文章编号：1000-6788(2008)05-0122-09

量子连续粒子群优化算法及其应用

吕　强[1]，陈如清[2]，俞金寿[3]

(1. 杭州电子科技大学 自动化学院，杭州 310018；2. 嘉兴学院 机电工程学院，嘉兴 314001；
3. 华东理工大学 自动化研究所，上海 200237)

摘要： 提出了基于量子理论的连续粒子群优化(Continuous Particle Swarm Optimization based on Quantum Methodology，CPSO-QM)算法，主要是采用了量子理论中的叠加态特性和概率表达特性.其中，叠加态特性可以使单个粒子表达更多的状态，潜在地增加了种群的多样性；概率表达特性是将粒子的状态以一定的概率表达出来.在基准函数的实验测试中，对比其它常用算法，结果显示本文提出的算法性能较好.在实际应用中，以丙烯腈反应器作为建模研究对象，提出了三种进化策略，实验结果显示，这三种策略训练的神经网络软测量模型都可以较好地预测丙烯腈的收率.

关键词： 进化算法；粒子群；量子计算；软测量模型

中图分类号： TP274　　**文献标志码：** A

Quantum continuous particle swarm optimization algorithm and its application

LV Qiang[1]，CHEN Ru-qing[2]，YU Jin-shou[3]

(1.School of Automation，Hangzhou Dianzi University，Hangzhou 310018，China；2.College of Mechanical and Electrical，Jiaxing University，Jiaxing 314001，China；3.Research Institute of Automation，East China University of Science and Technology，Shanghai 200237，China)

Abstract： Continuous particle swarm optimization algorithm based on quantum methodology is proposed in the paper. The algorithm mainly uses superposition characteristic and probability representation. Superposition characteristic can make a single particle present several states. In other words，the characteristic potentially increases population diversity. Probability representation is to make particle's state be presented according to a certain probability. Compared with other methods for test function in the experiment，the results demonstrate the proposed algorithm is better and more effective. Additionally，acrylonitrile reactor is used as modeling object in the real application. Three evolutional schemes are used. The experimental results show that the networks trained can the better predict the acrylonitrile yield through using three evolutional schemes.

Key words： evolutionary algorithm；particle swarm；quantum computing；soft sensing model

1　引言

粒子群优化算法(Particle Swarm Optimization，PSO)是由 Kennedy 和 Eberhart 等人于 1995 年提出的一种基于种群搜索的自适应进化计算技术.它来源于鸟类或鱼类觅食过程中迁徙和群集的模拟[1,2].由于 PSO 概念和参数调整都很简单而且容易编程实现，因此短短几年便获得快速发展.但是，就像遗传算法一样，PSO 算法也存在早熟收敛，这是一个很难克服的问题.很多学者提出了改进算法，试图克服该困难.例如：曾建潮[3]提出了一种能够保证以概率 1 收敛于全局最优解的粒子群算法，即随机 PSO 算法，同时给出了两种不同的停止进化微粒的重新产生方法；模拟退火算法和遗传算法，实验结果显示出该算法可以较好的搜索到全局最优解.赫然[4]提出了一种自适应粒子群算法，他通过控制微粒速度的方式来协调算法的种群多样性，同时利用群体自动分家特性，改善 PSO 算法的性能.还有很多学者提出了其它方面的改进措施，虽然他们都证明了在迭代次数无穷大时，算法将会收敛到全局最优值.但是，在实际应用中，迭代次数是有限

收稿日期：2007-01-19

作者简介：吕强(1977－)，男，辽宁抚顺人，博士，讲师，主要研究方向：群体智能技术，机器人的导航与控制；陈如清(1979－)，男，江西萍乡人，博士生，主要研究方向：过程智能建模、优化控制；俞金寿(1939－)，男，浙江海宁人，教授，博士生导师，主要研究方向：过程模型化及优化控制、先进控制等.

的,并且由于优化问题的复杂性不同,有些问题,改进算法可以很快地找到最优解,但有些问题,算法则很难找到最优解.因此,目前粒子群算法的研究就是使提出的改进算法,在有限的迭代次数内,尽可能的适应多的优化问题.

