Delft University of Technology
Software Engineering Research Group
Technical Report Series

A Diagnostic Approach to Test Prioritization

Alberto Gonzalez-Sanchez, Rui Abreu, Hans-Gerhard Gross and Arjan van Gemund

Report TUD-SERG-2010-007

TUD-SERG-2010-007

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
`http://www.se.ewi.tudelft.nl/techreports/`

For more information about the Software Engineering Research Group:
`http://www.se.ewi.tudelft.nl/`

Note: Submitted for review at ASE'10

A Diagnostic Approach to Test Prioritization[*]

Alberto Gonzalez-Sanchez[†] Rui Abreu[‡] Hans-Gerhard Gross[†] Arjan J.C. van Gemund[†]

[†]Department of Software Technology
Delft University of Technology
The Netherlands
{a.gonzalezsanchez, h.g.gross, a.j.c.vangemund}@tudelft.nl

[‡]Department of Informatics Engineering
Faculty of Engineering, University of Porto
Portugal
rui@computer.org

ABSTRACT

In development processes with high code production rates testing typically triggers fault diagnosis to localize the detected failures. However, current test prioritization algorithms are tuned for failure detection rate rather than diagnostic information. Consequently, unnecessary diagnostic effort might be spent to localize the faults. We present a dynamic test prioritization algorithm that trades fault detection rate for diagnostic performance, minimizing overall testing and diagnosis cost. The algorithm exploits pass/fail information from each test to select the next test, optimizing the diagnostic information produced per test. Experimental results from synthetic test suites, and suites taken from the Software-artifact Infrastructure Repository show possible diagnostic cost reductions up to 10 and 19 percent, respectively, compared to the best of random selection, FEP, and ART. The cost reduction is sensitive to the quality of the test coverage matrices and component health, but tends to grow with the number of faults.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: testing and debugging

Keywords

diagnosis, test prioritization, test coverage, information-gain.

1. INTRODUCTION

Software testing is a time-consuming but rather important task for improving software reliability. Two processes can be distinguished: (1) testing to find failures (e.g., regression tests), and (2) finding the root causes of the failures (faults, defects, bugs). Whereas the former is commonly known as "testing", the latter is commonly denoted as (fault) "diagnosis" (or debugging). Given the significant cost associated with tests, test *prioritization* has emerged as predominant technique to reduce testing cost. Test prioritization is typically aimed to find failures as soon as possible [8, 9, 14, 16, 20, 25, 27, 30]. The sooner failures are found, the sooner diagnosis can commence. In diagnosis, tests are used to deduce a list of components (e.g., functions, statements) that are highly suspect to be at fault (the diagnosis). Again, these tests can be prioritized, in this case to optimize the diagnostic yield per test, i.e., to minimize the residual work the diagnostician (developer) performs off-line, when going through the diagnosis to verify (and fix) the defects [2, 3, 19, 23].

In software development processes with high code production rates, the probability of introducing at least one defect between subsequent (regression) tests is considerable. Consequently, the probability of having to apply diagnosis after testing is high. While test prioritization minimizes the delay between testing and diagnosis, it does not maximize diagnostic yield, and therefore does not minimize the overall cost of the combined process given defective code. The reason is that test prioritization aims at high code coverage, whereas diagnosis aims at partially revisiting already covered code to further exonerate or indict defective components. This drawback of test prioritization has indeed been addressed in recent work [12, 17, 31].

In this paper we study a novel, dynamic approach to test prioritization, dubbed *diagnostic prioritization*, that aims to minimize the overall cost of testing and diagnosis. Let $C_t(N)$ denote the aggregate time cost of testing where N is the number of tests. Let $C_d(N)$ denote the time cost associated with the diagnostic work performed by the software developer to debug the actual defects. Typically, $C_d(N)$ has a geometric decreasing shape [3, 10]. The essential motivation behind diagnostic prioritization is the reduction of $C_d(N)$. The overall time cost C of the combined testing and diagnosis process can be modelled by

$$C(N) = C_t(N) + \alpha \cdot C_d(N) \quad (1)$$

where α models the possible factor between the cost of the (regression) tests C_t and the cost of the inspection tests performed by the developer (typically high). Figure 1 illustrates the difference between a classic test prioritization scenario (a), and a diagnostic test prioritization scenario (b), where, for the sake of exposition simplicity, we assume all tests have the same cost, and $\alpha = 1$ (no distinction between the cost of testing and inspection). Per scenario three variables are

