Title	Proportional sampling strategy: A compendium and some insights
Author(s)	Chen, TY; Tse, TH; Yu, YT
Citation	Journal Of Systems And Software, 2001, v. 58 n. 1, p. 65-81
Issued Date	2001
URL	http://hdl.handle.net/10722/48423
Rights	Creative Commons: Attribution 3.0 Hong Kong License

To appear in *Journal of Systems and Software* **58** (1) (2001)

Proportional Sampling Strategy: a Compendium and some Insights

T.Y. Chen [a, *, †], T.H. Tse [b], Y.T. Yu [c]

[a] *School of Information Technology, Swinburne University of Technology, Hawthorn 3122, Australia. Email:* tychen@it.swin.edu.au

[b] *Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong. Email:* thtse@cs.hku.hk

[c] *Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon Tong, Hong Kong. Email:* csytyu@cityu.edu.hk

Abstract

There have been numerous studies on the effectiveness of partition and random testing. In particular, the proportional sampling strategy has been proved, under certain conditions, to be the only form of partition testing that outperforms random testing regardless of where the failure-causing inputs are. This paper provides an integrated synthesis and overview of our recent studies on the proportional sampling strategy and its related work. Through this synthesis, we offer a perspective that properly interprets the results obtained so far, and present some of the interesting issues involved and new insights obtained during the course of this research.

Keywords: Software testing, random testing, partition testing, proportional sampling

† Corresponding author. All correspondences should be sent to T.Y. Chen at School of Information Technology, Swinburne University of Technology, Hawthorn 3122, Australia. Tel.: +61-3-9214-4369. Fax: +61-3-9819-0823. Email: tychen@it.swin.edu.au

1 Introduction

There have been numerous studies in the last two decades on the effectiveness of the two major testing approaches in software testing: partition testing and random testing (Duran and Ntafos, 1984; Hamlet and Taylor, 1990; Weyuker and Jeng, 1991; Tsoukalas et al., 1993; Chen and Yu, 1994; Gutjahr, 1999). In particular, the proportional sampling (PS) strategy has been proved, under certain conditions, to be the only form of partition testing that outperforms random testing regardless of where the failure-causing inputs are, in terms of the probability of detecting at least one fault (Chen and Yu, 1994; Chen and Yu, 1996c; Chen and Yu, 2000).

This paper aims at providing an integrated synthesis and an informative overview of our studies on the fault-detecting effectiveness of partition and random testing, and in particular the proportional sampling strategy. Each of these studies has been done with a different focus and emphasis. Gradually, a rich body of knowledge has emerged and evolved as each individual study builds on the knowledge of earlier results. Through the synthesis in this paper, we hope to offer a holistic perspective and set the context for a comprehensive interpretation of the results that have been obtained so far. Furthermore, such a synthesis is also useful for highlighting some of the interesting issues involved and the insights obtained during the course of this research.

Section 2 sets the stage for discussion by introducing the background, outlining the scope of this paper, the terminologies and major assumptions used in the formal model under study. Section 3 summarizes the previous related empirical and analytical studies. Section 4 presents an overview and synthesis of our work on the proportional sampling strategy, discusses its applicability and clarifies issues that were recently misinterpreted. Section 5 discusses the interesting research issues involved and the insights obtained in the further generalisation and relaxation of some major assumptions of the formal model. Section 6 summarizes other related work. Section 7 concludes this paper.

2 Preliminaries

2.1 Background

Input partitioning has been considered as the "natural solution to the two fundamental testing problems of systematic method and test volume" (Hamlet and Taylor, 1990). Partition testing strategies divide the program's input domain (which is the set of all possible inputs) into subdomains, from each of which one or more test cases are selected for execution. These subdomains are formed from some pre-defined criteria, usually based on the program specification or implementation. For example the category-partition method (Ostrand and Balcer, 1988) divides the input domain according to different aspects identified from the specification to be relevant for the purpose of testing, whereas in path coverage testing (Myers, 1979), a subdomain corresponds to all inputs that execute a certain path in the program.

The motivation underpinning the use of partition testing is that the information from the specification or the program code will be useful to construct subdomains that are *homogeneous* (Hamlet and Taylor, 1990; Weyuker and Jeng, 1991) or *revealing* (Weyuker and Ostrand, 1980), in the sense that either all members of a subdomain will cause the program to succeed or all cause it to fail. If so, one representative from each subdomain will be sufficient to reveal the program faults. Unfortunately, as Weyuker and Jeng (1991) have noted, it is extremely unusual for every subdomain to be truly homogeneous. In practice, subdomains have to be sampled often enough to improve the chance of detecting failures (Hamlet and Taylor, 1990).

In contrast to the systematic approach of partition testing, random testing simply selects test cases from the entire input domain randomly and independently (Hamlet, 1994). Random testing makes minimal use of the information from the specification or the program code. Its advantages are that it is intuitively simple, does not bear the overhead of input domain subdivision, and allows statistical quantitative estimation of the program's reliability.

The origin of the debate on partition testing versus random testing can be dated back to the strong disagreement of their values between two classic books on software testing and reliability (Thayer et al., 1978; Myers, 1979). Myers (1979) described random testing as "probably the poorest [test case design] methodology" whereas Thayer et al. (1978) recommended the use of random testing at the final testing of a program. Motivated by their contrary opinions, Duran and Ntafos (1984) performed the now well-known simulations and experiments for evaluating the effectiveness of the two testing approaches. Their results were considered by many to be surprising and counter-intuitive. Other researchers have subsequently performed more simulations and experiments (Loo and Tsai, 1988; Hamlet and Taylor, 1990) that essentially confirmed the original findings of Duran and Ntafos (1984).

In search for theoretical explanations of the empirical results, Weyuker and Jeng (1991) conducted the first analytical analysis of partition testing by means of a formal model. Based on this model, their work has been substantially extended and generalized by many subsequent studies. In the rest of this section, we shall state the scope of this paper, introduce the notation and terminology used throughout this paper, and briefly discuss the assumptions that underlie the formal model. A more comprehensive account of the justification and validity of these assumptions can be found in (Duran and Ntafos, 1984; Hamlet and Taylor, 1990; Weyuker and Jeng, 1991; Chen and Yu, 1996a). As will be seen, most of these assumptions have also been used in one way or another in many other related studies. The effects of relaxing some of these assumptions have been investigated in some recent studies, which will be discussed in Section 5.

2.2 Scope of this paper

Testing serves a variety of purposes, of which two are considered fundamental (Li and Malaiya, 1994; Bache, 1997; Hamlet, 1997; Frankl et al., 1998). The first is the *detection of faults* for removal. For example, Myers (1976, 1979) defines testing as "the process of executing a program with the intent of finding errors". The terms *debug testing* (Frankl et al., 1998) and *directed testing* (Michael and Voas, 1997) have been used to refer to testing with the purpose of finding bugs. The second is the *assessment* of the software. The most common concern is with the prediction of the program's reliability (Tsoukalas et al., 1993; Li and Malaiya, 1994; Hamlet, 1997; Hierons and Wiper, 1997; Michael and Voas, 1997; Frankl et al., 1998), but other aspects are also of interest, such as the estimation of trustability (Howden and Huang, 1995), failure cost (Gutjahr, 1995) and failure rate (Hierons and Wiper, 1997). This paper focuses on the first purpose of testing, that is, it is primarily concerned with fault-detecting effectiveness.

We limit the scope of our discussion in this paper to the case that all subdomains are disjoint. This is also the case considered in most empirical and analytical studies of partition testing, such as in (Duran and Ntafos, 1984; Loo and Tsai, 1988; Hamlet and Taylor, 1990; Miller et al., 1992; Tsoukalas et al., 1993; Li and Malaiya, 1994; Gutjahr, 1995; Howden and Huang, 1995; Hierons and Wiper, 1997). The disjointness condition is actually not as unduly unrealistic as it appears to be. First, many testing strategies do satisfy this condition. They include the path coverage criteria (Myers, 1979; Beizer, 1990), the partition analysis method (Richardson and Clarke, 1985) and many functional and specification-based testing strategies such as the decision-table method (Goodenough and Gerhart, 1975), the category-partition method (Ostrand and Balcer, 1988), the classification-tree

method (Grochtmann and Grimm, 1993; Chen and Poon, 1997), and others (Vagoun and Hevner, 1997; Nair et al., 1998). Secondly, in principle, overlapping subdomains can be refined to form true partitions, as discussed in (Hamlet and Taylor, 1990; Weyuker and Jeng, 1991). The problem of analysing overlapping subdomains has been known to be very difficult (Hamlet and Taylor, 1990; Weyuker and Jeng, 1991) and some limited progress has been achieved in (Chen and Yu, 1996a). Some work has been done in comparing a few specific testing strategies (with overlapping subdomains) satisfying certain special relations (Frankl and Weyuker, 1993a; Frankl and Weyuker, 1993b), but a more general and more comprehensive analysis remains to be done.

The discussions in this paper are predicated on a complete lack of any information on the failure-causing inputs, even though some non-trivial results have been obtained when assuming otherwise (Weyuker and Jeng, 1991; Chen and Yu, 1996a; Chan et al., 1997; Nair et al., 1998). This is mainly because, in practice, it is very difficult to obtain information on failure-causing inputs with adequate accuracy prior to testing, and also partly because this paper would be substantially lengthened otherwise.

