Concise Papers

On the Relationship Between Partition and Random Testing

T. Y. Chen and Y. T. Yu

Abstract— **Weyuker and Jeng have investigated the conditions that affect the performance of partition testing and have compared analytically the fault-detecting ability of partition testing and random testing. This paper extends and generalizes some of their results. We give more general ways of characterizing the worst case for partition testing, along with a precise characterization of when this worst case is as good as random testing. We also find that partition testing is guaranteed to perform at least as well as random testing so long as the number of test cases selected is in proportion to the size of the subdomains.**

Index Terms—**Partition testing, random testing, software testing.**

I. INTRODUCTION

Partition testing is one of the main techniques towards selection of test data. For partition testing, the program's input domain is divided into subsets, called *subdomains*, and one or more representatives from each subdomain are selected to test the program. Path coverage is a typical example of partition testing.

Random testing can be viewed as a degenerate form of partition testing in the sense that there is only one "subdomain," the entire program domain. Since there is no partitioning, random testing does not bear the overhead of partitioning and of keeping track of which subdomains have been tested or not.

Duran and Ntafos [1], and Hamlet and Taylor [2], performed a series of simulations and experiments to compare the effectiveness of partition testing and random testing. Their results were at first surprising to many people. They found that even when partition testing was better than random testing at finding bugs, the difference in effectiveness was marginal. Thus, when the overhead of partitioning is expensive relative to random testing, it is likely that random testing will be more cost effective than partition testing in terms of cost per fault found.

Weyuker and Jeng [3] conducted a formal analysis of partition testing strategies and compared the effectiveness of partition testing and random testing. They found that partition testing can be an excellent testing strategy or a poor one, as compared to random testing. Essentially, they found that the effectiveness of a particular partition testing strategy depends on how well that strategy groups together the *failure-causing inputs*, that is, those inputs which produce incorrect outputs. In particular, partition testing is most successful when the subdomain definitions are fault-based. Their findings were very helpful in explaining the seemingly counter-intuitive results in the empirical analysis of partition testing and random testing by Duran and Ntafos [1] as well as Hamlet and Taylor [2].

Based on and inspired by the work of Weyuker and Jeng [3], this paper is aimed at adding some new results and generalizations. The necessary notation and assumptions are introduced in Section II. In Section III, we quote some of the results obtained by Weyuker and Jeng, and show how they can be extended or generalized. The practical implications and limitations of our results are discussed in Section IV.

Manuscript received February, 1994; revised July, 1994. Recommended by J. D. Gannon.

The authors are with the Department of Computer Science, University of Melbourne, Parkville 3052, Australia.

IEEE Log Number 9406725.

II. NOTATION

Since this paper is mainly a follow-up of Weyuker and Jeng's work [3], we basically follow their notation and recall only those definitions needed in this paper. For details, readers should refer to Weyuker and Jeng [3]. To facilitate cross-reference, their results will be quoted with their original reference numbers and followed by (WJ), such as Observation 2 (WJ).

For any general program P, its input domain will be denoted by D and the size of D is $d > 0$. Elements of D which produce incorrect outputs are known as *failure-causing inputs*, and we shall assume that there are m $(0 \leq m \leq d)$ of them. The remaining c $(= d - m)$ inputs will be called *correct inputs*. The *failure rate*, θ, is defined as $\theta = m/d$. Assume also that the total number of test cases selected is n.

When testing is done by dividing the program domain D into k (≥ 2) subdomains, these subdomains will be denoted by D_i where $i = 1, 2, \ldots, k$. Each subdomain will have size d_i, contain m_i $(0 \leq m_i \leq d_i)$ failure-causing inputs and c_i $(= d_i - m_i)$ correct inputs, and have failure rate $\theta_i = m_i/d_i$. The number of test cases selected from it is n_i (≥ 1). Note that, by definition, $\sum_{i=1}^{k} n_i = n$. Since in general testing performance increases with the number of test cases, we have assumed that partition testing and random testing have the *same* number of test cases when they are compared.

