Partition Testing vs. Random Testing: The Influence of Uncertainty

Walter J. Gutjahr

Abstract—This paper compares partition testing and random testing on the assumption that program failure rates are not known with certainty before testing and are, therefore, modeled by random variables. It is shown that under uncertainty, partition testing compares more favorably to random testing than suggested by prior investigations concerning the deterministic case: The restriction to failure rates that are known with certainty systematically favors random testing. In particular, we generalize a result by Weyuker and Jeng stating equal fault detection probabilities for partition testing and random testing in the case where the failure rates in the subdomains defined by the partition are equal. It turns out that for independent *random* failure rates with equal *expectation*, the case above is a boundary case (the worst case for partition testing), and the fault detection probability of partition testing can be up to k times higher than that of random testing, where k is the number of subdomains. Also in a related model for dependent failure rates, partition testing turns out to be consistently better than random testing. The dominance can also be verified for the expected (weighted) number of detected faults as an alternative comparison criterion.

Index Terms—Decisions under uncertainty, fault detection, partition testing, program testing, random testing, software testing.

1 Introduction

Few topics in software testing methodology seem to be more controversial than the question whether it is efficient or not to use randomly generated test data (Beizer [1]). While in the '70s experts tended to doubt the efficiency of random testing (cf. Myers' [18] verdict that "probably the poorest [testing] methodology is random input testing"), the judgment on the random testing approach became more positive during the '80s. In theoretical research, a surprising defense of random testing was given by Duran and Ntafos [5], who put the problem on a well-defined formal base by comparing random testing (i.e., selection of test inputs randomly from the whole input domain) to partition testing (i.e., dividing the input domain into nonoverlapping subdomains and selecting one test input from each subdomain). Under different model assumptions, the probabilities of detecting at least one program fault were estimated for both approaches by means of simulation experiments, and compared to each other. Although partition testing usually requires a larger effort in test data generation, it produced only slightly superior results in these experiments. So the outcomes suggested that random testing could possibly be more cost-effective.

Hamlet and Taylor [14] considered these results counterintuitive. Their own simulation experiments, however, essentially confirmed the observations by Duran and Ntafos.

Weyuker and Jeng [24] compared the two testing approaches from an *analytical* point of view. Their results pointed in the same direction again: A clear superiority of partition testing could not be stated; instead, it turned out that, in effectiveness, partition testing can be better, worse or the same as random testing, depending on the "adequacy" of the chosen partition with respect to the location of the failure-causing inputs. In particular, Weyuker and Jeng observed that if the failure rates in the subdomains defined by the partition are equal, then the fault detection probability of partition testing is always the same as that of random testing.

At a first glance, these results, which are undoubtedly correct and formally sound, might confound the practitioner: Many well-known testing methods like statement testing, branch testing, path testing, all-uses, all-p-uses, certain variants of mutation testing, etc., are subdomain-based (cf. [8]), i.e., they rely on a partition of the input domain into nonoverlapping or overlapping subdomains. The construction of test inputs from each of these subdomains usually requires path sensitizing, a task which is tedious in practice and unsolvable by general algorithms (see [27]). Why should one take so much pains to construct such test inputs, if randomly chosen inputs possibly do just as well? Despite their wide-spread use, subdomain-based testing methods seem to lack theoretical justification, compared to the random testing technique.

In the present paper, we want to show that Weyuker and Jeng's analytical comparison approach *can* be used to confirm the superiority of partition testing (or, more generally, subdomain-based testing with more than one subdomain) over random testing. All we have to do is to extend their deterministic model for the failure rates to a probabilistic model. We shall argue that the number of failure-causing inputs in a given subdomain (and, therefore, also the failure rate) is never known to the tester with certainty. Following the traditional statistical paradigm, we shall therefore model this number by a random variable. (A related approach has been chosen in [11], [12].) Since deterministic variables are special cases of random variables, our approach generalizes (in some aspects) the model in [24].

- *W.J. Gutjahr is with the Department of Statistics, Operations Research and Computer Science, University of Vienna, A-1010 Vienna, Austria. E-mail: walter.gutjahr@univie.ac.at.*

Manuscript received 11 July 1995; revised 30 July 1998.
Recommended for acceptance by R. Hamlet.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 101159.

Interestingly enough, it turns out that this probabilistic consideration changes the picture in favor of partition testing methods: In a certain sense, the case where the failure rates are known or deterministic, is the worst case for partition testing. In particular, it will be shown that for equal *expected* failure rates in the subdomains defined by a partition, the partition testing approach *has* in general a higher probability of fault detection than the random testing approach, and that the superiority of partition testing can be drastic. Weyuker and Jeng's results stating equal fault detection probabilities concerns the special case where the variances of the failure rates are zero, i.e., where there is no uncertainty on the failure rates at all.

Of course, we do not assert that the extent of prior knowledge on failure rates influences the relative performance of partition testing and random testing for a fixed given program. It influences, however, our *expectation* of the performance of both methods for the program currently under test. Therefore, also the optimal decision which method to choose necessarily depends on what prior information is available. (As to this point, see the discussion in Section 4.)

In management science, the representation of uncertainty is an issue of great importance (cf. [17]), and it is well-known that decision problems under uncertainty lead to quite different types of optimal solutions than decision problems under certainty. For example, a usual response to uncertainty in portfolio-split decisions is *diversification*. Translated to our problem context, we could say that partition testing methods are strategies of "forced diversification."

The paper is organized as follows: In Section 2, the formal framework for an analytical comparison of subdomain-based testing methods, as it was developed in [24], [26], [8], [9], will be recapitulated. Furthermore, basic arguments for a probabilistic modeling of the failure rates will be presented.

Section 3 contains our main results: It is shown that in a first generalization of Weyuker and Jeng's results, namely a replacement of the deterministic failure rates by independent random variables, equal expected failure rates in the subdomains imply the following bounds: The fault detection probability $\overline{P}_r$ of random testing can vary between a lower bound of order $\overline{P}_p / k$ and the upper bound $\overline{P}_p$, where $\overline{P}_p$ is the fault detection probability of partition testing, and k is the number of subdomains. The upper bound occurs if the failure rates are deterministic. It will be argued that at least in some practical applications, $\overline{P}_r$ can be expected to be nearer to the lower than to the upper bound. Moreover, a further generalization to dependent random variables will be outlined. As alternative criteria for measuring testing effectiveness, the expected number of found faults and the expected weighted number of found faults are considered, and it is shown that also under these criteria, partition testing compares favorably with random testing.

Section 4 discusses premises and scope of our results, considers *advantages* of random testing, and outlines topics for future research.

2 The Formal Framework

2.1 Fault Detection Probabilities

The following definitions are taken from [24], [26], [8], [9]. A *test suite* is a multiset (i.e., a set where elements can occur more than once) of test cases, each of which is a possible input to a program P. The set of all possible inputs of P is the *input domain* D. A *subdomain-based testing method* M determines, to each program P and specification S, a multiset $\mathcal{SD}_M(P, S)$ of subdomains of the input domain D, and requires that from each subdomain contained in $\mathcal{SD}_M(P, S)$, a fixed number m of test cases is selected for the test of P. Most frequently, this number is simply chosen as $m = 1$. For example, in the case of branch testing, $\mathcal{SD}_{\text{branch}}(P, S)$ contains for each decision δ in the program two sets of inputs: the first set $D_{\text{true}}(\delta)$ consists of those inputs that cause δ to evaluate to **`true`** at some time during the execution; the second set $D_{\text{false}}(\delta)$ consists of those inputs that cause δ to evaluate to **`false`** at some time. Branch testing requires that for each decision δ in P, there is both a test case from $D_{true}(\delta)$ and $D_{\text{false}}(\delta)$ in the test suite.

To give another example: For path testing, $\mathcal{SD}_{\text{path}}(P, S)$ contains for each path of the program the set of all inputs that cause the program to follow the respective path.

It is usually assumed that the union of the subdomains contained in $\mathcal{SD}_M(P, S)$ yields the input domain D. However, two different situations may be distinguished: The subdomains in $\mathcal{SD}_M(P, S)$ may be overlapping (as in the case of branch testing), or nonoverlapping (as in the case of path testing). If they do not overlap, we speak of *partition testing*, since then the sets in $\mathcal{SD}_M(P, S)$ form a partition of the input domain in the mathematical sense of the word.

Now, some of the inputs in D may be correctly processed by program P with respect to specification S, while other inputs may be failure-causing. In this article, we shall call the set of failure-causing inputs the *failure domain* of P with respect to S. The main goal of all testing methods is to hit the failure domain (provided that it is not empty) by some test case: If there is a fault in P, at least one of the test cases should unveil it. So testing methods can be compared according to their fault detection ability. In Section 3.2, we shall also use other comparison criteria.

Obviously, a subdomain-based testing method does not *completely* specify a test suite for each given program P and specification S. There is still the freedom *which* test inputs are to be selected from each subdomain. This leads to a definition problem, if testing methods are to be compared on the base of their fault detection abilities. In an early analytical comparison framework for testing methods by Gourlay [10], the problem was solved by a worst-case-consideration: Since the worst case for testing is the case where faults are *not* found, Gourlay took as a representative of each subdomain one of those test inputs (if present) that are *not* contained in the failure domain. He could show that, under this assumption, *subsumes*-relations between testing methods (e.g., branch testing subsumes statement testing) can be translated into assertions on higher fault detection ability.

