基于试探的变步长自适应粒子群算法

郑春颖 郑全弟 王晓丹 王玉冰
(空军工程大学导弹学院 三原 713800)

摘 要 针对粒子群算法容易陷入局部最优的缺陷,在分析惯性因子在算法中的作用机理的基础上,设计了一个根据种群多样性和进化代数自适应调节的惯性因子,并运用试探法,通过变换搜索步长,提高算法的局部搜索能力。最后,给出了3个典型函数的模拟例子,通过与APSO的对比结果显示,改进后的算法其性能得到极大提高。

关键词 粒子群算法,惯性因子,进化代数
中图法分类号 TP18 **文献标识码** A

Self-adaptive Particle Swarm OPtimization Algorithm Based on Tentative Adjusting Step Factor

ZHENG Chun-ying ZHENG Quan-di WANG Xiao-dan WANG Yu-bing
(The Missile Institute, Air Force Engineering University, Sanyuan 713800, China)

Abstract Aiming at premature defect and poor result of Particle Swarm Optimization algorithm, a new Self-adaptive inertia factor was designed according to diversity in the population and generation number based on analysing inertia factor's effect of algorithm. And through ploughing around adjusting step factors, the Particle's ability in local searching was enhanced. Three typical function tests were given. Comparing with APSO, the result indicates the effectiveness of this improvement.

Keywords Particle swarm optimization algorithm, Inertia factor, Generation number

1 引言

粒子群优化算法(Particle Swarm Optimization,简称PSO),是Kennedy博士和Eberhart博士于1995年提出的一种新型进化计算技术[1],其基本思想来源于对鸟群简化社会模型的研究,具有典型的群体智能特性,通过利用信息共享机制,使群体中的个体相互借鉴经验以促进群体发展。PSO具有深刻的智能背景,且计算简单,易于实现,已被成功应用于多目标优化、模式识别、信号处理和决策支持等领域[2]。

然而,PSO存在着容易陷入局部极值点,进化后期收敛慢,精度较差等缺点[3]。针对上述PSO存在的缺点,本文在分析PSO进化原理的基础上,对算法进行改进:在分析信息熵与种群的多样性内在联系的基础上,用信息熵和进化代数作为参数构造一种能自适应调节的惯性因子,并采用试探法通过变换步长,选取粒子的最佳落脚点,从而提高了局部搜索能力。最后,仿真实验显示,与标准PSO和APSO相比,本文算法性能更优。

2 标准PSO原理简介[3,4]

PSO的个体是没有体积没有质量的粒子,每个粒子代表一个候选解,具有位置和速度两个特征。从初始群体出发,粒子根据自己和同伴的飞行经验,不断调整位置和速度,从而产生新一代群体。设搜索空间为M维,群体中第i个粒子的空间位置为$X_i(x_{i1},x_{i2},\cdots,x_{iM})$,速度为$V_i(v_{i1},v_{i2},\cdots,v_{iM})$,它经历过的最佳空间位置为$P_i(p_{i1},p_{i2},\cdots,p_{iM})$,群体经历过的最佳空间位置为$P_g(p_{g1},p_{g2},\cdots,p_{gM})$。粒子分别按式(1)和式(2)更新自己的位置和速度:

$$v_{im}^{k+1}=w\cdot v_{im}^{k}+c_1\cdot \mathrm{rand}()\cdot(p_{im}^{k}-x_{im}^{k})+c_2\cdot \mathrm{rand}()(p_{gm}^{k}-x_{im}^{k}) \quad (1)$$

$$x_{im}^{k+1}=x_{im}^{k}+v_{im}^{k} \quad (2)$$

其中,w为惯性因子,k为迭代次数,rand( )为0与1之间变化的随机数,$1\leqslant m\leqslant M$,c_1,c_2为学习因子,通常为正常数,分别调节P_i和P_g方向的飞行步长,学习因子使粒子具有自我总结和向群体中优秀个体学习的能力,它们通常都取2。

