基于高斯扰动的量子粒子群优化算法

王小根[a], 龙海侠[a], 孙 俊[b]
(江南大学 a. 教育学院; b. 信息工程学院, 江苏 无锡 214122)

摘 要: 针对量子粒子群优化(QPSO)算法在优化过程中面临早熟问题,提出了在粒子的平均位置或全局最优位置上加入高斯扰动的 QPSO 算法,可以有效地阻止粒子的停滞,因此较容易地使粒子避免陷入局部最优。为了评估算法的性能,利用标准测试函数对标准 PSO 算法、QPSO 算法以及基于高斯扰动的 QPSO 算法进行了比较测试。其结果表明,该算法具有较强的全局搜索能力和较快的收敛速度。

关键词: 量子粒子群优化算法; 平均位置; 全局最优位置; 高斯扰动
中图分类号: TP18;TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2010)06-2093-04
doi:10.3969/j.issn.1001-3695.2010.06.028

Quantum-behaved particle swarm optimization based on Gaussian disturbance

WANG Xiao-gen[a], LONG Hai-xia[a], SUN Jun[b]
(*a. School of Education, b. School of Information Engineering, Jiangnan University, Wuxi Jiangsu* 214122, *China*)

Abstract: Due to shortcoming of quantum-behaved particle swarm optimization (QPSO) algorithm that it was often premature convergence, this paper proposed a revised QPSO with Gaussian disturbance on the mean best position or global best position of the swarm. The disturbance could effectively prevent the stagnation of the particles and therefore made them escape the local optima more easily. To evaluate the performance of the new method, tested the QPSO with Gaussian disturbance, along with QPSO and standard PSO on several well-known benchmark functions. Experiment simulations show that the proposed algorithm has powerful optimizing ability and more quickly convergence speed.

Key words: quantum-behaved particle swarm optimization algorithm; mean position; global best position; Gaussian disturbance

0 引言

美国社会心理学家 Kennedy 和电气工程师 Eberhart 于 1995 年提出了 PSO 算法[1],其主要思想来源于对鸟类群体行为的研究,已经在很多领域得到了成功的应用。但是在算法收敛性方面,Burgh 证明了标准 PSO 算法不能收敛于全局最优解,甚至于局部最优解[2]。

在 PSO 算法的改进方面,Kennedy[3]引入了邻域拓扑的概念来调整邻域的动态选择,增加邻域间的信息交流,提高种群的多样性;Lovbjerg 等人[4]于 2001 年将遗传算法中的子群体概念引入 PSO 算法中,同时引入繁殖算子以进行子群体的信息交流;Kennedy 于 2004 年从概率统计的角度,将粒子的运动改为正态扰动的随机扰动,并采用邻域环拓扑结构来改进 PSO 算法的性能[5];Riget 等人[6]利用种群的多样性出发,在 PSO 算法中增加了发散能力,较好地提高了算法的全局搜索能力;Sun Jun 等人[7]提出了基于量子行为的粒子群算法(QPSO),提高了粒子群的全局收敛能力。

QPSO 算法的思想来源于量子力学和 PSO 模型,每个粒子类似于量子空间中的基本粒子,问题可行解的整个空间都是每个粒子的搜索范围,因而其全局搜索性能大大优于经典 PSO 算法。但 QPSO 算法在运行过程中也存在粒子群体多样性衰减的现象,即随着算法的运行,部分粒子由于速度的减小而失去活力,导致后续的搜索中失去局部搜索能力和全局搜索能力,因此出现了许多改进的 QPSO 算法。Sun Jun 等人[8]提出了概率分布机制使种群在全局搜索中更加有效;而且 Sun Jun 等人[9]提出了基于多样性的 QPSO 来防止种群的聚集,使粒子更容易避开局部最优点;在 QPSO 算法中加入模拟退火(SA)不仅能跳出局部最优而且能够提高 QPSO 算法的全局搜索能力[10];Coelho[11]介绍了基于 Gaussian 概率分布的 QPSO 算法,在此算法中引入了变异算子;在 QPSO 算法中利用免疫记忆和接种技术的特征引导算法的搜索过程,以提高算法的收敛速度[12]。本文在 QPSO 算法的基础上引入高斯扰动来提高 QPSO 算法的全局收敛能力。

