文章编号:1000－2367(2017)02－0108－11 DOI:10.16366/j.cnki.1000－2367.2017.02.020

基于模糊推理的粒子群优化算法

史旭栋[1],高岳林[2],韩俊茹[1]

(1.宁夏大学 数学统计学院,银川 750021;2.北方民族大学 信息与系统科学研究所,银川 750021)

摘 要:针对无约束优化问题,提出了基于模糊推理的粒子群优化算法,该算法针对粒子群优化算法搜索能力的不足,先引入平均粒子,然后引入模糊推理来改进粒子群的速度更新公式,再利用模糊推理动态地改进算法惯性权重和速度更新公式的权重因子,再结合混沌扰动增加算法后期的局部搜索能力.数值试验采用 12 个测试函数并有 5 个算法进行对比,数值试验证明,改进算法的搜索能力有较大的提高.

关键词:粒子群优化;模糊推理;信息共享

中图分类号:TP18 **文献标志码**:A

群体智能(swarm intelligence SI)是一种自发的、系统的群体行为,它模仿鱼群,鸟群等动物[1]的社会行为来实现一种简单的优化算法.群体智能是从生物系统中得到灵感,它的各种进化算法在过去十年间得到较快的发展.其中,在 1995 年,文献[2]提出的粒子群优化算法(Particle Swarm Optimization PSO),因为概念简单且能快速收敛到目标函数[3－5]最优解等优点,所以成为最受欢迎的优化算法之一.又由于 PSO 的稳定性和实用性,常被广泛应用于处理很多复杂的工程问题[6－11],为了提高 PSO 的搜索能力,文献[12]等从亚种群中选择一个领域粒子来改进速度更新公式来扩大搜索范围.另外文献[13]等将中心粒子融入 PSO 使粒子更加接近全局最优解.文献[14]提出了带有权重粒子的改进 PSO,另外文献[15]提出了结合模糊推理和加权粒子的混合粒子群优化算法.鸟群算法是在 2015 年被文献[16]提出的一种新的优化算法,它从鸟类的捕食、飞行中得到启发.由于鸟群算法的信息共享比较好,所以能较好地解决高维复杂问题.

近年来,粒子群优化算法已成为进化计算和群体智能等领域研究的热点之一.与其他智能算法类似,粒子群优化算法也存在早熟收敛和局部寻优能力差等缺点.目前解决这些问题的主要方法是增加种群局部搜索能力,本文主要是通过引入模糊推理来解决该问题.

1 粒子群优化算法的基本概念以及模糊推理的简述

1.1 标准粒子优化算法

粒子群优化算法起源于人们对鸟群捕食的研究,从鸟群捕食的模型当中得到启发并将其用来求解优化问题,在粒子群优化算法的模型中把每一个寻找食物的鸟对应于搜索空间中的一个粒子,每个粒子飞翔的方向和距离是由它的速度决定的.粒子群优化算法从一些随机产生粒子开始,然后粒子们向最优粒子学习,直到达到终止条件.

假设在一个 d 维的搜索空间中,有 M 个粒子和各自所处的位置以及速度.其中这 M 个粒子所组成的种群为 $x=[x_1,x_2,\cdots,x_M]$.它们的位置可以表示为 $x_i=[x_{i1},x_{i2},\cdots,x_{iD}]$.速度可以表示为 $v_i=[v_{i1},v_{i2},\cdots,v_{iD}]$.根据粒子们的飞行经验 $pb_i=[p_{i1},p_{i2},\cdots,p_{iD}]$ 和领导飞行粒子的经验.PSO 的搜索行为可以用公式(1)和(2)来描述:

收稿日期:2016-12-14;**修回日期**:2017-02-17.

基金项目:国家自然科学基金(61561001);北方民族大学重点科研项目(2015KJ10).

作者简介(通信作者):高岳林(1963－),男,陕西榆林人,北方民族大学教授,博士生导师,主要研究方向为最优化理论方法及应用,智能计算与智能信息处理,E-mail:gaoyuelin@263.net.

$$v_{id}(t+1) = w \times v_{id}(t) + c_1 \times R \times (pb_{id} - x_{id}(t)) + c_2 \times R \times (gb_{id} - x_{id}(t)), \tag{1}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1), \tag{2}$$

式中 t 为当前的迭代次数;R 为 0 与 1 之间的随机数,c_1,c_2 是学习因子;w 为惯性权重,一般用(3)式表示:

$$w = w_{\max} - (w_{\max} - w_{\min}) \times t / i_{t\max}, \tag{3}$$

其中 $w_{\max}$,$w_{\min}$ 分别为惯性权重的最大值和最小值;$i_{t\max}$ 为最大迭代次数.

粒子速度的变化区间为 $v_{id} \in [v_{i\min}, v_{i\max}]$,粒子位置的变化区间为 $x_{id} \in [x_{i\min}, x_{i\max}]$.

1.2　模糊推理

本文利用文献[17]中的经典模糊推理,其形式如下:如果 θ_1 是 A_1^l,且如果 θ_2 是 A_2^l,…,且如果 θ_k 是 A_k^l,则 y 是 B^l,其中,θ_k 是模糊输入变量,A_1^l 和 B^l 分别为前隶属度函数和后隶属度函数.l 为规则序号,输出 y 为:

$$y = \sum_{i=1}^{l} \left(B^i \prod_{j=1}^{k} A_j^i(\theta_j)\right) / \sum_{i=1}^{l} \left(\prod_{j=1}^{k} A_j^i(\theta_j)\right). \tag{4}$$

2　一种改进的粒子群优化算法

2.1　对自我认知项的讨论

经数值试验发现,对于标准粒子群的速度更新公式(1),其中的第二项(自我认知部分)对整个算法影响不明显.下面用图 1～12 说明这个问题,其中图中 F_1～F_{12} 分别是有无自我认知项对 12 个测试函数的影响.从图中 1～12 可知,自我认知项对算法的影响不大.

