A Theoretical Analysis of Random Regression Test Prioritization

Pu Yi[1], Hao Wang[1], Tao Xie[1](✉), Darko Marinov[2], and Wing Lam[3]

[1] Peking University, Beijing, China
lukeyi@pku.edu.cn, tony.wanghao@stu.pku.edu.cn, taoxie@pku.edu.cn
[2] University of Illinois Urbana-Champaign, Urbana, IL, USA
marinov@illinois.edu
[3] George Mason University, Fairfax, VA, USA
winglam@gmu.edu

Abstract. Regression testing is an important activity to check software changes by running the tests in a test suite to inform the developers whether the changes lead to test failures. Regression test prioritization (RTP) aims to inform the developers faster by ordering the test suite so that tests likely to fail are run earlier. Many RTP techniques have been proposed and are often compared with the random RTP baseline by sampling some of the $n!$ different test-suite orders for a test suite with n tests. However, there is no theoretical analysis of random RTP. We present such an analysis, deriving probability mass functions and expected values for metrics and scenarios commonly used in RTP research. Using our analysis, we revisit some of the most highly cited RTP papers and find that some presented results may be due to insufficient sampling. Future RTP research can leverage our analysis and need not use random sampling but can use our simple formulas or algorithms to more precisely compare with random RTP.

Keywords: Regression Test Prioritization · Random · Analysis

1 Introduction

Software developers commonly check their code by running tests. *Regression testing* [48] runs tests after code changes, to check whether the changes break the existing functionality. A test that passes before the changes but fails after indicates that the changes should be debugged (unless the test is flaky [25]). Finding test failures faster enables the developers to start debugging earlier.

A popular regression testing approach is *regression test prioritization (RTP)* [12, 19, 21, 23, 38, 39, 48], which runs the tests from a test suite in an order that aims to find test failures sooner. For example, Google [14] and Microsoft [42] report on using RTP in industry. More formally, a test suite T is a set (unordered) of tests, and RTP techniques produce a test-suite *order*—a permutation of the tests in the test suite—in which to run the tests. Various RTP techniques have been proposed in the literature since the seminal papers from 20+ years ago [12, 36, 38, 47] that have garnered thousands of citations.

D. Fisman and G. Rosu (Eds.): TACAS 2022, LNCS 13244, pp. 217–235, 2022.
https://doi.org/10.1007/978-3-030-99527-0_12

RTP techniques are often compared with random RTP. Our inspection [44] of the 100 most cited papers on RTP shows that 56 papers use random RTP as a comparison baseline. Although random RTP often performs worse than advanced techniques, recent papers still use random RTP, because it has a small overhead and may perform well in certain scenarios. We additionally check papers published in the latest testing conferences (ICST and ISSTA 2020/2021) and find that 50% (2/4) of the RTP papers [6,15,30,34] use random RTP. While random RTP has been used as a baseline for 20+ years, *all* evaluations have been empirical, performed by randomly sampling some of the $n!$ orders for a test suite with n tests. The selected sample size varies (20, 50, 100, 200, 1000), with no clear correlation with n; some papers do not even report the sample size [44]. However, no prior work has presented a theoretical analysis of random RTP.

Before we summarize our analysis, we describe some metrics and scenarios most commonly used in RTP research. We first introduce some terms: *failure* is simply a failing test, *fault* is the root cause (bug in the code) for the failure, and we say that a failure *detects* a fault if the failure is caused by the fault [36]. In general, many failures may detect the same fault, and one failure may detect many faults. We capture the relationship between failures and faults by a *failure-to-fault matrix*. To compare RTP techniques, researchers quantify how fast (test-suite) orders find all *faults* (not failures because having many failures that detect the same fault is not as valuable as having a few failures that detect many faults).

RTP evaluations involve three aspects: RTP metric, failure-to-fault matrix, and allowed orders. The most widely used metric is Average Percentage of Faults Detected (APFD) [38], denoted as α for short. Another popular metric is Cost-Cognizant APFD (APFD_c) [11], denoted as γ for short. Section 2 formally defines these metrics based on the failure-to-fault matrix; each metric assigns to an order a value between 0 and 1, with higher values indicating better orders. Traditional RTP research used seeded faults, which allow fairly precisely deriving the failure-to-fault matrix [10,22,37] that can arbitrarily map failures and faults. Recent RTP research mostly uses real failures, e.g., analyzing real regression testing runs from continuous integration systems [14,15,23,24,27,34], making it rather difficult to precisely derive the failure-to-fault matrix. As a result, the increasingly popular failure-to-fault matrices are *all-to-one*, where all failures map to the same one fault, and *one-to-one*, where each failure maps to a distinct fault.

To describe allowed orders, we note that real test suites often partition tests, e.g., in JUnit [20], each test method belongs to a test class. Traditional research ignores this partitioning and allows all $n!$ orders ($\Omega_a(T)$ for short) of n tests. We introduced *compatible*[4] orders [46] ($\Omega_c(T)$ for short) that consider the partitioning and allow only orders that do not interleave tests from different classes.

We present the first theoretical analysis for the cases most commonly used in RTP research. We introduce an algorithm for efficiently computing the exact probability mass functions (PMFs) of α for all failure-to-fault matrices and $\Omega_a(T)$. We demonstrate the efficiency of our algorithm on the benchmarks from

[4] Our original term was *class-compatible* [46] because we considered as tests only test methods in test classes, but the concept easily generalizes to other kinds of tests.

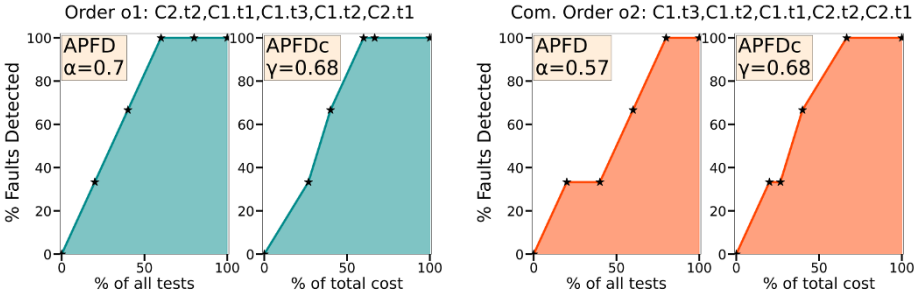

Fig. 1: Example metrics for two orders (Com. is compatible) for $n = 5, m = 3$; class C1 has 3 tests with costs $\langle 40, 20, 60 \rangle$, class C2 has 2 with $\langle 100, 80 \rangle$; C1.t1 detects fault F1; C1.t3 detects F2; C2.t1 detects F2 and F3; C2.t2 detects F3.

the largest RTP dataset for Java projects [34]. For the common all-to-one and one-to-one cases, we further derive a closed-form formula and a good approximation, respectively. We also derive closed-form formulas for the expected values for both α and γ for the general failure-to-fault matrix, for both $\Omega_a(T)$ and $\Omega_c(T)$, and we compare these values in various scenarios. Interestingly, on average, $\Omega_a(T)$ can perform much better (up to 1/2) than $\Omega_c(T)$ for certain scenarios, but cannot perform much worse (only up to 1/6) for any scenario; Section 5.1 presents this comparison, including two scenarios near the limits (1/2 and 1/6).

We finally derive two interesting properties for the α and γ metrics. Using these properties, we revisit some of the highly cited papers on RTP and find that some presented results may be biased due to insufficient sampling. Overall, our theoretical analysis provides new insights into the random RTP widely used in prior work but only via empirical sampling. Our results show that in many cases researchers need not run sampling but can use simple formulas or algorithms to obtain more precise statistics for the random RTP metrics.

2 Preliminaries

Our notation largely follows the prior work that introduced APFD (α) [38] and APFD$_c$ (γ) [11], but we make explicit the failure-to-fault matrix. Let n be the number of tests and m be the number of faults detected by (some of) these tests. Let M be a failure-to-fault matrix, i.e., a $n \times m$ Boolean matrix such that $M_{j,i} =$ true iff (failure of) test j detects fault i, and each fault has at least one failure (i.e., $\forall i.\exists j.M_{j,i}$). Let T be the set of tests in the test suite. We denote the set of tests that detect the fault i as $T_i = \{j | M_{j,i}\}$. In general, T_i and $T_{i'}$ for $i \neq i'$ need not be disjoint because one failing test can detect multiple faults. The total number of failures is $k = |\{j | \exists i.M_{j,i}\}|$, and we use $k_i = |T_i|$.