In the late '80s it was recognized that such worst-case comparisons can be misleading (see Hamlet [15]). Since then, *average-case* comparison models were developed. The basic assumption is the following ([26], [8]: From each subdomain, test cases are selected according to a certain probability distribution. Most investigations are based on the *uniform* distribution: It is supposed that each test input contained in a subdomain has the same probability of being selected for the test. Let us mention that in practice, not only the uniform distribution is applied. One may prefer to use some other strategy, either for practical reasons (easy test case generation) or in view of a further optimization of the testing effectiveness. Throughout this article, however, we confine ourselves exclusively to the uniform model.

Under the assumption above, random testing with inputs selected at equal probabilities becomes a special subdomain-based testing method: Let, in general, $D_1, \ldots, D_k$ be the subdomains contained in $\mathcal{SD}_M(P, S)$, with $D_1 \cup \ldots \cup D_k = D$. In the special case $k = 1$, we have $D_1 = D$, such that (random) selection of a test input from the only existing subdomain is just random selection of *any* test input. In order to draw a clear distinction between random testing and partition testing, we shall always assume in the sequel that for partition testing, the partition $\{D_1, \ldots, D_k\}$ of the input domain contains at least two subsets, i.e., that $k \geq 2$.

For a given partition $\{D_1, \ldots, D_k\}$, one test case per subdomain, and uniform selection from each subdomain, the fault detection probability of *partition testing* is given by (cf. [24])

$$P_p = 1 - \prod_{i=1}^{k} (1 - f_i / d_i). \tag{1}$$

Therein, $d_i = |D_i|$ is the number of elements in subdomain D_i, and $f_i = |D_i \cap D^{fail}|$ is the number of failure-causing inputs in D_i, where D^{fail} denotes the failure domain.

On the other hand, the fault detection probability of *random testing* with k test inputs (selected independently at random from D, with replacement) is given by

$$P_r = 1 - (1 - f / d)^k, \tag{2}$$

where $d = \sum_{i=1}^{k} d_i = |D|$, and $f = \sum_{i=1}^{k} f_i = |D^{fail}|$.

Note that we have chosen the sample size k in the case of random testing equal to the number of subdomains in the case of partition testing. Thus, the numbers of test cases for both methods are equal, which allows a fair comparison.

Equations (1) and (2) can be abbreviated by introducing the *failure rates* $\theta = f/d$ and $\theta_i = f_i / d_i$ in the whole input domain D and in subdomain D_i, respectively. We have then $P_p = 1 - \prod_{i=1}^{k} (1 - \theta_i)$ and $P_r = 1 - (1 - \theta)^k$. A failure rate of θ means that, given the operational profile is the uniform distribution, $100 \cdot \theta$ percent of all inputs lead to program failures. (The reader should notice that other operational profiles entail other failure rates. In [24], however, the authors argue that for a comparison between partition testing and random testing, it is appropriate to consider the case of a uniform profile.)

2.2 Random Failure Rates

In Section 2.1, we have adopted a "modern" viewpoint for analytical comparisons of subdomain-based testing methods, considering the test case selected from a subdomain no longer as deterministic, but as a random variable. In the following, a further generalization step will be suggested: Not only the input from D_i, but also the failure domain D^{fail} should be considered as a random variable.

This approach is simply motivated by the fact that we never *know* the failure domain. If the failure domain was known, testing would be unnecessary. In practice, we do not even know the *failure rate* for the program under test. All that is possibly known are *mean values* (or other statistical characteristics) of failure rates for particular classes of programs, either won explicitly from empirical investigations, or implicitly from the tester's professional experience.

So, a tester can never say:

> *"The program currently to be tested has, with certainty, a failure rate of 2 percent,"*

but he can possibly state:

> *"The given program is one of a larger class of similar programs (with respect to size, type, programming language, environment etc.), a class for which empirical information on failure rates is available, including statistical parameters such as mean value and standard deviation."*

Then, however, the failure rate of the concrete program under test becomes a random variable, the expected value of which can be obtained from the statistics.

We take account of this viewpoint by replacing the deterministic variables f and f_i ($i = 1, \ldots, k$) by random variables F and F_i ($i = 1, \ldots, k$), such that also the failure rates become random. Consequently, instead of the failure rates θ and θ_i, their *expected values* $\overline{\theta}$ and $\overline{\theta}_i$ will be considered as given:

$$\overline{\theta} = E\left(\frac{F}{d}\right), \quad \overline{\theta}_i = E\left(\frac{F_i}{d_i}\right) \quad (i = 1, \ldots, k). \tag{3}$$

Performing this probabilistic generalization, one obtains instead of (1) and (2) the following formulas for the fault detection probability of partition testing, respectively, random testing:

$$\overline{P}_p = E\left(1 - \prod_{i=1}^{k} (1 - F_i / d_i)\right) \tag{4}$$

respectively,

$$\overline{P}_r = E\left(1 - (1 - F / d)^k\right), \tag{5}$$

where the expectation E is taken with respect to the probability distributions of the random variables F_i and F.

Observe that (contrary to P_p and P_r) the numbers $\overline{P}_p$ and $\overline{P}_r$ do not depend on a fixed program and specification, but on a *class* of programs and specifications with an associated probability distribution. (This approach is quite similar to Gourlay's analysis of mutation testing in [10].)

In order to avoid technical complications and to exploit as much information on the program under test as possible, we adopt a concept chosen by Eckhardt and Lee [6, p. 1,512], restricting ourselves to a (hypothetical) reference class of programs with the following property: All programs in the reference class have the same input domain D as the program to be tested, and refer to the same specification.[1] This implies that the input domain of any program in the reference class can be subdivided into disjoint subsets D_i just in the same way as the chosen subdivision strategy prescribes it for the program under test. Especially, the numbers d_i are fixed, deterministic quantities, contrary to the failure rates F_i. The following simple example illustrates this point of view:

EXAMPLE 1. Consider the following program, assuming that the input domain is $D = \{1, \ldots, 20\}$:

```
var x: integer;
begin read (x);
      if (x < = 10) then write ("small")
          else write ("large")
end;
```

The path testing strategy subdivides D into the two subdomains $D_1 = \{1, \ldots, 10\}$ and $D_2 = \{11, \ldots, 20\}$. In the reference class of programs, we stick to this partition. Suppose that the program under test is correct, and the reference class contains a program written to the same specification, but with a domain error:

```
var x: integer;
begin read (x);
      if (x < 10) then write ("small")
          else write ("large")
end;
```

Then, the incorrect processing of input $x = 10$ would not be contributed to D_2, but still to D_1 (yielding a failure rate $\theta_1 = 0.1$ for this program), although in the last program, input 10 is processed by the second path. Suppose that the reference class only consists of the two indicated programs. Then the expected failure rates for the subdomains are $\overline{\theta}_1 = 0.05$ and $\overline{\theta}_2 = 0$.

In Duran's and Ntafos' paper [5], there is already an implicit probabilistic consideration of the failure rates in the subdomains, since in their simulation experiments, the authors have assigned failure rates to the subdomains according to carefully selected distributions on the interval [0, 1]. Nevertheless, their results are more favorable for random testing than ours. A possible explanation will be discussed after the presentation of our results in the next section.

For an illustration of the fact that deterministic assumptions on the failure rates entail essentially different optimal decisions than probabilistic assumptions, even if the expected failure rates are the same, the following example might be helpful:

EXAMPLE 2. Let us consider two extreme situations, one with maximum, the other with minimum certainty on failure rates:

Situation 1. A program with a single integer input variable between 1 and 20 is presented to a tester, and she/he is told that in the subdomain of inputs between one and 10, there are six failure-causing and four correctly processed integers, while in the subdomain of inputs between 11 and 20, there are five failure-causing and five correctly processed integers. Hence, the failure rates are 0.6 and 0.5 for the two subdomains, and are, in this situation, known with certainty. The tester is allowed to test two inputs, either both from the same subdomain, or one input from each subdomain, and obtains a certain amount of money, if she/he finds at least one failure-causing input. There are two reasonable options:

1) *Option 1.* Test two inputs from subdomain $\{1, \ldots, 10\}$. The probability of success turns out as $1 - (4/10 \cdot (3/9) = 0.866...$ in this case.
2) *Option 2.* Test one input from subdomain $\{1, \ldots, 10\}$ and one input from subdomain $\{11, \ldots, 20\}$. The probability of success is then $1 - (4/10) \cdot (5/10) = 0.8$.

So, option 1 is preferable to option 2.

Situation 2. In this situation, the tester is told that the code for inputs from subdomain $\{1, \ldots, 10\}$ has been written by one of 10 persons. She/he knows all these 10 programmers and is sure that six of them are totally incompetent, such that all inputs processed by their program parts will be failure-causing (failure rate 1), while the other four programmers are extremely competent, such that all inputs for their program parts will be correctly processed (failure rate 0). Moreover, she/he is informed that the code for subdomain $\{11, \ldots, 20\}$ has been written by one of ten *other* programmers; five of them are known as totally incompetent (failure rate 1), the other five as extremely competent (failure rate 0). However, the tester does not know *which* person from each team has been selected for implementing the corresponding part of the program.

