软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(1):14−25 [doi: 10.13328/j.cnki.jos.004672]

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

基于缺陷关联度的 Markov 模型软件优化测试策略*

包晓安[1], 谢晓鸣[1], 张 娜[1], 曹建文[2], 桂 宁[1,3]

[1](浙江理工大学 信息电子学院,浙江 杭州 310018)

[2](中国科学院 软件研究所 并行软件实验室,北京 100190)

[3](Distrinet Laboratory, University of Leuven, 3001, Belgium)

通讯作者: 桂宁, E-mail: guining@zstu.edu.cn

摘 要: 软件测试过程通常期望以最小的成本检测尽可能多的缺陷.为了降低建模复杂度,多数文献通常假设缺陷之间相互独立.但在实际测试中,缺陷之间往往存在关联,并且每个缺陷引发软件失效的严重程度也不相同.充分利用缺陷之间的关联信息,有助于增加相关缺陷的可检测率,提高软件测试效率.因此,提出一种新的思路:利用软件缺陷之间的关联构造缺陷相关系数,引入回扣机制,量化不同严重等级的缺陷所被检测到的价值,综合考虑缺陷相关系数、检测率、回扣三者的权值,以构造基于缺陷关联的最优测试策略.同时,提出复合的优化算法来构造相应的最小生成树,将测试剖面转换成带权的路径问题,以有效地寻找具有最大权值的最优测试路径.另外,改进了已有的剔除策略,以更有效地删除关联缺陷.通过实验仿真,并与其他测试策略相比较,证明了该方法的有效性.

关键词: 软件测试;受控马尔可夫链;关联缺陷;优化算法

中图法分类号: TP311

中文引用格式: 包晓安,谢晓鸣,张娜,曹建文,桂宁.基于缺陷关联度的 Markov 模型软件优化测试策略.软件学报,2015,26(1):14−25. http://www.jos.org.cn/1000-9825/4672.htm

英文引用格式: Bao XA, Xie XM, Zhang N, Cao JW, Gui N. Optimized software testing strategy based on the defect correlation Markov model. Ruan Jian Xue Bao/Journal of Software, 2015,26(1):14−25 (in Chinese). http://www.jos.org.cn/1000-9825/4672.htm

Optimized Software Testing Strategy Based on the Defect Correlation Markov Model

BAO Xiao-An[1], XIE Xiao-Ming[1], ZHANG Na[1], CAO Jian-Wen[2], GUI Ning[1,3]

[1](Faculty of Informatics & Electronics, Zhejiang Sci-Tech University, Hangzhou 310018, China)

[2](Parallel Software Laboratory, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Distrinet Laboratory, University of Leuven, 3001, Belgium)

Abstract: Software testing process normally expects to detect defects as many as possible with minimum cost. In order to reduce the modeling complexity, most works generally assume that all defects are independent of each other. However, in practical testing processes, defects are normally correlated. The software failure severity caused by different defects may also be distinctive. Making full usage of the relationships between correlated defects, it is argued, is beneficial to improve software testing efficiency. This paper proposes a new approach by making usage of the relationship between defects. Firstly, the defects correlation matrix is constructed, and the synthetic balancing weights are designed based on defect correlation coefficient, rebate and detecting rate. Next, the optimal testing problem is converted into a weighted routing problem and a composite optimization algorithm is provided to effectively construct a minimum spanning tree to find an optimal test strategy. Meanwhile, a new defect removing strategy is designed in accordance with the characteristic

* 基金项目: 国家自然科学基金(61202050, 61379036); 浙江省自然科学基金(LY12F02041, Y13F020175); 浙江省钱江人才计划(2013R10015); 浙江理工大学 521 人才培养计划; 浙江省新苗计划(2012R406071)

收稿时间: 2014-04-03; 修改时间: 2014-05-20; 定稿时间: 2014-05-29; jos 在线出版时间: 2014-08-19

CNKI 网络优先出版: 2014-08-19 14:25, http://www.cnki.net/kcms/doi/10.13328/j.cnki.jos.004672.html

of the correlated defects to eliminate defects more efficiently. Simulation results show that the proposed approach has higher effectiveness in terms of defect identification rate and system rewards.

Key words: software testing; controlled Markov chain; correlated defects; optimization algorithm

随着软件规模的不断扩大,结构不断复杂化,软件可靠性[1,2]问题日益成为关注的焦点.软件测试作为软件质量保证的重要手段,正发挥着越来越重要的作用[3].但是,传统的软件测试方法耗时长久,成本昂贵,使得测试效率大打折扣.因此,如何优化测试方法,提高测试效率,成为工业界和学术界研究的重点.为了提高软件测试效率,Cai[4,5]将控制论引入软件测试,首次提出了软件控制论思想.他将软件测试过程看作由被测软件、测试策略组成的闭环反馈系统,并提出利用受控马尔可夫链理论设计和优化测试策略.文献[6]给出了软件测试过程的一种数学模型,并用“m-stage testing model”方法估计模型中剩余缺陷数.文献[7]提出了一种基于软件控制论的优化测试和自适应测试方法,减少了缺陷检测时的测试用例数.针对构件软件内部不可知的特点,文献[8]将自适应软件测试方法推广到构件软件的测试.然而,以上研究大部分都是基于简化的受控马尔可夫链模型,对部分条件进行了特殊化处理(如测试资源不受限制、检测到缺陷立即剔除),使得模型的适用范围受到一定的限制.

为了减小测试模型与运行环境之间的差别,增强已有模型的适用性,文献[9]引入了每个缺陷检测率不等、批量调试等假设,扩大了模型的适用范围.为了反映测试资源对测试过程的影响,文献[10]提出了一种测试资源受约束的马尔可夫链模型,并设计了一种新算法能显著降低在线决策的计算复杂度.在马尔可夫决策模型下,文献[11]基于交叉熵方法,通过一种启发式算法获得软件测试的最优测试剖面,但该方法的优劣性在某种程度上依赖于在线参数估计的精确度.这些研究内容都是基于缺陷独立的假设,在已有模型下通过寻找一种优化测试策略来剔除缺陷,从而达到测试目标.

但在实际测试中,很多缺陷并不是相互独立的,它们之间往往存在某种关联关系.关联缺陷的表现形式多种多样,可以是基于业务逻辑的关联缺陷,也可以是分布在不同系统模块间具有共性的关联缺陷.此外,程序员固有的编程风格也会导致产生的缺陷具有相似性[12].Katerina 和 Trivedi[13]引入了失效关联的概念,并提出了一种 Markov 更新模型来对有失效关联的软件可靠性进行建模.Fiondella 等人[14,15]从组件失效关联的角度提出了一种多变量的伯努利分布(multivariate Bernoulli distribution,简称 MVB)来分析关联组件失效对软件可靠性的影响,并通过筛选关键的组件关联结点进行描述,有效地降低了算法的计算复杂度.徐高潮等人[16]从缺陷关联的角度提出了 P-NHPP(phase-nonhomogeneous poisson process)可靠性模型,认为过早地剔除单个关联缺陷会屏蔽其他相关缺陷的检测能力,是造成软件失效的根源之一,从而影响了软件可靠性模型的评估结果.针对关联缺陷的检测,景涛等人[17]通过 Space 软件[18]实例验证了缺陷之间的关联关系,证明了缺陷关联不满足交换律与结合律,并提出一种缺陷放回的测试方法来剔除关联缺陷.

