Tạp chí Tin học và Điều khiển học, T.20, S.3 (2004), 193–204

AN APPROACH TO IMPROVE THE PARTITION TESTING

HARETON LEUNG[1], NGUYEN HOANG PHUONG[2], TRAN NGOC CUONG[2], LE HAI KHOI[2]

[1] *The Hong Kong Polytechnic University, Hung Hom Kowloon, Hong Kong*
[2] *Viện CNTT, Viện Khoa học và Công nghệ Việt Nam*

Abstract. Some methods of software testing, including partition testing and random testing were studied and compared [1, 2]. Based on their studies, in general, the partition testing is better than random testing, but it is not always fitted for all cases. In this paper, we will continue to investigate in what conditions the efficacy of partition testing is performed and we will propose a strategy that makes partition testing always better or at least the same as random testing.

Tóm tắt. Trong lĩnh vực công nghệ phần mềm, kiểm tra sản phẩm là một trong những khâu quan trọng. Có hai phương pháp kiểm tra sản phẩm thông dụng được các nhà nghiên cứu quan tâm, đó là phương pháp kiểm tra theo vùng và kiểm tra ngẫu nhiên. Theo một số nghiên cứu, kiểm tra theo vùng tốt hơn kiểm tra ngẫu nhiên, nhưng điều đó không phải lúc nào cũng đúng cho mọi trường hợp. Bài báo đề xuất chiến lược phân vùng sao cho phương pháp kiểm tra phân vùng luôn tốt hơn hoặc ít nhất là bằng phương pháp kiểm tra ngẫu nhiên các sản phẩm phần mềm.

1. INTRODUCTION

Partition testing is a technique of testing which partitions the program input area into multiple classes of the equivalent values and tests the representative values of each class. Random testing is a technique of testing which does not divide the input domain into subdomains. In this case, the partition consists of one class, namely, the entire domain. Random testing can, therefore, be viewed as a degenerate form of partition testing.

Some comparisons of the fault detection capabilities of partition testing and random testing are as follows:

Hamlet and Taylor [3] and Duran and Ntafos[4] compared experimentally random testing and partition testing and their research results showed that although the partition testing was generally better than random testing at finding bugs for a given number of test cases, the difference in effectiveness was relatively small.

Elaine J. Weyuker and Bingchiang Jeng [2], based on the probability of detecting at least one failure-causing input, found that in some cases partition testing is worth the effort and in other cases partition testing is not.

To continue develop the partition testing analysis of Elaine J. Weyuker and Bingchiang Jeng, we aim to find the way to make partition testing to be a good strategy, which can overcome the existing weakness of the methods [3, 4] for some conditions.

In this paper we propose an approach to improve the partition testing method in some conditions by making a good partition and controlling the selected test cases. The rest of the

paper is organized as follows: The second section reviews some notions of partition testing method and random testing method. The third section presents an approach to improve the partition testing effect. The fourth section presents an approach to a practical support for testing method. The final section discusses limitations of the methods and further research.

2. SOME NOTIONS OF PARTITION AND RANDOM TESTING

Let us review some notions and symbols of partition testing and random testing:

Consider that a program P, with domain D of size d, m points of which produce incorrect output- called that inputs *failure-causing inputs*, and assume that $d >> m$. Let n be the number of test cases selected. And let θ denote the failure rate, the probability that a failure-causing input will be selected as a test case.

In random testing, if it uses a uniform probability distribution of case random selection then $\theta = m/d$. If it is done based on another operation distribution of case random selection then $\theta = \sum_{i=1}^{k} p_i\theta_i$, with assuming that a probability distribution of inputs divides the program inputs into k subdomains and p_i is the probability that randomly chosen test case is from subdomain D_i.

Denote that the probability of finding at least one failure causing input in n randomly selected tests:

$$P_r = 1 - (1 - \theta)^n \tag{1}$$

In partition testing, the domain is divided into k subsets, $D_1, D_2, ..., D_k$, of size $d_1, d_2, ..., d_k$, and failure rate $\theta_1, \theta_2, ..., \theta_k$, respectively. Assume that the number of subdomains is at least two and the subdomains are disjoint. Some reference documents about partition testing can be found in [7–11].

