软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(7):1805−1815 [doi: 10.3724/SP.J.1001.2012.04128]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种多尺度协同变异的粒子群优化算法*

陶新民[1+], 刘福荣[2], 刘 玉[1], 童智靖[1]

[1](哈尔滨工程大学 信息与通信工程学院,黑龙江 哈尔滨 150001)

[2](黑龙江省电力有限公司,黑龙江 哈尔滨 150090)

Multi-Scale Cooperative Mutation Particle Swarm Optimization Algorithm

TAO Xin-Min[1+], LIU Fu-Rong[2], LIU Yu[1], TONG Zhi-Jing[1]

[1](College of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, China)

[2](Heilongjiang Electric Power Company Limited, Harbin 150090, China)

+ Corresponding author: E-mail: taoxinmin@hrbeu.edu.cn, http://www.hrbeu.edu.cn

Tao XM, Liu FR, Liu Y, Tong ZJ. Multi-Scale cooperative mutation particle swarm optimization algorithm. *Journal of Software*, 2012,23(7):1805−1815 (in Chinese). http://www.jos.org.cn/1000-9825/4128.htm

Abstract: To deal with the problem of premature convergence and low precision of the traditional particle swarm optimization algorithm, a particle swarm optimization (PSO) algorithm based on multi-scale cooperative mutation, is proposed, which is guaranteed to converge to the global optimal solution with probability one. The special multi-scale Gaussian mutation operators are introduced to make the particles explore the search space more efficiently. The large-scale mutation operators can be utilized to quickly locate the global optimal space during early evolution. The small-scale mutation operators, which are gradually reduced according to the change of the fitness value can implement the accuracy of the solution at the late evolution. The proposed method is applied to six typical complex function optimization problems, and the comparison of the performance of the proposed method with other PSO algorithms is experimented. The results show that the proposed method can effectively speed up the convergence and improve the stability.

Key words: particle swarm optimization; premature convergence; multi-scale; cooperative mutation; ftness

摘 要: 为了改善粒子群算法易早熟收敛、精度低等缺点,提出一种多尺度协同变异的粒子群优化算法,并证明了该算法以概率 1 收敛到全局最优解.算法采用多尺度高斯变异机制实现局部解逃逸.在算法初期阶段,利用大尺度变异及均匀变异算子实现全局最优解空间的快速定位;随着适应值的提升,变异尺度随之降低;最终在算法后期阶段,利用小尺度变异算子完成局部精确解空间的搜索.将算法应用 6 个典型复杂函数优化问题,并同其他带变异操作的 PSO 算法比较,结果表明,该算法在收敛速度及稳定性上有显著提高.

关键词: 粒子群算法;早熟收敛;多尺度;协同变异;适应度

中图法分类号: TP18 **文献标识码**: A

* 基金项目: 国家自然科学基金(61074076); 国家博士后科学基金(20090450118); 中国博士点新教师基金(20092304120017); 黑龙江省博士后科学基金(LBH-Z08227)

收稿时间: 2009-11-17; 修改时间: 2011-04-25, 2011-06-24; 定稿时间: 2011-10-08

粒子群算法(particle swarm optimization,简称 PSO)是由 Eberhart 博士提出的一种基于群体智能的优化算法,其基本思想源于对鸟群捕食行为的研究.由于其简单、易实现、收敛快等优点,在目标函数优化、神经网络训练等方面都有很好的表现,其应用涉及系统控制、模式识别、通信工程等各个领域.然而,早熟和收敛慢[1−5]一直是困扰粒子群算法的两个难题.

在避免粒子群算法早熟收敛的研究中,很多改进方案被提了出来,包括变更公式法、分群方法等.对 PSO 的参数进行自适应选择策略,在改善算法的性能方面也取得了良好的效果,如在惯性权值中引入随机成分[6,7]、利用模糊逻辑调节惯性权值[8,9]、采用一个独立的二级 PSO 优化一级 PSO 参数[10]、利用 Q-学习调节参数[11]、采用自适应评论策略调节参数[12]等.此外,Zhang 等人[13]通过评价每个粒子的性能改善指标自适应调节惯性权值、粒子数以及粒子邻域大小;Ratnaweera 等人[14]通过引入时变加速因子和时变惯性权因子,有效地增强了算法的局部搜索能力;Zhan 等人提出的自适应粒子群优化算法(adaptive particle swarm optimization,简称 APSO)[15]采用数学统计的技术对算法运行过程中的种群位置分布信息和适应值信息进行分析,并以此定义了一个称为“进化因子(evolutionary factor)”的变量 f.APSO 采用模糊逻辑的方法对 f 的取值进行模糊划分,从而估计算法运行的不同进化状态,并根据不同的进化状态设计有效的参数自适应控制策略以加快算法的求解速度,取得了良好效果.

近年来,许多学者又将进化算法中的变异操作引入 PSO.如文献[16]通过对粒子引入一个小概率随机变异操作来增加种群多样性(dissipative PSO,简称 DPSO),这样虽然可以对解空间进行更好的搜索,但如果变异率控制不当极易导致原始种群的混乱,达不到局部精确搜索的目的.在此基础上,文献[17]给出了一种变异操作随时间变化的自适应层次 PSO 算法(self-organizing hierarchical PSO,简称 HPSO),确定了自适应参数的选择方式.但由于该算法没有考虑速度公式中惯性因素的影响,使得算法仍然未能迅速而有效地逃离局部极小值.文献[18]为了提高种群多样性和搜索速度之间的平衡,对阈值设定进行了改进,并结合群体自动分家机制,提出一种自适应逃逸的微粒群算法(self-adaptive escape PSO,简称 AEPSO).然而,上述算法在变异时均采用单一的均匀变异机制,逃逸能力大为减弱,使改进 PSO 算法在有限的迭代次数内无法实现全局最优解的探索.因此,如何增强算法的逃逸能力,使其在快速定位到最优解区域的同时提高最优解的精度,值得我们进一步加以研究.