以上文献大多是从关联缺陷对软件可靠性所带来的负面影响进行研究,但是已有研究也表明:充分利用缺陷之间的关联信息,会使得那些具有共性的缺陷更容易被测试人员所发现.如文献[19]指出,覆盖相同或相似测试需求的测试用例往往会检测出相同或相似的软件缺陷.合理利用缺陷关联信息有助于提高测试效率,如文献[17]指出:在 Space 软件中共包含 36 个缺陷,其中仅仅存在 D_1,D_2,D_{32} 这 3 个缺陷时,它们的检测率几乎为 0,即,完全不能被测试用例库检测出来,只有在其他关联缺陷存在的情况下才能检测到它们.同时,由于每个缺陷对于引起软件失效的严重程度各不相同,这就导致缺陷被剔除后产生的回扣(即测试价值)也不相同.因此,如何利用关联缺陷所带来的正面影响,增加相关缺陷的可检测率,并结合其他测试策略优化缺陷剔除回扣,成为本文研究的重点.

在前期研究中,包晓安等人[20]提出了一种改进的、资源受约束的受控马尔可夫链(controlled Markov chain,简称 CMC)模型.利用该模型,通过在线调整测试策略,从而实现软件的自适应测试.在已有研究的基础上,本文创新性地提出利用缺陷关联信息扩展已有受控马尔可夫链模型,以缺陷的 3 种属性(关联系数、检测率、回扣)作为决策选取权值,将测试过程看作带权最优路径求解问题,优化缺陷剔除回扣.通过与随机测试策略和自适应测试策略相比较,本文的测试策略在缺陷检测数量、测试成本、产生回扣等指标上都有所提升,从而证明了该方法的有效性.

本文的创新点及贡献如下:

(1) 利用缺陷关联特性以及缺陷重要性等级,引入缺陷关联系数与缺陷回扣机制,进一步改善 CMC 测试模型;

(2) 将测试剖面转换为带权路径优化问题,通过改进的优化算法构造最小生成树来获取决策选取序列.该决策序列综合权衡关联系数、缺陷回扣、缺陷检测率三者组成的权值,优先选取权值高的缺陷集;

(3) 为了避免关联缺陷过早删除带来的影响,提出以缺陷集为批量剔除单位的优化缺陷剔除策略.在资源约束的条件下,该方法使得测试决策以关联系数为路径去引导软件状态转移,优先检测关联缺陷集.同时,使得被检测到的关联缺陷集产生尽可能多的回扣值,从而有效节约测试资源,提高测试效率.

本文第 1 节介绍结合关联缺陷的受控马尔可夫链软件测试模型.第 2 节以缺陷关联度、回扣、检测率三者所组成的权值为基础,利用改进的优化算法构造软件测试策略;并提出一种基于优化算法的剔除策略.第 3 节通过仿真实验,将本文的测试策略与随机测试策略和自适应测试策略相比较,进一步验证该方法的有效性.第 4 节总结并探讨未来研究方向.

1 结合关联缺陷的受控马尔可夫链软件测试模型

为了定量化的描述缺陷之间的关联关系,引入关联缺陷矩阵.同时,针对每个缺陷引起软件失效的严重程度的不同,增加缺陷回扣机制,从而进一步扩大 CMC 测试模型的适用性.

1.1 改进的马尔可夫链测试模型

已有的 Markov 链模型将每个时间点 $t(t=0,1,2,\ldots)$上软件剩余缺陷数作为时刻 t 系统所处的状态,则整个状态空间为 $S=\{S_t,x_t\}=\{N\ldots1,0,x_0\}$,其中,$N$ 为期望缺陷总数,x_t 为可用测试资源.在每个时刻 t,从决策空间 $A=\{a_t,t\geq 0\}=\{1,2,\ldots,k\}$中选择一种决策作为测试行为,并且软件状态 S_{t+1} 与历史状态和历史决策无关,只与当前状态 S_t 和决策 a_t 有关.

在传统的 CMC 测试模型中,软件状态的转移率只与缺陷检测率有关,并且忽略了缺陷之间的关联.本文在已有模型的基础上,利用关联缺陷的性质,将缺陷相关系数、缺陷检测率、回扣机制三者组成的权值作为软件状态转移的条件,取代原有模型中仅以缺陷检测率为标准的单一的转换条件,从而使得测试模型在高效性、复杂性、适用性这 3 个方面达到一个平衡.图 1 为改进后的 CMC 测试模型,其中,W_i 表示由关联系数、检测率、回扣三者组成的权值,是选取决策 A_i 的标准.

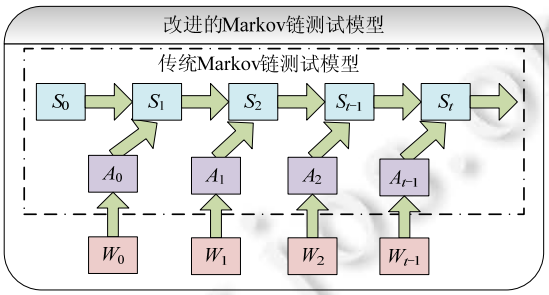


Fig.1 An improved Markov chain testing model

图 1 改进的 Markov 链测试模型

根据软件测试过程,本文的测试模型基于以下定义:

1) 假设时间域是离散的.令:

$$Z_t=\begin{cases}n_1, & t\text{时刻决策检测到缺陷}\\ n_0, & t\text{时刻决策未检测到缺陷}\end{cases} \tag{1}$$

2) 将所有缺陷分成以下 3 种状态:已移除、未检测到、已检测到但未移除,用如下符号表示:

$$Y_t^k=\begin{cases}S_0, & t\text{时刻第}k\text{个缺陷已移除}\\ S_1, & t\text{时刻第}k\text{个缺陷未被检测到}\\ S_2, & t\text{时刻第}k\text{个缺陷已检测到但未被移除}\end{cases} \tag{2}$$

其中, $k=1,2,...,N, Y_0^{(1)}=Y_0^{(2)}=...=Y_0^{(n)}=...=Y_0^{(N)}=S_1$.

3) $\xi_t=(Y_t^{(1)},Y_t^{(2)},...,Y_t^{(n)},Y_t^{(N)},X_t)$ 表示软件状态,其中,X_t 表示 t 时刻剩下的测试资源(包括 t 时刻选择和执行测试用例所需的资源).由于期望缺陷为 n,则 t 时刻的状态可表示为 $\xi_t=(Y_t^{(1)},Y_t^{(2)},...,Y_t^{(n)},X_t)$.