Let m_i denote the number of inputs in subdomain i for which the program produces an incorrect output. In partition testing, the elements of a subdomain are grouped together because it is believed that they are closely related in some essential way, and that any member of the class is a representative as good as any other. Therefore, when selecting members from a subdomain, the distribution is assumed uniform, and $\theta_i = m_i/d_i$.

Let P_p denote the probability of finding at least one failure causing input using partition testing with n_i test cases chosen randomly from each D_i :

$$P_p = 1 - \prod_{i=1}^{k}(1 - \theta_i)^{n_i} \tag{2}$$

When comparing random testing and partition testing, we assume that $n = \sum_{i=1}^{k} n_i$. That is, we are comparing how the two techniques behave with the same number of test cases.

3. APPROACH TO IMPROVE PARTITION TESTING

Why we need to improve partition testing? Recall Weyuker and Jeng's observations [2] in brief:

- P_p is maximized if one subdomain contains only inputs that produce incorrect inputs.

- Partition testing can be better, worse, or the same as random testing, depending on how the partitioning is performed.

Based on Weyuker and Jeng's observations [2], there are some cases in which partition testing is not better than random testing:

Case 1: Weyuker and Jeng's observation 3

P_p is minimized when $n_1 = n_2 = \cdots = n_k = 1$, $\sum_{i=1}^{k=1} d_i = n-1$, $d_k = d-(n-1)$, (k is the number of subdomains) with all m failure - causing inputs in D_k.

Following the formula (2):

$$P_p = 1 - \prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i},$$

$$P_p = 1 - \prod_{i=1}^{k-1} \left(1 - \frac{0}{d_i}\right)^1 \times \left(1 - \frac{m}{d_k}\right)^1 = 1 - \left(1 - \frac{m}{d_k}\right),$$

$$P_p = \frac{m}{d_k} = \frac{m}{d-n+1}.$$

This case is the worst case since for subdomain D_k, the failure rate is minimized by making its size as large as possible (namely $d-n+1$). In most cases, this partitioning will be worse than random testing.

Case 2: Weyuker and Jeng's observation 5.

If $d_1 = d_2 = \cdots = d_k$ and $n_1 = n_2 = \cdots = n_k$ but $p_i \neq d_i/d$ for some $1 \leq i \leq k$, then partition testing can be better, worse, or the same as random testing.

Case 3: Weyuker and Jeng's observation 8.

Let D be partitioned into k subdomains and assume that $n_1 = n_2 = \cdots = n_k = c$ test cases are selected from each subdomain. Then partition testing can be better, worse, or the same as random testing.

For cases 2 and 3, without knowing anything about the distribution of failure-causing inputs, if the partition divides the domain into equal sized subdomains, and we sample them equally, then we will never do worse than random testing. But notice that unless there is a very large number of subdomains (or the number of test cases chosen from each subdomain is large relative to its size), the assumption that $m \ll d$ means that even in the best case, when all failure-causing inputs are grouped into one subdomain, the probability of finding a failure-causing input with partition testing with equal-sized subdomains will be relatively low.

Based on Weyuker and Jeng's results, and the formula (2) there are two important elements that make P_p better or worst than P_r are: how the partitioning is performed and how to control the selected test cases:

Following the formula (2):

$$P_p = 1 - \sum_{i=1}^{k} (1-\theta_1)^{n_i} = 1 - \sum_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i},$$

the partition will make $\frac{m_i}{d_i}$ change its value, and the selected test cases on each subdomain will make n_i change its value.

Then, we can make partition testing worth the effort by controlling these elements. In other words, make the partition performance and the distribution of test cases to be more effective for partition testing.

3.1. Developing a better partition

It is possible to choose the subdomains for a good partition in testing strategy. This partition tensile uses the input conditions of the program and parts the range of the effective values and the range of the ineffective values of each condition to divide the subdomain. Specifically, when it uses specification- context of the program to divide the subdomain, or in other words, it resorts to heuristic techniques.