Now, the actual failure rates for the program under test are *not* known, but their expected values are known: The expected value of a failure rate is the average of the possible actual failure rates, so it turns out to be $(6/10) \cdot 1 + (4/10) \cdot 0 = 0.6$ for subdomain $\{1, \ldots, 10\}$ and $(5/10) \cdot 1 + (5/10) \cdot 0 = 0.5$ for subdomain $\{11, \ldots, 20\}$. These are the same values as in situation 1.

In situation 2, however, the probability of success for option 1 (two test inputs from $\{1, \ldots, 10\}$) is $6/10 = 0.6$, while the probability of success for option 2 (one test input from each subdomain) is still $1 - (4/10) \cdot (5/10) = 0.8$.

So, in situation 2, option 1 is preferable to option 2. We see that uncertainty on the actual failure rates favors the diversification strategy of option 2.

Of course, this is an extreme example, since it merges (in situation 2) only failure rates of 0 and 100 percent. But even if the actually possible failure rates are less diverse, their deviation from their mean value effects a shift of the optimal selection strategy towards diversification.

1. Note that this reference class of programs can be realized physically by *n*-version-programming ([19], [6]). So, in principle, the distribution of the failure rates could be measured empirically to any desired degree of accuracy in a quite objective way.

3 Expected Testing Effectiveness Under Uncertainty

3.1 Comparison Under the Fault Detection Probability Criterion

In order to investigate the effect of random (i.e., uncertain) failure rates on the relative performance of random testing vs. partition testing, let us compare the situation where the tester *knows* that all subdomains have the same failure rate, with the situation where the tester has no prior information on especially error-prone subdomains and therefore *estimates* the same failure rates for all subdomains. In the first case, we have

$$\theta_1 = \ldots = \theta_k \,. \tag{6}$$

Observation 7 in [24] states that (6) entails equality of the two fault detection probabilities: $P_p = P_r$.

In the second case, the absence of concrete information on subdomain failure rates can be expressed by the assumption of equal *expected* failure rates:

$$\overline{\theta}_1 = \ldots = \overline{\theta}_k \,, \tag{7}$$

where $\overline{\theta}_i$ is defined by (3). Contrary to (6), assumption (7) does not imply that the failure rates themselves are equal.

The assumption of equal expected failure rates is, in a certain sense, the natural one for all applications of partition testing where an *equal* number of test cases is selected from each subdomain. If we should expect that a special subdomain is more error-prone than another, say, $\overline{\theta}_i > \overline{\theta}_j$, then we could use this information by selecting more test cases from D_i than from D_j. (This intuitive consideration can be quantified, cf. [11].) So partition testing with equal numbers of test cases from each subdomain seems to be based on the implicit assumption that particularly error-prone subdomains are not identified in advance.

In other words, the standard application type of a partition testing technique treats each subdomain in the same way by selecting the same number of test cases, no matter which special properties the respective subdomain has. In such a situation of equal status for all subdomains, it evidently does not matter which subdomain is denoted by D_1, which by D_2, etc. So we can assume that the labeling of the subdomains by the indices 1, …, k is performed *at random*: A fixed subdomain has the same probability $1/k$ of getting the index 1, 2, …, k, respectively; conversely, a fixed integer i ($1 \le i \le k$) has, for each of the k subdomains, the same probability $1/k$ of being the index of this particular subdomain. Then, by symmetry, the expected value of the failure rate F_i/d_i (and even the distribution of F_i/d_i) must be the same for each i. Let us illustrate this consideration again by Example 2: If we do not know which of both subdomains has been treated by a programmer from the first and which by a programmer from the second team, then the expected value of the failure rate has to be set to $(0.6 + 0.5)/2 = 0.55$ for both subdomains.

We start our investigation by assuming that the numbers F_i of failure-causing inputs in the subdomains (and therefore also the failure rates) are *independent* random variables. This is a (first) generalization of the deterministic case: It is well-known that deterministic numbers $f_1, \ldots, f_k$ can be considered as a special case of independent random variables, having variances equal to zero. In technical terms, F_i is independent of F_j if the (unconditional) distribution of F_i is the same as the distribution of F_i conditional on the event that F_j has some fixed value f_j. So, informally, the independence assumption means that finding out the failure rate in one subdomain does not change our estimates of the failure rates in other subdomains.

The following theorem shows that on the assumptions above, $\overline{P}_p$ is an upper bound for $\overline{P}_r$. (The proofs of the theorems can be found in Appendix A.)

THEOREM 1. *For independent failure rates with equal expected value $\overline{\theta}$, partition testing is better or the same as random testing*:

$$\overline{P}_r \le \overline{P}_p \,.$$

Next, we derive a tight *lower* bound for $\overline{P}_r$:

THEOREM 2. *On the conditions of Theorem 1, the fault detection probability $\overline{P}_r$ of random testing is bounded below by*

$$\overline{P}_r \ge \frac{\overline{\theta}}{1-(1-\overline{\theta})^k}\,\overline{P}_p \,, \tag{8}$$

and there are special cases for which $\overline{P}_r$ gets arbitrarily close to the lower bound on the right-hand side of (8).

REMARK 1. For (compared to $1/k$) small expected failure rate $\overline{\theta}$,

$$\frac{\overline{\theta}}{1-(1-\overline{\theta})^k} \approx \frac{\overline{\theta}}{1-(1-k\overline{\theta})} = \frac{1}{k} \,,$$

so the tight lower bound for $\overline{P}_r$ is approximately $\overline{P}_p/k$. In other words: If a partition consists of, say, 100 subdomains, then the fault detection probability of partition testing can be up to about 100 times higher than that of random testing! This result is in a distinct contrast with its deterministic counterpart, Weyuker and Jeng's observation $P_r = P_p$ for equal failure rates.

REMARK 2. The special instance constructed in the second part of the proof of Theorem 2 (Appendix A) indicates under which circumstances $\overline{P}_r$ will approach the lower bound. This will be the case whenever

1) there are many small subdomains and one (or few) large subdomains,
2) the subdomains are (near to) *revealing* in the sense of [25]; a subdomain is called revealing if either all inputs contained in it are correctly processed, or they are all failure-causing (i.e., the failure rate is zero or one).

The *upper* bound $\overline{P}_p$ for $\overline{P}_r$, on the other hand, occurs in the "antirevealing" case, where the failure rate in each subdomain exactly mimics the overall failure rate.

There are arguments for the conjecture that, in some practical applications, both steps 1 and 2 are at least *approximately* satisfied, such that $\overline{P}_r$ can be expected to be closer to the lower than to the upper bound in these cases. Let us present these arguments in the following discussion of how the subdivision of the input domain is usually done:

Argument 1: Subdomain size. Most structural partition testing techniques define subdomains (directly or indirectly) on the base of predicates occurring in the program. Essentially, predicates may contain equality or inequality conditions, and equality conditions lead to extremely unbalanced subdomain sizes. Thus it may happen that there is a majority of subdomains with negligible size, compared to the size of one or a few "giant" subdomain(s), such that condition 1 is approximately satisfied.

EXAMPLE 3. Consider the simple "triangle-classification" procedure in Fig. 1, a traditional example in the software testing literature ([18]). There are three paths in this procedure, generating three subdomains D_1, D_2, D_3 for the path testing strategy, where D_s contains the inputs leading to output s (s = 1, 2, 3). If N is the number of representable integers, we have

$$d_1 = N$$
$$d_2 = 3N^2 - 3N^2 - 3N$$
$$d_3 = N^3 - 3N^2 + 2N,$$

so both D_1 and D_2 are of negligible size, compared to the D_3. (Of course, this effect is still more dramatic, if `i,j,k` are declared as reals instead of integers.)

The choice of subdomain sizes of nearly the same order of magnitude might be the main reason why a clear superiority of partition testing did not show in Duran and Ntafo's simulation experiments [5], although their failure-rate distributions for the subdomains were chosen in a very realistic way: In [5], operational probabilities p_i were selected randomly according to a *uniform* distribution on [0, 1]. In terms of the Weyuker-Jeng-model, this would mean that the subdomain sizes d_i are uniformly distributed. Then, however, most of the subdomains will have sizes of a comparable order of magnitude: If the values d_i vary, for example, between 1 and 10^8, about 90 percent of all subdomain sizes d_i will be integers with exactly eight decimal digits. Compared to Example 3, this is still a "relatively balanced" situation, where a significant dominance of partition testing is not yet to be expected.

Our analytical results are partially supported by the outcomes of the experimental study by Loo and Tsai [16]. They experimented with unbalanced (although not extremely unbalanced) subdomain sizes and found out that in this situation, random testing reaches the performance of partition testing only in particular cases where there are high failure rates in large subdomains or low failure rates in small subdomains.

```
ocedure triangle-classification;
var i, j, k: integer;
begin read (i, j, k);
      if (i = j) and (j = k)
      then write (1)
      else if (i = j) or (j = k) or (k = i)
           then write (2) {else} write (3)
end;
```

Fig. 1. Example program for path testing.

Argument 2: *Revealing domains*. A partition testing method where each subdomain is revealing cannot be hoped for. In the case of path testing, for example, already single bit differences in inputs processed along the same path can lead to radically different behavior in failure. Nevertheless, reasonable subdivision techniques tend to bundle up such inputs to subdomains that are processed by the program in a similar way. So, if one input in a subdomain D_i is recognized as failure-causing, this knowledge undoubtedly increases at least the *probability* that also the other inputs in D_i are failure-causing, and, conversely, if one input in D_i is recognized as correctly processed, then the other inputs in D_i have an increased probability of being correctly processed as well. As a consequence, in some subdomains, the failure rate will be zero or low, in others it will possibly be far above the overall failure rate F/d of the program (cf. the basic failure rate distribution assumed in Duran and Ntafo's simulations [5]). Therefore, it can be hoped that failure rate distributions obtained by the practical application of "clever" partition testing techniques tend to be rather U-shaped (and hence similar to the "revealing" case) than centered near $\overline{\theta}$ (i.e., similar to the "antirevealing" case). So, condition 2 above can often be expected to be approximately satisfied.