4) 对于一个软件系统,软件中包含 n 个缺陷,$Defects=\{d_1,d_2,...,d_n\}$,其中,第 i 个缺陷记为 d_i.

缺陷检测率(defect detected probability,简称 DDP)用以描述在软件状态 S_t 下,d_i 被检测到的概率 $DDP(d_i)$.将关联缺陷表示如下:

对于任意 $s\in S_t$,如果缺陷 d_i 的发现能够使得 d_j 的检测率改变为 $DDP1(d_j)$,则称缺陷 d_i 与 d_j 为一组关联缺陷(defects correlation),记为 $d_i\wedge d_j=\{(d_i,d_j)|r_{ij}\}$,读作 d_i 关联 d_j,其中,符号∧为关联运算符,但它不满足交换律与结合律[17];r_{ij} 为缺陷之间的相关系数,用以描述缺陷之间的关联程度.

5) 为了定量化地描述缺陷之间的关联关系,引入缺陷关联矩阵,从而利用这种关联关系将测试过程转换为带权值的路径问题.缺陷关联矩阵为

$$\Omega=\begin{bmatrix}r_{11} & r_{12} & \cdots & r_{1n}\\ r_{21} & r_{22} & \cdots & r_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ r_{n1} & r_{n2} & \cdots & r_{nn}\end{bmatrix},\begin{cases}r_{ij}=1, & i=j,\text{表示缺陷的自关联}\\ r_{ij}=0, & \text{表示缺陷之间不存在关联}\\ r_{ij}=\alpha_{ij}, & i\neq j\text{表示存在系数为}\alpha_{ij}\text{的关联性且}\alpha_{ij}\in(0,1)\end{cases} \tag{3}$$

其中,Ω为 n 阶矩阵,r_{ij} 表示第 i 个缺陷与第 j 个缺陷之间的关联系数.一般地,关联关系的量化可以通过需求静态分析方法或者软件耦合度分析的方法来实现.如何较为准确地度量这种关系,并随着测试过程的深入动态调整关联系数,是本文未来的研究方向.

6) $A=\{1,2,\ldots,m,Fin,Del\}$共包含 m+2 个可能的不同决策,每个决策的代价为 $W_{\xi_t}(A_t)$,决策 Del 用以执行配置的剔除策略,决策 Fin 用以将软件置于吸收态.

$$W_{\xi_t}(A_t=i)=\begin{cases}W_{\xi_t}(i)>0, & x_t>0\\ 0, & x_t=0\end{cases} \tag{4}$$

$$W_{\xi_t}(A_t=Del)=\begin{cases}C, & \text{执行剔除策略且}x_t>C\\ \infty, & \text{其他}\end{cases} \tag{5}$$

$$W_{\xi_t}(A_t=Fin)=\begin{cases}\infty, & \text{若}(\min\{W_{\xi_t}(1),W_{\xi_t}(2),...,W_{\xi_t}(m)\}\leqslant x_t)\text{或}(\min\{W_{\xi_t}(1),W_{\xi_t}(2),...,W_{\xi_t}(m)\}>x_t,d\neq 0)\\ x_t, & \text{其他}\end{cases} \tag{6}$$

公式(4)表示,当测试资源为 0 时,应执行决策 Fin 将软件置于吸收态;公式(5)表示执行决策 Del,根据配置的参数剔除所检测到的缺陷,其代价 C 根据所选参数而定,默认配置为检测到 d 个缺陷时执行一次批量剔除;公式(6)表示测试资源充足时,不可执行决策 m+2,而应执行决策 1,2,...,m 或 Del.如果剩余的总资源数小于执行任何一个决策所需的代价,则被测软件转移到吸收态,测试过程也相应地结束.从某种程度上来看,测试资源决定了各个状态下决策的选择.

7) Z_t 由软件当前状态 ξ_t 决定,缺陷检测率用矩阵Θ表示.

$$\Theta=\begin{bmatrix}\rho_1^{(1)} & \rho_1^{(2)} & \cdots & \rho_1^{(n)}\\ \rho_2^{(1)} & \rho_2^{(2)} & \cdots & \rho_2^{(n)}\\ \vdots & \vdots & \ddots & \vdots\\ \rho_m^{(1)} & \rho_m^{(2)} & \cdots & \rho_m^{(n)}\end{bmatrix} \tag{7}$$

其中, $P_r=\{Z_t=n_1 \mid A_t=i\}=\sum_{i=1}^{n}\rho_i^k$, $P_r=\{Z_t=n_0 \mid A_t=i\}=1-\sum_{i=1}^{n}\rho_i^k$,k=1,2,…,n.向量 $\rho=[\rho_{k_1}^{(1)},\rho_{k_2}^{(2)},...,\rho_{k_m}^{(n)}],k_m\subset m$ 表示在 m 个决策下每个缺陷被检测到的最大概率,即,矩阵Θ每列的最大元素.

8) 引入缺陷回扣向量δ=[δ_1,δ_2,…,δ_n],表示每个缺陷所产生的回扣,决策 *Del* 与 *Fin* 不产生回扣.如果决策 A_t 引发了一个未被检测到的缺陷,则该缺陷被剔除后产生回扣δ_x.定义矩阵:

$$R=\begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ R_{21} & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ R_{n1} & R_{n2} & \cdots & R_{nn} \end{bmatrix},$$

其中,$R_{ij}=\delta_i\times\delta_j\times r_{ij}(1\leqslant i\leqslant n,1\leqslant j\leqslant n)$,即,矩阵$\Omega$中每个元素乘以相对应的缺陷回扣$\delta_i$.该矩阵通过一定的运算规则,定量化地表示了缺陷关联回扣值.

缺陷回扣用于描述在测试过程中检测到不同重要性等级的缺陷所产生的测试价值大小.由于缺陷有不同的类型,每个缺陷被发现的概率与剔除的难易程度都不一样,这就导致每个缺陷所产生的回扣也可能不一样.同时,软件系统中每个模块所具有的重要性不同,如某些接口、底层框架、核心业务等.发现一个严重缺陷所引发的软件失效,显然要比发现一个轻微的缺陷回报要大,因此,测试人员更希望能发现这样的缺陷.

1.2 基于马尔可夫模型的资源函数计算

为了增加单位缺陷的剔除回扣率,减少测试代价,根据缺陷检测率、回扣、关联系数设计资源函数,使得测试过程中产生的总回扣最大,从而达到提高测试效率的目的.

