文章编号: 1001-0920(2008)01-0095-04

一种弹性粒子群优化算法

李勇刚[1], 桂卫华[1], 阳春华[1], 陈志盛[2]
(1. 中南大学 信息科学与工程学院, 长沙 410083; 2. 长沙理工大学 能源与动力工程学院, 长沙 410076)

摘 要: 当某个粒子与最优粒子很接近时,其飞行速度将趋于零,这是粒子群优化算法容易陷入局部极小的主要原因. 为此,提出一种弹性粒子群优化算法. 算法中,粒子速度不依赖其与最优粒子之间距离的大小,而仅依赖于其方向信息,并采用一种自适应策略弹性地修正粒子速度的幅值. 将弹性粒子群优化算法应用于几种典型测试函数的优化,数值仿真结果表明,弹性粒子群优化算法能有效地找出全局最优点.

关键词: 粒子群优化算法; 弹性修正; 全局最优点
中图分类号: TP18 **文献标识码**: A

A resilient particle swarm optimization algorithm

LI Yong-gang[1], *GUI Wei-hua*[1], *YANG Chun-hua*[1], *CHEN Zhi-sheng*[2]
(1. School of Information Science and Engineering, Central South University, Changsha 410083, China; 2. School of Energy and Power Engineering, Changsha University of Science and Technology, Changsha 410076, China. Correspondent: LI Yong-gang, E-mail: liyonggang@mail.csu.edu.cn)

Abstract: When an individual is closed to the optimal particle, its velocity will approximate to zero. This is the main reason why particle swarm optimization (PSO) algorithm is prone to trap into local minima. A resilient particle swarm optimization (RPSO) is proposed, in which the velocity of an individual is not dependent on the size of distance between the individual and the optimal particle but only dependent on its direction. An adaptive scheme is adopted to adjust the magnitude of the velocity resiliently. Finally, RPSO is applied to optimize several test functions. Simulation results show that RPSO can find global optima effectively.

Key words: Particle swarm optimization algorithm; Resilient adjustment; Global optima

1 引 言

粒子群优化(PSO)算法是由 Kennedy 和 Eberhart 于 1995 年提出的一种进化算法[1],它源于对鸟类捕食行为的模拟. PSO 算法概念简单、容易实现且优化性能良好,因此受到国内外学者的广泛关注. 在基本 PSO 算法的基础上,目前已提出了多种改进算法. 文献[2,3]分别采用线性递减和模糊自适应方法动态优化惯性因子,提高了算法的全局搜索能力和搜索精度;Clerc[4]提出一种带收缩因子的 PSO 算法,改善了算法的收敛性;文献[5]借鉴遗传算法中的自然选择机制,提高了算法的局部搜索能力.

与其他进化算法一样,PSO 算法存在早熟收敛问题. 对此,人们进行了大量的研究工作[6-8]. 文献[6]采用变异算子,以维持群体的多样性并跳出局部最优;文献[7]在 PSO 算法中利用梯度信息,不仅加快了寻优速度,而且可以避免局部极小;文献[8]采用拉伸技术改变目标函数,同样可以避免出现局部最优解. 这些方法在一定程度上改善了 PSO 算法的全局优化能力,但也存在一些不足. 例如:采用变异算子的算法中,变异概率的选取是一个难题;利用梯度信息的方法则要求优化目标函数是可微的,但实际应用中很多优化目标函数是不可微的;采用拉伸技术改变目标函数的方法实现起来比较复杂.

造成 PSO 算法早熟收敛的主要原因是:在算法后期,所有粒子将聚集到一个极值点附近,各粒子的速度将趋于零. 如果这是一个局部极值点,则所有粒子将很难跳出其约束范围,从而导致算法早熟收敛.

收稿日期: 2006-10-11; **修回日期**: 2007-01-09.
基金项目: 国家 973 计划项目(2002CB312200); 国家自然科学基金项目(60634020,60574030); 国家 863 计划项目(2006AA04Z181).
作者简介: 李勇刚(1974—),男,长沙人,副教授,博士,从事复杂过程建模与优化控制的研究; 桂卫华(1950—),男,武汉人,教授,博士生导师,从事大系统理论、鲁棒控制等研究.

为此,本文提出一种弹性粒子群优化算法(RPSO).该算法在计算各粒子的速度时,不考虑它与最优粒子距离之间的大小,而只利用其方向信息,并采用一种自适应策略弹性地修正粒子速度的幅值.这样能保证各粒子朝最优粒子飞行,而且当它与最优粒子很接近时,仍能保持一定的飞行速度.因此,如果找到一个局部极值点,其他粒子能比较容易地跳出其约束范围,以避免算法的早熟收敛.实验结果表明,与 GA 和带变异的 PSO 相比,RPSO 的全局搜索能力有了较大的改善.

