基于高斯函数递减惯性权重的粒子群优化算法

张 迅, 王 平, 邢建春, 杨启亮
(解放军理工大学, 南京 210007)

摘 要: 为了有效地平衡粒子群优化算法的全局搜索和局部搜索能力,提出了一种基于高斯函数递减惯性权重的粒子群优化(GDIWPSO)算法。此算法利用高斯函数的分布性、局部性等特点,实现了对惯性权重的非线性调整。仿真过程中,首先对测试函数优化以确定惯性权重的递减方式;然后比较了该算法与权重线性递减、凸函数递减、凹函数递减的粒子群算法优化不同测试函数的性能;最后结果表明,提出的算法在搜索能力、收敛速度及执行效率等方面均有很大提高。

关键词: 粒子群优化; 高斯函数; 惯性权重; 收敛速度; 执行效率
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2012)10-3710-03
doi:10.3969/j.issn.1001-3695.2012.10.027

Particle swarm optimization algorithms with decreasing inertia weight based on Gaussian function

ZHANG Xun, WANG Ping, XING Jian-chun, YANG Qi-liang
(*PLA University of Science & Technology*, *Nanjing* 210007, *China*)

Abstract: To efficiently balance the global search and local search ability, this paper presented a particle swarm optimization (PSO) algorithm with decreasing inertia weight based on Gaussian funtion(GDIWPSO), this algorithm took advantage of the distribution and locality property of Gaussian function to implement nonlinear inertia weight adjustment. In simulation experiment, optimizing the benchmark function to determine the strategy of decreasing inertia weight and comparing the performance with weight of linear decreasing, convex function decreasing and concave function decreasing. The stimulation results show that the proposed PSO algorithm has better improvement in search ability, convergence rate and computation efficiency.
Key words: particle swarm optimization; Gaussian function; inertia weight; convergence rate; computation efficiency

粒子群优化算法是一种基于群体智能的进化计算技术,其思想来源于人工生命和进化计算理论,最早是由美国的 Kennedy 等人[1]受鸟群觅食行为的启发提出的。该算法与遗传算法、进化规划等进化算法相比,具有易于实现、调节参数少、收敛速度快的优点,因此,既适合于科学研究,又特别适合于工程应用[2]。尽管粒子群优化算法有很多优势,实际上对算法的研究发现,粒子群优化算法早期收敛速度较快,但到寻优的后期,算法可能会陷入局部最优。因此,Shi 等人[3]将惯性权重引入到粒子群优化算法中,并通过实验指出,较大的惯性权重有利于全局寻优,较小的惯性权重有利于局部寻优。为了平衡算法的全局与局部搜索能力,目前,很多学者提出了对惯性权重的改进策略,例如文献[4]提出了一种线性递减惯性权重的粒子群(LDIWPSO)算法,其中,惯性权重是迭代次数的函数且沿直线线性递减;Yadmellat 等人[5]用模糊逻辑规则调整惯性权重,提出了一种模糊惯性权重的粒子群(FIPSO)算法;黄轩等人[6]在对惯性权重调查研究的基础上,提出了一种惯性权重随机均匀分布的粒子群优化算法;Yang 等人[7]引入混沌逻辑映射的方法调整惯性权重,提出了一种新型混沌优化惯性权重的粒子群优化(NCIWPSO)算法;任子晖等人[8]提出了一种动态改变惯性权重的粒子群(DCWPSO)算法;杜振鑫等人[9]在综合考虑影响惯性权重因素的基础上,提出了一种改进的动态改变惯性权重的粒子群算法。本文针对标准粒子群算法局部寻优的缺点,进行惯性权重以不同方式递减影响粒子群局部搜索能力的研究,提出了一种基于高斯函数递减惯性权重的粒子群优化(GDIWPSO)算法。

1 标准粒子群算法(SPSO)

