doi:10.3772/j.issn.1002-0470.2014.05.006

基于量子行为粒子群优化的软件可靠性模型参数估计①

陈 晓② 江建慧

(同济大学软件学院 上海 201804)

摘 要 研究了准确估计软件可靠性模型参数这一难题,在分析现有软件可靠性模型参数估计方法的基础上,提出了一种基于量子行为粒子群优化(QPSO)算法的软件可靠性模型参数估计方法。针对该方法,选取了 7 个数据样本和三个软件可靠性模型(G-O 模型、M-O 模型和 Weibull 模型)进行了仿真实验,并且与粒子群算法、蚁群算法仿真实验进行对比。实验结果表明,基于 QPSO 算法的软件可靠性模型参数估计方法与粒子群算法、蚁群算法相比,求解精度高,误差较小,模型适应性强。

关键词 量子行为粒子群优化(QPSO),软件可靠性模型,参数估计

0 引 言

应用程序随着规模与复杂性迅速增大,其可靠度呈现降低趋势,因此对软件系统的可靠性评估与预测变得十分重要。软件可靠性模型与硬件可靠性模型一样,都是随机过程并且可以使用概率分布来描述,但是软件可靠性的变化不同于硬件可靠性,它不随时间的增加而降低。硬件可靠性依赖于分析静态过程,软件可靠性的增长或衰减不是静态的过程[1]。对大多数软件可靠性模型参数进行准确估计是非常困难的。本文研究了以往软件可靠性模型参数估计方法,尤其分析了利用孙俊提出的量子行为粒子群优化(quantum behaved particle swarm optimization, QPSO)算法[2]对软件可靠性模型参数估计的可行性,在此基础上提出了一种基于 QPSO 算法的软件可靠性模型参数估计方法,并通过实验验证了该方法的有效性。

1 相关研究

软件在测试或者维护过程中,随着故障被发现和消除,其失效率随着时间而降低。文献[3]给出了软件产品稳定失效率的概念。在软件产品发布后,软件生产商对软件系统的差错不断地排除,其失效率会随着时间而下降,理想状态下失效率将会趋于零。但是由于排错可能引入新的差错,因此经过一段时间,软件系统的失效率会逐渐达到一个稳定的值,即 λ_f,这段时间称为软件系统的稳定时间 t_f。这样,通过观察与测试得到大量的软件失效数据之后,便可以通过软件可靠性建模来预测失效率。

到目前为止,大约有近百种软件可靠性模型。文献[4]把软件可靠性模型中的大部分模型归为复杂的非线性函数。传统的参数估计方法有极大似然法和最小二乘法,当失效数据的样本规模较大时,常采用极大似然参数估计方法;而当失效数据的样本规模较小时,常采用最小二乘法。但传统的软件可靠性模型参数估计存在两方面的问题:(1)极大似然估计法与最小二乘法均含有概率论与数理统计相关特性,可能会破坏软件可靠性模型中参数估计的约束条件[5];(2)由于大多数软件可靠性模型均为复杂的非线性函数,并且当软件失效数据规模较大时,上述两种求解方法常不能求得模型参数的最优估计,在这种情况下一般要利用数值解法[6]。

文献[7]提出用人工神经网络来建立可靠性模型并进行预测,文献[8]将粒子群优化(particle swarm optimization,PSO)算法应用到求解软件可靠性模型中的参数,文献[9]利用支持向量机(support vector machine, SVM)对软件可靠性模型进行参数估计,文献[10]提出了基于相关向量机(relevance

① 863 计划(2007AA01Z142)资助项目。
② 男,1987 年生,博士;研究方向:软件可靠性,容错计算;联系人,E-mail: chenxiao_study@126.com
(收稿日期:2013-08-27)

vector machine, RVM)的软件可靠性模型,文献[11]通过蚁群算法对软件可靠性模型参数进行估计。然而,用神经网络模型求解软件可靠性模型参数时存在学习时间过长,当网络规模过大时会降低网络的泛化能力;由于 SVM 是借助二次规划来解决支持向量,当求解问题规模较大时,效率较低;而采用蚁群算法的估算模型存在参数对算法的收敛性影响较大,同时,规模较大时计算时间较长,蚁群算法收敛速度慢,极易陷入局部最优解;文献[12]通过引入 Solis 和 Wets 关于随机性算法的收敛准则,证明出标准 PSO 算法不能收敛于全局最优解,甚至不能收敛于局部最优解。

