多层缺陷关联效应对软件可靠性增长过程的影响

弋泽龙[1] 温玉梅[1] 林燕敏[1] 陈伟庭[2] 吕冠宇[3]

(深圳大学经济学院 广东 深圳 518060)[1] (深圳大学电子科学与技术学院 广东 深圳 518060)[2]

(深圳大学物理与能源学院 广东 深圳 518060)[3]

摘 要 软件系统中的缺陷通常以非常复杂的方式互相关联,并最终导致系统失效。基于非齐次泊松过程的软件可靠性增长模型,是一种描述软件随机失效行为和测量软件可靠性增长过程的常见工具。为此,考虑到有关联作用的多层缺陷,提出一个基于非齐次泊松过程的软件可靠性增长模型来研究软件系统的可靠性增长过程,并通过现实数据集对模型的性能进行评估。研究表明,新模型抓住了多层缺陷的关联效应,很好地拟合了缺陷数据集,且优于传统模型。此外,对于同时考虑了可靠性要求和测试成本的软件发行策略,研究发现,如果测试团队忽略缺陷不同层之间的关联效应,会使软件包发行到市场的最佳时间提前,从而相应的增加整体成本。

关键词 软件可靠性增长模型,非齐次泊松过程,多层缺陷,关联效应,软件发行策略

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.02.042

Impacts of Correlation Effects among Multi-layer Faults on Software Reliability Growth Processes

YI Ze-long[1] WEN Yu-mei[1] LIN Yan-min[1] CHEN Wei-ting[2] LV Guan-yu[3]

(College of Economics, Shenzhen University, Shenzhen, Guandong 518060, China)[1]

(College of Electronic Science and Technology, Shenzhen University, Shenzhen, Guandong 518060, China)[2]

(College of Physics and Energy, Shenzhen University, Shenzhen, Guandong 518060, China)[3]

Abstract Faults in the software systems, which eventually cause the system failures, are usually connected with each other in complicated ways. Software reliability growth models based on non-homogeneous Poisson processes are widely adopted tools when describing the stochastic failure behavior and measuring the reliability growth in software systems. Considering a group of correlated faults, a new model was built to examine the reliability of software systems and assess the model's performance from real-world data sets. Numerical studies show that the new model captures correlation effects among multi-layer faults, fits the failure data well and performs better than traditional models. The optimal software release policy, which considers both the reliability requirement and the software testing cost, was also formally studied. It is found that if the correlation effects among different layers of faults are ignored by the software testing team, the best time to release the software package to the market will be much earlier while the overall cost will be much higher.

Keywords Software reliability growth model, Non-homogeneous poisson processes, Multi-layer faults, Correlation effects, Software release policy

1 引言

软件可靠性是指一个软件在指定时间间隔内成功运行且不失效的概率,是测量软件水平的一个关键指标。事实上,为软件可靠性增长过程建立理论模型并对其发展趋势进行预测,对软件开发团队十分重要,因为这会影响到软件的发行时间,以及预测终端用户使用该软件时的出错频率。因此,人们构造了一系列软件可靠性增长模型来刻画软件随机失效行为,其中大部分基于非齐次泊松过程[1-8]。

值得注意的是,现有许多模型都假设软件系统中的缺陷是独立存在的,但这种假设在很多情况下并不适合[9-18]。例如,Goševa-Popstojanova 等[13]利用马尔可夫更新过程,提出

到稿日期:2016-11-09 返修日期:2017-03-21

弋泽龙 男,博士,讲师,主要研究方向为软件可靠性,E-mail:yizl@szu.edu.cn(通信作者);**温玉梅**(1996-),女,主要研究方向为软件可靠性;**林燕敏**(1996-),女,主要研究方向为软件可靠性;**陈伟庭**(1995-),男,主要研究方向为软件可靠性;**吕冠宇**(1995-),男,主要研究方向为软件可靠性。

了连续运行软件之间的相互依存关系的软件可靠性模型。之后,Dai 等人[14]又基于马尔可夫更新过程,建立了一个由多种故障类型组成的软件可靠性增长模型。类似地,Sahinoglu[15]提出了复合泊松过程软件可靠性模型,考虑了在同一时间发生多次软件失效的情况。Singh 等人[16]在两种故障类型存在的条件下提出了新的软件可靠性增长模型。Ho 等人[17]基于人工神经网络系统,抽象地将代码的组成比作神经网络的信息交换,建立了软件可靠性增长模型。Fiondella 等人[20]提出了一种可扩展的方法来分析基于组件的软件系统的可靠性增长过程,考虑了相关组件故障,并通过两个实验研究证实了该方法的有效性。

我们还注意到,此前研究工作还考查了软件最佳发行时间的问题。例如,Rafi 等人[21]研究了软件可靠性增长模型,并引入了广义威布尔测试工作函数来分析软件最优发行策略。Li 等人[22]研究了开源软件的版本更新问题,通过一个 Logistic 函数来刻画软件测试工作。Li 等人[23]利用 Apache 和 GNOME 的现实数据集进行可靠性分析,并探讨了开源软件的最佳版本更新策略。Pachauri 等人[24]在一个不完美的调试环境中,引用广义威布尔测试工作函数,用多属性效用理论对软件可靠性增长模型进行了研究,考查了软件最佳发行政策。Peng 等人[25]基于多属性效用理论的模型,同时考虑了软件测试费用和风险,以确定最佳的发行时间。这里的风险指参数在评估时的不确定性带来的不能达到所期望的可靠性的风险。关于发行一个软件版本的最佳时间,Huang 等人[26]考虑了缺陷的依赖性和调试时间的滞后性,得到了一系列理论成果。

本文主要针对以嵌套调用为主要结构的软件程序,研究每一层程序的出错规律,提出一种考虑了相互依存的网状多层缺陷的新模型。具体而言,第一层缺陷的检测过程是独立的,测试速率正比于残留在系统中第一层缺陷的期望值。但与第一层缺陷不同的是,第二层缺陷的检测过程取决于第一层,测试速率同时正比于第一层和第二层剩余缺陷的期望值。类似地,第三层缺陷的测试过程依赖于第二层,测试速率正比于第三层剩余缺陷的期望值,同时还正比于第一层和第二层剩余缺陷的期望值。该规律同样适用于软件系统其他所有层的缺陷。