For an order o (a permutation of T), we use $<_o$ to compare the positions of two tests t and t' in the order: $t <_o t'$ denotes that t precedes t' in o, and $t \leq_o t'$ denotes that $t = t'$ or $t <_o t'$. We denote the j^{th} test in an order o as $t_j(o)$. Let $\tau_i(o) = \min_j M_{t_j(o),i}$ be the position of the first test to detect the fault i in o. Prior work [11,38] defined metrics α and γ (using the notation TF instead of τ).

We use $\alpha(o)$ and $\gamma(o)$ to indicate α and γ, respectively, for a given order o. We drop o from $<_o$, $\leq_o$, $t_j(o)$, $\tau_i(o)$, $\alpha(o)$, and $\gamma(o)$ when clear from the context.

The most popular RTP metric is α [38], defined for an order o as follows.

Definition 1 (α)**.** APFD *is defined as*

$$\alpha = 1 - \frac{\sum_{i=1}^{m} \tau_i}{nm} + \frac{1}{2n} \tag{1}$$

Plotting the percentage of faults detected against the percentage of executed tests, α represents the area under the curve, as shown in two examples in Fig. 1. The diagonal lines interpolate the percentage of faults detected and lead to nice properties of mean/median α values and symmetry (Section 6). α ranges between 0 and 1, more precisely between $1/(2n)$ and $1-1/(2n)$. A larger α indicates that an order detects faults earlier, on average.

While α effectively considers the number of tests, the "cost cognizant" metric γ considers the cost of tests [11]. The cost can be measured in various ways, but most work uses the test runtime. We use $\sigma(t)$ to denote the cost (runtime) of a test t; the total cost of a set of tests T is $\sigma(T) = \sum_{t \in T} \sigma(t)$.

Definition 2 (γ)**.** APFD_c *is defined as*

$$\gamma = \frac{\sum_{i=1}^{m} \left(\sum_{j=\tau_i}^{n} \sigma(t_j) - \frac{1}{2}\sigma(t_{\tau_i}) \right)}{m \cdot \sigma(T)} \tag{2}$$

Plotting the percentage of faults detected against the percentage of total test-suite cost, γ represents the area under the curve, as shown in Fig. 1. Note that α can be viewed as a special case of γ where $\forall t, t' \in T.\sigma(t) = \sigma(t')$.

In practice, tests often belong to *classes*[5]—e.g., JUnit [20] test methods belong to test classes, Maven [28] test classes belong to modules, and pytest [35] test functions belong to test files—and tests from each class run together. Our prior work [46] defined *compatible* orders as those where all tests from each class are consecutive. We use T_C to denote the set of tests in a class C. An order o is compatible iff $\forall C, j \leq j' \leq j''.t_j(o) \in T_C \wedge t_{j''}(o) \in T_C \Rightarrow t_{j'}(o) \in T_C$. For example, $o2$ in Fig. 1 is compatible, while $o1$ is not. To distinguish the cases for all orders from the cases for only compatible orders, we use the subscripts $_a$ and $_c$, respectively, e.g., $\text{E}_\text{a}[x]$ and $\text{E}_\text{c}[x]$ represent the expected value of x for the uniform selection of all orders and compatible orders, respectively, and $P_a(A)$ and $P_c(A)$ represent the probability of event A for the uniform selection of all orders and compatible orders, respectively. We denote the set of all orders and all compatible orders for T as $\Omega_a(T)$ and $\Omega_c(T)$, respectively [46].

We analyze RTP techniques in *scenarios*, each of which consists of a test suite with n tests, m faults, the failure-to-fault matrix, the cost of each test, and for $\Omega_c(T)$ the class of each test. To analyze compatible orders, we introduce some new notation to indicate the class of tests. We use $T_{i,C} = T_i \cap T_C$ to denote the

[5] The term *class* for a set of tests that run together need not represent a test class.

set of tests in class C that detect the fault i. Let $\mathcal{C}$ be the set of all classes, and $\mathcal{C}_i$ be the set of classes that contain at least one test that detects the fault i, i.e., $\mathcal{C}_i = \{C \in \mathcal{C} | T_{i,C} \neq \emptyset\}$. Let $C(t)$ be the class that t belongs to, i.e., $t \in T_{C(t)}$. The number of compatible orders is $|\Omega_c(T)| = |\mathcal{C}|! \prod_{C \in \mathcal{C}} |T_C|!$.

For a set of orders S, be it $\Omega_a(T)$ or $\Omega_c(T)$, the probability mass function (PMF) of a metric, α or γ, is a function p from the metric value to its probability: $p(x) = P(\text{metric} = x) = |\{o \in S | \text{metric}(o) = x\}|/|S|$. We next derive some PMFs as all prior RTP work shows only sampled distributions of random RTP.

3 PMF of α

To analyze the PMF of the metric α, we first propose an algorithm to calculate the PMF of α for the general case of M. We then discuss two special cases, i.e., all-to-one and one-to-one, which are the most common in recent RTP research.

3.1 Algorithm to Calculate PMF of α for the General Case

To calculate the PMF of α, a naïve algorithm would enumerate all $n!$ orders and compute α for each order. In theory, α can take $O(n!)$ different values, e.g., when $m = \sum_{i=1}^{n} n^i$ and all n tests fail and detect $n, n^2, \ldots, n^n$ different faults, then each of the $n!$ orders has a different α. In practice, however, the number of faults m and the number of failing tests k are usually small, e.g., in our evaluation dataset [34], 2906 out of 2980 (98%) scenarios have $k \leq 10$. We present an algorithm that computes the exact PMF with $O(n^2 mk \cdot k!)$ time complexity. Despite the $k!$ factor, the algorithm runs in reasonable time in practice, under 30sec for any of the 2906 scenarios. When $k > 10$, one can resort to sampling.

We next describe the intuition for our algorithm. $\sum_{i=1}^{m} \tau_i$ is the only part of α that depends on the (test-suite) order, so we first calculate the PMF of this sum and then convert it to the PMF of α. Iterating over the faults does *not* lead to a nice recursive formulation. Our key insight is to instead *iterate over the positions of all k failing tests.* We view $\sum_{i=1}^{m} \tau_i$ as a *weighted sum*

$$\sum_{i=1}^{m} \tau_i = \sum_{j=1}^{k} w_j \phi_j \tag{3}$$

where ϕ_j is the position of the j^{th} failing test in the order, and $w_j \geq 0$ is the weight, calculated as the number of faults detected *first* by the j^{th} failing test (Line 11 of Algorithm 1). For example, consider the order $o1$ in Fig. 1. The *relative order* of the $k = 4$ failing tests is $\rho = \langle \text{C2.t2}, \text{C1.t1}, \text{C1.t3}, \text{C2.t1} \rangle$; we use metavariable ρ to distinguish the notation from o for the order of all n tests. For this relative order, $w = \langle 1, 1, 1, 0 \rangle$ because the $m = 3$ faults are detected first by C2.t2, C1.t1, and C1.t3. The positions for this relative order ρ are $\phi = \langle 1, 2, 3, 5 \rangle$ because the 4 failing tests in ρ appear in these positions in the order $o1$.

We call a $\phi = \langle \phi_1 \ldots \phi_k \rangle$ *valid* if $1 \leq \phi_1 < \ldots < \phi_k \leq n$. Both sequences ϕ and $w = \langle w_1 \ldots w_k \rangle$ can vary for different orders. While ϕ has $\binom{n}{k}$ valid

Algorithm 1: Calculate the PMF of α

```
1  Input: n,m,M          // the number of tests and faults, and the failure-to-fault matrix
2  Output: p             // the PMF of α: p(x) = P(α = x)
3  Function PMF()        // main function; return the PMF of α for all orders
4      k = |{j|∃i.M_{j,i}}|          // number of failing tests in M, in practice k≪n
5      q = PMF_sum()                 // compute the PMF of Σ_{i=1}^{m} τ_i
6      return λx.q(mn − mnx + m/2)   // convert that PMF to the PMF of α
7  Function PMF_sum()    // return the PMF of Σ_{i=1}^{m} τ_i for all orders
8      P = ⟨PMF_rorder(ρ), ∀ρ ∈ perms({j|∃i.M_{j,i}})⟩   // enumerate all relative orders
9      return λx. Σ_{p∈P} p(x)/|P|   // average PMFs of Σ_{i=1}^{m} τ_i for each relative order
10 Function PMF_rorder(ρ)    // return the PMF of Σ_{i=1}^{m} τ_i for a relative order ρ
11     w = ⟨|{i|M_{ρ_j,i} ∧ ∄j′ < j.M_{ρ_{j′},i}}|, ∀j ∈ 1..k⟩   // w are the weights in formula (3)
12     return λs.f(w, k, n)(s)/C(n,k)   // the total number of φ is C(n,k)
13         // the function should be memoized to reuse the results for the repeated w,g,h
       Function f(w, g, h)   // return f_{g,h} given weights w, calculated with formula (4)
14     if g > h then
15         return λs.0
16     if g = 0 then
17         return λs.1_{s=0}
18     return λs.f(w, g, h − 1)(s)+f(w, g − 1, h − 1)(s − w_g h)
```

possibilities, we note that w has at most $k!$ possibilities (with $k! \ll \binom{n}{k}$ as $k \ll n$ in practice) because w depends only on ρ. Therefore, we first fix w by enumerating the $k!$ relative orders of the k failing tests. Then for each relative order, the problem of calculating the PMF of $\sum_{i=1}^{m} \tau_i = \sum_{j=1}^{k} w_j \phi_j$ becomes *"given w, count the number of valid ϕ such that $\sum_{j=1}^{k} w_j \phi_j = s$ for each s"*, which can be solved recursively as follows.