REMARK 3. Possibly, one might object our treatment of the numbers d_i as fixed, deterministic quantities: For example, it may be felt that, if two programs of the reference class contain corresponding `then` branches with assigned subdomains of different sizes, the respective variable d_i should be allowed to take different values (contrary to our treatment in Example 1). If this point of view is taken, each d_i gets a random variable, since the selection of the program from the reference class is assumed to be random. It can be shown that this extended model does not essentially modify our main results:

THEOREM 3. *The assertions of Theorem 1 and Theorem 2 remain valid, if the numbers d_i are considered as random variables.*

Let us study the influence of the distributions of the failure rates F_i/d_i on the effectiveness of random testing, compared to partition testing, in more detail by considering the special case k = 2 of only two subdomains. In this case (which is also the prototype for branch testing and some other subdomain-based testing methods that finally generate *overlapping* subdomains), the difference $\overline{P}_p - \overline{P}_r$ can be represented explictly. We give a general result for (possibly) *dependent* F_1, F_2. For the sake of simplicity, the numbers d_i are considered as deterministic again.

THEOREM 4. *For k = 2 subdomains and failure rates with expected value equal to $\overline{\theta}$, the difference of the fault detection probabilities for partition testing and random testing is given by*

$$\overline{P}_p - \overline{P}_r = \frac{1}{d^2}\left[Var(F_1) + Var(F_2) - \frac{d_1^2 + d_2^2}{d_1 d_2}\, Cov(F_1, F_2)\right], \quad (9)$$

where $Var(X)$ and $Cov(X, Y)$ denote the variance of X and the covariance between X and Y, respectively.

COROLLARY. *For k = 2 subdomains and independent failure rates with equal expected value* $\overline{\theta}$,

$$\overline{P}_p - \overline{P}_r = (1 / d^2)\left(Var(F_1) + Var(F_2)\right) \geq 0 \ . \qquad (10)$$

PROOF. Follows immediately from the fact that independent random variables F_1, F_2 are uncorrelated:

$$Cov(F_1, F_2) = 0 \ . \qquad \square$$

Equation (10) shows very clearly that $\overline{P}_p = \overline{P}_r$ is that boundary case where both failure rates have variance zero, i.e., are deterministically known. As soon as there is any uncertainty on the current failure rates, the stricter diversification strategy, partition testing, begins to dominate. Maximal variance is obtained in the case of revealing subdomains, where the failure rates F_i / d_i can only assume the extreme values 0 or 1.

Now let us turn to the situation where the variables F_i are *dependent*. It is clear from (9) that $\overline{P}_r \leq \overline{P}_p$ still holds when F_1 and F_2 are *negatively* correlated, i.e., $Cov(F_1, F_2) < 0$. Unfortunately, this is not the relevant case for practice, as the following intuitive consideration shows: If, for example, our a priori estimate for the expected failure rate has been $\overline{\theta} = 0.1$, and for the subdomain D_1 corresponding to a path 1 of the program, a failure rate of, say, 0.2 is observed during the test, then we shall not be inclined to decrease our estimate for the subdomain D_2 corresponding to path 2 from 0.1 to a lower value, but either to increase it (e.g., if path 2 has been implemented by the same programmer as path 1), or to leave it at the value of 0.1 (e.g., if path 2 has been implemented by another programmer.) So, inputs in *different* subdomains of a program may be expected to be positively correlated or not correlated at all with respect to their failure behavior, and an eventual positive correlation can be assumed to be much smaller than the positive correlation *within* the subdomains which frequently produces a "near-to-revealing" behavior.

The next theorem shows that also for positively correlated failure rates, (9) always leads to $\overline{P}_r \leq \overline{P}_p$, provided that not only the expectations, but also the variances of the failure rates are equal. (Remember that for random labelings of the subdomains, all failure rates F_i / d_i are identically distributed and have therefore the same variance.)

THEOREM 5. *On the conditions of Theorem 4 and with* $Var(F_1 / d_1) = Var(F_2 / d_2)$, *partition testing is better or the same as random testing:*

$$\overline{P}_r \leq \overline{P}_p \ .$$

It is difficult to find conditions under which a generalization of the last result to the case $k > 2$ holds. The reason is that for $k > 2$ not only the variance, but also higher moments enter into the formulas. Also simulation is of little help in this context, as long as it is not clear how the common distribution of the variables should be modeled in the general case. We confine ourselves here to the following, conceptually simple model assumption for more than two positively correlated failure rates:

Let us assume that the influence of randomness on the rate of correctly processed inputs in a subdomain can be decomposed into two factors as:

- *factor 1*, which influences all subdomains to the same extent (say, the professional skill of the programmers team or the dependence on the used programming language),
- *factor 2*, which is specific for the considered subdomain.

Factor 1 results in a positive correlation of the failure rates; factor 2 depends only on the particularities of the subdomain under consideration, so it can be assumed to be independent of factor 1 and of the values of factor 2 for other subdomains.

More formally, we assume that the rate $(d_i - F_i) / d_i$ of correctly processed inputs in subdomain D_i can be decomposed as follows:

$$(d_i - F_i) / d_i = H \cdot H_i \ ,$$

where the (random) factor $H \leq 1$ is the same for all subdomains, while the (random) factor $H_i \leq 1$ is specific for D_i, and $H, H_1, \ldots, H_k$ are independent. It is easy to check that for $Var(H) > 0$ and $E(H_i) > 0$ ($i = 1, \ldots, k$), the rates of correctly processed inputs, and therefore also the failure rates, are indeed positively correlated (see Appendix A). Nevertheless, for equal expected failure rates, the dominance of partition testing can be verified again:

THEOREM 6. *On the conditions above and with* $E(F_i / d_i) = \overline{\theta}$ *for all i, partition testing is better or the same as random testing:*

$$\overline{P}_r \leq \overline{P}_p \ .$$

REMARK 4. Our results should not be interpreted in the way that subdividing the input domain *anyhow* and selecting test cases from each subdomain is a cheap method to increase the effectiveness of testing. According to the bounds in Theorems 1 and 2, the gain of subdividing can be negligible as well as significant, depending on how appropriate the subdivision strategy is. In particular, *random* partitions are (nearly) useless, as the following example illustrates:

EXAMPLE 4. Let the following program with input domain $D = \{1, \ldots, 100\}$ be given:

```
var x: integer;
begin read (x);
      if (x >= 1 and x <= 50) then proc1 (x);
      if (x >= 51 and x <= 100) then proc2 (x)
end;
```

We make the assumption that procedure `proc1` either treats all values x correctly or none, and the same for `proc2`. Then both $D_1 = \{1, \ldots, 50\}$ and $D_2 = \{51, \ldots, 100\}$ are revealing subdomains. Furthermore, we assume that both `proc1` and `proc2` are correct with probability 1/2 each, independently from each other, such that there are four equiprobable situations: `proc1` and `proc2` correct; `proc1` correct and `proc2` incorrect; `proc1` incorrect and `proc2` correct; `proc1` and `proc2` incorrect.

Let the number of test inputs be $k = 2$. Since the number F of failure-causing inputs is 0, 50, and 100 with probabilities 0.25, 0.5, and 0.25, respectively, we obtain from (5):

$$\begin{aligned}\overline{P}_r &= \frac{2}{d}E(F) - \frac{1}{d^2}E(F^2)\\ &= \frac{2}{100}\cdot 50 - \frac{1}{100^2}\cdot 3750\ .\\ &= 0.625\end{aligned}$$

Next, consider any partition of the input domain D into two subdomains D_1, D_2. Let $m = |D_1 \cap \{1, \ldots, 50\}|$ be the number of integers smaller or equal 50 in D_1. One finds

$$\begin{aligned}\overline{\theta}_1 &= E\left(\frac{F_1}{d_1}\right)\\ &= \frac{1}{50}\cdot\frac{1}{4}\cdot[0 + m + (50 - m) + 50]\ ,\\ &= 0.5\end{aligned}$$

and analogously, $\overline{\theta}_2 = 0.5$. So both subdomains have the same expected failure rates. From (4), we get

$$\begin{aligned}\overline{P}_p &= E\left(\frac{F_1}{d_1}\right) + E\left(\frac{F_2}{d_2}\right) - E\left(\frac{F_1F_2}{d_1d_2}\right)\\ &= 1 - \frac{1}{50^2}\cdot\frac{1}{4}\cdot\left[0 + m(50 - m) + (50 - m)m + 50^2\right]\\ &= 1 + 10^4\cdot(2m^2 - 100m - 2500).\end{aligned}$$

It is easy to see that the last expression takes its minimum for $m = 25$ and its maximum for $m = 0$ or $m = 50$. Therefore, e.g., the partition

$$\begin{aligned}D_1 &= \{1, \ldots, 25\} \cup \{50, \ldots, 75\},\\ D_2 &= \{26, \ldots, 50\} \cup \{76, \ldots, 100\}\end{aligned}$$

is a worst case with respect to $\overline{P}_p$; we obtain $\overline{P}_p = 0.625 = \overline{P}_r$, i.e., the lower bound of $\overline{P}_p$ according to Theorem 1. On the other hand, the partition

$$\begin{aligned}D_1 &= \{1, \ldots, 50\},\\ D_2 &= \{51, \ldots, 100\}\end{aligned}$$

(or vice versa) is the best case with respect to $\overline{P}_p$; one computes $\overline{P}_p = 0.75$, which is equal to the probability that the program is incorrect at all, and significantly larger than $\overline{P}_r$.