在标准粒子群算法中,每个粒子的位置代表优化问题在 n 维空间中的潜在解。设种群中每个粒子在 n 维空间中搜索最优值,用 $X_i=(X_{i1},X_{i2},\cdots,X_{in})$ 来表示第 i 个粒子的位置,其中,$X_{id}\in[l_d,u_d]$,$d\in[1,n]$,l_d、u_d 是 d 维空间的上、下限。其速度为 $V_i=(V_{i1},V_{i2},\cdots,V_{in})$,用来控制粒子的飞行方向和距离,它受最大速率 $V_{\max}$ 的限制,$V_{\max}$ 的设定视不同情况而定。第 i 个粒子经历的个体最好位置为 P_i,当前组成群体的所有粒子经历的最好位置为 P_g。粒子在找到上述两个极值后,就根据式(1)和(2)来更新自己的速度与位置。

$$V_{id}^{t+1}=wV_{id}^{t}+c_1\text{rand}()(P_{id}-X_{id}^{t})+c_2\text{rand}()(P_{gd}-X_{id}^{t}) \quad (1)$$

$$X_{id}^{t+1}=X_{id}^{t}+V_{id}^{t+1} \quad (2)$$

收稿日期: 2011-12-15; **修回日期**: 2012-02-14

作者简介: 张迅(1987-),男,湖北襄阳人,硕士,主要研究方向为智能结构(xunzhang893@163.com);王平(1970-),男,四川武胜人,副教授,硕导,主要研究方向为复杂智能系统;邢建春(1964-),男,河北正定人,教授,博导,主要研究方向为复杂智能系统;杨启亮(1977-),男,河南信阳人,副教授,博士,主要研究方向为计算机软件理论.

其中:w 为惯性权重;c_1 和 c_2 为学习因子或加速常数,通常取为 2;rand( ) 为介于(0,1)之间的随机数;$V_{id}{}^t$ 和 $X_{id}{}^t$ 分别为粒子 i 在第 t 次迭代中第 d 维的速度和位置。

2　基于高斯函数递减惯性权重的 PSO 算法

2.1　设计思想

在统计学与概率论中,高斯函数是正态分布的密度函数。其表达式为

$$f(x) = ae^{-(x-b)^2/c^2} \tag{3}$$

其中:a、b 与 c 为实常数,c 称为高斯函数的宽度或扩展常数,其值越大,函数曲线就越平坦;其值越小,函数曲线就越陡峭。

文献[10]指出,当惯性权重服从正态分布时,能够提高算法的全局搜索能力,但该文中并未给出惯性权重具体的分布方式。受此启发,本文根据高斯函数的分布特性,把相应的参数代入其中,实现对 w 的非线性映射,从而提出了一种基于高斯函数递减惯性权重的调整策略,即

$$w(t) = (w_{\max} - w_{\min}) \exp\left[-\frac{t^2}{(k \times \mathrm{iter}_{\max})^2} \right] + w_{\min} \tag{4}$$

其中:k 为一常数,调整 k 值就是改变曲线的扩展常数,从而改变曲线的变化率。图 1 中给出了不同的 k 值对应的惯性权重 w 的变化曲线。

由图 1 可以看出,w 的变化曲线近似服从正态分布,当 $k > 0.3$ 时,惯性权重在整个迭代过程中的变化相对平缓;当 $k < 0.3$ 时,惯性权重 w 在迭代初期取值较大,而且递减的速度很快,迭代后期 w 几乎不变,表现出一定的局部特性。为利用 w 的这一特性,本文将 k 的取值设定为[0.1,0.3],并在编写程序时进行如下改进:

$$\text{if } w - w_{\min} \geqslant T\text{value then } w = w\text{;else } w = w_{\min} \tag{5}$$

其中:Tvalue 为计算 w 的停止阈值,通过比较 w、$w_{\min}$ 之差与 Tvalue 的大小,从而确定 w 是否进行调整。容易看出,如果 Tvalue 取值较小,按式(5)的方法来计算 w 的值不会改变其递减趋势,而且还可以减少在迭代过程中惯性权重 w 的计算频度,提高算法的执行效率。下面分析进行上述改进后算法中计算惯性权重 w 的时间复杂度。