量子计算被广泛关注是在 20 世纪 90 年代中期,由于可以预期量子计算技术的强大计算能力,因此,考虑将量子计算中的一些基本概念引进到经典计算上,用来改善经典计算的性能,成为众多学者所关注的一个问题.例如:Han[5] 将量子概念和遗传算法相结合,从根本上改善遗传算法的性能,他提出的量子遗传算法在解决经典的组合优化问题上,显示出了很好的性能.在此基础上又有很多学者提出了改进算法,相应提高了算法的性能[6~8].由于量子概念的引进,所产生性能上的巨大提高,促使我们开始研究如何将量子概念和粒子群算法相结合,Sun[9~11] 提出了基于量子行为的粒子群优化算法,它的本质是模拟 Schrodinger 方程,这是封闭量子系统演化的一种方式,这种方法表现出非常好的性能.

本文提出了一种量子连续粒子群优化算法,主要是采用了量子理论中的叠加态特性和概率表达特性,其中,叠加态特性可以使单个粒子表达更多的状态,潜在地增加了种群的多样性.概率表达特性是将粒子的状态以一定的概率表达出来,本文是通过预先定义的选择概率,确定粒子基本态$|0\rangle$和$|1\rangle$中一个状态,参加运算的则是该发生状态的概率.基准函数测试的结果显示出该算法的有效性,然后本文采用该算法优化神经网络权值,并且提出了三种进化方案,以丙烯腈反应器作为建模研究对象,建立软测量模型来预测丙烯腈的收率,并对实验结果进行了讨论.

2　量子连续粒子群算法

2.1　粒子的表示方式

同文献[6]一样,粒子中的每一位采用量子比特的方式表示,称为量子位,量子位具有两个基本态,分别是$|0\rangle$态和$|1\rangle$态,在任意时刻,量子位的状态可以是基本态的线性组合,被称为叠加态,如式(1)所示:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

其中 α 和 β 是复数,并被称为概率幅,也就是说,我们得到量子位状态$|0\rangle$的概率是$|\alpha|^2$,得到量子位状态$|1\rangle$的概率是$|\beta|^2$.由于本文主要在实数域内讨论问题,所以 α 和 β 限定为实数,这样我们就可以写出量子位状态的几何表示方式,如式(2)所示:

$$|\varphi\rangle = \cos\theta|0\rangle + \sin\theta|1\rangle \tag{2}$$

其中 θ 为量子位的相位,并且和概率幅之间的关系满足(3)式.

$$\theta = \arctan\frac{\beta}{\alpha} \tag{3}$$

因此,粒子的量子表示方式可以通过使用概率幅或相位加以表示,如(4)式和(5)式.

$$\left[\begin{array}{c|c|c|c|c} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \beta_3 & \cdots & \beta_m \end{array}\right] \tag{4}$$

$$[\theta_1 \mid \theta_2 \mid \theta_3 \mid \cdots \mid \theta_m] \tag{5}$$

具体举例来说,如下所示:

$$\left[\begin{array}{c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{\sqrt{2}}{\sqrt{3}} & \frac{\sqrt{3}}{2} \end{array}\right] \tag{6}$$

则该粒子的状态是:

$$\frac{1}{2\sqrt{6}}|000\rangle + \frac{1}{2\sqrt{2}}|001\rangle + \frac{1}{2\sqrt{3}}|010\rangle + \frac{1}{2}|011\rangle + \frac{1}{2\sqrt{6}}|100\rangle + \frac{1}{2\sqrt{2}}|101\rangle + \frac{1}{2\sqrt{3}}|110\rangle + \frac{1}{2}|111\rangle,$$

可见该粒子的状态是由 8 个基本态的线性叠加而成,该粒子具有 8 个普通粒子的信息,最终粒子将以一定的概率坍塌成 8 个基本态中的一个,这样每个基本态都有机会表达出来,所以粒子量子的表示方式潜在增加了种群的多样性,也就扩大了算法的搜索空间,增大了找到最优解的几率.粒子初始化时概率幅表示方

式的参考代码为：

```
Procedure initialize
begin
    for j = 1 :n
      for d = 1 :m
        r = random[0,1] ;
        Popu(1,d,j) = r ;
        Popu(2,d,j) = (1 - r^2)^0.5 ;
      end for
    end for
end
```

其中：n 是粒子的个数，m 是粒子的维数，Popu 是粒子的概率幅表示方式，Popu(1，d，j)表示第 j 个粒子第 d 维的 α，Popu(2，d，j)表示第 j 个粒子第 d 维的 β.

