在测试用例不放回时比较随机测试和分割测试*

方木云, 赵保华, 屈玉贵

(中国科学技术大学 计算机科学与技术系,安徽 合肥 230027)

E-mail: bhzhao@ustc.edu.cn

http://www.ustc.edu.cn/ch/school

摘要: 在测试用例放回的情况下,关于随机测试和分割测试的比较,许多研究者做了大量的工作,取得了显著成果.在测试用例不放回的情况下,类似的比较工作在国内外文献中尚未见到.然而在实际工作中,尤其是在软件测试早期和模块测试阶段,测试用例是不放回的.因此,在测试用例不放回的情况下,对随机测试和分割测试进行了比较,得出4个结论.与Chen和Yu在测试用例放回情况下的研究成果相比,一个不同的发现是,在平分子域、错误数、测试次数时,分割测试不如随机测试效果好.另外还发现,如何利用各种信息分割出错误集中的区域,然后着重测试,这是分割测试的核心.

关键词: 随机测试;分割测试;软件可靠性;软件测试

中图法分类号: TP311 **文献标识码**: A

分割测试是一种重要的测试技术,方法是将程序的输入域分成若干个子域,然后在每个子域内选取元素进行测试.语句覆盖、分支测试、路径测试、结构测试等都可以看成是分割测试.要注意的是,每个子域并不一定是相互独立的,有可能产生重叠.分割测试的目的是让子域内的元素具有代表意义,或者让引起错误输出的那些输入元素更多地集中在某些子域内,然后集中测试这些子域,以发现更多错误,提高软件可靠性.由于测试者事先并没有错误分布的信息,不一定能分出这样的子域,所以,分割策略对于分割测试的效果至关重要.

随机测试可以看成是分割测试的特例,因为它仅有一个子域——整个输入域,因此它不必花费精力去研究如何分割输入域以及记录那些子域是否测试过.

Duran和Ntafos[1]以及Hamlet和Taylor[2]分别做过一系列模拟实验,比较分割测试和随机测试的效果.其结果令人吃惊.他们发现,即便是在发现错误这一点上分割测试也比随机测试强,但其优势微乎其微.如果加上分割工作的费用,对于每发现一个错误来说,随机测试的开销要小于分割测试的开销.

Weyuker和Jeng[3]对分割策略进行了正规的研究,并比较了分割测试和随机测试的效果.他们发现,分割测试可能好于、等同于、不如随机测试.好的分割测试的效果依赖于分割策略是否能集中引起错误输出的输入(failure-causing inputs).他们的研究成果成功地解释了Duran和Ntafos以及Hamlet和Taylor得出的与人们的直觉相反的实验结果.

Chen和Yu[4]扩展并总结了Weyuker和Jeng的一些成果,使得随机测试和分割测试的比较在理论上更为完善.详细的内容请读者参阅文献[4].

* **收稿日期**: 2000-02-29; **修改日期**: 2000-08-18

基金项目: 国家自然科学基金资助项目(90104010); 国家教育部博士点基金资助项目(2000035802)

作者简介: **方木云**(1968—),男,湖北罗田人,讲师,主要研究领域为软件工程;**赵保华**(1947—),男,安徽霍丘人,教授,博士生导师,主要研究领域为软件工程,通信软件与协议工程;**屈玉贵**(1945—),女,山东聊城人,副教授,主要研究领域为软件工程.

以上的研究工作是在假设测试用例放回(with replacement)的条件下做的.然而在软件测试早期,为了发现错误、除去错误和提高软件可靠性,测试用例是不放回的.我们发现,在测试用例放回时,随机测试和分割测试发现错误的能力极其低下,只适用于在软件测试后期评估软件可靠性,决定软件投放时间.因此,有必要研究测试用例不放回(without replacement)时的随机测试和分割测试,并对它们的测试效果进行比较.本文第1节分析当测试用例放回时,随机测试和分割测试在发现错误能力上的不足,指出它们适合评估软件可靠性.第2节研究测试用例不放回时,随机测试和分割测试在进行 n 次实验后至少发现1个错误的概率公式.第3节利用该公式比较随机测试和分割测试的效果.第4节总结我们的结果并指出其在实践上的应用.