1 QPSO 算法

在一个 n 维的目标搜索空间中,QPSO 算法有 m 个代表潜在问题解的粒子组成群体 $X=\{x_1,x_2,\cdots,x_m\}$,在 t 时刻第 i 个粒子的位置为 $x_i(t)=(x_{i1}(t),x_{i2}(t),\cdots,x_{in}(t))$,粒子没有速度向量。个体最好的位置表示为 $P_i(t)=(P_{i1}(t),P_{i2}(t),\cdots,P_{in}(t))$;群体全局最好的位置为 $P_g(t)=(P_{g1}(t),P_{g2}(t),\cdots,P_{gn}(t))$,其中 g 为处于全局最好位置粒子的下标。

Clerc 等人[13]在对粒子轨道的分析中证明了这样一个事

收稿日期: 2009-11-26; **修回日期**: 2009-12-24

作者简介:王小根(1965-),男,浙江湖州人,副教授,博士研究生,主要研究方向为进化计算和图像处理;龙海侠(1980-),女,江苏徐州人,博士研究生,主要研究方向为进化计算(haixia_long@163.com);孙俊(1971-),男,江苏无锡人,副教授,硕导,主要研究方向为人工智能、计算机控制技术等.

实:假如每一个粒子收敛到它的局部吸引点 $A_i=(A_{i1},A_{i2},\cdots,A_{im})$ 并且满足下面的条件式(1),那么 PSO 算法是收敛的。

$$A_{ij}(t)=(c_1r_1P_{ij}(t)+c_2r_2P_{gj}(t))/(c_1r_1+c_2r_2)$$
$$\text{or}\quad A_{ij}(t)=\varphi\times P_{ij}(t)+(1-\varphi)\times P_{gj}(t) \tag{1}$$

其中:$\varphi=c_1r_1/(c_1r_1+c_2r_2)$。从式(1)中可以看出局部吸引点 A_i 是一个位于超矩形中的随机点。下面介绍 QPSO 算法。

假如粒子在以吸引点为中心的一维 δ 势阱中运动,解一维 δ 势阱的 Schrödinger 方程,得到概率扰动函数 $D(x)=e^{-2|A-x|/L}$。使用 Monte Carlo 方法可以得到:

$$x=A\pm\frac{L}{2}\ln(1/u),u\sim U(0,1) \tag{2}$$

Sun Jun 等人在 QPSO 算法中引入了平均最好位置[7]。它定义为所有粒子个体最好位置的平均,即

$$C(t)=(C_1(t),C_2(t),\cdots,C_m(t))=$$
$$\left(\frac{1}{M}\sum_{i=1}^{n}P_{i,1}(t),\frac{1}{M}\sum_{i=1}^{n}P_{i,2}(t),\cdots,\frac{1}{M}\sum_{i=1}^{n}P_{i,n}(t)\right) \tag{3}$$

L 的值用下面的公式计算:$L=2\alpha\times|C_j(t)-x_{ij}(t)|$,粒子位置更新方程为

$$x_{ij}(t+1)=A_{ij}(t)\pm\alpha\times|C_j(t)-x_{ij}(t)|\times\ln(1/u) \tag{4}$$

其中,参数 α 为收缩—扩张系数,可以通过调节 α 的值来控制算法的收敛速度,α 必须满足 $\alpha<1.782$ 来保证粒子的收敛[14]。通常 α 从 α_0 到 α_1 线性减小($\alpha_0>\alpha_1$)。

