**	f_{10}	8.79848e+000	3.65538e+000	**6.31309e+000**
	Best		9.96846e+004	9.50606e+003	**4.13897e+004**		8.95463e+000	6.96471e+000	**3.97984e+000**
	Mean		2.90701e+002	3.45906e−006	**2.26797e+000**		**3.49931e+000**	4.32363e+000	3.56889e+000
1e+5	Std	f_4	1.46799e+002	1.48859e−005	**1.13399e+001**	f_{11}	**1.29509e+000**	1.08021e+000	1.16431e+000
	Best		5.63891e+001	3.39185e−010	**8.69704e−012**		**1.22304e+000**	1.39111e+000	1.40420e+000
	Mean		**0.00000e+000**	0.00000e+000	1.23400e−010		**1.66136e+003**	1.11779e+002	2.56000e+003
	Std	f_5	**0.00000e+000**	0.00000e+000	4.18602e−010	f_{12}	**3.31335e+003**	2.95680e+002	3.67812e+003
	Best		**0.00000e+000**	0.00000e+000	0.00000e+000		**5.41956e−002**	2.06501e−002	1.84613e+000
	Mean		2.35081e+007	2.01382e+000	**2.46135e+001**		**6.90303e−001**	6.09965e−001	7.96083e−001
	Std	f_6	6.46658e+007	2.87824e+000	**4.20677e+001**	f_{13}	**2.58602e−001**	1.52055e−001	4.08367e−001
	Best		8.90593e+000	3.34489e−003	**1.94132e−002**		**3.66839e−001**	3.54147e−001	3.80897e−001
	Mean		1.32061e+003	1.26723e+003	**1.26705e+003**		2.83719e+000	3.02135e+000	**2.76565e+000**
	Std	f_7	1.84228e+002	1.24514e−001	**5.18907e−013**	f_{14}	3.87662e−001	3.26014e−001	**4.66739e−001**
	Best		1.26723e+003	1.26705e+003	**1.26705e+003**		1.74585e+000	2.13465e+000	**1.84029e+000**