1 测试用例放回时,随机测试和分割测试发现错误的能力极其低下

1.1 基本概念

对于一个程序 P,它的输入域用 D 表示,D 的大小用 d 表示.输入域中引起故障输出的元素称为故障输入(failure-causing inputs),其总数用 $m(o\leqslant m\leqslant d)$ 表示,余下的元素 $c(c=d-m)$ 称为正确输入(correct inputs),则故障率(failure rate)$\theta=m/d$.假设选取的测试用例总数为 n.

分割测试中 D 被分成 $k(\geqslant 2)$ 个子域,用 $D_i(i=1,2,\ldots,k)$ 表示.每个子域包含 d_i 个元素,其中有 m_i 个故障输出,$c_i(=d_i-m_i)$ 个正确输出;故障率 $\theta_i=m_i/d_i$;选取的测试用例为 n_i.只有在测试用例总数相等的情况下,比较随机测试和分割测试的效果才是公平的,因此,$n=\sum_{i=1}^{k}n_i$.

假设所有子域都是相互独立的,因此,$d=\sum_{i=1}^{k}d_i$,$m=\sum_{i=1}^{k}m_i$,$c=\sum_{i=1}^{k}c_i$.所有测试用例独立选取,并且允许放回,子域是均匀的.这意味着每次选取例子时,出错概率恰为 $\theta_i=m_i/d_i$,当进行 n 次实验以后,至少发现一个错误的概率为

$$P_r=1-(1-\theta)^n \quad \text{(随机测试)},$$

$$P_p=1-\prod_{i=1}^{k}(1-\theta_i)^{n_i} \quad \text{(分割测试)}.$$

1.2 考察随机测试和分割测试发现错误的能力

假设 $d=1000$,$m=1$.当我们进行 $n=1000$ 次测试时,至少发现1个错误的概率为

$$P_r=1-(1-\theta)^n=1-(1-1/1000)^{1000}=0.6323.$$

当 $P_r=0.999$ 时,$n=6904$;当 $P_r=0.918$ 时,$n=2500$;当 $P_r=0.865$ 时,$n=2000$.

进行2 000次、2 500次,甚至高达6 904次实验还不能使发现1个错误成为必然事件,可见其发现错误的能力是非常低的.在用例放回时,至少发现1个错误的概率永远不可能等于1,也就是说,有可能永远发现不了这1个错误.

对于分割测试,测试用例放回的缺点更为明显.分割测试的目的是发现错误集中的子域.假设分割策略很成功,有以下例子.

例1:$d_1=2$,$m_1=1$,对该子域进行两次测试,$P_p=1-\prod_{i=1}^{k}(1-\theta_i)^{n_i}=1-(1-1/2)^2=0.75$,当 $P_p=0.9$ 时,$n=3.321$.

例2:$d_1=100$,$m_1=1$,$n=100$,$P_p=1-\prod_{i=1}^{k}(1-\theta_i)^{n_i}=1-(1-1/100)^{100}=0.6339$.当 $P_p=0.9$

时,$n=229$.

在例1的情况下,我们进行3次测试还不一定发现两个输入元素中的1个错误;在例2的情况下,我们进行229次测试还不一定发现100个输入元素中的1个错误.可见其发现错误的能力是非常低的.

另一个极端是重复发现某一错误输入.Weyuker和Jeng认为输入域D非常大,重复取测试用例的可能性很小.我们则认为,当D非常大而m较小时,利用随机测试要想发现错误,n必须很大,因此重复取测试用例的可能性并不小.尤其是对于分割测试,当获得好的子域D_i,m_i,即D_i不大而m_i较大时,重复取测试用例的可能性大,既会夸大故障率θ(重复取产生故障的例子),也会缩小故障率θ(重复取不产生故障的例子).

1.3 测试用例放回时,随机测试和分割测试适合应用于估计软件可靠性

随机测试假设输入域是均匀的(uniform).当测试用例放回时,每次测试的故障率是$\theta=m/d$.如果进行n_r次测试,发现m_r个错误,则用m_r/n_2的值估计θ的值,即$\theta=m_r/n_r$.软件可靠性则利用Nelson公式为$R=1-m_r/n_r$.因此,可以说随机测试的发现错误能力是一种平均水平,它不是积极主动的发现错误的测试方法.