在本文的研究中,将引入两个实际软件包 SoftPM V2.8 和 RVLIS 的数据集进行模型检验,用非线性最小二乘估计法进行参数估计。通过最小化真实数据和理论值之间的偏差的平方和来实现对参数的估计。估计结果表明,该模型能很好地拟合真实数据,比传统的不考虑多层缺陷关联影响的 Goel-Okumoto 模型(简称为 G-O 模型)更好。同时,本文着重分析了软件的最佳发行政策及其最低成本。我们发现,如果软件测试团队忽视多层缺陷关联效应,将会造成软件测试资源配置不合理。通过与基本模型(G-O 模型)进行比较,我们发现,适当推迟软件包发行时间可以降低总预期成本。

本文第 2 节阐述多层缺陷的关联效果,然后介绍新的软件可靠性增长模型;第 3 节首先提出了数据集用例和参数估计法,然后得到估计值并进行比较分析评估;第 4 节考查软件最佳发行策略;最后总结全文。

2 模型与结果

2.1 关联效应

本文研究主要针对基于嵌套调用的程序。这种程序的结构特点是:在主程序之下有子程序,子程序之下还有其子程序,通过多次调用来实现某些特定功能。为了方便描述,我们赋予这些程序“层”的概念,相对于主程序,将所有子程序分层。其中,主程序称为第一层程序,主程序的子程序称为第二层程序。特别地,在同一层被调用的子程序往往不止一个,可能存在“平行关系”,因此统称它们为第二层程序。以此类推,每一个子程序都相对应地可以归属于某一个层。因此,我们将存在于主程序的缺陷称为第一层缺陷,将存在于第 L 层程序的缺陷称为第 L 层缺陷。

在描述软件可靠性增长过程时,考虑多层缺陷的相互依赖性是十分重要的。具体而言,本文所提缺陷不是独立的,而是相互关联的。我们发现,在嵌套调用的程序中,一个代码错误很有可能会引起其他错误,层与层之间更是有直接或间接的联系。因此,本文提出有关联性的多层缺陷网状结构:第一层缺陷的检测过程是独立的;第二层缺陷的检测依赖于第一层;第三层缺陷的检测既依赖于第一层,又依赖于第二层;以此类推,一直到第 L 层。图 1 是一个包括六层缺陷的网状结构。

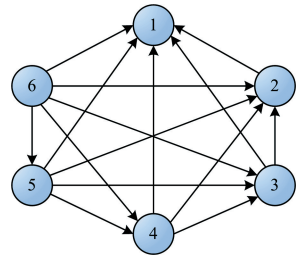


图 1 软件系统六层缺陷的关联效应图解

Fig. 1 Illustration for software system with networking effects among six layers of faults

本文并未考虑每一层具体调用的程序,也不考虑其内容是否相同,因为相同与否并不能改变其嵌套的结构,因此基于这种结构的关联猜想依然成立。子程序在不同层被调用,有可能在一定程度上改变其对每一层的依赖性,但不会“消灭”关联性的存在。我们只需要获得缺陷的总数量,通过数值运算得到每一层的缺陷数,即可检验我们的猜想。

2.2 基本假设

令$\{N(t),t\geqslant 0\}$为一个计数过程,表示时间$(0,t]$内的累积缺陷数,然后用均值函数 $m(t)$来表示缺陷累积数期望值,则有:

$$m(t)=E[N(t)]=\int_0^t \lambda(t')\mathrm{d}t'$$

其中，$E[\cdot]$表示期望，$\lambda(t')$表示在时间 t'内的失效强度函数。假设 $N(t)$是一个具有均值函数 $m(t)$的非齐次泊松过程，则有如下数学等式：

$$\Pr[N(t)=n]=\frac{(m(t))^n\times \mathrm{e}^{-m(t)}}{n!},n=0,1,2,3,\cdots$$

通过选取不同形式的均值函数可以得到不同的数学模型。

本节为了描述软件失效过程的随机行为，并获得均值函数的表达式，拟基于如下假设进行理论建模：

(1)软件的失效行为是由系统中残余的缺陷随机引起的。

(2)软件缺陷检测过程是一个非齐次泊松过程。

(3)缺陷被检测后立即被修复，修复时间忽略不计，且缺陷修复不引入新的缺陷。

(4)软件测试速率是一个常数。

(5)在软件中存在 L 层($L=1,2,3,\cdots$)缺陷，每层缺陷由不同的增长曲线来刻画。

(6)在单位时间$(t,t+\Delta t)$内，检测到的软件的第一层缺陷均值正比于恒定的测试速率，也正比于残留在该系统第一层的缺陷数量。

(7)在单位时间$(t,t+\Delta t)$内，第 l 层($l=2,3,\cdots,L$)检测到的缺陷数量既正比于恒定的测试速率，也正比于系统中第 l 层剩余的缺陷数量，还正比于第 j 层缺陷的剩余值与初始值之比($j=1,2,\cdots,l-1$)。

在上述 7 个假设中，前 4 个假设是现有大多数软件可靠性增长模型的基本元素，适用于多数模型，而其他 3 个假设则是为反映多层缺陷关联效应而特别提出的。

2.3　建模过程

基于上述假设，我们将提出每一层缺陷对应的均值函数。

根据假设(6)，软件系统中的第一层缺陷的测试速率正比于第一层剩余缺陷的数量和测试速率 r。因此，第一层缺陷的均值函数满足以下微分方程：

$$\frac{\mathrm{d}m_1(t)}{\mathrm{d}t}=r[a_1-m_1(t)]$$

其中，a_1 是系统中第一层缺陷的初始值，未知待求。$m_1(t)$ 表示在时间$(0,t]$内检测出的第一层缺陷数量的期望值。具体地说，假设 $m_1(t)$是关于时间 t，并以 $m_1(0)=0$ 为初始条件的有界非递减函数，通过求解上述微分方程，可以得到如下表达式：

$$m_1(t)=a_1(1-\mathrm{e}^{-rt})$$

根据假设(7)，第二层缺陷数量的期望值满足以下微分方程：

$$\frac{\mathrm{d}m_2(t)}{\mathrm{d}t}=r[a_2-m_2(t)]\frac{a_1-m_1(t)}{a_1}$$

其中，a_2 是系统中第二层缺陷的初始值，$m_2(t)$代表了在时间$(0,t]$内检测到第二层缺陷数量的期望值。同上，$m_2(t)$也是关于时间 t，且初始条件为 $m_2(0)=0$ 的有界非递减函数。求解上述微分方程，可得：

$$m_2(t)=a_2(1-\mathrm{e}^{\mathrm{e}^{-rt}-1})$$

按照上述分析，第三层缺陷的均值函数应满足以下微分方程：

$$\frac{\mathrm{d}m_3(t)}{\mathrm{d}t}=r[a_3-m_3(t)]\frac{a_1-m_1(t)}{a_1}\cdot\frac{a_2-m_2(t)}{a_2}$$