In the case that subdomains are not of equal size, may be we meet the third Weyuker and Jeng's observation [2]: P_p is minimized when $n_1 = n_2 = \cdots = n_k = 1,\ \sum_{i=1}^{k-1} d_i = n-1,\ d_k = d-(n-1)$, with all m failure-causing inputs in $D_k,\ P_p = \dfrac{m}{d_k} = \dfrac{m}{d-n+1}$, what do we do to enhance this problem?

Observation 1: In this case, let the test cases n_k on D_k greater than 1 ($n_1 = n_2 = \cdots = n_k = c > 1$), it will make P_p take higher value. Because of: $\forall c > 1,\ c \in N$:

$$\left(1 - \frac{m}{d-n+1}\right)^c < \left(1 - \frac{m}{d-n+1}\right)$$

where $\left(1 - \dfrac{m}{d-n+1}\right) < 1$ then $P_p^c = 1 - \left(1 - \dfrac{m}{d-n+1}\right)^c > 1 - \left(1 - \dfrac{m}{d-n+1}\right) = P_p^1$.

Consider the limitation of P_p when c tends to $+\infty$:

$$\lim_{c \to +\infty}(P_p) = \lim_{c \to +\infty}\left(1 - \left(1 - \frac{m}{d-n+1}\right)^c\right) = 1,$$

because of $\left(1 - \dfrac{m}{d-n+1}\right) < 1$.

Then P_p always takes a higher value when the test cases n_k take higher value. It is therefore unnecessary that some subdomains be relatively small and contain only failure-causing inputs, or at least nearly so.

But may be we meet the eighth Weyuker and Jeng's observation [2]: "Let D be partitioned into k subdomains and assume that $n_1 = n_2 = \cdots = n_k = c$ test cases are selected from each subdomain. Then partition testing can be better, worse, or the same as random testing", or the fifth Weyuker and Jeng's observation [2]: "If $d_1 = d_2 = \cdots = d_k$ and $n_1 = n_2 = \cdots = n_k$ but $p_i \neq d_i/d$ for some $1 \leq i \leq k$, then partition testing can be better, worse, or the same as random testing."

It can be enhanced by using the control of test cases: do not let $n_1 = n_2 = \cdots = n_k = c$, it will make P_p higher than P_r on the fault-based measure.

3.2. Control of the selected test cases

We consider a set of test cases as a result of a random process. There are two components concerning with this problem: the probability distribution of selected test cases and the limitation of selected test cases.

Proposition. *IF the probability distributions of selected test cases are arranged sensibly, it will then make partition testing always better than or at least the same as random testing.*

Proof: We have some constraints on P_p and P_r on this problem:

$$\begin{cases} P_r = 1 - (1-\theta)^n \\ \theta = \sum\limits_{i=1}^{k} p_i \theta_i \\ P_p = 1 - \sum\limits_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i} \\ \sum\limits_{i=1}^{k} n_i = n \end{cases} \tag{3}$$

What conditions make $P_p \geq P_r$?

We have:

$$P_p = 1 - \sum_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i} \geq 1 - (1-\theta)^n = P_r$$

$$\Rightarrow P_p = 1 - \sum_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i} \geq 1 - \left(1 - \sum_{i=1}^{k} p_i \theta_i\right)^n = P_r$$

$$\Rightarrow P_p = 1 - \sum_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i} \geq 1 - \left(1 - \sum_{i=1}^{k} p_i \frac{m_i}{d_i}\right)^n = P_r$$

$$\Rightarrow \left(1 - \sum_{i=1}^{k} p_i \frac{m_i}{d_i}\right)^n \geq \prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i}. \tag{4}$$

Assume that there is a distribution of n_i, denotes as $\{p'_i\}$, that means $p'_i = \dfrac{n_i}{n}$ or $n_i = [np'_i]$, symbol [ ] this formula denotes an operation which returns an integer value of n_i. Change the values of n_i in (4), we have:

$$\Rightarrow \left(1 - \sum_{i=1}^{k} p_i \frac{m_i}{d_i}\right)^n \geq \prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{[np'_i]}$$

$$\Rightarrow \left(1 - \sum_{i=1}^{k} p_i \frac{m_i}{d_i}\right)^n \geq \left(\prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{p'_i}\right)^n.$$

Because of $0 \leq \left(1 - \sum\limits_{i=1}^{k} p_i \frac{m_i}{d_i}\right) \leq 1$ and $0 \leq \left(\prod\limits_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{p'_i}\right) \leq 1$ then:

$$\Rightarrow \left(1 - \sum_{i=1}^{k} p_i \frac{m_i}{d_i}\right) \geq \prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{p'_i}. \tag{5}$$

That means if there exists a probability distribution $\{p'_i\}$ of selected test cases make (5) become true, P_p is always greater than or at least equal to P_r.