同样地,分割测试假设每个子域是均匀的.当测试用例放回时,子域每次测试的故障率是$\theta_i=m_i/d_i$.如果进行n_i次测试,发现m_i个错误,则用m_i/n_i的值估计θ_i的值,即$\theta_i=m_i/n_i$.设每个子域选取的概率为p_i,则整个程序的可靠性利用Brown-Lipow公式为$R=\sum_{i=1}^{k}p_i(1-\theta_i)=\sum_{i=1}^{k}p_i(1-m_i/n_i)$.从直觉上来讲,它比随机测试发现错误的能力要强,但如果测试用例分配不合理,发现错误的能力就会不如随机测试.当测试用例放回时,就不利于我们集中精力测试错误集中的小子域,因为重复取例子的概率增大了.下一节我们研究测试用例不放回的情况.它增加了跟踪测试用例的费用,但使得发现错误的能力大大提高.

2 当测试用例不放回时,进行n次测试后,至少发现1个错误的概率比放回时增大

本节用到的所有基本概念与第1节相同,P_{fr},P_{fp},P_{nr},P_{np}分别表示测试用例放回和不放回时的随机测试和分割测试进行n次测试后,至少发现1个错误的概率.当测试用例不放回时,进行n次测试后,至少发现1个错误的概率公式如下:

当$n\leqslant c$时:$P_{nr}=1-\frac{C_{c}^{n}}{C_{d}^{n}}=1-\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-n+1}{d-n+1}$,　当$n>c$时:$P_{nr}=1$;

当$n_i\leqslant c_i$时:$P_{np}=1-\prod_{i=1}^{k}\frac{C_{c_i}^{n_i}}{C_{d_i}^{n_i}}=1-\prod_{i=1}^{k}\frac{c_i}{d_i}\cdot\frac{c_i-1}{d_i-1}\cdot\ldots\cdot\frac{c_i-n_i+1}{d_i-n_i+1}$,　当$n_i>c_i$时:$P_{np}=1$.

这是利用等古典概率知识推导出的两个公式.由前面的叙述可知:

$$P_{fr}=1-(1-\theta)^n=1-\left(\frac{c}{d}\right)^n,$$

$$P_{fp}=1-\prod_{i=1}^{k}(1-\theta_i)^{n_i}=1-\prod_{i=1}^{k}\left(\frac{c_i}{d_i}\right)^{n_i}.$$

因为　$\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-n+1}{d-n+1}<\left(\frac{c}{k}\right)^n$,$\prod_{i=1}^{k}\frac{c_i}{d_i}\cdot\frac{c_i-1}{d_i-1}\cdot\ldots\cdot\frac{c_i-n_i+1}{d_i-n_i+1}<\prod_{i=1}^{k}\left(\frac{c_i}{d_i}\right)^{n_i}$,

所以　$P_{nr}>P_{fr}$,$P_{np}>P_{fp}$.

也就是说,当测试用例不放回时,发现错误的能力提高了.最明显的是,当测试次数超过子域内的正确输入时,至少发现1个错误的概率为1.从下一节开始,我们比较当测试用例不放回时,随机测试和分割测试发现错误的能力.

3 当测试用例不放回时比较随机测试和分割测试

3.1 分割测试的最好和最坏情况

结论1. 当分割最好时,即某子域仅包含引起错误的输入时,$P_{np}=1>P_{nr}$.

证明:假设 $d_i=m_i$,则 $c_i=0,\frac{c_i}{d_i}=0$,

$$P_{np}=1-\prod_{i=1}^{k}\frac{C_{c_i}^{n_i}}{C_{d_i}^{n_i}}=1-\prod_{i=1}^{k}\frac{c_i}{d_i}\cdot\frac{c_i-1}{d_i-1}\cdot\ldots\cdot\frac{c_i-n_i+1}{d_i-n_i+1}=1,$$

$P_{nr}<1$,即 $P_{np}=1>P_{nr}$. □

在另一种极端情况下,即分割最坏情况,如同 Weyuker 和 Jeng 所描述的,假设 $1\leqslant n_i\leqslant d_i$,$d$ 远大于 k 和 m. 当 $n_1=\ldots=n_k=1,d_1=\ldots=d_{k-1}=1$ 时,$d_k=d-n+1$,所有的错误都集中在 D_k,此时 $P_{np}=1-\frac{c_k}{d_k}=1-\frac{c-n+1}{d-n+1}$. 上式有一个暗含关系:$n=k$.

结论2. 在分割测试的最坏情况下,$P_{nr}>P_{np}$.