2 粒子群优化算法

与遗传算法类似,PSO 算法同样基于群体与适应度.PSO 算法随机地初始化为目标函数的一个解群体,群体中的每个个体称为一个粒子.每个粒子模仿鸟类的觅食行为,通过跟踪两个"极值"来实现在搜索空间寻找最优解的目的:一个是每个粒子当前已搜索到的最优位置(适应度最大),称为个体极值;另一个是整个粒子群当前已搜索到的最优位置,称为全局极值.PSO 算法可描述如下:

假设在 D 维搜索空间有 m 个粒子,第 i 个粒子在搜索空间的位置用向量 $X_i = [x_{i1}, \cdots, x_{iD}]^{\mathrm{T}}$ 表示,其个体极值记为 $P_i = [p_{i1}, \cdots, p_{iD}]^{\mathrm{T}}$,而全局极值记为 $P_g = [p_{g1}, \cdots, p_{gD}]^{\mathrm{T}}$.在迭代过程中,第 i 个粒子以速度 $V_i = [v_{i1}, \cdots, v_{iD}]^{\mathrm{T}}$ 在搜索空间飞行.每个粒子的飞行速度及位置按下式进行修正:

$$\begin{aligned} v_{id}(t+1) = {} & \\ & w * v_{id}(t) + c_1 * r_1 * [p_{id}(t) - x_{id}(t)] + \\ & c_2 * r_2 * [p_{gd}(t) - x_{id}(t)], \\ & 1 \leq i \leq m,\ 1 \leq d \leq D; \end{aligned} \tag{1}$$

$$\begin{aligned} x_{id}(t+1) = {} & x_{id}(t) + v_{id}(t+1), \\ & 1 \leq i \leq m,\ 1 \leq d \leq D. \end{aligned} \tag{2}$$

其中:c_1,c_2 为正常数,称为加速因子,通常取 $c_1 = c_2 = 2.0$;r_1,r_2 为 $[0, 1]$ 之间的随机数;w 为惯性因子,一般取 0.1 ~ 0.9 之间的数.在迭代过程中,粒子的速度向量被限制在 $[-V_{\max}, V_{\max}]$ 范围内,以降低粒子飞出搜索空间的概率;而粒子的位置向量被限制在 $[X_{\min}, X_{\max}]$ 范围内.$X_{\min}$ 和 $X_{\max}$ 由实际问题决定,$V_{\max}$ 通常选为 $k \cdot X_{\max}$,其中 $0.1 \leq k \leq 1.0$[9].

3 弹性粒子群优化算法

3.1 算法描述

由式(1)可以看出,在算法后期,粒子群中所有粒子都聚集到一个极值点附近.此时,p_{id},p_{gd} 与 x_{id} 相差很小,而 w 一般取值范围为 $[0.1, 0.9]$,因此粒子的飞行速度 v_{id} 将趋于 0.如果该极值是一个局部极值点,则粒子很难跳出其约束范围,直至所有粒子都集中到该局部极值点上,因而引起算法的早熟收敛.

为改善 PSO 算法的全局搜索能力,必须避免 PSO 算法后期粒子的飞行速度过小.在本文提出的弹性粒子群优化算法中,粒子飞行的方向与原算法一样,都是指向两个"极值".但其飞行速度的大小与基本粒子群优化算法不一样,它不是由该粒子与最优粒子之间距离的大小决定,而是采用一种自适应策略弹性地修正,即

$$\begin{aligned} v_{id}(t+1) = {} & w * v_{id}(t) + c_1 * r_1 * \Delta v_{id}^1(t) + \\ & c_2 * r_2 * \Delta v_{id}^2(t). \end{aligned} \tag{3}$$

其中

$$\begin{cases} \Delta v_{id}^1(t) = \Delta_{id}^1(t) \cdot \mathrm{sgn}[p_{id}(t) - x_{id}(t)], \\ \Delta v_{id}^2(t) = \Delta_{id}^2(t) \cdot \mathrm{sgn}[p_{gd}(t) - x_{id}(t)], \end{cases} \tag{4}$$