Let $f_{g,h}(s)$ be the number of assignments for the values of $\phi_1, \ldots, \phi_g$ such that $1 \le \phi_1 < \ldots < \phi_g \le h$ and $\sum_{j=1}^{g} w_j \phi_j = s$. The problem is to find $f_{k,n}(s)$. As the base case, (1) $f_{g,h}(s) = 0$ for $g > h$ because $\phi_g < g$ cannot hold; (2) $f_{0,h}(s) = \mathbf{1}_{s=0}$, where $\mathbf{1}$ is the indicator function, because only the empty sequence $\langle\rangle$ is valid and $\sum_{j=1}^{0} w_j \phi_j = 0$. For all $h \ge g > 0$, the number of assignments for $f_{g,h}(s)$ has two cases: (1) if $\phi_g \le h - 1$, the number is equal to $f_{g,h-1}(s)$ by definition; (2) if $\phi_g = h$, the number for s is equal to the number of assignments for $\phi_1, \ldots, \phi_{g-1}$ such that $\phi_{g-1} \le \phi_g - 1 = h - 1$ and $\sum_{j=1}^{g-1} w_j \phi_j = (\sum_{j=1}^{g} w_j \phi_j) - w_g \phi_g = s - w_g h$, which is $f_{g-1,h-1}(s - w_g h)$. In total,

$$f_{g,h}(s) = \begin{cases} 0 & g > h \\ \mathbf{1}_{s=0} & g = 0 \\ f_{g,h-1}(s) + f_{g-1,h-1}(s - w_g h) & \text{otherwise} \end{cases} \tag{4}$$

After solving $f_{k,n}$, we get the PMF of $\sum_{i=1}^{m} \tau_i$ for each relative order of the k failing tests. Because each of $k!$ relative orders has the same probability by symmetry, we simply take the average of their PMFs to get the PMF of $\sum_{i=1}^{m} \tau_i$ for all orders. Finally, we convert the PMF of $\sum_{i=1}^{m} \tau_i$ to the PMF of α.

Table 1: Number of tests, failures, runtime (in ms), and Jensen-Shannon (JS) distance for 10 largest scenarios [34] and one synthetic scenario (TSmax)

Test suite	**#Tests** (n)	**#Failures** (k)	**Runtime [ms]** **all-to-one**	**one-to-one**	**Jensen-Shannon distance** (§3.2.2)
TS1	2118	1	513	505	0.0000
TS2	1986	2	563	629	0.0005
TS3	2080	3	617	871	0.0003
TS4	1929	4	680	1147	0.0004
TS5	1795	5	731	1408	0.0006
TS6	339	6	627	732	0.0040
TS7	465	7	678	756	0.0034
TS8	813	8	829	2009	0.0023
TS9	52	9	1496	1846	0.0442
TS10	161	10	10989	27095	0.0150
TSmax	2118	10	32801	242400	0.0011

We next describe Algorithm 1 in more detail. The input is the number of tests n, the number of faults m, and the failure-to-fault matrix M. The main function `PMF` invokes `PMF_sum` to get the PMF of $\sum_{i=1}^{m} \tau_i$ and converts it to the PMF of α. The function `PMF_sum` enumerates all relative orders ρ of the k failing tests, invokes `PMF_rorder(`ρ`)` to get the PMF of $\sum_{i=1}^{m} \tau_i$ for each relative order, and averages these PMFs to get the PMF of $\sum_{i=1}^{m} \tau_i$ for all (relative) orders. Function `PMF_rorder(`ρ`)` computes the weights w from formula (3), invokes `f(w,k,n)` to get $f_{k,n}$ for w, and converts it to the PMF of $\sum_{i=1}^{m} \tau_i$.

We finally discuss the time complexity and the empirical performance of Algorithm 1. The major cost comes from computing the function `f`. Because there are $O(k!)$ different w and $0 \leq g \leq k, g \leq h \leq n$, we have $O(nk \cdot k!)$ different inputs for which to compute `f`. With memoization, `f` is computed only once for each input. Each computation takes $O(nm)$ because $|\text{support}(f_{g,h})| = O(nm)$ as $1 \leq \tau_i \leq n$ for $1 \leq i \leq m$. Therefore, the cost of computing `f` for all inputs is $O(n^2 mk \cdot k!)$. The other costs in the algorithm are lower than the cost of `f`; hence, the overall time complexity of Algorithm 1 is $O(n^2 mk \cdot k!)$.

Implementation: While top-down recursion makes it easier to present the algorithm, for better performance our implementation uses bottom-up dynamic programming to compute `f`. Our implementation fits in only 117 lines of C++.

Dataset: We use the RTP dataset with the most Java projects [34] for our evaluation. In this dataset, each test is a test class and each class is a Maven module [28]. The dataset has 2980 scenarios, and 2906 (98%) have $k \leq 10$. We select, for each $k \leq 10$, the scenario with the maximum number of tests (n) from the dataset. We also make a synthetic scenario with 2118 tests, being the largest number of tests in the dataset, and 10 failures. We use both all-to-one and one-to-one failure-to-fault matrices on the selected scenarios.

Evaluation: As Table 1 shows, the code finishes in under 30sec (on a common laptop) for all real scenarios; it takes more time on the synthetic one for all-to-one and one-to-one, but the runtime is still 33sec and 4min, respectively.

3.2 PMFs of α for Special Cases

As mentioned in Section 1, recent RTP research uses real failures and faults, with two kinds of failure-to-fault matrices: all-to-one and one-to-one. We discuss the PMFs of α for these two commonly used cases.

3.2.1 All-to-One: We first derive the PMF of α for all-to-one. In this case, $m = 1$, $k \geq 1$, and $w_1 = 1, \forall j > 1.w_j = 0$ in formula (3). Therefore, the recursive formula (4) becomes $f_{g,h}(s) = f_{g,h-1}(s) + f_{g-1,h-1}(s)$ for $g > 1$, which is similar to Pascal's triangle. This observation hints that the PMF of α for all-to-one may have a closed formula with binomial coefficients.

Theorem 3 (The PMF of α for all-to-one failure-to-fault matrix)**.**

$$P(\alpha = 1 - \frac{s}{n} + \frac{1}{2n}) = \frac{\binom{n-s}{k-1}}{\binom{n}{k}}, s \in \{1, 2, \ldots, n-k+1\} \tag{5}$$

Proof. For all-to-one, the α value depends solely on τ_1, which is essentially ϕ_1 in formula (3). For $1 \leq s \leq n-k+1$, $\tau_1 = s$ holds as long as $s = \phi_1 < \ldots < \phi_k \leq n$. To satisfy the condition, we just need to choose the $k-1$ positions after position s. Therefore, $\binom{n-s}{k-1}$ out of $\binom{n}{k}$ ways to choose k positions in n satisfy the condition, so $P(\tau_1 = s) = \binom{n-s}{k-1}/\binom{n}{k}$, and formula (5) directly follows. □

With (5), we can use $O(n)$ time to compute the PMF of α for all-to-one. We can compute the needed binomial coefficients iteratively, starting from $\binom{k-1}{k-1} = 1$, with the recurrence $\binom{n'+1}{k-1} = \frac{n'+1}{n'-k+2}\binom{n'}{k-1}, n' \geq k-1$, and get $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$.