证明:在分割测试的最坏情况下有 $0<m<d-k+1,d>k,c=d-m>k-1,m>0\Rightarrow d>c$.

$$P_{nr}=1-\frac{C_c^n}{C_d^n}=1-\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-n+1}{d-n+1},$$

$$P_{np}=1-\frac{c_k}{d_k}=1-\frac{c-n+1}{d-n+1}.$$

因为 $$\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-n+1}{d-n+1}<\frac{c-n+1}{d-n+1},$$

所以 $$P_{nr}=1-\frac{C_c^n}{C_d^n}=1-\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-n+1}{d-n+1}>P_{np}=1-\frac{c_k}{d_k}=1-\frac{c-n+1}{d-n+1}.$$ □

3.2 当 $d_1=\ldots=d_k,n_1=\ldots=n_k,m_1=\ldots=m_k$ 时,比较分割测试和随机测试的好坏

结论3. 当 $d_1=\ldots=d_k,n_1=\ldots=n_k,m_1=\ldots=m_k$ 时,$P_{nr}>P_{np}$.

证明:当 $d_1=\ldots=d_k,n_1=\ldots=n_k,m_1=\ldots=m_k$ 时,$d=kd_i,n=kn_i,m=km_i,c=kc_i$.

$$P_{nr}=1-\frac{C_c^n}{C_d^n}=1-\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-n+1}{d-n+1}$$

$$=1-\overbrace{\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-(k-1)}{d-(k-1)}}^{k}\cdot\ldots\cdot\overbrace{\frac{c-(n-k)}{d-(n-k)}\cdot\ldots\cdot\frac{c-(n-1)}{d-(n-1)}}^{k}$$

$$P_{np}=1-\prod_{i=1}^{k}\frac{C_{c_i}^{n_i}}{C_{d_i}^{n_i}}=1-\prod_{i=1}^{k}\frac{c_i}{d_i}\cdot\frac{c_i-1}{d_i-1}\cdot\ldots\cdot\frac{c_i-n_i+1}{d_i-n_i+1}$$

$$=1-\prod_{i=1}^{k}\frac{c/k}{d/k}\cdot\frac{c/k-1}{d/k-1}\cdot\frac{c/k-2}{d/k-2}\cdot\ldots\cdot\frac{c/k-(n/k-1)}{d/k-(n/k-1)}$$

$$-1-\prod_{i=1}^{k}\frac{c}{d}\cdot\frac{c-k}{d-k}\cdot\frac{c-2k}{d-2k}\cdot\ldots\cdot\frac{c-(n-k)}{d-(n-k)}$$

$$=1-\left(\frac{c}{d}\right)^k\cdot\left(\frac{c-k}{d-k}\right)^k\cdot\left(\frac{c-2k}{d-2k}\right)^k\cdot\ldots\cdot\left(\frac{c-(n-k)}{d-(n-k)}\right)^k.$$

因为 $\overbrace{\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-(k-1)}{d-(k-1)}}^{k}\cdot\ldots\cdot\overbrace{\frac{c-(n-k)}{d-(n-k)}\cdot\ldots\cdot\frac{c-(n-1)}{d-(n-1)}}^{k}<\left(\frac{c}{d}\right)^k\cdot\left(\frac{c-k}{d-k}\right)^k\cdot\left(\frac{c-2k}{d-2k}\right)^k\cdot\ldots\left(\frac{c-(n-k)}{d-(n-k)}\right)^k$,

所以 $P_{nr}>P_{np}$. □

可以看出,当测试用例不放回,平均分配子域、测试次数、错误数时,随机测试要好于分割测试.这与测试用例放回时,Weyuker 和 Jeng 以及 Chen 和 Yu 的研究结果不同.

我们认为这与实践相吻合.它说明了分割测试的本质不是平均分配测试,而是为了找出错误集中的区域,然后集中精力测试这样的子域.

3.3 在一般情形下比较随机测试和分割测试

我们通常难以得到分割测试的最好、最坏和平均分割这 3 种情况.分割策略的好坏直接决定分割测试的好坏.对于一般情形,我们得出如下观察结果:

如果在错误集中的子域,我们进行了较多的测试,那么分割测试一般要优于随机测试,即 $P_{np}\geqslant P_{nr}$.

结论 4. 如果 $d_1=\ldots=d_k, m_1=\ldots=m_2, n_1=\ldots=n_{k-1}=0, n_k=n$,那么 $P_{np}>P_{nr}$.