Now let us turn to the interesting question which value $\overline{P}_p$ will take in the *average case*, when the subdivision is performed *randomly*. Short reflection, using the equal status of all integers in D, shows the following: A random partition of D into two disjoint subsets D_1, D_2 with $|D_1| = |D_2| = 50$ (where all such partitions are equiprobable) gives rise to two test inputs $X_1 \in D_1$, $X_2 \in D_2$ in such a way that all unordered pairs $\{X_1, X_2\}$ with $X_1, X_2 \in D$ and $X_1 \neq X_2$ have the same probability. Therefore, distinguishing the four possible combinations of correct or incorrect behavior of `proc1` and `proc2`, we obtain for the random partition:

$$\begin{aligned}\overline{P}_p &= \frac{1}{4}\cdot\left[0 + 2\cdot\left(1 - \frac{50}{100}\cdot\frac{49}{99}\right) + 1\right]\\ &= 0.6262\ldots\end{aligned}$$

This value is only slightly larger than that for random testing. The (minimal) advantage is here only due to the fact that partition testing prevents a double test of the same input.

The observation in Example 4 that random subdividing provides no essential advantage, compared to random testing, can be made quite general: Let $\overline{P}_{p(random)}$ denote the average value of $\overline{P}_p$, if the partition of D into k subdomains is performed randomly (giving each input the same probability of being put into a specific subdomain D_i), and let $\overline{P}_{r(without\ replacement)}$ denote the failure detection probability of random testing with k test inputs selected randomly from D *without replacement*. (Note that in Section 2, we have defined $\overline{P}_r$ as the failure detection probability of random testing with k test inputs selected randomly *with replacement*). Then

$$\overline{P}_{p(random)} = \overline{P}_{r(without\ replacement)}\ ,$$

since the effect of first distributing the elements of D randomly into k subdomains and then selecting one element from each subdomain is just the same as selecting k different random elements from D with equal probabilities at once. On the other hand,

$$\overline{P}_{r(without\ replacement)} \geq \overline{P}_{r(with\ replacement)} = \overline{P}_r\ ,$$

as it is evident from the observation that selecting test inputs with replacement may produce the same test input more than once, which reduces the test input set and hence also the failure detection probability. In total,

$$\overline{P}_{p(random)} \geq \overline{P}_r\ ,$$

in accordance with Theorem 1. (For $k > 1$, even $\overline{P}_{p(random)} \geq \overline{P}_r$ can be shown.) It should be noticed, however, that the difference between $\overline{P}_{r(without\ replacement)}$ and $\overline{P}_{r(with\ replacement)}$ is small for large input domain size d. Therefore, $\overline{P}_r$ is usually only slightly worse than $\overline{P}_{p(random)}$, but, as Theorem 2 shows, it can be considerably worse than $\overline{P}_p$ for a suitably chosen partition.

3.2 Extension to Other Comparison Criteria

Until now, we have used the *fault detection probability* as a comparison criterion for testing methods. It may be argued that this criterion is not the only interesting one: In many cases of application, testers will rather consider the number of faults found as a measure of effectiveness than the probability of finding *any* fault. On the other hand, also the number of detected fault may sometimes be misleading: Not all faults have the same severity; some of them may be that insignificant that it does not even pay to remove them, while other faults cause eminent financial loss (or even harm to human life). The most adequate criterion in such situations seems to be the *weighted* number of detected faults, where the weights are chosen proportional to the failure severities. In total, our results stating superiority of partition testing over random testing would surely be questionable if they should depend exclusively on the special criterion of fault detection probabilities.

However, it turns out that this is not the case: Roughly speaking, all the three indicated criteria (fault detection probability, expected number of detected faults, expected weighted number of detected faults) lead, in our framework, to the same order of testing methods. To verify this formally, we need some further notation.

First, in order to be able to speak of the *number* of faults, one must be in the position to distinguish between different faults. Let us assume that this can be done by some mean or other, such that there is a list of (possible) faults: fault 1, fault 2, etc., and for each input for the given program, one may determine by testing which fault(s), if any at all, this input exposes. Let d_1^{fail}, d_2^{fail}, … be the failure domains corresponding to fault 1, fault 2, …, respectively. The set D_j^{fail} consists of all inputs that cause fault j to be exposed during the execution of the program. Of course, the sets d_1^{fail}, d_2^{fail}, … may be overlapping, and some or all of these sets may be empty as well. The failure domain D^{fail} considered in Section 2 is the union of all sets d_j^{fail}

Moreover, let $P_p^{(j)}$ and $P_r^{(j)}$ denote the fault detection probabilities of partition testing and random testing, respectively, for a program with failure domain d_j^{fail} (i.e., a program where only fault j occurs, but none of the other faults). $P_p^{(j)}$ and $P_r^{(j)}$ can be interpreted as the probabilities of detecting fault j in the given program by partition testing and by random testing, respectively. They are computed in a formally analogous way as P_p and P_r:

$$P_p^{(j)} = 1 - \prod_{i=1}^{k} \left(1 - F_i^{(j)} / d_i\right)$$

and

$$P_r^{(j)} = 1 - \left(1 - F^{(j)} / d\right)^k ,$$

where $F_i^{(j)} = \left|D_i \cup D_j^{fail}\right|$ and $F^{(j)} = \left|D_j^{fail}\right|$. So we obtain a straightforward generalization of the framework in Section 2. If all possibly occurring failures are considered as identical, then $D_1^{fail} = D^{fail}$, $P_p^{(1)} = P_p$ and $P_r^{(1)} = P_r$. Otherwise, the formalism of Section 2 has to be applied for each fault j separately.

Analogously as in Section 2.2, we introduce the expected values of $P_p^{(j)}$ and $P_r^{(j)}$ in the considered reference class of programs, which yields the quantities $\overline{P}_p^{(j)} = E(P_p^{(j)})$ and $\overline{P}_r^{(j)} = E(P_r^{(j)})$.

In a consequent continuation of the analogy, the expected failure rate $\overline{\theta}_i^{(j)}$ in subdomain i with respect to fault j is defined as $E(F_i^{(j)} / d_i)$. Again by the argument that the labels of the subdomains D_i are arbitrary or random (cf. Section 3.1), it is reasonable to assume that the values $\overline{\theta}_i^{(j)}$ do not depend on the subdomain D_i, i.e., that $\overline{\theta}_i^{(j)} = \overline{\theta}^{(j)}$ for all i. So, provided that the other respective conditions are also satisfied, we are within the premises of Theorems 1, 5, and 6, and may conclude that

$$\overline{P}_r^{(j)} \le \overline{P}_p^{(j)}$$

for each j.

By $I(A)$, we denote the indicator of the event A, i.e., $I(A) = 1$ if A occurs, and $I(A) = 0$ otherwise. Using this notation, the number of detected faults is given by

$$\sum_j I(\text{fault } j \text{ is detected}),$$

and the *weighted* number of detected faults is given by

$$\sum_j w_j \, I(\text{fault } j \text{ is detected}),$$

where $w_j \ge 0$ is the weight (or severity) of fault j.

Now we can state the announced theorem which makes it possible to derive dominance with respect to expected (weighted) numbers of found faults from dominance with respect to fault detection probabilities:

THEOREM 7. *Whenever $\overline{P}_r^{(j)} \le \overline{P}_p^{(j)}$ for each j, then also the following two assertions hold:*

1) *The expected number of detected faults under random testing is smaller or equal to that under partition testing.*
2) *The expected weighted number of detected faults under random testing is smaller or equal to that under partition testing.*

This result is especially instructive for the case of programs with a large number of faults. For such programs, $\overline{P}_r$ and $\overline{P}_p$ are close to one anyway, so the assertion $\overline{P}_r \le \overline{P}_p$ is here of little practical relevance. The expected (weighted) number of faults found by random testing and partition testing, however, may nevertheless differ considerably: Note that the probabilities $\overline{P}_r^{(j)}$ and $\overline{P}_p^{(j)}$ for *special* (severe) faults j may be essentially smaller than one. For the final test of safety-critical applications, e.g., where the occurrence of a severe fault has to be considered as a rare event, our last derivations may be combined with Theorem 2, yielding the assertion that on these premises the expected value of the weighted number of found faults can be up to k times higher for partition testing than for random testing.

4 Concluding Remarks

We have shown that in a comparison between random testing and partition testing, deterministic assumptions on the failure rates systematically favor random testing, and that this effect is especially strong, if a partition consists of few large and many small subdomains. As a consequence, the fault detection ability of partition testing, compared to that of random testing, seems to be better than suggested by some prior investigations. In particular, it was demonstrated that, for example, independent failure rates with identical expected value in the subdomains guarantee a higher fault detection probability for partition testing, exceeding that of random testing up to a factor of about k, where k is the number of subdomains.