将粒子的量子位的表示方式(4)根据式(3)转换成(5)的方式加以表示，粒子相位的表示方式参考代码如下.

```
Procedure change
begin
  for j = 1 :n
    for d = 1 :m
      LocPopuPhase(j,d) = atan(LocPopu(2,d,j)/LocPopu(1,d,j)) ;
      PopuPhase(j,d) = atan(Popu(2,d,j)/Popu(1,d,j)) ;
    end for
  end for
  for d = 1 :m
    GloPopuPhase(d) = atan(GloPopu(2,d)/GloPopu(1,d)) ;
  end for
end
```

其中：LocPopu 为粒子历史最优的概率幅表示方式，GloPopu 为粒子全局最优的概率幅表示方式，LocPopuPhase 为粒子历史最优的相位表示方式，PopuPhase 是粒子的相位表示方式，GloPopuPhase 是粒子全局最优的相位表示方式.

2.2 粒子的更新方式

在这里我们使用了基本的粒子群算法更新公式(也可以使用其它改进型)，主要是为了突出量子概念的引进对粒子群算法性能的贡献.所以，根据式(7)求出相移量.

$$\Delta\theta_{jd}^{t+1} = \omega \times \Delta\theta_{jd}^{t} + c_1 \cdot \mathrm{rand}() \cdot (\arctan(p_{jd}) - \arctan(y_{jd})) + c_2 \cdot \mathrm{rand}() \cdot (\arctan(p_{gd}) - \arctan(y_{jd}))$$
$$j = 1,2,\cdots,n; \quad d = 1,2,\cdots,m \tag{7}$$

其中：ω 为惯性权重，$\Delta\theta_{jd}^{t}$为在第 t 次迭代中第 j 个粒子第 d 维的相移量(当 $t = 1$ 时，$\Delta\theta_{jd}^{t} = \mathrm{rand}()$)，$c_1$ 是加速常数 1，rand()是均匀分布的随机数，$\arctan(p_{jd})$是第 j 个粒子历史最优值第 d 维的相位，$\arctan(y_{jd})$是第 j 个粒子当前第 d 维的相位，$\arctan(p_{gd})$是全局最优粒子第 d 维的相位，c_2 是加速常数 2.

根据得到的相移量，计算出量子旋转门(基本粒子群算法中第 2 步根据相移量相位更新，然后转换成概率幅的表示方式，将这两个步骤合并就是量子旋转门)，然后更新粒子，如公式(8)

$$\begin{bmatrix} \alpha_{jd}^{t+1} \\ \beta_{jd}^{t+1} \end{bmatrix} = \begin{bmatrix} \cos\Delta\theta_{jd}^{t+1} & -\sin\Delta\theta_{jd}^{t+1} \\ \sin\Delta\theta_{jd}^{t+1} & \cos\Delta\theta_{jd}^{t+1} \end{bmatrix} \begin{bmatrix} \alpha_{jd}^{t} \\ \beta_{jd}^{t} \end{bmatrix} \tag{8}$$

其中：$\Delta\theta_{jd}^{t+1}$为在第 $t+1$ 次迭代中第 j 个粒子第 d 维的相移量，α_{jd}^{t}、β_{jd}^{t}为在第 t 次迭代中第 j 个粒子第 d 维的概率幅，α_{jd}^{t+1}、β_{jd}^{t+1}是第 $t+1$ 次迭代中第 j 个粒子第 d 维的概率幅.粒子更新的具体参考代码如下：

```
Procedure update
```

```
begin
  for j = 1 :n
    for d = 1 :m
      if epock = = 1
        detaPhase (j,d) = rand() ;
      else
        detaPhase(j,d) = w * detaPhase(j,d) + c1 * rand()
          * (LocPopuPhase(j,d) - PopuPhase(j,d)) + c2 * rand() * (GloPopuPhase(d) - PopuPhase(j,d)) ;
      end if
      TempPopu(1,d,j) = Popu(1,d,j) * cos(detaPhase(j,d)) - Popu(2,d,j) * sin(detaPhase(j,d)) ;
      TempPopu(2,d,j) = Popu(1,d,j) * sin(detaPhase(j,d)) + Popu(2,d,j) * cos(detaPhase(j,d)) ;
    end for
  end for
  Popu = TempPopu ;
end
```