鉴于此,本文提出一种多尺度协同变异的自适应粒子群算法(multi-scale cooperative mutatingly self-adaptive escape PSO,简称 MAEPSO).该算法利用不同大小方差的自适应高斯变异机制实现解空间的探索,这种多个或大或小的变异机制能够促使整个种群以尽量分散的变异尺度来对解空间进行更加详尽的探索.高斯变异的范围随着适应值的变小逐渐降低,在算法后期有利于提高最优解的精度.同时,利用均匀算子维护种群多样性.通过对不同评测函数进行测试,均验证了新算法优良的优化性能.

1 标准粒子群算法

一个由 m 个粒子组成的群体在 n 维搜索空间中以一定速度 $v_i=(v_{i1},v_{i2},\ldots,v_{in})^T$ 飞行,粒子的好坏由一个事先设定的适应值函数来确定,每个粒子在搜索时根据目前为止自身搜索到的历史最好点 $p_i=(p_{i1},p_{i2},\ldots,p_{in})^T$(也称为 $p$best)和整个群体中所有粒子发现的最好点 $p_g=(p_{g1},p_{g2},\ldots,p_{gn})^T$(也称为 $g$best)进行位置 $x_i=(x_{i1},x_{i2},\ldots,x_{in})^T$ 的变化.粒子迭代过程中速度与位置的更新公式为

$$v_{id}(t+1)=w\times v_{id}(t)+c_1\times r_1\times(p_{id}(t)-x_{id}(t))+c_2\times r_2\times(p_{gd}(t)-x_{id}(t)) \tag{1}$$

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

其中,r_1 和 r_2 是均匀分布在[0,1]范围内的随机数,用来保持群体的多样性;c_1 和 c_2 为学习因子,使粒子具有向群体中优秀个体学习的能力;w 为惯性权重,起着权衡局部和全局最优能力的作用.粒子的每维速度被限制在一个最大速度为 $V_{\max}$ 的范围内.

2 多尺度协同变异粒子群算法

粒子群算法迭代过程中,如果某个粒子发现了一个当前最优位置,其他粒子会迅速向其靠拢.如果该最优位

置是局部最优点,粒子群就无法在解空间内重新搜索,算法就会陷入局部最优,从而出现早熟收敛的现象.为了避免这种情况的发生,此时,如果我们对粒子的速度进行变异,就可以改变粒子的前进方向,使粒子进入解空间的其他区域搜索,从而有可能发现新的个体最优位置和种群最优位置,增加算法找到全局最优解的几率.

然而,在上述提到的通过增加变异操作来帮助算法逃出局部最优的改进算法中,粒子的逃逸能力都取决于均匀变异尺度,均匀变异虽然具有很强的逃离原点的能力,但是由于我们事先无法预知函数局部极值间的距离,因此无法给出合适的变异尺度.如果初始尺度较大,则无法保证所逃离到的新位置的适应值一定优于现有的最优解,尤其是在算法进化后期最优解可能就存在于现有最优解周边的区域,这样,通过单纯的均匀变异无法确定更优的位置,最终因迭代次数的限制使其无法收敛到全局最优解.为此,本文提出一种多尺度协同变异粒子群优化算法,算法的逃逸能力取决于不同尺度方差的高斯变异算子,不同尺度的变异有助于算法在搜索空间中进行分散式搜索.同时,变异尺度随着适应度的提升逐渐减少.这样,在保证算法逃逸能力的同时,提高了最优解的精度.

2.1 多尺度高斯变异算子

设尺度个数为 M,首先初始化多尺度高斯变异算子的方差:

$$\boldsymbol{\sigma}^{(0)}=(\sigma_1^{(0)},\sigma_2^{(0)},...,\sigma_M^{(0)}) \tag{3}$$

初始时,方差一般设定为优化变量的取值范围;随着迭代次数的增加,多尺度高斯变异算子的方差会随之进行调整,具体调整方式如下:首先,根据适应值函数 f 的大小对种群中的粒子由小到大排序;然后对其进行组合,生成 M 个子群,每个子群的粒子个数为 $P=N/M$.K 是当前迭代次数,计算每一个子群的适应度值:

$$FitX_m^{(K)}=\sum_{i=1}^{P}f(x_i^m)\Big/P,m=1,2,...,M \tag{4}$$

不同变异尺度之间相互竞争,根据适应能力的不同设置不同的变异能力.因此,第 m 个变异算子的标准差为

$$\sigma_m^{(K-1)}\exp\left(\frac{M\cdot FitX_m^{(K)}-\sum_{m=1}^{M}FitX_m^{(K)}}{FitX_{\max}-FitX_{\min}}\right) \tag{5}$$

$$FitX_{\max}=\max(FitX_1^{(K)},FitX_2^{(K)},...,FitX_M^{(K)}) \tag{6}$$

$$FitX_{\min}=\min(FitX_1^{(K)},FitX_2^{(K)},...,FitX_M^{(K)}) \tag{7}$$

根据每个子群适应度得到的 M 个不同尺度算子随迭代次数的变化如图 1 所示,由于变异算子的进化是一个递归过程,排在后面的变异算子可能很大,因此,对变异算子的标准差作如下规范:如果 $\sigma_i^{(k)}>W/4$, 则

$$\sigma_i^{(k)}=|W/4-\sigma_i^{(k)}| \tag{8}$$

其中,W 为待优化变量空间的宽度.重复上式,直到满足 $\sigma_i^{(k)}<W/4$.