$\mathrm{sgn}(\cdot)$ 为符号函数.$\Delta_{id}^1(t)$ 和 $\Delta_{id}^2(t)$ 采用如下自适应策略弹性地修正:当粒子前后两次飞行方向一致时,说明粒子正从个体"极值"(或全局"极值")的一侧向其逼近,因此可以加大 $\Delta_{id}^1(t)$(或 $\Delta_{id}^2(t)$),以加快算法的收敛速度;如果前后两次飞行的方向不一致,则说明该粒子正在个体"极值"(或全局"极值")附近徘徊,此时应降低 $\Delta_{id}^1(t)$(或 $\Delta_{id}^2(t)$),以避免算法在极值点附近徘徊过久;如果粒子前后两次迭代有一次与个体"极值"(或全局"极值")重合,则不改变 $\Delta_{id}^1(t)$(或 $\Delta_{id}^2(t)$).于是,可按下式修正 $\Delta_{id}^1(t)$ 和 $\Delta_{id}^2(t)$:

$$\Delta_{id}^1(t) = \begin{cases} \eta^+ \Delta_{id}^1(t-1), \text{ if } [p_{id}(t-1) - x_{id}(t-1)] * \\ \qquad [p_{id}(t) - x_{id}(x)] > 0; \\ \eta^- \Delta_{id}^1(t-1), \text{ if } [p_{id}(t-1) - x_{id}(t-1)] * \\ \qquad [p_{id}(t) - x_{id}(t)] < 0; \\ \Delta_{id}^1(t-1), \text{ others}; \end{cases} \tag{5a}$$

$$\Delta_{id}^2(t) = \begin{cases} \eta^+ \Delta_{id}^2(t-1), \text{ if } [p_{gd}(t-1) - x_{gd}(t-1)] * \\ \qquad [p_{gd}(t) - x_{gd}(t)] > 0; \\ \eta^- \Delta_{id}^2(t-1), \text{ if } [p_{gd}(t-1) - x_{gd}(t-1)] * \\ \qquad [p_{gd}(t) - x_{gd}(t)] < 0; \\ \Delta_{id}^2(t-1), \text{ others}. \end{cases} \tag{5b}$$

其中:η^- 和 η^+ 分别称为递减因子和递增因子,它们满足 $0 < \eta^- < 1 < \eta^+$.同时,为了保证算法的收敛性和快速性,$\Delta_{id}^1(t)$ 和 $\Delta_{id}^2(t)$ 都必须限定在一个合理的范围内,即 $\Delta_{id}^1(t)$,$\Delta_{id}^2(t) \in [\Delta_{\min}\ \ \Delta_{\max}]$.

采用上面的策略,即使在某个粒子与最优粒子

相距很近时,其飞行速度也不会太小. 这样,如果所有粒子被一个局部极值点吸引至其附近时,一些粒子可以跳出其约束范围,有可能找到更优的位置,从而提高算法寻找全局最优解的概率.

3.2　算法的实现

按 RPSO 的思想,可以给出其具体算法如下:

Step1: 依据实际问题确定粒子位置及飞行速度的边界 $[X_{\min}\quad X_{\max}]$ 和 $[-V_{\max}\quad V_{\max}]$. 设定最大迭代次数,随机初始化 m 个粒子的位置 X_i 及飞行速度 $V_i(i=1,2,\cdots,m)$,并计算出相应的适应度 F_i. 设定参数 w,$\Delta_{id}^1(0)$,$\Delta_{id}^2(0)$,η,η^+,$\Delta_{\min}^d$ 和 $\Delta_{\max}^d$,同时令 $p_{id}(0)=p_{gd}(0)=x_{id}(0)$.

Step2: 令粒子 $i(i=1,2,\cdots,m)$ 当前的最优位置为 $P_i=X_i$,对应的适应度为 $F_{\text{best}_i}=F_i$;并从粒子群中找出全局最优粒子,令其位置为 P_g,对应的适应度为 G_{best}.

Step3: 按式(5) 修正 Δ_{id}^1 和 Δ_{id}^2,并将其限定在范围 $[\Delta_{\min}^d\quad\Delta_{\max}^d]$ 内.

Step4: 按式(4) 计算 Δv_{id}^1 和 Δv_{id}^2.

Step5: 对所有粒子执行如下操作:

1) 按式(3) 修正粒子飞行速度,并将其限定在范围 $[-V_{\max}\quad V_{\max}]$ 内;

2) 按式(2) 修正粒子位置,并将其限定在范围 $[X_{\min}\quad X_{\max}]$ 内,同时计算其适应度 F_i;

3) 如果 $F_i>F_{\text{best}_i}$,则令 $P_i=X_i$,$F_{\text{best}_i}=F_i$;

4) 如果 $F_i>G_{\text{best}}$,则令 $P_g=X_i$,$G_{\text{best}}=F_i$.

Step6: 如果最优适应度几乎不再变化或达到最大迭代次数,则停止;否则,跳至 Step3.