迭代过程中,粒子的速度和位置都限制在特定的范围内,同时P_i和P_g不断更新,最后输出的P_g就是全局最优解。

式(1)的第一项对应多样化的特点,第二项、第三项对应于搜索过程的集中化特点,因此这三项之间的相互平衡和制约决定了算法的主要性能。

3 改进的自适应策略

从式(1)不难看出,惯性因子w表明粒子原先的速度能在多大程度上得到保留,体现了局部搜索能力和全局搜索能力的比例关系。较大的w可以增强PSO的全局搜索能力,而

到稿日期:2008-12-16 返修日期:2009-03-23 本文受国家自然基金(No. F0975026),陕西省自然科学研究计划项目(No. 2007f19)资助。
郑春颖(1979—),女,博士生,主要研究方向为智能信息处理、模式识别,E-mail:zcy_1999_79@163.com;郑全弟(1962—),男,副教授,主要研究方向为网络安全等;王晓丹(1966—),女,博士,教授,主要研究方向为智能信息处理和机器学习等;王玉冰(1980—),女,博士,主要研究方向为优化理论。

较小的 w 能加强 PSO 算法的局部搜索能力。因此，随着迭代次数的增加，惯性因子 w 应不断减小。

于是，Shi[5] 等提出 APSO，按线性规律改变惯性因子，其计算公式为：

$$w(k)=\frac{K-k}{K}(w_{\max}-w_{\min})+w_{\min} \tag{3}$$

其中，$w_{\max}$ 为最大惯性因子，$w_{\min}$ 为最小惯性因子，k 为当前迭代次数，K 为最大迭代次数。

显然，$w(k)$ 是 k 的线性递减函数，这并不符合现实中事物变化的曲线特质，因此，刻画不够精确。

神经网络中构造神经元激活函数常用的 sigmoid 函数拥有比余弦函数更平滑的顶部和底部，在线性和非线性行为之间显现出较好的平衡。因此，本文将反映种群多样性的信息熵及进化代数引入 sigmoid 函数，以此来刻画惯性因子 w 随算法进化的变化趋势。

3.1 sigmoid 函数

$$\phi(v)=\frac{1}{1+\exp(-av)} \tag{4}$$

当 $av\geqslant 9.903438$ 时，$\phi(v)$ 接近 1；当 $av<-9.903438$ 时，$\phi(v)$ 接近 0[6]。该函数的图像如图 1 所示。

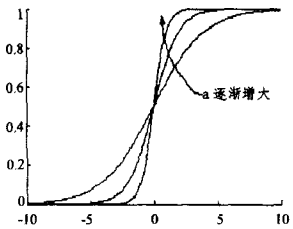

图 1　sigmoid 函数

3.2 信息熵

为了描述事物的不确定性，信息理论的鼻祖之一 C. E. Shannon[7] 引入了信息熵的概念，以此来度量离散随机事件的不确定度。

所谓信息熵，就是某种特定信息的出现概率：

$$S_n=-k\sum_i^n P_i\ln P_i\ (k\text{ 为常数因子}) \tag{5}$$

S_n 具有具有如下性质：

1）等概率时的单调递增性，即等概率时，S_n 是 n 的增函数；

2）$0\leqslant S_n\leqslant\log_2 n$，当 n 固定时，随着随机变量确定性地递减，等概率时取得最大值 $\log_2 n$。

本文旨在利用信息熵来衡量种群的多样性。那么两者之间有何联系呢？

本文定义 P_i 为当前代第 i 种（互异的）粒子在种群中所占的比例（在具体实现中设定一阈值，对粒子进行聚类，同一类中的粒子认为是同一种粒子）。由于 S_n 是 n 的递增函数。显然，在 PSO 的进行过程中，如果种群的多样性高，则 n 值大，每种粒子所占的比例较小，其概率差别不大，也就是信息熵大；如果种群的多样性低，则 n 值小，每种粒子所占的比例差距较大，其概率差别较大，那么种群的信息熵就小。由此可见，用信息熵的概念可以很好地反映种群的多样性。

从上面对信息熵与种群多样性的关系的分析可知：在早期，分散在解空间的粒子在遗传操作的作用下迅速向可能的最优区域靠拢，这样其信息熵必然会迅速减少；在进化中期，解空间的粒子继续根据自身信息和全局信息调整其位置，向当前最优粒子聚集，其信息熵也会减少，但是速度没有前期快；在后期，大部分解都集中在最优区域附近，解集基本上在进行微调，其信息熵在一定范围内进行随机摆动。

3.3 自适应调节惯性因子

在 PSO 中，所期望的惯性因子 w 的走势是：在进化初期，种群多样性好，此时选取较大的 w，以增强 PSO 的全局搜索能力，克服过早收敛于局部最优；随着进化的进行，其他粒子向最优粒子聚集，w 应相应减小，以免飞过最优位置；在进化后期，其它粒子团聚在最优粒子周围，应采用较小的 w，以加强 PSO 算法的局部搜索能力，提高搜索精度。