其中，a_3 为系统第三层的初始缺陷总数，$m_3(t)$指在时间$(0,t]$内检测到的第三层缺陷的期望值。同样，$m_3(0)=0$，求解上述微分方程可得：

$$m_3(t)=a_3(1-\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1})$$

同样，第四层缺陷的均值函数满足以下微分方程：

$$\frac{\mathrm{d}m_4(t)}{\mathrm{d}t}=r[a_4-m_4(t)]\frac{a_1-m_1(t)}{a_1}\cdot\frac{a_2-m_2(t)}{a_2}\cdot\frac{a_3-m_3(t)}{a_3}$$

其中，a_4 为系统第四层的初始缺陷总数，$m_4(t)$指在时间$(0,t]$内检测到的第四层缺陷的期望值。求解上述微分方程得到：

$$m_4(t)=a_4(1-\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}-1})$$

以此类推，可以推导出在时间$(0,t]$内检测到的第 l 层缺陷期望值的一般表达式，即推论 1。

推论 1　假设一个软件系统共有 L 层缺陷，$l=1,2,\cdots,L$，则：

$$m_l(t)=a_l(1-\underbrace{\mathrm{e}^{\cdots^{\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}-1}}}_{l\text{次}\mathrm{e}^{\{\cdot\}}})$$

用数学归纳法证明推论 1 的具体过程如下。

证明：对一个有 L 层缺陷的软件系统，假设第 j 层缺陷满足均值函数：

$$m_j(t)=a_j(1-\underbrace{\mathrm{e}^{\cdots^{\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}-1}}}_{j\text{次}\mathrm{e}^{\{\cdot\}}})$$

其中，$j=1,2,3,\cdots,l-1,l=2,3,\cdots,L$。然后，由缺陷层之间的关联效应可以得到：

$$\begin{aligned}&\frac{\mathrm{d}m_l(t)}{\mathrm{d}t}\\&=r[a_l-m_l(t)]\frac{a_1-m_1(t)}{a_1}\cdot\frac{a_2-m_2(t)}{a_2}\cdot\cdots\cdot\frac{a_{l-1}-m_{l-1}(t)}{a_{l-1}}\\&=r[a_l-m_l(t)]\mathrm{e}^{-rt}\mathrm{e}^{\mathrm{e}^{-rt}-1}\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}\cdots\underbrace{\mathrm{e}^{\cdots^{\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}-1}}}_{(l-1)\text{次}\mathrm{e}^{\{\cdot\}}})\end{aligned}$$

等价于

$$\frac{\mathrm{d}m_l(t)}{a_l-m_l(t)}=r\mathrm{e}^{-rt}\mathrm{e}^{\mathrm{e}^{-rt}-1}\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}\cdots\underbrace{\mathrm{e}^{\cdots^{\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}-1}}}_{(l-1)\text{次}\mathrm{e}^{\{\cdot\}}}\mathrm{d}t$$

对方程两边取积分得：

$$\begin{aligned}&\int_0^t\frac{\mathrm{d}m_l(t)}{a_l-m_l(t)}\\&=\int_0^t r\mathrm{e}^{-rt}\mathrm{e}^{\mathrm{e}^{-rt}-1}\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}\cdots\underbrace{\mathrm{e}^{\cdots^{\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}-1}}}_{(l-1)\text{次}\mathrm{e}^{\{\cdot\}}}\mathrm{d}t\\&=\int_0^t\mathrm{e}^{-rt}\mathrm{e}^{\mathrm{e}^{-rt}-1}\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}\cdots\underbrace{\mathrm{e}^{\cdots^{\mathrm{e}^{\mathrm{e}^{\mathrm{e}^{-rt}-1}-1}-1}}}_{(l-1)\text{次}\mathrm{e}^{\{\cdot\}}}\mathrm{d}rt\end{aligned}$$

$$=\int_0^t -e^{e^{e^{-rt}-1}e^{e^{e^{-rt}-1}-1}\cdots\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}}de^{-rt}$$

$$=\int_0^t -e^{e^{e^{-rt}-1}\cdots\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}}de^{e^{-rt}-1}$$

$$\cdots$$

$$=\int_0^t -d\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}$$

解得：

$$-\ln(a_l-m_l(t))\Big|_0^t=-\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}\Big|_0^t$$

即

$$-\ln\frac{a_l-m_l(t)}{a_l}=1-\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}$$

等价于

$$m_l(t)=a_l\left(1-\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}\right)$$

当 $j=1$ 时，推论 1 的结论也成立，即

$$m_1(t)=a_1(1-e^{-rt})$$

以此类推，当 $l=1,2,3,\cdots,L$ 时，结论均成立，因此推论得证。

然后，定义总均值函数 $m(t)$，它包括每一层缺陷的均值函数。假设用 $a=a_1+a_2+a_3+\cdots+a_L$ 表示预期最终可检测到的缺陷总数，则有：

$$m(t)=m_1(t)+m_2(t)+m_3(t)+\cdots+m_L(t)$$

$$=\sum_{l=1}^{L}a_l\left(1-\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}\right)$$

$$=a-\sum_{l=1}^{L}a_l\left(\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}\right) \tag{1}$$

因此，$m(t)$ 是关于 t 的有界非递减函数，其中 $m(0)=0$，$m(\infty)=a$。

当 $L=1$ 时，可以得到传统的 G-O 模型：

$$m(t)=a(1-e^{-rt}) \tag{2}$$

这也是本文进行比较分析的基础模型。在这种情况下，系统中的缺陷是独立的，均值函数 $m(t)$ 是关于 t 的凹函数，该模型在传统的软件可靠性分析中被广泛使用。

基于以上均值函数，所提模型的失效强度函数可以表示为：

$$\lambda(t)=\frac{dm(t)}{dt}$$

$$=\sum_{l=1}^{L}ra_l\cdot\underbrace{e^{\cdots^{e^{e^{e^{-rt}-1}-1}-1}}}_{l次e^{\{\cdot\}}}\cdot\underbrace{e^{\cdots^{e^{e^{-rt}-1}-1}}}_{(l-1)次e^{\{\cdot\}}}\cdot\cdots\cdot e^{-rt}$$