表 5　在 30 维空间基准函数运行 25 次的均值、标准差和最好值的统计结果

FES		No.	PSO	SPSO	PSOIncr	No.	PSO	SPSO	PSOIncr
	Mean		4.98355e+003	1.98963e+002	**5.34280e+000**		2.10895e+001	2.10101e+001	**2.10682e+001**
	Std	f_1	3.93648e+003	6.31508e+001	**5.65236e+000**	f_8	6.20321e−002	8.01338e−002	**6.73100e−002**
	Best		6.39021e+002	9.74439e+001	**6.21580e−001**		2.09720e+001	2.07798e+001	**2.09142e+001**
	Mean		1.20179e+004	1.62976e+004	**9.18511e+003**		1.51851e+002	1.35379e+002	**8.33113e+001**
	Std	f_2	4.77066e+003	3.57612e+003	**2.22819e+003**	f_9	2.52005e+001	1.92836e+001	**1.86810e+001**
	Best		6.40977e+003	1.07217e+004	**4.15375e+003**		1.01153e+002	1.00520e+002	**4.89668e+001**
	Mean		5.97909e+007	3.99383e+007	**2.73287e+007**		2.50903e+002	2.04541e+002	**2.07759e+002**
	Std	f_3	2.73796e+007	1.04956e+007	**8.33413e+006**	f_{10}	2.09862e+001	2.27919e+001	**4.06657e+001**
	Best		1.96856e+007	1.01893e+007	**1.30247e+007**		2.15117e+002	1.57519e+002	**1.42224e+002**
	Mean		2.07876e+004	2.84372e+004	**1.76718e+004**		3.12858e+001	3.09114e+001	**2.97586e+001**
1e+4	Std	f_4	4.24848e+003	6.56060e+003	**5.74941e+003**	f_{11}	3.55945e+000	2.15239e+000	**2.90378e+000**
	Best		1.29491e+004	1.63640e+004	**6.97224e+003**		2.48021e+001	2.75458e+001	**2.55055e+001**
	Mean		8.65650e+003	6.25166e+003	**6.66049e+003**		1.15154e+005	1.08577e+005	**6.71376e+004**
	Std	f_5	3.06770e+003	1.02627e+003	**1.77452e+003**	f_{12}	5.82942e+004	3.60097e+004	**3.04705e+004**
	Best		1.39592e+003	3.81210e+003	**3.86507e+003**		1.65965e+004	4.63080e+004	**3.06352e+004**
	Mean		6.83765e+008	2.11047e+006	**1.30423e+004**		2.01263e+001	1.07216e+001	**1.00210e+001**
	Std	f_6	9.82399e+008	1.11223e+006	**1.07657e+004**	f_{13}	2.29771e+000	1.83736e+000	**2.46971e+000**
	Best		6.34127e+006	5.84183e+005	**1.76865e+003**		1.65305e+001	5.97308e+000	**5.77336e+000**
	Mean		4.79521e+003	4.69658e+003	**4.69724e+003**		1.35172e+001	1.33836e+001	**1.33256e+001**
	Std	f_7	1.31326e+002	1.57566e+000	**1.75879e+000**	f_{14}	2.48143e−001	1.64681e−001	**2.88264e−001**
	Best		4.69629e+003	4.69629e+003	**4.69634e+003**		1.29993e+001	1.30844e+001	**1.27715e+001**
	Mean		5.03297e+003	5.68434e−014	**1.53154e−004**		2.09298e+001	2.09193e+001	**2.09660e+001**
	Std	f_1	3.03181e+003	0.00000e+000	**4.18914e−004**	f_8	6.64300e−002	6.44776e−002	**5.27812e−002**
	Best		6.07129e+002	5.68434e−014	**6.41194e−010**		2.07949e+001	2.07742e+001	**2.08467e+001**
	Mean		1.96347e+003	2.54377e+002	**2.99441e+002**		6.16224e+001	8.13724e+001	**4.66076e+001**
	Std	f_2	2.01172e+003	9.34284e+001	**2.56881e+002**	f_9	1.71755e+001	1.68406e+001	**1.15194e+001**
	Best		2.43535e+002	8.90017e+001	**9.49609e+001**		2.76431e+001	5.45266e+001	**2.88538e+001**
	Mean		1.54834e+007	6.43300e+006	**6.28145e+006**		**1.34417e+002**	9.77850e+001	1.51357e+002
	Std	f_3	9.17658e+006	2.44953e+006	**3.20706e+006**	f_{10}	**4.97047e+001**	2.00968e+001	4.53854e+001
	Best		4.68044e+006	1.29791e+006	**1.26064e+006**		**7.13564e+001**	6.97065e+001	7.37419e+001
	Mean		4.00006e+003	8.32103e+003	**2.80629e+003**		**2.23226e+001**	2.76867e+001	2.52088e+001