令函数 $f=f(\rho_i^j,R_{ij},A_t)$ 表示计算关联缺陷被检测到的期望回扣,其中,参数 $\rho_i^j(1\leqslant i\leqslant m,1\leqslant j\leqslant n)$ 表示缺陷的可检测率,R_{ij} 表示缺陷关联回扣值,A_t 表示 t 时刻采取的决策.软件的状态转移率只与当前状态和 t 时刻采取的决策 A_t 有关,则有如下状态转移函数:

$$q_{\xi_t\xi_{t+1}}(i)=P_r\{\xi_{t+1} \mid \xi_t,A_t=i\}=\begin{cases}\sum_{k=1}^{n}\rho_i^k, & Z_t=n_1,x_{t+1}\geqslant 0\\ 1-\sum_{k=1}^{n}\rho_i^k, & Z_t=n_0,x_{t+1}\geqslant 0\\ 0, & \text{其他}\end{cases} \tag{8}$$

其中, $x_{t+1}=x_t-W_{\xi_t}(i)+lf(\rho_i^j,R_{ij},A_t)$ 且 $l=\begin{cases}1, & Z_t=n_1\\ 0, & Z_t=n_0\end{cases},i=1,2,...,m.$

$$q_{\xi_t\xi_{t+1}}(del)=P_r\{\xi_{t+1} \mid \xi_t,A_t=del\}=\begin{cases}n_1, & \text{如果}c<x_t\text{,且满足批量剔除条件}\\ n_0, & \text{其他}\end{cases},$$

$$q_{\xi_t\xi_{t+1}}(fin)=P_r\{\xi_{t+1} \mid \xi_t,A_t=fin\}=\begin{cases}n_0, & \text{若}(\min\{W_{\xi_t}(1),W_{\xi_t}(2),...,W_{\xi_t}(m)\}\leqslant x_t)\text{或}\\ & \quad(\min\{W_{\xi_t}(1),W_{\xi_t}(2),...,W_{\xi_t}(m)\}>x_t,d\neq 0).\\ n_1, & \text{其他}\end{cases}$$

令τ为软件首次到达吸收态的时间,则总的测试资源满足以下等式:

$$x_0=\sum_{t=0}^{\tau}[W_{\xi_t}(A_t)-\rho_{A_t}^{*}f(\rho_i^j,R_{ij},A_t)]+x_\tau+(n/d+1)C \tag{9}$$

令

$$\rho_{A_t}^{*}(A_t=i)=\begin{cases}\sum_{k=1}^{n}\rho_i^k sign(Y_t^{(k)}-S_2)sign(-Y_t^{(k)}), & i=1,2,...,m\\ 0, & i=Del,Fin\end{cases} \tag{10}$$

其中,*sign*(x)为符号函数.由于公式(9)中最后一项中 n 和 C 都是常数,因此其值只与 d 有关.当 d 一定时,该项为常

数,消耗的测试资源一定,即,不影响测试决策选取.$x(\tau)$虽然不是一个定值,但它是测试策略确定之后剩余的测试资源,同样可以认为不影响测试决策的形成.令

$$J_\omega(x_0)=\sum_{t=0}^{t}\rho_{A_t}^{*}f(\rho_i^j,R_{ij},A_t) \tag{11}$$

其中,ω表示测试策略,它确定如何在测试过程中选取测试用例;$J_\omega(x_0)$表示测试过程停止时ω产生的代价总折扣.为此,将研究的问题转化为设计一个基于 CMC 模型的测试剖面,使 $f(\rho_i^j,R_{ij},A_t)$ 函数产生的总折扣最大,即,获取 $J_\omega(x_0)$的最大值,则总的测试代价就相应地有所减小.在软件测试过程中资源总是有限的,而利用缺陷相关系数使得缺陷剔除时产生的期望回扣越大,测试代价就越少,从而增加了可检测的缺陷数,同时也减少了相对严重的缺陷引起软件失效的概率,增加了软件的可靠性.

2 基于关联缺陷的测试剖面优化

在受控马尔可夫链的测试模型中,由于关联缺陷的存在,使得缺陷之间可以形成一张无向连通网,并通过最小生成树算法求边权值的极值,进而达到对资源的优化.同时,为了计算测试模型中 $f(\rho_i^j,R_{ij},A_t)$ 函数的最大值,我们将测试模型看作带权路径最优求解问题,那么就可以利用最小生成树求解测试策略的优化问题.其基本思路是:

(1) 每个缺陷作为顶点 V,缺陷之间的关联系数、缺陷产生回扣以及缺陷的可检测率这 3 个元素的乘积作为路径的权值.如果缺陷之间不存在关联,则不存在相应的路径;

(2) 在路径中,寻找由关联缺陷组成的连通分量,并入集合 S;

(3) 根据权值大小,对 S 中元素进行从大到小的排序;

(4) 通过优化算法,依次从集合 S 中选择具有最大权值的连通分量构造最小生成树,由于最小生成树能用最小的代价连通网中的缺陷顶点,因此它在构成过程中生成的顶点序列集即为选取决策基于路径权值下的最优测试策略;

(5) 批量剔除生成连通路径中所有可剔除的缺陷;

(6) 根据关联缺陷及资源情况判断测试停止条件.

2.1 算法初始化

将每个缺陷作为顶点,则 V={1,2,3,...,n},令 C_{ij} 为边的权值.用矩阵表示如下:

$$C=\begin{bmatrix}1 & C_{12} & \cdots & C_{1n}\\ C_{21} & 1 & \cdots & C_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ C_{n1} & C_{n2} & \cdots & 1\end{bmatrix},C_{ij}=R_{ij}\times\rho_{k_1}^{i}\times\rho_{k_2}^{j}(1\leqslant i\leqslant n,1\leqslant j\leqslant n,1\leqslant k_1\leqslant m,1\leqslant k_2\leqslant m).$$

即,用关联缺陷系数、缺陷回扣、缺陷可检测率的乘积表示缺陷 i 与 j 之间的权值.由于软件系统中关联缺陷只占部分,剩余缺陷都具有独立性,使得矩阵 C 中存在零元素,所有顶点不一定能构成连通图,但一定存在连通分量.因此,需要对矩阵 C 进行遍历,搜索具有最大权值的连通分量并以此构造最小生成树.其基本步骤如下:

1) 定义集合 S={S_i}用于保存搜索到的各个连通分量;栈 *stack* 用于保存当前搜索路径中的顶点元素;L 为 LinkedList 类型,用于存储所经过的顶点元素并记录相应的权值;

2) 随机选取一个关联缺陷作为“当前顶点”,从该顶点出发开始搜索;

3) 如果“当前顶点”存在关联缺陷,则选择拥有最大权值的边的顶点纳入连通分量,并将该缺陷作为新的“当前顶点”;

4) 从新的“当前顶点”出发,继续搜索下一个关联缺陷;

5) 如果当前顶点不存在未连通的缺陷,则应顺着原先的“来向”退回到前一个顶点位置,然后继续遍历除“来向”之外的其余关联顶点;

6) 当没有顶点可以遍历时,L 中元素即为连通分量 S_i,所经过的路径即为该连通分量 S_i 的权值;

7) 重复步骤 2)~步骤 6),直到矩阵中没有关联缺陷为止.