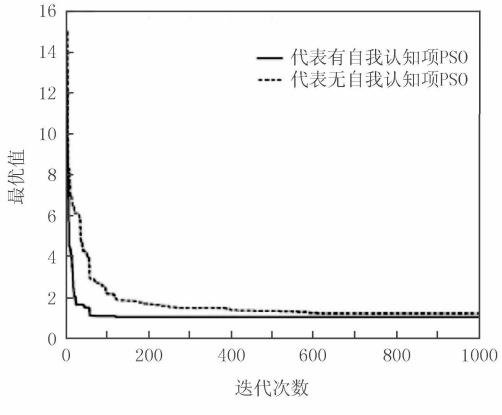

图1　有无自我认知项对F_1的影响

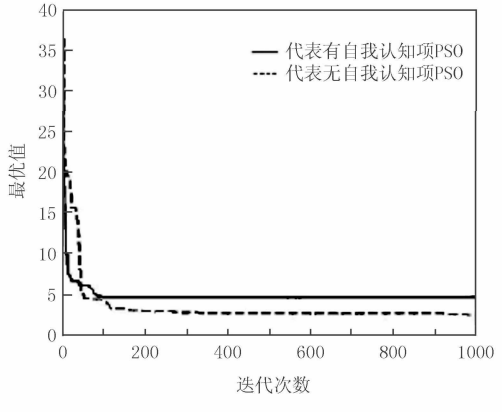

图2　有无自我认知项对F_2的影响

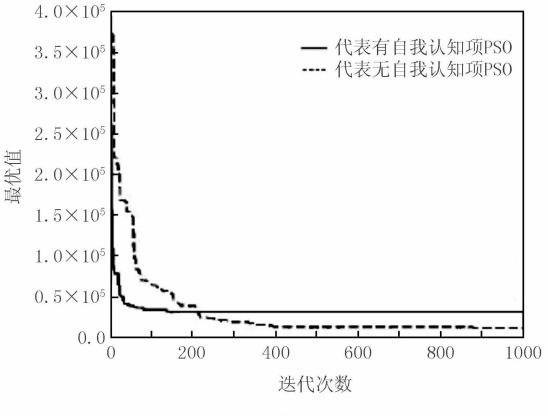

图3　有无自我认知项对F_3的影响

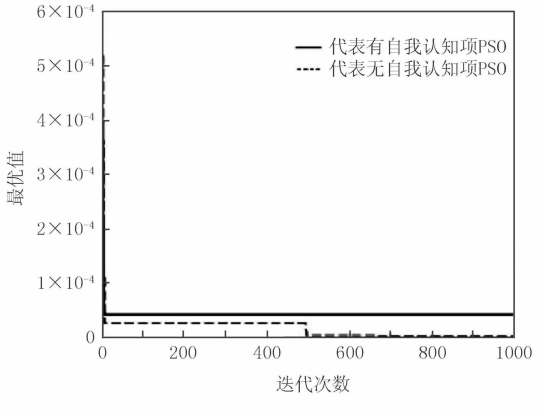

图4　有无自我认知项对F_4的影响

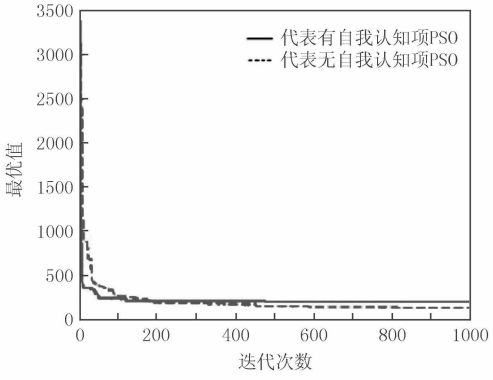

图5 有无自我认知项对F_5的影响

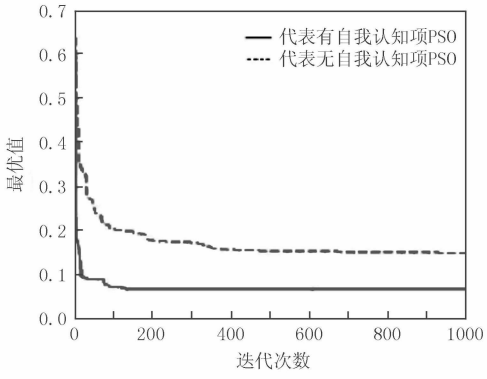

图6 有无自我认知项对F_6的影响

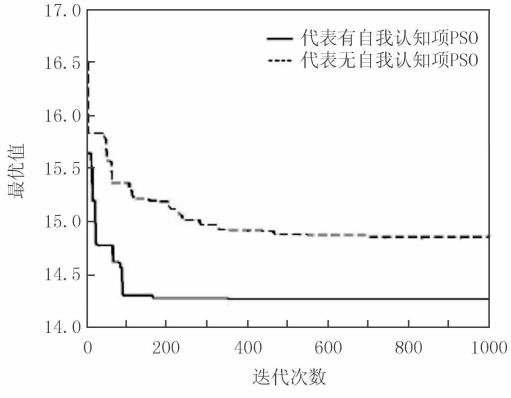

图7 有无自我认知项对F_7的影响

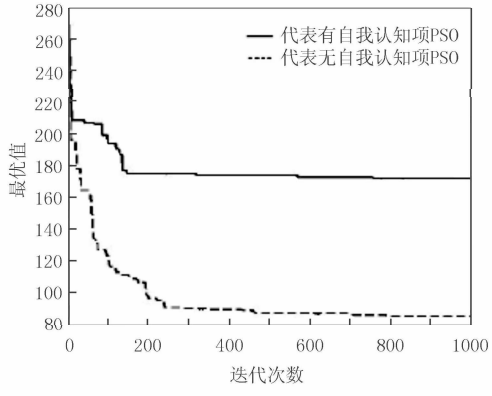

图8 有无自我认知项对F_8的影响

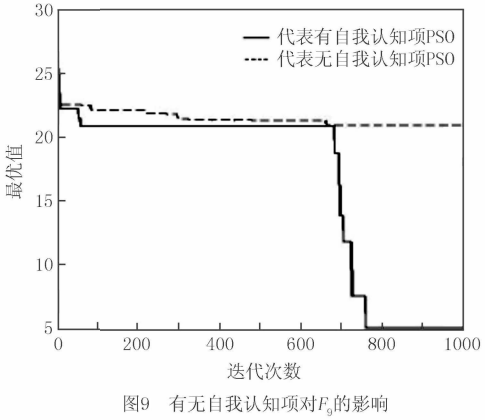

图9 有无自我认知项对F_9的影响

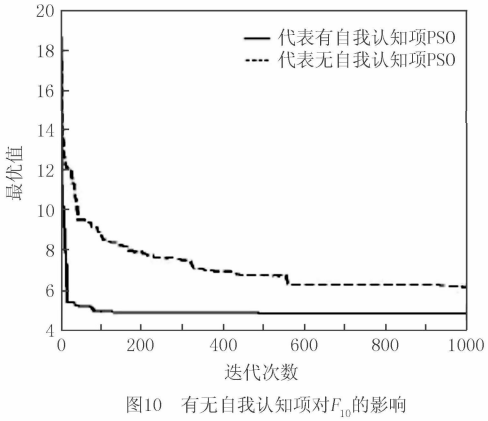

图10 有无自我认知项对F_{10}的影响

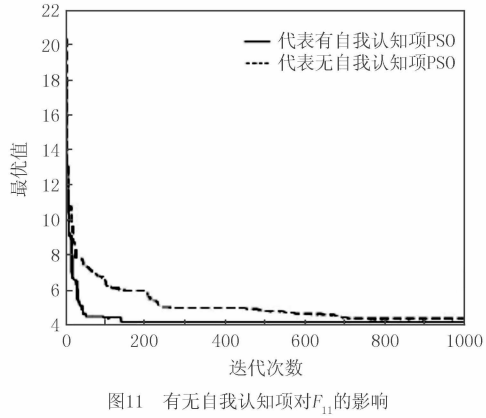

图11 有无自我认知项对F_{11}的影响

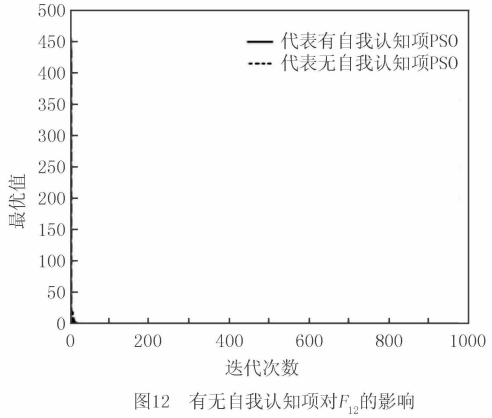

图12 有无自我认知项对F_{12}的影响

2.2 速度和位置更新公式的改进

下面将速度与位置更新公式改为如下形式

$$v_{id}(t+1) = wv_{id}(t) + \phi_1(M \times m^e_{id} - x_{id}(t)) + \phi_2(gb_{id} - M \times m^e_{id}) + \phi_3(x^w_d(t) - M \times m^e_{id}), \tag{5}$$

$$x_{id}(t+1) = x_{id}(t) + \mu_1\phi_4(x^w_d(t) - x_{id}(t)) + \mu_2 v_{id}(t+1), \tag{6}$$

其中,$\phi_1 = (\phi_2 + \phi_3) \cdot (1-R)$,$\phi_2 = c_2 \cdot R$,$\phi_3 = c_3 \cdot R$,$\phi_4 = c_4 \cdot R$,这里 c_2,c_3,c_4 是一个正常数,M 是一个随机变化的非负数,$\mu_1 = R \cdot \alpha_1$,$\mu_1 + \mu_2 = 1$,α_1 为一个正数,x^w_d 是惯性粒子,其计算方法由公式(6)给出:

$$x^w_d(t) = \sum_{i-1}^{P} c^w_i x^p_{ij}(t) / \sum_{j-1}^{P} c^w_i, \tag{7}$$

其中,