1e+5	Std	f_4	3.98677e+003	2.60171e+003	**1.11333e+003**	f_{11}	**2.91847e+000**	2.37191e+000	2.74081e+000
	Best		1.06316e+003	2.65577e+003	**7.00163e+002**		**1.76669e+001**	2.23630e+001	1.97119e+001
	Mean		8.26812e+003	4.03671e+003	**4.00182e+003**		7.11653e+004	2.31141e+004	**2.90461e+004**
	Std	f_5	2.16989e+003	1.09741e+003	**8.50689e+002**	f_{12}	6.44392e+004	1.41319e+004	**1.41910e+004**
	Best		5.24738e+003	1.67555e+003	**1.98746e+003**		1.15463e+004	5.37377e+002	**9.18983e+003**
	Mean		5.85082e+008	5.61777e+001	**3.14410e+002**		4.19409e+000	4.30779e+000	**3.72728e+000**
	Std	f_6	5.47286e+008	5.09707e+001	**3.51614e+002**	f_{13}	1.76370e+000	8.55066e−001	**8.33247e−001**
	Best		3.17191e+007	3.69320e+000	**5.61478e+000**		1.89062e+000	2.84523e+000	**2.21216e+000**
	Mean		4.81549e+003	4.69752e+003	**4.69629e+003**		**1.27317e+001**	1.26418e+001	1.28308e+001
	Std	f_7	1.35471e+002	5.37612e+000	**2.42769e−012**	f_{14}	**3.60124e−001**	2.86245e−001	3.54577e−001
	Best		4.69629e+003	4.69629e+003	**4.69629e+003**		**1.16439e+001**	1.17959e+001	1.17271e+001
	Mean		3.26394e+003	5.68434e−014	**3.86209e−009**		**2.09111e+001**	**2.08954e+001**	2.09328e+001
	Std	f_1	1.54532e+003	0.00000e+000	**1.51146e−008**	f_8	**5.86618e−002**	**5.83142e−002**	5.36313e−002
	Best		6.79230e+002	5.68434e−014	**5.68434e−014**		**2.07410e+001**	**2.07997e+001**	2.07620e+001
	Mean		2.50007e+003	9.16847e−002	**4.82456e+001**		6.04269e+001	7.36202e+001	**4.63317e+001**
	Std	f_2	2.38445e+003	5.59774e−002	**1.05699e+002**	f_9	1.98466e+001	1.60303e+001	**1.25820e+001**
	Best		2.74046e+002	2.05889e−002	**3.75959e−001**		3.52026e+001	4.43526e+001	**1.89042e+001**
	Mean		2.26380e+007	2.73654e+006	**3.12054e+006**		**1.27756e+002**	8.77582e+001	1.30126e+002
	Std	f_3	2.50423e+007	1.70620e+006	**1.24667e+006**	f_{10}	**2.85435e+001**	1.30657e+001	5.56891e+001
	Best		5.57688e+006	1.23519e+006	**1.51001e+006**		**7.27676e+001**	6.66820e+001	4.46771e+001
	Mean		2.92579e+003	2.82301e+003	**7.19188e+002**		**2.07348e+001**	2.61465e+001	2.39575e+001
3e+5	Std	f_4	2.79145e+003	8.36743e+002	**3.81057e+002**	f_{11}	**3.58800e+000**	3.26704e+000	2.98696e+000
	Best		5.02410e+002	1.33677e+003	**1.63892e+002**		**1.44917e+001**	2.11965e+001	1.86127e+001
	Mean		7.61173e+003	3.75562e+003	**3.60649e+003**		7.35230e+004	1.45138e+004	**1.95793e+004**
	Std	f_5	1.99169e+003	9.95512e+002	**1.02481e+003**	f_{12}	3.74951e+004	6.92272e+003	**1.11956e+004**
	Best		3.29871e+003	2.26279e+003	**2.27419e+003**		1.47947e+004	5.60166e+003	**7.84640e+002**
	Mean		6.97906e+008	3.06231e+001	**7.87078e+001**		3.23641e+000	4.47266e+000	**2.83982e+000**
	Std	f_6	1.05009e+009	3.87801e+001	**6.58561e+001**	f_{13}	2.13593e+000	6.92963e−001	**7.82875e−001**
	Best		4.10872e+007	3.47640e−002	**3.50280e+000**		1.84510e+000	2.71106e+000	**1.54405e+000**
	Mean		4.85854e+003	4.69629e+003	**4.69629e+003**		**1.23318e+001**	1.21451e+001	1.25458e+001
	Std	f_7	1.61513e+002	3.03165e−013	**2.87008e−012**	f_{14}	**4.30478e−001**	3.86448e−001	3.02936e−001
	Best		4.69629e+003	4.69629e+003	**4.69629e+003**		**1.12582e+001**	1.13661e+001	1.20758e+001