我国学者孙俊通过对人类思维和学习模式进行研究,在粒子群优化算法的基础上提出了量子行为的粒子群优化算法(QPSO)[2]。本文首先介绍 QPSO 算法的原理,然后结合软件可靠性模型的参数估计特点,分析利用 QPSO 算法来对软件可靠性模型参数估计的可行性,再结合软件可靠性模型给出具体的算法流程,分析算法针对软件可靠性模型参数估计的收敛性问题,最后通过实验来分析本文的算法模型,同时与 PSO、蚁群算法模型进行对比与分析。

2 QPSO 算法

2.1 PSO 算法

PSO 算法是一种群体智能算法,算法模拟鸟群飞行觅食行为,它是通过群鸟之间的集体协作与竞争来获得最优位置。设问题的搜索空间维度为 N,粒子规模为 M,每一个粒子有一个位置向量 $\boldsymbol{X}_i$,速度向量 $\boldsymbol{V}_i$,其中 $i = 1,2,\cdots,M$。单个粒子根据自己和种群中其它粒子飞行的历史轨迹不断地调整飞行方向,设粒子个体 i 的最优位置为 $\boldsymbol{P}_i$,那么 $\boldsymbol{P}_i^t = (p_{i,1}^t, p_{i,2}^t, \cdots, p_{i,N}^t)$,其中 t 为算法迭代次数,粒子个体 i 的当前速度为 $\boldsymbol{V}_i$,那么,粒子 i 的第 t 次迭代的速度为 $\boldsymbol{V}_i^t = (v_{i,1}^t, v_{i,2}^t, \cdots, v_{i,N}^t)$,粒子 i 在第 t 次迭代的位置为 $\boldsymbol{X}_i^t = (x_{i,1}^t, x_{i,2}^t, \cdots, x_{i,N}^t)$,粒子群搜索到的当前最好位置为全局最好位置,设为 $\boldsymbol{G}^t = \boldsymbol{P}_g^t = (p_{g,1}^t, p_{g,2}^t, \cdots, p_{g,N}^t)$,其中 $1 \leqslant g \leqslant M, g = \arg\min_{1\leqslant i\leqslant M}\{f[\boldsymbol{P}_i^t]\}$,$f$ 表示求解问题的适应度函数。粒子速度更新方程为

$$v_{i,j}^{t+1} = w \times v_{i,j}^t + c_1 \times r_{1,i}^t \times (p_{i,j}^t - x_{i,j}^t) + c_2 \times r_{2,i}^t \times (p_{g,j}^t - x_{i,j}^t) \quad (1)$$

粒子位置更新方程为

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1} \quad (2)$$

在式(1)与(2)中,$1 \leqslant i \leqslant M$, $1 \leqslant g \leqslant M$, $1 \leqslant j \leqslant N$, c_1 表示认知项系数,调节粒子飞向自身最优位置方向的步长,c_2 表示社会项系数,调节粒子飞向全局最优位置的步长;$r_{1,i}^t, r_{2,i}^t$ 是服从(0,1)间均匀分布,且相互独立,t 表示算法迭代次数。

PSO 算法为了降低粒子飞离搜索空间的概率,可以设置粒子的速度与位置的界限,设 $v_{i,j}^t \in [V_{\min}, V_{\max}]$, $V_{\min}$, $V_{\max}$ 分别表示粒子飞行速度的下界与上界,$x_{i,j}^t \in [X_{\min}, X_{\max}]$, $X_{\min}$, $X_{\max}$ 分别表示粒子位置的下界与上界。

2.2 QPSO 算法

在经典力学中,粒子的状态可以用坐标(位置)和动量(速度)来描述,而在量子力学中,粒子具有波动-粒子二象性,粒子的位置与速度具有概率的特征,因此粒子状态的描述是用波函数来表示的,即,$\Psi(\boldsymbol{X},t)$,$\boldsymbol{X} = (x,y,z)$ 是粒子在三维空间中的位置向量,波函数在空间中的某一点的强度(即波函数振幅绝对值的平方)和粒子在该点出现的概率成正比,即满足下式:

$$\iiint |\Psi(\boldsymbol{X},t)|^2 \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = 1 \quad (3)$$

表示粒子在空间某点出现的概率为 1。在量子力学中,粒子运动的动力学方程为薛定谔方程,它的描述如下:

$$i \times \hbar \times \frac{\partial}{\partial t}\psi(\boldsymbol{X},t) = \hat{H} \times \psi(\boldsymbol{X},t) \quad (4)$$