以上就是考虑关联效应的模型的推导过程，为了后文的比较分析，本文还引入另外两个常见的经典模型。

Musa-Okumoto(M-O)模型[18]：

$$m(t)=a_1\ln(1+rt)$$

二阶指数(Exp2)模型[19]：

$$m(t)=a_1(1-e^{-rt^2})$$

3 估算与分析

3.1 数据集

为了比较新模型与传统模型的准确性和预测能力，并对新模型进行评估，引入 SoftPM V2.8 项目[30]和 RVLIS[31]的数据集。SoftPM V2.8 是一个由中国科学院软件研究所开发的关于软件过程管理的软件。表 1 给出了其缺陷累积数据，包括软件的发布时间和检测到的缺陷总数，测试团队从第 1 周开始测试，到第 32 周结束，共发现 1384 个缺陷。RVLIS 是一个用于监测反应器容器内的水位的检测系统，表 2 给出了其缺陷累积数据，测试团队从第 1 周到第 25 周共发现 230 个缺陷。

表 1 SoftPM V2.8 发布后的周数和对应检测到的缺陷数目

Table 1 Weeks after release and the number of errors detected for SoftPM V2.8

发布周数	缺陷数量	发布周数	缺陷数量
1	80	17	1055
2	213	18	1070
3	301	19	1101
4	360	20	1112
5	389	21	1150
6	447	22	1200
7	509	23	1217
8	643	24	1243
9	670	25	1280
10	795	26	1289
11	795	27	1310
12	795	28	1340
13	823	29	1350
14	886	30	1364
15	899	31	1378
16	1013	32	1384

表 2 RVLIS 发布后的周数和对应检测到的缺陷数目

Table 2 Weeks after release and the number of errors detected for RVLIS

发布周数	缺陷数量	发布周数	缺陷数量
1	44	14	181
2	75	15	197
3	75	16	205
4	75	17	214
5	75	18	215
6	75	19	225
7	75	20	227
8	100	21	228
9	124	22	230
10	130	23	230
11	130	24	230
12	159	25	230
13	175		

3.2 估算技术

为了评估该模型，需要比较软件实际出错数据与理论模型。非线性最小二乘估计法是一种被广泛使用的参数估算技术，尤其适合处理中小型数据集。本文将运用这种技术来估计所提可靠性增长模型中含有的参数。其原理为：最小化理论值与实际观测值之差的平方和。也就是说，该问题等价于最小化如下目标函数：

$$S=\sum_{j=1}^{J}(m(t_j)-m_j)^2 \tag{3}$$

其中，m_j 表示第 j 次观测得到的缺陷真实值，J 表示观测的总数，t_j 表示发生的第 j 次观测的时间，$m(t_j)$ 是在式(1)中介绍的理论值。通过最小二乘法求得当 S 取得最小值时，每个函数中未知的参数 $a_1,a_2,a_3,\cdots,a_L$，以及 r 的取值。

对于理论值和真实值之间存在的误差，通常引入一个区间范围来使估计值落于其中，即我们希望在误差允许的范围内得到较为理想的估值。在缺陷类别有限的泊松模型中，均值函数的 $100\times(1-\alpha)\%$ 的置信区间可以近似表示为[31-32]：

$$\text{上限}=m(t)+k_{1-\alpha/2}\sqrt{m(t)}$$

$$\text{下限}=m(t)-k_{1-\alpha/2}\sqrt{m(t)}$$

其中，$k_{1-\alpha/2}$ 是标准正态分布的 $100\times(1-\alpha/2)$ 百分位数。因此，本文假设 80% 的置信区间的上限和下限是较为合理的范围，可以用式(4)表示：

$$m(t)\pm k_{0.90}\sqrt{m(t)}=\sum_{l=1}^{L}a_l\Big(1-\underbrace{e^{\cdots^{e^{e^{-rt}-1}-1}-1}}_{(l-1)\text{次}e^{\{\cdot\}}}\Big)\pm k_{0.09}\sqrt{\sum_{l=1}^{L}a_l\Big(1-\underbrace{e^{\cdots^{e^{e^{-rt}-1}-1}-1}}_{(l-1)\text{次}e^{\{\cdot\}}}\Big)} \tag{4}$$

3.3　对比分析的标准

根据模型参数的估算值即可得到模型表达式，并可以比较分析各模型。本节将介绍几个常用的评估分析指标，用于比较理论模型与实际数据的拟合程度。

(1)R^2：用 R^2 的值来衡量模型拟合优度，即理论曲线与现实数据的吻合程度[33]。定义如下：

$$R^2=1-\frac{\sum_{j=1}^{J}(m(t_j)-m_j)^2}{\sum_{j=1}^{J}(m_j-\sum_{q=1}^{J}\frac{m_q}{J})^2} \tag{5}$$

R^2 的取值范围为 0～1，R^2 值越大说明理论模型与实际数据越吻合，而 R^2 值较小则表明该模型与数据吻合得不好。在这个测试用例中，如果 R^2 高于 0.95，则认为拟合程度较高。

(2)Theil 统计(TS)：TS 表示在选定的时间周期内，模型相对于实际缺陷数据的平均百分比偏差[34]，定义如下：

$$TS=\sqrt{\frac{\sum_{j=1}^{J}(m(t_j)-m_j)^2}{\sum_{j=1}^{J}y_j^2}} \tag{6}$$

本文把 TS 小于 10% 的结果视为较好的结果，TS 取值越小，模型越好。

(3)置信区间的覆盖(CCI)：CCI 表示落在 80% 的置信区间内的样本的百分比[30]，可以表示为：

$$CCI=\frac{|\{m(t_j)-k_{0.90}\sqrt{m(t_j)}\leqslant m_j\leqslant m(t_j)+k_{0.90}\sqrt{m(t_j)}\}|}{J}\times 100\%,j=1,2,\cdots,J \tag{7}$$

其中，$|\{\cdot\}|$ 表示满足条件的元素数量。CCI 值越高，表示均值函数与缺陷数据集越拟合。

(4)均方误差(MSE)：MSE 用于计算预测值和真实值之间差的平方的平均值[17]。定义如下：

$$MSE=\frac{1}{J}\sum_{j=1}^{J}(m(t_j)-m_j)^2 \tag{8}$$

通常 MSE 的值越小，模型与实际数据越吻合，预测能力越强。

3.4　评估结果

参数估计有助于评估理论模型，确定在测试过程的缺陷数量如何随时间变化，以及预测终端用户最终遇到的缺陷数量。具体地说，通过最小化式(3)中的目标函数，可以分别得到当 $L=1,2,3,4$ 时相应参数的估计值，本文将其我们还与 M-O 模型、Exp2 模型进行对比，结果如表 3、表 4 所列。从 SoftPM V2.8 数值估算结果来看：