[*]This work has been carried out as part of the Poseidon project under the responsibility of the Embedded Systems Institute (ESI), Eindhoven, The Netherlands. This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK03021 program.

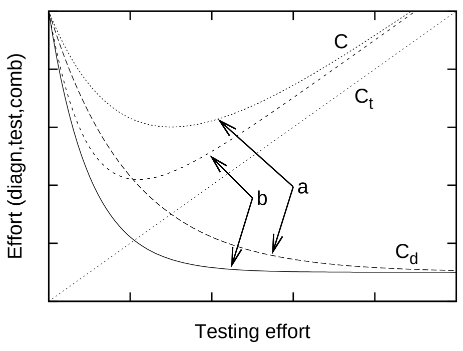


Figure 1: Classic vs. diagnostic test prioritization

plotted: testing cost, diagnosis cost, and overall cost. Unlike static test prioritization, in diagnostic prioritization the tests are dynamically selected based on the actual pass/fail results of previously executed tests (which varies per regression cycle), leading to higher diagnostic performance per test, as shown by the much steeper $C_d(N)$ curve in scenario (b).

While a dynamic approach has the potential to outperform a static approach, a critical factor is the algorithmic complexity of prioritization as the computational cost has now to be spent for each test. In this paper we show that diagnostic prioritization delivers higher diagnostic performance at sufficiently low algorithmic complexity to provide a cost-effective solution in scenarios where testing time is limited and/or where diagnostic cost (α) is high.

Our paper makes the following contributions.

- We present SEQUOIA (SEQUencing fOr dIAgnosis), a low-complexity sequencing algorithm that selects the next test given the outcome of the previous test. Unlike Information Gain (IG) methods known from Bayesian approaches to test sequencing [23] which have exponential complexity for multiple faults, we introduce a novel, low-cost heuristic function, which is inspired by the concept of similarity coefficients used in spectrum-based fault localization approaches [3, 19].

- We study the properties of SEQUOIA for synthetic benchmarks where we can vary independent parameters such as (1) number of tests, (2) number of components, (3) the quality of the test set, (4) the number of injected faults, and (5) component health (the probability a defective component will not produce a test failure).

- We study the performance of SEQUOIA for the Siemens set as well as larger programs `space`, `gzip`, `sed`, available from the Software Infrastructure repository [7], which we have extended to accommodate multiple faults. To overcome limitations on statistical significance due to the very small sample size of originally seeded faults, we have extended this study to include random fault seeding.

In the above experiments we compare the performance of SEQUOIA to the test prioritization algorithms FEP [25] and ART [16], random prioritization (lower performance bound), and the Bayesian IG heuristic (upper performance bound).

Our results show that SEQUOIA delivers a performance close to the optimum (IG), is superior to FEP and random prioritization, and comparable to ART. This performance is achieved at sufficient low time and space complexity ($O(N \cdot M)$ in practice) to enable significant testing and diagnosis cost reduction. Apart from contributing to the field of test prioritization, SEQUOIA also contributes to the field of automatic software debugging. Compared to current automatic debugging approaches which effectively assume a random prioritization order, our approach provides the same diagnostic accuracy using fewer tests.

The paper is organized as follows. In the next section we present some basic concepts and terminology. In Section 3 our diagnosis-based approach to test case prioritization is presented. In Section 4, the approach is theoretically evaluated, while in Section 5 real programs are used to assess our technique. We compare SEQUOIA with related work in Section 6. In Section 7, we conclude and discuss future work.

2. PRELIMINARIES

As SEQUOIA is essentially based on applying fault diagnosis while prioritizing tests, in the following we introduce basic concepts and terminology on fault diagnosis and test prioritization.