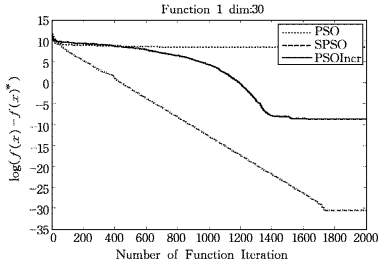

图 3　函数 f_1 的值与其最优值差的对数值随进化代数的变化

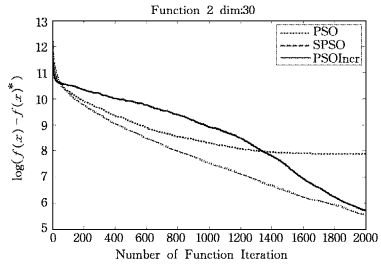

图 4　函数 f_2 的值与其最优值差的对数值随进化代数的变化

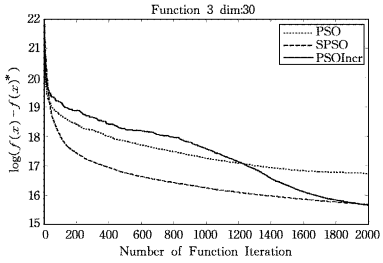

图 5　函数 f_3 的值与其最优值差的对数值随进化代数的变化

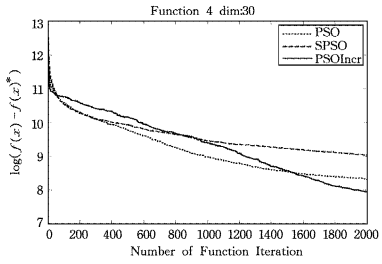

图 6　函数 f_4 的值与其最优值差的对数值随进化代数的变化

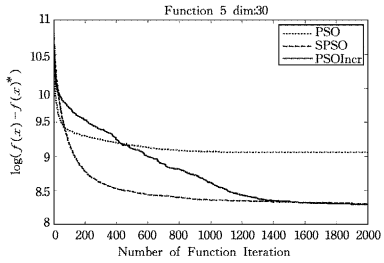

图 7　函数 f_5 的值与其最优值差的对数值随进化代数的变化

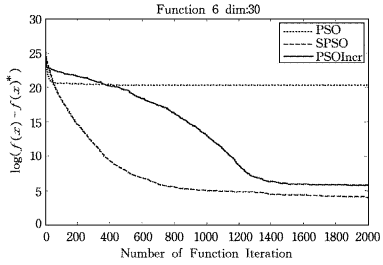

图 8　函数 f_6 的值与其最优值差的对数值随进化代数的变化

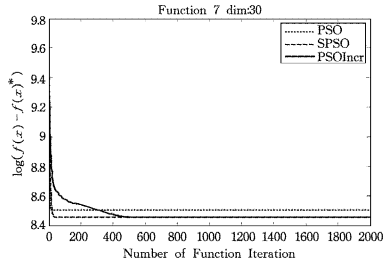

图 9　函数 f_7 的值与其最优值差的对数值随进化代数的变化

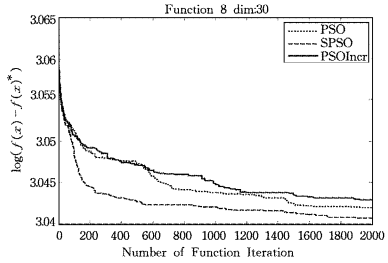

图 10　函数 f_8 的值与其最优值差的对数值随进化代数的变化

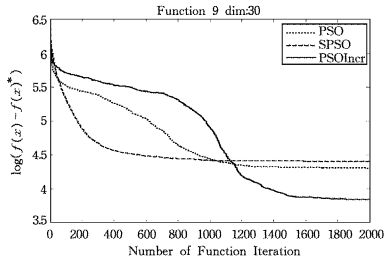

图 11　函数 f_9 的值与其最优值差的对数值随进化代数的变化

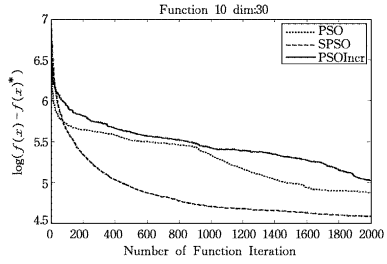

图 12　函数 f_{10} 的值与其最优值差的对数值随进化代数的变化

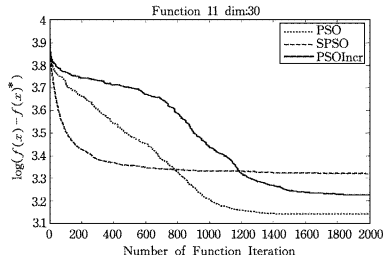

图 13　函数 f_{11} 的值与其最优值差的对数值随进化代数的变化

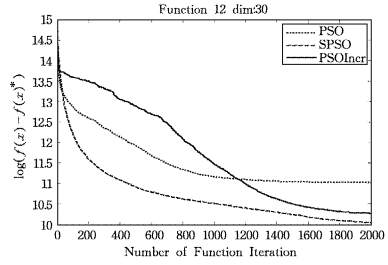

图 14　函数 f_{12} 的值与其最优值差的对数值随进化代数的变化

（下转第 84 页）

[4] 娄平，陈幼平，周祖德. 敏捷供应链中供应商选择的 AHP/DEA 方法[J]. 华中科技大学学报：自然科学版，2002，30(4)：33-35
[5] 刘开元，王蓉，金宝辉. 一种供应商的模糊评价模型及其实现[J]. 武汉理工大学学报，2004，28(3)：109-112
[6] 杨玉中，张强，吴立云. 基于熵权的 TOPSIS 供应商选择方法[J]. 北京理工大学学报，2006，26(1)：109-112
[7] Weber C A，Current J R，Benton W C. Vendor selection criteria and methods [J]. European Journal of Operational Research，1991，50：2-18
[8] 许树伯. 层次分析法原理[M]. 天津：天津大学出版社，2003
[9] 吴晓云，吴萍. 基于知识的层次分析法及其应用[J]. 南京理工大学学报，2005，29(4)：451-454
[10] 钟嘉鸣，李订芳. 粗糙集与层次分析法集成的综合评价模型[J]. 武汉大学学报：工学版，2008，41(4)：126-130
[11] Pawlak Z. Rough set theory and its applications to data analysis [J]. Cybernetics and Systems，1998，29(1)：661-688
[12] 刘清. Rough 集及 Rough 推理[M]. 北京：科学出版社，2001：11-15
[13] 张文修，吴伟志，梁吉业，等. 粗糙集理论与方法[M]. 北京：科学出版社，2001：30-36
[14] 王国胤. Rough 集理论与知识获取[M]. 西安：西安交通大学出版社，2001：25-45
[15] 苗夺谦，王国胤，刘清，等. 粒计算：过去、现在与展望[M]. 北京：科学出版社，2007：142-154

