文章编号: 1001-0920 (2014) 10-1881-04 **DOI:** 10.13195/j.kzyjc.2013.0915

带两类正态变异的PSO算法

高圣国[a], 刘 升[a], 郑中团[b]

(上海工程技术大学 a. 管理学院，b. 基础学院，上海 201620)

摘 要: 针对基本粒子群优化算法 (PSO) 容易陷入局部最优点和收敛速度较慢的缺点, 提出在PSO更新过程中加入两类基于正态分布投点的变异操作. 一类变异用来增强局部搜索能力, 另一类变异用来提高发现全局最优点的能力, 避免所有粒子陷入到一个局部最优点的邻域内. 数值结果表明, 所提出算法的全局搜索能力有显著提高, 并且收敛速度更快.

关键词: 粒子群优化算法；变异；正态分布；全局收敛

中图分类号: TP18 **文献标志码:** A

Improved PSO with two mutations based on normally throwing points distribution

GAO Sheng-guo[a], *LIU Sheng*[a], *ZHENG Zhong-tuan*[b]

(a. School of Management，b. School of Fundamental Studies，Shanghai University of Engineering Science，Shanghai 201620，China．Correspondent：GAO Sheng-guo，E-mail：gsg1688@163.com)

Abstract: The basic particle swarm optimization algorithm(PSO) is easy to fall into local minima and convergence slowly, so an improved PSO algorithm with two mutations based on normally throwing distribution points in the updating process is presented. One of the mutations is used to enhance the local searching ability, the other is used to increase the ability of finding the global optimum and avoid all particles falling into a neighborhood of a local minima. Experimental results show that the global search capability of the proposed algorithm is improved significantly and the convergence speed is faster.

Key words: particle swarm optimization；mutation；normal distribution；global convergence

0 引 言

自然界中有些动物在寻找食物时不仅依靠自身的探测能力, 同时还能借助其他动物的行为来引导自己的寻找方向. 受这种现象的启发, Kennedy 等[1]于1995年提出了一种进化计算技术——粒子群优化算法(PSO), 其基本思想源于对鸟群觅食行为的模拟. 粒子可以看作一个寻优的主体, 在寻优过程中, 有两个因素决定下一步的方向, 即它自己曾经到达过的最优位置和所有粒子所达到的最优位置. 同时, 每个粒子也可以看作是最优问题的一个备选解, 所有粒子按某些规则移动一次, 得到更多的备选解, 从而得到更好的群体最优解.

PSO算法的优势在于算法的简洁性, 易于实现, 并具有调整参数较少的优点, 是解决非线性连续优化问题的有效工具[2]. 对于单调函数、严格凸和单峰函数, PSO算法能在初始时很快向最优值行进, 但在最优值附近收敛缓慢. 对于多峰函数, 则容易陷入局部收敛, 不论计算多少步都无法逃出[3]. PSO算法在计算高维问题时效率通常下降得较快. 大量的文献从各个方面对基本的PSO算法进行了改进, 其中QPSO (quantum-Behaved particle swarm optimization)[4-5]算法具有独特性. QPSO算法赋予粒子波的特性, 在下一个时刻, 每个粒子按一定概率可能处在任何位置. 粒子的具体位置使用Monte Carlo随机投点的方法确定. QPSO算法具有启发性, 因为新粒子的产生突破了PSO算法更新过程中矩形体的限制, 在维数变大时具有明显的优势.

本文受QPSO算法的启发, 提出一个带两类正态变异的PSO (N-PSO) 算法. N-PSO算法保留PSO算法的更新过程, 在PSO算法更新过程结束后, 让两类粒

收稿日期: 2013-07-08；**修回日期:** 2013-09-29.

基金项目: 国家自然科学基金项目(61075115/F030707)；国家自然科学基金青年基金项目(11101265/A0107)；上海市教委高校青年教师培养计划项目(shgcjs020).

作者简介: 高圣国(1968−), 男, 讲师, 博士, 从事最优化算法的研究；刘升(1966−), 男, 教授, 博士, 从事智能算法等研究.

