# 改进 PSO 算法的性能分析与研究 *

雷秀娟, 付阿利, 孙晶晶
(陕西师范大学 计算机科学学院, 西安 710062)

**摘 要**: 分析了粒子群优化(PSO)算法的进化式,针对其容易发生早熟、收敛速度慢、后期搜索性能和个体寻优能力降低等缺点,结合遗传算法的思想,提出一种新的混合 PSO 算法——遗传 PSO(GAPSO)。该算法是在 PSO 算法的更新过程中,对粒子速度引入遗传算法的变异操作,对粒子位置引入遗传算法交叉操作。对速度的变异降低了算法后期因种群过于密集而陷入局部最优的可能,对位置的交叉使得父代中优良个体的基因能够更好地遗传给下一代,从而得到更优、更多样化的后代,加快进化过程,提高了收敛速度和群体搜索性能。选取了其他几种典型的改进 PSO 算法,从算法执行过程、参数设置及优化性能等方面对各算法进行全面的分析比较,其中对模拟退火 PSO 算法采用了一种新的可提高算法执行速度的退火方式。最后针对选取的六个 Benchmark 函数优化问题进行数值仿真实验。仿真结果表明了所提出的遗传 PSO 算法不但收敛速度加快,而且后期搜索性能提高,能更有效地收敛到全局最优。为了形象地显示粒子的收敛过程,还仿真了 GAPSO 算法对二维多模态 Griewank 函数的动态寻优过程。

**关键词**: 粒子群优化(PSO); 遗传 PSO; 二阶振荡 PSO; 量子 PSO; 模拟退火 PSO
**中图分类号**: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2010)02-0453-06
doi:10.3969/j.issn.1001-3695.2010.02.013

# Performance analyzing and researching of improved PSO algorithm

LEI Xiu-juan, FU A-li, SUN Jing-jing
(*School of Computer Science, Shaanxi Normal University, Xi'an* 710062, *China*)

**Abstract**: To deal with the slow search speed, premature convergence and lower search performance and individual optimizing ability in late stage, this paper proposed a new PSO called genetic PSO. Produced mutation and crossover of GA into velocity and position updating of PSO. The mutation to velocity could reduce the possibility of the algorithm trapping in the local optimal because of the over dense of the population in late stage. The crossover to position could make the gene of excellent elder individuals passed down to the next generation, and by doing so, attained the more excellent and more various next generations, so increased the evolution and search performance of the population. Selected several other typical improved PSO algorithms for comparing and analyzing from implementing process, setting of parameters and optimization performance. To simulated annealing PSO, proposed a new annealing method which could increase the speed of implementation of the algorithm. The simulation experiments were done to the six selected Benchmark functions. The results show that the proposed algorithm not only speeded up the convergence, but also improved the search performance in late stage and could converge to the global optimal solution more efficiently. And lastly, presented the simulation of dynamic optimizing process of genetic PSO to the Griewank function so that converging process of the particles could be viewed vividly.

**Key words**: particle swarm optimization(PSO); genetic PSO; SOPSO; QPSO; SAPSO

## 0 引言

粒子群优化(PSO)算法是由 Kennedy 等人[1]于 1995 年提出的一种基于种群搜索的自适应进化计算技术。PSO 算法实现容易、参数较少,能有效解决复杂优化任务[2],在过去几年中得到了飞速发展,并在图像处理、模式识别、优化等领域得到了广泛应用。

PSO 算法作为一种崭新的随机搜索算法存在着易陷入局部极值点、进化后期收敛速度慢、精度较差等不足。针对这些问题,研究人员从各个方面对 PSO 算法进行了改进,涌现了大量的研究成果。主要的改进方法表现为改变参数、改变进化方程以及与其他智能优化算法的融合等。文献[3]给出了一种自适应逃逸粒子群算法,算法中的逃逸行为是一种简化的速度变异操作,通过速度的自适应变化改变微粒在搜索空间内的飞行速度,使得 PSO 算法收敛速度得到提高。但是该变异操作[3]仅仅增加了个体微粒的搜索性能,在算法迭代后期没有充分利用种群间的相互信息,对于 Schaffer 问题仍然存在收敛速度较慢、不能快速逃出局部极小点的缺点。

本文结合遗传算法的思想和文献[3]中的逃逸思想,提出一种新的融合 PSO 算法——遗传 PSO。该算法是在速度变异的同时,对粒子位置执行一种交叉操作,通过交叉将上一代种群优良个体的基因遗传给下一代,使得一次进化得到的新种群趋于更优、更多样化状态。新算法和其他几种典型改进 PSO

**收稿日期**: 2009-06-02; **修回日期**: 2009-07-18 **基金项目**: 国家自然科学基金资助项目(60773224)

**作者简介**:雷秀娟(1975-),女,陕西长安人,副教授,博士后,主要研究方向为智能计算与智能优化(xjlei168@163.com);付阿利(1980-),女,陕西武功人,硕士研究生,主要研究方向为粒子群优化;孙晶晶(1986-),女,河南洛阳人,硕士研究生,主要研究方向为粒子群优化.