其中：epock **是迭代次数.**

2.3　粒子状态的观测

和通常的量子遗传算法不同,本文是当粒子坍塌成某一个基本态时,将该基本态发生的概率表达出来参加粒子适应度评估.假定算法搜索的空间为$[a,b]$**,由于某一状态发生的概率在**$[0,1]$**内,则需要将概率转化到**$[a,b]$**内,参考代码如下.**

```
Procedure observe
begin
    for j = 1 :n
        for d = 1 :m
            if rand(1) < p
                PopuValue(j,d) = Popu(2,d,j)^2 * (b - a) + a ;
            else
                PopuValue(j,d) = Popu(1,d,j)^2 * (b - a) + a ;
            end if
        end for
    end for
end
```

其中：p **为状态表达的选择概率,**$\mathrm{PopuValue}(j,d)$**为粒子测量后,发生状态的概率,并转换到算法搜索空间的值.**

2.4　收敛性说明和算法流程

本文提出的 CPSO-QM **算法主要具有** 2 **个特点,第一个特点是叠加态,此时,粒子的表示方式如式(**4**)表示,但是它和式(**5**)是对应的,这也是粒子的普通表示方式;第二个特点是概率表达,它的主要作用是用来计算粒子的适应度.因此我们能够看出** CPSO-QM **算法的主要更新机制仍然和基本粒子群算法相似,因此,本文提出的算法收敛性可以根据文献[**12**]的方法证明,将收敛于个体历史最优和群体最优中的某个权平衡点.**

具体的算法流程如下所示：

```
Procedure CPSO-QM
输入：优化问题
输出：最优解
begin
    set parameters
    initialize Popu
```

```
observe Popu
while(epock < = Max _ Generation) do
begin
  evaluate Popu
  store the global best position and fitness
  store personal best position and fitness
  change Popu
  update Popu
  observe Popu
  epock = epock + 1
end
end
```

其中：change Popu 是将粒子的概率幅表示方式转换成相位的表示方式，update Popu 是根据相移量，使用量子旋转门更新粒子，observe Popu 是对更新后的粒子进行测量，evaluate Popu 是根据测量后粒子计算该粒子的适应度.

2.5 算法的时间复杂性分析

假设待优化问题的规模为 $m(m>1)$（原子操作的运行时间近似相同），对于 CPSO-QM 算法，将体现在粒子的长度上，因此算法的时间复杂度为 $T(m)$. 考虑算法执行的主体部分：change Popu，update Popu 和 observe Popu，我们可知算法的时间复杂度是 $O(Epock\times(n(2m+1)+3nm+nm))$（$n$ 是种群规模，$Epock$ 是最大迭代次数）.

考虑本文对比算法的复杂性，经过分析算法主体部分，我们得到：RCQGA[7] 算法的时间复杂度为 $O(Epock\times(m\times 4n))$；AEPSO[4] 算法的时间复杂度为 $O(Epock\times(m\times(7n+1)+n))$；AQPSO[9] 算法的时间复杂度为 $O(Epock\times(m\times 4n+3n))$. 从时间复杂度可以看出，RCQGA 算法的时间复杂度最小，其次是 AQPSO 算法，然后是本文提出的 CPSO-QM 算法，最后是 AEPSO 算法，它的时间复杂度最大.