As in Weyuker and Jeng, all subdomains are assumed to be *disjoint*. Therefore, $\sum_{i=1}^{k} d_i = d$, $\sum_{i=1}^{k} m_i = m$ and $\sum_{i=1}^{k} c_i = c$. All random selections are also assumed to be *independent, with replacement*, and based on a *uniform distribution*. This means that when a test case is selected from D (D_i), the probability that it is a failure-causing input will be exactly θ (θ_i). We are interested in the probability that at least one failure-causing input is found when all test cases have been selected and executed. This will be denoted by P_r for random testing, and P_p for partition testing. They are defined by the following formulas:

$$P_r = 1 - (1 - \theta)^n$$

and

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

III. RELATIONSHIP BETWEEN PARTITION AND RANDOM TESTING

In this section, we first look at the best and worst cases for partition testing as identified by Weyuker and Jeng. We shall show that the worst case of partition testing can be more generally characterised in two different ways. The fault-detecting ability of random testing is then compared to that of the worst case of partition testing in a more precise way.

Next we look at the observation by Weyuker and Jeng which provides a sufficient condition to guarantee that partition testing is at least as good as random testing. Common partition testing strategies

do not usually satisfy such a condition. We find, however, that the condition can be easily relaxed so that it can be satisfied by any partitioning method. We conclude this section by comparing, using some examples, the partition scheme that corresponds to Weyuker and Jeng's condition to the scheme that corresponds to our condition.

A. Best and Worst Cases of Partition Testing

Weyuker and Jeng have shown that partition testing performs best in the following situation.

Observation 1: (WJ) P_p is maximized if one subdomain contains only inputs that produce incorrect outputs.

An obvious consequence of this observation is that partition testing is most effective when fault-based strategies are used to partition the input domain. However, it is difficult to define good fault-based strategies that correspond to real faults.

At the other extreme, the worst case of partition testing is identified by Weyuker and Jeng, under the assumptions that $1 \leq n_i \leq d_i, d \gg k$ and $d \gg m$, as follows.

Observation 3: (WJ) P_p is minimized when $n_1 = \cdots = n_k = 1$, $\sum_{i=1}^{k-1} d_i = n - 1$, and $d_k = d - n + 1$, with all of the m failure-causing inputs in D_k. In this case, $P_p = m/d_k = m/(d - n + 1)$.

As obvious, there is also an implicit assumption that $n = k$, or else some of the n_i may be greater than 1. However, we have found that the minimum value of P_p can still be equal to $m/d_k = m/(d-n+1)$ even without the assumptions $n = k, d \gg k$ and $d \gg m$. Intuitively, we follow the same idea that only one test case is to be selected from D_k, and we try to minimize θ_k by making the size of D_k as large as possible. Following is a generalized form of Observation 3.

First Generalized Form of Observation 3: Assume that $n \geq k \geq 2$, $d \geq k$, $0 \leq m \leq d - n + 1$ (that is, there are at least $n - 1$ correct inputs), and $n_i \leq d_i$ for all i. Then P_p is minimized when $\sum_{i=1}^{k-1} n_i = n - 1$, $d_i = n_i$ for $i = 1, \ldots, k - 1$, $n_k = 1$ and $d_k = d - n + 1$, with all of the m failure-causing inputs in D_k. In this case, $P_p = m/d_k = m/(d - n + 1)$.

A further generalization can be achieved even with the removal of the constraint that $n_i \leq d_i$ for all i, resulting in the following form:

Second Generalized Form of Observation 3: Assume that $n \geq k \geq 2$, $d \geq k$, $0 \leq m \leq d - k + 1$ (that is, there are at least $k - 1$ correct inputs). Then P_p is minimized when $\sum_{i=1}^{k-1} n_i = n - 1$, $d_1 = \cdots = d_{k-1} = 1$, $n_k = 1$ and $d_k = d - k + 1$, with all of the m failure-causing inputs in D_k. In this case, $P_p = m/d_k = m/(d - k + 1)$.