$$c^w_i = \frac{F_{\max} - f(x^p_i(t)) + \varepsilon}{F_{\max} - F_{\min} + \varepsilon}, i = 1,2,\cdots,M, \tag{8}$$

m^e 是一种平均数,下面对 m^e 提出几种想法.

(1)m^e 是每一代粒子的中心.即

$$m^e = \frac{1}{N}\sum_{i-1}^{N} x(i,:). \tag{9}$$

(2)m^e 是在 3-范数下的每一代的中心,即

$$m^e = \frac{1}{N}\Big(\sum_{i-1}^{N} x(i,:)^3\Big)^{\frac{1}{3}}. \tag{10}$$

(3)m^e 是在无穷范数下的每一代的中心,即

$$m^e = \frac{1}{N}\{x(i,:) \mid \max_{1<i<N} \mid x(i,:)\}. \tag{11}$$

为了增强粒子的全局搜索能力,即粒子间的信息共享增强,于是对粒子使用两种交叉.

交叉 1:粒子之间的信息共享.

若 $f(x_i(t)) < f(x_k(t))$,则粒子 k 应该向粒子 i 学习

$$x_{kd}(t+1) = x_{kd}(t) + R \times (x_{id}(t) - x_{kd}(t)), \tag{12}$$

否则粒子 i 就向粒子 k 学习

$$x_{id}(t+1) = x_{id}(t) + R \times (x_{kd}(t) - x_{id}(t)). \tag{13}$$

交叉 2:粒子之间的经验共享.

若 $f(x^p_i(t)) < f(x^p_k(t))$,则粒子 k 应该向粒子 i 学习好的经验

$$x_{kd}(t+1) = x_{kd}(t) + A \times R \times (x^p_{id}(t) - x_{kd}(t)), \tag{14}$$

否则粒子 i 就向粒子 k 学习好的经验

$$x_{id}(t+1) = x_{id}(t) + A \times R \times (x^p_{kd}(t) - x_{id}(t)), \tag{15}$$

$$A = a \times \exp\left(\left(\frac{pF^i_i - pf^i_k}{\mid pF^i_k - pF^i_i \mid + \varepsilon}\right)\frac{N \times pF^i_k}{sF^i + \varepsilon}\right), \tag{16}$$

这里 $k(k \neq i)$ 是一个介于 1 和 N 之间的正整数.

2.3 模糊推理

在文献[13]中,利用 3 个参数 nf_i,α_i 和 w_i 对算法进行自适应改变,其中 nf_i 由(17)式得出

$$nf_i = \frac{f(x_i(t)) - F_{\min} + \varepsilon}{F_{\max} - F\min + \varepsilon}. \tag{17}$$

利用表 1、表 2 对参数进行调整,又可以得出 $\Delta\alpha_i$ 和 Δw_i 的类型,见图 13,图 14.