子发生波动变异, 其中一类只有1个粒子, 是当前的最优粒子, 让它按正态分布波动(即以它为中心按正态分布投点), 产生若干个新粒子, 如果有比它更好的粒子产生, 则取代它; 另一类是少量(例如5个)适应值较差(通常是适应值较大)的粒子, 让它们也按正态分布波动(它们使用相同的方差, 但与第1类方差不同), 各自产生1个新粒子直接替代原来的粒子. 第1类变异用来在局部区域内找到更好的解, 第2类变异用来避免所有粒子被限制在当前区域内. 实验结果表明, 与基本PSO算法和QPSO算法相比, 本文算法的全局收敛能力得到显著提高, 能有效改善PSO算法过早收敛到局部最优点的问题.

1 标准的PSO算法和QPSO算法

标准粒子群算法[2]中第i个粒子有两个关键属性, 当前时间的位置$X_i(t)$和速度$V_i(t)$, 每个粒子还储存自己经历的最优点$P_i(t)$, 同时还知道群体的最优点$P_g(t)$. 这里群体最优可以是全局的, 也可以是局部的, 本文中使用全局最优的含义.

粒子群算法中的更新公式如下:

$$V_i(t+1)=\omega(t)V_i(t)+c_1r_1(P_i(t)-X_i(t))+ c_2r_2(P_g(t)-X_i(t)), \tag{1}$$

$$X_i(t+1)=X_i(t)+V_i(t+1). \tag{2}$$

其中: $V_i(t+1)$为第$t+1$代时第i个粒子的速度, $X_i(t+1)$为第$t+1$代时第i个粒子的位置, $\omega(t)$为惯性因子, 是t的线性递减函数, 如$\omega(t+1)=0.9-0.5t/T$, T[6]为最大的运算步数; c_1、c_2,为常数, 称为学习因子, 在算法实现时, 通常在2附近取相等的值, 如$c_1=c_2=2$; r_1、r_2为(0,1)之间的随机数, 服从均匀分布.

粒子群算法只有$\omega(t)$、c_1和c_2三个参数, 即容易控制, 改进算法便是用不同的策略调整这些参数在迭代过程中的取值. QPSO算法[5-6]不使用速度属性, 认为粒子可以出现在空间中的任何位置, 在一个特定位置上出现的概率由一个分布函数确定, 利用Monte Carlo方法可以在第t步确定第$t+1$步的具体位置.

QPSO算法引入的中心点

$$m_{\rm best}=\frac{1}{M}\sum_{i=1}^{M}P_i$$

即为平均的最好位置. 更新过程采用如下公式:

$$P_i(t+1)=\varphi P_i(t)+(1-\varphi)P_g(t), \tag{3}$$

$$X_i(t+1)=P_i(t+1)\pm\alpha|m_{\rm best}-X_i(t)|\ln(1/u). \tag{4}$$

其中: φ和u为随机变量, 均服从(0,1)上的一致分布; 取+和−的概率都是0.5; α为唯一的参数, 称为创新系数, 在文献[4]中设定为由1～0.5递减取值.

经典的PSO算法虽然实现简单, 但容易收敛到局部最优, 这一不足在维数变大时更为突出. 当问题的维数变大时, QPSO算法具有明显的优势.

2 N-PSO算法

PSO算法的本质体现在式(1)中, 从形式上看, 它是一个线性表达式, 由于式中含有两个随机变量, 本质上讲它是非线性的, 正是该特性极大地提高了PSO算法的搜索能力. 如果将该随机变量去掉, 其搜索能力会大大减弱. QPSO算法完全抛弃了式(1)和速度变量, 对每个粒子的位置更新采用一种随机的投点方法进行决定. 比较式(2)和(4)容易看出, QPSO算法增加了粒子更新的随机性, 突破了式(1)和(2)的矩形框架, 从而在一次更新过程里可以在更大的范围内按非均匀的分布随机取点。