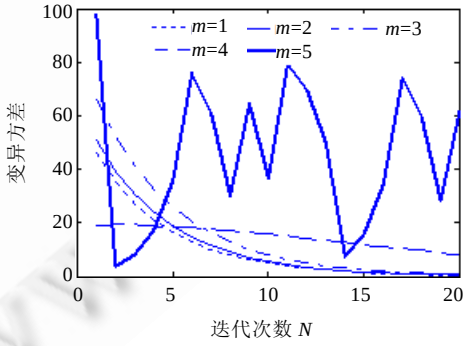

Fig.1 Changing of different scale-variations with iterations when M=5

图 1 M=5 时不同尺度方差随迭代次数的变化示意图

从图 1 可以看出,随着迭代次数的变化,不同尺度算子的方差变化情况不同.本文提出的多尺度变异算子能够实现整个解空间的覆盖性搜索,其中,大尺度(m=5)变异算子具有振荡性质,完成对解空间的粗搜索,使粒子快速定位到最优解周围;而小尺度(m=1,2,3,4)变异算子呈递减趋势,在进化后期完成局部精确解的探索.为了最大范围地实现空间勘探能力,进行完多尺度高斯变异后,按公式(9)进行一次均匀变异操作,比较变异后所有粒子的适应值,取最好的位置作为新的逃逸点.阈值 T_d 用来保存种群中第 d 维的当前速度阈值且其值恒大于 0;$rand$ 为一个均匀分布在[0,1]范围内的随机数.

$$\text{If } (v_{id}<T_d) \text{ then } v_{id}=rand\times V_{\max} \tag{9}$$

2.2 勘探和开采能力的自适应协同

我们知道,任何一种进化算法都存在勘探和开采两类不同的操作,PSO 算法随着迭代次数的增加粒子会收敛于某个最优个体,多样性逐渐丧失.为了防止粒子陷入局部最优,上述算法通过变异操作改善种群多样性来提高算法的全局搜索能力.然而在算法后期,变异操作往往会使微粒群不能进行精确的局部搜索.因此,如何协调勘探和开采能力是进化算法性能提高的关键.为此,本文算法采用两种方式实现勘探和开采能力间的均衡.

首先,本文算法采用 HPSO 中消除了速度惯性部分的公式进行更新,这样就使得 PSO 进化只负责整个算法的开采能力,当速度小于一定阈值时,给予速度一个变异操作——逃逸运动,描述如下:

If $(v_{id}<T_d)$ then

$$f(x_i+randn\times\sigma_i^{(K)})=\min_{0\leqslant j\leqslant M} f(x_i+randn\times\sigma_j^{(K)})$$

If $f(x_i+randn\times\sigma_i^{(K)})<f(x_i+rand\times V_{\max})$

$$v_{id}=randn\times\sigma_i^{(K)}$$

Else

$$v_{id}=rand\times V_{\max} \tag{10}$$

PSO 算法进化公式消除了惯性部分,使粒子速度快速下降.这样,在有限迭代次数内能进行多次逃逸,提高了种群多样性,增强了算法全局搜索性能.另外,本文算法的逃逸运动采用多尺度高斯变异方式,小尺度能够保证精确解局部搜索能力,避免了 HPSO 算法进化后期因下降过快和均匀变异导致无法进行详细局部搜索的不足.

另外,公式(10)中阈值的大小将会对算法的执行效果产生一定影响:如果阈值过大,则会导致逃逸次数增加,从而打乱 PSO 种群进化的原有结构,使算法无法进行局部深度搜索,开采能力下降;阈值过小,则会导致微粒速度需较长时间达到逃逸条件,在有限的迭代次数内无法进行有效的全局搜索,勘探能力下降.为此,结合文献[18],本文给出一种自适应的阈值设定方法.设微粒各维速度间相互独立,并对每维速度均给予一个不同的阈值,当该维度中有多个微粒速度达到这个值时,阈值就自动下降,通过这种方式达到动态调整微粒速度的目的.具体描述如下:

$$G_d(t)=G_d(t-1)+\sum_{i=1}^{Size}c_{id}(t) \tag{11}$$

其中, $c_{id}(t)=\begin{cases}0, & v_{id}(t)>T_d\\ 1, & v_{id}(t)<T_d\end{cases}$.

If $G_d(t)>k_1$ then

$$G_d(t)=0;T_d=T_d/k_2 \tag{12}$$

其中,k_1,k_2 为常数,$Size$ 为种群大小,频率 $G_d(t)$表示种群第 d 维速度曾经发生逃逸的总次数.k_1 标记调整阈值 T_d 和频率 $G_d(t)$的条件值,k_2 用来控制阈值速度下降的幅度.通过控制粒子飞行速度的方式,不仅能够保证算法的全局解空间勘探性能,同时在算法后期还能使粒子群进行精确的局部搜索,保证算法的收敛速度.

2.3 算法流程

多尺度协同变异的粒子群优化算法的流程如下:

(1) 种群的初始化:随机初始化微粒的速度、位置、全局最优解,计算微粒的适应度,同时作为微粒的个体

最优位置.反复进行 N 次,共生成 N 个初始微粒群;

(2) 对每个微粒,比较其适应度和它经历过的最好位置适应度,如果更好,则更新该微粒最好位置;

(3) 对每个微粒,比较其适应度和群体所经历的最好位置的适应度,如果更好,则更新全局最好位置;

(4) 根据公式(1)PSO 进化公式调整微粒的速度;

(5) 利用公式(11)判断微粒是否需要逃逸;

(6) 若满足逃逸条件,则利用公式(10)进行逃逸;

(7) 更新微粒的位置;

(8) 根据公式(4)~公式(7)计算多尺度变异算子;

(9) 根据公式(11)、公式(12)计算每一维速度的阈值;

(10) 判断终止条件,若满足则终止,否则转到步骤(2).