Some readers might wonder how prior information or assumptions on failure rates can influence the fault detection probabilities of partition testing and random testing: It could be argued that, as soon as a concrete program is given (together with its specification), there will be certain fault detection probabilities for partition testing and random testing, which are independent of what we *know* about the program. Let us make this point quite clear. It is true that the fault detection probabilities P_r and P_p for the given program are independent of our prior information. However, they are unknown to us, and there is no way to determine them before testing! So our decision cannot be based on the values P_r and P_p themselves, but only on our *expectation* of what these values will be, i.e., on $\overline{P}_r$ and $\overline{P}_p$. This expectation may be established by objective information on failure rates in a programs similar to the program under test. Also concrete information on the program under test could be exploited; this would be a Bayesian approach. The only condition for our results is that there is *any* way of assigning probability distributions to the failure rates, no matter whether an objective (frequentist) or subjective (Bayesian) interpretation of "probability" is chosen (cf. [17], [20]).

As to the interpretation of our results, let us add a warning remark. Partition testing strategies may be roughly classified into *clear box* and *black box* strategies, according to the criterion used for subdividing, which can be structural or functional. Our formal results hold for both types of partition testing strategies. Nevertheless, some of our informal arguments (especially in Remark 2, Section 3) rather refer to structural subdivision criteria, so their validity should be carefully checked when a special black box strategy is considered. Example 4 shows that at least random subdividing, which is a black box subdivision strategy, does not produce essentially better outcomes than random testing.

The presented results might lead to the impression that random testing is a technique of low value and should not be applied. In the author's opinion, however, more cautious conclusions should be drawn. There are some aspects still to be considered:

Aspect 1. First of all, the reader should remember that we have investigated a particular type of random testing, using a *uniform* distribution on the input domain. (The same holds for the investigations in [24].) Using other distributions has already been discussed in [5]. Appropriate nonuniform distributions may lead to significantly higher fault detection probabilities. Of course, there is a price to be paid for such improvements: the advantage that for test case generation, one does not need any information on the program except the knowledge of the input domain, gets lost.

Aspect 2. Secondly, there seems to be a broad consensus in the literature that random testing (e.g., with the operational distribution) can provide quantitative reliability estimates, while deterministic testing (e.g., deterministic path testing) cannot (see [21], [5], [14], [22], [7], [23]). In particular, random testing allows a quantitative judgment of the achieved reliability in the form of statistical confidence bounds (see [14], [22]).

Aspect 3. Third, random test data generators, where they can be applied, are able to produce very large test data sets. (Of course, not every application is well suited for the automated generation of test inputs.) On the additional condition that a *test oracle*, an automated or at least partially automated tool for the evaluation of test results, is available (the development of such a test oracle may be a very difficult task, cf. [4]!), random testing may win against partition testing simply by means of an overwhelming number of test cases. This effect, which has already been discussed by other authors, deserves some closer inspection in future research. It would be desirable to have necessary and / or sufficient conditions indicating under what circumstances K random test cases are better than k partition test cases. To give an example for such a condition: It is possible to show, using Theorem 2, that for *very small* independent failure rates with equal expectation, $K = k^2$ random test cases have always a higher fault detection probability than k partition test cases from k subdomains. Conditions of this type could be helpful for the comparison of random and partition testing on the base of cost-effectiveness. (Incidentally, let us mention that random test data generation is not the only way to produce large test data sets. In [13], e.g., a method for improving the fault detection probability of random testing for a class of numerical programs, using an appropriate "derandomization" of the generation process, was outlined.)

Let us discuss a few special questions left open by our results. Some of them may possibly be attacked by analytical methods, others are obvious candidates for simulation experiments and / or empirical studies.

Question left open 1. We have assumed that in partition testing, only one test case is selected from each subdomain. A more general assumption would be the following: From each subdomain D_i, exactly $m \geq 1$ test cases have to be selected. It is not clear whether in such a situation our result $\overline{P}_r \leq \overline{P}_p$ for independent failure rates with identical expectation remains valid, and how multiple selection of test inputs from subdomains influences the lower bound on $\overline{P}_r$. An analytical treatment of this question is difficult, so simulation results would be of great value. Of course, also *varying* numbers m_i of test cases selected from the subdomains D_i may be investigated, as they have already been considered in [5], [24]. Chen and Yu ([2], [3]) have shown that if m_i is chosen proportional to d_i ("proportional sampling"), then partition testing is always better than random testing. The difficulty lies in achieving an exact proportion between m_i and d_i. Our approach might possibly be applied for judging *approximations* to proportional sampling.

Question left open 2. Our results were based on the assumption of equal expected failure rates in the subdomains defined by the partition. As it was stated in Section 3, this assumption is almost inevitable for the standard application type of a partition testing technique, where subdomains are not distinguished according to prior information on their error-proneness or other criteria. Of course, also the opposite case may be studied, where such information is taken into account. In this case, it is obviously more efficient to select varying numbers m_i of test cases from the subdomains D_i. To have a fair comparison, also the distribution of the test cases selected by random testing should then be changed from a uniform distribution on D to a nonuniform distribution giving more weight to the (presumably) more error-prone subdomains. This is again a possible topic for simulation experiments.

Question left open 3. By restricting ourselves to partition testing instead of the more general case of subdomain-based testing methods, we have assumed that all subdomains are *nonoverlapping*. For practical applications, it would be very interesting to have results concerning the "overlapping" case (cf. the discussions in [14], [24]), since many well-established subdomain-based testing techniques (like, e.g., branch testing) define subdomains that are not pairwise disjoint. Intuitively, one can expect that the trends outlined in this paper are still valid for overlapping subdomains, but weaker: The extreme case for overlapping subdomains is $D_1 = \ldots = D_k = D$, and in this case, partition testing coincides with random testing. So one may conjecture that the behavior of the techniques that do *not* define proper partitions lies somewhere between partition testing and random testing. Mathematical results making this consideration precise and/or simulation results would be worthwhile.

Question left open 4. The aspect of *varying failure costs* should be investigated in more detail than in Section 3.2. In the discussion on the relative performance of partition and random testing, this aspect was studied by Tsoukalas, Duran and Ntafos [22]. As one of their results, they stated that in an extended model where varying degrees of failure severity are distinguished, the superiority of partition testing over random testing is more pronounced than in the model investigated in [5]. Although the criterion of comparison in [22] is not the fault detection probability or the expected weighted number of detected faults, but the obtained confidence bound on reliability estimates, it can be conjectured that for varying failure costs, an investigation along the lines of our derivations in Section 3.1 would unveil an even stronger advantage for partition testing. This conjecture could possibly be accessible to an analytical treatment.

Finally, let us briefly summarize consequences of our results for the work of a practicing test engineer:

- In spite of (erroneous) conclusions that might possibly be drawn from previous investigations, partition–based testing techniques are well-founded. Even if no especially error-prone subdomains of the input domain can be identified in advance, partition testing can provide substantially better results than random testing.
- Because of the close relations between partition testing and other subdomain-based testing methods (branch testing, all-uses, mutation testing etc.), also the superiority of the last-mentioned methods over random testing can be justified. The wide-spread practice of spending effort for satisfying diverse coverage criteria instead of simply choosing random test cases is *not* a superstitious custom; it is a procedure the merits of which can be understood by sufficiently subtle, but formally precise models.
- The effort for satisfying partition-based coverage criteria is particularly well spent, whenever the partition leads to subdomains of largely varying sizes, each of which is processed by the program or system in a rather homogeneous way (i.e., the processing steps are similar for all inputs of a given subdomain). Contrary, the advantages of partition testing are only marginal in the case of subdomains of comparable sizes and heterogeneous treatment by the program. In any case, the partition should not be arbitrarily chosen, but carefully derived from the structure or function of the program.

We admit that the main intention of this article was to get more clarity in regard to the question *whether* partition-testing techniques should be applied or not and not *how* they can be applied in a more effective way. Since partition testing techniques are expensive, such that doubts whether they are really superior to arbitrary ("random") choice of test cases would drastically discourage their application, we think that our results are also of considerable practical interest. Nevertheless, some readers might desire more concrete hints on how their partition testing strategies could be improved. This question exceeds the scope of this article, but we are optimistic that further investigations along the lines of the presented approach will also produce results of this type.

APPENDIX A

PROOF OF THEOREM 1. Induction with respect to k. For $k = 1$, we obtain $\overline{P}_r = \overline{P}_p = \overline{\theta}$. Let $k > 1$. By (5),

$$\overline{P}_r = 1 - E\left((1 - F/d)^k\right).$$

We use the stochastic inequality

$$E(Y^{a+b}) \geq E(Y^a)E(Y^b) \quad (a, b \geq 0; \quad Y \geq 0) \tag{11}$$

which follows from Lyapunov's inequation

$$\left[E(U^\alpha)\right]^{1/\alpha} \leq \left[E(U^\beta)\right]^{1/\beta} \quad (0 < \alpha \leq \beta; \quad U \geq 0).$$

Setting $Y := 1 - F/d$, $a := k - 1$ and $b := 1$ in (11), we obtain

$$\overline{P}_r \leq 1 - E\left((1 - F/d)^{k-1}\right) \cdot E(1 - F/d)$$
$$= 1 - E\left((1 - F/d)^{k-1}\right) \cdot (1 - \overline{\theta}).$$

(Observe that $E(F/d) = (1/d)\, E(d_1 F_1 / d_1 + \ldots + d_k F_k / d_k) = \overline{\theta}$.)