2.1 Fault Diagnosis

The objective of fault diagnosis is to pinpoint the precise location of a number of faults in a program (bugs) by observing the program's behavior given a number of tests. The following inputs are usually involved in automated diagnosis:

- A finite set $\mathcal{C} = \{c_1, c_2, \ldots, c_j, \ldots, c_M\}$ of M components (typically source code statements) which are potentially faulty.

- A finite set $\mathcal{T} = \{t_1, t_2, \ldots, t_i, \ldots, t_N\}$ of N tests with binary outcomes $O = (o_1, o_2, \ldots, o_i, \ldots, o_N)$, where $o_i = 1$ if test t_i failed, and $o_i = 0$ otherwise.

- A $N \times M$ coverage matrix, $A = [a_{ij}]$, where $a_{ij} = 1$ if test t_i involves component c_j, and 0 otherwise.

The output of fault diagnosis, i.e., a *diagnosis*, is a component *ranking*, i.e., a list $R =< r_1, r_2, \ldots, r_M >$ of component indices, ordered by the *likelihoods* $L =< l_{r_1}, l_{r_2}, \ldots, l_{r_M} >$ of each component c_{r_m} being faulty, (i.e., $l_{r_m} \geq l_{r_{m+1}}$). In this paper we consider two approaches for obtaining the component likelihoods L, i.e., a Bayesian approach, where the likelihoods are exact fault probabilities, and a statistical approach, where the l_{r_m} are so-called similarity coefficients (SC).

The diagnosis ranking R is returned to the developer who typically verifies the faults by going through R in descending order. The diagnosis cost, C_d, is modeled by

$$C_d = |c_j| \exists c'_j \in d_* : r(j) < r(j')| \quad (2)$$

This represents the effort needlessly spent by the developer inspecting the c_{r_m} that were not defective (false positives), while going through R until all the actual faults (the fault set d_*) are found [3]. We refer to $M_f = |d_*|$ as the actual number of faulty components in the system. The above model for C_d is similar to existing diagnostic performance metrics [19, 24] but excludes the actual faults from the counting to allow unbiased comparison for different M_f in the *multiple*-fault case [3].

2.1.1 Probabilistic Fault Diagnosis

Bayesian diagnosis is a probabilistic reasoning approach, originating from the AI domain, aimed at obtaining a set of diagnostic explanations $D = \{d_1, \ldots, d_k\}$. Each explanation d_k is a subset of the components which, at fault, explain the observed failures. The following additional inputs are involved in Bayesian diagnosis:

- An *a priori* fault probability ("prior") p_j for each component c_j, which represents the knowledge available before any test is executed. Component priors are typically derived from defect density data. Since our components are single statements we shall assume uniform priors, as the prior distribution is also not very critical to diagnostic performance [3].
- A set of *health* values $0 \leq h_j \leq 1$ for each component c_j. Component health is the probability that a faulty component will *not* cause a failure when covered in a test. A health value of $h_j = 1$ means a failure will never occur (even if the component is defective), while $h_j < 1$ means that the defective component will cause a fraction h_j of tests to pass (false negative rate h_j). In hardware, component health is referred to as (fault) *intermittency* [3, 6]. In software, h is related to the concept of *failure exposing potential* [25] mentioned later on, and *testability* [28].

Explanations are ranked according to their expected correctness expressed as probability $\Pr(d_k)$. As there can only be one correct explanation, all the individual probabilities add up to 1.

After each test case t_i, the probability of each diagnostic explanation $d_k \in D$ is updated depending on the outcome o_i of the test, following Bayes' rule:

$$\Pr(d_k|o_i, o_{i-1}, \ldots) = \frac{\Pr(o_i|d_k)}{\Pr(o_i)} \cdot \Pr(d_k|o_{i-1}, \ldots) \qquad (3)$$

In this equation, $\Pr(d_k|o_{i-1}, \ldots)$ represents the prior probability of explanation d_k before the test is executed. $\Pr(o_i|d_k)$ represents the probability of the observed outcome o_i produced by a test t_i, if that diagnostic explanation d_k was the correct one. This depends on the health h_j of the faulty components involved (given by A and d_k). $\Pr(o_i)$ is the probability of the observed outcome, independent of which diagnostic explanation is the correct one. More details can be found in [11].