3 多尺度协同变异粒子群算法分析

3.1 算法的优化机理分析

为了说明方便,设优化函数 $f(x)$只有一个自变量函数,如图 2 所示,多尺度变异算法中种群的某一个粒子经过多次迭代后到达 a 点,经过小尺度逃逸后,向下“下山”找到个体 a',经过 PSO 算法进化到 a'',然后经过大尺度变异后找到适应值更高的粒子 b,则粒子 b 被选择进入下一世代,经过 PSO 若干代进化找到粒子 b'.粒子 b'经过大尺度逃逸操作找到了粒子 c,则粒子 c 被选择进入下一个世代,经过 PSO 进化和小尺度逃逸最终找到最优解 c'.由上可知,多尺度逃逸操作将整个解空间看作是一个山谷,经过大尺度变异逃逸操作进行“下山”以寻求适应值更高的区域($a''\rightarrow b,b'\rightarrow c$).大尺度逃逸操作用于搜索全局解空间,是粗搜索.如果大尺度逃逸操作找到了适应值更高的区域,则 PSO 进化操作和小尺度变异操作在该区域进行局部搜索,以寻求更高精度的解($a\rightarrow a',a'\rightarrow a'',b\rightarrow b',c\rightarrow c'$).因此,本文算法是一种勘探和开采能力相互交替进行的搜索算法,具有全局和局部两层邻域搜索机制,从而保证了算法全局和精确的局部寻优性能.

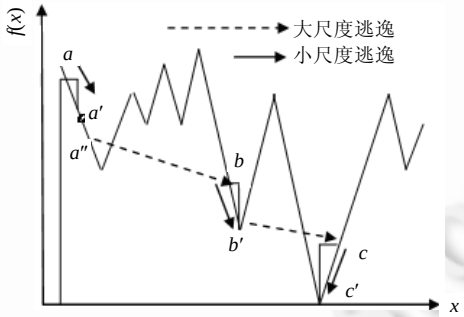

Fig.2 Mechanism of cooperative multi-scale escape

图 2 多尺度协同逃逸优化机理

3.2 算法的收敛性分析

引理 1. 设$\Delta x_{id}=v_{id},r_1,r_2\sim N(0,1)$,则$\Delta x_{id}\sim N(\mu_d,\sigma_d)$.

证明:由公式(2)可得:

$$\Delta x_{id}=c_1\times r_1\times(p_{id}-x_{id})+c_2\times r_2\times(p_{gd}-x_{id}) \tag{13}$$

设

$$\phi_1=c_1\times(p_{id}-x_{id}),\phi_2=c_2\times(p_{gd}-x_{id}) \tag{14}$$

则

$$\Delta x_{id} = \phi_1 \times r_1 + \phi_2 \times r_2 \tag{15}$$

由于 $r_1, r_2 \sim N(0,1)$,显然 $\Delta x_{id} \sim N(\mu_d, \sigma_d)$.对于 MAEPSO 算法来说,在粒子运行过程中,公式(10)定期给粒子一个较大的高斯变异速度冲量,使得 Δx_{id} 在搜索过程中始终是非零高斯变量.

定义 1. 定义优化问题(p)的全局最优粒子集合为

$$X^* \cong \{x^* : \neg\exists x \neq x^*, f(x) > f(x^*)\} \tag{16}$$

对于微粒种群 x,令 $\vartheta(X) \cong |X \cap X^*|$ 表示微粒种群 x 中包含最优粒子的个数.

定义 2. 如果对于任意的初始状态 X_0 均有

$$\lim_{K\to\infty} P\{\vartheta(X(K)) \geqslant 1 \mid X(0) = X_0\} = 1 \tag{17}$$

则称算法以概率 1 收敛于全局最优解.

定理 1. 多尺度协同变异的微粒群算法以概率 1 收敛.

由引理 1 可知,$\Delta x_{id} \sim N(\mu_d, \sigma_d)$,由公式(10)可知,MAEPSO 算法是一种带有变异操作的进化策略算法.在 MAEPSO 算法中,微粒种群序列 $\{X(K), K \geqslant 0\}$ 是一个有限齐次可约马尔可夫链,对于任意初始状态 X_0,算法以概率 1 收敛于全局最优粒子集合 X^*.证明如下:

证明:记 $P_0(K)=P\{\vartheta(X(K))=0\}$,由贝叶斯条件概率公式有

$$\begin{aligned} P_0(K+1) &= P\{\vartheta(X(K+1)) = 0\} \\ &= P\{\vartheta(X(K+1)) = 0 \mid \vartheta(X(K)) \neq 0\} \times P\{\vartheta(X(K)) \neq 0\} + \\ &\quad P\{\vartheta(X(K+1)) = 0 \mid \vartheta(X(K)) = 0\} \times P\{\vartheta(X(K)) = 0\} \end{aligned} \tag{18}$$

由微粒群算法的保优性质可得 $P\{\vartheta(X(K+1))=0 \mid \vartheta(X(K))\neq 0\}=0$,所以,

$$P_0(K+1) = P\{\vartheta(X(K+1)) = 0 \mid \vartheta(X(K)) = 0\} \times P_0(K) \tag{19}$$

又由公式(10)可知,

$$P\{\vartheta(X(K+1))>0 \mid \vartheta(X(K))=0\} > 0 \tag{20}$$

记

$$\xi = \min_K P\{\vartheta(X(K+1)) > 0 \mid \vartheta(B(K)) = 0\}, K = 0,1,2,... \tag{21}$$