We will control the selected test cases by applying a probability distribution to the set of selected test cases.

3.2.1. Applying a distribution for the set of selected test cases

Using an adequacy simulation model for an adequacy probability distribution we can find a distribution of selected test cases as we want.

There exist a probability distribution of inputs that the software will actually encounter during it will have been used, in practice that information is frequently not available, particularly before the software has actually been operational for some time. In addition, for many software products the operational distribution changed as the software matures, and it is therefore meaningless to speak of the operational distribution. This distribution is $p_1, p_2, ..., p_k$. That is the distribution which makes $\theta = \sum_{i=1}^{k} p_i\theta_i$ in random testing.

Applying this distribution to our selected test cases, we will take a set of selected test cases, respectively:

$$\begin{cases} n_i = [n.p_i] \\ \sum_{i=1}^{k} n_i = n \end{cases} \quad (6)$$

Where n_i is the number of selected test cases of subdomain D_i. Symbol [ ] in formula (6) denotes an operation which returns an integer value from a real value of $n * p_i$, because n_i is an integer number.

Observation 2: In the case m_i takes a high value on the subdomain D_i which has high probability value (p_i), partition testing is always better than or at least the same as random testing.

Example 1. Assume that domain size is 100, among them 7 of which are failure causing inputs, and 10 test cases are selected ($n = 10$). Let m_i denote a number of inputs in subdomain i for which the program produces an incorrect output. Let k denote the number of subdomains.

Let $k = 10$, the detail of subdomains and its probability is shown in table 1:

Table 1

i	D_i	p_i	m_i
1	1-10	0.3	2
2	11-20	0.2	1
3	21-30	0.1	1
4	31-40	0.1	1
5	41-50	0.05	1
6	51-60	0.05	1
7	61-70	0.05	0
8	71-80	0.05	0
9	81-90	0.05	0
10	91-100	0.05	0

Where i is the order number of subdomain. D_i is the subdomain i-th of $D.p_i$ is the probability that a randomly chosen test case is from subdomain $D_i.m_i$ is the number of inputs in D_i for which the program produces an incorrect output.

In random testing:

With an uniform distribution, follow the formula (1):

$$P_r = 1 - \left(1 - \frac{m}{d}\right)^n = 1 - \left(1 - \frac{7}{100}\right)^{10} = 0.52$$

With distribution $\{p_i\}$, shown in table 1, we have: $\theta = \sum_{i=1}^{10} p_i \times \theta_i$.

$$\theta = 0.3 \times \frac{2}{10} + 0.2 \times \frac{1}{10} + 0.1 \times \frac{1}{10} + 0.1 \times \frac{1}{10} + 0.05 \times \frac{1}{10} + 0.05 \times \frac{1}{10} = 0.11,$$

$$P_r = 1 - (1 - \theta)^{10} = 1 - (1 - 0.11)^{10} = 0.69.$$

In partition testing:

If using the uniform distribution of selected test cases, which means $n_1 = n_2 = \cdots = n_{10} = 1$, as in Weyuker and Jeng's observation 8, P_p will take the value:

$$P_p = 1 - \prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i},$$

$$P_p = 1 - \left(1 - \frac{2}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1,$$

$P_p = 0.57$.

In this case, $P_p = 0.57 < P_r = 0.69$, which means the partition testing is worst than random testing (E. J. Weyuker and B. Jeng's observation 8 [2]).