The internal result of Bayesian diagnosis, D, which can constitute up to 2^M individual explanations d_k, has to be mapped to the diagnosis R. The likelihood of each component being faulty l_j is the sum of the probabilities of every explanation where component c_j is involved, given by

$$l_j = \Pr(c_j|o_i, \ldots) = \sum_{d_k \in D : c_j \in d_k} \Pr(d_k|o_i, \ldots) \qquad (4)$$

Note that, unlike later likelihoods, the above l_j constitutes a proper probability value, which, e.g., also allows estimating the expected number of faults present in the system by

$$E[M_f] = \sum_{j=1}^{M} \Pr(c_j|o_i, \ldots) = \sum_{j=1}^{M} l_j \qquad (5)$$

The above Bayesian diagnosis is the basis for the IG test prioritization heuristic which we shall use as (optimal) reference in our evaluation of SEQUOIA. The above calculations

	Program: Character Counter	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	B	SC
c_1	`main() {`	1	1	1	1	1	1	1	1	.00	.62
c_2	`int let, dig, other, c;`	1	1	1	1	1	1	1	1	.00	.62
c_3	`let = dig = other = 0;`	1	1	1	1	1	1	1	1	.00	.62
c_4	`while(c = getchar()) {`	1	1	1	1	1	1	1	1	.00	.62
c_5	`if ('A'<=c && 'Z'>=c)`	1	1	1	1	1	1	1	0	.01	.71
c_6	`let += 2; /* FAULT */`	1	0	1	1	1	0	1	0	**1.0**	**1.0**
c_7	`elsif ('a'<=c && 'z'>=c)`	1	1	0	1	1	1	1	0	.00	.57
c_8	`let += 1;`	1	0	0	0	1	0	1	0	.01	.60
c_9	`elsif ('0'<=c && '9'>=c)`	1	1	0	1	1	1	0	0	.00	.43
c_{10}	`dig += 1;`	1	1	0	1	0	0	0	0	.00	.33
c_{11}	`elsif (isprint(c))`	0	0	0	0	1	1	0	0	.00	.17
c_{12}	`other += 1;}`	0	0	0	0	1	0	0	0	.01	.20

c_{13}	`printf("%d %d %d\n",`
	 `let, dig, others);}`
		1	1	1	1	1	1	1	1	.00	.62

	Test case outcomes	1	0	1	1	1	0	1	0		

Table 1: Example diagnosis ($h_j = 0.1$, $p_j = 0.01$)

(Eq. 3) have also inspired the low-cost heuristic used in SEQUOIA.

2.1.2 Statistical Fault Diagnosis

A well-known statistical approach to fault diagnosis that originates from the Software Engineering domain is Spectrum-based Fault Localization [2, 19]. Here, the likelihood l_j is quantified in terms of *similarity coefficients* (SCs). A SC measures the statistical similarity between component c_j's test coverage $(a_{1j}, \ldots, a_{Nj})$ and the observed test outcomes, $(o_1, \ldots, o_N)$. Similarity is computed by means of four counters $n_{pq}(j)$ that count the number of times a_{ij} and o_i form the combinations $(0, 0), \ldots, (1, 1)$, respectively, i.e.,

$$n_{pq}(j) = |\{i \mid a_{ij} = p \wedge o_i = q\}| \quad p, q \in \{0, 1\} \qquad (6)$$

For instance, $n_{10}(j)$ and $n_{11}(j)$ are the number of tests in which c_j is executed, and which passed or failed, respectively. The four counters sum up to the number of tests N. For example, the likelihood l_j based on the simple and well-known Jaccard SC is given by coefficient [2]

$$l_j = \frac{n_{11}(j)}{n_{11}(j) + n_{01}(j) + n_{10}(j)} \qquad (7)$$

A great advantage of SCs is their ultra-low computational complexity compared to probabilistic approaches. Despite their lower diagnostic accuracy [3] SCs have, therefore, gained much interest. Our SEQUOIA heuristic is partially inspired by SCs.