其中 t 表示时间,$\hbar$ 表示普朗克常数,$\hat{H}$ 表示哈密顿算子,式为

$$\hat{H} = -\frac{\hbar^2}{2 \times m} \times \nabla^2 + V(\boldsymbol{X}) \quad (5)$$

其中 m 表示粒子的质量,$V(\boldsymbol{X})$ 是粒子所在的势场。在 QPSO 算法中描述粒子运动的方程不再是经典力学中的牛顿运动方程,而采用的是量子力学中的薛定谔方程。

孙俊根据分析 PSO 算法中粒子收敛行为,得出必然存在以点 p_i 为中心的某种形式的吸引势,把该点称为粒子的吸引子,然后通过在吸引子建立一维 δ 势阱,推导出粒子在 δ 势阱中的定态薛定谔方程[13],解得粒子的定态波函数为

$$\phi(Y) = \frac{1}{\sqrt{L}} \times e^{-|Y|/L} \quad (6)$$

其中,$Y = X - p$,p 为粒子的吸引子,$\phi(X)$ 为定态

波函数,$L = 1/\beta = \hbar^2/m \times \gamma$。粒子出现在相对于吸引子 p 点位置 Y 的概率密度函数为 $|\phi(Y)|^2$,这样可以通过蒙特卡罗随机模拟的方式来测量粒子的位置。粒子在以 p 点为中心的一维 δ 势阱中运动,其位置由以下随机方程确定[13]:

$$X = p \pm \frac{L}{2} \times \ln(\frac{1}{u}) \tag{7}$$

其中,$L = 1/\beta = \dfrac{\hbar^2}{m \times \gamma}$,$u$ 是(0,1)区间上的均匀分布随机数。

根据式(1)、(2)和(7),可推导出具有量子行为的粒子进化方程,即

$$x_{i,j}^{t+1} = p_{i,j}^t \pm \alpha \times |C_j^t - x_{i,j}^t| \times \ln(\frac{1}{u_{i,j}^t}) \tag{8}$$

其中,$1 \leqslant i \leqslant M$,$1 \leqslant j \leqslant N$,$t$ 是算法迭代次数,M、N 依次表示粒子规模、求解空间的搜索维度。$\boldsymbol{C}^t$ 表示所有粒子最好位置的平均,即 $\boldsymbol{C}^t = (C_1^t, C_2^t, \cdots, C_N^t) = (\frac{1}{M}\sum_{i=1}^{M} P_{i,1}^t, \frac{1}{M}\sum_{i=1}^{M} P_{i,2}^t, \cdots, \frac{1}{M}\sum_{i=1}^{M} P_{i,N}^t)$,$x_{i,j}^t$ 表示粒子 i 在第 j 维、第 t 次迭代的位置,α 为搜索扩张系数,这个系数是需要我们在算法中控制的。可以通过两种方式来确定,一种是通过线性减小的方式,另外一种是通过设定固定值[14],$u_{i,j}^t$ 是(0,1)区间上的均匀分布随机数。$p_{i,j}^t$ 表示粒子 i 在第 j 维、第 t 次迭代中的吸引子,其值由下式决定:

$$p_{i,j}^t = \varphi_j(t) \times P_{i,j}^t + [1 - \varphi_j(t)] \times G_j^t \tag{9}$$

其中 $\varphi_j(t)$ 是(0,1)区间上的均匀分布随机数,$P_{i,j}^t$ 是粒子个体最优位置,G_j^t 是粒子全局最优位置。

QPSO 算法是采用概率方法对粒子的运动进行模拟的一类方法[13]。QPSO 算法提出至今,已经出现许多相关的应用与改进,被成功应用到系统故障诊断[15]、电力系统经济负荷分配[16]、多目标优化[17]、QoS 多播路由规划[18]等。

3　基于 QPSO 的软件可靠性模型参数估计

3.1　模型建立

建立模型时需要考虑的核心问题是适应度函数的设计。不妨设软件可靠性模型函数为 $F(t, E)$,其中 t 表示失效发生的时间,E 是软件可靠性模型的参数,设 E 含有 k 个参数,则 $E = (e_1, e_2, \cdots, e_k)$,$k$ 在软件可靠性模型中表示需要估计的参数个数,即问题的求解维度,$F(t,E)$ 表示在参数为 E,到时间 t 的累计失效数。根据最小二乘法的原理,设定模型估计的累计失效数与实际软件的累计失效数之差的平方平均数来表示模型的适应度函数,为 $f(\boldsymbol{E}) = \sqrt{\sum_{i=0}^{T}[F(i, E) - F(t)']^2/n}$,其中 T 表示终止测试时间,$F(i, \boldsymbol{E})$ 表示时刻 i 时的估计累计失效数,$F(t)'$ 表示时刻 i 时的实际软件累计失效数,n 表示样本数据规模。