2.3 Notation and terminology

Let D denote the program's input domain, d be the size of D and m be the number of failure-causing inputs in D. The *overall failure rate* of the input domain, denoted by θ, is the proportion of failure-causing inputs in the input domain, that is, $\theta = \frac{m}{d}$. We shall assume that $0 < m < d$ (and hence $0 < \theta < 1$) in order to exclude the trivial cases of a program being perfectly correct or everywhere incorrect.

We shall denote by n the total number of test cases to be selected from the entire input domain. The *overall sampling rate*, denoted by σ, is the proportion of test cases selected from the input domain, that is, $\sigma = \frac{n}{d}$. Normally, because of limited resources, the number of test cases is so small that n is only a small fraction of d. In other words, we would normally expect that $n \ll d$, or $\sigma \ll 1$.

A partition testing strategy is composed of two components: a *partitioning scheme* which divides the input domain D into k subdomains $D_1, D_2, \ldots, D_k$, and a *test allocation scheme* which determines the number of test cases selected from the subdomains. For $i = 1, 2, \ldots, k$, we denote the size of subdomain D_i by d_i, the number of failure-causing inputs in D_i by m_i, and the number of test cases selected from D_i by n_i, respectively. The failure rate and sampling rate of subdomain D_i are then equal to $\theta_i = \frac{m_i}{d_i}$ and $\sigma_i = \frac{n_i}{d_i}$, respectively.

Owing to the disjointness condition, we have $\sum_{i=1}^{k} d_i = d$ and $\sum_{i=1}^{k} m_i = m$. Moreover, for a fair comparison, the same total number of test cases is selected from both partition and random testing, thus giving $\sum_{i=1}^{k} n_i = n$.

2.4 Basic assumptions and effectiveness measures

It is assumed that the selection of test cases is done independently, with replacement and based on a uniform probability distribution. The assumption of selection with replacement is common in most formal models of software testing and reliability, and is made primarily to facilitate the tractability of analysis. In reality, although duplicated test cases are usually not useful and should be avoided, they should occur rarely since usually the number of test cases selected is small relative to the domain

size. In many parts of the analysis, the assumption of low failure rate is actually made explicit in the statement of the results. The effect of selection without replacement was recently addressed in (Leung and Chen, 1999; Leung et al., 2000), and we will discuss it later in Sections 5.2 and 5.3.

Next, we discuss the choice of the uniform distribution in selecting test cases. For selection from subdomains, the reason for this choice is obvious: subdomains are usually so designed that, within the same subdomain, all inputs are considered "equivalent" with no known difference for the purpose of testing (Goodenough and Gerhart, 1975; Weyuker and Ostrand, 1980; Ostrand and Balcer, 1988; Grochtmann and Grimm, 1993). As Hamlet and Taylor (1990) put it, "Were the appropriate distribution skewed in any way it would be a basis for further refinement [of the subdomains]." Note that this does not contradict with the possibility that only some inputs in the subdomain are *actually* failure-causing while others are not. Prior to testing, there is simply no knowledge on which inputs are failure-causing, and hence there is no ground for having a preference of certain inputs over others.

Many authors have argued for the selection of test cases based on the operational distribution, which is the probability distribution of inputs that the program will encounter during its use. Although this choice is appealing and seems ideal, there are considerable difficulties with the operational distribution, both in practice and in theory.

First, the actual operational distribution is frequently not available in practice, especially for brand new software products. Beizer (1990), for instance, argued that "there may be no rational basis for predicting the statistical characteristics of the users. In such cases it isn't even random testing, it's arbitrary testing." Secondly, the operational distribution may change significantly during the use of the software (Hamlet, 1994). Thirdly, a software product is often used by different users with different usage patterns. This is particularly true for re-usable software components, for which predicting their possible usage patterns would be very difficult.

More fundamentally, as Hamlet (1989) put it, "the lack of a distribution exposes a larger conceptual flaw: software quality does not intuitively depend on normal usage." Voas et al. (1996) argued that it is also essential to assess the failure tolerance of the "rare region" of the input domain, in which the inputs are highly *unlikely* to be selected for execution under typical operation. Otherwise, they argued, the software may not be robust enough to perform acceptably in case unusual events occur during operation.

Furthermore, one primary reason of using the operational distribution is that it tends to detect faults that are most likely to occur during operation. In our investigation, the focus is on the fault-detecting ability of the testing strategy without special preference on detecting any particular fault over another, and hence using a uniform distribution seems even more appropriate (Weyuker and Jeng, 1991). Hamlet (1997) argued that, for failure finding, a uniform distribution eases the requirement that ultimate usage be correctly modelled.

The fault-detecting effectiveness of a testing strategy has to be quantified before analytical comparisons can be meaningfully made. The most common measures used are the *probability of detecting at least one failure* (abbreviated as the *P-measure*) (Duran and Ntafos, 1984; Loo and Tsai, 1988; Hamlet and Taylor, 1990; Weyuker and Jeng, 1991; Nair et al., 1998; Gutjahr, 1999), and the *expected number of failures* (abbreviated as the *E-measure*) (Loo and Tsai, 1988; Frankl and Weyuker, 1993b; Chen and Yu, 1996a; Chen and Yu, 1997). We shall first consider these two measures and later introduce other measures as necessary in Section 5.3.

The P-measure is more popular since it is the same as the probability of detecting at least one *fault*. On the other hand, the E-measure has a simpler formula and it can identify the ability of a testing strategy in detecting more than one failure (Chen and Yu, 1996a). Moreover, the two measures bear simple and yet very close relations. In particular, when all the failure rates involved are small enough, the E-measure can be used as a first approximation of the P-measure (Chen and Yu, 1996a).

In this paper, the P-measure is used as the primary metric when evaluating the fault-detecting ability of testing strategies. However, when an exact analysis involving the P-measure is intractable, using the E-measure often helps the understanding of the essential properties or behaviour under investigation.

For random testing, the values of the P-measure and the E-measure are given by $P_r = 1-(1-\theta)^n$ and $E_r = n\theta$, respectively. The corresponding values for partition testing are given by $P_p = 1-\prod_{i=1}^{k}(1-\theta_i)^{n_i}$ and $E_p = \sum_{i=1}^{k} n_i\theta_i$, respectively.

3 Previous work

3.1 Early empirical studies

Intuitively, there are strong reasons to believe that partition testing should perform better than random testing in revealing failures. First, partition testing strategies make use of more information of the specification or the program code in selecting test cases. Secondly, a subdomain usually consists of inputs that "exercise" a particular function or feature of the program or specification. By explicitly requiring at least one test case from every subdomain, partition testing aims at covering all the functions or features in a systematic manner. This is in contrast to random testing in which some functions or features may never get "exercised", thereby failing to detect any faults associated with these functions or features. Thirdly, many partition testing strategies are fault-based, that is, the subdomains are designed in the hope of better revealing certain types of faults. Partition testing represents what Howden and Huang (1995) regard as "intelligent" or "guided" choices that are commonly accepted to be "more likely to reveal faults than random sampling".

Despite all these popular conceptions, the well-known empirical study by Duran and Ntafos (1984) has produced results that are surprising and counterintuitive to many. Essentially, they found that with the same number of test cases, partition testing is slightly more effective in detecting faults than random testing, but the difference is marginal. They concluded that random testing is likely to be more cost-effective in terms of the cost per fault identified, if the overhead of partitioning is substantial.

Considering it "certainly counterintuitive that the best systematic method is little improvement over the worst", Hamlet and Taylor (1990) performed more experiments by varying the relevant parameters, and yet obtained similar results. Independently, Loo and Tsai (1988) conducted even more experiments that simulated a wider range of situations, and again confirmed that random testing performs better under certain conditions, particularly at the early stages of testing.

3.2 A formal analysis

Weyuker and Jeng (1991) are the first to analyse partition testing analytically using a formal model. Their model, on which our work is based, proves to be very useful in exposing the strengths and weaknesses of partition testing that are less obvious and sometimes ignored. It helps to explain why partition testing is *not* unconditionally better. Essentially, the strength of a partition testing strategy lies in its ability to group together the failure-causing inputs. With a well-designed partitioning scheme, there is at least one subdomain with a high density of failure-causing inputs so that the chance of detecting failures will be high. A poor partitioning scheme forms subdomains with low failure rates, and if these subdomains are not sampled frequently enough, the fault-detecting effectiveness can be many times weaker than random testing.

Moreover, Weyuker and Jeng have identified several precise conditions under which partition testing outperforms random testing. Most of the conditions are, however, of limited practical use

as they involve failure rates that are rarely known prior to testing. One notable exception is the observation that partition testing is at least as effective as random testing when all subdomains have the same size and an equal number of test cases are selected from each subdomain. We shall refer to this condition as the "equal-size-and-equal-sampling" condition. We now quote their result, Observation 4 in (Weyuker and Jeng, 1991), formally as follows.

Theorem 1 If $d_1 = \ldots = d_k$ and $n_1 = \ldots = n_k$, then $P_p \geq P_r$. If, in addition, the failure-causing inputs are equally divided among the subdomains, so that $m_1 = \ldots = m_k$, then $P_p = P_r$.