结合上述两种思想, N-PSO算法的基本想法是将粒子的更新过程分为两个阶段: 第1阶段与PSO算法相同, 按式(1)和(2)产生新一代的粒子, 完成后不是立刻进行下一次循环, 而是进入第2阶段; 第2阶段按适应值对粒子进行排序, 排在第1的是$P_g(t)$, 即到第t步为止所获得的最优解, 它所对应的粒子位置是$X_g(t)$. 以$X_g(t)$为中心按正态分布(方差为$\sigma_1 I$, I是问题空间的单位矩阵, 这个正态分布是对称的, σ_1是一个数)在它周围空间投m个点(如20个), 在这些点(包含$X_g(t)$)中找出适应值最好的那个点作为新的$X_g(t)$, 如果找不到更好的, 则$X_g(t)$暂时不变. 在第2阶段还同时进行另一操作, 即是在按适应值排序处在最后的m_1个点($P'_N(t)$, $P'_{N-1}(t)$, $\cdots$, $P'_{N-m_1+1}(t)$)也进行类似$P_g(t)$的操作, 不同的是正态分布的方差是σ_2(σ_2是正定对角矩阵, 其对角线上的分量一般大于σ_1, 这时的正态分布不一定对称), 且每次仅产生一个新点, 直接替代原来的$P'_N(t)$、$P'_{N-1}(t)$等中心点.

N-PSO算法步骤如下.

Step 1: 初始化. 令$t=1$, 在问题空间中随机产生N个粒子的位置$X_i(t)$和粒子的速度$V_i(t)$, 计算适应值$f_i(t)$, $f_i(t)=f(X_i(t))$, 确定$P_i(t)$和$P_g(t)$, 其中$1\leqslant i\leqslant N$.

Step 2: 按式(1)和(2)更新$V_i(t)$和$X_i(t)$, 得到$V_i(t+1)$和$X_i(t+1)$, 同时更新$P_i(t+1)$和$P_g(t+1)$.

Step 3: 以$P_g(t+1)$为中心, 按方差为$\sigma_1 I$的正态分布投点, 即取随机变量$Y\sim N(P_g(t+1),\sigma_1 I)$的$m$个随机点, 选择$m$个随机点和$P_g(t+1)$中适应值最好的一个替代$P_g(t+1)$. 其中: σ_1为动态参数, I为单位向量.

Step 4: 分别以适应值最差的m_1个粒子$P'_N(t)$, $P'_{N-1}(t)$, $\cdots$, $P'_{N-m_1+1}(t)$为中心, 按方差为σ_2的正

态分布投点, 即在 m_1 个随机变量 $Z \sim N(P_i'(t+1), \sigma_2)$ 中各取一个随机点代替原粒子.

Step 5: 更新 $P_i(t+1)$ 和 $P_g(t+1)$, 对于变异产生的新粒子 $P_j(t+1)$, 按下式更新速度:

$$V_j(t+1) = P_g(t+1) - P_j(t+1). \tag{5}$$

如果 $t \leqslant T$, 转至 Step 2, 否则计算终止.

N-PSO 算法本质上是在 PSO 算法中插入两个产生新粒子的过程, 新粒子的产生使用 QPSO 算法波的思想, 其每一维的坐标都是投点产生的, 且每个粒子都使用这个方法产生下一代粒子. N-PSO 算法在每一次更新时将粒子分为 3 类, 第 1 类粒子只有 1 个, 即 $P_g(t)$, 在其周围多次投点, 所用正态分布的方差 σ_1 较小, 目的是直接找到更好的替代粒子, 增强算法的局部寻找能力; 第 2 类粒子不只 1 个, 但不超过总粒子数的一半, 针对每个粒子只投点一次, 所用正态分布的方差 σ_2 的分量值较大, 目的是拓展粒子的范围, 避免过早收敛到局部最优点; 其他的粒子均为第 3 类粒子, 其行为完全按 PSO 算法进行更新. 需要注意的是, 粒子没有固定分类, 一个粒子在第 t 步时是某类粒子, 在 $t+1$ 步时可能会变成另外一种类型.

形象地说, PSO 算法的执行过程就像抛出去一张渔网, 在一个动态最优点的拉力下这张渔网逐渐向最优点收拢. 整个过程中有两个重要指标: 1) 动态最优点的收敛速度, 这是局部搜索能力; 2) 整个渔网跨越的范围, 这是全局搜索能力. N-PSO 算法通过引入两类容易实现的正态投点产生不同用途的新粒子, 对经典的 PSO 算法的局部搜索能力和全局搜索能力均有较大的提高.