2.1.3 Example

To illustrate how the above approaches work, consider the character counter program in Table 1. Initially (for $N = 0$ tests) all c_j have equal prior probability ($p_j = 0.01$), R is random and $C_d(0) = 6.5$. When using SC ("SC" column), after $N = 8$ tests the four n_{pq} counters of c_6 are $n_{11} = 5$, $n_{00} = 3$, $n_{10} = n_{01} = 0$ yielding similarity 1. The other components either have non-zero n_{10} or n_{10}, yielding lower similarities. As a result $R =< 6, \ldots >$ yielding $C_d(8) = 0$ (i.e., perfect diagnosis, no unnecessary inspections). When using Bayesian diagnosis ("B" column), if a candidate d_k is involved (not involved) in a test, and the test passes (fails), its probability will be decreased (increased). Given a statement health estimation of $h_j = 0.1$, after applying the $N = 8$ tests, the top ranked component is c_6 with probability 0.99. The other ranked diagnosis candidates have a much lower probability of being the true fault explanation. For example, the second ranked diagnosis candidate is c_5 with probability 0.01.

2.2 Test Prioritization

Test case prioritization techniques order test cases such that failures occur as early as possible in the testing process, so that confidence in the presence or absence of faults is reached quicker. Due to the uncertainty on the location of faults, in practice, the search process is directed by a heuristic function. The following heuristics have been proposed:

Random: this is the most straightforward prioritization criterion, which orders test cases according to random permutations of the original test suite. Random permutations are used as baseline in many prioritization experiments [9, 25, 27].

Fault-exposing Potential: FEP is a coverage-based prioritization algorithm that assigns each component a confidence value [25]. As high confidence is assigned to a component that has been exercised by a number of (passing) tests, those components need less coverage in subsequent tests. The algorithmic complexity (time and space) of FEP prioritization is $O(N \cdot M)$ per selected test.

Adaptive Random Testing: ART is a hybrid random-coverage-based algorithm [16]. ART selects its next test case in two steps. First, it samples tests randomly until one of the samples does not add additional coverage. Second, it selects the test which maximizes a distance function with the already selected test cases. This distance function can be either the minimum distance with all executed tests, the maximum distance, or the average distance. In this paper we compare with the minimum Jaccard distance heuristic cited in [16] as the most promising one. The best-case time complexity is $O(N^2)$ per selected test, while the worst-case time complexity is $O(N^2 \cdot M)$. With respect to space complexity, besides the $O(N \cdot M)$ coverage matrix, a $O(N^2)$ matrix is needed to store the distances between tests.

2.2.1 Example (continued)

In the following we continue the example in Section 2.1.3. The way L converges with N to the final SC and Bayesian results shown in Table 1 depends on the *order* in which tests $t_1, \ldots, t_8$ are executed. The prioritization order that achieves fastest reduction of C_d (based on L as computed through, e.g., Bayesian diagnosis) is $< t_5, t_8, t_2, \ldots >$, producing $R =< 6, 8, 11, 12, \ldots >$ with $C_d(3) = 0.125$. In contrast, the order $< t_1, t_5, t_4, \ldots >$ as computed by a static prioritization algorithm such as FEP yields $R =< 1, 2, 3, 4, \ldots >$ with $C_d(3) = 0.33$. Hence, the second prioritization order results in higher diagnostic cost if diagnosis would be started after the 3rd test.

3. DIAGNOSTIC PRIORITIZATION

In the following, we first present the classic IG heuristic based on the Bayesian diagnosis as introduced in Section 2. Although prohibitively complex, the heuristic is known to be optimal when all tests have equal cost [26], as in this paper. Just like random prioritization serves as a lower bound, the IG heuristic serves as an upper bound reference when assessing the diagnostic performance of the other prioritization heuristics (FEP, ART, SEQUOIA).