假设程序循环执行 M 次,每循环一次,算法进行 T 次迭代。若不采用上述改进策略,则循环结束后,算法计算 w 的时间复杂度为 $O(M \times T)$;若采用改进策略,当惯性权重 w 与 $w_{\min}$ 之差小于设定阈值时,就不再进行权重的计算,而是保持其值不变。由于改进后 w 的计算次数为一个常数,所以算法计算 w 的时间复杂度为 $O(1)$。由此可见,改进后的算法计算惯性权重的时间复杂度有很大的改善。当然,由于程序中计算权重的语句在整个算法中所占的比重较小,当循环次数及迭代次数较少时,算法执行效率的改善并不明显,但对于需要多次循环迭代的复杂优化计算问题,改进算法对执行效率的改善就不容忽视了。

2.2　算法结构

通过前面的分析可知,本文提出的基于高斯函数递减惯性权重的粒子群算法的结构如下:

```
procedure GDIWPSO
begin
初始化粒子,设定阈值 Tvalue
while (终止条件未满足时) do
for i = 1 to N do
if f(X_i) < f(P_i) then P_i = X_i
P_g = min(Pneighbors)
for d = 1 to D
if w - w_min < Tvalue then w = w_min
else if w - w_min ≥ Tvalue then 按式(4)调整 w
按式(1)更新粒子速度
if (V_i > V_max) then V_i = V_max
else if (V_i < -V_max) then V_i = -V_max
X_i = X_i + V_i
end for
end for
end while
end begin
```

其中:N 表示群体规模;D 表示搜索空间维数。初始化粒子包括对粒子群中每个粒子的位置、速度和个体极值初始化,Pneighbors 表示粒子的邻域,函数 $f(\cdot)$ 计算粒子适值(即其位置对应的函数值),X_i、P_i、P_g 与式(1)中相同。

3　仿真实验

3.1　测试函数与参数设置

优化算法的性能比较通常是基于典型的测试函数展开的,本文采用了以下四个常用的测试函数进行算法验证:

a) DeJong 函数

$$f_1(x) = \sum_{i=1}^{D} x_i^2$$

b) Rastrigrin 函数

$$f_2(x) = \sum_{i=1}^{D} [x_i^2 - 10 \times \cos(2\pi x_i) + 10]$$

c) Rosenbrock 函数

$$f_3(x) = \sum_{i=1}^{D} [100 \times (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

d) Griewank 函数

$$f_4(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

采用粒子群算法优化测试函数的参数具体设置为:a) $c_1 = c_2 = 2$;b) 粒子群的规模 N 取为 20 ~ 40,搜索维数 D 取为 10 ~ 20 维;c) 最大迭代次数 $\mathrm{iter}_{\max}$ 取为 1000 ~ 1500,每个实验循环次数 M 取为 50,最终结果是计算 50 次循环得出最优适值的平均值;d) w 从 0.9 递减到 0.4,即 $w_{\max} = 0.9$,$w_{\min} = 0.4$。

测试函数的变化区间、最大速度 $V_{\max}$ 以及期望的到的目标值的设置如表 1 所示。

表 1　测试函数的参数设置

函数名称	维数	$[X_{\max}, X_{\min}]$	$V_{\max}$	目标值
DeJong	10 ~ 20	[-100,100]	100	1e-3
Rastrigrin	10 ~ 20	[-10,10]	10	100
Rosenbrock	10 ~ 20	[-30,30]	30	100
Griewank	10 ~ 20	[-600,600]	600	0.1

3.2　实验步骤与结果

本文的仿真实验主要分为两部分:a) 参数 k 的选择实验,即用本文提出的基于高斯函数递减惯性权重的粒子群算法对测试函数进行优化,通过对优化结果的分析来选择设定最合适的 k 值,以确定惯性权重 w 的变化趋势;b) 不同算法性能的比较实验,即用本文已设定好 k 值的算法与惯性权重线性递减及文献[11]提出的以凸函数递减、凹函数递减的粒子群算法分别对不同的测试函数进行优化,比较每种算法的优