Weyuker and Jeng have further introduced the notion of "subdomain refinement" as follows. Given that n_i test cases are to be selected from a subdomain D_i and $n_i > 1$, it is possible to algorithmically construct a refinement of the subdomain D_i without *a priori* knowledge of faults and yet is guaranteed not to decrease P_p. This can be achieved by subdividing D_i into equal-sized subdomains of D_i and selecting an equal number of test cases from each resulting subdomain. It then follows from Theorem 1 that the value of P_p for the refined partition is no worse than the original value of P_p. Obviously, this refinement process can be continued until $n_i = 1$ for all i without ever diminishing the value of P_p.

4 The proportional sampling strategy

The "equal-size-and-equal-sampling" condition, required by Theorem 1, is obviously unduly restrictive, since most partition testing strategies do not form subdomains of equal sizes. The restriction of equal subdomain sizes has subsequently been relaxed in (Chen and Yu, 1994; Chen and Yu, 1996a), where the proportional sampling (PS) strategy, which is of much wider applicability, was proposed. In this section, we first show that the PS strategy is a universally safe testing strategy, and when it is not feasible, how it can be approximated by the maximin algorithms. We then discuss the merits of partition testing, and when it should be used with proportional sampling. Finally, we shall clarify the issues that were recently misinterpreted.

4.1 A universally safe partition testing strategy

4.1.1 Definitions and the proportional sampling theorem

For a given program to be tested with respect to its specification, the number of failure-causing inputs is fixed and independent of the testing strategy.[1] However, the number and locations of the failure-causing inputs are usually not known before testing. To play safe, it is desirable to know the testing strategies that would perform at least as well as random testing. This motivates the introduction of the notions of "safeness" and "universal safeness" (Chen and Yu, 1996c) as follows.

Definition 1 A partition testing strategy is said to be *safe* for a program if $P_p \geq P_r$ no matter where the failure-causing inputs are located.

Definition 2 A partition testing strategy is said to be *universally safe* if it is safe for every program.

Theorem 1 shows that the "equal-size-and-equal-sampling" condition is sufficient to guarantee a partition testing strategy to be universally safe. As discussed at the beginning of this section, such a condition is unduly restrictive. A more general condition, known as the proportional sampling condition, was defined in (Chan et al., 1996; Chen and Yu, 1996a) as follows:

[1] For the economy of expression, references to the specification of a program will be made implicit hereafter.

Definition 3 A partition testing strategy is said to satisfy the *proportional sampling* (PS) *condition* if $\frac{n_1}{d_1} = \cdots = \frac{n_k}{d_k}$. Such a strategy is known as a *proportional sampling* (PS) *strategy*.

Chen and Yu (1994, 1996a, 1996c) have found that the PS condition also guarantees universal safeness:

Theorem 2 A partition testing strategy is universally safe if it satisfies the PS condition.

Since Theorem 1 requires all subdomains to be of equal sizes while Theorem 2 does not have this constraint, the latter is more useful in practice.

4.1.2 A necessary and sufficient condition for universal safeness

Knowing that the PS condition is *sufficient* to ensure universal safeness of a partition testing strategy, it is natural to ask the following question: "Is there any other universally safe strategy that is independent of or more general than the PS strategy?" This problem was recently solved in (Chen and Yu, 2000), where it was proved that the PS condition is not only sufficient, but also *necessary* for a partition testing strategy to be universal safe.

Theorem 3 A partition testing strategy is universally safe if and only if it satisfies the PS condition.

Note that the PS condition is necessary only if we require partition testing to be safe for *every* program. Indeed, there do exist partition testing strategies that are safe for a particular program under test and yet do not satisfy the PS condition (Chen and Yu, 2000). More precisely, the following necessary condition for a partition testing strategy to be safe was identified in (Chen and Yu, 1996c).

Theorem 4 If a partition testing strategy is safe and $d > (k-1)[m(n-1)-1]$, then for all i, $\left\lfloor \frac{nd_i}{d} \right\rfloor \leq n_i \leq \left\lceil \frac{nd_i}{d} \right\rceil$, where $\lfloor \alpha \rfloor$ denotes the largest integer that does not exceed α, and $\lceil \alpha \rceil$ denotes the smallest integer that is greater than or equal to α.

Note that, when the PS condition holds, the number of test cases selected from subdomain D_i is given by $n_i = \frac{nd_i}{d}$. Thus, when the input domain is large enough with respect to the number of test cases and failure-causing inputs, a safe partition testing strategy must allocate test cases in such a way that does not deviate from the PS condition other than rounding n_i to an integer.

The practical implication of Theorems 3 and 4 is clear. With no prior information whatsoever of the failure-causing inputs, proportional sampling is the only right direction towards the goal of safeness or universal safeness. Any search of other conditions is doomed to be in vain. Whether the goal of safeness is appropriate, though, depends on the context. We defer the discussion of when to recommend the use of the proportional sampling strategy to Section 4.3. Meanwhile, we shall show in the next section that the notion of "subdomain refinement", first introduced by Weyuker and Jeng (1991), can also be modified for the PS condition.

4.1.3 Refinement of the PS strategy

Given n_i test cases to be selected from a subdomain D_i, where $n_i > 1$, we may refine D_i into smaller subdomains such that the n_i test cases are allotted proportionately among the smaller subdomains. By Theorem 2, the resulting testing strategy should be no worse than the original one in terms of the P-measure. This is a more general way of refinement than as suggested in (Weyuker and Jeng, 1991), since the smaller subdomains need not be of the same size.

If we continue this refinement process for all subdomains D_i with $n_i > 1$, then the process ends when every subdomain has one test case selected. Applying such a refinement process to the PS strategy will result in n equal-sized subdomains with $n_i = 1$ for all i. The resulting strategy is known as the *optimally refined proportional sampling* (ORPS) strategy (Chan et al., 1997; Chan et al., 1999). Clearly, the ORPS strategy is a special form of the PS strategy as it also satisfies the PS condition. Alternatively, the ORPS strategy can also be obtained from the "equal-size-and-equal-sampling" strategy through the refinement process suggested in (Weyuker and Jeng, 1991).

The properties regarding the bounds of the P-measure of the ORPS strategy and its comparison with random testing have been studied in (Chan et al., 1997). An empirical study of the effectiveness of the ORPS strategy using seeded errors in published programs has been reported in (Chan et al., 1999). There the improvement of the ORPS strategy over random testing is found to range from as much as 27% to as little as statistically insignificant, with an average of about 7.5%, for the programs under study. This confirms the theoretical result that the ORPS strategy should outperform random testing. Thus, when the overhead of partitioning the input domain into equal-sized subdomains is small, the ORPS strategy may be used on its own right as a preferred strategy to random testing.

4.2 Optimal improvement of lower bound effectiveness

4.2.1 Problem formulation

The PS condition requires that $n_i = \dfrac{nd_i}{d}$ for all i. Since the right hand side is not always an integer but n_i must be integral, it is often necessary to approximate the PS condition rather than to satisfy it strictly. Intuitively, to retain the benefits of universal safeness as much as possible, the approximation should be as close as one can make it. Theorem 4 suggests rounding up or down the fractions $\dfrac{nd_i}{d}$, but it is not entirely clear what the effect will be. Some intuitive guidelines have been provided in (Chan et al., 1996), but it is preferable to have a theoretical basis on which the approximations can be done methodically. More fundamentally, when the PS condition has to be satisfied approximately and not strictly, by Theorem 3, universal safeness can only be approached but never reached. In such situations, random testing will outperform partition testing under some rare and yet unfavourable circumstances, but the difference is expected to be fairly small if the approximation to the PS condition is close enough (Chen and Yu, 1996c; Chen and Yu, 2000). Even so, however, partition testing still retains its merits, as it can be significantly better than random testing when circumstances are favourable (such as when there is a subdomain full of failure-causing inputs). In other words, by following the PS condition even only approximately, partition testing can be much better and will not be much worse than random testing.

An entirely different approach is taken in the work reported in (Chen and Yu, 1997). Rather than trying to achieve safeness "approximately", it examines ways of optimally improving the lower bound effectiveness of partition testing. It makes use of the *maximin* criterion, which is well known in operations research, artificial intelligence and decision theory. The essence of the maximin criterion is to try to achieve the best possible out of the worst-case scenarios.

Technically, the problem is formulated as follows. When the desired partitioning scheme has been chosen, the values of k (the number of subdomains) and $d_1, d_2, \ldots, d_k$ (the subdomain sizes) are fixed. The effectiveness of the testing strategy depends on the failure distribution $\mathbf{m} = (m_1, m_2, \ldots, m_k)$ and the test allocation $\mathbf{n} = (n_1, n_2, \ldots, n_k)$. Let $\phi(\mathbf{m}, \mathbf{n})$ be an effectiveness measure of the partition testing strategy. The maximin effectiveness problem is to find feasible test allocations $\hat{\mathbf{n}}$ that satisfy the following relation:

$$\min_{\mathbf{m}} \phi(\mathbf{m}, \hat{\mathbf{n}}) = \max_{\mathbf{n}} \left[\min_{\mathbf{m}} \phi(\mathbf{m}, \mathbf{n}) \right] . \tag{1}$$

The problem is also called a *MaxiMin E-measure* (MME) problem or a *MaxiMin P-measure* (MMP) problem when the effectiveness measure used is the E-measure and the P-measure, respectively. Detailed analysis of the MME and MMP problems can be found in (Chen and Yu, 1997; Chen and Yu, 2001). Here we highlight some of the main results and discuss their relationships with the PS condition.