3.1 Bayesian Prioritization

From a diagnostic point of view, the best test is the one that yields the highest diagnostic information gain averaged over the two possible test outcomes (pass/fail). The information gain heuristic [18], IG, is defined as

$$\begin{aligned}\mathcal{H}_{IG}(D, t_i) = \mathrm{H}(D) \\ - \Pr(o_i = 0) \cdot \mathrm{H}(D|o_i = 0) \\ - \Pr(o_i = 1) \cdot \mathrm{H}(D|o_i = 1)\end{aligned} \tag{8}$$

where $\mathrm{H}(D)$ is the *information entropy* of the diagnostic candidate set D, defined as

$$\mathrm{H}(D) = -\sum_{d_k \in D} \Pr(d_k|o_i, \ldots) \cdot \log_2(\Pr(d_k|o_i, \ldots)) \tag{9}$$

and $D|o_i = 0$ represents the updated diagnosis if test t_i passes, and $D|o_i = 1$ if it fails.

The rationale for this heuristic is that H expresses the uncertainty in D and hence estimates the average residual diagnostic cost $E[C_d]$, which is our minimization target (note that C_d cannot be computed as d_* is not (yet) known). Measurements have shown that H is a good estimator of $E[C_d]$. For a matrix A that accommodates all 2^M possible tests, and for one, persistent fault ($M_f = 1$, $h_j = 0$), IG-based prioritization effectively performs a binary search, bisecting the set of candidate components after each test.

Conceptually, when considering all the possible test outcome combinations, a test suite prioritized for diagnosis is a binary tree with $O(2^N)$ nodes, in contrast with off-line prioritization techniques using a static list of $O(N)$. However, if only the nodes corresponding to the current test path are expanded, the $O(N)$ complexity is retained. Still, due to the exponential complexity of computing D and H, IG's cost is prohibitive.

3.2 Sequoia Algorithm

In this section we present our new, low-cost heuristic and associated SC for diagnostic prioritization.

3.2.1 Area-based Heuristic

Due to the exponential computational complexity of H, SEQUOIA uses another, low-cost approximation of $E[C_d]$, which is based on the area difference between L and the best diagnosis, denoted L^∞. as depicted in Figure 2. L^∞ is reached when $N \to \infty$, for which H and C_d would approach zero. The heuristic function is given by

$$E[C_d] = \sum_{r=1}^{M} |l_r - l_r^\infty| \tag{10}$$

The shape of L^∞ is a step function given by;

$$l_r^\infty = \begin{cases} l_f & \text{if } r \le M_f \\ l_n & \text{if } r > M_f \end{cases} \tag{11}$$

where the likelyhoods of the M_f faulty components at the top of L will all have become $l_r = l_f$ and the likelihood of the healthy components will all have become $l_r = l_n$. l_f is the asymptotic likelihood for all faulty components, and l_n for all healthy components. For instance, for Bayesian diagnosis, $l_f = 1$ and $l_n = 0$, and M_f can be estimated by Eq. 5. This situation corresponds to Figure 2.

During prioritization, the chosen test at each step will be the test that minimizes $E[C_d]$, given by

$$\begin{aligned}\mathcal{H}_{seq}(L, t_i) = E[C_d](L) \\ - \Pr(o_i = 0) \cdot E[C_d](L|o_i = 0) \\ - \Pr(o_i = 1) \cdot E[C_d](L|o_i = 1)\end{aligned} \tag{12}$$

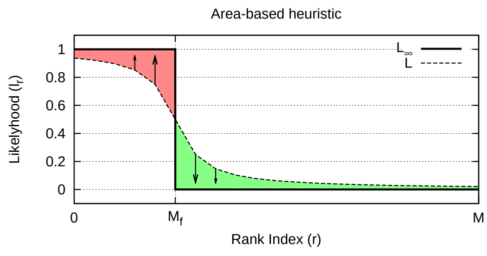


Figure 2: Area difference with the best diagnosis.

where $L|o_i = 0$ represents the updated L if the test passes, and $L|o_i = 1$ if it fails. The value of $\Pr(o_i = 0)$ is the same used in Eq. 3.