则

$$P\{\vartheta(X(K+1)) > 0 \mid \vartheta(X(K)) = 0\} \geqslant \xi > 0 \tag{22}$$

所以,

$$\begin{aligned} P\{\vartheta(X(K+1)) = 0 \mid \vartheta(X(K)) = 0\} &= 1 - P\{\vartheta(X(K+1)) \neq 0 \mid \vartheta(X(K)) = 0\} \\ &\leqslant 1 - P\{\vartheta(X(K+1)) > 1 \mid \vartheta(X(K)) = 0\} \\ &\leqslant 1 - \xi < 1 \end{aligned} \tag{23}$$

因此,

$$0 \leqslant P_0(K+1) \leqslant (1-\xi) \times P_0(K) \leqslant (1-\xi)^2 \times P_0(K-1) \leqslant ... \leqslant (1-\xi)^{K+1} \times P_0(0) \tag{24}$$

因为 $\lim_{K\to\infty}(1-\xi)^{K+1} = 0, 1 \geqslant P_0(0) \geqslant 0$, 所以,

$$0 \leqslant \lim_{K\to\infty} p_0(K) \leqslant \lim_{K\to\infty}(1-\xi)^{K+1} P_0(0) = 0 \tag{25}$$

故 $\lim_{K\to\infty} p_0(0) = 0$, 因此,

$$\lim_{K\to\infty} P\{\vartheta(X(K)) \geqslant 1 \mid X(0) = X_0\} = 1 - \lim_{K\to\infty} P\{\vartheta(X(K)) = 0 \mid X(0) = X_0\} \geqslant 1 - \lim_{K\to\infty} P_0(K) = 1 \tag{26}$$

所以 $\lim_{K\to\infty} P\{\vartheta(X(K)) > 1 \mid X(0) = X_0\} = 1$. 定理 1 得证. □

因此,MAEPSO 算法以概率 1 收敛于全局最优粒子集合 X^*.

3.3 算法计算时间复杂度分析

对于传统 PSO 算法,一个粒子每维分量进行更新所需要的运算如下:5 次乘法运算,5 次加法运算.假设乘法和加法运算所需要的时间分别为 T_m 和 T_n.实验中,设置传统 PSO 算法中的粒子数量为 m,维数为 d,所需迭代次

数为 n_1,则传统的 PSO 算法完成优化所需的平均时间为 $d\times m\times n_1\times(5T_m+5T_n)$.对于本文算法,如果尺度个数 M=5,所需迭代次数为 n_2,则一个粒子进行更新、变异所需要的运算如下:11 次乘法运算,10 次加法运算.因此,算法完成优化所需的平均时间为 $d\times m\times n_2\times(11T_m+10T_n)$.由此得出,当 $n_1\approx 2n_2$ 时,本文算法的平均时间与传统 PSO 算法相当.由于本文算法进化初期大尺度变异算子快速定位最优解附近区域的勘探能力和进化后期小尺度变异算子的深度开采能力,使算法搜索到问题最优解所需的迭代次数远远小于传统 PSO 算法及其改进算法,$n_2<<n_1/2$,图 3~图 8 中,本文算法与其他算法收敛速度的对比结果可以证明这一点.因此,从某种意义上说,本文算法找到问题最优解所需的平均时间与传统 PSO 算法不相上下,因此并没有提高传统 PSO 算法的时间复杂度.