3 实验测试

本文将 CPSO-QM 算法和三种进化算法进行实验比较，它们是量子遗传算法 RCQGA[7]、普通改进型粒子群算法 AEPSO[4] 和量子粒子群算法 AQPSO[9]. 所有实验中 AEPSO 所用的参数设置为文献[4]给出的建议值：惯性权重为 0.7，加速常数为 1.4、1.4，标记条件值为 5/10，控制幅度值为 10. AQPSO 所用的参数为 Sun[9] 给出的建议值：收缩膨胀系数采用自适应的方式进行更改，具体更改方式可参见文献[9]. CPSO-QM 的参数设置是惯性权重 0.6，加速常数分别为 $c_1=1.6$、$c_2=1.8$，选择概率为 0.7（参数的选取是通过实验而统计得出的，具有一定的普遍性，实验过程本文没有列出）. 另外，我们知道对于进化算法，初始种群的位置对最后的收敛结果有很大的影响，如果认为，一个优化算法性能好，那么，其中一个方面就是指它的收敛结果对初始种群位置不敏感，也就是对初值鲁棒性好，因此，在我们的所有实验中，初始种群都是采用随机的方式产生.

所有实验种群规模均为 40，每个函数独立运行 50 次，每次运行 2000 代. 实验所用的硬件为 AMD Sempron 1.79GHz，512MB RAM 软件为 Windows XP 和 Matlab.

本文选择了优化算法经常使用的 6 个基准函数问题进行数值实验.

1）Tablet 函数

$$f(x)=10^6x_1^2+\sum_{i=2}^{n}x_i^2 \tag{9}$$

其中 $n=30$，$-100\leqslant x_i\leqslant 100$ 具有最小值 0.

2）Ackley 函数

$$f(x)=-20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n}x_j^2}}-e^{\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)}+22.71282 \tag{10}$$

其中 $n=30$，$-10\leqslant x_i\leqslant 10$ 具有最小值 0.

3）Griewangk 函数

$$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \tag{11}$$

其中 $n=30$，$-100\leqslant x_i\leqslant 100$ 具有最小值 0.

4）Rastrigin 函数

$$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10) \tag{12}$$

其中 $n=10$，$-5.12\leqslant x_i\leqslant 5.12$ 具有最小值 0.

5）Schwefel 函数

$$f(x)=418.9829n+\sum_{i=1}^{n}x_i\sin(\sqrt{|x_i|}) \tag{13}$$

其中 $n=10$，$-500\leqslant x_i\leqslant 500$ 具有最小值 0.

6）Rosenbrock 函数

$$f(x)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2) \tag{14}$$

其中 $n=20$，$-2.048\leqslant x_i\leqslant 2.048$ 具有最小值 0.

表 1 显示四种算法优化 6 个函数所得的最优值、平均值、标准差和平均时间等指标.

表 1　函数实验结果对比

函数	算法	最优值	平均值	标准差	平均时间
Tablet	RCQGA	1.277	3.1362	3.6974	5.99430
	AEPSO	1.6772	2.1537	0.3292	22.4323
	AQPSO	0.488	0.6375	0.1040	10.8217
	CPSO-QM	8.7157e-22	4.974e-14	3.517e-13	17.4840
Ackley	RCQGA	8.9130	12.1685	1.0313	6.9497
	AEPSO	8.1439	12.4751	1.3163	24.4011
	AQPSO	4.0260	5.15820	0.6134	12.6090
	CPSO-QM	1.3350	3.03030	0.9556	19.7610
Griewangk	RCQGA	4.8380	8.1637	3.3522	6.67210
	AEPSO	1.4943	5.4419	1.0449	22.9174
	AQPSO	0.1896	1.2494	0.2132	11.7028
	CPSO-QM	0	0.0963	0.1259	18.2166
Rastrigin	RCQGA	11.2659	30.5632	8.9187	3.009
	AEPSO	9.15660	16.1847	3.3879	12.196
	AQPSO	2.98800	12.9691	5.7875	6.3136
	CPSO-QM	0	3.4040	2.6667	9.8431
Schwefel	RCQGA	36.5022	71.1477	24.2947	3.6252
	AEPSO	28.4655	36.4609	7.6096	12.4923
	AQPSO	0.13130	7.75560	9.1505	6.51530
	CPSO-QM	1.2804e-04	0.2116	1.3028	10.3861
Rosenbrock	RCQGA	55.8751	193.6183	118.7841	4.2906
	AEPSO	90.8838	146.656	32.2596	22.2677
	AQPSO	22.0642	38.5762	15.9526	12.5935
	CPSO-QM	17.2517	43.9263	38.5442	19.5299