If we use the distribution $n_i = [n * p_i]$ of selected test cases, the test cases for each subdomain (n_i) are shown in table 2:

Table 2

i	D_i	p_i	m_i	n_i
1	1-10	0.3	2	3
2	11-20	0.2	1	2
3	21-30	0.1	1	1
4	31-40	0.1	1	1
5	41-50	0.05	1	1
6	51-60	0.05	1	0
7	61-70	0.05	0	1
8	71-80	0.05	0	0
9	81-90	0.05	0	1
10	91-100	0.05	0	0

Where i, D_i, p_i, m_i are the same as in table 1, n_i is the number of test cases on D_i. In the table 2, some n_i take value 0 and some take 1 because of constraint (6), and n_i is an integer number.

Then, P_p becomes:

$$P_p = 1 - \prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i},$$

$$P_p = 1 - \left(1 - \frac{2}{10}\right)^3 \times \left(1 - \frac{1}{10}\right)^2 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 = 0.7.$$

In this case, although P_r is higher when it uses the distribution $\{p_i\}$ than it uses the uniform distribution, but $P_p = 0.70 > P_r = 0.69$, which means partition testing is better than random testing. Using the same example with others values of n, and the test cases on each subdomain (n_i) are in table 3:

Table 3

i	D_i	p_i	m_i	$n_i(n=10)$	$n_i(n=20)$	$n_i(n=30)$	$n_i(n=40)$	$n_i(n=50)$
1	1-10	0.3	2	3	6	9	12	15
2	11-20	0.2	1	2	4	6	8	10
3	21-30	0.1	1	1	2	3	4	5
4	31-40	0.1	1	1	2	3	4	5
5	41-50	0.05	1	1	1	2	2	3
6	51-60	0.05	1	0	1	1	2	2
7	61-70	0.05	0	1	1	2	2	3
8	71-80	0.05	0	0	1	1	2	2
9	81-90	0.05	0	1	1	2	2	3
10	91-100	0.05	0	0	1	1	2	2

we have a result:

Table 4

n	10	20	30	40	50
P_r	0.69	0.90	0.9695	0.990	0.9970
P_p	0.70	0.93	0.9720	0.992	0.9975

Chart 1 shows that P_p is higher than P_r graphically.

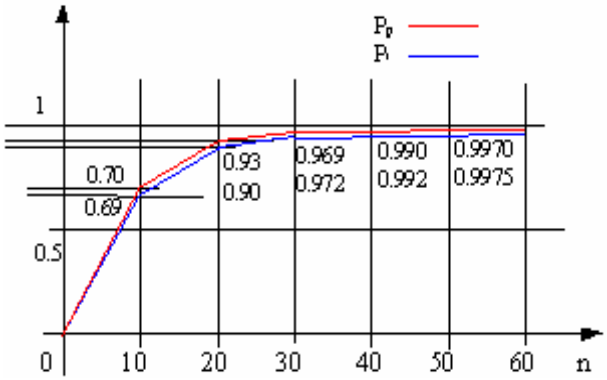

Figure 1

Observation 3: In the case m_i takes a high value on the subdomain which has low probability value, partition testing may be worst than random testing.

Example 2. Assume that the domain size is 100, among them 9 of which are failure causing inputs, and 10 test cases are selected. Let k denote the number of subdomains.

Let $k = 10$, the detail of subdomains and its probability $\{p_i\}$ are shown in table 5:

Table 5

i	D_i	p_i	m_i
1	1-10	0.3	0
2	11-20	0.2	0
3	21-30	0.1	0
4	31-40	0.1	0
5	41-50	0.05	1
6	51-60	0.05	1
7	61-70	0.05	1
8	71-80	0.05	1
9	81-90	0.05	2
10	91-100	0.05	3

In random testing: With this distribution $\{p_i\}$:

$$\theta = \sum_{i=1}^{10} p_i \times \theta_i = 0.05 \times \frac{1}{10} + 0.05 \times \frac{1}{10} + 0.05 \times \frac{1}{10} + 0.05 \times \frac{1}{10} + 0.05 \times \frac{2}{10} + 0.05 \times \frac{3}{10} = 0.045.$$

$$P_r = 1 - (1 - \theta)^{10} = 1 - (1 - 0.045)^{10} = 0.37.$$

In partition testing: The distribution of selected test cases (n_i) is shown in on the table 6:

Table 6

i	D_i	p_i	m_i	n_i
1	1-10	0.3	0	3
2	11-20	0.2	0	2
3	21-30	0.1	0	1
4	31-40	0.1	0	1
5	41-50	0.05	1	1
6	51-60	0.05	1	0
7	61-70	0.05	1	1
8	71-80	0.05	1	0
9	81-90	0.05	2	1
10	91-100	0.05	3	0

Then, P_p is:

$$P_p = 1 - \prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i} = 1 - \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{2}{10}\right)^1 = 0.35.$$

In this case, $P_p < P_r$.

What do we do to improve this problem? We try to find the way, in order $P_p > P_r$.

Observation 4: In the case m_i take a high value on the subdomain which has a low probability value, partition testing is better than random testing if we use a big enough number of test cases base on this simulation.

Example 3. Use the same assumption of example 2, but let $n = 20$.

In random testing:

$$P_r = 1 - (1 - \theta)^{20} = 1 - (1 - 0.045)^{20} = 0.6$$

In partition testing: The distribution of selected test cases is:

Table 7

i	D_i	p_i	m_i	n_i
1	1-10	0.3	0	6
2	11-20	0.2	0	4
3	21-30	0.1	0	2
4	31-40	0.1	0	2
5	41-50	0.05	1	1
6	51-60	0.05	1	1
7	61-70	0.05	1	1
8	71-80	0.05	1	1
9	81-90	0.05	2	1
10	91-100	0.05	3	1

Then, P_p is:

$$\begin{aligned} P_p &= 1 - \prod_{i=1}^{k} \left(1 - \frac{m_i}{d_i}\right)^{n_i} \\ &= 1 - \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{1}{10}\right)^1 \times \left(1 - \frac{2}{10}\right)^1 \times \left(1 - \frac{3}{10}\right)^1 \\ &= 0.63. \end{aligned}$$

In this case, $P_p > P_r$, that mean partition testing is still better than random testing on the failure -based measure. Using this example with other values of n, and the test cases on each subdomain (n_i) are in table 8:

Table 8

i	D_i	p_i	m_i	n_i ($n = 10$)	n_i ($n = 20$)	n_i ($n = 30$)	n_i ($n = 40$)	n_i ($n = 50$)	n_i ($n = 60$)
1	1-10	0.3	2	3	6	9	12	15	18
2	11-20	0.2	1	2	4	6	8	10	12
3	21-30	0.1	1	1	2	3	4	5	6
4	31-40	0.1	1	1	2	3	4	5	6
5	41-50	0.05	1	1	1	2	2	3	4
6	51-60	0.05	1	0	1	1	2	2	4
7	61-70	0.05	0	1	1	2	2	3	4
8	71-80	0.05	0	0	1	1	2	2	4
9	81-90	0.05	0	1	1	2	2	3	4
10	91-100	0.05	0	0	1	1	2	2	4

we have:

Table 9

n	10	20	30	40	50	60
P_r	0.35	0.60	0.75	0.84	0.90	0.94
P_p	0.37	0.63	0.76	0.87	0.94	0.97

This result is represented as a chart below, in this P_p begins higher than P_r from a value of n between 10 and 20:

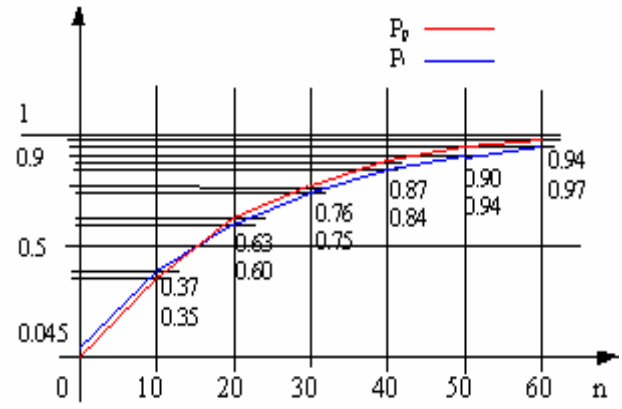

Figure 2

In this chart we show that if there exists a big enough number of selected test cases, it will make P_p greater than P_r when m_i takes a high value on the subdomain which has low probability value.