3　数值实验分析

用于测试 PSO 算法的函数较多, 这里选择 CEC'05 测试函数.

1) Sphere 函数

$$f_1(x) = \sum_{i=1}^{n} x_i^2,\ x_i \in [-100, 100]. \tag{6}$$

2) Rosenbrock 函数

$$f_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2],$$
$$x_i \in [-100, 100]. \tag{7}$$

3) Rastrigin 函数

$$f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10],$$
$$x_i \in [-5, 5]. \tag{8}$$

4) Griewank 函数

$$f_4(x) = \frac{1}{4\,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$$
$$x_i \in [-600, 600]. \tag{9}$$

5) Goldstein-Price 函数

$$f_5(x) = $$
$$[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 -$$
$$14x_2 + 6x_1x_2 + 3x_2^2)](18 - 32x_1 + 12x_1^2 -$$
$$48x_2 - 36x_1x_2 + 27x_2^2),\ x_i \in [-2, 2]. \tag{10}$$

6) J.D.Schaffer 函数

$$f_6(x) = \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} - 0.5,$$
$$x_i \in [-100, 100]. \tag{11}$$

N-PSO 算法实现时, 式 (1) 和 (2) 中: $\omega_{t+1} = 0.9 - 0.5t/T$, T 为最大的运算步数; $c_1 = c_2 = 2$; σ_2 取每维宽度的 1/6. 例如, 求解空间为 $[-30, 30] \times [0, 30]$, 则

$$\sigma_2 = \begin{bmatrix} 10 & 0 \\ 0 & 5 \end{bmatrix}.$$

σ_2 的取值决定了算法逃离局部最优点的能力, 一般而言, 取每维宽度的 1/30 ~ 1/5 较为合适. σ_1 的取值复杂一些, 在将粒子按适应值排序后, 排在第 1 的为 $X_1'(t)$, 排在第 3 的为 $X_3'(t)$ (这里不使用 $X_2'(t)$ 的原因是, $X_1'(t)$ 与 $X_2'(t)$ 太近会导致 σ_1 过小), 则 σ_1 等于 $X_1'(t) - X_3'(t)$ 坐标分量绝对值最大值的 1/6, 如果计算出来的 σ_1 大于 σ_2 中的最大分量, 则用 σ_2 中的最大分量除以 50 代替 σ_1. 另外, $m = 20$, $m_1 = 5$, 即第 1 类变异时产生 20 个随机点, 第 2 类变异对 5 个粒子使用.

f_5 和 f_6 是 2 维函数, 文献 [7] 利用其来比较 PSO 算法和改进后的 SPSO1 算法、SPSO2 算法的性能, 这里引入它们作为 N-PSO 算法的比较对象, 对每个函数计算 50 次. 由于 N-PSO 算法能在较少的步数内达到最优值, 无需使用误差设定 (即误差为 0), 经过不超过 50 步的计算可以达到最优值, 计算结果如表 1 所示.

表 1　f_5 和 f_6 的计算结果

函数	算法	误差	收敛率/%	平均收敛步数
f_5	PSO	1.0e-5	100	186.9
	SPSO1	1.0e-5	88	11.9
	SPSO2	1.0e-5	100	18.5
	N-PSO	**0**	**100**	**10.2**
f_6	PSO	0.1	76	21.97
	SPSO1	0.1	52	6.81
	SPSO2	0.1	96	72.96
	N-PSO	**0**	**100**	**19.4**

对函数 f_6 进行 50 次计算过程中, 每步产生的最优点适应值图像如图 1 所示. 由图 1 可见, 在 26 步之

前适应值全部达到最优值 -1, 这时 x_1 和 x_2 在 10^{-7} 左右或者更小.