当 $L=1$ 时，估计结果是 $\hat{a}_1=1667.3197$，$\hat{r}=0.0557$。

当 $L=2$ 时，测试速率 $\hat{r}=0.0500$。该软件中第一层的缺陷率约为 70%，第二层约为 30%。虽然第二层缺陷比第一层的少得多，但也说明了缺陷不是独立存在的。

当 $L=3$ 时，$\hat{a}_1=1231.9972$，$\hat{a}_2=531.5164$，$\hat{a}_3=436.7121$，$\hat{r}=0.0450$。因此，该软件包含的 3 层缺陷各占 56%，24% 和 20%。

当 $L=4$ 时，4 层缺陷的比例分别为 54%，25%，18%，3%。

同理，运用 RVLIS 数据的估算情况为：

当 $L=1$ 时，估计结果是 $\hat{a}_1=325.4899$，$\hat{r}=0.0559$。

当 $L=2$ 时，测试速率 $\hat{r}=0.0557$。软件中第一层的缺陷率约为 99%，第二层约为 1%。虽然第二层缺陷比第一层的少得多，但也说明了缺陷不是独立存在的。

当 $L=3$ 时，$\hat{a}_1=322.3452$，$\hat{a}_2=1.5371$，$\hat{a}_3=13.4723$，$\hat{r}=0.0545$。因此，该软件包含的 3 层缺陷的比例约为 95%，0.4% 和 4.6%。

当 $L=4$ 时，估计结果显示测试速率为 $\hat{r}=0.0443$。此时，4 层缺陷的比例为 87.2%，0.2%，0.3%，12.3%。

表 3　模型参数估算值与模型拟合结果(SoftPM V2.8)

Table 3　Estimated values of model parameters and model fitting results for SoftPM V2.8

	$\hat{a}_1$	$\hat{a}_2$	$\hat{a}_3$	$\hat{a}_4$	$\hat{r}$	R^2	TS	CCI/%
$L=1$(G-O)	1667.3197	—	—	—	0.0405	0.9951	0.0267	75.000
$L=2$	1363.3073	567.5292	—	—	0.0500	0.9952	0.0264	71.875
$L=3$	1231.9972	531.5164	436.7121	—	0.0450	0.9952	0.0264	68.750
$L=4$	1215.1209	558.8687	389.9431	76.6822	0.0443	0.9952	0.0264	71.875
M-O	877.1547	—	—	—	0.1279	0.9941	0.0292	78.125
Exp2	1266.1006	—	—	—	0.0079	0.9196	0.1080	75.000

表 4 模型参数估算值与模型拟合结果(RVLIS)

Table 4 Estimated values of model parameters and model fitting results for RVLIS

	$\hat{a}_1$	$\hat{a}_2$	$\hat{a}_3$	$\hat{a}_4$	$\hat{r}$	R^2	TS	$CCI/\%$
$L=1$(G-O)	325.4899	—	—	—	0.0559	0.9403	0.0936	72.000
$L=2$	324.3290	2.0234	—	—	0.0557	0.9404	0.0936	72.000
$L=3$	322.3452	1.5371	13.4723	—	0.0545	0.9404	0.0936	72.000
$L=4$	318.2804	0.8687	0.9431	45.0827	0.0443	0.9405	0.0935	72.000
M-O	189.8020	—	—	—	0.1080	0.9389	0.0948	72.000
Exp2	288.5801	—	—	—	0.0092	0.8753	0.1354	72.000

据此,在这两组数据下的实验都可一定程度上说明如果忽略或仅部分考虑缺陷的不同层之间的关联效应,在分析软件系统关联性缺失时将出现偏差,从而证明了本文模型的合理性。

为了进一步说明以上结果的合理性,我们分析 3 项评估标准的值。首先,在两组数据对比之下,与 G-O 模型($L=1$)相比,R^2 都非常接近 1;同时,CCI 普遍很高,均在 70%以上,即大多数观测数据点落在 80%的置信区间;此外,每一层的 TS 值都远远小于 10%,说明提出的模型与现实缺陷数据是非常吻合的。虽然我们无法获得每一层的缺陷数,也不能直接证明 $a_1,a_2,a_3,\cdots,a_L$ 的估计值等于真实值,但参数之间存在的微小差异足以说明本文模型要优于 G-O 模型,进而较好地证明了模型的合理性,即设置 $L>1$ 的必要性。而在 SoftPM V2.8 数据下,针对 M-O 模型和 Exp2 模型,虽然两者对应的 CCI 略大,但考虑了关联效应的模型对应的 R^2 值均更大,TS 取值更小。同样,在 RVLIS 数据下,虽然 CCI 取值相同,但是考虑了关联效应的模型对应的 R^2 值均更大,TS 值均更小。因此,考虑关联效应是十分必要的。

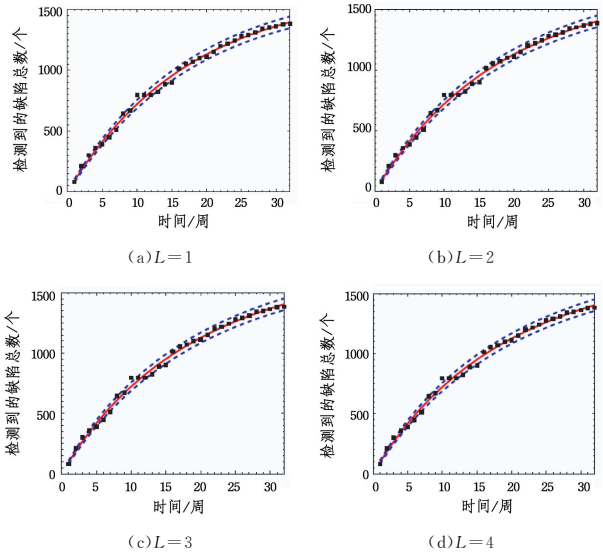


图 2 预测值和观测值

Fig. 2 Predicted values and observation values

基于以上结果,对 SoftPM V2.8 第 1,2,3,4 层缺陷的理论值进行讨论。图 2 给出缺陷理论值与观测值的对比,可以看到,理论值的上界和下限大部分落在 80%的置信区间内。图 2(a)刻画的是 G-O 模型的拟合曲线。图 2 中实线表示预测值;虚线表示上限和下限为 80%的置信区间;点表示观测值。

本文基于所估计的参数值进一步检验失效强度函数,图 3 给出了 $\lambda(t)$的曲线。实际上考虑更大的 L 时,失效强度函数都随时间递减,且不受 L 的影响,这也证明了本文模型属于凹状类的软件可靠性增长模型[37]。