3.3　参数的选择

与基本 PSO 算法相比,弹性粒子群优化算法还有另外几个参数对其性能影响较大. 主要包括: $\Delta_{id}^1(0)$,$\Delta_{id}^2(0)$,η,η^+,$\Delta_{\min}$ 以及 $\Delta_{\max}$.

首先,将所有的 $\Delta_{id}^1(0)$,$\Delta_{id}^2(0)$ $(i=1,2,\cdots,m;\ d=1,2,\cdots,D)$ 都初始化为 $\Delta(0)$. $\Delta(0)$ 的值将直接影响算法的收敛性和收敛速度:如果 $\Delta(0)$ 过小,则其收敛速度很慢;如果 $\Delta(0)$ 过大,则可能导致优化算法难以收敛. 实验表明,$\Delta(0)$ 取值范围为 $[0.01*\Delta x,0.1*\Delta x]$ 比较合适,其中 Δx 为优化变量的上、下限之差. 另外,将 $\Delta_{id}^1(t)$ 和 $\Delta_{id}^2(t)$ 限制在同一范围内,即上限为 $\Delta_{\max}=0.5*\Delta x$,下限为 $\Delta_{\min}=0.001*\Delta x$,以保证算法的收敛性及收敛速度.

递减因子 η 的选择:如果前后两次飞行方向不同(一正一负),则说明前一次已"飞过"极值点,因此应适当降低飞行速度,即 $0<\eta<1$. 如果 η 太大,则由于两次飞行方向相反,可能使粒子又重新飞到原来的位置;但如果 η 太小,又可能导致粒子飞行速度太慢而降低收敛速度. 因此,一种折衷的办法是取 0 和 1 的平均值,即 $\eta=0.5$,大量的仿真结果也表明 $\eta=0.5$ 效果比较好.

递增因子 η^+ 是为了加快算法的收敛速度,如果 η^+ 太小,则加速效果不明显;如果 η^+ 太大,又可能使粒子很快"飞过"极值点,从而导致粒子在极值点附近来回摆动,降低飞行速度. 通过仿真研究表明,当 η^+ 取 1.1 时效果较好.

4　仿真研究

为了测试 RPSO 的性能,本文利用 Rastrigrin, Rosenbrock 及 Griewank 三个测试函数进行仿真研究,并将 RPSO 算法与遗传算法(GA),基本 PSO 算法及带变异的 PSO 算法(MPSO) 进行了比较.

3 个测试函数分别为

$$f_1=\sum_{i=1}^{10}[x_i^2-10\cos(2\pi x_i)+10],$$

$$f_2=\sum_{i=1}^{9}[100*(x_{i+1}-x_i^2)^2+(x_i-1)^2],$$

$$f_3=\frac{1}{4\,000}\sum_{i=1}^{10}x_i^2-\prod_{i=1}^{10}\cos(x_i/\sqrt{i})+1. \qquad (6)$$

其中:Rastrigrin 函数 f_1 是很难最小化的病态二次函数,它具有一个全局极小点 $x^*=(0,\cdots,0)$,其函数值为 $f(x^*)=0$;Rosenbrock 函数 f_2 是具有大量局部最优点的多峰函数,它也有一个全局极小点 $x^*=(1,\cdots,1)$,其函数值为 $f(x^*)=0$;Griewank 函数 f_3 是一个典型的多峰函数,它具有一个全局极小点 $x^*=(0,0)$,其函数值为 $f(x^*)=0$. 3 个函数变量范围都是 $[-10,10]$.

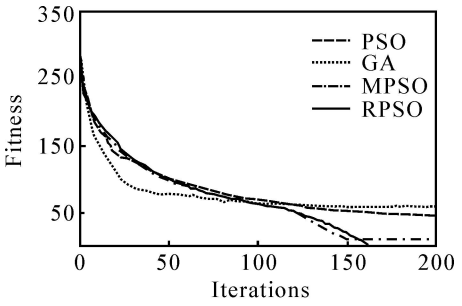
分别利用 4 种算法对上述 3 个函数进行仿真研究. 4 种算法的种群大小均为 50. 每种算法取 3 组不同的参数,每组参数进行 20 次优化运算,其结果如表 1 所示. 表中 M 和 F 分别表示 60 次优化运算能找出全局最优点(寻优结果与全局最优点目标函数值误差在 10^{-4} 以内便认为找到了全局最优点) 的次数和最优目标函数的平均值.