化结果,从而分析不同方式递减的惯性权重的粒子群优化算法的性能。

3.2.1 参数 k 的选择实验

通过对 Griewank 及 Rastrigrin 测试函数的优化来选择方程式(4)中合适的 k 值。算法中惯性权重 w 的停止阈值 tvalue 取为 0.001,循环次数设为 50 次(其他参数参照 3.1 节设置)。采用文中算法分别优化上述两个函数得到的最优适值(best)、平均适值(mean)及标准差(std)如表 2、3 所示。

表 2 k 取不同值时 Griewank 函数的优化结果

参数	适值	$k=0.1$	$k=0.15$	$k=0.2$	$k=0.25$	$k=0.3$
$N=20$ $D=10$ iter = 1000	best	0.012 3	0.034 5	0.012 3	0.007 4	0.014 8
	mean	0.096 3	0.119 1	0.095 3	0.112 1	0.086 1
	std	0.053 7	0.053 3	0.046 5	0.056 7	0.046 4
$N=20$ $D=20$ iter = 1500	best	0	0	0	0	0
	mean	0.029 5	0.029 8	0.023 8	0.023 9	0.027 8
	std	0.029 8	0.042 6	0.024 1	0.026 1	0.024 6
$N=40$ $D=10$ iter = 1000	best	0.019 7	0.009 9	0.014 8	0.022 1	0.019 7
	mean	0.086 0	0.089 6	0.075 3	0.079 1	0.081 5
	std	0.037 9	0.036 9	0.031 7	0.032 4	0.043 6
$N=40$ $D=20$ iter = 1500	best	0	0	0	0	0
	mean	0.023 8	0.025 1	0.011 0	0.023 6	0.013 5
	std	0.017 7	0.023 5	0.015 1	0.023 9	0.015 6

表 3 k 取不同值时 Rastrigrin 函数的优化结果

参数	适值	$k=0.1$	$k=0.15$	$k=0.2$	$k=0.25$	$k=0.3$
$N=20$ $D=10$ iter = 1000	best	1.989	1.989	0.994 9	0.994 9	0.994 9
	mean	6.595 8	6.387 7	5.625 1	6.606 5	5.790 7
	std	3.423 4	3.429 5	2.820 3	3.644 7	2.861 8
$N=20$ $D=20$ iter = 1500	best	11.94	13.929	9.949 6	13.929	12.934
	mean	31.003	32.343	28.22	29.411	29.362
	std	9.716	10.78	8.938	8.653 2	8.713 5
$N=40$ $D=10$ iter = 1000	best	0	0	0.994 6	0.994 9	0
	mean	5.055	4.616 7	4.338 3	4.497 2	4.397 9
	std	2.341 7	2.532 7	2.049 9	2.267 4	2.184 4
$N=40$ $D=20$ iter = 1500	best	11.94	11.939	9.949 6	8.954 6	4.974 8
	mean	23.72	21.63	20.439	20.237	20.417
	std	7.367 4	5.448 9	5.673 1	6.543 3	6.652 4

表 2、3 中分别给出了本文提出的算法在不同种群规模、搜索空间维数及迭代次数下的四组优化结果,比较表中这四组优化结果可以看出,当 $k=0.2$ 时,该算法对 Griewank 及 Rastrigrin 函数的总体优化结果较小,为获得较好的性能,本文将 k 值取为 0.2。