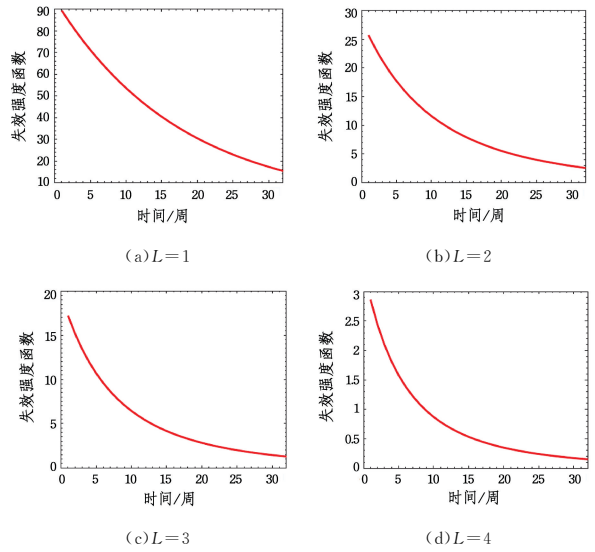


图 3 随时间变化的失效强度函数

Fig. 3 Failure intensity function with time variation

3.5 预测分析

基于 3.4 节的分析可以得到较为理想的评估结果,而对于模型的预测能力,本文将重点考查 MSE 数值的大小。将 SoftPM V2.8 的数据平均分为两部分:1)1—16 周的数据用于重新估算模型中的参数;2)17—32 周的数据用于检验模型的预测能力。实验结果如表 5 所列。

表 5 模型参数和 MSE 的估算值(SoftPM V2.8)

Table 5 Estimated values for parameters in the proposed model and MSE(SoftPM V2.8)

	$\hat{a}_1$	$\hat{a}_2$	$\hat{a}_3$	$\hat{a}_4$	$\hat{r}$	MSE
$L=1$(G-O)	1465.1918	—	—	—	0.0677	466.5748
$L=2$	1267.3046	428.8579	—	—	0.0405	242.6497
$L=3$	1241.9165	398.2277	169.8472	—	0.0405	168.7652
$L=4$	1277.9901	381.0683	104.7805	64.6317	0.0563	165.9641
M-O	957.5182	—	—	—	0.1112	291.6959
Exp2	892.1864	—	—	—	0.0211	1511.7540

观察表 5 中的数据可以明显看到，从 $L=1$ 到 $L=4$，*MSE* 的值呈现逐渐减小的趋势，*MSE* 越小说明预测能力越强；同时注意到，当 $L=1$ 时，G-O 模型的 *MSE* 值最大，说明其预测能力远不及考虑了关联效应的新模型。除此之外，我们还可以看到，随着考虑的层数的增加，*MSE* 的值会继续减小，拟合程度更好，因此我们有必要认真考虑模型的合适层数。在与 M-O 模型和 Exp2 模型的对比中也可以观察到，考虑了关联效应的模型对应的 *MSE* 值更小，预测能力更强。用 RVLIS 数据进行检验也可获得相同的结论。

4　软件发行策略

本节具体研究软件的最佳发行时间问题，并就多层缺陷关联效应如何影响软件最优发布时间和总成本进行深入分析。本节采用 SoftPM V2.8 数据进行分析。

已知，软件出错过程是一个满足均值函数为 $m(T)$ 的非齐次泊松过程，则可以建立如下软件总成本函数[37]：

$$C(T)=c_1 m(T)+c_2(m(\infty)-m(T))+c_3 T$$

其中，T 是由测试团队确定的该软件发布到市场的时间，是决策变量；c_1 是在软件测试阶段每去除单位故障的预期成本；c_2 是在软件使用阶段每去除单位故障的预期成本；c_3 是每单位检测时间的预期成本。一般来说，$c_2>c_1$，则最佳发行时间应满足 $m'(T^*)=c_3/(c_2-c_1)$[27]。

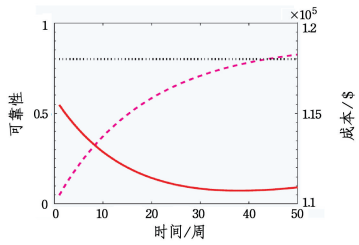
通常，在实际操作中，当软件系统剩余缺陷少于某预期目标时，应该停止软件测试并将软件向市场发布。这样，软件可靠性的一个简单标准可以定义为[23,27]：

$$R(T)=\frac{m(t)}{a} \tag{9}$$

在大多数情况下，项目经理需要平衡软件总成本和可靠性。也就是说，在决定发布时间时，应提出一个关于软件可靠性的要求，可靠性应至少高达提前设定的某个水平。设软件可靠性的目标水平为 β，则可靠性应该满足：

$$R(T)\geqslant\beta$$

图 4 表示当 $L=3$ 时，软件可靠性和总成本随时间的变化情况。先设定 c_1，c_2，c_3 的值，并假设软件可靠性必须在 0.75 以上，即 $\beta=0.75$。只有 $T\geqslant T_r=45.99$ 时，才满足可靠性要求。我们发现，当成本在 $T_c=54.37$ 时总成本可取得最小值。



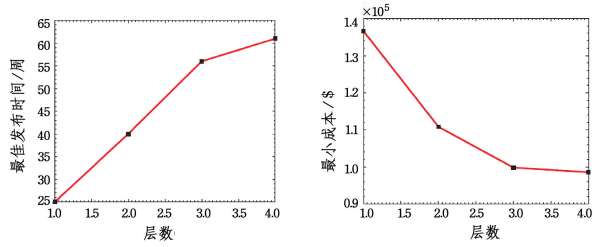
注：实线表示成本；虚线表示可靠性；点线表示可靠性目标水平。假设 $c_1=\$80$，$c_2=\$85$，$c_3=\$60$

图 4　可靠性和成本关于时间的函数

Fig. 4　Functions of reliability and cost vs. time

当同时考虑软件可靠性和总成本时，最佳发行时间 $T^*=54.37$。

图 5 表示对应不同的层数 $L=1,2,3,4$，最佳发行时间分别是 25，40，56 和 61，相对应的成本为 $\$1.366\times10^5$，$\$1.108\times10^5$，$\$0.998\times10^5$ 和 $\$0.9855\times10^5$。