2 基于高斯扰动的 QPSO 算法

在经典 PSO 算法中,随着粒子群的集体性不断提高、多样性的快速下降,粒子间信息的快速流动导致算法很难逃离局部最优位置,因此粒子的群集性导致多样性的降低和适应值变化的停滞。在 QPSO 算法中,尽管每次迭代中单个粒子的搜索空间为问题可行解的所有空间,由于粒子的群集性,算法执行过程中多样性的损失也不可避免。

从经典 PSO 和 QPSO 算法的更新方程可以推导得出,经典 PSO 和 QPSO 算法中的所有粒子将收敛到一个公共的点,使得种群的多样性非常低,并且粒子在下一次迭代之前没有进一步的搜索。为了克服这个问题,对 QPSO 算法进行了改进研究,以期提高 QPSO 算法后期阶段粒子的多样性和算法的性能。改进的方法是在 QPSO 算法中加入高斯扰动(Gaussian disturbance),采用下面三种策略:

a)在平均最好位置上加入高斯扰动

$$C_j(t)=C_j(t)+\varepsilon\times Rn;\ j=1,2,\cdots,N \tag{5}$$

b)在全局最好位置上加入高斯扰动

$$P_g(t)=P_g(t)+\varepsilon\times Rn;\ j=1,2,\cdots,N \tag{6}$$

c)在平均最好位置和全局最好位置上均加入高斯扰动,利用式(5)(6)。

在式(5)(6)中,ε 是一个预先设定的参数;Rn 是均值为 0 和标准方差为 1 的满足高斯扰动的随机数。

在 QPSO 算法中加入高斯扰动可以有效地避免多样性的下降和早熟收敛的产生,这是因为算法运行过程中粒子群体不断进化时,时刻对平均最好位置和全局最好位置进行扰动,保持粒子的活动性,帮助粒子逃离局部最优位置,使得算法得到更好的性能。具有高斯扰动的 QPSO 算法能够保持粒子的活力,在算法的后续阶段能够有效地帮助粒子逃离局部最优,从而提高算法的性能。

改进后的 QPSO 算法的执行过程如下:

a)在问题空间中初始化粒子群中粒子的位置,并设置初始个体最优值。

b)利用式(3)计算平均最优位置 C。

c)利用式(5)或式(6)得到扰动点。

d)对于粒子群体中的每一个粒子,计算粒子的当前适应值并与前一次迭代的适应值进行比较,更新粒子的个体最优位置 P_i,即如果 $f[x_i(t+1)]<f[P_i(t)]$,则 $P_i(t+1)=x_i(t+1)$。

e)计算粒子群体的群体最优位置 $P_g(t+1)=\arg\min_{1\leqslant i\leqslant M}\{f[P_i(t)]\}$。

f)更新粒子群体的全局最优位置。

g)根据 $A_{ij}(t)=\varphi_j(t)\times P_{ij}(t)+[1-\varphi_j(t)]\times P_{gj}(t)$,$\varphi_j(t)\sim U(0,1)$ 计算得到一个随机点的位置。

h)根据式(4)计算粒子的新位置。

i)重复步骤 b)~h),直到满足结束循环条件。

3 仿真实验

3.1 实验设置

本文实验中引入了五个标准函数来测试具有高斯扰动的 QPSO 算法的总体性能,对测试函数的函数形式、搜索范围、初始化范围的设置如表 1[15] 所示,所有函数的理论最小值均为 0。本文设计了四类测试实验:a)SPSO 优化实验;b)QPSO 优化实验;c)平均最好位置上加入高斯扰动的 QPSO 算法(MQPSO);d)全局最好位置上加入高斯扰动的 QPSO 算法(BQPSO);e)平均和全局最好位置上均加入高斯扰动的 QPSO 算法(MBQPSO)。