3.2.2 不同算法性能比较实验

经上一步实验选定参数 k 的值后,w 的变化趋势就确定了,接下来就是对测试函数优化以验证算法性能。为了进行对比,分别将惯性权重 w 线性递减、凸函数递减、凹函数递减以及本文提出的基于高斯函数递减的粒子群算法用于不同测试函数的优化。这里将上述几种算法分别用 w_0-PSO、w_1-PSO、w_2-PSO、w_3-PSO 表示,这四种算法的惯性权重 w 的递减曲线如图 2 所示。仿真实验中,每种算法都循环执行 50 次(其他参数按 3.1 节进行设置)。采用这四种算法对每个测试函数优化所得到的最优适值(best)、平均适值(mean)、标准差(std)、优化成功率(suc)(即优化结果达到目标值的比例)及算法执行时间(tim)如表 4 所示。当 $N=40$,$D=20$,iter = 1500 时,这四种算法优化测试函数得到的平均最优适值随迭代次数的变化曲线如图 3 ~ 5 所示。

表 4 四种优化算法的比较结果

f	参数	结果	w_0-PSO	w_1-PSO	w_2-PSO	w_3-PSO
f_1	$N=40$ $D=20$ iter = 1500	best	1.57e-13	4.42e-23	6.66e-10	1.09e-39
		mean	6.75e-10	5.81e-18	1.466e-7	1.50e-34
		std	1.464e-9	1.61e-17	1.930e-7	4.99e-34
		suc/%	100	100	100	100
		time/s	9.235	9.296	9.282	9.047
f_2	$N=40$ $D=20$ iter = 1500	best	6.965 6	6.964 7	5.983 5	9.949 6
		mean	17.690	18.831	17.275	20.439
		std	4.923 8	6.754 0	6.777 3	5.673 1
		suc/%	100	100	100	100
		time/s	13.250	13.078	13.125	12.485
f_3	$N=40$ $D=20$ iter = 150 0	best	0.081 9	1.724 1	2.737 4	0.028 7
		mean	58.927	39.234	104.24	16.213
		std	95.465	59.009	186.67	20.674
		suc/%	80	88	78	96
		time/s	10.438	12.219	15.015	10.285

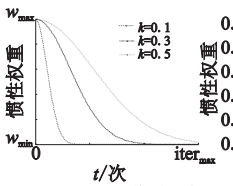

图1 k取不同值时w随迭代次数的变化曲线

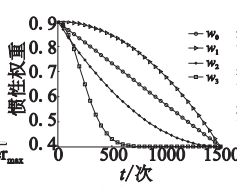

图2 惯性权重不同的变化方式

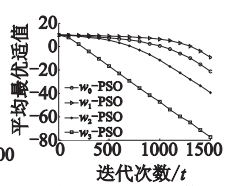

图3 DeJong函数

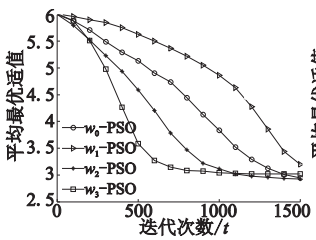

图4 Rastrigrin函数

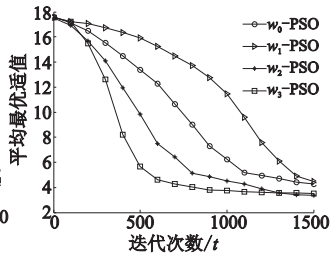

图5 Rosenbrock函数

4 讨论

对比表 4 中四个测试函数的优化结果可以得出:

a)从单峰值 DeJong 函数的优化结果可以看出,四种算法对其优化的成功率(达到期望值所占的比例)均相同,但用 w_3-PSO算法得到的最优适值、平均适值及标准差要远小于其他几种算法,且算法执行耗时最短;采用 w_1-PSO 算法的优化结果最差,算法执行耗时最长。

b)比较多峰值 Rastrigrin 函数的优化结果,四种算法总体的优化结果相差不大,采用 w_2-PSO 算法的优化结果稍好,w_1-PSO算法的优化结果较差,使用 w_3-PSO 算法所耗时间最短,效率最高。从算法的优化结果可以看出,几种算法的性能各有优势,单从优化结果难以比较算法的性能。

c)从单峰值 Rosenbrock 函数的优化结果可以看出,w_3-PSO算法对其优化的成功率最高,且算法执行时间最短,性能最高;采用 w_1-PSO 算法却与之相反,性能最差。