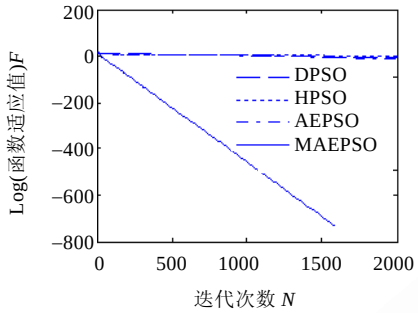

Fig.3 Convergence performance comparison of 30-dimension Tablet function

图 3 30 维 Tablet 函数收敛情况对比图

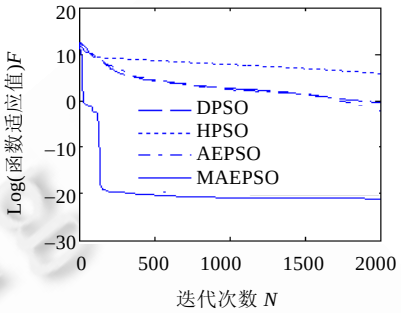

Fig.4 Convergence performance comparison of 30-dimension Quadric function

图 4 30 维 Quadric 函数收敛情况对比图

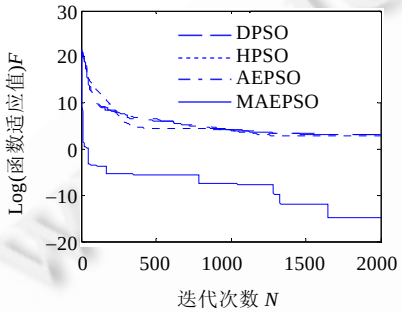

Fig.5 Convergence performance comparison of 30-dimension Rosenbrock function

图 5 30 维 Rosenbrock 函数收敛情况对比图

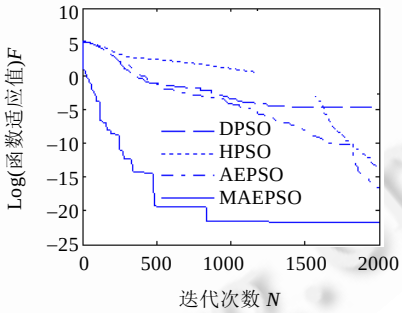

Fig.6 Convergence performance comparison of 30-dimension Griewank function

图 6 30 维 Griewank 函数收敛情况对比图

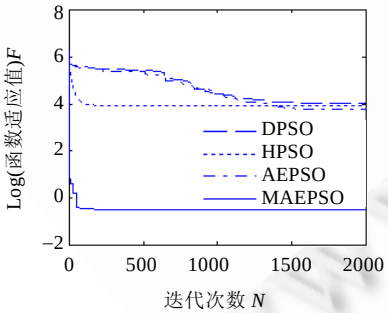

Fig.7 Convergence performance comparison of 30-dimension Rastrigrin function

图 7 30 维 Rastrigrin 函数收敛情况对比图

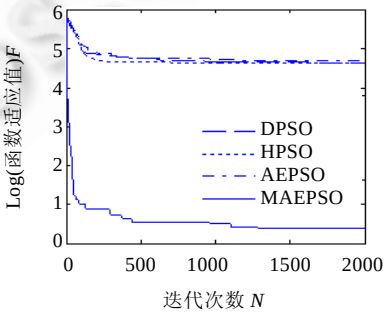

Fig.8 Convergence performance comparison of 30-dimension Schaffer function

图 8 30 维 Schaffer 函数收敛情况对比图

4 对比实验及结果分析

为了验证本文算法的优化性能,选择文献[18]中的 6 个 Benchmark 函数:Tablet,Quadric,Rosenbrock, Griewank,Rastrigin,Schaffer F7 进行数值实验.其中,前 3 个函数为单模态函数,后 3 个函数为多模态函数.6 个函数当自变量取合适值时,极值都为 0.将本文算法与传统 PSO 算法及其改进的 DPSO,HPSO,AEPSO 算法进行对比实验.5 种算法参数设置如下:采用线性下降惯性权重,w 在[0.95,0.4]之间随迭代数线性递减;c_1,c_2 均为 1.4,DPSO 中 C_v=0.001,AEPSO 和本文算法的 K_1=5,K_2=10,本文算法 M=5,初始方差 σ_0 为优化变量的范围,T_d=0.5,种群规模均为 20,函数维度为 30,每次运行 6 000 代,每个函数独立运行 50 次.

4.1 单模态Benchmark函数对比实验

图 3~图 5 反映了 5 种算法处理单模态 Benchmark 函数的对比结果.如图 3、图 4 所示,对于简单的 Tablet 函数和 Quadric 函数,本文算法能够迅速定位到最优解 0 的附近区域.对于复杂的单模态 Rosenbrock 函数,由于函数山谷提供的信息量很少,使传统的 PSO 算法很难在短时间内辨别搜索方向,易陷入局部最优.从图 5 可以看出,本文算法能够在短时间内找到正确的搜索方向,下降速度较快,这是由于在进化初期采用大尺度变异算子对解空间进行大范围搜索的结果;而其他算法单一的均匀变异机制具有一定的盲目性,因此下降得比较缓慢,减缓了算法的收敛速度.

为了进一步证明本文算法的优越性,分别统计 5 种算法在处理 6 个 Benchmark 函数问题时运行 50 次所得的最优值、最差值、中值、平均值和标准差,单模态问题的统计结果见表 1.可以看出,本文算法在各项数值上均优于其他算法,特别是对于 Tablet 函数,50 次寻优实验能够 100%地找到最优解.Tablet 函数和 Quadric 函数的统计数据显示,某些情况下,DPSO 和 HPSO 的性能比传统的 PSO 算法更差.这是因为仅通过单一的均匀变异和消除速度惯性部分的做法,使算法丧失了全局解空间的勘探能力.由于粒子群算法属于随机搜索算法,因此稳定性是衡量算法好坏的重要标准.表 1 显示,PSO 及其他改进算法在 Quadric 和 Tablet 函数优化问题上具有较好的稳定性,但对于复杂的 Rosenbrock 函数,由于很难确定最优解搜索方向,因此各种算法单次结果差异很大.而本文算法单次得到的解之间的差异性相对较小,50 次实验中寻优率达到 94%,因此算法具有更好的稳定性和健壮性.

Table 1 Performance comparison of MAEPSO and other PSO algorithms in uni-model function

表 1 MAEPSO 和其他 PSO 算法在单模态 Benchmark 问题上的性能对比

函数	算法	最小值	中间值	平均值	方差	最大值
Tablet	PSO	6.2e–22	1.1e–20	1.1e–19	8.2e–19	5.6e–18
	DPSO	1.4e–11	1.8e–10	1.6e–10	1.3e–11	2.8e–10
	HPSO	2.1e–26	2.5e–24	3.9e–24	4.7e–24	1.6e–23
	AEPSO	1.2e–128	5.6e–124	2.6e–122	8.1e–122	6.9e–121
	MAEPSO	**0 (100%)**	**0**	**0**	**0**	**0**
Quadric	PSO	0.722	70.65	1.2e+2	1.1e+2	4.8e+02
	DPSO	7.8e–04	3.5e–03	3.8e–03	2.0e–3	1.1e–02
	HPSO	0.255	1.198	1.412	0.811	3.837
	AEPSO	4.4e–11	6.5e–10	1.2e–09	1.4e–09	7.5e–09
	MAEPSO	**1.6755e–215**	**4.1340e–081**	**3.1277e–010**	**1.1703e–009**	**4.3788e–009**
Rosenbrock	PSO	4.873	77.24	93.71	1.5e+02	1.0e+03
	DPSO	8.2e–02	20.64	32.08	32.17	1.7e+02
	HPSO	5.3e–02	72.57	1.9e+02	1.1e+02	6.9e+02
	AEPSO	2.3e–03	8.322	12.28	17.19	79.69
	MAEPSO	**0 (94%)**	**0**	**2.9273e–007**	**1.6430e–006**	**1.1155e–005**