表 1 标准测试函数及其参数

function	function expression	search domain	initial range
Sphere	$f_1(x)=\sum_{i=1}^{n}x_i^2$	[-100,100]	[50,100]
Rosenbrock	$f_2(x)=\sum_{i=1}^{n-1}(100\times(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	[-100,100]	[15, 30]
Rastrigin	$f_3(x)=\sum_{i=1}^{n}(x_i^2-10\times\cos(2\pi x_i)+10)$	[-5.12,5.12]	[2.56,5.12]
Griewank	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600,600]	[300,600]
Schaffer	$f_5(x)=0.5+\frac{(\sin(\sqrt{x^2+y^2})^2-0.5}{(1.0+0.001(x^2+y^2))^2}$	[-100,100]	[30,100]

在 MQPSO、BQPSO 和 MBQPSO 三种算法中,参数 ε 设置为常数 0.000 5。

实验测试过程中,将测试函数的函数值设置为粒子的适应值并对每个函数进行 50 次测试,记录平均最优适应值和标准偏差。为了研究算法的可扩展性,每个函数的测试过程中采用了不同的种群尺寸 M 和不同的维度 D。种群大小设置为 20、

40 和 80,除了 Schaffer 函数外,其他所有函数的维度分别设置为 10、20 和 30,对应的算法最大迭代次数设置为 1 000、1 500 和 2 000;Shaffer 函数的最大迭代次数设置为 2 000,维度设置为 2。对于标准 SPSO 算法,其加速度系数设置为 $c_1 = c_2 = 2$,惯性权重因子线性地从 0.9 减少到 0.4。对于 QPSO、MQPSO、BQPSO、MBQPSO 算法,参数 α 随算法的运行从 1.0 线性地降低到 0.5。算法 50 次运行后最小适应值的均值和标准方差如表 2 ~ 6 所示。

表 2　Sphere function 的仿真结果

M	dim	g_{max}	SPSO mean best (St. Dev.)	QPSO mean best (St. Dev.)	MQPSO Mean Best (St. Dev.)	BQPSO Mean Best (St. Dev.)	MBQPSO Mean Best (St. Dev.)
	10	1000	4.6636e-021 (1.0816e-020)	9.9733e-044 (4.8708e-043)	5.0300e-008 (1.3118e-008)	5.0497e-006 (3.4412e-006)	5.3813e-006 (3.1992e-006)
20	20	1500	9.0397e-012 (3.6044e-011)	1.3330e-023 (3.8578e-023)	4.7465e-007 (9.9433e-008)	3.5660e-005 (2.1636e-005)	4.7677e-005 (2.3478e-005)
	30	2000	5.4652e-008 (1.2866e-007)	7.3807e-014 (3.9980e-013)	1.4537e-006 (2.4669e-007)	9.3845e-005 (5.4052e-005)	1.1948e-004 (5.4181e-005)
	10	1000	6.0519e-025 (1.3454e-024)	3.7703e-075 (2.2257e-074)	3.9535e-008 (9.5266e-009)	3.2510e-006 (2.2243e-006)	5.0716e-006 (3.0487e-006)
40	20	1500	8.8091e-015 (2.7541e-014)	6.2598e-043 (3.9131e-042)	3.7711e-007 (8.2315e-008)	3.5805e-005 (2.4493e-005)	4.0116e-005 (2.3457e-005)
	30	2000	5.7175e-011 (9.2317e-011)	5.8240e-029 (2.7907e-028)	1.2868e-006 (1.7584e-007)	8.4143e-005 (3.8269e-005)	1.0903e-004 (5.3538e-005)
	10	1000	1.0585e-028 (3.2288e-028)	1.2629e-101 (7.5408e-101)	2.9073e-008 (9.0087e-009)	3.5302e-006 (2.4952e-006)	4.2719e-006 (1.8701e-006)
80	20	1500	7.7687e-018 (1.8497e-017)	5.3031e-069 (2.4450e-068)	3.4336e-007 (7.2103e-008)	3.0502e-005 (1.4564e-005)	3.5635e-005 (1.7060e-005)
	30	2000	9.7939e-013 (4.2925e-012)	4.5013e-050 (2.1861e-049)	1.0576e-006 (1.5789e-007)	7.5891e-005 (3.7680e-005)	1.1553e-004 (6.9752e-005)