对比图 3 ~ 5 中四种算法所得平均最优适值随迭代次数的收敛曲线可以得出:

a)对于单峰值 DeJong 和 Rosenbrock 函数的优化,w_3-PSO 算法不仅收敛的最优值小,而且收敛速度快,w_2-PSO 算法的收敛速度次之,w_1-PSO 算法的收敛速度最差。

b)比较多峰值 Rastrigrin 函数的优化曲线,虽然 w_3-PSO 算法收敛的最优值比 w_1-PSO 算法和 w_0-PSO 算法的最优值要稍大,但其收敛速度比其他几种算法的收敛速度明显要快。

(下转第 3724 页)

(上接第 3712 页)

综合比较上述四种算法的优化结果和平均最优适值的收敛曲线可以看出,对于单峰值函数的优化,本文提出的基于高斯函数递减惯性权重的粒子群算法,其搜索能力、收敛速度及执行效率均有明显改善;对于多峰值函数的优化,本文提出的算法其搜索性能改善不明显,但其执行效率和收敛速度要优于其他几种算法。

5 结束语

本文提出了一种基于高斯函数递减惯性权重的粒子群优化算法,并通过对四个典型测试函数的优化,比较了该算法与三种不同递减惯性权重粒子群优化算法的性能,结果表明本文提出的算法在搜索能力、收敛速度与执行效率上有很大改善,特别是单峰函数,其算法性能的改善更为明显。

然而本文提出的算法也存在一些不足,比如只对一个多峰值函数进行了优化测试,对其优化精度的改善并不明显;另外算法中 k 值的选择方法并不明确。下一步将继续研究用本文算法优化更复杂的多峰值函数及其 k 值合理的选择方法。

参考文献:

[1] KENNEDY J, EBERHART R C. Particle swarm optimization[C]// Proc of IEEE International Conference on Neural Networks. New York: IEEE Press,1995:1942-1948.

[2] POLI R. Analysis of the publications on the application of particle swarm optimization [J]. Journal of Artificial Evolution and Applications,2008,8(2):4.

[3] SHI Yu-hui, EBERHART R C. A modified particle swarm optimizer [C]//Proc of IEEE Congress on Evolutionary Computation. New York: IEEE Press,1998: 69-73.

[4] SHI Yu-hui, EBERHART R C. Empirical study of particle swarm optimization [C]//Proc of Congress on Evolutionary Computation. Washington DC: IEEE Press,1999:1945-1950.

[5] YADMELLAT P, SALEHIZADEH S M A, MENHAJ M B. A new fuzzy inertia weight particle swarm optimization[C]//Proc of International Conference on Computational Intelligent and Natural Computing. 2009:507-510.

[6] 黄轩,张军,詹志辉. 基于随机惯性权重的快速粒子群优化算法[J]. 计算机工程与设计,2009,30(3):647-650.

[7] YANG Cheng-hong, CHENG Y H, CHUANG L Y. A novel chaotic inertia weight particle swarm optimization for PCR primer design [C]//Proc of International Conference on Technologies and Applications of Artificial Intelligence. 2010:373-378.

[8] 任子晖,王坚. 一种动态改变惯性权重的自适应粒子群算法[J]. 计算机科学,2009,36(2):226-229.

[9] 杜振鑫,王兆青. 一种改进的动态改变惯性权重的粒子群算法[J]. 微电子学与计算机,2011,28(3):85-88.

[10] 胡建秀,曾建潮. 具有随机惯性权重的 PSO 算法[J]. 计算机仿真,2006,23(8):164-167.

[11] 陈贵敏,贾建援,韩琪. 粒子群算法的惯性权值递减策略研究[J]. 西安交通大学学报,2006,40(1):53-61.