4.2 多模态Benchmark函数对比实验

从图 6~图 8 可以看出,对于 3 个多模态函数,本文算法能够比其他算法在较少迭代次数内定位到全局最优解附近的区域,全局搜索能力明显优于其他算法.从表 2 可知,MAEPSO 算法的中值、平均值和标准差等各项数据均显著优于其他算法,显示了新算法的稳定性和健壮性.在处理多模态 Griewank 函数和 Rastrigrin 函数优化问

题时,在有限次代数内几乎全部找到了问题的最优解,寻优率分别达到 95.5%和 92%.由于 Schaffer 函数的特殊性质,其他算法通过增加均匀变异操作未能找到问题的最优解 0,而本文算法在 50 次的实验中有 22%的机会找到了全局最优解.虽然本文算法对于多模态问题没有在每次运行都能找到全局最优解,但是只要给予足够长的迭代次数,MAEPSO 算法总能逃出局部最优找到全局最优解.从表 2 中最优解和最差解实验结果之间的差异可以看出,无论是 PSO 算法还是其他改进算法,对于处理多模态函数优化问题的稳定性都很差,而本文算法相对于其他算法而言稳定性大为提高,这些都是由于本文算法采用了多尺度变异算子使得算法能够在搜索空间内进行分散式搜索的结果.

Table 2 Performance comparison of MAEPSO and other PSO algorithms in multi-model function

表 2 MAEPSO 和其他 PSO 算法在多模态 Benchmark 问题上的性能对比

函数	算法	最小值	中间值	平均值	方差	最大值
Griewank	PSO	6.7e−11	1.96e−02	2.6e−02	2.8e−02	0.108
	DPSO	4.4e−12	1.7e−02	2.5e−02	3.3e−02	0.147
	HPSO	1.1e−16	9.9e−03	1.5e−02	2.1e−02	8.5e−02
	AEPSO	0	8.6e−03	1.2e-02	1.6e−02	6.6e−02
	MAEPSO	**0 (95.5%)**	**0**	**1.0102e−010**	**6.2184e−010**	**4.3974e−009**
Rastrigrin	PSO	30.87	56.71	55.18	1.2.31	81.59
	DPSO	7.2e−09	7.0e−06	0.201	0.498	2.058
	HPSO	12.93	26.86	29.02	8.963	55.81
	AEPSO	0	4.0e−12	0.577	1.087	3.980
	MAEPSO	**0(92%)**	**0**	**2.0467e−008**	**7.0900e−008**	**2.4560e−007**
Schaffer	PSO	2.4e+02	2.9e+02	2.9e+02	30.91	3.5e+02
	DPSO	82.35	1.5e+02	1.3e+02	31.17	2.3e+02
	HPSO	1.9e+02	2.4e+02	1.9e+02	22.42	2.1e+02
	AEPSO	44.57	1.2e+02	1.1e+02	39.19	2.3e+02
	MAEPSO	**0 (22%)**	**0.2974**	**1.4397**	**2.2676**	**9.0606**

4.3 不同初始参数T_d对算法性能的影响

为了验证初始阈值参数对本文算法性能的影响,利用不同取值范围的 Schaffer,Griewank,Rosenbrock 及 Rastrigrin 函数进行实验,实验中取不同的初始值,实验结果如图 9 所示.

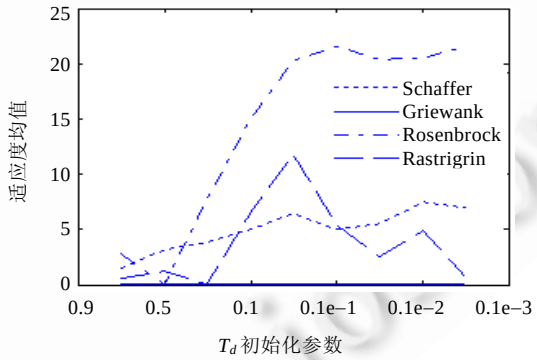

Fig.9 Effect of initial parameters T_d on performance

图 9 不同 T_d 初始化参数对算法性能的影响

我们可以看出,阈值取[0.1,0.9]时获得的最优解最为理想.这是由于,在算法初始阶段充分利用了多尺度高斯变异算子的大尺度实现快速的全局最优解区域定位,提高了算法全局搜索性能.在进化的中后期,由于本文算法的阈值自适应调节,使得阈值随着迭代次数的增加而下降,最终利用小尺度变异实现局部最优解空间的搜索.因此,在优化变量取值较宽且搜索空间复杂时,可以选取较大的初始阈值,这样可以在算法初期阶段有效地提高算法的全局最优解的搜索能力.

5 结束语

本文提出一种新的多尺度变异粒子群算法,利用不同大小方差的高斯变异机制实现解空间的搜索,这种多个或大或小的变异机制能够促使整个种群以尽量分散的变异尺度来对解空间进行更加详尽的探索.同时,高斯变异的范围随着适应值的变小也逐渐缩小,有利于提高最优解的精度.同时,利用均匀变异算子维护种群多样性.通过 6 个 Benchmark 函数进行测试,结果表明,新算法能够在算法初期快速定位到最优解附近区域,进而使得微粒通过进化 PSO 及小尺度变异算子向着最优精确解空间逼近,使其在进化过程中保持局部最优解空间开采的能力,加快了算法的收敛速度.其中,不同尺度个数对本文算法性能的影响将是本课题下一阶段研究的重点.

References:

[1] Kennedy J, Eberhart R. Particle swarm optimization. In: Proc. of the IEEE Int'l Conf. on Neural Networks. Piscataway: IEEE Press, 1995. 1942−1948.