算法,包括标准 PSO(SPSO)、带收缩因子的 PSO(KPSO)、二阶振荡 PSO(SOPSO)、量子 PSO(QPSO)和模拟退火 PSO(SAPSO),对典型复杂函数优化的仿真结果进行对比,充分显示了新算法的快速收敛性和全局收敛性。

## 1 基本粒子群优化(PSO)算法[1]

PSO 算法最初是为了图形化地模拟鸟群优美而不可预测的运动。PSO 初始化为一群随机粒子,然后通过迭代找到最优解。在每一次迭代中,粒子通过跟踪两个极值来更新自己:第一个就是粒子本身所找到的最优解,这个解叫做个体极值 pbest;另一个极值是整个种群目前找到的最优解,这个极值是全局极值 gbest。在找到上述两个最优值时,粒子根据如下公式来更新自己的速度和新的位置:

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_1 (\text{pbest}_{id}^{k} - x_{id}^{k}) + c_2 r_2 (\text{gbest}_{id}^{k} - x_{id}^{k}) \quad (1)$$

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

其中:$c_1$ 和 $c_2$ 为加速常数(acceleration constants),通常 $c_1$、$c_2$ 在[0,4],一般取 $c_1 = c_2 = 2$;$r_1$ 和 $r_2$ 为两个在[0,1]内服从均匀分布的随机变量。每个粒子的位置和速度都以随机方式进行初始化,而后粒子的速度就朝着全局最优和个体最优的方向靠近。

## 2 各种改进 *PSO* 算法

### 2.1 标准 *PSO* 算法(*SPSO*)

为了使粒子保持运动惯性,使其有扩展搜索空间的趋势,有能力探索新的区域。Shi 等人[4]提出了对基本粒子群算法的改进,即对速度更新方程加惯性权重 $w$:

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 (\text{pbest}_{id}^{k} - x_{id}^{k}) + c_2 r_2 (\text{gbest}_{id}^{k} - x_{id}^{k}) \quad (3)$$

分析式(3)可以发现,粒子群到达局部最优附近时,粒子速度的更新主要由第一项来决定。由于固定参数的 PSO 算法的惯性权重 $w$ 通常小于 1,粒子的速度将会越来越小,甚至停止运动,发生早熟收敛。对此,很多学者研究了自适应改变惯性权重,如线性递减惯性权重。

### 2.2 带收缩因子的 *PSO* 算法(*KPSO*)

文献[5]的研究发现,压缩因子 $k$ 有助于确保 PSO 算法收敛,它将速度更新方程描述为

$$v_{id}^{k+1} = k[v_{id}^{k} + c_1 r_1 (\text{pbest}_{id}^{k} - x_{id}^{k}) + c_2 r_2 (\text{gbest}_{id}^{k} - x_{id}^{k})] \quad (4)$$

其中:$k = \dfrac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}$,$\varphi = c_1 + c_2$。

收缩因子法控制系统行为最终收敛,且可以有效搜索不同的区域,该算法能得到高质量的解。

将这两种改进策略综合到一起形成带收缩因子和线性递减惯性权重的粒子群优化算法(KPSO),此算法目前效果较其他改进算法好。在进行速度更新时,将惯性权重和收缩因子同时用在速度更新方程中:

$$v_{id}^{k+1} = k[wv_{id}^{k} + c_1 r_1 (\text{pbest}_{id}^{k} - x_{id}^{k}) + c_2 r_2 (\text{gbest}_{id}^{k} - x_{id}^{k})] \quad (5)$$

### 2.3 二阶振荡 *PSO* 算法(*SOPSO*)

为了便于分析,在式(1)中取 $\varphi_1 = c_1 r_1$,$\varphi_2 = c_2 r_2$,将式(3)和(2)写成式(6)和(7):

$$v_i(t+1) = wv_i(t) + \varphi_1 (p_i(t) - x_i(t)) + \varphi_2 (p_g(t) - x_i(t)) \quad (6)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (7)$$

从控制论的角度分析,式(6)相当于分别以 $p_i$ 和 $p_g$ 为输入的两个惯性环节的并联。为了增加种群多样性,采用二阶振荡环节来代替惯性环节,那么,粒子群算法的进化方程式(6)和(7)可描述为

$$v_i(t+1) = wv_i(t) + \varphi_1 [p_i(t) - (1+\xi_1) x_i(t) + \xi_1 x_i(t-1)] + \varphi_2 [p_g(t) - (1+\xi_2) x_i(t) + \xi_2 x_i(t-1)] \quad (8)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (9)$$

即为二阶振荡粒子群算法[6]。

文献[6]已证明,当 $\xi_1 \geqslant \dfrac{2\sqrt{\varphi_1} - 1}{\varphi_1}$,$\xi_2 \geqslant \dfrac{2\sqrt{\varphi_2} - 1}{\varphi_2}$时,算法渐进收敛;当 $\xi_1 < \dfrac{2\sqrt{\varphi_1} - 1}{\varphi_1}$,$\xi_2 < \dfrac{2\sqrt{\varphi_2} - 1}{\varphi_2}$时,算法振荡收敛。在全局搜索算法中,一般地,希望算法前期有较高的探测能力以得到合适的粒子,而在后期有较高的开发能力以加快收敛速度。假设算法的最大迭代次数为 MaxIt,本文经过多次实验测试,在前 MaxIt/8 取 $\xi_1 < \dfrac{2\sqrt{\varphi_1} - 1}{\varphi_1}$,$\xi_2 < \dfrac{2\sqrt{\varphi_2} - 1}{\varphi_2}$,使得算法有较强的全局搜索能力;之后取 $\xi_1 \geqslant \dfrac{2\sqrt{\varphi_1} - 1}{\varphi_1}$,$\xi_2 \geqslant \dfrac{2\sqrt{\varphi_2} - 1}{\varphi_2}$,增强算法的局部搜索能力。

### 2.4 量子 *PSO* 算法(*QPSO*)

在量子物理学中,具有动量和能量的粒子状态可以用波动函数来表示。因此在 QPSO 模型[7]中,将粒子的运动状态用其波动函数 $\Psi$ 来表示,代替了标准 PSO 中粒子的速度和位置。QPSO 将每个个体看做是 $N_d$ 维搜索空间中的一个没有重量和体积的微粒在搜索空间中以一定的速度飞行,该飞行速度由个体的飞行经验和群体的飞行经验动态调整。每个粒子代表 $N_d$ 维空间中的一个位置,朝着个体最优和全局最优两个方向调整自己的位置。