3.2.2 One-to-One: We next consider the PMF of α for one-to-one. In this case, $m = k$ and each failing test finds a distinct fault, so for every relative order of the k failing tests, $\forall j.w_j = 1$ in formula (3). Therefore, running Algorithm 1 and memoizing on w, the complexity becomes $O(n^2k^2 + k!)$. $k!$ is because we need to iterate through all the relative orders. We can avoid $k!$ if we check in advance that the failure-to-fault matrix is one-to-one, so the complexity is $O(n^2k^2)$.

Moreover, considering formula (4) when $\forall j.w_j = 1$, $f_{k,n}$ essentially models the problem *"counting the number of partitions of s into k distinct summands from $\{1, 2, \ldots, n\}$"*. Specifically, $f_{g,h}(s)$ can be viewed as the number of partitions of s into g distinct summands in $\{1, 2, \ldots, h\}$, and $f_{g,h}(s) = f_{g,h-1}(s) + f_{g-1,h-1}(s-h)$ holds because the summand g can be either less than h or exactly h, corresponding to $f_{g,h-1}(s)$ and $f_{g-1,h-1}(s-h)$, respectively. To the best of our knowledge, no closed formula is known for this problem. Considering that in our evaluation dataset, 99.8% (2975/2980) of scenarios have $n^2k^2 < 10^9$, the $O(n^2k^2)$ algorithm is efficient enough for practical use for almost all cases.

Approximation: Furthermore, we can approximate the PMF by ignoring the distinct-number constraint, i.e., *"counting the number of partitions of s into k summands from $\{1, 2, \ldots, n\}$"*. This problem has a nice generating function $(x + x^2 + \ldots + x^n)^k$, where the coefficient of x^s is the number of partitions [43]:

$$\sum_{i=0}^{\lfloor \frac{s-k}{n} \rfloor} \binom{k}{i} (-1)^i \binom{s - ni - 1}{k - 1} \tag{6}$$

We can calculate these coefficients using two algorithms with different tradeoffs. The first algorithm first pre-calculates the binomial coefficients with Pascal's triangle and then calculates all the coefficients with formula (6). The first step takes $O(nk^2)$ because $s - ni - 1 \leq nk$ and $i \leq k$. The second step takes $O(nk^2)$ because each of $O(nk)$ coefficients takes $O(k)$ to compute as $\lfloor \frac{s-k}{n} \rfloor \leq k$. Thus, the overall time complexity of the first algorithm is $O(nk^2)$. The second algorithm calculates the generating function directly with the fast Fourier transform [4] by first converting $x + x^2 + \ldots + x^n$ to the point-value representation, calculating each point value to the k^{th} power, and interpolating to get the coefficients. The second algorithm takes $O(nk \log(nk))$ because the length of the polynomial is $O(nk)$. Comparing the complexity, the first algorithm is better when k is small compared to n (i.e., $k - \log k < \log n$), and the second is better otherwise.

To evaluate the approximation, we use Jensen–Shannon (JS) distance [16] between the exact and the approximated PMFs. We check our approximation on the same real scenarios as in Section 3.1. As Table 1 shows, the approximation yields PMFs with a small JS distance, the largest only 0.0442 for $n = 52, k = 9$.

3.3 PMF of γ

The PMF of γ is more complex than that of α because even for the simplest all-to-one failure-to-fault matrix, the number of possible values of γ can be $\Omega(2^n)$. For example, consider n tests with costs $1, 2, 4, \ldots, 2^{n-1}$, and only one test fails and detects the only fault. The γ value depends on the sum of the costs of the tests that precede the failure. 2^{n-1} different sets of the tests can precede the failure, and every set has a distinct sum of the costs. Even for the example in Fig. 1, the support of PMF for γ (33) is much bigger than that for α (8).

4 Expected Values for All Orders $\Omega_a(T)$

While some comparisons of RTP techniques use full samples of PMFs, many use just the arithmetic mean of the samples. We next derive formulas for expected values to obtain the mean faster and without the imprecision from sampling.

In this section, we consider the case where order o is uniformly selected from $\Omega_a(T)$, allowing $n!$ orders of n tests. Because α is a special case of γ where $\forall t, t' \in T.\sigma(t) = \sigma(t')$, we first derive γ.

To start with a simple example, consider a test suite with only one failing test ($k = 1$). For a random order, the test can be at any position with equal probability. Intuitively, the expected position across all of the orders is at the middle of the sequence, hence α and γ should be about 1/2. In fact, we will show that they are exactly 1/2. Moreover, the expected values of both α and γ are 1/2 as long as each fault is detected by only one failing test ($\forall i.k_i = |T_i| = 1$, which includes one-to-one). In general, the failure-to-fault matrix can be more complex: many tests could detect the same fault, and a test could detect many faults. To compute the expected values of α and γ, we first prove a useful lemma.

Lemma 4. *For every fault i,*

$$\forall t \notin T_i . P_a(t < t_{\tau_i}) = P_a(\forall t' \in T_i . t < t') = \frac{1}{k_i + 1} \tag{7}$$

Proof. Since τ_i is the position of the first test from T_i in the order, t precedes t_{τ_i} iff t precedes every $t' \in T_i$. Consider the relative position of each $t \notin T_i$ with respect to all the tests from T_i in a random order. By symmetry, it is equally likely that t is in any of the $k_i + 1$ relative positions created by the relative order of the k_i tests from T_i. Therefore, the probability that t is in the relative position preceding all the k_i tests from T_i is $\frac{1}{k_i+1}$. □

We first use this lemma to compute $\mathrm{E_a}[\gamma]$.

Theorem 5 (The expected value of γ for $\Omega_a(T)$)**.**

$$\mathrm{E_a}[\gamma] = 1 - \frac{\sum_{i=1}^{m} \left(\frac{\sigma(T \setminus T_i)}{k_i+1} + \frac{\sigma(T_i)}{2k_i} \right)}{m \cdot \sigma(T)} \tag{8}$$

Proof. From (2), the two key terms in γ are $\sigma(t_{\tau_i})$ and $\sum_{j=\tau_i}^{n} \sigma(t_j)$. By symmetry, any test $t \in T_i$ can be the first in the order, or equivalently $t = t_{\tau_i}$, with probability $\frac{1}{k_i}$. Thus

$$\mathrm{E_a}[\sigma(t_{\tau_i})] = \sum_{t \in T_i} P(t = t_{\tau_i}) \sigma(t) = \frac{\sigma(T_i)}{k_i} \tag{9}$$

Next, consider that $\sum_{j=\tau_i}^{n} \sigma(t_j) = \sum_{t \in T} \sigma(t) \mathbf{1}_{t_{\tau_i} \le t}$ can be also calculated as $\sum_{t \in T_i} \sigma(t) \mathbf{1}_{t_{\tau_i} \le t} + \sum_{t \notin T_i} \sigma(t) \mathbf{1}_{t_{\tau_i} \le t}$. For every test $t \in T_i$, $t_{\tau_i} \le t$ by definition, so $\forall t \in T_i . \mathrm{E_a}[\mathbf{1}_{t_{\tau_i} \le t}] = 1$. For every test $t \notin T_i$, $\mathrm{E_a}[\mathbf{1}_{t_{\tau_i} \le t}] = P_a(t_{\tau_i} \le t) = 1 - P_a(t < t_{\tau_i}) = \frac{k_i}{k_i+1}$. The last equality stems from Lemma 4. Therefore, by the linearity of expectation, we get

$$\mathrm{E_a}[\sum_{j=\tau_i}^{n} \sigma(t_j)] = \sigma(T_i) + \frac{k_i}{k_i + 1} \sigma(T \setminus T_i) \tag{10}$$

From (2), (9), and (10), we get (8). □

Corollary 5.1 (The expected value of α for $\Omega_a(T)$)**.**

$$\mathrm{E_a}[\alpha] = 1 - \frac{(n+1) \sum_{i=1}^{m} \frac{1}{k_i+1}}{nm} + \frac{1}{2n} \tag{11}$$

Revisiting the case where each fault can be detected by only one failing test, setting $\forall i . k_i = 1$ in (8) or (11), gives exactly $1/2 = \mathrm{E_a}[\alpha] = \mathrm{E_a}[\gamma]$. In fact, even in the general case of any failure-to-fault matrix, we find that the two expected values are similar if not the same, inspiring us to derive the following bound:

Theorem 6 (The expected difference of α and γ for $\Omega_a(T)$)**.**

$$-\frac{1}{12} < \mathrm{E_a}[\alpha] - \mathrm{E_a}[\gamma] < \frac{1}{2n} \tag{12}$$