[2] Shi Y, Eberhart R. A modified particle swarm optimizer. In: Lee J K, Andrew B, Schmid B, eds. Proc. of the IEEE Int'l Conf. on Evolutionary Computation. Piscataway: IEEE Press, 1998. 67−73. [doi: 10.1109/ICEC.1998.699146]

[3] Parsopoulos KE, Vrahatis MN. On the computation of all global minimizers through particle swarm optimization. IEEE Trans. on Evolutionary Computation, 2004,8(3):211−224. [doi: 10.1109/TEVC.2004.826076]

[4] Trelea IC. The particle swarm optimization algorithm: Convergence analysis and parameter selection. Information Processing Letters, 2003,85(9):317−325. [doi: 10.1016/S0020-0190(02)00447-7]

[5] Mendes R, Kennedy J, Neves J. The fully informed particle swarm: Simpler, maybe better. IEEE Trans. on Evolutionary Computation, 2004,8(3):204−210. [doi: 10.1109/TEVC.2004.826074]

[6] Mohagheghi S, Valle YD, Venayagamoorthy GK, Harley RG. A comparison of PSO and backpropagation for training RBF neural networks for identification of a power system with STATCOM. In: Sharkawi M, Lerman K, Galstyan A, eds. Proc. of the IEEE Swarm Intell. Symp. Piscataway: IEEE Press, 2005. 381−384. [doi: 10.1109/SIS.2005.1501646]

[7] Eberhart RC, Shi Y. Particle swarm optimization: Developments, applications and resources. In: Pham T, ed. Proc. of the 2001 Congress on Evolutionary Computation. Piscataway: IEEE Press, 2001. 81−86. [doi: 10.1109/CEC.2001.934374]

[8] Shi Y, Eberhart RC. Fuzzy adaptive particle swarm optimization. In: Pham T, ed. Proc. of the 2001 Congress on Evolutionary Computation. 2001. 101−106. [doi: 10.1109/CEC.2001.934377]

[9] Doctor S, Venayagamoorthy GK, Gudise VG. Optimal pso for collective robotic search applications. In: Isasi P, Hernandez JC, Clark JA, eds. Proc. of the CEC 2004 Congress on Evolutionary Computation. Piscataway: IEEE Press, 2004. 1390−1395. [doi: 10.1109/CEC.2004.1331059]

[10] Khajenejad M, Afshinmanesh F, Marandi A, Araabi BN. Intelligent particle swarm optimization using Q-learning. In: Shi YH, Arabshahi P, eds. Proc. of the IEEE Swarm Intell. Symp. Piscataway: IEEE Press, 2006. 7−12.

[11] Doctor S, Venayagamoorthy G. Improving the performance of particle swarm optimization using adaptive critics designs. In: Sharkawi M, Lerman K, Galstyan A, eds. Proc. of the IEEE Swarm Intell. Symp. Piscataway: IEEE Press, 2005. 393−396. [doi: 10.1109/SIS.2005.1501649]

[12] Zhang W, Liu Y, Clerc M. An adaptive PSO algorithm for reactive power optimization. In: Wong KP, ed. Proc. of the 6th Int'l Conf. on Advances in Power System Control, Operation and Management. Hong Kong: Institution of Electrical Engineers, 2003. 302−307.

[13] Zhan ZH, Zhang J, Li Y, Chung HS. Adaptive particle swarm optimization. IEEE Trans. on System, Man, and Cybernetics, 2009, 39(6):1362−1381. [doi: 10.1109/TSMCB.2009.2015956]

[14] Xie XF, Zhang WJ, Yang ZL. A dissipative particle swarm optimization. In: Tumer K, Wolpert D, eds. Proc. of the IEEE Int'l Conf. on Evolutionary Computation. Piscataway: IEEE Press, 2002. 1456−1461. [doi: 10.1109/CEC.2002.1004457]

[15] Ratnaweera A, Halgamuge SK, Watson HC. Self-Organizing hierarchical particle swarm optimizer with time-varying acceleration coeffients. IEEE Trans. on Evolutionary Computaion, 2004,8(3):240−255. [doi: 10.1109/TEVC.2004.826071]

[16] He R, Wang YJ, Wang Q, Zhou JH, Hu CY. An improved particle swarm optimization based on self-adaptive escape velocity. Journal of Software, 2005,16(12):2036−2044 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/2036.htm [doi: 10.1360/jos162036]

[17] Tao XM, Xu J, Yang LB. Improved cluster algorithm based on *k*-means and particle swarm optimization. Journal of Electronics & Information Technology, 2010,32(1):92−97 (in Chinese with English abstract).

[18] Peng Y, Peng XY, Liu ZQ. Statistic analysis on parameter efficiency of particle swarm optimization. Acta Electronica Sinica, 2004, 32(2):209−213 (in Chinese with English abstract).

附中文参考文献:

[16] 赫然,王永吉,王青,周津慧,胡陈勇.一种改进的自适应逃逸微粒群算法及实验分析.软件学报,2005,16(12):2036−2044. http://www.jos.org.cn/1000-9825/16/2036.htm [doi: 10.1360/jos162036]

[17] 陶新民,徐晶,杨立标,刘玉.一种改进的粒子群和 *K* 均值混合聚类算法.电子与信息学报,2010,32(1):92−97.

[18] 彭宇,彭喜元,刘兆庆.微粒群算法参数效能的统计分析.电子学报,2004,32(2):209−213.

陶新民(1973－),男,安徽蚌埠人,博士,副教授,主要研究领域为进化计算,网络安全,智能信号处理.

刘玉(1985－),男,硕士,主要研究领域为智能计算,信号处理.

刘福荣(1970－),女,博士,副教授,主要研究领域为故障诊断,群智能计算,人工智能.

童智靖(1985－),男,硕士,主要研究领域为信号处理,模式识别.