QPSO 算法中粒子的更新式[7]为

$$\text{mbest} = \frac{1}{M}\sum_{i=1}^{M} p_i = \left(\frac{1}{M}\sum_{i=1}^{N} P_{i1}, \frac{1}{M}\sum_{i=1}^{N} P_{i2}, \cdots, \frac{1}{M}\sum_{i=1}^{N} P_{id}\right) \quad (10)$$

$$p = (\text{pbest} \times \varphi_1 + \text{gbest} \times \varphi_2)/(\varphi_1 + \varphi_2) \quad (11)$$

$$x(t+1) = p \pm \alpha \times |\text{mbest} - x(t)| \times \ln(1/u) \quad (12)$$

其中:$\varphi_1$、$\varphi_2$、$u$ 是在(0,1)均匀分布的随机数,叫做收缩—膨胀因子,是该算法中惟一的参数,对这个参数的取值文献[7]作了详细讨论。本文中设置 $\alpha$ 的取值在 1.0~0.5,呈线性递减。

### 2.5 模拟退火 *PSO* 算法(*SAPSO*)

SAPSO[8]是在 PSO 的每次迭代过程中加入模拟退火的思想,使得粒子 $i$ 在第 $t+1$ 步时,按照某一概率用 $x_i(t+1)$ 取代 $x_i(t)$,保证粒子不易陷入局部最优;同时采用温度 $T$ 来控制这一概率,温度 $T$ 随着算法的执行缓慢下降;若 $f(x_i(t+1))$ 差于 $f(x_i(t))$,随着温度的下降,用 $x_i(t+1)$ 取代 $x_i(t)$ 的概率不断减小,从而控制粒子使之不能从有“希望”的搜索区域中跳出。文献[8]中的退火参数由 $\beta$ 决定,$\beta$ 是一个取值在(0,1)的常数。本文结合 PSO 算法的执行过程,提出一种新的退火方式:

让温度随迭代次数的增加从初始温度线性递减到最低温度。假设初始温度和最低温度分别为 $T_0$ 和 $T_{end}$,最大迭代次数和当前迭代次数分别为 MaxIt 和 iter,则当前温度:

$$T_i = (T_0 - T_{end}) \times (\text{MaxIt} - \text{iter})/\text{MaxIt} + T_{end} \quad (13)$$

这样在 PSO 算法的开始,温度较高种群以较大概率接受差解,保证粒子可以跳出局部最优;随着迭代次数增加,温度降低,接受差解的概率较低,可以保证算法后期粒子在某个局部空间进行细致搜索。算法描述如下:

```
Initialize Population, T0, Tend, MaxIt and so on;
do
  for iter = 1 to MaxIt
    Ti = (T0 - Tend) × (MzaIt - iter)/MaxIt + Tend;
    for i = 1 to Population Size
    if f(xi) < f(pi) then pi = xi;
    pg = min(pi);
    for d = 1 to Dimension
    vtemp-d = wvid + c1r1(pid - xid) + c2r2(pgd - xid);
  if  vtemp-d > Vmax then  vtemp-d = Vmax;
    xtemp-d = xid + vtemp-d;
  next d;
  if f(xtemp) ≤ f(xi) then xi = xtemp;
  else
    xi = xtemp with Probability: pT = e^(-Δf(·)/Ti)
  next i;
next iter;
if termination criterion is met then stop;
```

## 3　新的具有遗传特性的 PSO 算法(GAPSO)

文献[3]结合生物界中物种发现生存密度过大时会自动分家迁移的习性,给出了一种自适应逃逸微粒群算法。本文在文献[3]算法思想的基础上,结合生物进化特征,给出一种不仅可以提高收敛速度,而且可增加种群多样性,提高种群进化质量的新型混合 PSO 算法——GAPSO 算法。

GAPSO 算法是对文献[3]中速度变异操作进行改进的同时引入对位置的一种交叉操作。该交叉操作可使 PSO 算法更好地摆脱局部极值点,提高算法的收敛速度和全局收敛性。该交叉操作的思想如下:

对一次更新后的种群,从中随机挑出 $m$ 个粒子,将这 $m$ 个粒子的当前位置 $X_i$ 与其当前个体极值 $p_i$ 及按适应度值排序后的个体极值中的前 $m$ 个极值($sp_i$)进行交叉,得到 $m$ 个新的粒子位置 $X_i'$。如果新位置的适应值 $f(X_i')$ 优于排序后对应个体极值的历史最优适应值 $f(sp_i)$,则用 $f(X_i')$ 取代 $f(X_i)$,同时用 $X_i'$ 取代 $X_i$。显然,这样的交叉使得粒子在一次进化中不但利用了自己的历史经验信息,而且利用了种群中优良个体的经验信息,增加了粒子的多样性,更增加了种群的进化质量,使粒子找到全局最优的可能性增大。

假定粒子的维数为 $d$,种群规模为 Popsize,则交叉操作的伪码如下:

```
M = floor(d/3); N = floor(2d/3);
随机产生 m 个大小在 1 ~ Popsize 的整数存放于 J 中;
for i = 1ń m
X'i = (pJ(i)1, pJ(i)2, …, pJ(i)M, XJ(i)(M+1), XJ(i)(M+2), …, XJ(i)N,
spi(N+1), spi(N+2), …, spid);
if f(X'i) < f(spi)
    XJ(i) = X'i;
    f(XJ(i)) = f(X'i);
end
```