2.2 改进的优化算法测试剖面

构造最小生成树可以有多种方法,其中多数算法都利用了一种简称为 MST 的性质.普里姆(Prim)算法与克鲁斯卡尔(Kruskal)算法是两种利用 MST 性质构造最小生成树的典型算法.在普里姆算法中,假设 $S_{max}=(V,\{E\})$ 是连通网,TE 是 S_{max} 上最小生成树边的集合,U 是顶点集 V 的一个非空子集.算法从 $U=\{u_0\}(u_0\in V)$,$TE=\{\cdot\}$开始,重复执行下述操作:在所有 $u\in V,v\in V-U$ 的边$(u,v)\in E$ 中找一条代价最大的边(u_0,v_0)并入集合 TE,同时,v_0 并入 U,直到 $U=V$ 为止.此时,TE 中必有 $n-1$ 条边,则 $T=(V,\{TE\})$为 S 的最小生成树.而克鲁斯卡尔算法是用另一种方法求网的最小生成树:假设 $S_{max}=(V,\{E\})$是连通网,则令最小生成树的初始状态为只有 n 个顶点且无边的非连通图 $T=(V,\{\cdot\})$.在 E 中选择代价最大的边,若该边依附的顶点落在 T 中不同的连通分量上,则将此边加入到 T 中;否则去除此边,而选择下一条代价最大的边.以此类推,直到 T 中所有顶点都在同一连通分量上为止.

对比以上两种算法可以看出:

- 普里姆算法主要是一个两重循环,其中每一重循环的次数均为结点个数 n,因而该算法的时间复杂度为 $O(n^2)$.由于该算法的时间复杂度只与网中结点个数有关,与网中边的条数无关,因此,适用于求边稠密的网的最小生成树;
- 与此相反,克鲁斯卡尔算法的时间复杂度为 $O(e\log e)$(e 为网中边的数目,log 为对数),因此适用于求边稀疏的网的最小生成树.

随着测试过程的推进,被测软件系统中的关联缺陷会逐渐减少,因而采用某种单一的算法来构造最小生成树就显示出效率低下的弊端.针对以上问题,本文提出一种复合的优化算法:引入参数 *param*,当软件系统中关联缺陷数量大于 *param* 设定的值时,则执行普里姆算法;反之,则执行克鲁斯卡尔算法.该参数以软件系统中关联缺陷的数量为标准,结合普里姆算法与克鲁斯卡尔算法各自的优点,动态地选取最优算法来构造最小生成树,从而降低算法的时间复杂度,有效提高测试效率.基本算法为:

1) 每次构造前从集合 S 中筛选具有最大权值的连通分量 S_{max};
2) 根据 *param* 参数动态选择最优算法构造最小生成树;
3) 集合 G 依次记录每次构成的最小生成树的顶点 V,初始化集合 $S=\{S-S_{max}\}$,重复执行步骤 1)、步骤 2),直到集合 S 为空集或者满足测试停止条件;
4) 令 $T_k=\sum S_{max}$, 表示从集合 S 中选取具有最大权值的连通域构造最小生成树的权值之和;
5) 目标函数值 $f(\rho_i^j,R_{ij},A_i)=\sum T_k+C_1,\sum T_k$ 为最小生成树按照权值高低排序后依次相加所构成,其中,C_1 表示剩余没有关联性的缺陷所产生的权值.

算法 1. 构造最小生成树算法.

```
Public LinkedList OptAlgorithm(S) {
  LinkedList G=new LinkedList(·);
  SumCost=0;
  For (i=1; i<S.length; i++) {
    S=SelectMax(S);                          //选择具有最大权值的连通分量
    MinSpanTree tree=new MinSpanTree(·);
  If (defects<param) {                       //根据参数动态选择构造最小生成树的方法
    Tree=MinSpanTree_Prim(S,U);
  } else {
    tree=MinSpanTree_Kruakal(S,U);
  }
    SumCost=SumCost+GetWeight(tree);
```

```
            G.add(tree);
        }
        return G;
    }
```

综上可得,该算法可使目标函数 $f(\rho_i^j, R_{ij}, A_t)$ 取得最大回扣.集合 G 中的顶点序列就构成了最优测试序列.利用缺陷关联系数,使得缺陷回扣、检测率以及关联度所组成的三维向量达到一个相对平衡.同时,综合权衡三者所产生的回扣,令测试过程中目标函数 $f(\rho_i^j, R_{ij}, A_t)$ 的决策 A_t 选取满足集合 G 的序列,即构造最小生成树的路径.那么,该测试策略即为本文的最优测试策略.

2.3 基于关联缺陷的剔除策略优化

一般来说,软件缺陷剔除策略可分为立即剔除与批量剔除两种.在实际测试过程中,考虑到时间、人力等成本,剔除缺陷通常需要消耗一定的资源.由于立即剔除相对于批量剔除来说会增加剔除次数,增加测试成本,因而目前普遍采用批量剔除方式.批量剔除策略一般是当测试过程满足某个预定义的阈值,如测试次数、软件失效的次数和测试时间[21]时进行剔除.然而,当前的批量剔除策略没有考虑缺陷之间的关联,这样的删除可能会导致某些缺陷被屏蔽,从而影响测试结果.

因此,本文利用基于关联缺陷的优化算法,提出一种新的剔除策略:每当构造一棵新的最小生成树时,就批量剔除该树中所包含的所有缺陷,这样也避免了相关缺陷的过早删除.由于运用优化算法构造的最小生成树是一组关联缺陷的集合,因此,以该集合为单位进行批量剔除可以有效降低剔除过程的复杂度,节约剔除成本.当存在关联缺陷时,采用上一节给出的动态 *Del* 参数配置策略;如果所有关联缺陷都被剔除,则剩余缺陷按照默认策略执行.假设每次剔除都不会引入新的缺陷,每次剔除消耗的资源为常数,那么,改进后的公式(9)可以写成

$$x_0 = \sum_{t=0}^{\tau}[W_{\xi_t}(A_t) - \rho_{A_t}^* f(\rho_i^j, R_{ij}, A_t)] + x_\tau + (n'/d+1)C + kC_0 \tag{12}$$

其中,$(n'/d+1)C$ 表示按照原先 CMC 模型中设定的剔除策略执行所消耗的测试资源,kC_0 表示优化后的剔除策略所消耗的资源.

该剔除策略根据测试过程的动态结果来设定批量剔除的阈值,并利用关联缺陷的性质,使得具有关联度的缺陷一起被剔除,从而降低了批量剔除的难度,节约了批量剔除的成本.同时,由优化算法所构造的最小生成树根据权重按降序排列,可满足最先剔除的缺陷产生最大回扣的条件,从而提高了测试价值和测试效率.