图 1，AQPSO 算法开始时好于 CPSO-QM 算法，但是曲线变化缓慢，最终值和初始值相差不大，算法寻找函数最优值的能力不强，相反 CPSO-QM 算法表现出较优的性能，算法能够不断地向全局最优值的方向

搜索，AEPSO 算法和 RCQGA 算法表现得较差．表 1 中的具体数据也显示出 CPSO-QM 算法的搜索最优值的能力好于 AQPSO，远好于 AEPSO 和 RCQGA．

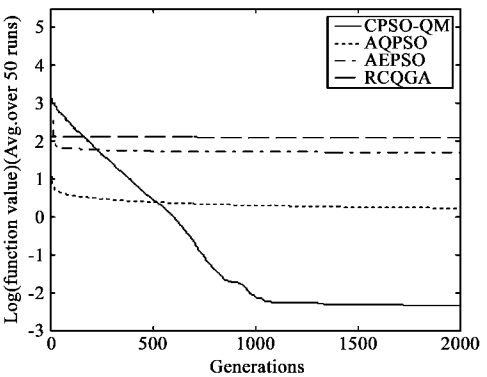

图 1 30 维函数 Griewangk 四种算法优化时，平均值的收敛图

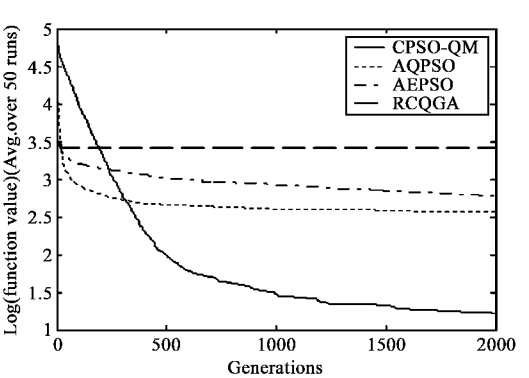

图 2 10 维函数 Rastrigin 四种算法优化时，平均值的收敛图

图 2，四种算法在迭代次数为 500 左右所得到的平均值就分出了高低，CPSO-QM 算法最好，其次是 AQPSO 算法、AEPSO 算法和 RCQGA 算法．图 3 我们可以看到，CPSO-QM 算法下降很快，而且在迭代次数到达最大值时，仍然保持下降的趋势，AQPSO 算法开始时也保持下降的趋势，但是在到达一定迭代次数后，算法则趋于平缓．表 1 显示出对于 Rastrigin 函数，在四种算法中，只有 CPSO-QM 算法能够找到全局最优值 0．对 Schwefel 函数，则无法找到全局最优值 0，但所获得的最优值、平均值相对较好．从前 5 种函数的优化结果可以看出，CPSO-QM 算法的优化能力强于其它三种算法．对于第 6 个函数，Rosenbrock 函数，表 1 中的结果显示出，四种算法的优化结果均不好，并且，从四种算法的结果对比看出，在最优值指标上，CPSO-QM 算法能够获得相对其它三种算法好的函数值，但是，在平均值指标上，AQPSO 算法获得的函数值要好于 CPSO-QM 算法．

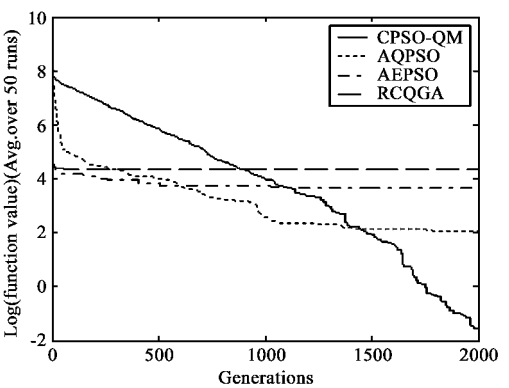

图 3 10 维函数 Schwefel 四种算法优化时，平均值的收敛图

4 量子进化神经网络软测量模型

4.1 进化方案

本文采用 CPSO-QM 算法优化三层前馈神经网络权值和阈值，主要研究了三种进化策略，它们分别是：

进化方案 A：该方案直接采用 CPSO-QM 算法，以网络的权值和阈值组成粒子，根据公式(15)计算粒子的适应度，按照 CPSO-QM 算法的步骤执行，在一定的迭代次数后，算法停止，将算法取得的全局最优值作为最终的网络权值和阈值．