On the other hand, by (4) we find

$$\overline{P}_p = 1 - \prod_{i=1}^{k} E(1 - F_i/d_i) = 1 - (1 - \overline{\theta})^k, \tag{12}$$

using the independence of the failure rates F_i/d_i. So it remains to show that

$$E\left((1 - F/d)\right)^{k-1} \geq (1 - \overline{\theta})^{k-1}. \tag{13}$$

For that purpose, define a partition of D by joining subdomains D_{k-1} and D_k in the given partition:

$$\tilde{D}_i := D_i \quad (i = 1, \ldots, k-2), \quad \tilde{D}_{k-1} := D_{k-1} \cup D_k .$$

The corresponding numbers of failure-causing inputs are then

$$\tilde{F}_i = F_i \quad (i = 1, \ldots, k-2), \quad \tilde{F}_{k-1} = F_{k-1} + F_k ,$$

and analogous equations hold for the subdomain sizes $\tilde{d}_i$. $\tilde{F}_1, \ldots, \tilde{F}_{k-1}$ are independent random variables, and it is easy to verify that all expected failure rates $E(\tilde{F}_i / \tilde{d}_i)$ are again equal to $\overline{\theta}$. By induction assumption and with the use of (12), we obtain

$$1 - E\left(\left(1 - \frac{\tilde{F}_1 + \ldots + \tilde{F}_{k-1}}{\tilde{d}_1 + \ldots + \tilde{d}_{k-1}}\right)^{k-1}\right) \le 1 - (1-\overline{\theta})^{k-1} .$$

Hence

$$E\big((1 - F/d)\big)^{k-1} = E\left(\left(1 - \frac{\tilde{F}_1 + \ldots + \tilde{F}_{k-1}}{\tilde{d}_1 + \ldots + \tilde{d}_{k-1}}\right)^{k-1}\right), \le 1 - (1-\overline{\theta})^{k-1} ,$$

which yields (13). □

PROOF OF THEOREM 2.

Case 1. We show the inequation. Since $0 \le 1 - F/d \le 1$, one has $(1 - F/d)^k \le 1 - F/d$, and therefore

$$1 - \overline{P}_r = E\big((1-F/d)^k\big) \le E(1 - F/d) = \overline{\theta} . \qquad (14)$$

Thus, $\overline{P}_r \ge \overline{\theta}$. Because of (12), this implies

$$\frac{\overline{P}_r}{\overline{P}_p} \ge \frac{\overline{\theta}}{1 - (1-\overline{\theta})^k} .$$

Case 2. We show that the lower bound is tight in the sense indicated in the theorem, i.e., that the bound cannot be improved. Consider the following case:

$$d_1 = d - k, \quad d_i = 1 \quad (i = 2, \ldots, k) ,$$

and

$$F_i = \begin{cases} 0 & \text{with probability } 1 - \overline{\theta}, \\ d_i & \text{with probability } \overline{\theta}, \end{cases}$$

for $i = 1, \ldots, k$, where $F_1, \ldots, F_k$ are independent. It is immediate that $E(F_i / d_i) = \overline{\theta}$ $(i = 1, \ldots, k)$. Since $F_1 = 0$ with probability $1 - \overline{\theta}$, it follows that $F = F_1 + \ldots + F_k \le k - 1$ with probability $\ge 1 - \overline{\theta}$, i.e., $(1 - F/d)^k \ge (1 - (k-1)/d)^k$ with probability $\ge 1 - \overline{\theta}$. Therefore,

$$\overline{P}_r = 1 - E\big((1 - F/d)^k\big) \le 1 - (1-\overline{\theta})(1 - (k-1)/d)^k . \quad 15)$$

For fixed k and $d \to \infty$, one has $(1 - (k-1)/d)^k \to 1$, so the expression on the right-hand side of (15) tends to $\overline{\theta}$ for growing d. Together with the inequation $\overline{P}_r \ge \overline{\theta}$ derived above, we obtain the result that for appropriate special cases, $\{P\}_r$ gets arbitrarily close to $\overline{\theta}$, which proves the assertion, again because of (12). □

PROOF OF THEOREM 3. Let $\mathbf{d} = (d_1, \ldots, d_k)$ be the vector of random variables d_i. Now consider any *fixed* vector $\mathbf{d}^{(0)} = (d_1^{(0)}, \ldots, d_k^{(0)})$ of values $d_i^{(0)}$ of the random variables d_i. Let

$$\overline{P}_p(\mathbf{d}^{(0)}) = E\left(1 - \prod_{i=1}^{k}\left(1 - \frac{F_i}{d_i^{(0)}}\right)\right)$$

(cf. (4)). By $E^{(d)}$, we denote the expectation with respect to the distribution of $\mathbf{d} = (d_1, \ldots, d_k)$, such that now

$$\overline{P}_p = E^{(d)}\big(\overline{P}_p(\mathbf{d})\big) .$$

$\overline{P}_r$ is still given by (5). Theorem 1 yields $\overline{P}_r \le \overline{P}_p(\mathbf{d}^{(0)})$ for each $\mathbf{d}^{(0)}$. Because of the monotonicity of the expectation operator $E^{(d)}$, this implies $\overline{P}_r \le \overline{P}_p$, thus the generalization of Theorem 1 to random subdomain sizes d_i is proved. The generalization of the inequation in Theorem 2 is demonstrated in a quite analogous way. The tightness of the lower bound in the generalization of Theorem 2 follows immediately, since for showing it, it is sufficient to indicate special cases, and these are provided by the deterministic construction in the proof of Theorem 2, case 2. □

PROOF OF THEOREM 4. By (4),

$$\begin{aligned}\overline{P}_p &= 1 - E\big((1 - F_1/d_1)(1 - F_2/d_2)\big) \\ &= E(F_1/d_1) + E(F_2/d_2) - E\big((F_1F_2)/(d_1d_2)\big) \\ &= 2\overline{\theta} - E(F_1F_2)/(d_1d_2).\end{aligned}$$

By (5),

$$\begin{aligned}\overline{P}_r &= 1 - E\big((1 - F/d)^2\big) \\ &= 1 - \big(1 - 2E(F/d) + E(F^2/d^2)\big) \\ &= 2\overline{\theta} - E(F^2)/d^2 ,\end{aligned}$$

hence

$$\begin{aligned}\overline{P}_p - \overline{P}_r &= E(F^2)/d^2 - E(F_1F_2)/(d_1d_2) \\ &= \frac{1}{d^2}\big(E(F_1^2) + E(F_2^2) + 2E(F_1F_2)\big) - \frac{1}{d_1d_2}E(F_1F_2) \quad (16) \\ &= \frac{1}{d^2}\left[E(F_1^2) + E(F_2^2) + \left(2 - \frac{d^2}{d_1d_2}\right)E(F_1F_2)\right].\end{aligned}$$

Using

$$\begin{aligned}E(F_i^2) &= Var(F_i) + \big(E(F_i)\big)^2 \\ &= Var(F_i) + d_i^2\overline{\theta}^2 \quad (i = 1, 2)\end{aligned}$$

and

$$\begin{aligned}E(F_1F_2) &= Cov(F_1, F_2) + E(F_1)\,E(F_2) \\ &= Cov(F_1, F_2) + d_1d_2\overline{\theta}^2 ,\end{aligned}$$

we finally obtain (9) by insertion into (16). □

PROOF OF THEOREM 5. Let $Var(F_i\ /\ d_i) = \sigma^2$. We obtain

$$Var(F_i) = d_i^2 \sigma^2 \quad (i = 1, 2)\ ,$$

and

$$Cov(F_1, F_2) \le [Var(F_1)]^{1/2}\ [Var(F_2)]^{1/2} = d_1 d_2 \sigma^2\ .$$

This yields

$$d^2(\overline{P}_p - P_r) \ge d_1^2\sigma^2 + d_2^2\sigma^2 - (d_1^2 + d_2^2)\sigma^2 = 0\ . \qquad \square$$

PROOF OF THE ASSERTION BEFORE THEOREM 6. We have to show that for $i \ne j$, the random variables $(d_i - F_i)/d_i$ and $(d_j - F_j)/d_j$ are positively correlated, provided that $Var(H) > 0$, $E(H_i) > 0$ $(i = 1, \ldots, k)$, and $H, H_1, \ldots, H_k$ are independent. For that purpose, we show that the covariance of the two random variables is strictly positive:

$$\begin{aligned} Cov\left(\frac{d_i - F_i}{d_i}, \frac{d_j - F_j}{d_j}\right) &= Cov(HH_i, HH_j) \\ &= E(HH_i \cdot HH_j) - E(HH_i) \cdot E(HH_j) \\ &= E(H^2)E(H_i)E(H_j) - [E(H)]^2 E(H_i)E(H_j) \\ &= Var(H)E(H_i)E(H_j) > 0. \end{aligned}$$