2.4 混沌 Logistic 映射扰动

为了使算法在迭代后期仍然具有较好的多样性,本文采用混沌扰动,算法 1 如下.

步骤 1　随机生成一个初始点 x_0,设最大混沌迭代次数为 M,令 $m = 1$;

步骤 2　令粒子数 $n = 1$;

步骤 3　令维数 $d = 1$;

步骤 4　根据 Logistic 映射混沌方程计算混沌序列 x_d，即

$$x_d = 4x_{d-1}(1-x_{d-1}); \tag{18}$$

步骤 5　对第 n 个支配解的第 d 维进行混沌局部搜索

$$x_{id} = x_{id} + \alpha \cdot \beta \cdot x_d, \tag{19}$$

其中 $\alpha = R, \beta = (-1)^d$；

步骤 6　对超出边界的粒子进行处理；

步骤 7　$d = d+1$，如果 $d \leq D$，则转步骤 4；

步骤 8　$n = n+1$，如果 $n \leq N$，则转步骤 3；

步骤 9　判断新生成的解是否为最优解；

步骤 10　$m = m+1$，如果 $m \leq M$，则转步骤 2，否则结束搜索.

表 1　参数 α_i 调整的模糊规则

规则号	输入		结果
	nf_i	α_i	$\Delta\alpha_i$
1	S	S	PL
2	S	M	PS
3	S	L	Z
4	M	S	PS
5	M	M	Z
6	M	L	NS
7	L	S	Z
8	L	M	NS
9	L	L	NL

表 2　参数 w_i 调整的模糊规则

规则号	输入		结果
	nf_i	w_i	Δw_i
1	S	S	PL
2	S	M	PL
3	S	L	Z
4	M	S	NL
5	M	M	Z
6	M	L	Z
7	L	S	Z
8	L	M	NL
9	L	L	NL

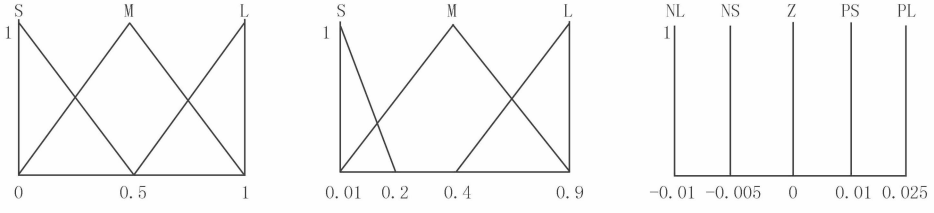

图13　Δa_i 的模糊推理规则

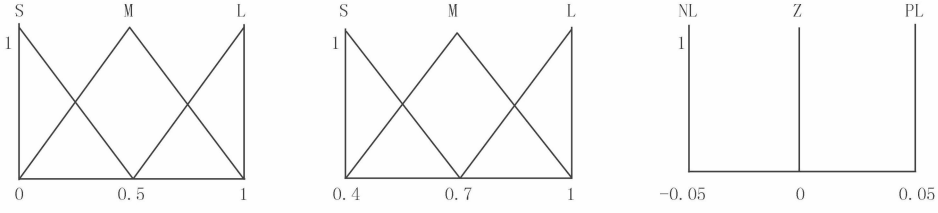

图14　Δw_i 的模糊推理规则

2.5　算法总流程

步骤 1　随机生成 M 个粒子的速度和位置；

步骤 2　求得每个粒子的适应度；

步骤 3　利用公式(7)和(8)求得惯性粒子 x^w；

步骤 4　求得全局最优解 gb；

步骤 5　求得粒子的平均 m^e；

步骤 6　比较 x^w、gb 与 m^e，适应值最小的那个作为最优值；

步骤 7　根据表 1 和表 2 计算 α_i 与 w_i；

步骤 8　根据公式(12) ～ (15) 对粒子进行交叉；

步骤 9　利用算法 1 对粒子进行混沌扰动；

步骤 10　利用(5) 和(6) 式对粒子进行更新；

步骤 11　判断是否满足终止条件，如果满足就输出 gb，否则，转步骤 2.

3　数值分析

3.1　参数分析

首先对学习因子 c_1 进行讨论，用了几个测试函数对 c_1 进行测试，测试结果见图 15～22.