Proof. From formulas (8) and (11), we have $\mathrm{E_a}[\alpha] - \mathrm{E_a}[\gamma] = \Delta_\gamma - \Delta_\alpha + \frac{1}{2n}$, where $\Delta_\gamma = \frac{\sum_{i=1}^{m}(\frac{1}{2k_i} - \frac{1}{k_i+1})\sigma(T_i)}{m \cdot \sigma(T)}$ and $\Delta_\alpha = \frac{\sum_{i=1}^{m} \frac{1}{k_i+1}}{nm}$. Since $k_i \geq 1$, we have $-\frac{1}{12} \leq \frac{1}{2k_i} - \frac{1}{k_i+1} \leq 0$ (with basic calculus, minimum is for $k_i = 2$ or $k_i = 3$), which, combined with $\sigma(T_i) \leq \sigma(T)$, gives $-\frac{1}{12} \leq \Delta_\gamma \leq 0$. Since $k_i \geq 1$, we also have $0 < \frac{1}{k_i+1} \leq \frac{1}{2}$, which gives $0 < \Delta_\alpha \leq \frac{1}{2n}$. Thus, we have $-\frac{1}{12} \leq \Delta_\gamma - \Delta_\alpha + \frac{1}{2n} < \frac{1}{2n}$. However, $\Delta_\gamma - \Delta_\alpha + \frac{1}{2n} = -\frac{1}{12}$ would require $\Delta_\alpha = \frac{1}{2n}$ and thus $\forall i.k_i = 1$, in which case $\Delta_\gamma = 0$ and $\Delta_\gamma - \Delta_\alpha + \frac{1}{2n} = 0 \neq -\frac{1}{12}$. Therefore, the equality cannot hold and $-\frac{1}{12} < \mathrm{E_a}[\alpha] - \mathrm{E_a}[\gamma] < \frac{1}{2n}$. □

5 Expected Values for Compatible Orders $\Omega_c(T)$

In this section, we consider the expected values of α and γ for $\Omega_c(T)$. Compatible orders do not interleave tests from different classes, as defined in Section 2. Similar to $\Omega_a(T)$, we first prove a useful lemma for $\Omega_c(T)$.

Lemma 7. *For every fault i, (note that if $t \notin T_i$, $C(t)$ may have another $t' \in T_i$)*

$$\forall t \notin T_i . P_c(t < t_{\tau_i}) = P_c(\forall t' \in T_i . t < t') = \begin{cases} \frac{1}{|\mathcal{C}_i|(|T_{i,C(t)}|+1)} & C(t) \in \mathcal{C}_i \\ \frac{1}{|\mathcal{C}_i|+1} & C(t) \notin \mathcal{C}_i \end{cases} \tag{13}$$

Proof. For $C(t) \in \mathcal{C}_i$ case, two conditions must hold for $t \notin T_{i,C(t)}$ to precede all tests that detect the fault i. First, among all classes in $\mathcal{C}_i$, $C(t)$ must be the first in the order, and by symmetry, each class in $\mathcal{C}_i$ can be the first with the same probability $\frac{1}{|\mathcal{C}_i|}$. Second, t must precede all tests from $T_{i,C(t)}$, which (similar to Lemma 4) holds with the probability $\frac{1}{|T_{i,C(t)}|+1}$. The two conditions are independent because they are about the class order and the test order inside the class, respectively, and these orders are independent of each other. Therefore, the probability that t precedes the first test that detects the fault i is $\frac{1}{|\mathcal{C}_i|(|T_{i,C(t)}|+1)}$.

For $C(t) \notin \mathcal{C}_i$ case, only one condition—$C(t)$ precedes all classes in $\mathcal{C}_i$—must hold for t to precede the first test that detects the fault i, which (similar to Lemma 4) happens with probability $\frac{1}{|\mathcal{C}_i|+1}$. □

Theorem 8 (The expected value of γ for $\Omega_c(T)$)**.**

$$\mathrm{E_c}[\gamma] = 1 - \frac{1}{m \cdot \sigma(T)} \sum_{i=1}^{m} \left(\frac{\sum_{C \notin \mathcal{C}_i} \sigma(T_C)}{|\mathcal{C}_i|+1} + \frac{1}{|\mathcal{C}_i|} \sum_{C \in \mathcal{C}_i} \left(\frac{\sigma(T_C \setminus T_{i,C})}{|T_{i,C}|+1} + \frac{\sigma(T_{i,C})}{2|T_{i,C}|} \right) \right) \tag{14}$$

Proof. We first compute the two key terms $\sigma(t_{\tau_i})$ and $\sum_{j=\tau_i}^{n}\sigma(t_j)$ in γ. For each test $t \in T_i$ to be the first, its class $C(t) \in \mathcal{C}_i$ should be the first among all classes in $\mathcal{C}_i$ with probability $\frac{1}{|\mathcal{C}_i|}$, and t must be the first among all tests in $T_{i,C(t)}$ with probability $\frac{1}{|T_{i,C(t)}|}$. These two events are independent, so the joint probability is $\frac{1}{|\mathcal{C}_i||T_{i,C(t)}|}$. By $\sigma(t_{\tau_i}) = \sum_{t\in T_i}\sigma(t)\cdot\mathbf{1}_{t=t_{\tau_i}}$, we have

$$\mathrm{E_c}[\sigma(t_{\tau_i})] = \sum_{t\in T_i}\frac{\sigma(t)}{|\mathcal{C}_i||T_{i,C(t)}|} = \frac{1}{|\mathcal{C}_i|}\sum_{C\in\mathcal{C}_i}\frac{\sigma(T_{i,C})}{|T_{i,C}|} \tag{15}$$

Next, consider $\sum_{j=\tau_i}^{n}\sigma(t_j) = \sum_{t\in T}\sigma(t)\cdot\mathbf{1}_{t_{\tau_i}\le t}$. Each t is either (1) $t \in T_i$, where $\mathbf{1}_{t_{\tau_i}\le t} = 1$ by definition of τ_i; or (2) $t \notin T_i$, where $\mathrm{E_c}[\mathbf{1}_{t_{\tau_i}\le t}] = \mathrm{E_c}[\mathbf{1}_{t_{\tau_i}<t}] = P_c(t_{\tau_i} < t) = 1 - P_c(t < t_{\tau_i})$ can be obtained from Lemma 7. Combining these cases, we have

$$\begin{aligned}\mathrm{E_c}[\textstyle\sum_{j=\tau_i}^{n}\sigma(t_j)] = \sigma(T_i) + \tfrac{|\mathcal{C}_i|}{|\mathcal{C}_i|+1}\textstyle\sum_{C\notin\mathcal{C}_i}\sigma(T_C) + \\ \textstyle\sum_{C\in\mathcal{C}_i}\left(1 - \frac{1}{|\mathcal{C}_i|(|T_{i,C}|+1)}\right)\sigma(T_C \setminus T_{i,C})\end{aligned} \tag{16}$$

From (2), (15), and (16), we get (14). □

Corollary 8.1 (The expected value of α for $\Omega_c(T)$)**.**

$$\mathrm{E_c}[\alpha] = 1 - \frac{1}{nm}\sum_{i=1}^{m}\left(\frac{\sum_{C\notin\mathcal{C}_i}|T_C|}{|\mathcal{C}_i|+1} + \frac{1}{|\mathcal{C}_i|}\sum_{C\in\mathcal{C}_i}\frac{|T_C|+1}{|T_{i,C}|+1}\right) + \frac{1}{2n} \tag{17}$$

We next discuss the expected difference of $\mathrm{E_c}[\alpha]$ and $\mathrm{E_c}[\gamma]$. Unlike the case with $\Omega_a(T)$, where the difference has a rather small bound, we find that the difference can be rather large for $\Omega_c(T)$.