In this second generalized form, it may happen that there is a subdomain D_i with the number of test cases selected (n_i) greater than the number of elements in that subdomain (d_i). Although this rarely happens in practice, it is still possible in theory, as we have assumed that the selections could be done with replacement.

Since $k \leq n$, $d - k + 1 \geq d - n + 1$. Therefore, the value of P_p in the Second Generalized Form is smaller than that of Observation 3 (WJ).

Weyuker and Jeng have argued without proof that the worst case of partition testing will, in most cases, be worse than random testing, based on the following intuition that

"in the random testing case, the tester gets n attempts at finding a bug, with each try having a likelihood of m/d of finding one. In this partitioning, the tester gets only one try, with almost the same failure rate, $m/(d - n + 1)$."

In fact, the worst case of partition testing would be better than random testing only when m is equal to a certain value. The following lemma is needed before we present this property and its formal proof.

Lemma 1: Let d, k, m and n be integers such that $n \geq k \geq 2$, $d \geq k$, $0 \leq m \leq d - k + 1$. Then $1 - \left(1 - \frac{m}{d}\right)^n \geq \frac{m}{d-k+1}$ unless $m = d - k + 1$.

Proof: Let $A = 1 - \left(1 - \frac{m}{d}\right)^n$ and $B = \frac{m}{d-k+1}$. We divide the proof into cases.

Case 1: When $m = d - k + 1$, $B = 1 > A$.

Case 2: When $m = 0, B = 0 = A$.

Case 3: When $d = k$, we must have $0 \leq m \leq 1$. If $m = 0$, this reduces to case 2 above. If $m = 1$, this reduces to case 1 above.

Case 4: Suppose $0 < m < d - k + 1$ and $d > k$. Then $c = d - m > k - 1$. Now $m > 0 \Rightarrow d > c$. Also,

$$\begin{aligned} A - B &= 1 - \left(1 - \frac{m}{d}\right)^n - \frac{m}{d-k+1} \\ &= \frac{c-k+1}{d-k+1} - \left(\frac{c}{d}\right)^n \\ &= \frac{d^n(c-k+1) - c^n(d-k+1)}{d^n(d-k+1)}. \end{aligned}$$

The denominator is clearly positive. Let N be the numerator of the above fraction. Then

$$\begin{aligned} N &= d^n(c-k+1) - c^n(d-k+1) \\ &= cd(d^{n-1} - c^{n-1}) + (-k+1)(d^n - c^n) \\ &= (d-c)\left[cd\sum_{i=0}^{n-2} c^i d^{n-2-i} + (-k+1) \times \sum_{i=0}^{n-1} c^i d^{n-1-i}\right] \\ &= (d-c)\left[c\sum_{i=0}^{n-2} c^i d^{n-1-i} - k\sum_{i=0}^{n-1} c^i d^{n-1-i} + \sum_{i=0}^{n-1} c^i d^{n-1-i}\right] \\ &> (d-c)\left[(c-k)\sum_{i=0}^{n-2} c^i d^{n-1-i} - kc^{n-1} + \sum_{i=0}^{n-1} c^{n-1}\right] \text{ since } d > c > 0 \\ &= (d-c)\left[(c-k)\sum_{i=0}^{n-2} c^i d^{n-1-i} + (n-k)c^{n-1}\right] \\ &\geq 0 \qquad \text{since } c \geq k \text{ and } n \geq k. \end{aligned}$$

Hence we have $N > 0$, and therefore $A > B$. □

Proposition 1:

1) The minimized values of P_p in Observation 3 (WJ) and the First Generalized Form of Observation 3 are not greater than P_r unless $m = d - n + 1$ (that is, unless there are exactly $n - 1$ correct inputs).
2) The minimized value of P_p in the Second Generalized Form of Observation 3 is not greater than P_r unless $m = d - k + 1$ (that is, unless there are exactly $k - 1$ correct inputs).