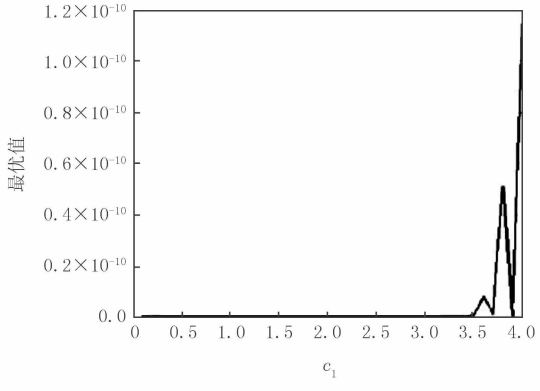

图15　不同的c_1取值对F_1的影响

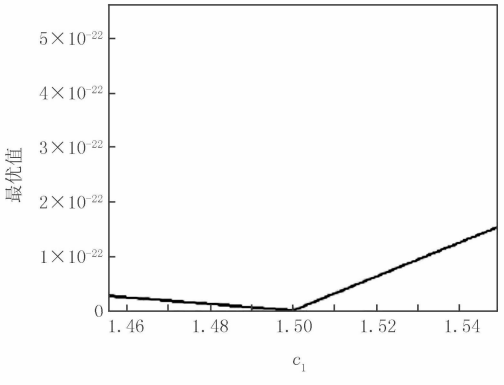

图16　对图15局部放大

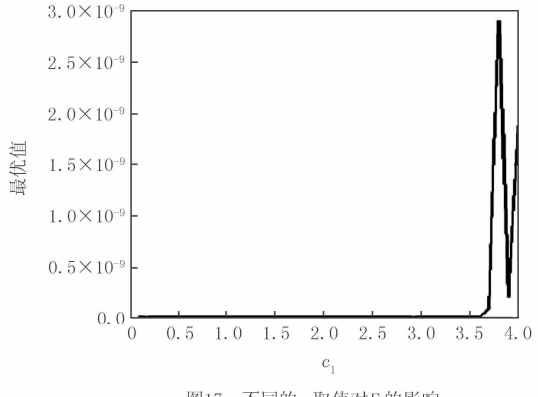

图17　不同的c_1取值对F_2的影响

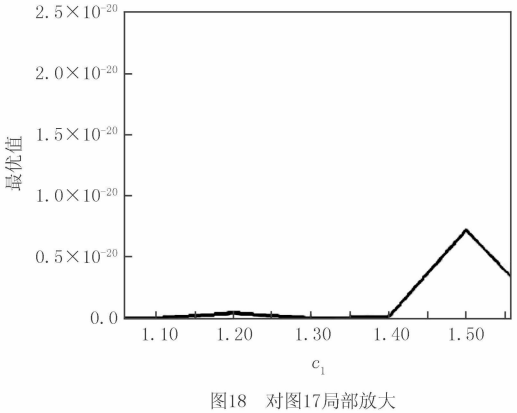

图18　对图17局部放大

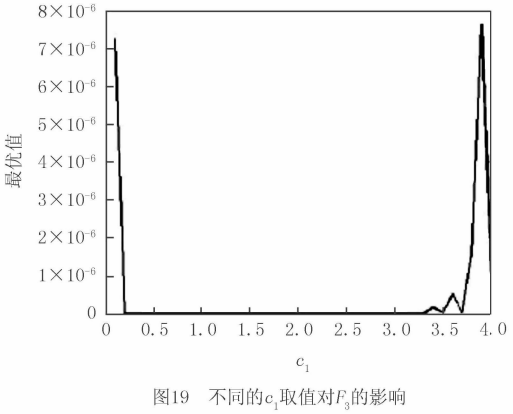

图19　不同的c_1取值对F_3的影响

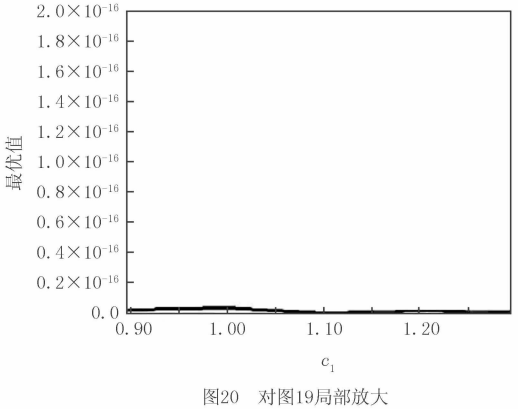

图20　对图19局部放大

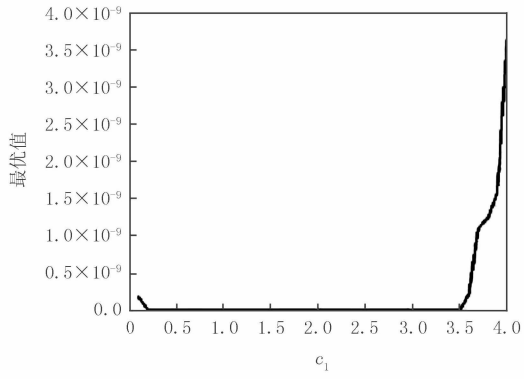

图21 不同的c_1取值对F_4的影响

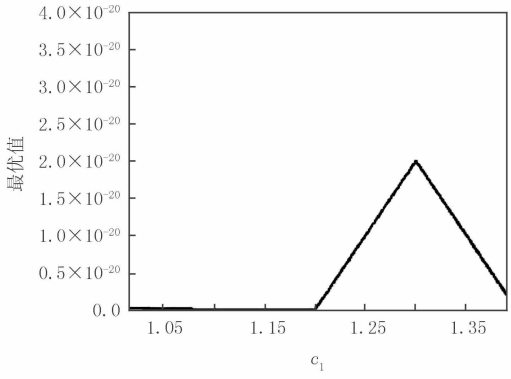

图22 对图21局部放大

上述图 15～22 分别在 c_1 取 1.5、1.3、1.1、1.2 时得到的结果最好，又由于测试函数在 $c_1 \in [1,1.5]$ 之间时，函数的优化效果较好，故取 c_1 为线性递减的函数

$$c_1(t) = c_{\max} - \frac{c_{\max} - c_{\min}}{i_{t\max}} \times t, \tag{20}$$

其中 $c_{\max} = 1.5, c_{\min} = 1, i_{t\max}$ 为最大迭代次数.

下面对公式(5)中M进行讨论，为了对不同m^e有更好的优化效果，于是分别对不同的m^e进行讨论，只对对公式(9)的M进行数据测试，其余公式中M方法类似，直接给出结果.

由表 3～表 4 给出测试函数的测试结果.

表 3 $F_1 \sim F_6$ 测试函数对不同 M 的最优值情况

M	F_1	F_2	F_3	F_4	F_5	F_6
0.1	5.717 70e-25	1.569 13e-24	2.025 58e-21	0.003 875 370	1.220 66e-06	0
0.2	2.505 22e-25	3.295 03e-23	3.653 21e-20	0.207 625 897	2.436 08e-06	0
0.3	9.170 99e-25	1.359 31e-24	4.102 26e-19	0.001 772 954	2.549 24e-06	0
0.4	4.876 33e-25	1.639 81e-23	6.655 46e-20	0.189 996 410	2.207 73e-06	0
0.5	2.975 55e-24	1.103 72e-21	3.152 84e-19	0.013 200 987	1.002 59e-05	0
0.6	1.908 62e-23	3.412 35e-21	3.210 61e-16	1.733 735 873	5.624 54e-06	0
0.7	1.196 85e-21	1.602 17e-20	4.449 47e-14	0.009 918 605	2.193 43e-06	0
0.8	1.143 57e-19	1.751 22e-18	1.133 82e-14	0.002 260 873	2.319 10e-08	0
0.9	4.123 57e-16	6.245 39e-16	2.816 45e-10	0.286 832 340	1.580 23e-06	0
1.0	0.416 883 704	0.487 255 951	1.554 045 585	2.456 864 212	2.28 384e-06	0.018 628 532
1.1	2.060 79e-04	1.231 32e-04	2.330 431 448	1.482 090 352	4.492 22e-08	1.709 57e-05
1.2	6.444 69e-06	3.354 68e-05	0.008 089 714	2.471 135 608	1.713 41e-06	1.412 43e-06
1.3	1.624 97e-07	2.321 59e-06	2.455 67e-04	0.209 775 403	4.756 84e-06	1.599 34e-08
1.4	1.705 25e-07	4.725 81e-08	0.002 272549	0.266 255 105	5.018 46e-07	4.291 87e-11
1.5	6.228 00e-07	4.571 97e-06	3.240 94e-04	0.969 747 362	1.845 22e-06	5.537 48e-09
1.6	6.868 67e-10	6.651 16e-08	1.401 28e-05	1.219 829 920	7.576 03e-07	2.286 37e-07
1.7	1.123 36e-09	2.044 82e-09	1.22823e-04	0.4308 756 27	1.4139 2e-07	3.173 22e-11
1.8	5.077 38e-08	1.609 64e-08	5.268 87e-04	2.756 702 473	1.162 26e-05	4.802 87e-10
1.9	4.882 43e-09	1.020 50e-06	1.956 19e-05	0.100 733 463	8.068 21e-06	1.385 26e-10
2.0	1.257 38e-08	2.966 89e-08	6.025 88e-05	0.050 797 781	5.227 15e-06	1.427 51e-09