3 仿真实验

为了验证本文所设计的测试策略的有效性,将其分别与随机测试策略[22]和自适应测试策略[20]进行比较.在随机测试中,通过均匀概率分布从决策集中随机地选择决策,每个决策被选取的概率相等.在自适应测试中,大部分文献所描述的方法都整合了参数估计的模块,通过在线收集测试过程中的数据进行参数估计,动态调整测试策略,从而提高测试效率.但自适应测试的效率在一定程度上依赖于参数估计的准确度,更确切地说,是与所采用的参数估计算法的优劣性密不可分,甚至还与测试模型的有效性、测试环境的特殊性等有关.因此,为了减少以上各种因素对于参数估计模块的影响,统一比较标准,并降低计算复杂度,本文假设在自适应测试中,参数估计的准确度为实际测试结果的 90%.即,将模型中给定参数的 90%近似当作仿真实验的参数.

假设测试开始时,令 n=6,d=2,*param*=5,决策集 A_t={1,2,...,12}包含 12 个决策,其中,决策 1~决策 10 执行缺陷检测;决策 11 执行对应的剔除策略;当满足测试停止条件时,决策 12 将软件状态转移到吸收态.δ_i 为第 i 个缺陷被检测到所产生的回扣,设置δ_i=[10,15,40,25,20,18]表示缺陷回扣向量.

在实际测试过程中,由于每一步测试所消耗的资源都不等,为了降低计算复杂度,这里假设执行每个决策所消耗的资源相等,即 $W_{\xi_t}(1) = W_{\xi_t}(2) = ... = W_{\xi_t}(10) = 15$. 同时,由于缺陷关联矩阵中元素分布结构不影响后期决策的选取,因此将其看作对称矩阵以便简化计算.Ω表示缺陷关联矩阵,Θ表示转移概率矩阵,如下所示:

$$\Omega=\begin{bmatrix}1.00 & 0.00 & 0.03 & 0.05 & 0.08 & 0.10\\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00\\ 0.03 & 0.00 & 1.00 & 0.04 & 0.05 & 0.00\\ 0.05 & 0.00 & 0.04 & 1.00 & 0.00 & 0.05\\ 0.08 & 0.00 & 0.05 & 0.00 & 1.00 & 0.02\\ 0.10 & 0.00 & 0.00 & 0.05 & 0.02 & 1.00\end{bmatrix},\Theta=\begin{bmatrix}0.06 & 0.20 & 0.15 & 0.16 & 0.18 & 0.31\\ 0.08 & 0.15 & 0.20 & 0.10 & 0.27 & 0.25\\ 0.06 & 0.16 & 0.24 & 0.12 & 0.36 & 0.27\\ 0.05 & 0.18 & 0.10 & 0.15 & 0.20 & 0.25\\ 0.06 & 0.14 & 0.12 & 0.16 & 0.26 & 0.45\\ 0.08 & 0.15 & 0.16 & 0.12 & 0.28 & 0.32\\ 0.07 & 0.12 & 0.25 & 0.14 & 0.32 & 0.28\\ 0.08 & 0.12 & 0.12 & 0.15 & 0.25 & 0.38\\ 0.08 & 0.18 & 0.18 & 0.18 & 0.25 & 0.25\\ 0.09 & 0.15 & 0.21 & 0.15 & 0.30 & 0.36\end{bmatrix}.$$

在给定的关联度矩阵中,由于关联系数的整体放大与缩小并不会影响以权值为标准的决策选取结果,同时,为了降低计算复杂度,在计算时,我们将非 0 与非 1 的系数转化成整数带入公式,于是有:

$$\delta_i\rho^i_{\max}=[0.9\ 3.0\ 10.0\ 4.5\ 7.2\ 8.1].$$

根据算法规则,得到测试路径权值矩阵:

$$C=\begin{bmatrix}1.00 & 0.00 & 27.00 & 20.25 & 51.84 & 72.90\\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00\\ 27.00 & 0.00 & 1.00 & 180.00 & 360.00 & 0.00\\ 20.25 & 0.00 & 180.00 & 1.00 & 0.00 & 182.25\\ 51.84 & 0.00 & 360.00 & 0.00 & 1.00 & 116.64\\ 72.90 & 0.00 & 0.00 & 182.25 & 116.64 & 1.00\end{bmatrix}.$$

所构造的最小生成树的顶点集合 G={5,3,4,6,1},因此,所选测试决策对缺陷的检测顺序为 D_i={5,3,4,6,1,2},相应选取的决策顺序为 A_i={3,7,9,5,10,1}.

根据以上决策,采用基于关联缺陷的优化剔除策略,并结合给定的相关参数分别对优化测试策略与随机测试和自适应测试策略进行仿真实验,表 1 为随机选取的 10 次仿真结果.$Rd(i)$,$Ad(i)$,$Od(i)$分别为第 i 次仿真随机测试(RT)、自适应测试(AT)、优化测试(OT)检测并剔除全部缺陷所需的执行次数,$Rc(i)$,$Ac(i)$,$Oc(i)$分别为第 i 次仿真随机测试、自适应测试、优化测试所花费的总成本.

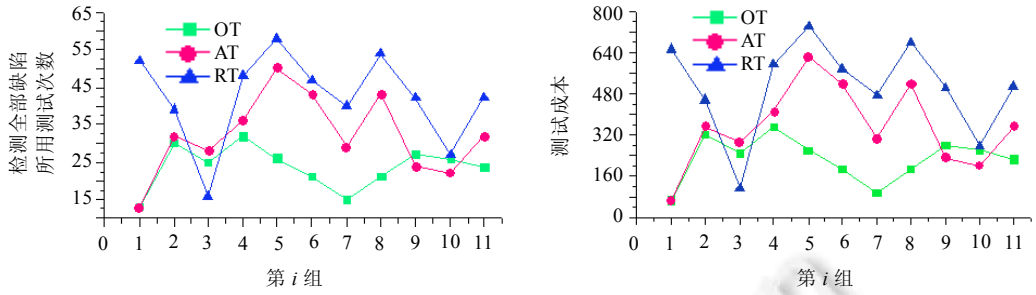
Table 1 Testing times and costs for detecting all defects with three different testing strategies: Random testing strategy, adaptive testing strategy and optimal testing strategy

表 1 分别使用 RT,AT 和 OT 检测所有缺陷所需测试次数与成本比较

第 i 次	1	2	3	4	5	6	7	8	9	10	平均值
$Rd(i)$	52	39	16	48	58	47	40	54	42	27	42.3
$Ad(i)$	13	32	28	36	50	43	29	43	24	22	32.0
$Od(i)$	13	30	25	32	26	21	15	21	27	25	23.6
$Rc(i)$	652	457	112	592	742	577	472	682	502	277	506.5
$Ac(i)$	67	352	292	412	622	517	307	517	232	202	352.0
$Oc(i)$	67	322	247	352	262	187	97	187	277	262	226.0