表 3　Rosenbrock function 的仿真结果

M	dim	g_{max}	SPSO mean best (St. Dev.)	QPSO mean best (St. Dev.)	MQPSO mean best (St. Dev.)	BQPSO mean best (St. Dev.)	MBQPSO mean best (St. Dev.)
	10	1000	27.1159 (46.8982)	34.0254 (63.4626)	25.8956 (46.9880)	25.2115 (56.3068)	26.2399 (51.3287)
20	20	2000	94.5540 (184.6187)	71.9134 (117.2893)	68.2976 (74.2750)	82.7217 (135.8673)	68.9907 (102.4339)
	30	2000	146.0952 (160.1071)	122.8972 (142.4230)	117.2070 (182.7348)	152.8106 (231.8329)	121.8970 (176.2897)
	10	1000	27.8007 (63.0610)	18.1606 (25.5968)	10.4080 (14.5591)	16.7396 (36.5529)	14.1402 (16.3845)
40	20	1500	52.0378 (55.7566)	49.8761 (46.4435)	37.4891 (41.7028)	41.5129 (44.5833)	41.2398 (37.4944)
	10	1000	107.3789 (166.7348)	69.0404 (58.9664)	54.0157 (44.7894)	51.0299 (44.3111)	71.5374 (69.0916)
	10	1000	12.6281 (24.4214)	12.2803 (23.4514)	11.6420 (15.8026)	5.5796 (7.8057)	9.3444 (12.5441)
80	20	1500	53.2359 (125.5355)	37.0901 (38.3374)	31.7689 (33.3099)	37.9613 (44.7665)	32.3365 (37.8422)
	30	2000	108.0814 (150.4659)	56.1391 (37.6330)	51.6486 (35.0710)	53.1795 (41.3760)	48.3516 (35.7636)

表 4　Rastrigin function 的仿真结果

M	dim	g_{max}	SPSO mean best (St. Dev.)	QPSO mean best (St. Dev.)	MQPSO mean best (St. Dev.)	BQPSO mean best (St. Dev.)	MBQPSO mean best (St. Dev.)
	10	1000	5.0041 (2.6778)	4.6331 (2.1420)	4.7510 (3.1345)	4.5472 (2.6457)	4.2412 (2.1028)
20	20	1500	22.5019 (7.1447)	13.9361 (4.1434)	14.2945 (4.0683)	15.0436 (6.9150)	13.9171 (5.6081)
	30	2000	51.9810 (13.9263)	28.8876 (8.6535)	27.6005 (7.2470)	28.5434 (6.8237)	29.8254 (12.6546)
	10	1000	3.7436 (1.6012)	2.6205 (1.6433)	2.8383 (1.4988)	2.9309 (1.5199)	3.0445 (1.6246)
40	20	1500	16.4207 (5.73480)	11.0822 (3.9558)	10.7361 (3.5389)	11.4977 (3.8580)	11.0816 (3.5947)
	30	2000	39.9508 (9.7971)	21.2964 (5.2381)	20.1047 (5.3041)	19.3349 (6.2123)	21.0376 (4.9071)
	10	100	2.3688 (1.2046)	2.3989 (2.0275)	1.8728 (1.3937)	2.1618 (1.1852)	1.8095 (1.2067)
80	20	1500	12.5326 (4.3126)	9.1060 (3.7092)	7.8344 (2.1418)	8.4864 (2.7149)	7.9112 (2.5657)
	30	2000	32.1621 (8.1184)	15.9700 (4.3547)	15.8483 (4.0925)	16.5147 (4.1843)	16.0957 (3.8550)