表 4　$F_7 \sim F_{12}$ 测试函数对不同 M 的最优值情况

M	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
0.1	0	5.098 46e-11	2.455 11e-24	8.763 45e-46	3.610 12e-23	7.011 50e-24
0.2	1.776 35e-15	3.057 63e-11	5.896 26e-24	3.074 20e-46	1.582 00e-21	1.400 14e-23
0.3	7.975 84e-13	1.101 86e-10	5.983 69e-25	3.703 51e-44	1.535 52e-24	2.636 53e-23
0.4	0	1.967 27e-10	1.037 96e-24	8.265 75e-47	6.597 96e-23	1.993 77e-22
0.5	1.172 39e-13	6.181 43e-11	1.246 89e-23	6.594 36e-44	3.092 71e-23	3.801 56e-22
0.6	7.396 74e-12	1.535 09e-10	1.707 26e-22	2.162 68e-41	2.152 72e-21	5.698 70e-22
0.7	1.027 97e-11	1.385 41e-09	3.983 73e-23	3.716 37e-43	5.789 85e-20	4.631 58e-20
0.8	4.813 92e-13	6.285 99e-08	3.226 94e-20	6.966 37e-44	1.150 78e-19	4.337 76e-18
0.9	1.267 993 618	9.006 55e-07	7.711 09e-16	5.00 670e-39	1.680 92e-16	9.462 41e-15
1.0	47.372 533 08	1.947 256 873	0.287 856 810	1.279 36e-08	0.883 618 849	5.256 244 097
1.1	65.644 616 51	0.035 331 337	1.962 49e-04	7.252 70e-14	0.002 350 073	4.682 84e-04
1.2	48.570 024 78	0.009 087 514	4.145 22e-06	7.900 87e-16	1.662 41e-05	4.085 48e-05
1.3	0.919 1099 09	0.003 112 820	1.026 36e-07	9.518 14e-16	1.456 83e-06	7.084 86e-06
1.4	94.362 274 85	0.001 462 887	2.217 71e-08	7.250 28e-15	8.460 07e-06	2.522 81e-06
1.5	35.173 265 51	0.001 364 475	2.106 29e-08	5.911 28e-18	1.145 54e-06	1.375 70e-07
1.6	8.562 603 769	0.001 036 440	4.356 21e-09	3.544 23e-17	2.482 36e-07	3.384 90e-09
1.7	27.287 736 95	3.369 158 554	1.815 91e-09	3.015 66e-17	1.790 01e-09	1.749 92e-06
1.8	119.153 912 8	8.829 88e-05	3.011 99e-07	3.278 80e-21	1.434 01e-07	1.211 67e-07
1.9	113.455 698 2	1.034 60e-04	1.314 87e-09	6.514 23e-19	2.194 39e-06	1.458 64e-07
2.0	48.368 305 20	2.301 21e-04	1.398 97e-07	2.160 12e-16	7.797 49e-06	5.538 39e-08

注:这里的最大迭代次数为 100.

通过上述数据发现,当 M 介于 0 和 0.9 之间数据结果较好. 故引用教与学算法[18]中的做法,取 $M = 0.9 \times r(R)$.

3.2　测试函数

测试函数见表 5.

表 5　测试函数

函数名	测试函数	搜索空间	最优解
F_1	$f_1(x) = \sum_{i-1}^{D} x_i^2$	$[-100,100]^D$	$f_{\min} = 0$
F_2	$f_2(x) = \sum_{i-1}^{D} \left(\sum_{k-1}^{i} x_k \right)^2$	$[-100,100]^D$	$f_{\min} = 0$
F_3	$f_3(x) = \sum_{i-1}^{D} x_i^2 \times 10^{6\frac{i-1}{D-1}}$	$[-100,100]^D$	$f_{\min} = 0$
F_4	$f_4(x) = \sum_{i-1}^{D} \left(\sum_{k-1}^{i} x_k \right)^2 (1+R)$	$[-100,100]^D$	$f_{\min} = 0$
F_5	$f_5(x) = \max_{1 \le i \le D} \lvert x_i \rvert$	$[-100,100]^D$	$f_{\min} = 0$
F_6	$f_6(x) = \frac{1}{4000} \sum_{i-1}^{D} x_i^2 - \prod_{i-2}^{D} \frac{\cos(x_i)}{\sqrt{i}} + 1$	$[-600,600]^D$	$f_{\min} = 0$
F_7	$f_7(x) = x_i^2 - 10 \times \cos(2\pi x_i) + 10$	$[-5.12,5.12]^D$	$f_{\min} = 0$
F_8	$f_8(x) \sum_{i-1}^{D} \lvert x_i \rvert + \prod_{i-1}^{k} \lvert x_i \rvert$	$[-10,10]^D$	$f_{\min} = 0$
F_9	$f_9(x) = \sum_{i-1}^{D} (x_i + 0.5)^2$	$[-100,100]^D$	$f_{\min} = 0$
F_{10}	$f_{10}(x) = \sum_{i-1}^{D} \lvert x_i \rvert^{i+1}$	$[-1,1]^D$	$f_{\min} = 0$
F_{11}	$f_{11}(x) = \sum_{i-1}^{D} x_i^2 + \left(\sum_{i-1}^{D} 0.5 i x_i \right)^2 + \left(\sum_{i-1}^{D} 0.5 i x_i \right)^4$	$[-5,10]^D$	$f_{\min} = 0$
F_{12}	$f_{12}(x) = \sum_{i-1}^{D} (i x_i^2)$	$[-5.12,5.12]^D$	$f_{\min} = 0$