通过分析表中数据可知:当测试停止时,优化测试平均花费 23.6 的测试次数和 226.0 的测试成本,分别低于自适应测试(32.0 的测试次数和 352.0 的测试成本)和随机测试(42.3 的测试次数和 506.5 的测试成本).同时,优化测试比自适应测试和随机测试平均少执行 26.25%和 44.21%的测试次数;比自适应测试和随机测试平均少花费 35.80%和 55.38%的成本.为了更直观地反映表 1 中 3 种测试策略间的差异,用图 2(a)和图 2(b)分别表示检测所有缺陷所需的测试次数和测试成本.从图中可以看出:尽管在第 3 组实验结果中,随机测试策略比其他两种测试策略消耗更少的测试次数与测试成本,但在多数情况下,优化测试比其他两种测试策略具有一定的优势.造成这一现象的原因在于:由于随机测试以相等概率从决策集中选取测试决策,即,每个决策被选取的概率都相等,因而存在一定的概率使得在某一时刻 t 所选取的决策恰好优于其他两种策略.但从总体上来看,基于缺陷关联的

优化测试在测试次数与测试成本方面要优于随机测试策略与自适应测试策略.



(a) 3 种策略执行测试次数比较 (b) 3 种策略花费成本比较

Fig.2 The number of tests and cost under three strategies

图 2 3 种策略下测试次数与成本的比较

由于假设每步测试所消耗的资源都相等,因此测试问题就转化为在给定有限的测试步数内,比较两种测试策略的优劣性.表 2 为在给定测试步数为 20 时,优化测试与随机测试、自适应测试在缺陷剔除数量、检测次数和产生回扣上的比较情况.表中分别列出 4 组数据,每组数据的均值由随机选取的 100 次测试结果求得.很明显,对比表中数据可以看出,优化测试比随机测试在缺陷剔除数量上约多 55.41%,在产生回扣上约多 85.32%.同时,与自适应测试相比,在剔除缺陷数量上约多 18.56%.在产生回扣上约多 34.10%.注意到,OT 对于回扣的增长率要高于剔除缺陷的增长率,即,对于剔除相同数量的缺陷,优化测试可优先选取回扣值更大的缺陷进行检测,从而产生更多的回扣.

Table 2 The number of removed defects and rebates under three strategies

表 2 3 种测试策略在剔除缺陷数和回扣值上的比较

i	基于关联缺陷的优化测试策略(OT)			自适应测试策略(AT)			随机测试策略(RT)		
	剔除缺陷数	执行次数	产生回扣	剔除缺陷数	执行次数	产生回扣	剔除缺陷数	执行次数	产生回扣
1	4.48	15.21	105.95	3.79	14.52	78.11	3.06	13.32	59.89
2	4.69	15.28	109.04	3.84	14.97	79.90	2.96	14.30	57.68
3	4.60	15.89	107.73	3.98	14.25	82.74	3.03	13.58	60.48
4	4.64	15.91	108.39	3.89	14.55	80.72	2.80	14.02	54.59
平均值	4.60	15.57	107.78	3.88	14.57	80.37	2.96	13.81	58.16
百分比(%)	–	–	–	18.56	6.86	34.10	55.41	12.74	85.32

针对实验结果,相比于缺陷独立的假设,充分利用缺陷的关联信息能够有助于发现相似缺陷,提高相关缺陷的检测率,降低检测与剔除的难度,因而在测试过程中能用更少的测试资源检测到尽可能多的缺陷.同时,利用缺陷之间的关联关系,将测试过程转换成基于缺陷检测率、关联系数和缺陷回扣的路径优化问题,从而使得选取的决策序列能够产生更多的回扣.因此,基于缺陷关联的优化测试策略具有更高的检测效率.

4 总结以及未来工作

本文在前期研究成果的基础上,利用关联缺陷的性质,引入关联缺陷系数矩阵,将测试问题转换成带权的路径问题.即,缺陷编号为顶点,缺陷回扣、缺陷检测率、关联系数组成的三维向量为权值,并以软件系统中关联缺陷数量为标准,结合普里姆算法和克鲁斯卡尔算法的优点,动态选择最优算法构造最小生成树来选取拥有最大连通图的关联缺陷集,使得检测到的缺陷回扣为最大.从而在资源约束的情况下,增加了缺陷检测的回扣率,提高了测试的可持续性.同时,分析现有剔除策略,根据最优测试剖面,将原有的剔除条件从静态转变为动态,一方面降低了剔除的复杂性,另一方面也节约了批量剔除的成本.通过实验仿真数据表明:相比于随机测试和自适应测试,最优测试策略在检测缺陷数量和测试总回扣上明显优于其他两种测试策略.因此,我们的测试策略是有效的.

由于本文的测试模型是假设关联系数矩阵已知,但在实际的运行环境及不同的软件种类中,关联缺陷特性可能有所不同.因此,如何较为准确地估计关联缺陷系数,继续深入研究缺陷关联的其他形式(如一对多的关联),成为我们下一阶段的研究方向.当前,我们正在研究如何有效地根据测试过程中收集到的关联缺陷信息动态调整关联系数,以获得更加准确的缺陷关联关系.

References:

[1] Hu H, Jiang CH, Cai KY, Wong WE, Mathur AP. Enhancing software reliability estimates using modified adaptive testing. Information and Software Technology, 2013,55(2):288−300. [doi: 10.1016/j.infsof.2012.08.012]

[2] Zhou KJ, Wang XL, Hou G, Wang J, Ai SB. Software reliability test based on Markov usage model. Journal of Software, 2012,7(9):2061−2068. [doi: 10.4304/jsw.7.9.2061-2068]

[3] Yan J, Wang J, Chen HW. Deriving software Markov chain usage model from UML models. Ruan Jian Xue Bao/Journal of Software, 2005,16(8):1386−1394 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1386.htm

[4] Cai KY, Li YC, Jing T, Bai CG. Software testing in the context of software cybernetics. Acta Aeronautica ET Astronautica Sinica, 2002,23(5):448−454 (in Chinese with English abstract). [doi: 10.3321/j.issn:1000-6893.2002.05.011]

[5] Cai KY, Cangussu JW, De Carlo RA, Mathur AP. An overview of software cybernetics. In: O'Brien L, ed. Proc. of the 11th Annual Int'l Workshop on Software Technology and Engineering Practice. Los Alamitos: IEEE Computer Society Press, 2004. 77−86. [doi: 10.1109/STEP.2003.4]

[6] Cai KY, Dong Z, Liu K. On several issues in software reliability testing. Chinese Journal of Engineering Mathematics, 2008,25(6): 967−978 (in Chinese with English abstract). [doi: 10.3969/j.issn.1005-3085.2008.06.002]