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{l}(d_k^{(p)} - O_k^{(p)})^2 \tag{15}$$

式中，$d_k^{(p)}$ 为网络理想输出、p 为样本数、l 为输出层神经元个数．

进化方案 B：该方案将梯度下降训练法和 CPSO-QM 算法相结合，也就是说，执行一次 CPSO-QM 算法，然后执行梯度下降训练法一次，作为结合后算法的一次迭代，由于采用梯度下降训练法，可能会使粒子超出解空间，对于这种情况，本文采用的是将边界值乘上在[0,1]内随机产生的值．

进化方案 C：先采用梯度下降训练法，当训练误差在连续 5 次迭代都没有下降时，算法终止．以该网络的权值和阈值作为后面采用的 CPSO-QM 算法的一个粒子，然后再执行 CPSO-QM 算法．

4.2 实际应用

上海石化某厂年产 5 万吨丙烯腈装置,采用美国标准石油公司 Sohio 的生产工艺,以 C-41 作催化剂,以丙烯、氨、空气为原料,在沸腾反应器中一次直接氧化制取丙烯腈(即丙烯氨氧化法).在丙烯腈装置的生产过程中,丙烯腈的收率是一个关键指标.通过工艺分析,本文将反应压力、中段温度、纯丙烯量、空比、氨比、反应线速、触媒量作为模型的辅助变量.

采用该厂提供的反应器数据 342 组进行建模.并对原始数据进行数据处理(3σ 准则、七点线性平滑、归一化)后得到 317 组数据.将数据前 253 组组成训练样本集,另外 64 组数据组成测试样本集.

本文采用的神经网络网络结构为 7-25-1,BP 神经网络最大迭代次数为 30000,学习率为 0.0005.在三种进化方案中,种群数为 40,在进化方案 A 中迭代次数为 2000,进化方案 B 中迭代次数为 800,进化方案 C 中迭代次数为 500.算法的搜索空间为[-1.1,1.1].每种方案运行 20 次,结果取平均值(误差绝对值小于 1 的个数取近似整数),泛化结果见表 2.

从表 2 可以看出,进化方案 B 获得的泛化误差 0.4531 为最小,而且泛化最大误差 1.9376 也是最小的.进化方案 A 虽然获得的泛化误差较优,但是泛化最大误差 2.5755 却是四种算法中最大的,然而,误差绝对值小于 1 的个数是四种算法中最好的.BP 算法泛化最大误差 2.0973 好于进化方案 A,但是,泛化误差和误差绝对值小于 1 的个数这两个指标是最差的.进化方案 C 无论哪个指标,其值均处于中游水平.从泛化误差和误差绝对值小于 1 的个数这两个指标看,三种进化方案所得的网络均能较好的预测丙烯腈的收率.

表 2　泛化结果对比

模型	泛化误差	泛化最大误差	误差绝对值 $\lvert e\rvert \leqslant 1$
BP	0.6941	2.0973	47
进化方案 A	0.5439	2.5755	53
进化方案 B	0.4531	1.9376	52
进化方案 C	0.5999	1.9699	52

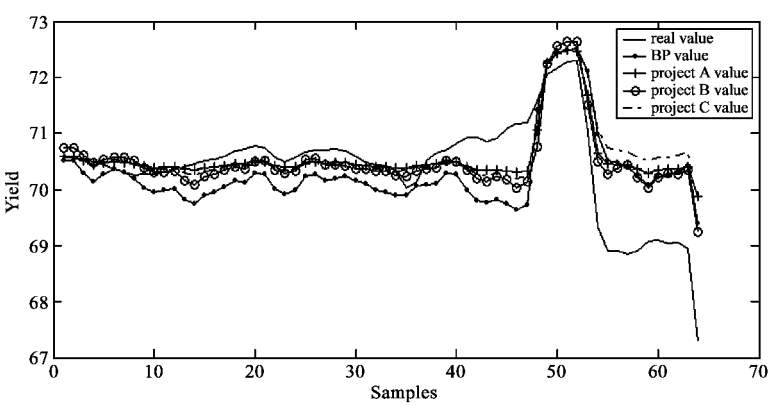

图 4　泛化结果对比

BP 算法和三种进化方案的丙烯腈收率预测值和真实值的对比如图 4 所示.