How to estimate the value of the number of selected test cases that make P_p greater than P_r in the observation 4?

Denote this number as N_{01}, we have some estimations of N_{01} but we still not prove this result.

We show an estimation of N_{01} :

$$N_{01} = \left\lceil \frac{\ln\left(\prod_{i=1}^{k}\left(1-\frac{m_i}{d_i}\right)^{p_i}\right)}{\ln\left(1-\sum_{i=1}^{k} p_i \frac{m_i}{d_i}\right)} \right\rceil \tag{7}$$

In some experiments it has become true, but we have not proved that it becomes true for other cases.

Example 4. Using this estimation for the conditions in example 3:

$$N_{01} = \left\lceil \frac{\ln\left(\prod_{i=1}^{k}\left(1-\frac{m_i}{d_i}\right)^{p_i}\right)}{\ln\left(1-\sum_{i=1}^{k} p_i \frac{m_i}{d_i}\right)} \right\rceil = \left\lceil \frac{\ln(0.3674)}{\ln(0.995)} \right\rceil \approx 21. \tag{7}$$

We have shown the changing effect on P_p while keeping the domain size and total number of inputs which produce an incorrect output constant. By controlling the selected test cases, we will improve partition testing. In the problem of controlling the selected test case, we must reach a limitation of selected test cases. If too many test cases are needed, we may not have enough time to test the software. Assume that the time of testing is not tend to forever.

4. CONCLUSIONS

We have shown an approach that partition testing can be a good strategy. In particular, it depends largely on how the inputs that produce an incorrect output are concentrated within the subdomains defined by the partition.

When analyzing the partition testing based on the probability of detecting at least one failure-causing inputs, or failure-based measure, we have approached a way that makes partition testing strategy to be really effective and therefore worth doing. It has clearly to perform substantially in an adequa way and control the set of selected test cases. By using an adequacy probability distribution of selected test cases and choosing an adequate number of selected test cases for partition testing, the partition testing will be better or at least as effective as random testing.

REFERENCES

[1] Debra J. Richardson and Lori A. Clarke, Partition Analysis: A method combining testing and verification, *IEEE Transactions on Software Engineering* **SE-11** (12) (1983).

[2] Elaine J. Weyuker and Bingchiang Jeng, Analyzing Partition Testing Strategies, *IEEE Transactions on Software Engineering* **17** (7) (1991).

[3] R. Hamlet, R. Taylor, Partition testing dose not inspire confidence, *Proc. 2nd Workshop on Software testing, Verification, and Analysis* July 1988 (206–215).

[4] J. W. Duran, S. C. Ntafos, An evaluation of random testing, *IEEE Trans. Software Eng.* **SE-10** (1984) (438–444).

[5] J. D. Musa, A. Iannino, K. Okumoto, *Software reliability: measurement, prediction, application,* Mc-Crow Hill Co., 1987.

[6] H. T. Nguyen, V. Krenovich, *Application of Continuous Mathematics to Computer Science,* Chapter 3: "Program testing: a problem", Kluwer Academic Press, 1997.

[7] Tsong Yueh Chen, Yuen Tak Yu, On the maximin algorithms for test allocations in partition testing - *Journal of Information and Software Technology* **43** (2001) 97–107.

[8] Tsong Yueh Chen, Yuen Tak Yu, The universal safeness of test allocation strategies for partition testing, *Journal of Information Sciences* **129** (2000) 105–118.

[9] Tsong Yueh Chen, Yuen Tak Yu, On the test allocations for the best lower bound performance of partition testing - *Proceedings of 1998 Australian Software Engineering Conference,* IEEE Computer Society Press, New York (1998) 160–167.

[10] Tsong Yueh Chen, Yuen Tak Yu, Optimal improvement of the lower bound performance of partition testing strategies, *IEEE Proceedings of Software Engineering* **44** (5–6) (1997) 271–278.

Received on October 14 - 2003
Revised on January 14 - 2004