4.2.2 The Basic Maximin Algorithm

Basically, the MME problem has been completely solved in (Chen and Yu, 1997; Chen and Yu, 2001) for small values of m. Algorithms have been found for computing the optimal test allocations that are solutions to the MME problem. The most fundamental of these algorithms, called the Basic Maximin Algorithm, is shown in Figure 1. Its main idea is to initially allocate one test case to every subdomain to ensure coverage, and then in each subsequent iteration, allocate additional test cases one by one to the subdomain whose sampling rate is currently the lowest. For the ease of reference, we shall refer to a test allocation computed by the Basic Maximin Algorithm as a *maximin test allocation*.

The Basic Maximin Algorithm

1. Set $n_i := 1$ and $\sigma_i := \dfrac{1}{d_i}$ for $i = 1, 2, \ldots, k$.
2. Set $q := n - k$.
3. While $q > 0$, repeat the following:
   (a) Find j such that $\sigma_j = \min \sigma_i$.
   (b) Set $n_j := n_j + 1$.
   (c) Set $\sigma_j := \sigma_j + \dfrac{1}{d_j}$.
   (d) Set $q := q - 1$.

Figure 1: The Basic Maximin Algorithm

The correctness of this algorithm, which can be stated as follows, has been proved in (Chen and Yu, 1997).

Theorem 5 The Basic Maximin Algorithm always produces solutions to the MME problem, provided that $m \leq \dfrac{d}{n}$ or for all i, $m \leq d_i$.

Regarding the MMP problem, an exact analysis is more difficult because of the complexity of the formula for the P-measure. For fixed m and n, the failure distribution $\mathbf{m}$ that would lead to the lower bound P-measure of $\mathbf{n}$, together with the value of the lower bound, have been identified in (Chen and Yu, 1996b), but an exact solution for the MMP problem remains to be found. Nevertheless, it has been shown in (Chen and Yu, 1996a) that the P-measure possesses properties very similar to those of the E-measure, and in particular the former can be approximated by the latter when all the failure rates involved are small. Thus, when the number of failure-causing inputs is small, it is reasonable to conjecture that the Basic Maximin Algorithm should produce test allocations that are close to the solutions of the MMP problem.

The maximin test allocations possess several interesting properties, some of which are related to the PS strategy. First, the rationale of the Basic Maximin Algorithm is to allocate test cases incrementally to the most "under-sampled" subdomains so far (that is, those currently having the lowest sampling rate), thereby gradually uplifting their sampling rates. With enough test cases, the "under-sampled" subdomains will eventually "catch up" with other subdomains, bringing the sampling rates of all subdomains closer to being equal. If the number of test cases and the subdomain sizes are such that the PS condition can actually be strictly satisfied, the Basic Maximin Algorithm will always produce such a test allocation. Otherwise, the algorithm will produce test allocations that approach the PS condition as much as practically possible. Thus, the algorithm may be used as an elegant and systematic way of approximating the PS condition, and it has the added merit of being founded on a sound theoretical basis, namely the certainty of achieving the optimal lower bound in terms of the E-measure.

Secondly, the Basic Maximin Algorithm is an "incremental" algorithm, even though it has been described as if the total number of test cases, n, is predetermined and fixed. In fact, the algorithm may be interpreted as one that produces solutions to the MME problem for t test cases, where t successively takes the value of k, $k+1$, ..., n, or even beyond. Thus, the tester may decide to execute more or fewer tests than as originally planned, and yet the resulting test allocations are still solutions to the MME problem.

The Basic Maximin Algorithm keeps picking additional test cases from subdomains with the lowest sampling rate so far, until the targeted number of test cases have been selected. In the lack of other information, a sampling rate lower than the average may be treated as an indicator that the subdomain might have been inadequately tested. This suggests a more general guideline that is actually applicable no matter what test allocation strategy has been chosen initially: look for subdomains that have been relatively ignored and test them more thoroughly.

4.2.3 The conservative nature of the PS strategy

With the clearer understanding gained from the relevant studies, it is not too difficult to see that the similarity between the PS condition and the maximin criterion is not entirely coincidental. The PS condition is a necessary and sufficient one for partition testing to be universally safe, that is, better than random testing no matter what the failure distribution is. This certainly includes the worst-case scenario when the failure-causing inputs are adversely spread across the subdomains. Thus, both the maximin criterion and the PS condition attempt to improve the performance of partition testing so that it will not become unduly ineffective, and therefore reflect an attitude that is more conservative.

One may wonder whether it is appropriate to adopt a conservative approach in the context of partition testing. There are ample reasons for such an approach. First, if the overall failure rate is high, or if the failure-causing inputs are located favourably (so that there is a subdomain with a high failure rate), the faults will probably be easily caught with most test allocations anyway. In other

words, there is no need to worry too much about the favourable circumstances. On the other hand, testers do need to be concerned with the possibility of the tests being too ineffective, especially when the cost of failures is high. Ineffective tests not only leave possible faults undetected, but also reduce our confidence on the reliability of the program. Under the premise of a complete lack of information about the actual faults, it appears more advisable to avoid unnecessary risks than to hope wishfully for the best.

Moreover, although the maximin criterion aims principally at improving the worst-case performance, it actually improves the effectiveness of the testing in other ways as well. In a sense, improving the worst case also helps to increase our confidence that the program is correct, particularly when no failures are detected. Indeed, the reliability and dependability of a program are often based on conservative measures rather than optimistic ones. By improving the worst-case performance, the conservative estimate of the reliability or dependability of the program is also raised.

To better understand what this may mean quantitatively, consider the following argument put forward by Nair et al. (1998). Let

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

Then

$$P_p = 1 - (1-\rho)^n \ . \tag{2}$$

Thus, in terms of the effectiveness of fault detection, partition testing is equivalent to random testing as if the program has a failure rate of ρ. Therefore, partition testing is better than, equal to, or worse than random testing according to whether $\rho > \theta$, $\rho = \theta$ or $\rho < \theta$, respectively. Note that the PS strategy always gives $\rho \geq \theta$, and that a solution to the maximin P-measure problem will give the greatest lower bound value of ρ among the different test allocations under a given partitioning scheme.

For n randomly selected test cases from a program with failure rate θ, if no failures are found, the quantity $1-\alpha^{1/n}$ provides an exact $(1-\alpha)$-level upper confidence bound (Nair et al., 1998) for the unknown value of θ. In a similar way, if no failures are found by partition testing, equation (2) shows that $1-\alpha^{1/n}$ is an exact $(1-\alpha)$-level upper confidence bound for the unknown value of ρ. Now, suppose we have two testing strategies A and B, such that their corresponding values of ρ are ρ_A and ρ_B, where $\rho_A \geq \rho_B$. Then we have at least as much confidence in the program based on testing strategy A as we will have using testing strategy B.

Since the PS strategy always has a better P-measure than random testing, the above argument shows that the PS strategy gives a sharper upper confidence bound. Moreover, a test allocation that gives the maximin P-measure will also provide the best upper confidence bound among all test allocations, under the most unfavourable distribution of failure-causing inputs for the chosen partitioning scheme.

4.3 Guidelines on when to use the PS strategy

We now offer some suggestions as to when the PS strategy or the maximin test allocations should be considered. Such a decision would depend on particular situations as detailed as follows:

- *With no preferred partitioning scheme*

  Suppose that the tester has no preferred partitioning scheme in mind and considers the use of random testing. Then, from the point of view of improving the fault-detecting ability, we recommend the use of a proportional sampling strategy that requires as little partitioning overhead

as possible. That is, the tester should attempt to divide the input domain into subdomains across some "natural boundaries" so that the selection of test cases from the subdomains is easy. Test cases should be allocated to satisfy the PS condition if possible or the maximin criterion if not.

If, however, it is preferred to optimally improve the fault-detecting effectiveness, then the ORPS strategy (Chan et al., 1997; Chan et al., 1999) can be used instead of random testing. Recall in Section 4.1.3 that the ORPS strategy divides the input domain into n equal-sized subdomains with one test case selected from each subdomain. Depending on the program, such an equal-size partitioning of the input domain frequently requires only minimal overhead. In such cases, the ORPS strategy not only provides better fault-detecting ability than random testing, but is also cost-effective as well.

- *With pre-determined partitioning scheme but no preferred test allocation scheme*

  When the tester has a pre-determined partitioning scheme to follow but no associated test allocation scheme preferred and no information whatsoever of the failure-causing inputs, we recommend the use of the PS condition if possible, and the use of the maximin test allocations if not. The reasons have already been elaborated in Section 4.2.3. However, the tester should be aware of the conservative nature of the maximin criterion.

- *With pre-determined partitioning scheme and preferred test allocation scheme*

  When the tester has a pre-determined partitioning scheme together with a preferred test allocation scheme, they should not be obliged to change the test allocation scheme to satisfy the PS condition or the maximin criterion if they find these conditions disagreeable. However, we recommend the tester to ensure the validity of the reasons behind their choice, taking into account the possible risk that the resulting tests may be very ineffective in certain circumstances. Although the choices are frequently due to legitimate reasons such as the distribution of failure-causing inputs, sometimes these reasons may be based on ungrounded beliefs or unjustified risk-taking attitude that the tester should dispense with upon careful introspection.