5　结论

本文提出了 CPSO-QM 算法,总的来说,该算法具有 2 个贡献:1)和其它的量子算法相比,本文提出的量子粒子群算法可以直接优化函数(其它算法往往把量子概念引进后,采用二进制的编码方法来优化函数);2)和基本的 PSO 算法相比,量子概念的引入,提高了 PSO 算法的性能,也高于普通改进型 PSO 算法,并且,从 CPSO-QM 算法的特点看出,我们可以采用其它改进的 PSO 算法更新机制来代替 CPSO-QM 算法的更新机制,这样新产生的量子算法优化性能会更好.因此,本文提出的算法提供了一种量子包容机制,它可以包容各种改进型的 PSO 算法.在函数测试中,我们看到了 AQPSO 算法和 CPSO-QM 算法的表现相对较好,特别是 CPSO-QM 算法在前 5 个测试函数上均有较优的表现.然后,本文将 CPSO-QM 算法作为神经网络软测量模型的训练算法,提出了三种进化策略,结果显示,三种方案训练的网络均可较好地预测丙烯腈的收率.

参考文献：

[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]//Proceedings of IEEE International Conference on Neural Networks. Perth, WA, Australia, 1995: 1942 – 1948.

[2] Kennedy J, Eberhart R C. A new optimizer using particle swarm theory[C]//Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya, Japan, 1995: 39 – 43.

[3] **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.
Zheng J C, Cui Z H. A guaranteed global convergence particle swarm optimizer[J]. Journal of Computer Research and Development, 2004, 41(8): 1333 – 1338.

[4] **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,** 2006,16(12): 2036 – 2044.
He R, Wang Y J, Wang Q, et al. An improved particle swarm optimization based on self-adaptive escape velocity[J]. Journal of Software, 2006, 16(12): 2036 – 2044.

[5] Han K, Kim J H. Genetic quantum algorithm and its application to combinatorial optimization problems[C]//Proceedings of the 2000 IEEE Conference on Evolutionary Computation, Piscataway, IEEE Press, 2000: 1354 – 1360.

[6] **王凌,吴昊,唐芳,等. 混合量子遗传算法及其性能分析[J]. 控制与决策,** 2005,20(2): 156 – 160.
Wang L, Wu H, Tang F, et al. Hybrid quantum genetic algorithms and performance analysis[J]. Control and Decision, 2005, 20(2): 156 – 160.

[7] **陈辉,张家树,张超. 实数编码混沌量子遗传算法[J]. 控制与决策,** 2005,20(11): 1300 – 1303.
Chen H, Zhang J S, Zhang C. Real-coded chaotic quantum – inspired genetic algorithm[J]. Control and Decision, 2005, 20(11): 1300 – 1303.

[8] Zhang G X, Li N, Jin W D. Novel quantum genetic algorithm and its applications[J]. Front Electr Electron Eng China, 2006, 1: 31 – 36.

[9] Sun J, Xu W B, Liu J. Parameter selection of quantum-behaved particle swarm optimization[C]//ICNC 2005, Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2005: 543 – 552.

[10] Sun J, Xu W B, Fang W. Quantum-behaved particle swarm optimization algorithm with controlled diversity[C]//ICCS 2006, Part III, Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2006: 847 – 854.

[11] Sun J, Feng B, Xu W B. Particle swarm optimization with particles having quantum behavior[C]//Proceedings of IEEE Conference on Evolutionary Computation, 2004: 326 – 331.

[12] Bergh F D, Engelbrecht A P. A study of particle swarms optimization particle trajectories[J]. Information Sciences, 2006, 176(8): 937 – 971.