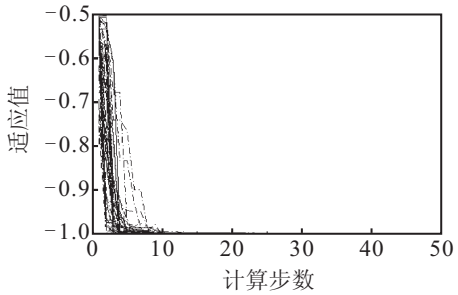

图 1 f_6 的 50 次计算的适应值收敛过程

表 2 50 次计算的平均最优适应值和方差

f	PSO		QPSO		N-PSO	
	Mean	STD	Mean	STD	Mean	STD
f_1	1.04e+1	1.71e+1	1.32e-23	3.32e-23	**0**	**0**
f_2	2.08e+2	1.33e+2	9.03e+01	1.33e+02	**6.62e+0**	**1.20e+1**
f_3	1.37e+2	5.63e+0	1.57e+01	5.63e+00	**2.99e-1**	**9.44e-1**
f_4	3.66e+1	1.94e-2	1.88e-02	1.94e-02	**2.68e-2**	**1.52e-2**

对于函数 $f_1 \sim f_4$, 表 2 给出了空间维数是 20 维、计算步数是 1 500、独立计算 50 次的结果. PSO 算法和 QPSO 算法的结果均来自文献 [4-6], 除了 f_4 的结果略差一点外, 其他函数都有很大的提高. 例如, 在对 f_3 的 50 次计算中, 有 41 次达到最优解 0; f_1 的计算中, 有 15 次没有达到 0, 但适应值均在 10^{-323} 以下. 如果将 f_2 的计算步数改为 2 000, 则平均值可以达到 4.997 0e - 004, 标准差为 8.262 2e - 004, 且其中 60% 的适应值在 10^{-10} 以下, 30% 的适应值达到 10^{-26} 以下.

这里仅给出一部分计算结果, 对其他函数进行计算的结果 (包括 σ_1、σ_2、m 等参数的不同选择) 也表明, N-PSO 算法具有很强的跳出局部最优点的能力, 且收敛速度较快, 可以认为这受益于插入到 PSO 算法中的两类变异.

4 结 论

针对基本 PSO 算法容易陷入局部极值且局部寻优能力不强的缺点, 提出在迭代过程中增加两类变异操作, 分别用于提高算法的局部寻优能力和逃离局部最优点的能力. 数值结果表明, 改进后的算法较大地提高了寻找全局最优的能力, 同时收敛速度更快. 未来的工作是研究如何根据不同问题自动调节两个正态分布的方差, 从而使算法具有更高的效率和更好的适应性.

参考文献(References)

[1] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. New York: IEEE Press, 1995: 1942-1948.

[2] Maurice Clerc. Standard particle swarm optimization[EB/OL]. (2012-09-23)[2013-04-01]. http://hal.archives-ouvertes.fr/hal-00764996.

[3] 罗辞勇, 陈民铀. 克服恋食行为的 PSO 算法改进研究[J]. 控制与决策, 2008, 23(7): 776-780.
(Luo C Y, Chen M Y. Improved PSO algorithm with overcoming behaviour of indulging in food[J]. Control and Decision, 2008, 23(7): 776-780.)

[4] Sun J, Xu W B, Feng B. A global search strategy of quantum-behaved particle swarm optimization[C]. Proc of IEEE Conf on Cybernetics and Intelligent Systems. Singapore, 2004: 111-116.

[5] Li Yang-yang, Xiang Rong-rong, Jiao Li-cheng, et al. An improved cooperative quantum-behaved particle swarm optimization[J]. Soft Computing, 2012, 16(6): 1061-1069.

[6] Ma Gang, Zhou Wei, Chang Xiaolin. A novel particle swarm global optimization algorithm based on particle migration[J]. Applied Mathematics and Computation, 2012, 218(11): 6620-6626.

[7] 曾建潮, 崔志华. 一种保证全局收敛的 PSO 算法[J]. 计算机研究与发展, 2004, 41(8): 1333-1338.
(Zeng J C, Cui Z H. A guaranteed global convergence particle swarm optimizer[J]. J of Computer Research and Development, 2004, 41(8): 1333-1338.)

(责任编辑: 郑晓蕾)