3.3 数值分析

设置参数:粒子群种群大小 $D=40$;最大迭代次数 $M=1000$;惯性权重 w 采用线性递减的方式 $w_{\max}=0.9$,$w_{\min}=0.4$,学习因子 c_1 采用线性递减的方式 $c_{\max}=1.5$,$c_{\min}=1$,学习因子 $c_2=2$,$a=1$;记加入(9)~(11)式的改进粒子群优化算法分别为 MPSO1,MPSO2,MPSO3,不加入 m^e 为 MPSO4,每次测试函数试验为 20 次.

对于表 6~11 的对比,在测试函数 F_1~F_{12} 上,本文的几个算法 MPSO1,MPSO2,MPSO3,MPSO4 在收敛到全局最优解上都有较大的提升,对于 20 词中的最好值、最差值以及平均值,在整体上提升比较大,从方差的值也可以看出算法都比较稳定,对于 PSO 来说,改进的算法唯一的不足就是运行时间较长,在提高算法寻优能力的同时,降低了算法的运行效率. 相对于文章中的几个改进算法而言,无论是最好值、最差值、平均值以及方差,MPSO3 的效果均相对较差,显然这个无穷范数型的改进粒子群相对于其他的意义不大. 对于 MPSO1、MPSO2、MPSO4 而言,最优值都相对接近,但是可以发现的是 MPSO2 的运行时间相对其他算法较长.

表 6 不同算法对函数 F_1,F_2 的测试结果

类别	维数	F_1					F_2				
		PSO	MPSO1	MPSO2	MPSO3	MPSO4	PSO	MPSO1	MPSO2	MPSO3	MPSO4
最好	30	0.9313	2.0e-265	2.8e-280	8.6e-117	7.6e-284	2.3400	6.9e-266	7.0e-278	3.2 e-96	1.0e-282
最差	30	2.977 7	1.2e-256	2.9e-275	6.7e-89	7.3e-276	8.0308	1.9e-257	3.9e-271	2.3 e-69	5.2e-272
平均	30	1.830 7	7.3e-258	2.9e-276	3.4e-90	4.1e-277	4.5787	1.2e-258	3.9e-272	1.1 e-70	2.6e-273
方差	30	0.354 5	0	0	2.2e-178	0	2.2637	0	0	2.7e-139	0
时间/s	30	4.695 9	178.88	224.40	178.46	189.556	101.28	4 304.08	4 671.44	4 922.16	4 944.62

表 7 不同算法对函数 F_3,F_4 的测试结果

类别	维数	F_3					F_4				
		PSO	MPSO1	MPSO2	MPSO3	MPSO4	PSO	MPSO1	MPSO2	MPSO3	MPSO4
最好	30	1.25e+04	8.92e-260	3.8e-274	2.5e-102	1.8e-277	9.380 96	6.7e-13	2.2e-12	5.4e-10	1.4e-13
最差	30	7.09e+04	2.59e-251	4.0e-266	6.5e-78	1.1e-260	166.033	1.4e+03	2.0e+3	1.5e+3	0.115 90
平均	30	3.91e+04	1.90e-252	4.4e-267	7.0e-79	5.6e-262	83.228 8	77.997 7	110.870	91.231 4	0.011 56
方差	30	2.78e+08	0	0	3.9e-156	0	137.799	1.0e+05	2.1e+05	1.4e+05	8.6e-04
时间/s	30	12.980 0	517.097	549.548	510.755	538.173	101.962	6 817.22	9 258.33	5 461.52	4 920.58

表 8 不同算法对函数 F_5,F_6 的测试结果

类别	维数	F_5					F_6				
		PSO	MPSO1	MPSO2	MPSO3	MPSO4	PSO	MPSO1	MPSO2	MPSO3	MPSO4
最好	30	2.44e-06	1.3e-180	1.4e-216	1.3e-251	3.7e-191	0.061 52	0	0	0	0
最差	30	2.69e-04	1.6 e-06	3.5e-07	8.7e-06	2.1e-06	0.124 18	0	0	0	0
平均	30	5.97e-05	2.4 e-07	8.7e-08	9.2e-07	5.0e-07	0.087 85	0	0	0	0
方差	30	5.90e-09	1.7 e-13	1.2e-14	6.3e-12	4.6e-13	2.43e-04	0	0	0	0
时间/s	30	5.252 00	201.014	248.413	199.105	208.852	17.176 0	587.672	626.468	616.534	609.283

表 9　不同算法对函数 F_7，F_8 的测试结果

类别	维数	F_7					F_8				
		PSO	MPSO1	MPSO2	MPSO3	MPSO4	PSO	MPSO1	MPSO2	MPSO3	MPSO4
最好	30	125.160	0	0	0	0	125.160	4.486 0	2.5e-130	2.3e-137	7.2e-51
最差	30	211.901	0	0	0	0	211.901	7.4187	2.7e-124	3.0e-132	4.3e-37
平均	30	159.846	0	0	0	0	159.846	5.928 6	1.5e-125	1.5e-133	2.2e-38
方差	30	371.887	0	0	0	0	371.887	0.756 4	3.7e-249	4.5e-265	9.2e-75
时间/s	30	9.984 99	252.391	295.623	294.889	269.536	9.984 99	6.286 0	413.207	484.455	440.876