Theorem 9 (The expected difference of α and γ for $\Omega_c(T)$)**.**

$$-\frac{1}{2} < \mathrm{E_c}[\alpha] - \mathrm{E_c}[\gamma] \le \frac{1}{2} - \frac{1}{2n} \tag{18}$$

Proof. From (14) and (17), we get $\mathrm{E_c}[\alpha] - \mathrm{E_c}[\gamma] = \Delta_\gamma - \Delta_\alpha + \frac{1}{2n}$, where $\Delta_\gamma = \frac{1}{m\cdot\sigma(T)}\sum_{i=1}^{m}\left(\frac{\sum_{C\notin\mathcal{C}_i}\sigma(T_C)}{|\mathcal{C}_i|+1} + \frac{1}{|\mathcal{C}_i|}\sum_{C\in\mathcal{C}_i}\left(\frac{\sigma(T_C\setminus T_{i,C})}{|T_{i,C}|+1} + \frac{\sigma(T_{i,C})}{2|T_{i,C}|}\right)\right)$ and $\Delta_\alpha = \frac{1}{nm}\sum_{i=1}^{m}(\frac{\sum_{C\notin\mathcal{C}_i}|T_C|}{|\mathcal{C}_i|+1} + \frac{1}{|\mathcal{C}_i|}\sum_{C\in\mathcal{C}_i}\frac{|T_C|+1}{|T_{i,C}|+1})$. $\Delta_\gamma > 0$ because all the terms in Δ_γ are positive. From $\forall i, C \in \mathcal{C}_i.|\mathcal{C}_i| \ge 1, |T_{i,C}| \ge 1$, we have

$$\begin{aligned}\Delta_\gamma &\le \tfrac{1}{m\cdot\sigma(T)}\textstyle\sum_{i=1}^{m}\left(\frac{\sum_{C\notin\mathcal{C}_i}\sigma(T_C)}{1+1} + \frac{1}{1}\sum_{C\in\mathcal{C}_i}\left(\frac{\sigma(T_C\setminus T_{i,C})}{1+1} + \frac{\sigma(T_{i,C})}{2\cdot 1}\right)\right) \\ &= \tfrac{1}{m\cdot\sigma(T)}\cdot\tfrac{1}{2}\textstyle\sum_{i=1}^{m}\sigma(T) = \frac{1}{2}\end{aligned}$$

Similarly,

$$\begin{aligned}\Delta_\alpha &\le \tfrac{1}{nm}\textstyle\sum_{i=1}^{m}(\frac{\sum_{C\notin\mathcal{C}_i}|T_C|}{|\mathcal{C}_i|+1}+\frac{1}{|\mathcal{C}_i|}\sum_{C\in\mathcal{C}_i}\frac{|T_C|+1}{1+1})\\ &\le \tfrac{1}{nm}\textstyle\sum_{i=1}^{m}(\frac{\sum_{C\notin\mathcal{C}_i}|T_C|}{2|\mathcal{C}_i|}+\frac{\sum_{C\in\mathcal{C}_i}(|T_C|+1)}{2|\mathcal{C}_i|})=\frac{1}{nm}\sum_{i=1}^{m}(\frac{n}{2|\mathcal{C}_i|}+\frac{1}{2})\le\frac{n+1}{2n}\end{aligned}$$

From $0 \le |T_{i,C}| \le |T_C|$, we also have $\Delta_\alpha \ge \frac{1}{n}$. Combining $0 < \Delta_\gamma \le \frac{1}{2}$ and $\frac{1}{n} \le \Delta_\alpha \le \frac{n+1}{2n}$, we get $-\frac{1}{2} < \Delta_\gamma - \Delta_\alpha + \frac{1}{2n} \le \frac{1}{2} - \frac{1}{2n}$ □

Considering many inequalities in the preceding proof, one may expect the bounds to be loose, but we show two scenarios where bounds are close to tight. Both scenarios have only one fault. Scenario one has two classes: C_1 has only one passing test t with cost qN ($q > 0$ is arbitrary), and C_2 has N failing tests each with cost $\frac{q}{N}$. We assume $N \gg 1$. t must be the first or last in any compatible order, each with probability 1/2 (when C_1 is first or second). $\mathrm{E_c}[\alpha]$ is close to 1, and $\mathrm{E_c}[\gamma]$ is only about 1/2. Precisely, $\mathrm{E_c}[\alpha] - \mathrm{E_c}[\gamma] = \frac{N^2-2N+2}{2N^2+2N} \approx \frac{1}{2}$ when $N \gg 1$. Scenario two has two classes: C_2 has N failing tests with cost $\frac{q}{N}$, and C_3 has N^2 passing tests each with cost $\frac{q}{N^3}$. The two classes have only two orders, each with probability 1/2. $\mathrm{E_c}[\gamma]$ is close to 1, and $\mathrm{E_c}[\alpha]$ is only about 1/2. Precisely, $\mathrm{E_c}[\alpha] - \mathrm{E_c}[\gamma] = \frac{1}{N+1} - \frac{N^2+2}{2N^2+2N} + \frac{1}{2N} \approx -\frac{1}{2}$ when $N \gg 1$.

5.1 Comparison of $\Omega_a(T)$ and $\Omega_c(T)$

Orders that are compatible have more constraints on the PMF, which could increase or decrease average α or γ values. To compare how orders in $\Omega_a(T)$ and $\Omega_c(T)$ perform on average, we compare $\mathrm{E_a}[\alpha]$ with $\mathrm{E_c}[\alpha]$ and $\mathrm{E_a}[\gamma]$ with $\mathrm{E_c}[\gamma]$.

Theorem 10 (Difference of $\mathrm{E_c}[\gamma]$ and $\mathrm{E_a}[\gamma]$)**.**

$$\frac{1}{2n} - \frac{1}{2} \le \mathrm{E_c}[\gamma] - \mathrm{E_a}[\gamma] \le \frac{1}{6} \tag{19}$$

Proof. From (8) and (14), we have

$$\begin{aligned}\mathrm{E_c}[\gamma] - \mathrm{E_a}[\gamma] = \tfrac{1}{m\cdot\sigma(T)}\textstyle\sum_{i=1}^{m}\Bigg(&\frac{\sigma(T_i)}{2k_i}+\frac{\sigma(T\backslash T_i)}{k_i+1}-\frac{\sum_{C\notin\mathcal{C}_i}\sigma(T_C)}{|\mathcal{C}_i|+1}-\\ &\frac{1}{|\mathcal{C}_i|}\sum_{C\in\mathcal{C}_i}\left(\frac{\sigma(T_C\backslash T_{i,C})}{|T_{i,C}|+1}+\frac{\sigma(T_{i,C})}{2|T_{i,C}|}\right)\Bigg)\end{aligned} \tag{20}$$

Because $\forall i.1 \le k_i \le n, |C_i| \ge 1, |T_{i,c}| \ge 1$, we have

$$\mathrm{E_c}[\gamma] - \mathrm{E_a}[\gamma] \ge \tfrac{1}{m\cdot\sigma(T)}\textstyle\sum_{i=1}^{m}(\frac{1}{2n}-\frac{1}{2})\sigma(T) = \frac{1}{2n}-\frac{1}{2}$$

For the other side, because $\forall i.|C_i| \le k_i, |T_{i,c}| \le k_i$, we have

$$\begin{aligned}\mathrm{E_c}[\gamma] - \mathrm{E_a}[\gamma] &\le \tfrac{1}{m\cdot\sigma(T)}\textstyle\sum_{i=1}^{m}\Bigg(\frac{\sigma(T_i)}{2k_i}+\frac{\sigma(T\backslash T_i)}{k_i+1}-\frac{\sum_{C\notin\mathcal{C}_i}\sigma(T_C)}{k_i+1}-\\ &\qquad\frac{(\sum_{C\in\mathcal{C}_i}\sigma(T_C))-\sigma(T_i)}{|\mathcal{C}_i|(k_i+1)}-\frac{\sigma(T_i)}{2|\mathcal{C}_i|k_i}\Bigg)\\ &= \tfrac{1}{m\cdot\sigma(T)}\textstyle\sum_{i=1}^{m}\left(1-\frac{1}{|\mathcal{C}_i|}\right)\left(\frac{\sum_{C\in\mathcal{C}_i}\sigma(T_C)}{k_i+1}-\sigma(T_i)\left(\frac{1}{k_i+1}-\frac{1}{2k_i}\right)\right)\\ &\le \tfrac{1}{m\cdot\sigma(T)}\textstyle\sum_{i=1}^{m}\left(1-\frac{1}{|\mathcal{C}_i|}\right)\frac{\sum_{C\in\mathcal{C}_i}\sigma(T_C)}{k_i+1}\\ &\le \tfrac{1}{m\cdot\sigma(T)}\textstyle\sum_{i=1}^{m}\frac{|\mathcal{C}_i|-1}{|\mathcal{C}_i|(|\mathcal{C}_i|+1)}\sum_{C\in\mathcal{C}_i}\sigma(T_C)\\ &\le \tfrac{1}{m\cdot\sigma(T)}\textstyle\sum_{i=1}^{m}\frac{\sigma(T)}{6}=\frac{1}{6}\end{aligned}$$