Proof: The second part of Proposition 1 follows directly from Lemma 1, and the first part is simply a special case of Lemma 1 when $k = n$. □

As a reminder, it is obvious that when $m \geq d - k + 1$, $P_p = 1$ irrespective of how the domain is partitioned.

Proposition 2: If $d \gg m, d \gg n$ and $d \gg k$, then $P_r/P_{p,\text{worst}} = O(n)$, where $P_{p,\text{worst}}$ is used to denote the value of P_p in the worst case.

Proof: Assuming that d is large relative to m, n and k, then only case 4 in the proof of Lemma 1 is valid. Thus, we have $P_r = 1 - (1 - m/d)^n \approx 1 - (1 - mn/d) = mn/d$, and $P_{p,\text{worst}} = m/(d - k + 1) \approx m/(d - k) = (m/d)(1 - k/d)^{-1} \approx$

$(m/d)(1 + k/d)$. Therefore, $P_r/P_{p,\text{worst}} \approx n/(1 + k/d) = O(n)$. In fact, the constant factor is almost equal to 1. □

Note that if only one random test is performed, $P_r = 1 - (1 - m/d) = m/d$ which is only slightly less than $P_{p,\text{worst}}$. Proposition 2 shows that, in the worst case, the error revealing capability of n test cases from partition testing is almost the same as that of one random test case!

B. A Sufficient Condition for Partition Testing to be Better Than Random Testing

In general, we do not have *a priori* precise information about the distribution of the failure-causing inputs so as to assess the performance of partition testing. Despite the absence of information about the distribution of failure-causing inputs, Weyuker and Jeng did find a way of partitioning the domain that is guaranteed to be not worse than random testing.

Observation 4:(WJ) If $d_1 = \cdots = d_k$ and $n_1 = \cdots = n_k$, then $P_p \geq P_r$. If, in addition, the failure-causing inputs are equally divided among the subdomains, so that $m_1 = \cdots = m_k$, then $P_p = P_r$.

In other words, we will never do worse than random testing by having equal-sized subdomains and equal number of test data. This result gives a *sufficient* condition of partition testing to be better than random testing. However, when we divide the program input domain according to the commonly accepted criteria, it is extremely unusual for the subdivisions to be of equal size. An obvious but not intuitively appealing criterion is *equal size partition*, that is dividing the input domain into subdomains of equal sizes. But then this method of partitioning is not fault-based. While we can indeed guard against its worst case behavior by doing so, we have no way of guaranteeing that such a partition is an effective one as it is not fault-based.

In fact, a constant ratio of the number of test cases over the size of the subdomains is sufficent to guarantee that $P_p \geq P_r$. Since equal-sized subdomains and equal number of test data imply a constant ratio between them, Weyuker and Jeng's result is a special case of our result (as stated as Proposition 3 later).

Furthermore, when constant ratios of n_i and d_i replace equalities, the corresponding part for the result about the sufficient condition of $P_p = P_r$ should then be

"If, in addition, the failure rates of all subdomains are the same, so that $m_1/d_1 = \cdots = m_k/d_k$, then $P_p = P_r$."

This is exactly Observation 7 (WJ), which is stated below for easy reference:

Observation 7: (WJ) Assume that $\theta_1 = \cdots = \theta_k$. Then $\theta = \theta_i$ and $P_p = P_r$.

Before we present our result, we need to prove the following lemma first as it is needed for the proof of the new result.

Lemma 2: If $k = 2$ and $n_1/d_1 = n_2/d_2$, then $P_p \geq P_r$.