根据 PSO 算法,软件可靠性模型参数估计的搜索维度为 k,粒子规模为 M,设 t 为算法迭代的当前次数,粒子 i 在第 t 次迭代的位置 $\boldsymbol{X}_i^t = (x_{i,1}^t, x_{i,2}^t, \cdots, x_{i,k}^t)$,$i = 1,2,3,\cdots,M$。根据 QPSO 算法,粒子没有速度向量,粒子个体 i 在第 t 次迭代时的最优位置 $\boldsymbol{P}_i^t = (P_{i,1}^t, P_{i,2}^t, \cdots, P_{i,k}^t)$,全局最优位置 $\boldsymbol{G}^t = \boldsymbol{P}_g^t = (P_{g,1}^t, P_{g,2}^t, \cdots, P_{g,k}^t)$,其中 $1 \leqslant g \leqslant M$,

$$g = \arg\min_{1 \leqslant i \leqslant M}\{f[\boldsymbol{P}_i^t]\} \tag{10}$$

当前适应度函数的值越小,那么粒子的位置就越好,求解的问题即软件可靠性模型的参数估计误差就越小,粒子个体 i 的最优位置由下式决定:

$$\boldsymbol{P}_i^{t+1} = \begin{cases} \boldsymbol{X}_i^{t+1}, f(\boldsymbol{X}_i^{t+1}) < f(\boldsymbol{P}_i^t) \\ \boldsymbol{P}_i^t, f(\boldsymbol{X}_i^{t+1}) < f(\boldsymbol{P}_i^t) \end{cases} \tag{11}$$

全局最好位置 $\boldsymbol{G}^t = \boldsymbol{P}_g^t$,而 $g = \arg\min_{1 \leqslant i \leqslant M}\{f[\boldsymbol{P}_i^{t+1}]\}$。根据式(9),粒子 i 在第 j 维、第 t 次迭代中的吸引子为

$$p_{i,j}^t = \varphi_j(t) \times P_{i,j}^t + [1 - \varphi_j(t)] \times G_j^t \tag{12}$$

其中,$\varphi_j(t)$ 是(0,1)区间上的均匀分布随机数,$j = 1,2,3,\cdots,k$,$P_{i,j}^t$ 是粒子个体最优位置,G_j^t 是粒子全局最优位置,算法在每次迭代时都会比较 $f(G_j^t)$ 与 $f(P_{i,j}^t)$,若 $f(G_j^t) > f(P_{i,j}^t)$,则更新 G_j^t。由式(8)、(11)、(12)得出粒子的进化方程为

$$x_{i,j}^{t+1} = p_{i,j}^t \pm \alpha \times |C_j^t - x_{i,j}^t| \times \ln(\frac{1}{u_{i,j}^t}) \tag{13}$$

其中,$u_{i,j}^t$ 是(0,1)区间上的均匀分布随机数,$C_j^t = \dfrac{\sum_{i=1}^{M} P_{i,j}^t}{M}$,粒子位置的下界与上界分别用 $X_{\min}$,$X_{\max}$ 表示,则 $x_{i,j}^t \in [X_{\min}, X_{\max}]$。

3.2　算法流程描述

根据 3.1 节中模型建立的参数设定,算法的流程如下:

步骤 1:参数初始化。算法开始时对粒子群初始化相关参数,包括粒子的初始位置、个体最好位置、粒子位置的下界与上界。当 $t = 0$ 时,粒子 i 的当前位置为 x_i^0,个体最好位置 $P_i^0 = x_i^0$。

步骤 2：更新粒子个体最优位置。计算粒子 i 的适应度值 $f(\boldsymbol{X}_i^{t+1})$，根据式(11)更新粒子个体的最优位置 $\boldsymbol{P}_i^{t+1}$，若 $f(\boldsymbol{X}_i^{t+1}) < f(\boldsymbol{P}_i^t)$，则 $\boldsymbol{P}_i^{t+1} = \boldsymbol{X}_i^{t+1}$，否则 $\boldsymbol{P}_i^{t+1} = \boldsymbol{P}_i^t$。