分析式(3),迭代后期,当某些粒子的位置 $x_i$ 及其 pbest 接近群体的 gbest 时,其速度更新由 $wv$ 决定。由于 $w < 1$,粒子的运行速度将迅速趋于 0,粒子运行出现惰性。随着迭代的进行,其他粒子将很快聚集到这些惰性粒子的周围,使粒子过早地收敛于 gbest 而停止运动(粒子速度变为 0),而 gbest 只是种群目前找到的最好点,并不能保证一定是优化问题的全局最优解。

从以上分析可见,要让算法收敛到全局最优,就要使惰性粒子逃离局部最优点,而粒子接近 gbest 的程度与其速度大小有关,即可通过干扰惰性粒子速度使其跳出局部最优。

要使优化算法摆脱局部最优,首先要判断在什么情况下算法可能陷入局部最优。本文认为:当种群目前全局极值的适应值 $f$(gbest) > Err(Err 为给定误差精度)且种群中有惰性粒子出现时,算法有可能陷入局部最优。这时,对惰性粒子,即速度小于某个阈值的粒子速度进行变异,降低算法陷入局部最优的可能性。

假定速度阈值为 $Vth = (Vth_1, Vth_2, \cdots, Vth_d)$,种群规模为 Popsize,则本文速度变异的设计思路如下:

```
for j = 1ń d
  for i = 1ń Popsize
    if  Vid  < Vthd
      temp(j) = temp(j) + 1;
      if f(gbest) > Err
        flag = ture;
        Vid = Vmax × (2 × rand - 1); (Vmax为粒子的最大速度值,
rand 为(0,1)均匀分布的随机数)
        pid = Xid
      end
    end
  end
end
if flag = true
Err = Err/10;
end
t = find(temp > C1); (C1 为一个阈值常数,视种群的规模而定)
Vth1 = Vth1/C2;
end.
```

## 4　仿真实验及结果分析

为了显示新的 GAPSO 的高效性,本文选择了 PSO 算法和遗传算法(genetic algorithm,GA)经常使用的六个 Benchmark 函数问题[3]。根据函数性质分为具有单一极小点(单模态)($f_1 \sim f_3$)和多个局部极小点(多模态)($f_4 \sim f_6$)两大类,每类三个函数,如表 1 所示。Tablet 和 Quadric 函数是 Spherical 函数的变形,增加了函数各维之间的相互作用(Spherical 函数是一个连续的、简单单模态函数,通常用来分析算法的执行性能)。Rosenbrock 函数是一个经典复杂优化问题,它的全局最优点位于一个平滑、狭长的抛物线形山谷内。由于此函数仅仅为优化算法提供了少量信息,使算法很难辨别搜索方向,找到全局最小点的机会微乎其微,Rosenbrock 函数通常用来评价优化算法的执行效率[3]。函数 $f_4 \sim f_6$ 是典型的非线性多模态函数,它们具有广泛的搜索空间、大量的局部极小点和高大的障碍物,通常被认为是遗传算法很难处理的复杂多模态问题。本文的实验运行平台是 MATLAB 7.0,在内存为 1 GB,CPU 速度为 1.86 GHz 的双核 PC 机上运行。

表 1　用于测试的六个 Benchmark 函数

| Function | Name | Dim | Search Space | Min\Best position |
|---|---|---|---|---|
| $f_1(x)=10^6x_1^2+\sum_{i=2}^{n}x_i^2$ | Tablet | 30 | (-100,100) | 0　(0,…,0) |
| $f_2(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$ | Quadric | 30 | (-100,100) | 0　(0,…,0) |
| $f_3(x)=\sum_{d=1}^{D-1}[100(x_{d+1}-x_d^2)^2+(1-x_d)^2]$ | Rosenbrock | 30 | (-5.12,5.12) | 0　(1,…,1) |
| $f_4(x)=(1/4000)\sum_{d=1}^{D}x_d^2-\prod_{d=1}^{D}\cos\left(\frac{x_d}{\sqrt{d}}\right)+1$ | Griewank | 30 | (-300,300) | 0　(0,…,0) |
| $f_5(x)=\sum_{d=1}^{D}[x_d^2-10\cos](2\pi x_d)+10]$ | Rastrigin | 30 | (-5.12,5.12) | 0　(0,…,0) |
| $f_6(x)=\sum_{i=1}^{D-1}(x_i^2+x_{i+1}^2)^{0.25}[\sin2(50(x_i^2+x_{i+1}^2)^{0.1})+1.0]$ | Schaffer's $f_7$ | 30 | (-100,100) | 0　(0,…,0) |

### 4.1　参数设置

所有实验的最大迭代次数均设为 2 000 次。除 QPSO 算法外,其他算法均采用阻尼墙策略[9]处理超出边界问题;对于 SAPSO 采用本文提出的线性递减退火策略。经过参数分析和多次实验,最终对各个算法的相关参数的设置如表 2 所示,其中惯性权重 $w$ 的取值:SPSO、KPSO 和 SAPSO,单模态函数取 0.6,多模态函数取[0.9,0.4]线性递减;GAPSO,单模态函数取 0.6,多模态函数取[0.6,0.5]线性递减;SOPSO 均取[0.9,0.4]线性递减。GAPSO 对每个函数的初始误差精度 *Err* 和速度阈值 *Vth* 的设定如表 3 所示,$Vth=\text{rand}\times Mvth$。