$\square$

PROOF OF THEOREM 6. One finds

$$\begin{aligned} 1 - \overline{P}_r &= E\left(\left(\frac{d-F}{d}\right)^k\right) \\ &= \frac{1}{d^k}\ E\left(\left(\sum_{i=1}^k (d_i - F_i)\right)^k\right) \\ &= \frac{1}{d^k}\ E\left(\left(\sum_{i=1}^k d_i HH_i\right)^k\right) \\ &= \frac{1}{d^k}\ E(H^k)\ E\left(\left(\sum_{i=1}^k d_i H_i\right)^k\right), \end{aligned}$$

and

$$\begin{aligned} 1 - \overline{P}_p &= E\left(\prod_{i=1}^k \frac{d_i - F_i}{d_i}\right) \\ &= E\left[\prod_{i=1}^k HH_i\right] \\ &= E(H^k) \prod_{i=1}^k E(H_i). \end{aligned}$$

Hence

$$\frac{1}{E(H^k)}\ (\overline{P}_p - \overline{P}_r) = \frac{1}{d^k}\ E\left(\left(\sum_{i=1}^k d_i H_i\right)^k\right) - \prod_{i=1}^k E(H_i). \quad (17)$$

Set $G_i = d_i(1 - H_i)$ $(i = 1, \ldots, k)$. Then $G_1, \ldots, G_k$ are independent random variables, and the "rates" G_i/d_i have equal expectation:

$$\begin{aligned} E(G_i\ /\ d_i) &= 1 - E(H_i \\ &= 1 - (1 - \overline{\theta})\ /\ E(H). \end{aligned}$$

(The last equation follows from $E(H)\ E(H_i) = E(HH_i) = 1 - \overline{\theta}$) Using the variables G_i, the right-hand side of (17) can be represented (after a short calculation) as

$$1 - E\left(\left(1 - \frac{1}{d}\sum_{i=1}^k G_i\right)^k\right) - E\left(\prod_{i=1}^k \left(1 - \frac{G_i}{d_i}\right)\right). \qquad (18)$$

It is not difficult to see that the assertion

$$1 - E\left(\left(1 - \frac{1}{d}\sum_{i=1}^k F_i\right)^k\right) \le 1 - E\left(\prod_{i=1}^k \left(1 - \frac{F_i}{d_i}\right)\right)$$

of Theorem 1 can be generalized from integer variables F_i to real-valued variables G_i. As a consequence, the expression (18) and hence also the expressions in (17) are nonnegative. This yields the theorem because of $E(H^k) > 0$. $\square$

PROOF OF THEOREM 7. It is sufficient to prove assertion 2, since assertion 1 is only the special case of identical weights $w_1 = w_2 = \ldots = 1$. Let, for a *fixed* given program with known failure rates, E_p and E_r denote the expectation of a quantity in the case when partition testing and random testing is applied, respectively. We obtain

$$\begin{aligned} &E_p\left(\sum_j w_j\ I(\text{fault } j \text{ is detected})\right) \\ &\quad = \sum_j w_j\ E_p\big(I(\text{fault } j \text{ is detected})\big) \\ &\quad = \sum_j w_j\ P_p^{(j)}, \end{aligned}$$

and

$$\begin{aligned} &E_r\left(\sum_j w_j\ I(\text{fault } j \text{ is detected})\right) \\ &\quad = \sum_j w_j\ E_r\big(I(\text{fault } j \text{ is detected})\big) \\ &\quad = \sum_j w_j\ P_r^{(j)}, \end{aligned}$$

In the second step, we take the expected value E in the considered reference class of programs. This yields

$$\begin{aligned} E\left(\sum_j w_j\ P_p^{(j)}\right) &= \sum_j w_j\ (P_p^{(j)}) \\ &= \sum_j w_j\ \overline{P}_p^{(j)} \end{aligned}$$

as the expected weighted number of detected faults for partition testing, and

$$\begin{aligned} E\left(\sum_j w_j\ P_r^{(j)}\right) &= \sum_j w_j\ E(P_r^{(j)}) \\ &= \sum_j w_j\ \overline{P}_r^{(j)} \end{aligned}$$

as the expected weighted number of detected faults for random testing. The assertion follows now from $\overline{P}_r^{(j)} \le \overline{P}_p^{(j)}$ and $w_j \ge 0$. $\square$

ACKNOWLEDGMENTS

The author wishes to thank I. Bomze, A. Futschik, and E.J. Weyuker for valuable hints and comments. Furthermore, the author is indebted to the anonymous referees for their profound and helpful remarks on the first versions. In particular, the content of Theorem 3 was suggested by one of the referees.

REFERENCES

[1] B. Beizer, *Software Testing Techniques.* New York: Van Nostrand Reinhold, 1990.
[2] T.Y. Chen and Y.T. Yu, "On the Relationship between Partition and Random Testing," *IEEE Trans. Software Eng.*, vol. 20, pp. 977–980, 1994.
[3] T.Y. Chen and Y.T. Yu, "A More General Sufficient Condition for Partition Testing to be Better than Random Testing," *Information Processing Letters*, vol. 57, pp. 145–149, 1996.
[4] J.D. Day and J.D. Gannon, "A Test Oracle Based on Formal Specifications," *Softfair, A Second Conf. Software Development Tools, Techniques, and Alternatives*, pp. 126–130, San Francisco, Dec. 1985.
[5] J.W. Duran and S.C. Ntafos, "An Evaluation of Rrandom Testing," *IEEE Trans. Software Eng.*, vol. 10, pp. 438–444, 1984.
[6] D.E. Eckhardt and L.D. Lee, "A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors," *IEEE Trans. Software Eng.*, vol. 11, pp. 1,511–1,517, 1985.
[7] W.D. Ehrenberger, "Combining Probabilistic and Deterministic Verification Efforts," *Proc. SAFECOMP'92*, H. Frey, ed., Oxford, England: Pergamon, pp. 299–304, 1992.
[8] P.G. Frankl and E.J. Weyuker, "A Formal Analysis of the Fault-Detecting Ability of Testing Methods," *IEEE Trans. Software Eng.*, vol. 19, pp. 202–213, 1993.
[9] P.G. Frankl and E.J. Weyuker, "Provable Improvements on Branch Testing," *IEEE Trans. Software Eng.*, vol. 19, pp. 962–975, 1993.
[10] J.S. Gourlay, "A Mathematical Framework for the Investigation of Testing," *IEEE Trans. Software Eng.*, vol. 9, pp. 686–709, 1983.
[11] W.J. Gutjahr, "Optimal Test Distributions for Software Failure Cost Estimation," *IEEE Trans. Software Eng.*, vol. 21, pp. 219–228, 1995.
[12] W.J. Gutjahr, "Importance Sampling of Test Cases in Markovian Software Usage Models," *Probability in the Eng. and Informational Sciences*, vol. 11, pp. 19–36, 1997.
[13] W.J. Gutjahr and G. Danninger, "Efficient Selection of Test Data from a Polyhedral Input Domain," *Operations Research Proc. 1994*, U. Derigs et al., eds., pp. 233–238, Berlin: Springer, 1995.
[14] R. Hamlet and R. Taylor, "Partition Testing Does Not Inspire Confidence," *IEEE Trans. Software Eng.*, vol. 16, pp. 1,402–1,411, 1990.
[15] R. Hamlet, "Theoretical Comparison of Testing Methods," *Proc. Third Symp. Testing, Analysis and Verification*, pp. 28–37, 1989.
[16] P.S. Loo and W.K. Tsai, "Random Testing Revisited," *Information and Software Technology*, vol. 30, pp. 402–417, 1988.
[17] P.G. Moore, *The Business of Risk.* Cambridge: Cambridge Univ. Press, 1993.
[18] G.J. Myers, *The Art of Software Testing.* New York: Wiley, 1979.
[19] B. Randell, "System Structure for Software Fault Tolerance," *IEEE Trans. Software Eng.*, vol. 1, pp. 220–232, 1975.
[20] M. Smithson, *Ignorance and Uncertainty*. New York: Springer, 1988.
[21] T.A. Thayer, M. Lipow, and E.C. Nelson, *Software Reliability*. Amsterdam, The Netherlands: North-Holland, 1978.
[22] M.Z. Tsoukalas, J.W. Duran, and S.C. Ntafos, "On Some Reliability Estimation Problems in Random and Partition Testing," *IEEE Trans. Software Eng.*, vol. 19, pp. 687–697, 1993.
[23] G.H. Walton, J.H. Poore, and C.J. Trammell, "Statistical Testing of Software Based on a Usage Model," *Software–Practice and Experience*, vol. 25, pp. 97–108, 1995.
[24] E.J. Weyuker and B. Jeng, "Analyzing Partition Testing Strategies," *IEEE Trans. Software Eng.*, vol. 17, pp. 703–711, 1991.
[25] E.J. Weyuker and T.J. Ostrand, "Theories of Program Testing and the Application of Revealing Subdomains," *IEEE Trans. Software Eng.*, vol. 6, pp. 236–245, 1980.
[26] E.J. Weyuker, St. N. Weiss, and R. Hamlet, "Comparison of Program Testing Strategies," *Proc. Fourth Symp. Testing, Analysis and Verification*, pp. 1–10, 1991.
[27] E.J. Weyuker, "The Applicability of Program Schema Results to Programs," *Int. J. Computing Information Sciences*, vol. 8, pp. 387–403, 1979.

Walter J. Gutjahr received his MSc and PhD degrees in mathematics from the University of Vienna, Austria, in 1980 and 1985, respectively. From 1980–1988 he was with Siemens Corporate, working on diverse program and system development projects and was head of a software testing group. Currently, he is a professor of computer science and applied mathematics at the University of Vienna. His interests include analysis of algorithms, optimization, and software engineering (especially testing and reliability).