证明:因为 $d=kd_i, m=km_i, c=kc_i$.

$$P_{nr}=1-\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-n-1}{d-n+1},$$

$$P_{np}=1-\frac{c_k}{d_k}\cdot\frac{c_k-1}{d_k-1}\cdot\ldots\cdot\frac{c_k-n+1}{d_k-n+1}=1-\frac{c}{d}\cdot\frac{c-k}{d-k}\cdot\ldots\cdot\frac{c-(n-1)k}{d-(n-1)k},$$

$$\frac{c}{d}\cdot\frac{c-1}{d-1}\cdot\ldots\cdot\frac{c-n+1}{d-n+1}>\frac{c}{d}\cdot\frac{c-k}{d-k}\cdot\ldots\cdot\frac{c-(n-1)k}{d-(n-1)k},$$

所以 $P_{np}>P_{nr}$. □

上面的情况说明,在平分子域和错误输入的情况下,集中精力测试某一子域,分割测试要优于随机测试.此结论在实践上的指导意义是,在测试次数等同的情况下,缩小测试区域更有利于发现错误.缩小测试区域的实践方法是,除去确信无错的区域,如软件复用部分,还要利用错误群集现象.

4 总 结

在测试用例放回时,Weyuker 和 Jeng 以及 Chen 和 Yu 等学者对随机测试和分割测试进行了较为深入的研究.尽管他们认为重复取同一例子的可能性较小,但是,我们认为下面几种情况使得这种可能性较大:(1) 随机测试次数 n 很大.我们对一个商业软件进行测试的时间长达几个月,测试次数达几千次.(2) 分割测试分得较小子域,并且该子域的测试次数较大.(3) 还有一个人为因素.人的思维容易局限于某些方面,这使得在不受鉴别的情况下,重复取同一例子的可能性较大.在测试初期,为了排除错误、提高软件可靠性,如果放回测试用例,那将是愚蠢的行为.

本文在测试用例不放回时对随机测试和分割测试进行了理论上的研究,提出一个完成 n 次测试后至少发现 1 个错误的概率公式.对随机测试和分割测试进行比较得出 4 个结论,前两个结论与 Weyuker 和 Jeng 以及 Chen 和 Yu 等学者的研究成果相同,后两个结论与 Weyuker 和 Jeng 以及 Chen 和 Yu 等学者的研究成果不同.结论 3 指出,当平均分配子域、错误、测试次数时,分割测试还

不如随机测试,分割测试的本质不是平均测试.结论4提出了分割测试优于随机测试的实践方法:缩小测试区域,然后集中精力测试该区域.分割测试还有一个优点,即多个测试小组可以同时进行测试.

在测试用例放回时,随机测试和分割测试适合于估计软件故障率,评估软件可靠性.在测试用例不放回时,随机测试和分割测试虽然使跟踪测试用例的费用有所增加,但却使发现错误的能力提高,适用于软件测试早期.

References:

[1] Duran, J. W., Ntafos, S. C. An evaluation of random testing. IEEE Transactions on Software Engineering, 1984, SE-10: 438~444.

[2] Hamlet, R., Taylor, R. Partition testing does not inspire confidence. IEEE Transactions on Software Engineering, 1990, SE-16: 1402~1411.

[3] Weyuker, E. J., Jeng, B. Analyzing partition testing strategies. IEEE Transactions on Software Engineering, 1991, SE-17: 703~711.

[4] Chen, T. Y., Yu, Y. T. On the relationship between partition and random testing. IEEE Transactions on Software Engineering, 1994, 20(12): 977~980.

Comparing Random Testing with Partition Testing Without Test Cases Replacement*

FANG Mu-yun, ZHAO Bao-hua, QU Yu-gui

(*Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China*)

E-mail: bhzhao@ustc.edu.cn

http://www.ustc.edu.cn/ch/school

Abstract: Many researchers have compared the fault-detecting ability of random testing and partition testing with test cases replacement. Great achievements have been gained. Comparing random testing with partition testing without test cases replacement has not been found in existing documents. However, in practice, especially at the early stage of testing and during module testing, test cases are not replaced. Therefore, it is necessary to compare random with partition testing under this case. In this paper it is done and four findings are proposed. One difference from the results done by Chen and Yu is that when the domain is divided into such subdomains with the same size, the same failures and the same test numbers, partition testing performed worst than random testing. How to form a subdomain with heavy failures by use of all kinds of information and then gives it a large number of tests is essential for partition testing.

Key words: random testing; partition testing; software reliability; software testing

* Received February 29, 2000; accepted August 18, 2000

Supported by the National Natural Science Foundation of China under Grant No. 90104010; the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 2000035802