表 5　Griewank function 的仿真结果

M	dim	g_{max}	SPSO mean best (St. Dev.)	QPSO mean best (St. Dev.)	MQPSO mean best (St. Dev.)	BQPSO mean best (St. Dev.)	MBQPSO mean best (St. Dev.)
	10	1000	0.0990 (0.0494)	0.0631 (0.0453)	0.0597 (0.0360)	0.0609 (0.0432)	0.0569 (0.0384)
20	20	1500	0.0327 (0.0287)	0.0249 (0.0261)	0.0247 (0.0197)	0.0195 (0.0194)	0.0226 (0.0236)
	30	2000	0.0186 (0.0248)	0.0105 (0.0128)	0.0113 (0.0117)	0.0093 (0.0123)	0.0094 (0.0133)
	10	1000	0.0770 (0.0304)	0.0475 (0.0391)	0.0473 (0.0355)	0.0401 (0.0226)	0.0456 (0.0338)
40	20	1500	0.0325 (0.0292)	0.0265 (0.0360)	0.0202 (0.0173)	0.0180 (0.0178)	0.0179 (0.0190)
	30	2000	0.0135 (0.0130)	0.0093 (0.0145)	0.0074 (0.0096)	0.0086 (0.0112)	0.0115 (0.0143)
	10	1000	0.0726 (0.0321)	0.0421 (0.0415)	0.0375 (0.0324)	0.0357 (0.0229)	0.0406 (0.0389)
80	20	1500	0.0264 (0.0228)	0.0139 (0.0140)	0.0104 (0.0115)	0.0102 (0.0152)	0.0154 (0.0162)
	30	2000	0.0104 (0.0143)	0.0084 (0.0118)	0.0077 (0.0103)	0.0077 (0.0105)	0.0073 (0.0117)

表 6　Schaffer function 的仿真结果

M	dim	g_{max}	SPSO mean best (St. Dev.)	QPSO mean best (St. Dev.)	MQPSO mean best (St. Dev.)	BQPSO mean best (St. Dev.)	MBQPSO mean best (St. Dev.)
20	2	2000	0.0012 (0.0032)	0.0016 (0.0036)	0.0011 (0.0030)	7.7769e-004 (0.0027)	0.0011 (0.0030)
40	2	2000	0 (0)	3.8865e-004 (0.0019)	7.5145e-005 (5.2839e-004)	1.3470e-008 (1.7547e-008)	8.7229e-008 (1.7562e-007)
80	2	2000	0 (0)	2.2252e-009 (4.8635e-009)	1.0159e-007 (7.0668e-007)	7.4479e-009 (1.0750e-008)	7.6312e-009 (1.6788e-008)

3.2　实验结果

从表 2 ~ 6 的实验结果中可以看出,三种具有高斯扰动的 QPSO 算法,其中 MQPSO 算法的性能总体上优于 BQPSO 和 MBQPSO 算法。对于函数 Sphere 来说,QPSO 效果最好,SPSO 次之,MQPSO 优于 BQPSO 和 MBQPSO,BQPSO 和 MBQPSO 的性能相当。对于 Rosenbrock 函数来说,MQPSO 和 MBQPSO 的性能在各种情况下都优于 SPSO 和 QPSO;BQPSO 除了在粒子数为 20、维数为 20 和 30,粒子数为 80、维数为 20 的三种情况下,其他所有的情况下都优于 SPSO 和 QPSO 算法。除 MQPSO、BQPSO、MBQPSO 每种情况下的最小值都小于 SPSO 和 QPSO 算法得到的结果,如对粒子数 20、维数为 10 的情况,MQPSO 的最小值为 25.895 6,BQPSO 的最小值为 25.211 5,MBQPSO 的最小值为 26.239 9,都小于 SPSO 和 QPSO 的函数平均值。对于 Rastrigin 函数,MQPSO、BQPSO、MBQPSO 大多数情况下取得的函数平均值都小于 SPSO 和 QPSO 算法,QPSO 算法的性能优于 SPSO 算法。对于 Griewank 函数,MQPSO、BQPSO、MBQPSO 在所有情况下取得的函数平均值都小于 SPSO,80% 的情况都优于 QPSO 算法,并且 QPSO 算法的性能优于 SPSO 算法。对于 Schaffer 函数来说,除了粒子数 80、维数为 2 的情况下,QPSO 算法的性能优于 MQPSO、BQPSO、MBQPSO 算法,其他情况下 MQPSO、BQPSO、MBQPSO 算法的性能优于 QPSO 算法。