步骤 3：更新群体粒子群全局最优位置。计算 $f(\boldsymbol{G}^t)$，若 $f(\boldsymbol{P}_i^{t+1}) < f(\boldsymbol{G}^t)$，则 $\boldsymbol{G}^{t+1} = \boldsymbol{P}_i^{t+1}$，否则 $\boldsymbol{G}^{t+1} = \boldsymbol{G}^t$。

步骤 4：计算粒子的吸引子。根据式(12)计算粒子 i 在第 j 维、第 t 次迭代中的吸引子 $p_{i,j}^t$；

步骤 5：更新粒子位置。根据步骤 4 的结果以及式(13)更新粒子的位置；

步骤 6：检查粒子是否越界。根据设定的粒子位置的上下界，若 $x_{i,j}^{t+1} > X_{\max}$，则 $x_{i,j}^{t+1} = X_{\max}$；若 $x_{i,j}^{t+1} < X_{\min}$，则 $x_{i,j}^{t+1} = X_{\min}$。

步骤 7：根据 3.1 节中参数的设定，粒子群中每一个粒子执行步骤 2 到步骤 6。

步骤 8：检查算法设定的终止条件，即迭代次数 t 的上界，若不满足，则执行 $t = t + 1$，执行步骤 2 到步骤 7；否则退出算法。

3.3 算法收敛性分析

传统 PSO 算法粒子是在一个线性系统中不断进化，每个粒子在进化过程中不断向各自平衡点 p 逼近，这样就限制了粒子的搜索空间。对于 QPSO 算法，粒子是在一个复杂的非线性系统中进化，并且在算法初始时，粒子的寻优轨迹是不确定的，根据量子系统的特性，粒子会以一定的概率出现在任何位置，因此其搜索空间范围大，算法不易陷入局部最优，同时根据量子的束缚特性，粒子的搜索空间束缚的状态空间中，以概率的形式出现在某个位置，这样保证了算法的全局收敛性。

对于式(7)，文献[19]证明了对于粒子 $\boldsymbol{X}^t$ 是依概率收敛于吸引子 p 的充分必要条件是：当 $t \to \infty$ 时，$L(t) \to 0$。在软件可靠性模型的参数估计中，随着测试时间递增，软件的失效率会逐渐趋于一个稳定的值，即软件产品的稳定失效率。对于软件可靠性模型函数 $F(t,E)$，估计参数 E 在实际的可靠性模型中包含一系列的参数，如失效率、初始软件系统的期望失效数、失效率的衰减程度等。这样在算法训练的过程中，随着 t 的增加，这些参数会不断地趋于一个稳定值。

4 实验与结果分析

4.1 仿真实验

根据软件可靠性模型的分类，本文选择有限故障类泊松模型(G-O 模型)[14]、无限失效类泊松模型(M-O 模型)[20]、有限故障二项型威布尔模型(Weibull 模型)[1]。G-O 非齐次泊松过程模型：$F(t, E) = \alpha \times (1 - e^{-b \times t})$，其中，$\boldsymbol{E} = (a,b)$，$a$ 表示期望的总差错数，b 表示在时刻 t 每个差错被检测出来的概率，$F(t, \boldsymbol{E})$ 表示到时间 t 累计失效数的期望。M-O 对数泊松模型：$F(t,\boldsymbol{E}) = \ln(\lambda_0 \times \theta \times t + 1)/\theta$，$\lambda_0$ 表示初始失效率，θ 是失效率递减参数，$F(t, \boldsymbol{E})$ 表示到时间 t 累计失效数的期望。Weibull 模型：$F(t,\boldsymbol{E}) = a \times (1 - e^{-\beta \times t \times \partial})$，其中 $\boldsymbol{E} = (a,\beta,\partial)$，$a$ 表示期望的总差错数，β,∂ 是 Weibull 模型的参数，$F(t, \boldsymbol{E})$ 表示到时间 t 累计失效数的期望。