The third last inequality holds because $\forall k_i \geq 1. \frac{1}{k_i+1} - \frac{1}{2k_i} \geq 0$. The last inequality holds because $\forall |\mathcal{C}_i| \geq 1. \frac{|\mathcal{C}_i|-1}{|\mathcal{C}_i|(|\mathcal{C}_i|+1)} \leq \frac{1}{6}$, which can be shown with simple calculus, and $\sum_{C \in \mathcal{C}_i} \sigma(T_C) \leq \sigma(T)$. □

Corollary 10.1 (Difference of $\mathrm{E_c}[\alpha]$ and $\mathrm{E_a}[\alpha]$)**.**

$$\frac{1}{2n} - \frac{1}{2} \leq \mathrm{E_c}[\alpha] - \mathrm{E_a}[\alpha] \leq \frac{1}{6} \tag{21}$$

We give two scenarios where the preceding bounds are close to tight. In both scenarios, we set $\forall t, t' \in T. \sigma(t) = \sigma(t')$, so that $\alpha = \gamma$ and $\mathrm{E_c}[\alpha] - \mathrm{E_a}[\alpha] = \mathrm{E_c}[\gamma] - \mathrm{E_a}[\gamma]$. The first scenario has one fault F, each of the $|\mathcal{C}|$ classes contains $\frac{n}{|\mathcal{C}|}$ tests, and tests from only one class detect F but all tests in that class detect F. In this scenario, $\mathrm{E_a}[\alpha] = 1 - \frac{|\mathcal{C}|(n+1)}{n(n+|\mathcal{C}|)} + \frac{1}{2n}$, and $\mathrm{E_c}[\alpha] = 1 - \frac{|\mathcal{C}|-1}{2|\mathcal{C}|} - \frac{1}{2n}$. If we consider $|\mathcal{C}| = \sqrt{n}$, when $n \gg 1$, $\mathrm{E_a}[\alpha] \approx 1$ but $\mathrm{E_c}[\alpha] \approx 1/2$, hence $\mathrm{E_c}[\alpha] - \mathrm{E_a}[\alpha] \approx -1/2$. The second scenario has one fault F and two classes with 1 and $n-1$ tests, and each class contains only one test that detects F. In this scenario, $\mathrm{E_a}[\alpha] = \frac{2}{3} - \frac{1}{2n}$ and $\mathrm{E_c}[\alpha] = \frac{3}{4}$. When $n \gg 1$, $\mathrm{E_c}[\alpha] - \mathrm{E_a}[\alpha] \approx \frac{1}{12}$, close to the upper bound of 1/6.

In brief, measured by α or γ, compatible orders can be much worse on average than all orders (up to 1/2) but cannot be much better (up to 1/6).

6 Properties of Metrics and Checking Prior RTP Work

Prior work on random RTP uses sampling and often visualizes α and γ values as *boxplots* that may show the median, mean, quartiles (25% and 75%), and "whiskers" (1.5 times the interquartile range) of the sampled distribution. For papers that show these boxplots, we identify two properties for the boxplots, focusing on $\Omega_a(T)$ because it is used in almost all prior work instead of $\Omega_c(T)$ [46]:

- **Mean/Median at Least Half:** $\mathrm{E_a}[\alpha], \mathrm{Med_a}(\alpha), \mathrm{E_a}[\gamma], \mathrm{Med_a}(\gamma) \geq 1/2$.
- **Symmetric PMF:** $\mathrm{E_a}[\alpha] = 1/2 \Leftrightarrow \mathrm{Med_a}(\alpha) = 1/2 \Leftrightarrow \mathrm{E_a}[\gamma] = 1/2 \Leftrightarrow \mathrm{Med_a}(\gamma) = 1/2 \Leftrightarrow \forall i. k_i = 1 \Rightarrow$ PMFs of α and γ are symmetric around 1/2.

To check the boxplots from prior work, we search on Google Scholar for papers related to "test prioritization" and keep only the papers that contain both "test" and "prioriti" in the titles. We sort these papers based on their citation count and check the top 100 papers with the highest citation count [44].

6.1 Mean/Median at Least Half

Lemma 11. *$\forall o \in \Omega_a(T)$ and its reverse order $\overline{o} \in \Omega_a(T)$,*

$$\gamma(o) + \gamma(\overline{o}) \geq 1 \tag{22}$$

The equality holds iff $\forall i. k_i = 1$.

Proof sketch. To give some intuition, when $\forall i.k_i = 1$, the test that detects the fault i first does not change by reversing the order, so the "prefixes" of the test in o and $\overline{o}$ complement each other and form the entire test suite. In this case, $\gamma(o) + \gamma(\overline{o}) = 1$. If $\exists i.k_i \geq 2$, the test that detects the fault i first in o is not the same test in $\overline{o}$, and the "prefixes" of these two tests in o and $\overline{o}$ do not form the entire test suite, so $\gamma(o) + \gamma(\overline{o}) > 1$. We omit the details due to space limit. □

Theorem 12 (Measures of central tendency are at least half)**.**

$$\min\{\mathrm{E_a}[\alpha], \mathrm{Med_a}(\alpha), \mathrm{E_a}[\gamma], \mathrm{Med_a}(\gamma)\} \geq 1/2 \tag{23}$$

The equality holds iff $\forall i.k_i = 1$.

Proof sketch. From (22), we get $\mathrm{E_a}[\gamma] = \frac{1}{2} \cdot \frac{\sum_{o \in \Omega_a(T)} (\gamma(o) + \gamma(\overline{o}))}{n!} \geq \frac{1}{2}$ and the equality holds iff $\forall i.k_i = 1$. Because α can be viewed as a special case of γ, we also have the same result for $\mathrm{E_a}[\alpha]$. The same result for $\mathrm{Med_a}(\alpha)$ and $\mathrm{Med_a}(\gamma)$ can also be derived from (22). We omit the details due to space limit. □

When we inspect the top 100 most cited RTP papers, we find at least five papers with boxplots clearly showing a mean or median below 1/2. These papers range from seminal papers [12, Figs. 2b, 2c, 2e] (year 2000) and [13, Fig. 3: schedule, tcas] (2002) to more recent [29, Fig. 4] (2007), [5, Fig. 2] (2016 – a co-author of this prior paper is also in this paper), and [41, Fig. 5] (2017). Instead of sampling random orders for an arbitrary number of times, future RTP research could use our formulas or algorithm to obtain correct mean and median values.

6.2 Symmetric PMF

We also prove that α and γ PMFs are symmetric when (23)'s equality holds.

Theorem 13 (Symmetry of the α and γ PMFs)**.** *If* $\mathrm{E_a}[\alpha] = 1/2 \vee \mathrm{Med_a}(\alpha) = 1/2 \vee \mathrm{E_a}[\gamma] = 1/2 \vee \mathrm{Med_a}(\gamma) = 1/2 \vee \forall i.k_i = 1$*, then*

$$\forall \delta. P(\alpha = 1/2 - \delta) = P(\alpha = 1/2 + \delta) \ \wedge \ P(\gamma = 1/2 - \delta) = P(\gamma = 1/2 + \delta) \tag{24}$$

Proof. From Theorem 12, $\min\{\mathrm{E_a}[\alpha], \mathrm{Med_a}(\alpha), \mathrm{E_a}[\gamma], \mathrm{Med_a}(\gamma)\} = 1/2 \vee \forall i.k_i = 1 \Leftrightarrow \forall i.k_i = 1 \Rightarrow \forall o.\alpha(o) + \alpha(\overline{o}) = 1 \wedge \gamma(o) + \gamma(\overline{o}) = 1$. Each order has exactly one reverse order, so the PMFs of α and γ are symmetric around 1/2. □

When we inspect the top 100 most cited RTP papers again, we find at least three papers relevant to this property. Based on the information in these papers, we believe that $\forall i.k_i = 1$ is true. Ideally, we would confirm each paper's failure-to-fault matrix, but papers often omit such details. On a positive note, the authors of one paper [38] released their dataset, which we analyze and confirm that $\forall i.k_i = 1$. The papers that violate this property include the most widely cited paper on RTP [38, Fig. 5: schedule, schedule2, tcas] (year 2001; 1563 citations per Google Scholar) and others, both older [36, Fig. 4: schedule, schedule2, tcas] (1999) and newer [40, Fig. 2] (2015) papers.

Instead of randomly sampling orders to approximate PMFs, future RTP papers could use our algorithm to compute exact PMFs. While we find only five and three papers that definitely violate Mean/Median at Least Half and Symmetric PMF, respectively, we suspect that many others may violate these or similar properties. However, due to the lack of data in many papers (e.g., no boxplot for random RTP), we cannot easily identify all violations.