[7] Cai KY. Optimal software testing and adaptive software testing in the context of software cybernetics. Information and Software Technology, 2002,44(2):841−855. [doi: 10.1016/S0950-5849(02)00108-8]

[8] Cai KY, Chen TY, Li YC, Ning WY, Yu YT. Adaptive testing of software components. In: Liebrock PM, ed. Proc. of the 2005 ACM Symp. on Applied Computing. Santa Fe: ACM Press, 2005. 1463−1469. [doi: 10.1145/1066677.1067011]

[9] Hu H, Jiang CH, Cai KY. Adaptive software testing in the context of an improved controlled Markov chain model. In: Kawada S, ed. Proc. of the 32nd Annual IEEE Int'l Conf. on Computer Software and Applications (COMPSAC 2008). Los Alamitos: IEEE Computer Society, 2008. 853−858. [doi: 10.1109/COMPSAC.2008.186]

[10] Cai KY, Li YC, Ning WY, Wong WE, Hu H. Optimal and adaptive testing with cost constraints. In: Zhu H, ed. Proc. of the 2006 Int'l Workshop on Automation of Software Test. Los Alamitos: IEEE Computer Society, 2006. 71−77. [doi: 10.1145/1138929.1138944]

[11] Zhang DP, Ni CH, Xu BW. Cross-Entropy method based on Markov decision process for optimal software testing. Ruan Jian Xue Bao/Journal of Software, 2008,19(10):2770−2779 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/2770.htm [doi: 10.3724/SP.J.1001.2008.02770]

[12] Zhang YQ, Zheng Z, Ji XH, Zhang WB, Zhang ZY. Markov model-based effectiveness predicting for software fault localization. Chinese Journal of Computers, 2012,36(2):445−456 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2013.00445]

[13] Katerina GP, Trivedi KS. Failure correlation in software reliability models. IEEE Trans. on Reliability, 2000,49(1):37−48. [doi: 10.1109/24.855535]

[14] Fiondella L, Rajasekaran S, Gokhale SS. Efficient software reliability analysis with correlated component failures. IEEE Trans. on Reliability, 2013,1(62):244−255. [doi: 10.1109/TR.2013.2241131]

[15] Fiondella L, Rajasekaran S, Gokhale S. Efficient system reliability with correlated component failures. In: Agarwal A, ed. Proc. of the 2011 IEEE 13th Int'l Symp. on High-Assurance Systems Engineering. Los Alamitos: IEEE Computer Society, 2011. 269−276. [doi: 10.1109/HASE.2011.31]

[16] Xu GC, Liu XZ, Hu L, Fu XD, Dong YS. Software reliability assessment models incorporating software defect correlation. Ruan Jian Xue Bao/Journal of Software, 2011,22(3):439−450 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3739.htm [doi: 10.3724/SP.J.1001.2011.03739]

[17] Jing T, Jiang CH, Hu DB, Bai CG, Cai KY. An approach for detecting correlated software defects. Ruan Jian Xue Bao/Journal of Software, 2005,16(1):17−28 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/17.htm

[18] Rothermel G, Unteh RH, Chu C, Harrold MJ. Prioritizing test cases for regression testing. IEEE Trans. on Software Engineering, 2001,27(10):929−948. [doi: 10.1109/32.962562]

[19] Zhao D, Upadhyaya S. Dynamically partitioned test scheduling with adaptive TAM configuration for power-constrained SoC testing. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2005,24(6):956−965. [doi: 10.1109/TCAD.2005.847893]

[20] Bao XA, Yao L, Zhang N, Song JY. Adaptive software testing based on controlled Markov chain. Journal of Computer Research and Development, 2012,49(6):1332−1338 (in Chinese with English abstract).

[21] Cao P, Dong Z, Liu K, Cai KY. Analysis strategy of removing defects in software testing. In: Long Y, ed. Proc. of the 11th National Youth Conf. on Information and Management Sciences. Chongqing: Global-Link Publisher, 2009. 137−141 (in Chinese with English abstract).

[22] Dalley JL. The art of software testing. In: Proc. of the IEEE 1991 National Aerospace and Electronics Conf. (NAECON'91). Piscataway: IEEE, 1991. 757−760.

附中文参考文献:

[3] 颜炯,王戟,陈火旺.基于UML的软件Markov链使用模型构造研究.软件学报,2005,16(8):1386−1394. http://www.jos.org.cn/1000-9825/16/1386.htm

[4] 蔡开元,李永超,景涛,白成刚.软件测试的控制论方法.航空学报,2002,23(5):448−454. [doi: 10.3321/j.issn:1000-6893.2002.05.011]

[6] 蔡开元,董昭,刘克.关于软件可靠性测试的若干问题.工程数学学报,2008,25(6):967−978. [doi: 10.3969/j.issn.1005-3085.2008.06.002]

[11] 张德平,聂长海,徐宝文.基于 Markov 决策过程用交叉熵方法优化软件测试.软件学报,2008,19(10):2770−2779. http://www.jos.org.cn/1000-9825/19/2770.htm [doi: 10.3724/SP.J.1001.2008.02770]

[12] 张云乾,郑征,季晓慧,张文博,张震宇.基于马尔可夫模型的软件错误定位方法.计算机学报,2013,36(2):445−456. [doi: 10.3724/SP.J.1016.2013.00445]

[16] 徐高潮,刘新忠,胡亮,付晓东,董玉双.引入关联缺陷的软件可靠性评估模型.软件学报,2011,22(3):439−450. http://www.jos.org.cn/1000-9825/3739.htm [doi: 10.3724/SP.J.1001.2011.03739]

[17] 景涛,江昌海,胡得斌,白成刚,蔡开元.软件关联缺陷的一种检测方法.软件学报,2005,16(1):17−18. http://www.jos.org.cn/1000-9825/16/17.htm

[20] 包晓安,姚澜,张娜,宋瑾钰.基于受控 Markov 链的软件自适应测试策略.计算机研究与发展,2012,49(6):1332−1338.

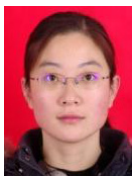
[21] 曹平,董昭,刘克,蔡开元.软件测试的剔除策略分析.见:龙勇,编.第 11 届中国青年信息与管理学者大会论文集.重庆:香港 Global-Link Publisher 出版社,2009.137−141. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=165837 [doi: 10.1109/NAECON.1991.165837]

包晓安(1973－),男,浙江东阳人,教授,主要研究领域为自适应软件,软件测试,智能信息处理.

曹建文(1969－),男,博士,研究员,博士生导师,主要研究领域为高性能并行软件与算法的研究与开发.

谢晓鸣(1987－),男,硕士生,主要研究领域为软件工程,软件测试技术.

桂宁(1977－),男,博士,讲师,主要研究领域为软件工程,自适应软件.



张娜(1977－),女,副教授,主要研究领域为软件工程,软件测试技术.