表 2　实验中各个算法的参数设置

| Algorithm | $w_1$ | $c_1$ | $c_2$ | α | $T_0$ | $T_{end}$ | $m$ | $C_1$ | $C_2$ | Popsize |
|---|---|---|---|---|---|---|---|---|---|---|
| SPSO | 0.6/[0.9,0.4] | 1.7 | 1.7 | - | - | - | - | - | - | 30 |
| KPSO | 0.6/[0.9,0.4] | 1.7 | 1.7 | - | - | - | - | - | - | 30 |
| SOPSO | [0.9,0.4] | 0.4 | 0.8 | - | - | - | - | - | - | 30 |
| 2PSO | - | - | -[1.0,0.5]- | | - | - | - | - | - | 30 |
| SAPSO | 0.6/[0.9,0.4] | 1.7 | 1.7 | - | leb | - | - | - | - | 30 |
| GAPSO | 0.6/[0.6,0.5] | 1.7 | 1.7 | - | - | - | 10 | 10 | 10 | 30 |

表 3　GAPSO 针对各函数的初始误差精度和速度阈值

| Function | Tablet | Quadric | Rosenbrock | Griewank | Rastrigin | Schaffer's $f_7$ |
|---|---|---|---|---|---|---|
| *Err* | 1e-50 | 1e-5 | 1e-5 | 1e-50 | 1e-5 | 1e-5 |
| *Mvth* | 1e-40 | 1e-8 | 1e-8 | 1e-40 | 1e-8 | 1e-8 |

对每个测试函数分别用每种算法做 10 次独立实验,图 1~6分别表示七种算法对六个函数进行优化时 10 次实验的平均种群适应值变化趋势(图中为了曲线变化对比明显,纵轴是对平均种群适应值取了自然数对数,横轴是种群的进化代数)。SAPSO 算法采用两种退火策略的结果对比如表 4 所示。对于各测试函数每种优化算法,运行 10 次求得的结果的最小值($f_{Min}$)、最大值($f_{Max}$)、平均值($f_{Mean}$)以及所得的最小运行时间($t_{Min}$)、最大运行时间($t_{Max}$)、平均运行时间($t_{Mean}$),如表 5 和 6 所示。

表 4　SAPSO 算法采用两种退火策略的结果对比

| Function | 线性递减退火策略的 SAPSO 算法 | | | | β参数控制退火的 SAPSO 算法 | | | |
|---|---|---|---|---|---|---|---|---|
| | $f_{Min}$ | $f_{Max}$ | $f_{Mean}$ | $t_{Mean}$/s | fMin | fMax | fMean | tMean/s |
| Tablet | 4.3547e-037 | 3.6908e-030 | 3.7552e-031 | 3.0531 | 9.4209e-293 | 9.3300e-023 | 9.3300e-024 | 17.0500 |
| Quadric | 0.0352 | 14.3857 | 2.7572 | 3.7297 | 2.6937 | 35.1592 | 15.2020 | 21.4875 |
| Rosenbrock | 0.7119 | 22.8059 | 16.3471 | 2.6375 | 15.3500 | 24.1027 | 19.4144 | 14.5594 |
| Griewank | 0 | 0.0957 | 0.0184 | 3.1141 | 0 | 0.0736 | 0.0162 | 17.3016 |
| Rastrigin | 15.2192 | 38.1728 | 28.0068 | 3.3422 | 8.0211 | 16.9143 | 13.7513 | 18.5297 |
| Schaffer'$f_7$ | 0.0041 | 13.8293 | 3.1182 | 3.9828 | 11.582e-004 | 5.1592 | 1.9718 | 34.0547 |

### 4.2　两种退火策略下 SAPSO 算法结果对比

对于文献[8]中的 β 参数控制退火 SAPSO(β-SAPSO)算法,取退火参数 $\beta=9.6$,$T_0=100$,$T_{end}=0.1$,$c_1=c_2=1.8$,种群最大迭代次数为 100;对于本文提出的线性递减退火策略的 SAPSO(LD-SAPSO)算法,其参数设置同表 2,最大迭代次数为 3 000。用两种方法对六个函数进行优化,对每个函数都做 10 次独立实验,得到 10 次结果的最小值($f_{Min}$)、最大值($f_{Max}$)、平均值($f_{Mean}$)和算法平均运行时间($t_{Mean}$),结果如表 4 所示。

从表 4 可以看出,对于单模态函数,LD-SAPSO 比 β-SAPSO 优化结果好;对于多模态函数,β-SAPSO 比 LD-SAPSO 优化结果好一些。但 β-SAPSO 运行消耗的时间要比 LD-SAPSO 多几倍。因此,在实际应用中,对于单模态和实时性要求较高的多模态优化问题,选择本文提出的 LD-SAPSO 效果更理想。

表 5 显示了单模态 Benchmark 问题的实验结果对比。从表 5 的实验结果中各个算法的最优值、最差值和平均值可以看出,GAPSO 算法在各个函数上都具有较快的收敛速度和全局搜索能力,特别是 Quadric 函数所得到的优化结果明显好于其他算法。对于复杂 Rosenbrock 函数优化问题,由于很难辨别搜索方向,其他算法单次实验结果存在较大差异,而 GAPSO 算法单次实验结果比较稳定,数据差异相对较小,在 Rosenbrock 函数优化问题上具有更好的稳定性和健壮性。