图 1 给出了粒子数为 40、维数为 20 的情况下前四个函数对应的 SPSO、QPSO 和 GQPSO 三种算法的收敛过程;对于 Shaffer 函数,维数为 2 的情况下 SPSO、QPSO 和基于高斯扰动的 QPSOC(GQPSO)三种算法的收敛过程。其中 GQPSO 描绘的是三种扰动下取得最小值的算法的收敛过程。对于函数 Sphere 来说,GQPSO 描述的是 MQPSO 的收敛过程;对于函数 Rosenbrock 来说,GQPSO 描述的是 MQPSO 的收敛过程;对于函数 Rastrigin 来说, GQPSO 描述的是 MQPSO 的收敛过程;对

于函数 Griewank 来说,GQPSO 描述的是 MBQPSO 的收敛过程;对于函数 Shaffer 来说,在粒子数为 40、维数为 2 的情况下,GQPSO 描述的是 BQPSO 的收敛过程。

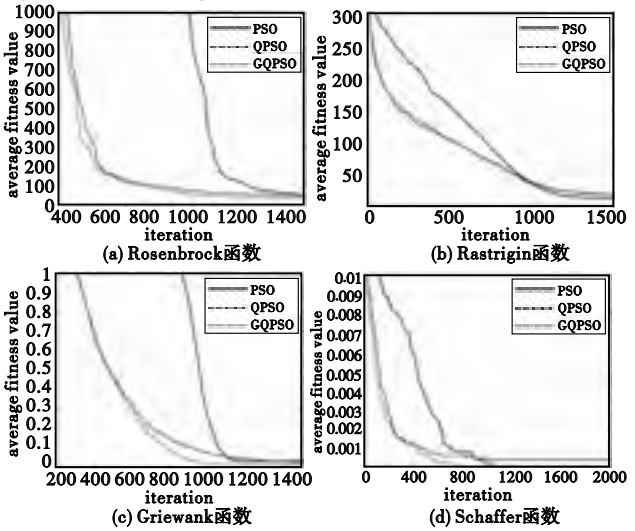

图1　SPSO、QPSO、GQPSO三种算法对应的四个函数在粒子数为40、维数为20的情况下的收敛过程

从图 1 的收敛过程可以看出,QPSO 和 GQPSO 的收敛速度大于 SPSO 的收敛速度,QPSO 与 GQPSO 的收敛速度相似。总体上,GQPSO 的性能优于 QPSO 的性能。

图 2 给出了粒子数为 20、维数为 10 的情况下 QPSO 算法和高斯扰动的 QPSO 算法对应的四个函数的多样性变化曲线,多样性的计算参考文献[16]。从图中可以看出,加入高斯扰动的 QPSO 算法提高了种群的多样性。