表 10　不同算法对函数 F_9，F_{10} 的测试结果

类别	维数	F_9					F_{10}				
		PSO	MPSO1	MPSO2	MPSO3	MPSO4	PSO	MPSO1	MPSO2	MPSO3	MPSO4
最好	30	2.1521	1.4e-264	1.3e-278	2.1e-117	1.2e-283	6.10e-05	0	0	8.1e-181	0
最差	30	9.052 1	5.8e-257	3.1e-272	2.7e-80	1.9e-271	0.621 6	0	0	7.4e-131	0
平均	30	5.500 9	4.1e-258	1.5e-273	1.4e-81	9.9e-273	0.068 4	0	0	5.5e-132	0
方差	30	3.427 8	0	0	3.7e-161	0	0.022 2	0	0	3.2e-262	0
时间/s	30	4.914 0	77.168 9	107.287	77.370 0	80.108 9	18.892	625.218	638.510	589.411	654.857

表 11　不同算法对函数 F_{11}，F_{12} 的测试结果

类别	维数	F_{11}					F_{12}				
		PSO	MPSO1	MPSO2	MPSO3	MPSO4	PSO	MPSO1	MPSO2	MPSO3	MPSO4
最好	30	2.4151	6.7e-265	1.7e-274	5.2e-79	3.3e-283	12.627	2.2e-262	1.7e-277	2.1e-108	2.7e-283
最差	30	21.737	9.1e-240	1.3e-260	3.4e-51	3.1e-275	39.301	2.8e-252	1.5e-262	1.2e-82	1.5e-273
平均	30	7.328 1	4.5e-241	6.8e-262	1.7e-52	2.5e-276	25.763	1.4e-253	7.8e-264	6.2e-84	1.1e-274
方差	30	16.850	0	0	5.9e-103	0	38.241	0	0	7.5e-166	0
时间/s	30	6.988 9	454.326	518.393	516.396	460.708	5.007 9	77.426 9	109.962	77.174 9	80.628 0

4　结束语

本文提出了融合模糊推理的粒子群优化算法，该算法中模糊推理机制提高了算法的全局搜索能力，这大大提高了粒子搜索能力，将鸟群中的信息共享机制引入粒子群优化算法，使得对局部搜索能力增强，又引入了 Logistic 混沌避免算法出现早熟现象. 尽管如此，但与真实解仍然存在差距，同时运行速度还有待提高，因此进一步改进算法将是下一步研究的重点.

参　考　文　献

[1] Reynolds C W. Flocks, herds, and schools: a distributed behavioral model[J]. Comput Graph, 1987, 21: 25-34.

[2] Kennedy J, Eberhart R C. Particle swarm optimization[J]. Proceeding of 1995 IEEE International Conference on Neural Networks, 1995, 4: 1942-1948.

[3] Shi Y, Eberhart R. A modified particle swarm optimizer[C]//Proceedings of 1998 IEEE International Conference on Evolutionary Computation. Berlin: Springer, 2005: 69-73.

[4] Ratnaweera A, Halgamuge S, Watson H. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients [J]. IEEE Trans Evol Comput, 2004, 8: 240-255.

[5] Gao H, Xu W B. A new particle swarm algorithm and its globally convergent modifications[J]. IEEE Trans Syst Man Cybern Part B:

Cybern, 2011,41:1334-1351.
[6] Kundu R, Das S, Mukherjee R, et al. An improved particle swarm optimizer with difference mean based perturbation[J], Neurocomputing,2014,129:315-333.
[7] Gaing Z L. A particle swarm optimization approach for optimum design of PID controller in AVR system[J]. IEEE Trans Energy Convers, 2004,19:384-391.
[8] Hong Y Y, Lin F J, Chen S Y,et al. A novel adaptive elite-based particle swarm optimization applied to VAR optimization in electric power systems[J]. Math Probl Eng, 2014 ,2014: 1-14.
[9] Cavuslu M A, Karakuzu C, Karakaya F. Neural identification of dynamic systems on FPGA with improved PSO learning[J]. Appl Soft Comput, 2012,12: 2707-2718.
[10] Razieh S. A Survey on semi-supervised feature selection methods[J]. Pattern Recognition, 2017,64:141-158.
[11] Ma D L. Parameter identification for continuous point emission source based on Tikhonov regularization method coupled with particle swarm optimization algorithm[J]. Journal Of Hazardous Materials,2017,325:239-250.
[12] Chang W D,Shih S P. PID controller design of nonlinear systems using an improved particle swarm optimization approach[J]. Commun Nonlinear Sci Numer. Simul,2010,15: 3632-3639.
[13] Liu Y, Qin Z,Shi Z W,et al. Center particle swarm optimization[J]. Neurocomputing,2007,70: 672-679.
[14] Li N J, Wang W J,James Hsu C C,et al. Enhanced particle swarm optimizer incorporating a weighted particle[J]. Neurocomputing, 2014,124: 218-227.
[15] Li N, Wang W, Hsu C. Hybrid particle swarm optimization incorporating fuzzy reasoning and weighted particle[J]. Neurocomputing, 2015,167:488-501.
[16] Meng X,GAO X. A new bio-inspired optimization algorithm:Bird Swarm Algorithm[J]. Journal of Experomental & Theoretical Atificial Intelligence,2015,1362-3079
[17] Teng W J, Wang W J. Constructing a user-friendly Ga-based fuzzy system directly from numerical data[J]. IEEE Trans. Syst Man Cybern Part B Cybern, 2004,34: 2016-2070.
[18] Rao R,Savsani V, Vakharia D. Teaching-learning-based optimization:A novel method for constrained mechanical design optimization problems[J]. Computer-Aided Design, 2011,43:303-315.

Particle Swarm Optimization Algorithm Based on Fuzzy Reason

Shi Xudong[1], Gao Yuelin[2], Han Junrun[1]

(1. School of Mathematics and Computer, Ningxia University, Yinchuan 750021,China;
2. Research Institute of Information and System Computation Science, Beifang University of Nationalities, Yinchuan 750021,China)

Abstract: It is well know that the standard PSO is unreliable to solve unrestraint problems with global optimization, due to the insufficient search ability nature of the PSO. In order to overcome the shortcoming, the particle swarm optimization algorithm based on fuzzy reason uses average particle, fuzzy reasoning to improve the speed of the particle swarm update formula. Fuzzy reasoning dynamically inertia weight and velocity updating formula of weight factor. Chaos disturbance increase algorithm local search ability in the late. A number of numerical examples which are used twelve trial functions have shown the present method is found to be ultra-accurate, capable of modeling global optimization.

Keywords: particle swarm optimization; Fuzzy reasoning; information sharing

[责任编辑 陈留院]