4.4 Concerns and clarifications

The PS strategy has been criticized to be infeasible and unrealistic in (Ntafos, 1998), where a number of queries have been raised. This shows that some of the issues related to the PS strategy have been misinterpreted. We take this opportunity to clarify these issues, explain our position and answer these queries and concerns.

4.4.1 Is PS the best way to perform partition testing?

Ntafos (1998) stated that our results "led [us] to suggest proportional partition testing (where the number of test cases per subdomain is proportional to the probability/size of the subdomain) as the optimum way to do partition testing (Chan et al., 1996; Chen and Yu, 1996c)." In fact, we have never made this suggestion.

In (Chan et al., 1996), the fact that the PS condition is sometimes not strictly satisfiable has been acknowledged, and several intuitive guidelines have been proposed to get around with this difficulty. Some other factors affecting the effectiveness of the PS strategy, such as the pattern of failure-causing inputs, have also been investigated. However, it was *not* suggested in (Chan et al., 1996) that the PS strategy is optimum. It simply provided some practical advice on how to use the PS strategy were it considered desirable.

In (Chen and Yu, 1996c), the focus of investigation was the constraints that a partition testing strategy must satisfy in order to be safe. The paper concluded that a safe strategy must allocate test cases in a way that does not deviate from proportional sampling other than rounding the number of test cases to integers (Theorem 4). Again, it has *not* suggested that the PS strategy, or even a (universally) safe strategy, is optimal.

The fact that the PS strategy has been proved to be at least as effective as random testing in detecting faults does not qualify it to be unconditionally the best strategy. The proper way of interpreting this and other derived results from our point of view has been discussed in (Chan et al., 1996) and in Section 4.3 of this paper. The recommendations are based on a premise of no knowledge on the failure-causing inputs; otherwise other test allocation schemes may be better (Weyuker and Jeng, 1991; Chen and Yu, 1996a; Chan et al., 1997; Nair et al., 1998). Besides, the PS strategy is *not* suitable for a tester who prefers better results despite higher risks.

4.4.2 Does PS require an inordinately large number of test cases?

The feasibility of the PS strategy was queried on the basis that, in some cases, true proportional allocation is only possible with an inordinately large number of test cases. However, it should be clarified that none of the work on the PS strategy advocated to achieve true proportional allocation by increasing the number of test cases. The need for approximating the PS condition when it cannot be satisfied strictly has been acknowledged in almost every related study. Intuitive guidelines have been suggested in (Chan et al., 1996) and the maximin criterion proposed in (Chen and Yu, 1997) to deal with the problem of how the PS condition may be approximated.

4.4.3 Does PS force a "stronger" strategy to imitate a "weaker" one?

It was argued that

(1) the only difference [between random testing and the PS strategy] is that random testing will result in proportional allocation "on average" while [the PS strategy] can force such an allocation;

(2) as the number of test cases increases, partition testing with proportional allocation and random testing will tend to become the same; and,

(3) proportional allocation forces the "stronger" [partition testing] strategy to imitate the "weaker" [random testing] strategy.

Concerning (1), we think that the fundamental differences between partition and random testing has been discarded too lightly. Suffices here to reiterate two of the fundamental differences that are true regardless of the choice of the test allocation. First, the PS strategy forces at least one test case from every subdomain (thereby ensuring the coverage of the software features based on which subdomains are formed), whereas *random testing does not*. This is particularly significant when there exist very small subdomains which may have very little chance of being hit by random test cases, unless the number of test cases is extraordinarily large. Secondly, the strength of a partition testing strategy comes principally from a good partitioning scheme that it comprises, and only secondarily from the test allocation scheme. Note that, for any j,

$$P_p = 1 - \prod_{i=1}^{k} (1-\theta_i)^{n_i} \geq 1 - (1-\theta_j)^{n_j} \geq 1 - (1-\theta_j) = \theta_j \ ,$$

and hence $P_p \geq \max_j \theta_j$. Thus, with a good partitioning scheme that forms at least one subdomain D_i with θ_i close to 1, the testing must be highly effective. This is because, regardless of the allocation scheme being used, it must be true that $P_p \geq \max_j \theta_j \geq \theta_i$. Random testing does *not* have this property.

On the other hand, a poor partitioning scheme spreads the failure-causing inputs across many large subdomains so that all failure rates are small. Together with an arbitrary test allocation, the testing may happen to perform very poorly. The PS strategy provides the remedy that prevents testing from becoming too ineffective.

Random testing and the PS strategy will exhibit the same effectiveness only when the number of test cases increases beyond what is practically affordable. Statement (2), therefore, is unlikely to have much practical significance even though it may be theoretically true.

Concerning (3), we have reservations on the *unqualified* use of the words "stronger" and "weaker" to describe, respectively, partition and random testing. Many empirical and analytical studies have confirmed that partition testing is *not* necessarily stronger than random testing (Duran and Ntafos, 1984; Loo and Tsai, 1988; Hamlet and Taylor, 1990; Weyuker and Jeng, 1991; Chen and Yu, 1994; Chen and Yu, 1996a). On the contrary, it is precisely because partition testing can be weaker than random testing under certain circumstances that the PS condition is proposed to ensure that the former is truly "stronger".

The differences between random and partition testing are more fundamental than as suggested by statements (1) to (3). That is why we disagree that the PS strategy is trying to "imitate" random testing.

4.4.4 Is PS cost effective?

Two simulations have been done in (Ntafos, 1998) to compare between proportional partition testing and random testing for certain ranges of values of n. Results show that the difference $P_p - P_r$ is small in most of the cases studied, and that P_p only increases from 93% to 99.95% when n increases from 100 to 4000. These observations were put forward as evidences that the PS strategy is not cost-effective because "it simply makes no sense to use an additional 3,900 test cases to increase the probability of detecting at least one error from 93% to 99.95%" (Ntafos, 1998).

We would like to point out that these observations are somewhat irrelevant to the use of the PS strategy and can be readily explained by the fact that P_p and P_r are probabilities. Therefore, the claim put forward that the PS strategy is not cost-effective was not properly justified. First, in these simulations, the difference $P_p - P_r$ is necessarily small simply because, in all but a few cases shown there, both P_p and P_r are close to 1. (Note that since P_p and P_r are probabilities, they can never exceed 1.) There is no room for P_p to grow significantly greater than P_r when P_r is, say, greater than 90%. Secondly, because of the law of diminishing marginal returns, once a probability reaches a value close to 1, it will probably take an extraordinary amount of effort to increase it substantially. This is true to both partition and random testing. Moreover, whether doing so is worthwhile or not depends on the context. (In safety-critical systems, the difference between 93% and 99.95% may mean life and death.) In any case, as discussed in Section 4.4.2, none of the work related to the PS strategy has advocated to increase the number of test cases inordinately for whatever purpose. Our position is that, if the PS condition is not strictly satisfiable because n is too small, it may be approximated or the maximin criterion may be used.

It is true that the cost factor has not been considered in many studies that compare random and partition testing. One reason is that the cost issue is rather complex, since it "is hard to measure [and] data are not easy to obtain" (Ntafos, 1998). More fundamentally, most studies are primarily

concerned with the ability of testing strategies to detect faults, which is related to but not the same as cost-effectiveness. (For example, it is not entirely clear what it will mean to include a cost factor into the formula for the probability of detecting at least one fault.) Fault-detecting ability is at least important enough to be investigated on its own right. As Weyuker et al. (1991) have noted, cost considerations should be secondary to effectiveness, because "an ineffective criterion, no matter how cheap, is a poor choice". This is even more true when testing software that is safety-critical.

The above arguments, however, should not be interpreted as to say that the cost issue is not important at all. Rather, they only serve to clarify that there may be legitimate reasons for different studies to focus on different aspects. Indeed, we agree that the cost factor may be included in a formal model when investigating various aspects of testing strategies, such as failure cost estimation (Gutjahr, 1995) and reliability estimation (Tsoukalas et al., 1993). Interestingly, the PS condition is found in (Tsoukalas et al., 1993) to be optimal (with respect to minimizing the weighted sum of failure costs) in the case when all the cost factors are equal.

5 Extensions and interesting issues

The formal study of partition testing strategies not only results in practically applicable guidelines based on solid theoretical bases, but also highlights some interesting issues that provide new insights to further research work. This section discusses some of the recent findings from attempts to further generalize our work on the PS strategy.

5.1 Generalized proportional sampling

Recently, the notion of proportional sampling was re-examined and generalized (Chen et al., 1999; Leung and Chen, 2000). Let $\mathbf{n}$ be a test allocation. If all the subdomain sampling rates are equal, then $\mathbf{n}$ satisfies the PS condition. Otherwise, there are at least two subdomains D_i and D_j, such that the former has a higher sampling rate than the latter. Intuitively, if $\mathbf{n}$ is close to proportional sampling, we would expect the difference between the sampling rates of D_i and D_j to be so small that moving a test case from D_i to D_j would reverse the order of the two sampling rates. This intuition was formalized into a definition of the *generalized proportional sampling* (GPS) *strategy* in (Chen et al., 1999; Leung and Chen, 2000) as follows:

Definition 4 A test distribution $\mathbf{n} = (n_1, n_2, \ldots, n_k)$ is said to satisfy the *generalized proportional sampling* (GPS) *strategy* if, for any i and j such that $\frac{n_i}{d_i} > \frac{n_j}{d_j}$, we have $\frac{n_i - 1}{d_i} < \frac{n_j + 1}{d_j}$. In this case, $\mathbf{n}$ is called a *generalized proportional test distribution* (GPTD).