注：假设 $c_1=\$80$，$c_2=\$85$，$c_3=\$60$，$\beta=0.75$

图 5　考虑层数的最佳发行时间函数和总成本函数

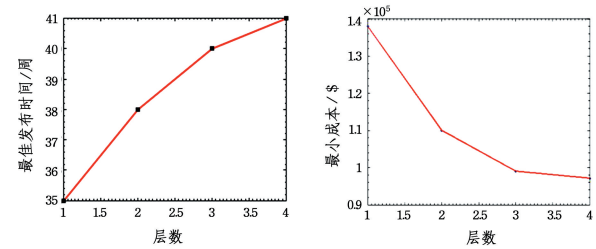
Fig. 5　The optimal release time function and total cost function considering the number of layers

可以发现，如果忽略或仅部分考虑软件多层缺陷关联效应，软件最佳发行时间和总成本的估值都会产生明显偏差。特别地，在本文模型中，软件最佳发行时间随着缺陷层数的增加而推迟。虽然测试团队会因此在测试中花费更多的时间，但总体成本会相对降低。这是因为，软件测试投入的越多，越能有效地提高软件系统的可靠性，从而降低其在使用阶段的成本。

接下来将采用另一种更常用的方法[6]来进一步讨论软件系统的可靠性：

$$R(\delta|t)=\mathrm{e}^{-(m(t+\delta)-m(t))},\delta>0 \tag{10}$$

令 $\delta=0.02$，其他条件同上，可求得不同层数对应的最佳发行时间分别为 35，38，40，41，最低成本分别为 $\$1.37\times10^5$，$\$1.107\times10^5$，$\$0.9953\times10^5$ 和 $\$0.9809\times10^5$，对应的趋势如图 6 所示。



注：假设 $c_1=\$80$，$c_2=\$85$，$c_3=\$60$，$\beta=0.75$

图 6　考虑层数的最佳发行时间函数和总成本函数

Fig. 6　The optimal release time function and total cost function considering the number of layers

综上所述，两种可靠性表示方法均可说明，在层数有限的程序里，考虑的层数越多，可靠性越高，软件的最佳发行时间越迟；而总成本则是呈现先降低后增加的变化。因此测试时间不宜太短，也不宜过长。除此，如果测试时间过长，还可能错过最佳市场时机，造成更大的利润损失。

结束语　本文基于多层缺陷层与层之间的关联效应，提出了一个新的软件可靠性增长模型。为了验证该模型，本文运用了软件 SoftPM V2.8 和 RVLIS 的缺陷数据集对模型进行检验。数值分析表明，该模型能很好地拟合现实数据集。本文还研究了软件发行策略，相比于不考虑多层缺陷关联效

应的基本模型，所提模型建议延迟软件发行时间，同时可以降低软件开发成本。从该方面来看，尽管所提模型似乎比传统的模型更复杂，但它包含了更多有关软件失效行为的信息，本文建模框架能为软件开发过程起到一定的理论指导作用。

本文研究还存在一定的不足，例如，提出的关联效应只限于从第一层到第 L 层的单方向，没有考虑层与层之间关联的逆向性；同时，与传统 G-O 模型一样，本文将测试速率 r 看作固定不变的常数，未考虑测试速率 r 会受到缺陷层数的影响而改变的问题，因此测试速率 r 还有待修正。

参 考 文 献

[1] GOEL A L, OKUMOTO K. Time-dependent error-detection rate model for software reliability and other performance measures[J]. IEEE Transactions on Reliability, 1979, 3: 206-211.

[2] HUANG C Y, LYU M R, KUO S Y. A unified scheme of some non-homogenous poisson process models for software reliability estimation[J]. IEEE Transactions on Software Engineering, 2003, 29(3): 261-269.

[3] KAPUR P K, GUPTA A, YADAVALLI S. Software reliability growth modeling using power function of testing time[J]. Int J Oper Quant Mgt, 2006, 12(2): 127.

[4] HUANG C Y, KUO S Y, LYU M R. An assessment of testing-effort dependent software reliability growth models[J]. IEEE Transactions on Reliability, 2007, 56(2): 198-211.

[5] COSTA E O, POZO A T R, VERGILIO S R. A genetic programming approach for software reliability modeling[J]. IEEE Transactions on Reliability, 2010, 59(1): 222-230.

[6] HSU C J, HUANG C Y. Optimal weighted combinational models for software reliability estimation and analysis[J]. IEEE Transactions on Reliability, 2014, 63(3): 731-749.

[7] LI Q, LI H, LU M. Incorporating S-shaped testing-effort functions into NHPP software reliability model with imperfect debugging[J]. Journal of Systems Engineering and Electronics, 2015, 26(1): 190-207.

[8] OKAMURA H, DOHI T, OSAKI S. Software reliability growth models with normal failure time distributions[J]. Reliability Engineering and System Safety, 2013, 16(2013): 135-141.

[9] WU Y P, HU Q P, XIE M, et al. Modeling and analysis of software fault detection and correction process by considering time dependency[J]. IEEE Transactions on Reliability, 2007, 56(4): 629-642.

[10] PIEVATOLO A, RUGGERI F, SOYER R. A Bayesian hidden Markov model for imperfect debugging[J]. Reliability Engineering & System Sofety, 2012, 103(103): 11-21.

[11] AKTEKIN T, CAGLAR T. Imperfect debugging in software reliability: A Bayesian approach[J]. European Journal of Operational Research, 2013, 227(1): 112-121.

[12] LI X, YIN Y, FIONDELLA L, et al. Software reliability analysis considering correlated component failures with coupling measurement framework[J]. System Engineering and Electronics, 2015, 26(5): 1114-1126.

[13] GOŠEVA-POPSTOJANOVA K, TRIVEDI K. Failure correlation in software reliability models[C]// 10th International Symposium Software Reliability Engineering. IEEE, 1999: 232-241.

[14] DAI Y S, XIE M, POH K L. Modeling and analysis of correlated software failures of multiple types[J]. IEEE Transactions on Reliability, 2005, 54(1): 100-106.

[15] SAHINOGLU M. Compound-Poisson software reliability model [J]. IEEE Transactions on Software Engineering, 1982, 18(7): 624-630.

[16] SINGH V B, YADAV K, KAPUR R, et al. Considering the fault dependency concept with debugging time lag in software reliability growth modeling using a power function of testing time[J]. International Journal of Automation and Computing, 2007, 4(4): 359-368.

[17] HO S L, XIE M, GOH T N. A study of the connectionist models for software reliability prediction[J]. Computers & Mathematics with Applications, 2003, 46(7): 1037-1045.

[18] MUSA J D, OKUMOTO K. A logarithmic Poisson execution time model for software reliability measurement[C]// Proceedings of the 7th International Conference on Software Engineering. IEEE Press, 1984: 230-238.