### 4.3　各种改进 PSO 算法对单模态 Benchmark 函数问题的实验结果比较分析

表 5　改进 PSO 算法在单模态 Benchmark 函数问题上的优化对比

| Function | Algorithm | $f_{Min}$ | $f_{Max}$ | $f_{Mean}$ | $t_{Mean}$/s |
|---|---|---|---|---|---|
| Tablet | SPSO | 7.5978e-039 | 2.6945e-029 | 2.7140e-030 | 1.8328 |
| | KPSO | 2.4370e-037 | 6.9668e-028 | 6.9700e-029 | 1.7813 |
| | SOPSO | 1.8096e-025 | 1.7728e-020 | 2.9188e-021 | 2.2625 |
| | QPSO | 5.7748e-019 | 1.8861e-015 | 5.2319e-016 | 2.6422 |
| | SAPSO | 1.8476e-037 | 2.0319e-029 | 2.2543e-030 | 1.8250 |
| | GAPSO | 2.1142e-043 | 3.1866e-035 | 3.2895e-036 | 2.0234 |
| Quadric | SPSO | 0.0028 | 0.0359 | 0.0131 | 2.2578 |
| | KPSO | 6.7760e-004 | 0.0280 | 0.0105 | 2.1797 |
| | SOPSO | 0.0144 | 0.7841 | 0.2705 | 2.9438 |
| | QPSO | 9.7295 | 66.2994 | 33.7382 | 3.0750 |
| | SAPSO | 0.0754 | 4.0617 | 0.7968 | 2.2359 |
| | GAPSO | 4.7928e-004 | 0.0203 | 0.0085 | 3.0906 |
| Rosenbrock | SPSO | 9.5518 | 73.5555 | 23.2376 | 1.5438 |
| | KPSO | 8.3805 | 76.0675 | 21.8435 | 1.5250 |
| | SOPSO | 15.1872 | 24.4350 | 21.4324 | 2.2828 |
| | QPSO | 25.2613 | 25.7271 | 25.5556 | 2.3500 |
| | SAPSO | 11.7406 | 26.5004 | 19.9360 | 1.5391 |
| | GAPSO | 6.9173 | 21.2778 | 17.3006 | 2.2281 |

从图 1、2 显示的单模态 30 维 Tablet 和 Quadric 函数的优化实验结果可以看出,GAPSO 算法能够保持较快的收敛速度,持续、有效地搜索全局最小点。最差的是 QPSO 算法,后期基本不再变化,这与其进化式中有对各粒子的极值取平均有关。SOPSO 算法前期下降缓慢,中间还出现停滞,这是因为 SOPSO 前期的取值使算法振荡收敛,微粒大幅度振动不能细致搜索,产生在最优点处振荡的可能。

从图 3 可看出,对于复杂单模态 30 维 Rosenbrock 函数问题,由于 Rosenbrock 山谷仅仅给算法提供了很少的信息,使算法不能有效地辨别搜索方向,图 3 中的所有算法都具有一个相对稳定的阶段,用来寻找搜索方向。在算法初始阶段,SAPSO

和 GAPSO 算法都具有较快的收敛速度,但由于 SOPSO 算法后期取渐进收敛,算法不能以相对大的步长进行搜索而导致出现稳定不再下降的局面;而 GAPSO 算法,由于后期的交叉和变异操作,使粒子在向全局方向进化的同时改变搜索步长,可在一个相对较短的时间内找到正确的搜索方向,从而保持继续下降。表 5 中 Rosenbrock 函数的最好值、最差值和平均值的数据还显示,GAPSO 具有较好的稳定性,单次执行结果之间的相差小于其他算法。

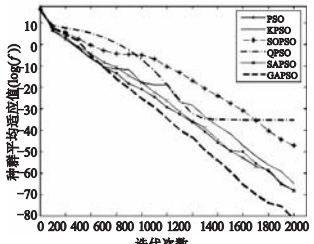


图1 30维Tablet函数优化对比图

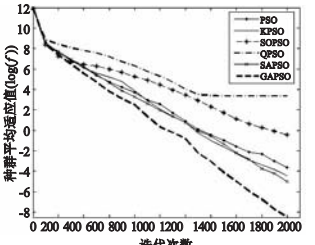


图2 30维Quadric函数优化对比图

### 4.4 各种改进 PSO 算法对多模态 Benchmark 函数问题的实验结果比较分析

对于 30 维多模态 Griewank 函数优化问题,表 6 中的数据显示,GAPSO 算法以 60% 的几率收敛到了全局最优值 0,其全局搜索能力显著优于其他改进算法。从图 4 可以看出,对于 Griewank 函数,除 GAPSO 和 QPSO,其他几个算法容易发生早熟陷入局部极小值,这是因为 Griewank 函数各维之间显著相关,存在多个极小值,某一维位置的变化并不能提高粒子的适应值,只有当各维同时发生变化时才有可能提高粒子的适应值,这使得粒子跳出极小点的概率非常小。GAPSO 在位置更新中,加入了对各维的交叉操作,保持了各维之间的相关性,能够有效跳出局部极小点在给定的迭代次数内找到近似全局最小点,显示了该算法在 Griewank 函数优化问题上的稳定性与健壮性,QPSO 虽然也可以跳出局部极小点,但其全局收敛速度很慢。

表 6　各种改进 PSO 算法在多模态 Benchmark 函数问题上的优化对比

| Function | Algorithm | $f_{Min}$ | $f_{Max}$ | $f_{Mean}$ | $t_{Mean}$/s |
|---|---|---|---|---|---|
| Griewank | SPSO | 0(20%) | 0.0344 | 0.0187 | 1.9047 |
| | KPSO | 0(30%) | 0.0764 | 0.0209 | 1.8625 |
| | SOPSO | 0(10%) | 0.1098 | 0.0355 | 2.3344 |
| | QPSO | 7.9425e-013 | 0.0380 | 0.0127 | 2.7563 |
| | SAPSO | 0(40%) | 0.0443 | 0.0152 | 1.9125 |
| | GAPSO | 0(60%) | 0.0369 | 0.0079 | 2.6625 |
| Rastrigin | SPSO | 20.4882 | 37.8900 | 28.6261 | 2.0656 |
| | KPSO | 12.6604 | 40.2683 | 26.5345 | 2.1875 |
| | SOPSO | 11.9395 | 30.8449 | 19.7998 | 2.7266 |
| | QPSO | 8.0860 | 15.9153 | 11.2560 | 2.7500 |
| | SAPSO | 16.1405 | 36.0287 | 25.9239 | 2.0172 |
| | GAPSO | 5.9862 | 15.9363 | 10.3586 | 2.7953 |
| Schaffer's $f_7$ | SPSO | 0.0332 | 3.3383 | 1.9165 | 3.9031 |
| | KPSO | 0.0554 | 6.1952 | 1.7281 | 3.8563 |
| | SOPSO | 0.0538 | 2.4496 | 0.8949 | 4.6750 |
| | QPSO | 11.0735 | 50.0168 | 32.3908 | 4.7500 |
| | SAPSO | 0.0124 | 3.7493 | 0.8985 | 3.9171 |
| | GAPSO | 0.0107 | 1.7970 | 0.4321 | 5.3828 |