Obviously, the PS strategy may be considered as a special case of the GPS strategy since a test distribution with equal subdomain sampling rates will, by definition, necessarily satisfy the GPS condition. Unlike the PS strategy, the GPS strategy is always feasible. In fact, if the PS strategy is feasible, then the GPTD will be unique. Otherwise, there may be more than one GPTD, and at least one of them will perform better than random testing unless all the failure rates are the same. These results were found in (Chen et al., 1999; Leung and Chen, 2000), and can be formally stated as follows:

Theorem 6 There exists a unique GPTD if and only if $\frac{nd_i}{d}$ is an integer for all i.

Theorem 7 Suppose that the failure rates of the subdomains are not all the same. Then there must exist a GPTD with a P-measure greater than that of random testing.

However, Theorem 7 does not tell us *which* GPTD outperforms random testing. Intuitively, this GPTD is likely to depend on how the failure-causing inputs are distributed among the subdomains. Even so, since the PS strategy is universally safe and the GPS strategy is an approximation of the PS strategy, it is reasonable to expect the GPS strategy to be better than random testing most of (though not all) the time.

In order to verify the above intuition, a set of simulation experiments were performed in (Chen et al., 1999). Six approximation methods were used to generate feasible test allocations from which only GPTDs were selected. For each approximation method, 40000 GPTDs were obtained, and the average of the P-measures over 100 failure distributions were computed. Results of the experiment show that about 99% of the GPTDs thus found have an average P-measure greater than the corresponding average for random testing. In other words, if we arbitrarily pick a GPTD generated from the methods of approximating the PS strategy, we have a high chance that $P_p \geq P_r$.

5.2 Number of distinct test cases

A major assumption in almost all the formal analyses on the effectiveness of testing is that the selection of test cases is done *with* replacement. This is due to the well-known difficulty of dealing with probabilities related to selection without replacement. Also, it is frequently taken for granted that the effect of this assumption is negligible, particularly when the sampling rates involved are small. In real life, it is not easy to justify the selection of the same test cases more than once. Recently, the effect of the assumption of "selection with replacement" was investigated and hence better understood (Chen et al., 1999; Leung and Chen, 1999; Leung et al., 2000; Leung and Chen, 2000), leading to several interesting results, some of which are discussed here and some in Section 5.3.

When selection is done with replacement, some of the test cases may coincide with one another. We would like to know the expected number of *distinct* test cases that are picked. For random testing in which n elements are selected from the input domain of size d, the expected number of distinct test cases is found in (Leung and Chen, 1999) to be given by $n_d = d - d(1-1/d)^n$. The corresponding number for partition testing is $\sum_{i=1}^{k} \left[d_i - d_i \left(1 - \frac{1}{d_i} \right)^{n_i} \right]$. The following two results show that the PS strategy is in fact optimal with respect to the expected number of distinct test cases selected (Leung and Chen, 1999; Leung and Chen, 2000).

Theorem 8 When there are two or more subdomains, the expected number of distinct test cases selected by the proportional sampling strategy is always greater than that selected by random testing.

Theorem 9 For a given partitioning scheme with two or more subdomains, the expected number of distinct test cases selected by a test allocation scheme that satisfies the PS condition is always greater than that selected by any other test allocation scheme.

Thus, when test cases are selected with replacement, the PS strategy is expected to produce more distinct test cases than (a) random testing or (b) partition testing with any other test allocation techniques. One may wonder whether the greater expected number of distinct test cases of the PS strategy completely accounts for its better fault-detecting ability over random testing. However, it should be noted that the PS strategy has the *same* effectiveness as random testing when all the subdomain failure rates are equal (Weyuker and Jeng, 1991; Chen and Yu, 1996a). Hence, the number of distinct test cases is clearly not the sole reason why the PS strategy is in general more effective. This prompts a further study of precisely what the fault-detecting ability would be in the case of selection without replacement, as will be discussed in the next subsection.

5.3 Selection without replacement

In random testing, when selection is done with replacement, the probability of detecting at least one failure is given by the function $P_r(n) = 1 - (1-\theta)^n$. The value of this probability would be different when selection is done *without* replacement. This new value has been named the *Q-measure* (Leung et al., 2000) to distinguish it from the P-measure. The Q-measure for random testing and partition testing are given by $Q_r(n) = 1 - \dfrac{(d-m)!(d-n)!}{d!(d-m-n)!}$ and $Q_p(\mathbf{n}) = 1 - \prod_{i=1}^{k} \dfrac{(d_i-m_i)!(d_i-n_i)!}{d_i!(d_i-m_i-n_i)!}$, respectively, where $\mathbf{n} = (n_1, n_2, \ldots, n_k)$ donotes the test allocation chosen in partition testing.

It is obvious that $P_r(n) \leq Q_r(n)$, since selection with replacement will never result in more distinct test cases than without replacement. In fact, selecting n_r test cases without replacement always results in n_r *distinct* test cases, whereas selecting n_r test cases with replacement will be expected to produce

$$n_d = d - d\left(1 - \frac{1}{d}\right)^{n_r} \tag{3}$$

distinct test cases. Hence, it would be more interesting to compare the values of $P_r(n_r)$ and $Q_r(n_d)$. On the other hand, we note that the formula for $P_r(n)$ is well defined for any real number n, but that for $Q_r(n)$ is only meaningful when n is an integer. We should therefore start with integral values of n_d and compute the equivalent value of n_r for the purpose of comparing $P_r(n_r)$ and $Q_r(n_d)$. This can be done by changing the subject of equation (3) to give $n_r = \dfrac{\log(1-n_d/d)}{\log(1-1/d)}$.

While most researchers in the past assumed that selection with replacement is a good approximation of selection without replacement, the following result from (Leung et al., 2000) highlights a fundamental difference between the two aspects:

Theorem 10 If n_d distinct test cases are selected randomly, where n_d is a non-negative integer smaller than d, then $Q_r(n_d) \geq P_r(n_r)$, where $n_r = \dfrac{\log(1-n_d/d)}{\log(1-1/d)}$.

Theorem 10 shows that, even after converting n_d to the corresponding $n_r (\leq n_d)$, the fault-detecting effectiveness with n_d distinct test cases is still not worse than n_r test cases selected with replacement. This means that selection without replacement outperforms selection with replacement not solely because the former produces more distinct test cases.

The following observation helps to explain why this is so. When selection is done without replacement, test cases that do not reveal failures would be discarded once executed, and hence the non-failure-causing inputs will eventually be exhausted for sufficiently large n_d, thus guaranteeing the detection of failure-causing inputs afterwards, if any exist. Selection without replacement does not have this advantage.

In order to know how close the values of $Q_r(n_d)$ and $P_r(n_r)$ are, the following ratio is defined:

$$closeness(d) = \min_{m,n_d} \frac{P_r(n_r)}{Q_r(n_d)} \ .$$

Through a series of simulations (Leung et al., 2000), it was found that the above ratio increases with d and is very close to 1 for even modest values of d. For example, $closeness(d) \geq 0.999$ for $d \geq 153$. The following conjecture was thus proposed in (Leung et al., 2000).

Conjecture 1 $P_r(n_r)$ is a close approximation of $Q_r(n_d)$ when d is large.

If this conjecture is true, then the analysis of fault-detecting effectiveness when test cases are selected *without* replacement could be much simplified. It also provides an effectiveness conversion formula for comparing selection with and without replacement. Roughly speaking, $(n_r - n_d) \times$ (cost of executing a test case) provides an estimate of the cost of doing selection with replacement over that without replacement. This cost may be compared with that required to filter out duplicated test cases for deciding whether to perform selection with or without replacement.

5.4 Adaptive random testing

Unlike fault-based testing strategies, random testing makes little use of the information from the specification and program code. In (Chan et al., 1996), it was observed that failure-causing inputs of different faulty programs may form different patterns. For example, for a program with two-dimensional input domain, the failure patterns may be classified into point pattern, strip pattern and block pattern as illustrated in Figures 2, 3 and 4, respectively. In these figures, the bounding box represents the input domain, and the black dots, strip or block indicate the locations of the failure-causing inputs. It was also observed that a partitioning scheme could be more effective in detecting failures of certain patterns than others (Chan et al., 1996).

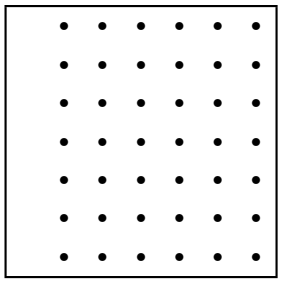

Figure 2: Point Pattern

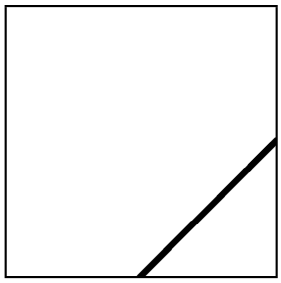

Figure 3: Strip Pattern

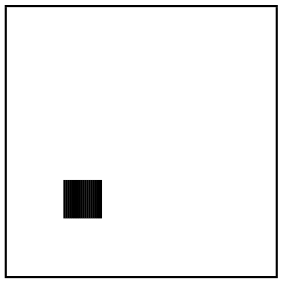

Figure 4: Block Pattern

Except for the point pattern, failure-causing inputs are characterized by being clustered into one or more regions of contiguous inputs. Within each region, the failure-causing inputs are very "dense" and close to one another. Intuitively, a fault in a program is likely to cause the failure of many inputs that are contiguous, and therefore failure-causing inputs that are *not* of point pattern should be common in practice. For example, faults in program predicates, known as *domain errors*, typically result in failure-causing inputs of the strip pattern (White and Cohen, 1980).