Proof: Let $n_1/n_2 = d_1/d_2 = r$, and $\delta = \theta - \theta_1$. Then $\theta_2 - \theta = m_2/d_2 - (m_1 + m_2)/d = \frac{m_2 d_1 - m_1 d_2}{d_2 d} = \frac{(m_2 d_1 - m_1 d_2 + m_1 d_1 - m_1 d_1)r}{d_1 d} = \frac{r(m d_1 - m_1 d)}{d_1 d} = r(m/d - m_1/d_1) = r(\theta - \theta_1) = r\delta$. Hence we have,

$$\begin{aligned} 1 - P_p &= (1 - \theta_1)^{n_1}(1 - \theta_2)^{n_2} \\ &= [1 - (\theta - \delta)]^{rn_2}[1 - (\theta + r\delta)]^{n_2} \\ &= [(1 - \theta + \delta)^r(1 - \theta - r\delta)]^{n_2} \end{aligned}$$

Let $f(\delta) = (\sigma + \delta)^r(\sigma - r\delta)$, where $\sigma = 1 - \theta$. Without loss of generality, we shall assume $r \geq 1$. Then

$$\begin{aligned} f'(\delta) &= r(\sigma + \delta)^{r-1}(\sigma - r\delta) + (\sigma + \delta)^r(-r) \\ &= r(\sigma + \delta)^{r-1}[(\sigma - r\delta) - (\sigma + \delta)] \\ &= -r(r + 1)\delta(\sigma + \delta)^{r-1} \end{aligned}$$

Now

$$\sigma + \delta = 1 - \theta + \delta = 1 - (\theta - \delta) = 1 - \theta_1.$$

If $\theta_1 = 1$, then $P_p = 1$ and obviously $P_p \geq P_r$.

Assume that $\theta_1 < 1$ so that $\sigma + \delta > 0$. Then we have $f'(0) = 0$, $f'(\delta) < 0$ for any $\delta > 0$ and $f'(\delta) > 0$ for any $\delta < 0$. Therefore, $f(\delta)$ attains its maximum at $\delta = 0$. That is, for any δ, $f(\delta) \leq f(0) = \sigma^{r+1}$. Hence $1 - P_p = [f(\delta)]^{n_2} \leq [f(0)]^{n_2} \leq [(1-\theta)^{r+1}]^{n_2} = (1-\theta)^{rn_2+n_2} = (1-\theta)^{n_1+n_2} = (1-\theta)^n = 1 - P_r$ and so $P_p \geq P_r$. □

We now state and prove our generalized result.

Proposition 3: If $k \geq 2$ and $n_1/d_1 = \cdots = n_k/d_k$, then $P_p \geq P_r$.

Proof: This is done by induction on k.

Basis step: $k = 2$

This follows immediately from Lemma 2.

Induction step:

Suppose we now have a program domain D which is partitioned into $k + 1$ disjoint subdomains $D_1, \ldots, D_{k+1}$, and that $n_1/d_1 = \cdots = n_{k+1}/d_{k+1} = r$. Let $D' = \bigcup_{i=1}^{k} D_i$, where $1 - P_p' = \prod_{i=1}^{k}\left(1 - \frac{m_i}{d_i}\right)^{n_i}$. By induction assumption, we have $1 - P_p' \leq 1 - P_r'$, where $1 - P_r' = \left(1 - \frac{m'}{d'}\right)^{n'}$, $m' = \sum_{i=1}^{k} m_i$, $d' = \sum_{i=1}^{k} d_i$, and $n' = \sum_{i=1}^{k} n_i$. Now,

$$\begin{aligned} 1 - P_p &= \left[\prod_{i=1}^{k}\left(1 - \frac{m_i}{d_i}\right)^{n_i}\right]\left(1 - \frac{m_{k+1}}{d_{k+1}}\right)^{n_{k+1}} \\ &= (1 - P_p')\left(1 - \frac{m_{k+1}}{d_{k+1}}\right)^{n_{k+1}} \\ &\leq (1 - P_r')\left(1 - \frac{m_{k+1}}{d_{k+1}}\right)^{n_{k+1}} \\ &= \left(1 - \frac{m'}{d'}\right)^{n'}\left(1 - \frac{m_{k+1}}{d_{k+1}}\right)^{n_{k+1}}. \end{aligned}$$