[19] YAMADA S, OHTERA H, NARIHISA H. Software reliability growth models with testing-effort[J]. IEEE Transactions on Reliability, 1986, 35(1): 19-23.

[20] FIONDELLA L, RAJASEKARAN S, GOKHALE S S. Efficient software reliability analysis with correlated component failures [J]. IEEE Transactions on Reliability, 2013, 62(1): 244-255.

[21] RAFI S M, RAO K N, AKTHAR . Incorporating generalized modified Weibull TEF in to software reliability growth model and analysis of optimal release policy[J]. Computer & Information Science, 2010, 3(2): 145.

[22] LI X, XIE M, NG S H. Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points[J]. Applied Mathematical Modelling, 2010, 34(11): 3560-3570.

[23] LI X, LI Y F, XIE M, et al. Reliability analysis and optimal version-updating for open source software[J]. Information & Software Technology, 2011, 53(9): 929-936.

[24] PACHAURI B, KUMAR A, DHAR J. Software reliability growth modeling with dynamic faults and release time optimization using GA and MAUT[J]. Applied Mathmatics & Computation, 2014, 242(2): 500-509.

[25] PENG R, LI Y F, ZHANG J G, et al. A risk-reduction approach for optimal software release time determination with the delay incurred cost [J]. International Journal of Systems Science, 2015, 46(9): 1628-1637.

[26] TANG A L, HUANG Q Y. Researches on GO Software Reliable Model[J]. Technology Information, 2009(31): 14-15. (in Chinese)
唐爱龙，黄秋勇. Goel-Okumoto 软件可靠性模型的研究[J]. 科技信息，2009(31): 14-15.

[27] LYU M R . Handbook of software reliability engineering[M]. IEEE Computer Society Press, 1996.

（下转第 275 页）

proach[J]. Computers & Operations Research, 2009, 36(6): 2082-2096.

[12] XIA W J, WU Z M. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J]. Computers & Industrial Engineering, 2005, 48(2): 409-425.

[13] STORN R, PRICE K. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[14] ZHOU X G, ZHANG G J, HAO X H, et al. A differential evolution algorithm based on local Lipschitz lower bound to estimate support surface [J]. Chinese Journal of Computers, 2016, 39(12): 2631-2651. (in Chinese)
周晓根，张贵军，郝小虎，等. 一种基于局部 Lipschitz 下界估计支撑面的差分进化算法[J]. 计算机学报，2016，39(12)：2631-2651.

[15] ZHOU X G, ZHANG G J, HAO X H. Differential evolution algorithm based on local abstract convex region[J]. Acta automatica Sinica, 2015, 41(7): 1315-1327. (in Chinese)
周晓根，张贵军，郝小虎. 局部抽象凸区域剖分差分进化算法[J]. 自动化学报，2015，41(7)：1315-1327.

[16] ZHOU X G, ZHANG G J, HAO X H, et al. Enhanced differential evolution using local Lipschitz underestimate strategy for computationally expensive optimization problems [J]. Applied Soft Computing, 2016, 48(11): 169-181.

[17] LIU X P, XU B Z, PENG J, et al. Job shop scheduling model and its solution for concurrent work processes[J]. Journal of Computer-aided Design & Computer Graphics, 2012, 24(1): 120-127. (in Chinese)
刘晓平，徐本柱，彭军，等. 工件工序可并行的作业车间调度模型与求解[J]. 计算机辅助设计与图形学学报，2012，24(1)：120-127.

[18] CUI J S, LI T K, ZHANG W X. Hybrid flow shop scheduling model and its genetic algorithm[J]. Journal of University of Science and Technology Beijing, 2005, 27(5): 623-626. (in Chinese)
崔建双，李铁克，张文新. 混合流水车间调度模型及其遗传算法[J]. 北京科技大学学报，2005，27(5)：623-626.

[19] ZHOU X. Design and implementation of workshop scheduling system for distributed CNC system[D]. Shenyang: University of Chinese Academy of Sciences, 2014. (in Chinese)
周鑫. 分布式数控系统车间排产系统的设计与实现[D]. 沈阳：中国科学院大学，2014.

[20] ZHOU X G, ZHANG G J, HAO X H, et al. A novel differential evolution algorithm using local abstract convex underestimate strategy for global optimization[J]. Computers & Operation Research, 2016, 75(11): 132-149.

[21] QIN A K, HUANG V L, SUGANTHAN P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(2): 398-417.

[22] WANG Y, CAI Z, ZHANG Q. Differential evolution with composite trial vector generation strategies and control parameters [J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 55-66.

(上接第 248 页)

[28] GOEL A L, OKUMOTO K. Time-dependent error-detection rate model for software reliability and other performance measures[J]. IEEE Transactions on Reliability, 1979, 3: 206-211.

[29] ZHANG R H, JIANG N, GOU L, et al. Software reliability growth model considering defect correlation[J]. Computer Engineering, 2008, 34(8): 44-46. (in Chinese)
张荣辉，姜楠，勾朗，等. 一种考虑缺陷关联的软件可靠性增长模型[J]. 计算机工程，2008，34(8)：44-46.

[30] MUSA J D, IANNINO A, OKUMOTO K. Software reliability, measurement, prediction and application [M]. McGraw Hill, 1987.

[31] HUANG C Y, LIN C T, KUO S Y, et al. Software reliability growth models incorporating fault dependency with various debugging time lags[C]// Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004 (COMPSAC 2004). IEEE, 2004: 186-191.

[32] LIN C T. Enhancing the accuracy of software reliability prediction through quantifying the effect of test phase transitions[J]. Applied Mathematical Computation, 2012, 219(5): 2478-2492.

[33] PILLAI K, NAIR V S. A model for software development effort and cost estimation[J]. IEEE Transactions on Software Engineering, 1997, 23(8): 485-497.

[34] CHIU K C, HUANG Y S, LEE T Z. A study of software reliability growth from the perspective of learning effects[J]. Reliability Engineering & System Safety, 2008, 93(10): 1410-1421.

[35] WOOD A. Predicting software reliability[J]. Computer, 1996, 29(11): 69-77.

[36] HUANG C Y, KUO S Y. Analysis of incorporating logistic testing-effort function into software reliability[J]. IEEE Transactions on Reliability, 2002, 51(3): 261-270.

[37] XIE M, YANG B. A study of the effect of imperfect debugging on software development cost[J]. IEEE Transactions on Software Engineering, 2003, 29(5): 471-473.