When failure-causing inputs indeed form contiguous regions, choosing a test case in the proximity of other test cases does not seem to be a good idea. Instead, common sense dictates that, in such situations, test cases should be spread evenly across the input domain to improve the chance of hitting the failure regions. Based on this intuition, the *adaptive random testing* (ART) method was recently proposed as a form of fault-based random testing (Mak, 1997). One way of implementing ART, as used in (Mak, 1997), is now briefly described as follows.

Basically, two disjoint sets of test cases are maintained, namely the *executed set* and the *candidate set*. Instead of randomly generating a test case at a time, a fixed number of test cases are randomly generated to form the candidate set. The executed set keeps all the test cases that have been executed but not yet revealed any failure. An initial test case is picked at random. If it does not reveal any failure, an initial executed set is formed with this test case as the only element. From the candidate

set, an element that is *farthest* away from the elements of the executed set is selected as the next test case, which is then put into the executed set if it does not reveal a failure. The remaining elements of the candidate set are then discarded and a new candidate set is constructed. The above process is repeated, until a failure is revealed, or until the stopping condition is reached (such as having executed the targeted number of test cases).

Empirical studies using published programs have been carried out to evaluate the effectiveness of ART (Mak, 1997). The results have been very encouraging, showing that in 9 of the 12 programs under study, ART requires much fewer test cases (with up to 50% savings) to detect the first failure. For the remaining 3 programs, ART shows slight improvements over random testing, but the improvement is statistically insignificant. This is expected, as the failures of these 3 programs follow the point pattern.

On closer examination, a comparison of ART with the PS strategy reveals an interesting similarity. Although the PS strategy is a kind of partition testing strategy, the test cases are still randomly selected in each subdomain. In this aspect, there is still an element of randomness associated with the PS strategy. It aims at having a uniform sampling rate across all subdomains, that is, more test cases from larger regions than from smaller ones. Intuitively, enforcing a uniform sampling rate somehow attempts, via the process of partitioning, to evenly spread out the test cases. In this perspective, the PS strategy can be regarded as a form of ART.[2]

Such a similarity between the PS strategy and ART appears striking, at least in two aspects. First, the two strategies were initially proposed for very different reasons. The PS strategy was motivated by the need of providing a universally safe strategy, whereas ART attempts to improve random testing in those situations where the failure-causing inputs tend to cluster. Secondly, no distribution of the failure-causing inputs was assumed when deriving the PS strategy. Yet, it turns out that the PS strategy has the same intrinsic properties as ART, and will therefore be more effective whenever the failure-causing inputs do not follow a point pattern. Thus, in addition to guaranteeing at least as good a P-measure as random testing, there is an inadvertent side effect of improving the effectiveness of testing in situations where failure-causing inputs are clustered.

6 Related work

As mentioned in the introductory section, there have been numerous studies on partition testing strategies since the pioneering work of Duran and Ntafos (1984). In this section, we briefly describe some of these analytical studies.

The PS strategy has been proved to outperform random testing in terms of fault-detecting effectiveness (Chen and Yu, 1994; Chen and Yu, 1996a). Other researchers have proposed test allocation techniques that optimize different aspects of the quality of testing. Tsoukalas et al. (1993) compared random and partition testing by looking at the upper confidence bounds for the cost weighted performance of the two strategies. They showed that, when no failures are detected, a test allocation will be optimal if the number of test cases selected from a subdomain D_i is proportional to the product of the cost of failure for D_i and the probability that a randomly selected input will belong to D_i.

Howden and Huang (1995) addressed the problem of estimating *dependability*, defined as the degree on which the software under test can be considered as dependable. Their approach was based on the concept of *trustability*, defined as the confidence in the absence of faults. Trustability measurement was in turn related to the *detectability* of the testing method, measured by the probability of detecting at least one fault (and hence the same as the P-measure in this paper). They distinguished between two partition testing models: *deterministic* and *statistical*.

[2] Our apologies for the unintended pun.

In the deterministic model, each subdomain is allocated the same number of test cases. In the statistical model, one subdomain is chosen at a time, randomly with equal probabilities. A test case is selected from each chosen subdomain. (Note that in the statistical model, the coverage of all subdomains is not guaranteed.) Howden and Huang found that deterministic partition testing always has a higher or the same detectability when compared with statistical partition testing. They also found a few conditions relating the detectabiliy of random testing and that of either (deterministic or statistical) model of partition testing. Then they further investigated the trustability, which is computed from the detectability, of applying single or multiple testing methods that are each effective for multiple fault classes.

Frankl et al. (1998) examined the relationship between *debug testing* and *operational testing*. These two testing methods were distinguished by their goal to detect faults or to assess reliability, respectively. They argued that "fault" is not a precise idea and hence is not tenable for analytical reviews, and proposed the use of the notion of "failure region", a collection of failure-causing inputs that some change of the program fixes exactly. They considered different scenarios of single or multiple failure regions and debugging with or without subdomains. They suggested that, if the tester has good intuition about the inputs that are likely to be failure-causing and belong to large failure regions, such insights can be used to devise testing strategies. Otherwise, they argued, operational testing may be better. However, they also warned of the risk to trust solely the debuggers' own judgement about their abilities, unless some measurements have been made for comparing the effectiveness of the testing profiles chosen by the tester with that of operational profiles. Such measurement would be a cost-effective step towards better quantitative decision-making.

Li and Malaiya (1994) showed that, for the purpose of defect removal, the optimal test allocation depends on the operational profile and the defect detectability profile of the program. They suggested that test inputs should have a distribution more biased than the operational profile towards the frequently used domains if only limited testing can be afforded, but they should be more evenly distributed if very high reliability is to be achieved through extensive testing.

Gutjahr (1995) presented a method to determine a test allocation that yields an unbiased estimator of the software failure cost with minimum variance. Hierons and Wiper (1997), on the other hand, derived certain conditions under which a functional partitioning scheme can provide a better estimator of the operational failure rate than those based on random testing.

In another study, Gutjahr (1999) modelled the failure rates statistically by random variables. He generalized a result by Weyuker and Jeng (1991) that partition testing is equal in fault-detecting effectiveness to random testing when failure rates are equal. Gutjahr found that, for independent *random* failure rates with equal *expectations*, the fault-detecting probability of partition testing can be up to k times higher than that of random testing, where k is the number of subdomains. This indicates that assumptions about failure rates have to be utilized with caution, as the uncertainty of such information can lead to rather different conclusions from those expected. However, their work clearly suggested that partition testing is well founded, particularly when subdomains are of largely varying sizes and each subdomain is processed by the program in a rather homogeneous way.

One limitation common to all the above studies, including our own work, is that the subdomains have to be disjoint. The problem of overlapping subdomains has been known to be very difficult (Hamlet and Taylor, 1990; Weyuker and Jeng, 1991), and so far little progress has been made. The notable exception is the work of Frankl and Weyuker (1993a). They used essentially the same formal model as that of Weyuker and Jeng (1991) to investigate several types of relationships among different partition testing criteria. They found that a testing criterion that subsumes another does not necessarily have better fault-detecting effectiveness. These results have helped to establish the superiority of some data flow criteria over branch testing (Frankl and Weyuker, 1993b). For a comparison between random

testing and general partition testing strategies with possibly overlapping subdomains, some small but non-trivial results have been reported in (Chen and Yu, 1996a). Clearly, more work on overlapping subdomains has yet to be done.

7 Conclusions

We have shown that the proportional sampling strategy is universally safe and hence solved the problem of whether there are other sufficient conditions for a partition testing strategy to always outperform random testing. Our study of the test allocation problem of partition testing not only yields this practically useful result, but also leads to interesting issues and suggests new insights on intrinsic properties of partition and random testing. The potentials and limitations of these testing strategies are now much better understood. We believe some of the results and observations should have an impact not only on the field of software testing but also on other fields of software engineering and computer science. These results and observations can be summarized as follows.

The striking similarity between the proportional sampling (PS) strategy and adaptive random testing (ART) provides a new perspective for the PS strategy. It should be noted that the PS strategy has actually been proposed as an improvement over random testing in the absence of any information about the failure-causing inputs, while ART was aimed at effectively detecting faults which result in non-point types of failure patterns. Since the PS strategy also possesses the most essential characteristic of ART, namely an "even spread of random test cases", it can be regarded as a form of ART, and hence a fault-based testing strategy. Without the research on ART, it would be counter-intuitive to comprehend the PS strategy as a fault-based testing strategy.

In response to the difficulty of imposing strictly the PS condition in practice, the intuition of the definition for generalized proportional sampling strategy, or equivalently, a generalized notion of uniform sampling rate, has been developed. It is anticipated that the intuition of the generalized uniform sampling rate, despite its simplicity, can be applied in other areas where a uniform sampling rate is targeted but cannot be realized as a result of integral constraints.