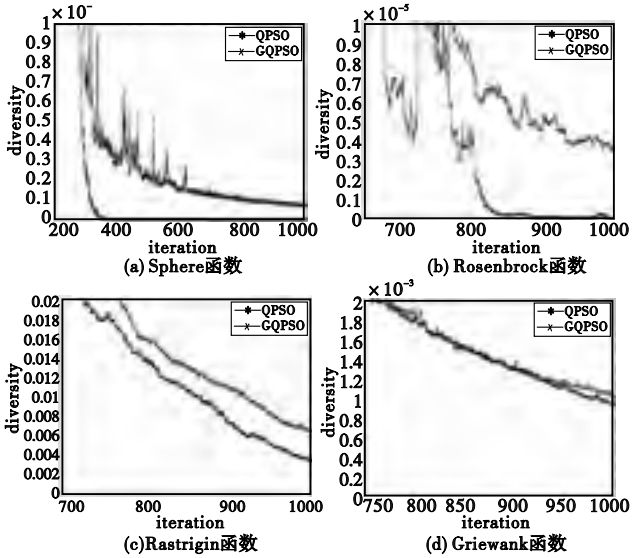

图2　QPSO、GQPSO两种算法对应的四个函数在粒子数为20、维数为10的情况下的多样性变化曲线

4　结束语

所有的实验结果显示,QPSO 算法和基于高斯扰动的 QPSO 算法在绝大多数测试函数上都优于 SPSO 算法。基于高斯扰动的 QPSO 算法优于 QPSO 算法,并且在平均最好位置上加入的扰动取得的结果最好,基于高斯扰动的 QPSO 算法的收敛速度最快,提高了种群的多样性。

参考文献:

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

[2] Van den BERGH F. An analysis of particle swarm optimizers[D]. Pretoria: University of Pretoria, 2001.

[3] KENNEDY J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance[C]//Proc of Congress on Evolutionary Computation. Washington DC: IEEE Press, 1999: 1931-1938.

[4] LOVBJERG M, RASUSSEN T K, KRINK T. Hybrid particle swarm optimizer with breeding and subpopulation[C]//Proc of the 3rd Genetic and Evolutionary Computation Conferences. Berlin: Springer, 2001: 469-476.

[5] KENNEDY J. Probability and dynamics in the particle swarm[C]// Proc of Congress on Evolutionary Computation. Washington DC: IEEE Press, 2004: 340-347.

[6] RIGET J, VESTERSTROM J S. A diversity-guided particle swarm optimizer: the ARPSO [R]. Arhus, Denmark: Department of Computer Science, University of Aarhus: 2002.

[7] SUN Jun, XU Wen-bo, FENG Bin. A global dearch dtrategy of quantum-behaved particle swarm optimization[C]//Proc of IEEE Conference on Cybernetics and Intelligent Systems. [S. l.]: IEEE Press, 2004: 291-294.

[8] SUN Jun, XU Wen-bo, FANG Wei. Quantum-behaved particle swarm optimization with a hybrid probability distribution[C]//Proc of the 9th Pacific Rim International Conference on Artificial Intelligence. Berlin: Springer, 2006: 737-746.

[9] SUN Jun, XU Wen-bo, FANG Wei. A diversity-guided quantum-behaved particle swarm optimization algorithm[C]//Proc of IEEE International Conference on Simulated Evolution and Learning. Washington DC: IEEE Press, 2006: 497-504.

[10] LIU Jing, SUN Jun, XU Wen-bo. Improving quantum-behaved particle swarm optimization by simulated annealing[C]//Proc of International Conference on Intelligent Computing. Berlin: Springer, 2006: 130-136.

[11] COELHO L S. Novel Gaussian quantum-behaved particle swarm optimizer applied to electromagnetic design [J]. Science, Measurement & Technology. 2007, 11(5): 290-294.

[12] LIU Jing, SUN Jun, XU Wen-bo, *et al*. Quantum-behaved particle swarm optimization with immune memory and vaccination[C]//Proc of IEEE International Conference on Granular Computing. Piscataway: IEEE Press, 2006: 453-456.

[13] CLERC M, KENNEDY J. The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space [J]. IEEE Trans on Evolutionary Computation, 2002, 6(1): 58-73.

[14] SUN Jun, XU Wen-bo, FENG Bin. Adaptive parameter control for quantum-behaved particle swarm optimization on individual level [C]//Proc of IEEE International Conference on Systems, Man and Cybernetics. Piscataway: IEEE Press, 2005: 3049-3054.

[15] YAO Xin, LIU Yong, LIN Guang-ming. Evolutionary programming made faster [J]. IEEE Trans on Evolutionary Computation, 1999, 3(2): 82-88.

[16] SUN Jun, XU Wen-bo, FANG Wei. A diversity-guided quantum-behaved particle swarm optimization algorithm[C]//Proc of IEEE International Conference on Simulated Evolution and Learning. Washington DC: IEEE Press, 2006: 497-504.