本文选取软件可靠性工程手册[1]中的 CSR1、CSR2、CSR3 和 SS3 以及 Musa 数据集中的 SYS1、SYS2、SYS3 共 7 组数据进行仿真实验。这些数据是以失效间隔时间序列形式给出的，在做模型参数估计时，需要将失效间隔时间序列数据转化为累计失效序列数据，同时除了 SS3 是以毫秒表示之外，其它的数据均是以秒来表示的，因此需要把 SS3 这组数据转化成秒。本文利用 7 组数据对 G-O 模型、M-O 模型和 Weibull 模型的参数进行估计，分别采取了 PSO 算法[8]、蚁群算法[11]以及本文提出的基于 QPSO 算法。实验中，式(13)中的搜索扩张系数从1.0线性减小至0.5，适应度函数采取 4.1 节中的 $f(\boldsymbol{E})$，每种求解模型迭代次数规定为 500，为了避免实验中的异常情形，每种求解模型运行 20 次，再取适应度的平均值。表 1 为 G-O 模型、M-O 模型和 Weibull 模型基于不同的数据集样本在不同的算法下的适应度值。从表中可以看出，基于 QPSO 算法的软件可靠性模型参数估计的误差(适应度值)均小于基于 PSO 算法与蚁群算法。

本文选取了 SYS1，CSR1 数据集的 G-O 模型、M-O 模型和 Weibull 模型的适应度值的演化过程，图 1 是基于数据集 SYS1 的 G-O 模型适应度值的演化过程，图 2 是基于数据集 CSR1 的 G-O 模型适应度值的演化过程，图 3 是基于数据集 SYS1 的 M-O 模型适应度值的演化过程，图 4 是基于数据集 CSR1 的 M-O 模型适应度值的演化过程，图 5 是基于数据集 SYS1 的 Weibull 模型适应度值的演化过程，图 6 是基于数据集 CSR1 的 Weibull 模型适应度值的演化过程。

表 1　参数估计的适应度值

	G-O 模型			M-O 模型			Weibull 模型		
	PSO 算法	蚁群算法	QPSO 算法	PSO 算法	蚁群算法	QPSO 算法	PSO 算法	蚁群算法	QPSO 算法
CSR1 数据	19.0970	18.1624	17.1602	23.2939	23.7154	20.3873	21.0463	18.4817	17.1494
CSR2 数据	10.8229	12.4149	8.3378	8.4162	7.2222	6.4343	12.6239	10.897	7.2541
CSR3 数据	9.3508	8.4840	7.2174	4.0108	3.3331	2.5374	7.0775	9.083	5.0824
SYS1 数据	12.7603	6.9745	5.8810	4.9041	10.7423	2.7098	7.2091	6.5302	5.881
SYS2 数据	21.5992	11.8593	2.6819	2.1722	8.2373	1.0964	1.7509	24.8244	1.28
SYS3 数据	4.4084	11.4970	3.3367	5.4001	11.8061	3.3015	4.2789	59.8881	3.279
SS3 数据	10.0794	19.9868	9.687	15.0793	33.4400	9.2633	21.0463	18.4817	17.1494