7 Related Work

Some prior work [45, 49] considers expected values of α and γ but in different contexts from ours. Random testing (but *not* random RTP) has been studied for a while [7–9,17,18,31–33,50]. The most related are theoretical analyses of random test generation. Böhme and Paul [2, 3] analyze how random sampling of test inputs compares to systematic generation: random can be more efficient when the cost to systematically generate a test input exceeds the cost to randomly sample an input by some factor. Böhme et al. [1] analyze the connection between Shannon's entropy and the discovery rate of a fuzzer that randomly generates inputs. They provide the foundation for identifying random seeds for the fuzzer to improve the overall efficiency. Their analysis also enables future systematic approaches for test generation to be more efficiently compared with random. Similarly, our analysis can help future RTP work more efficiently compare against random RTP and avoid insufficient sampling. Beyond random test generation, Majumdar and Niksic [26] present a theoretical analysis on the effectiveness of randomly inserted partition faults to find bugs in distributed systems. In contrast, our analysis is on test-suite orders for random RTP.

8 Conclusion

Regression test prioritization (RTP) is a popular regression testing approach. Majority of highly cited RTP papers have compared RTP techniques with random RTP. However, all evaluations have been empirical, with no prior theoretical analysis of random RTP. This paper has presented such analysis, by introducing an algorithm for efficiently computing the exact probability mass function of APFD, deriving closed-form formulas and approximations for various metrics and scenarios, and deriving two interesting properties forAPFD and APFD_c. Overall, our analysis provides new insights into the random RTP, and our results show that future RTP work often need not use random sampling but can use our simple formulas or algorithms to more precisely evaluate random RTP.

Acknowledgments. We thank Anjiang Wei, Dezhi Ran, and Sasa Misailovic for their help. This work was partially supported by US NSF grants CCF-1763788. CCF-1956374, NSFC grant No. 62161146003, Tencent Foundation, and XPLORER PRIZE. We acknowledge support for research on regression testing from Dragon Testing, Microsoft, and Qualcomm. Tao Xie is the corresponding author, and also affiliated with Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, China.

References

1. Böhme, M., Manès, V.J.M., Cha, S.K.: Boosting fuzzer efficiency: An information theoretic perspective. In: ESEC/FSE (2020)
2. Böhme, M., Paul, S.: On the efficiency of automated testing. In: FSE (2014)
3. Böhme, M., Paul, S.: A probabilistic analysis of the efficiency of automated software testing. TSE (2016)
4. Brigham, E.O.: The fast Fourier transform and its applications. Prentice-Hall, Inc. (1988)
5. Busjaeger, B., Xie, T.: Learning for test prioritization: An industrial case study. In: FSE (2016)
6. Cheng, R., Zhang, L., Marinov, D., Xu, T.: Test-case prioritization for configuration testing. In: ISSTA (2021)
7. Claessen, K., Hughes, J.: QuickCheck: A lightweight tool for random testing of Haskell programs. In: ICFP (2000)
8. Csallner, C., Smaragdakis, Y., Xie, T.: DSD-Crasher: A hybrid analysis tool for bug finding. TOSEM (2008)
9. Duran, J.W., Ntafos, S.C.: An evaluation of random testing. TSE (1984)
10. Elbaum, S., Kallakuri, P., Malishevsky, A., Rothermel, G., Kanduri, S.: Understanding the effects of changes on the cost-effectiveness of regression testing techniques. STVR (2003)
11. Elbaum, S., Malishevsky, A., Rothermel, G.: Incorporating varying test costs and fault severities into test case prioritization. In: ICSE (2001)
12. Elbaum, S., Malishevsky, A.G., Rothermel, G.: Prioritizing test cases for regression testing. In: ISSTA (2000)
13. Elbaum, S., Malishevsky, A.G., Rothermel, G.: Test case prioritization: A family of empirical studies. TSE (2002)
14. Elbaum, S., Rothermel, G., Penix, J.: Techniques for improving regression testing in continuous integration development environments. In: FSE (2014)
15. Elsner, D., Hauer, F., Pretschner, A., Reimer, S.: Empirically evaluating readily available information for regression test optimization in continuous integration. In: ISSTA (2021)
16. Endres, D.M., Schindelin, J.E.: A new metric for probability distributions. Transactions on Information Theory (2003)
17. Fraser, G., Zeller, A.: Generating parameterized unit tests. In: ISSTA (2011)
18. Hamlet, R.: Random testing. In: Encyclopedia of Software Engineering (1994)
19. Jiang, B., Zhang, Z., Chan, W.K., Tse, T.H.: Adaptive random test case prioritization. In: ASE (2009)
20. JUnit (2022), https://junit.org
21. Kim, J.M., Porter, A.: A history-based test prioritization technique for regression testing in resource constrained environments. In: ICSE (2002)
22. Kim, J.M., Porter, A., Rothermel, G.: An empirical study of regression test application frequency. STVR (2005)
23. Liang, J., Elbaum, S., Rothermel, G.: Redefining prioritization: Continuous prioritization for continuous integration. In: ICSE (2018)
24. Lu, Y., Lou, Y., Cheng, S., Zhang, L., Hao, D., Zhou, Y., Zhang, L.: How does regression test prioritization perform in real-world software evolution? In: ICSE (2016)
25. Luo, Q., Hariri, F., Eloussi, L., Marinov, D.: An empirical analysis of flaky tests. In: FSE (2014)

26. Majumdar, R., Niksic, F.: Why is random testing effective for partition tolerance bugs? In: POPL (2017)
27. Mattis, T., Rein, P., Dürsch, F., Hirschfeld, R.: RTPTorrent: An open-source dataset for evaluating regression test prioritization. In: MSR (2020)
28. Maven (2022), https://maven.apache.org
29. Mirarab, S., Tahvildari, L.: A prioritization approach for software test cases based on Bayesian networks. In: FASE (2007)
30. Mondal, S., Nasre, R.: Summary of Hansie: Hybrid and consensus regression test prioritization. In: ICST (2021)
31. Ntafos, S.: On random and partition testing. In: ISSTA (1998)
32. Ozkan, B.K., Majumdar, R., Oraee, S.: Trace aware random testing for distributed systems. OOPSLA (2019)
33. Pacheco, C., Lahiri, S.K., Ernst, M.D., Ball, T.: Feedback-directed random test generation. In: ICSE (2007)
34. Peng, Q., Shi, A., Zhang, L.: Empirically revisiting and enhancing IR-based test-case prioritization. In: ISSTA (2020)
35. pytest (2022), https://docs.pytest.org
36. Rothermel, G., Untch, R., Chu, C., Harrold, M.: Test case prioritization: An empirical study. In: ICSM (1999)
37. Rothermel, G., Elbaum, S., Malishevsky, A., Kallakuri, P., Davia, B.: The impact of test suite granularity on the cost-effectiveness of regression testing. In: ICSE (2002)
38. Rothermel, G., Untch, R.H., Chu, C., Harrold, M.J.: Prioritizing test cases for regression testing. TSE (2001)
39. Rummel, M.J., Kapfhammer, G.M., Thall, A.: Towards the prioritization of regression test suites with data flow information. In: SAC (2005)
40. Saha, R.K., Zhang, L., Khurshid, S., Perry, D.E.: An information retrieval approach for regression test prioritization based on program changes. In: ICSE (2015)
41. Spieker, H., Gotlieb, A., Marijan, D., Mossige, M.: Reinforcement learning for automatic test case prioritization and selection in continuous integration. In: ISSTA (2017)
42. Srivastava, A., Thiagarajan, J.: Effectively prioritizing tests in development environment. In: ISSTA (2002)
43. Stanley, R.P.: Enumerative Combinatorics, Volume 1. Cambridge University Press (2011)
44. A Theoretical Analysis of Regression Test Prioritization website (2022), https://sites.google.com/view/theoretical-analysis-of-rtp
45. Wang, Z., Chen, L.: Improved metrics for non-classic test prioritization problems. In: SEKE (2015)
46. Wei, A., Yi, P., Xie, T., Marinov, D., Lam, W.: Probabilistic and systematic coverage of consecutive test-method pairs for detecting order-dependent flaky tests. In: TACAS (2021)
47. Wong, W., Horgan, J., London, S., Agrawal, H.: A study of effective regression testing in practice. In: ISSRE (1997)
48. Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: A survey. STVR (2012)
49. Zhai, K., Jiang, B., Chan, W.: Prioritizing test cases for regression testing of location-based services: Metrics, techniques, and case study. IEEE TSC (2012)
50. Zhang, S., Saff, D., Bu, Y., Ernst, M.D.: Combined static and dynamic automated test generation. In: ISSTA (2011)