多模态 Rastrigrin 函数是一个典型的用来测试算法全局搜索性能的函数。从表 5 和图 5 中可以看出,GAPSO 算法具有良好的全局搜索性能和较快的搜索速度,在文中给定的迭代次数下能找到近似全局最优值。同时,图 5 还显示 SOPSO 算法在搜索前期收敛速度较快,搜索后期由于渐进收敛,搜索步长较短,导致粒子很难跳出局部最优而陷入局部极小点,不能进一步向全局极小点收敛。QPSO 算法虽然可以向全局小点收敛,但其全局收敛速度很慢,需要给予足够长的迭代次数才能找到近似全局最优解。

图 6 显示了 30 维 Schaffer 函数的运行结果,可以看出 GAPSO 算法不论是在搜索的初期阶段还是在后期阶段都具有较快的收敛速度,能保持正确的搜索方向,持续地寻找全局最优点,显著优于其他算法。前期除 QPSO 外,各算法都具有较快的收敛速度,但在后期收敛其他算法收敛速度明显下降,原因在于 SAPSO 后期温度降低使其接受差解的概率变小,降低了搜索性能;SOPSO 后期渐进收敛,搜索步长相对小,很难跳过局部极值点。Schaffer 函数在向全局极小值靠近的过程中,有无穷多个极小值点。与 Griewank 函数一样,由于粒子没有充分利用各维之间的信息,其他算法在后期都需要一个缓慢的过程才能找到全局最优点。

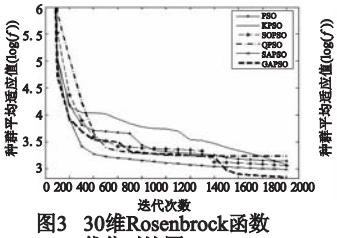


图3 30维Rosenbrock函数优化对比图

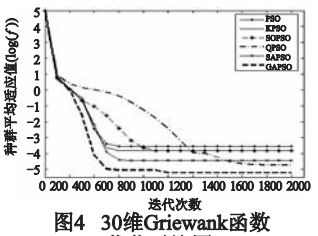


图4 30维Griewank函数优化对比图

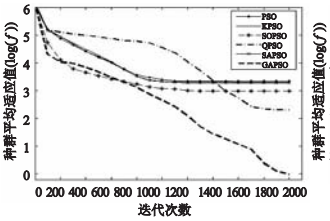


图5 30维Schaffer's $f_7$函数优化对比图

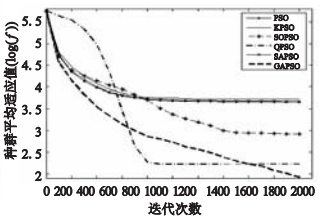


图6 30维Rastrigin函数优化对比图

从以上对比分析可看出,不论是对于单模态还是多模态 Benchmark 问题, GAPSO 算法均具有较高全局搜索能力,几乎可以在有限代数内找到函数近似理论最优值。从时间消耗上来看,虽然 GAPSO 相对于其他算法稍微有所增加,这是由于其在每一代粒子更新过程中加入了遗传和变异操作而增大了时间消耗,但是从平均消耗时间的结果来看,这种增加是在可以接受的范围之类。从算法设计的理论来讲,如果增加较小的时间消耗可以换取理想的效果,这种时间消耗就是值得的。

### 4.5 GAPSO 对 2 维 Griewank 函数的动态寻优过程

由于多模态函数存在大量的局部极小点,可以很好地检验一个算法是否具有跳出局部极值的能力。为了形象地看出 PSO 算法进行优化的动态过程, 本文仿真了 GAPSO 算法对二维多模态 Griewank 函数的动态寻优过程。仿真取群体规模为 30,最大迭代次数为 200,解空间边界为[ -5,5]。

图 7 是整个动态过程的部分动态截图, 其中图 7(a) ~ (d)分别为第 1、50、100、200 次寻优的结果。从图中可以看出粒子在第一代时随机地分布在函数的各个部位,随着迭代次数的增加逐渐向最小值位置(0,0,0)处移动,最终整个种群都收