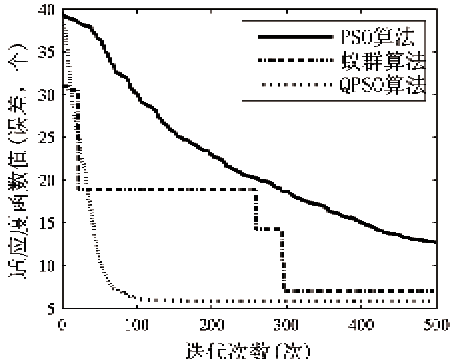

图 1　基于数据 SYS1 的 G-O 模型适应度值

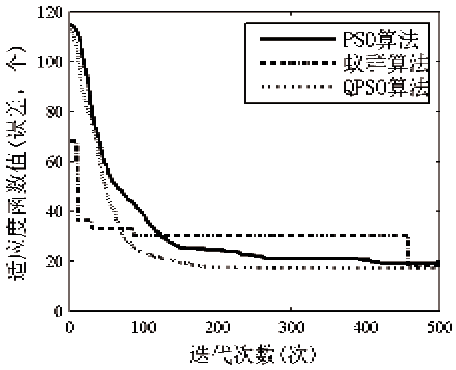

图 2　基于数据 CSR1 的 G-O 模型适应度值

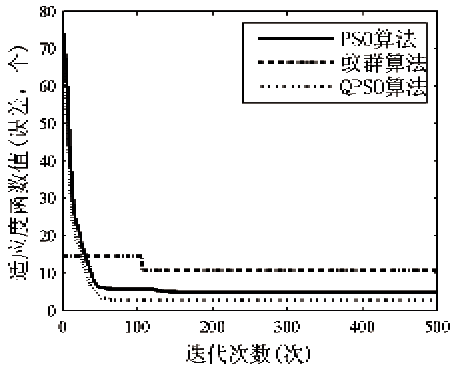

图 3　基于数据 SYS1 的 M-O 模型适应度值

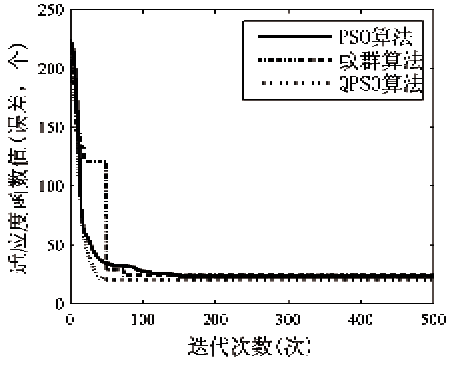

图 4　基于数据 CSR1 的 M-O 模型适应度值

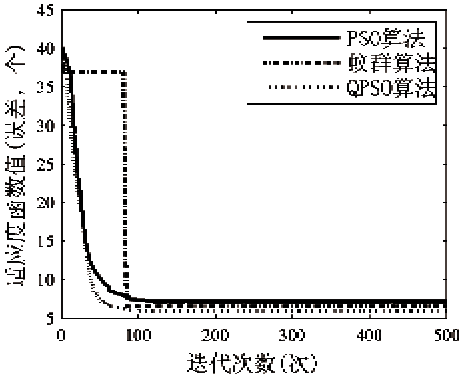

图 5　基于数据 SYS1 的 Weibull 模型适应度值

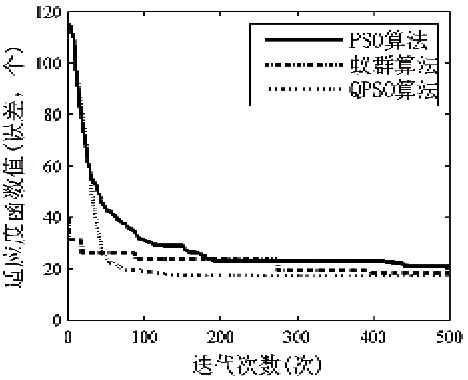

图 6　基于数据 CSR1 的 Weibull 模型适应度值

4.2 实验结果分析

从表 1 可以看出,基于 QPSO 算法对模型参数进行估计所得的误差最小,并且与 PSO 算法、蚁群算法比较,其误差值稳定。基于 7 组测试样本数据的两个软件可靠性模型的参数估计实验中,采用 PSO 算法与蚁群算法所得的结果不稳定,有的数据样本前者占优,而有的数据样本后者占优,因此其收敛性较差,而采用 QPSO 算法其结果对于所有数据集均是最小的,这说明该算法收敛性好,能求出参数估计的全局最优值。

从图 1 至图 6 可以看出,基于 QPSO 算法收敛迅速,能在较短的进化次数中达到最优值,而基于 PSO 算法收敛时间较长,并且其最优值也较大,基于蚁群算法的收敛性较差,在个别值处出现严重抖动,易陷入局部最优。如图 1 所示,QPSO 在第 150 代左右就已经达到了最优值,而 PSO 则到了 450 代左右才开始收敛,蚁群算法则在第 300 代才开始收敛,并且在寻优过程中抖动剧烈。

根据实验结果可以得出,采用 PSO 算法进行软件可靠性模型参数估计的寻优时间长,最优值效果较差,易陷入局部最优;采用蚁群算法进行软件可靠性模型参数估计,虽然寻优速度比 PSO 算法快,但是寻优过程抖动厉害,并且极易陷入局部最优。而采用本文提出的基于 QPSO 算法的软件可靠性模型参数估计,寻优速度快,算法收敛性好,能保证求得全局最优值。

5 结 论

软件可靠性模型众多,大多数模型均为复杂的非线性模型,怎样根据规模一定的数据进行模型的参数准确估计是一个难题。本文根据以往的软件可靠性模型的参数估计方法,提出了一种基于 QPSO 算法的软件可靠性模型参数估计方法,选取了 7 组失效数据、三个较常用的软件可靠性模型进行了仿真实验。通过与已有的效果较好的估计方法进行对比表明,该方法适应性强,对于不同的软件可靠性模型均能得出较好的参数估计结果,且算法能以较快的速度达到全局最优。但是 QPSO 算法也存在一些不足,当样本规模较大时,会出现寻优时间较长的问题。怎样改进 QPSO 算法,当软件失效数据样本规模很大时,优化算法寻优时间是下一步的研究工作。