D can be regarded as being partitioned into two disjoint subdomains D' and D_{k+1}. Then $m = m' + m_{k+1}$, $d = d' + d_{k+1}$ and $n = n' + n_{k+1}$. Obviously, $n'/d' = r = n_{k+1}/d_{k+1}$. Thus, by Lemma 2,

$$\left(1 - \frac{m'}{d'}\right)^{n'}\left(1 - \frac{m_{k+1}}{d_{k+1}}\right)^{n_{k+1}} \leq \left(1 - \frac{m}{d}\right)^{n}.$$

Hence $1 - P_p \leq 1 - P_r$, and $P_p \geq P_r$. □

Thus, in order to guarantee that partition testing is better than random testing, it is only necessary to have the number of selected test data proportional to its corresponding partition's size, even if the partitions are of different sizes. Since $d_1 = \cdots = d_k$ and $n_1 = \cdots = n_k$ imply $n_1/d_1 = \cdots = n_k/d_k$, Observation 4 (WJ) is a special case of Proposition 3.

C. Comparing the Two Partition Schemes

It is also of great interest to compare the values of P_p for the partition method of $d_1 = \cdots = d_k$ and $n_1 = \cdots = n_k$, and the partition method of $n_1/d_1 = \cdots = n_k/d_k$. As before, the total number of test data for these two methods are assumed to be the same. The following examples show that none of these two methods is always the best, regardless of whether the number of partitions for the first method is the same as, greater than or less than that for the second method.

Example 1: Let $d = 300, k = 3, m = 10$ and $n = 6$. Consider the four cases shown below. Here the partitions are equal in size in Case 1.1 and different in the other cases.

Case	d_1, n_1, m_1	d_2, n_2, m_2	d_3, n_3, m_3	P_p
1.1	100, 2, 10	100, 2, 0	100, 2, 0	0.19
1.2	50, 1, 10	100, 2, 0	150, 3, 0	0.2
1.3	50, 1, 0	100, 2, 10	150, 3, 0	0.19
1.4	50, 1, 0	100, 2, 0	150, 3, 10	0.187

Example 2: Let $d = 300, m = 10$ and $n = 6$. Consider the three cases shown below. Here the partitions are equal in size in Case 2.1 and different in the other cases.

Case	k	d_1, n_1, m_1	d_2, n_2, m_2	d_3, n_3, m_3	P_p
2.1	3	100, 2, 10	100, 2, 0	100, 2, 0	0.19
2.2	2	50, 1, 10	250, 5, 0	-	0.2
2.3	2	50, 1, 0	250, 5, 10	-	0.185

Example 3: Let $d = 300, m = 10$ and $n = 6$. Consider the three cases shown below. Here the partitions are equal in size in Case 3.1 and different in the other cases.

Case	k	d_1, n_1, m_1	d_2, n_2, m_2	d_3, n_3, m_3	P_p
3.1	2	150, 3, 10	150, 3, 0	-	0.187
3.2	3	50, 1, 10	50, 1, 0	200, 4, 0	0.2
3.3	3	50, 1, 0	50, 1, 0	200, 4, 10	0.185

Thus, we have the following proposition.

Proposition 4: The partition method of equal d_i's and equal n_i's (as described in Observation 4 (WJ)) can be better, worse or the same as the partition method of equal (n_i/d_i)'s (as described in Proposition 3).

IV. CONCLUSION

In this paper, we have extended and generalized some of the results by Weyuker and Jeng [3]. We have given more general characterizations of the worst case for partition testing, along with a precise characterization of when this worst case is as good as random testing. The most important result of our study is to extend Observation 4 (WJ) into a theoretically more general and also a practically more applicable form. As long as we select the number of test cases proportional to the size of the subdomains, partition testing is guaranteed to be not worse than random testing.