表 1　仿真结果

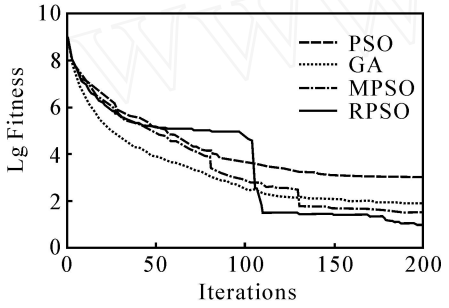
函数	GA		PSO		MPSO		RPSO	
	M	F	M	F	M	F	M	F
f_1	24	65.5	34	51.9	45	20.1	57	4.2
f_2	17	121.4	3	23 592	32	52.3	40	21.4
f_3	14	0.182	20	0.132	34	0.063	48	0.012

图 1 为 4 种算法各选 1 组参数(都是 3 组参数中最好的一组) 进行 20 次仿真的平均结果. 其中: PSO, MPSO 及 RPSO 的 c_1, c_2 均取 2.0; $w=0.8$; $\Delta(0)$, η, η^+ 分别取 1, 0.5 及 1.1. MPSO 中变异概

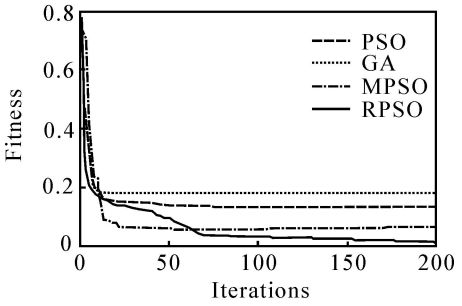
率采用文献[6]的方法取 1/2D(D 为问题的维数). GA 中的 P_c, P_m 分别取 0.8 和 0.1.



(a) Rastrigrin函数进化曲线



(b) Rosenbrock函数进化曲线



(c) Griewank函数进化曲线

图 1 3 个测试函数 20 次寻优平均最佳适应度进化曲线

由仿真结果可以看出,针对 3 个测试函数, GA 和 PSO 算法难以找出它们的全局最优解,因此其平均最优解与理论最优解的差距很大. 而 MPSO 和 RPSO 找到全局最优点的次数明显增多,这表明它们能有效地避免算法陷入局部最优. 显然,RPSO 算法比 MPSO 算法的效果更好.

5 结 语

本文针对 PSO 算法容易陷入局部最小的问题,分析其原因,并提出了一种弹性粒子群优化算法. 实验结果表明,该算法比 PSO 算法和 GA 算法具有更强的全局寻优能力,是一种非常有效的优化算法. 然而,弹性粒子群优化算法中,递增因子 η^+ 和递减因子 η^- 是固定的,而且如何选择合适的值要通过大量的仿真研究才能确定. 如果能在算法中自适应修正 η^+ 和 η^-,将给算法带来极大方便,这是下一步要研究的内容.

参考文献(References)

[1] Eberhart R, Kennedy J. A new optimizer using particle swarm theory[C]. Proc of the 16th Int Symposium on Micro Machine and Human Science. Nagoya, 1995: 39-43.

[2] Eberhart R C, Shi Y. Particle swarm optimization: Developments, applications and resources[C]. Proc of the IEEE Conf on Evolutionary Computation. Soul: IEEE, 2001: 81-86.

[3] Shi Y, Eberhart R C. Fuzzy adaptive particle swarm optimization [C]. Proc of the IEEE Conf on Evolutionary Computation. Soul: IEEE, 2001: 101-106.

[4] Clerc M. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization [C]. Proc of the IEEE Conf on Evolutionary Computation. Washington: IEEE, 1999: 1927-1930.

[5] Angeline P J. Using selection to improve particle swarm optimization [C]. Proc of the IEEE Conf on Evolutionary Computation. New York: IEEE, 1998: 84-89.

[6] 李宁, 刘飞, 孙德宝. 基于带变异算子粒子群优化算法的约束布局优化研究[J]. 计算机学报, 2004, 27(7): 897-903.
(Li Ning, Liu Fei, Sun De-bao. A study on the particle swarm optimization with mutation operator constrained layout optimization[J]. Chinese J of Computers, 2004, 27(7): 897-903.)

[7] Noel M M, Jannett T C. Simulation of a new hybrid particle swarm optimization algorithm[C]. Proc of the 36th Southeastern Symposium on System Theory. Atlanta: IEEE, 2004: 150-153.

[8] Parsopoulos K E, Plagianakos V P, Magoulas G D, et al. Stretching technique for obtaining global minimizers through particle swarm optimization [C]. Proc of the Workshop on Particle Swarm Optimization. Indianapolis, 2001: 22-29.

[9] Bergh F, Engelbrecht A. A new locally convergent particle swarm optimizer[C]. Proc of IEEE Int Conf on Systems, Man and Cybernetics. Tunisia: IEEE, 2002: 96-101.