综上所述，本文结合 sigmoid 函数和信息熵的概念，对惯性因子 w 的设计如下：

$$w=\frac{w_{\max}-w_{\min}}{(1+\exp(-aU(k)))}+w_{\min} \tag{6}$$

其中，$U(k)=\dfrac{2\cdot S(k)}{\ln(9+k)\cdot\log_2 n}-1$，$a=9.903438$，$S(k)$ 是当前代的信息熵，即 $S(k)=-\sum_{i=1}^{n}P_i(k)\ln P_i(k)$。由信息熵的定义知：$0\leqslant S(k)\leqslant\log_2 n$，初始种群分布均匀，多样性好，因此，此时 $S(1)\to\log_2 n$，则 $U(1)\to 1$，$w\to w_{\max}$，随着进化的推进，种群多样性越来越差，即 $S(k)$ 递减，$U(k)$ 递减，到进化尾部，$S(k)\to 0$，故，$U(k)\to -1$，$w\to w_{\min}$。显然，由式(6)确定的惯性因子在算法中的走势从理论分析上符合我们的期望。

3.4 基于试探的变步长自适应粒子群算法

算法伪代码如下：

1 随机产生初始粒子群，各粒子的适应值作为 P_i 的初值，$P_g=\max_i P_i$

2 for $k=1:K$ %K——最大进化代数

2.1 对粒子按阈值 d 进行聚类，统计每个聚类的样本数，计算 $S(k)$

2.2 利用式(6)计算 w

2.3 计算 T，$T=\max\left(3,\left\lfloor\frac{12*k}{K}\right\rfloor\right)$ %T——试探次数

2.4 for $i=1$:N %N——粒子群规模

2.4.1 利用式(1)更新粒子速度

2.4.2 $f_{i\max}^{t+1}=-\infty$

2.4.3 for $t=1$:T

2.4.3.1 $\lambda=\text{rand}\left(-\frac{6}{T},\frac{6}{T}\right)$ %λ——搜索步长

2.4.3.2 更新粒子位置：$x_{im}^{t+1}=x_{im}^{t}+\lambda v_{im}^{t}$

2.4.3.3 计算粒子的适应度 $f(x_i^{t+1})$

2.4.3.4 if $f(x_i^{t+1})>f_{i\max}^{t+1}$

$f_{i\max}^{t+1}=f(x_i^{t+1})$

$x_{i\max}^{t+1}=x_i^{t+1}$

endif

endfor

2.4.4 $x_i^{t+1}=x_{i\max}^{t+1}$

2.4.5 更新 P_i

Endfor

2.5 更新 P_g

Endfor

4 仿真实验

验证了本文的改进较 APSO 算法性能有所提升，其他策略与 APSO 同。

4.1 测试函数

本文采用以下 3 个基准函数进行测试。

(1) Rosenbrock 函数

$$f_1=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2] \quad -100\leqslant x_i\leqslant 100$$

(2) Rastrigrin 函数

$$f_2=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x)+10] \quad -100\leqslant x_i\leqslant 100$$

(3) Griewank 函数

$$f_3=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \quad -100\leqslant x_i\leqslant 100$$