参考文献

[1] Michael R L. Handbook of Software Reliability Engineering. New York:McGraw-Hill Book Company, 1996. 219-230

[2] Sun J, Feng B, Xu W. Particle swarm optimization with particles having quantum behavior. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, Portland, USA, 2004. 325-331

[3] Jalote P, Murphy B, Sharma V S. Post-release reliability growth in software products. *ACM Transactions on Software Engineering and Methodology*, 2008, 17(4): 1-20

[4] Musa J D, Launino A, Okumoto K. Software reliability: Measurement, prediction, application. New York: McGraw-Hill Book Company, 1987. 119-125

[5] Costa E O, De Souza G A, Pozo A T R, et al. Exploring genetic programming and boosting techniques to model software reliability. *IEEE Transactions on Reliability*, 2007, 56(3): 422-434

[6] Xie M. Software Reliability Modeling. Sweden: World Scientific, 1991. 113-120

[7] Khoshgoftaar T M, Szabor M. Using neural networks to predict software faults during testing. *IEEE Transactions on Reliability*, 1996, 45(3): 456-462

[8] Sheta A. Reliability growth modeling for software fault detection using particle swarm optimization. In: Proceedings of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, Canada, 2006. 3071-3078

[9] Tian L, Noore A. Dynamic software reliability prediction: An approach based on support vector machine. *International Journal of Reliability, Quality and Safety Engineering*, 2005, 12(2): 309-321

[10] Lou J G, Jiang J H, Shuai C Y. Software reliability prediction model based on relevance vector machine. In: Proceedings of the 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, Shanghai, China, 2009. 229-233

[11] 郑长友, 刘晓明, 黄松. 基于蚁群算法的软件可靠性模型参数估计方法. 计算机应用, 2012, 32(4): 1147-1151

[12] Bergh F V d. An Analysis of Particle Swarm Optimizers: [Ph. D dissertation]. Pretoria: natural and agricultural science of University of Pretoria, 2001. 21-29

[13] Kennedy J. Some issues and practices for particle swarms. In: Proceedings of the 2007 IEEE Swarm Intelligence Symposium, Honolulu, USA, 2007. 162-169

[14] Goel A L, Okumoto K. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, 1979, 28

(3):206-211

[15] 李晓. 基于粒子群算法和量子粒子群算法的电力系统故障诊断:[硕士学位论文]. 长沙:湖南大学电气与信息工程学院, 2010. 27-36

[16] Coelho L d S, Mariani V C. Particle swarm approach based on quantum mechanics and harmonic oscillator potential well for economic load dispatch with valve-point effects. *Energy Conversion and Management*, 2008, 49(11):3080-3085

[17] Omkar S N, Khandelwal R, Ananth T V S, et al. Quantum behaved particle swarm optimization (QPSO) for multi-objective design optimization of composite structures. *Expert Systems with Applications*, 2009, 36(8):11312-11322

[18] Sun J, Fang W, Wu X, et al. QoS multicast routing using a quantum-behaved particle swarm optimization algorithm. *Engineering Applications Intelligence*, 2011, 24(1):123-131

[19] Sun J, Fang W, Wu X J, et al. Quantum-behaved particle swarm optimization: analysis of the individual particle's behavior and parameter selection. *Evolutionary Computation*, 2012, 20(3):349-393

[20] Musa J D, Okumoto K. A logarithmic Poisson execution time model for software reliability measurement. In: Proceedings of the 7th International Conference on Software Engineering, NJ, USA, 1984. 230-238

Estimating parameters of software reliability models based on quantum-behaved particle swarm optimization

Chen Xiao, Jiang Jianhui

(School of Software Engineering, Tongji University, Shanghai 201804)

Abstract

The problem of precise estimation of the parameters of most software reliability models was studied, and a new approach for parameter estimating for software reliability models based on the quantum-behaved particle swarm optimization (QPSO) algorithm was proposed on the basis of the analysis of the existing methods for software reliability models' parameter estimation. To verify the performance of the proposed method, three software reliability models of G-O, M-O and Weibull were compared according to the classification of software reliabiliy models and seven data samples were adopted to conduct the simulation experiment, and the experimental results were compared with the particle swarm optimization (PSO) algorithm and the ant colony algorithm. The comparison results demonstrate that the proposed method has the higher precision, minor error and stronger adaptability.

Key words: quantum-behaved particle swarm optimization (QPSO), software reliability models, parameters estimating