敛到最小点。

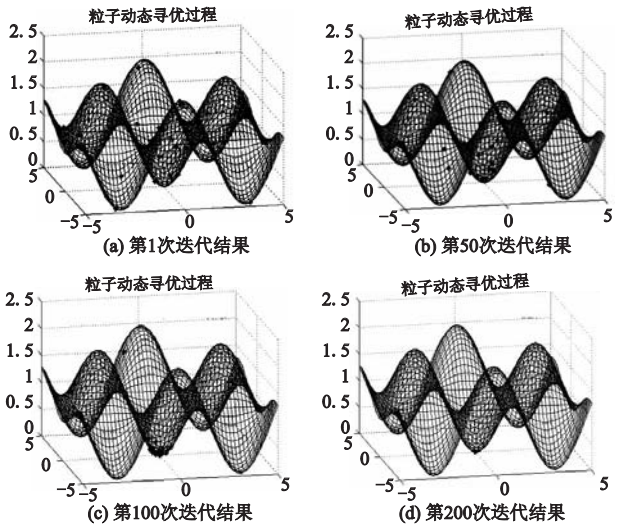


图7 二维Griewank函数动态寻优过程截图

## 5 结束语

本文分析了微粒群算法速度、种群最优值以及全局最优解之间的关系,将遗传算法的交叉和变异思想引入到粒子的位置和速度更新过程中,给出了一种解决数值优化问题的新的GAPSO算法。实验结果显示:

a)新算法不论是在单模态还是多模态 Benchmark问题上均好于现存 PSO及其改进算法,对位置更新引入的交叉思想增加了粒子的多样性和种群的进化质量,使粒子更容易找到正确的搜索方向,从而在多模态问题上易越过局部极值而向全局极值收敛;对速度变异加上判定条件,保证了变异操作不会使粒子远离全局极值,从而在单模态问题上能快速地向全局极值收敛。

b)新的 GAPSO算法提高了整个种群后期的搜索性能和最优个体的寻优能力,进一步提高了算法收敛速度,在 Shaffer问题上表现出了良好的性能。

c)交叉和变异操作的引入所增加的时间消耗在可接受的范围之内。

d)在多模态问题上 QPSO表现出的性能仅次于 GAPSO,但在单模态问题上性能却很差,其他改进算法针对不同函数表现出了各自的特点,但优化性能都不如提出的 GAPSO好。

本文不仅提出了一种新型改进 GAPSO,对模拟退火 PSO算法提出了一种新的退火方式,并对改进算法的关键参数作了深入分析,给出了 PSO算法优化的动态过程,并对目前的各种典型改进 PSO方法的仿真结果作了综合对比,选取的优化函数既有单模态又有多模态,也是优化问题的典型代表,以期对 PSO的深入研究作好前期的实验仿真基础和分析,并对以后研究 PSO算法在各种实际问题中的应用有所帮助。

**参考文献:**


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

[2] PARSOPOULOS K E, VRAHATIS M N. On the computation of all global minimizers through particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3):211-224.

[3] 赫然,王永吉,王青,等. 一种改进的自适应逃逸微粒群算法及实验分析[J]. 软件学报, 2005, 16(12):2036-2044.

[4] SHI Y, EBERHAART R C. A modified particle swarm optimizer [C]// Proc of Congress on Evolutionary Computation. Piscataway: IEEE Press, 1998:69-73.

[5] CLERC M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization[C]// Proc of the ICEC. [S.l.]: IEEE Press, 1999:1951-1957.

[6] 胡建秀, 曾建潮. 二阶振荡微粒群算法[J]. 系统仿真学报, 2007,19(5):997-999.

[7] SUN Jun, FENG Bin, XU Wen-bo. Particle swarm optimization with particles having quantum behavior[C]// Proc of Congress on Evolutionary Computation. 2004:325-331.

[8] 窦全胜, 周春光, 马铭. 粒子群优化的两种改进策略[J]. 计算机研究与发展, 2005,42(5):897-904.

[9] HUANG T, MOHAN A S. A hybrid boundary condition for robust particle swarm optimization[C]// Proc of IEEE Conference on Antennas and Wireless Propagation Letters. 2005:112-117.

[10] 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. 2005:3049-3054.

[11] 宋洁,董永峰,侯向丹,等. 改进的粒子群优化算法[J]. 河北工业大学学报, 2008, 37(4):55-59.

[12] 符杨,徐自力,曹家麟. 混合粒子群优化算法在电网规划中的应用[J]. 电网技术, 2008, 32(15):31-35.


(上接第 452 页)


[4] LEARY P, FRANCOIS O. BNT structure learning package: documentation and experiments[R]. 2004.

[5] FRIEDMAN N, MURPHY K, RUSSELL S. Learning the structure of dynamic probabilistic networks[C]// Proc of the 14th Conference on Uncertainty in Artificial Intelligence. 1998:139-147.

[6] SPELLMAN P T, SHERLOCK G, ZHANG M Q, et al. Comprehensive identification of cell cycle-regulated genes of the yeast saccaromyces cerevisiae by microarray hybridization[J]. MolBiolCell, 1998, 9(12):3273-3297.

[7] Home page of KEGG[EB/OL]. http://www.genome.ad.jp/kegg.

[8] KIM S, MOTO S, MIYANO S. Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data[J]. Biosystems, 2004, 75(1-3): 57-65.

[9] FRIEDMAN N. The Bayesian structural EM algorithm[C]// Proc of the 14th Conference on Uncertainty in Artificial Intelligence. San Francisco, CA: Morgan Kanfmann, 1998:571-578.

[10] ZHANG Yu, DENG Zhi-dong, JIANG Hong-shan, et al. Dynamic Bayesian network (DBN) with structure expectation maximization (SEM) formodeling of gene network from time series gene expression data[C]// Proc of International Conference on Bioinformatics & Computational Biology. Las Vegas, Nevada: [s.n.], 2006:26-29.

[11] 虞慧婷,吴骋,刘伟伟,等. 基因调控网络模型构建方法[J]. 第二军医大学学报, 2000, 27(7):737-740.

[12] MURPHY K, MIAN S. Modelling gene expression data using dynamic Bayesian networks[D]. Berkeley: Computer Science Division, University of California, 1999.

[13] 强波,王正志. 基于动态贝叶斯网构建基因调控网络[J]. 生物医学工程研究, 2008, 27(3):145-149.