4.2 实验参数

惯性因子最小值为 0，最大值为 0.9，种群大小为 100、进化代数取 100，3 个基准函数的自变量维数均取 10，即 $n=10$。

4.3 实验环境

算法使用 matlab7.1 代码，运行于 AMD 1.79GHz，384MB 内存的普通 PC 机上。

4.4 实验结果

3 个函数的惯性因子变化曲线分别如图 2、图 3、图 4 所示。

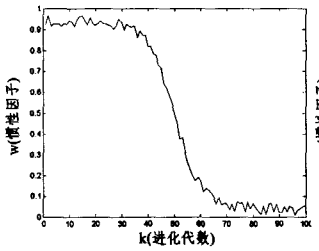

图 2 Rosenbrock 函数的惯性因子变化曲线

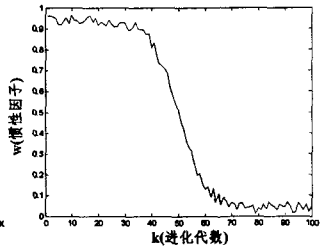

图 3 Rastrigrin 函数的惯性因子变化曲线

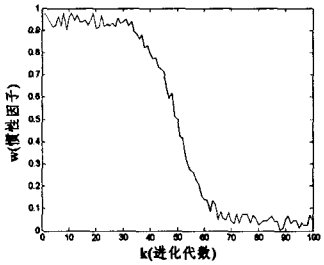

图 4 Griewank 函数的惯性因子变化曲线

表 1 分别给出了本文算法与 APSO 的 3 个函数的 20 次运行的平均最小适应值(均值±方差)。

表 1 两类算法运行 20 次的均值与方差

函数	APSO	本文算法
f_1	28.7665±31.2046	14.6624±12.3836
f_2	8.3372±6.2498	6.1126±4.1532
f_3	0.1136±0.0694	0.0561±0.0231

4.5 实验结果分析

从实验结果来看，本文算法较 APSO 优化性能有了较显著的提升。但由于每进化一代就要对粒子进行聚类，因此，本文算法耗时相对较长。其实，本文算法对聚类精度要求不高，因此，在保证一定精度的前提下，采用更加高速的聚类算法是本文算法进一步改进的一个方向。

结束语 本文在分析惯性因子在算法中的作用机理的基础上，设计了一个根据种群多样性和进化代数自适应调节的惯性因子，并运用试探法，通过变换搜索步长，提高了算法的局部搜索能力。实验证明，与 APSO 相比，本文的改进极大提高了算法的优化精度。进一步的工作将是提高聚类速度，以缩短寻优时间。

参 考 文 献

[1] Kennedy J, Eberhart R. Particle Swarm Optimization[A]//Proceedings of IEEE International Conference on Neural Networks [C]. Perth: IEEE Press, 1995

[2] 黄岚，王康平，周春光，等. 粒子群优化算法求解旅行商问题[J]. 吉林大学学报：理学版，2003，41(4)：477-480

[3] 张鹏帅，霍红卫. 多序列比对问题的粒子群优化算法求解[J]. 计算机工程与应用，2005，41(18)：84-87

[4] 黄友锐，等. 智能优化算法及其应用[M]. 北京：国防工业出版社，2008：93-100

[5] Shi Y H. Experimental Study of Partiele Swann Optimization [C]//Proceedings of SCI Conference. Orlando, FL, 2000

[6] Mo Y R, Xue X P. On stability of delayed cellular neural networks with sigmoid output functions[J]. Science in China, 2003, 46(5): 371-380

[7] Mutihac R, Cicuttin A, Mutihac R C. Entropic approach to information coding in DNA molecules[J]. Materials Science and Engineering, 2001, 12: 51-60

[8] Ivasm S I, Ikl P M. Adaptive probabilities of crossover and mutation in genetic algorithm[J]. IEEE Transactions on System, Man and Cybernetics, 1994, 24(4): 656-667

(上接第 159 页)

括哪些。当然这个框架也可用于其他的形式化建模技术。将来的工作就是在这一框架下开发支持这些框架活动的工具，并最终建立一个集成化的支持环境。

参 考 文 献

[1] van der Aalst W, van Hee K. Workflow Management Models, methods and systems[M]. The MIT Press, 2000

[2] Li Shuyou, Song Binheng. Normalized workflow net (NWF-net): Its definition and properties. doi: 10.1016/j.future.2005.02.003

[3] Milner R, Parrow J, Walker D. A Calculus of Mobile Processes Information and Computation. 1992

[4] Derek M E. Best Practice B P M. ACM QUEUE, March 2006

[5] Milner R. Communicating and Mobile Systems: The Pi Calculus [M]. Cambridge, UK: Cambridge University Press, 1999

[6] Gamma, et al. Design pattern[M]. 北京：机械工业出版社，2002

[7] Lucchi R, Mazzara M. A pi-calculus based semantics for WS-BPEL[J]. The Journal of Logic and Algebraic Programming, 2007, 70: 96-118

[8] Abouzaid F, Mullins J. A Calculus for Generation, Verification and Refinement of BPEL Specifications[J]. Electronic Notes in Theoretical Computer Science, 2008, 200: 43-65

[9] Crafa S, Mio M, Miculan M, et al. PicNIc Pi-calculus Non-Interference Checker[C]//29th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency. Xi'an, China, June 2008