The follow-up studies of the the PS strategy also help us better understand the difference between selecting test cases with and without replacement. We have proved that $Q_r(n_d) \geq P_r(n_r)$. Intuitively speaking, this result shows that selection of test cases without replacement is somehow better than selection with replacement even if the repeated test cases had been compensated for by extra test cases. Furthermore, our conjecture that $P_r(n_r)$ is a close approximation of $Q_r(n_d)$ when d is large, will facilitate the future study of the selection of test cases without replacement.

8 Acknowledgements

We take this opportunity to express our acknowledgement to Elaine Weyuker and Bingchiang Jeng, whose work has inspired ours. Our thanks are also due to F.T. Chan, H. Leung, I.K. Mak, S.M. Shen and P.K. Wong, who have collaborated with us on the study of the proportional sampling strategy.

References

Bache, R., 1997. The effect of fault size on testing. Software Testing, Verification and Reliability 7 (3), 139–152.

Beizer, B., 1990. Software Testing Techniques, Van Nostrand Reinhold, New York, USA, second edition.

Chan, F.T., Chen T.Y., Mak I.K., Shen S.M., 1997a. On some properties of the optimally refined proportional sampling strategy. The Computer Journal 40 (4), 194–199.

Chan, F.T., Chen, T.Y., Mak, I.K., Yu, Y.T., 1996. Proportional sampling strategy: guidelines for software testing practitioners. Information and Software Technology 38 (12), 775–782.

Chan, F.T., Chen, T.Y., Tse, T.H., 1997b. On the effectiveness of test case allocation schemes in partition testing. Information and Software Technology 39 (10), 719–726.

Chan, F.T., Mak I.K., Chen T.Y., Shen S.M., 1999. On the effectiveness of the optimally refined proportional sampling strategy. In: Proceedings of International Conference on Software Technology and Engineering Practice (STEP '99), IEEE Computer Society, Los Alamitos, California, pp. 95–104.

Chen, T.Y., Poon, P.L., 1997. Construction of classification trees via the classification-hierarchy table. Information and Software Technology 39 (13), 889–896.

Chen, T.Y., Wong, P.K., Yu, Y.T., 1999. Integrating approximation methods with the generalised proportional sampling strategy. In: Proceedings of 1999 Asia Pacific Software Engineering Conference (APSEC '99), IEEE Computer Society, Los Alamitos, California, pp. 598–605.

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

Chen, T.Y., Yu, Y.T., 1996a. On the expected number of failures detected by subdomain testing and random testing. IEEE Transactions on Software Engineering 22 (2), 109–119.

Chen, T.Y., Yu, Y.T., 1996b. On some characterisation problems of subdomain testing. In: Proceedings of 1996 Ada-Europe International Conference on Reliable Software Technologies (Ada-Europe '96). Lecture Notes in Computer Science 1088, Springer-Verlag, Berlin, pp. 147–158.

Chen, T.Y., Yu, Y.T., 1996c. Constraints for safe partition testing strategies. The Computer Journal 39 (7), 619–625.

Chen, T.Y., Yu, Y.T., 1997. Optimal improvement of the lower bound performance of partition testing strategies. IEE Proceedings — Software Engineering 144 (5–6), 271–278.

Chen, T.Y., Yu, Y.T., 2000. The universal safeness of test allocation strategies for partition testing. Information Sciences 129 (1–4), 105–118.

Chen, T.Y., Yu, Y.T., 2001. On the maximin algorithms for test allocations in partition testing. Information and Software Technology 43 (2), 97–107.

Duran, J.W., Ntafos, S.C., 1984. An evaluation of random testing. IEEE Transactions on Software Engineering 10 (4), 438–444.

Frankl, P.G., Hamlet, D., Littlewood, B., Strigini, L., 1998. Evaluating testing methods by delivered reliability. IEEE Transactions on Software Engineering 24 (8), 586–601.

Frankl, P.G., Weyuker, E.J., 1993a. A formal analysis of the fault-detecting ability of testing methods. IEEE Transactions on Software Engineering 19 (3), 202–213.

Frankl, P.G., Weyuker, E.J., 1993b. Provable improvements on branch testing. IEEE Transactions on Software Engineering 19 (10), 962–975.

Goodenough, J.B., Gerhart, S.L., 1975. Toward a theory of test data selection. IEEE Transactions on Software Engineering 1 (2), 156–173.

Grochtmann, M., Grimm, K., 1993. Classification trees for partition testing. Software Testing, Verification and Reliability 3, 63–82.

Gutjahr, W.J., 1995. Optimal test distributions for software failure cost estimation. IEEE Transactions on Software Engineering 21 (3), 219–228.

Gutjahr, W.J., 1999. Partition testing vs. random testing: the influence of uncertainty. IEEE Transactions on Software Engineering 25 (5), 661–674.

Hamlet, D., 1997. An essay on software testing for quality assurance — editor's introduction. Annals of Software Engineering 4, 1–9.

Hamlet, D., Taylor, R., 1990. Partition testing does not inspire confidence. IEEE Transactions on Software Engineering 16 (12), 1402–1411.

Hamlet, R., 1989. Theoretical comparison of testing methods. In: Proceedings of the 3rd ACM Symposium on Software Testing, Analysis and Verification (TAV '89), ACM Press, New York, pp. 28–37.

Hamlet, R., 1994. Random testing. In: Encyclopedia of Software Engineering, (J. Marciniak, ed.), pp. 970–978. Wiley, New York.

Hierons, R.M., Wiper, M.P., 1997. Estimation of failure rate using random and partition testing. Software Testing, Verification and Reliability 7 (3), 153–164.

Howden, W.E., Huang, Y., 1995. Software trustability analysis. ACM Transactions on Software Engineering and Methodology 4 (1), 36–64.

Leung, H., Chen, T.Y., 1999. A new perspective of the proportional sampling strategy. The Computer Journal 42 (8), 693–698.

Leung, H., Chen, T.Y., 2000. Generalized proportional sampling strategy. Submitted for publication.

Leung, H., Tse, T.H., Chan, F.T., Chen, T.Y., 2000. Test case selection with and without replacement. Information Sciences 129 (1–4), 81–103.

Li, N., Malaiya, Y.K., 1994. On input profile selection for software testing. In: Proceedings of the 5th International Symposium on Software Reliability Engineering (ISSRE '94), IEEE Computer Society, Los Alamitos, California, pp. 196–205.

Loo, P.S., Tsai, W.K., 1988. Random testing revisited. Information and Software Technology 30 (7), 402–417.

Mak, I.K., 1997. On the effectiveness of random testing. Masters thesis, Department of Computer Science, The University of Melbourne, Australia.

Michael, C.C., Voas, J., 1997. The ability of directed tests to predict software quality. Annals of Software Engineering 4, 31–64.

Miller, K.W., Morell, L.J., Noonan, R.E., Park, S.K., Nicol, D.M., Murrill, B.W., Voas, J.M., 1992. Estimating the probability of failure when testing reveals no failures. IEEE Transactions on Software Engineering 18 (1), 33–43.

Myers, G.J., 1976. Software Reliability, Wiley, New York.

Myers, G.J., 1979. The Art of Software Testing, Wiley, New York, second edition.

Nair, V.N., James, D.A., Ehrlich, W.K., Zevallos, J., 1998. A statistical assessment of some software testing strategies and application of experimental design techniques. Statistica Sinica 8 (1), 165–184.

Ntafos, S.C., 1998. On random and partition testing. In: Proceedings of 1998 International Symposium on Software Testing and Analysis (ISSTA '98), ACM Press, New York, pp. 42–48.

Ostrand, T.J., Balcer, M.J., 1988. The category-partition method for specifying and generating functional tests. Communications of the ACM 31 (6), 676–686.

Richardson, D.J., Clarke, L.A., 1985. Partition analysis: a method combining testing and verification. IEEE Transactions on Software Engineering 11 (12), 1477–1490.

Thayer, R.A., Lipow, M., Nelson, E.C., 1978. Software Reliability, North-Holland, Amsterdam, The Netherlands.

Tsoukalas, M.Z., Duran, J.W., Ntafos, S.C., 1993. On some reliability estimation problems in random and partition testing. IEEE Transactions on Software Engineering 19 (7), 687–697.

Vagoun, T., Hevner, A., 1997. Feasible input domain partitioning in software testing: RCS case study. Annals of Software Engineering 4, 159–170.

Voas, J., Charron, F., Miller, K., 1996. Investigating rare-event failure tolerance: reductions in future uncertainty. In: Proceedings of 1st IEEE High-Assurance Systems Engineering Workshop (HASE '96), IEEE Computer Society, Los Alamitos, California, pp. 78–85.

Weyuker, E.J., Jeng, B., 1991. Analyzing partition testing strategies. IEEE Transactions on Software Engineering 17 (7), 703–711.

Weyuker, E.J., Ostrand, T.J., 1980. Theories of program testing and the application of revealing subdomains. IEEE Transactions on Software Engineering 6 (3), 236–246.

Weyuker, E.J., Weiss, S.N., Hamlet, D., 1991. Comparison of program testing strategies. In: Proceedings of the Symposium on Testing, Analysis and Verification (TAV '91), ACM Press, New York, pp. 1–10.

White, L.J., Cohen, E.I., 1980. A domain strategy for computer program testing. IEEE Transactions on Software Engineering 6 (3), 247–257.