In practice, most partition testing strategies divide the program domain into unequal-sized subdomains. If the sizes of the subdomains are known, then by Proposition 3, we know how the test cases should be distributed so that the partition method would not be worse than random testing. In many cases, however, the exact sizes of the subdomains may not be easily known. However, it is sufficient to know the ratio of the subdomain sizes in order to be able to determine the distribution of test cases. Even in the absence of the knowledge of such a ratio, we can still estimate the subdomain sizes by using the Monte-Carlo method, that is, by randomly selecting elements of the program domain and calculating the ratio of the number of elements falling into the subdomains.

As an application of Observation 4 (WJ), Weyuker and Jeng [3] pointed out that once a partition has been made, it is possible to algorithmically construct a refinement to the partition that is guaranteed not to decrease P_p. In their algorithm, a given subdomain should be further subdivided into equal sized subdomains and an equal number of test cases should be selected from each resulting subdomain. It follows from Observation 4 (WJ) that the value of P_p for the refined partition is no worse than the original value of P_p. Since Proposition 3 is an extension of Observation 4 (WJ), their algorithm can be generalized accordingly. A given subdomain may be further subdivided into subdomains whose sizes are not necessarily equal, but the number of test cases selected should be proportional to the size of the resulting subdomains. The value of P_p for partitions refined in this way is also guaranteed to be not worse than the original value of P_p.

There is, however, a practical consideration that affects the applicability of Proposition 3. The exact ratios of the subdomain sizes may not be reducible to the ratios of small integers, thereby rendering the total number of test cases too large. For example, when $k = 2, d_1 = 503$ and $d_2 = 1000$, then we would require $n_1 : n_2 = 503 : 1000$. Obviously, the smallest possible integers satisfying this equality are 503 and 1000, and so we would need $n \geq 1503$. Thus, the total number of test cases is at least the same as the number of elements in the entire program domain! In this example, an approximation may be achieved by making $n_1 : n_2 = 1 : 2$. Further work has to be done to investigate the effect when only approximate ratios of the subdomain sizes are used.

A similar problem occurs when some subdomains are small relative to the rest. For example, when $k = 5, d_1 = 18, d_2 = 20, d_3 = 22, d_4 = 2400$ and $d_5 = 3000$, the exact ratio of the sizes of the subdomains is $9 : 10 : 11 : 1200 : 1500$. According to Proposition 3, the number of test cases needed is at least 2730. Suppose now we cannot afford to execute this number of tests. There are two possible ways of reducing the number of test cases. One way is to assign one test case to each of the small subdomains and select test cases for the rest proportionately. So if we are prepared to use not more than 100 test cases, the number of test cases allocated to the subdomains would be 1, 1, 1, 40 and 50 respectively. Another way is to combine the small subdomains into one single subdomain. In this example, the first three subdomains would be combined and the sizes of the new subdomains would be 60, 2400 and 3000, and their ratio would be $1 : 40 : 50$. Again, the effects of these two approaches have yet to be investigated.

V. ACKNOWLEDGMENT

This work was inspired by the work of E. J. Weyuker and B. Jeng to whom the authors would like to express their gratitudes. The authors would like to thank F. T. Chan and M. F. Lau for their invaluable discussions.

REFERENCES

[1] J. W. Duran and S. C. Ntafos, "An evaluation of random testing," *IEEE Trans. Software Eng.*, vol. SE-10, pp. 438–444, July 1984.

[2] R. Hamlet and R. Taylor, "Partition testing does not inspire confidence," *IEEE Trans. Software Eng.*, vol. SE-16, pp. 1402–1411, Dec. 1990.

[3] E. J. Weyuker and B. Jeng, "Analyzing partition testing strategies," *IEEE Trans. Software Eng.*, vol. SE-17, pp. 703–711, July 1991.