(上接第 65 页)

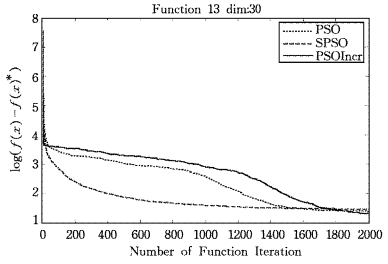

图 15　函数 f_{13}的值与其最优值差的对数值随进化代数的变化

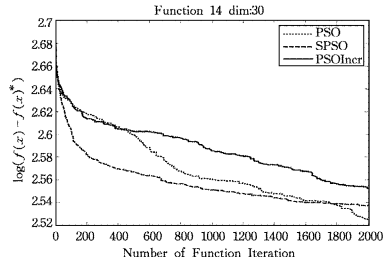

图 16　函数 f_{14}的值与其最优值差的对数值随进化代数的变化

结束语　粒子群算法是近几年来使用最频繁的基于群体的进化算法。由于使用方便且效果很好，PSO 算法已被广泛用于各种工程技术的优化问题。但其权重递减的模型一直以来是通过直观判断和实验验证以后加以推广应用的，其效果明显。

本文利用 PSO 算法的理论分析模型，分析了递减模型的基本原理，论证了权重递减的原由。同时在此基础上提出一种权重可以递增的 PSO 算法模型。通过基准函数测试，验证了递增模型的效果要优于递减模型的效果。本文给 PSO 算法的权重理论提出一个新颖的视角，为 PSO 算法权重选择提出一种新方法。

参 考 文 献

[1] Kennedy J，Eberhart R. Particle Swarm Optimization[C]// Proceeding of IEEE International Conference on Neural Networks. Piscataway，NJ：IEEE CS，1995：1942-1948
[2] Eberhart R，Kennedy J. A new optimizer using particle swarm theory[C]// Proceeding of the 6th International Symposium on Micro Machine and Human Science. 1995：39-43
[3] Shi Y，Eberhart R C. Fuzzy adaptive particle swarm optimization [C]// Proceedings of the 2001 IEEE Congress on Evolutionary Computation. 2001，1：101-106
[4] Rechenberg I. Evolutions strategie：Optimierung technischer Systeme nach Prinzipien derbiologischen Evolution[C]// Frommann-Holzboog. Stuttgart，1973
[5] Kirkpatrick S，Gelatt C D，Vecchi M P. Optimization by simulated annealing[J]. Science，1983，220：671-680
[6] Shi Y，Eberhart R. A Modified Particle Swarm Optimizer[C]// Proc. IEEE World Congr. Comput. Intell. 1998：69-73
[7] Shi Y，Eberhart R C. Empirical study of particle swarm optimization[C]// Proc. IEEE Congr. Evol. Computer. 1999：1945-1950
[8] Nickabadi A，Ebadzadeh M M，Safabakhsh R. A novel particle swarm optimization algorithm with adaptive inertia weight[J]. Applied Soft Computing，2011，11(4)：3658-3670
[9] Chatterjee A，Siarry P. Nonlinear inertia weight variation for dynamic adaptionin particle swarm optimization[J]. Computer and Operations Research，2006，33：859-871
[10] Ismail A，Engelbrecht A. The Self-adaptive Comprehensive Learning Particle Swarm Optimizer[J]. Swarm Intelligence(Lecture Note in Computer Science)，2012，7461：156-167
[11] 刘建华，樊晓平，瞿志华. 一种惯性权重动态调整的新型粒子群算法 [J]. 计算机工程与应用，2007，43(7)：68-70
[12] Van de Bergh F. An Analysis of Particle Swarm Optimizer[D]. University of Pretoria，2002
[13] Van den Bergh F，Engelbrecht A P. A study of particle swarm optimization particle trajectories [J]. Information sciences，2006，176(8)：937-971
[14] 刘建华，刘国买，杨荣华，等. 粒子群算法的交互性与随机性分析[J]. 自动化学报，2012，38(9)：1471-1484
[15] Bratton D，Kennedy J. Defining a standard for particle swarm optimization[C]// Swarm Intelligence Symposium，SIS 2007. IEEE，2007：120-127
[16] Suganthan P N，Hansen N，Liang J J，et al. Problem definitions and evalua-tion criteria for the cec 2005 special session on real-parameter optimization[C]// 2005 IEEE Congress on Evolution Computation(CEC). 2005：1-15