Our experiments confirm that for Bayesian diagnosis this area-based heuristic yields a performance comparable to H. However, Bayesian diagnosis is still prohibitively complex given D's exponential space complexity.

3.2.2 Sakura coefficient

Rather than using Bayesian likelihood values for l_r, we use low-cost SCs, which have been introduced in Section 2.1.2. However, the likelihood values of low-cost SCs, such as Jaccard, are no fault probabilities, and therefore Eq. 5 is invalid. Consequently, $E[M_f]$ can only be estimated from the shape of L, by finding the transition zone between l_f and l_n. The new value of $E[M_f]$ corresponds to the index of L where the difference with the next component in L is maximum

$$E[M_f] \approx \underset{r=1,\dots,M-1}{\arg\max}\ (l_r - l_{r+1}) \tag{13}$$

Our experiments have shown, however, that for classical SCs (Jaccard, Tarantula [19], Ochiai [2]) the difference between l_f and l_n compared to their variance is too small to become a reliable heuristic. Furthermore, unlike Bayesian probabilities, SCs have a very unstable evolution until a sufficient number of observations has been reached. This is insufficient, as good accuracy is needed at the very first steps into the sequencing (low N) as this is the stage where much diagnostic gain can be achieved.

To solve the problems with current SCs, we define a new SC, dubbed Sakura (SimilaArity ranKing Update foR diAgnosis), that emulates the update behavior of real probabilities. Starting from a prior value, l_j will increase if component c_j is involved in a failed run, and will decrease if it is involved in a passed run. However, unlike classical SCs the indiction or exoneration is more aggressive (i.e., requires less observations).

Sakura is calculated by an approximation of Bayes' rule:

$$l_j^{(i+1)} = \begin{cases} \dfrac{\Pr(c_j|o_i=0)}{\Pr(o_i=0)} \cdot l_j^{(i)} & \text{if } o_i = 0 \\[2ex] \dfrac{1-\Pr(c_j|o_i=0)}{1-\Pr(o_i=0)} \cdot l_j^{(i)} & \text{if } o_i = 1 \end{cases} \tag{14}$$

where $\Pr(o_i = 0)$ approximates the probability of a test passing, given by

$$\Pr(o_i = 0) \approx \prod_{m=1}^{M} (1 - a_{im} \cdot l_m \cdot (1 - h_m)) \tag{15}$$

and $\Pr(c_j|o_i = 0)$ approximates the same probability conditioned to component c_j being faulty by

$$\Pr(c_j|o_i = 0) \approx \frac{1 - a_{ij} \cdot (1 - h_j)}{1 - a_{ij} \cdot l_j \cdot (1 - h_j)} \cdot \Pr(o_i = 0) \tag{16}$$

The introduction of h_j in the Sakura SC allows us to push down in the ranking a component that is involved in a sufficient number of passed runs, independently of how many failed runs it was involved in. This yields $l_f = 1$ and $l_n = 0$, as well as a clear transition zone, that permits us to perform a much better estimation of M_f. It is very important to note that Eq. 14 is not the proper Bayes' rule. In a true Bayesian update the terms are independent of the test used, whereas in Sequoia their value is also conditioned by the chosen test. Furthermore, the l_r do not necessarily sum up to 1.

3.2.3 Time/Space Complexity

Like in the case of Bayesian prioritization, the complete test sequence is a binary test tree. However, unlike Bayesian prioritization, at each step, Sequoia only needs to store $O(M)$ l_j for each expanded node in the tree. Because at each step in the test sequence the l_j have to be updated for each possible test and outcome, each test selection has a total space complexity of $O(N \cdot M)$. In order to calculate the $E[C_d]$ heuristic, every newly expanded node has to be sorted to obtain $E[M_f]$ (Eq. 13. Consequently, Sequoia has a theoretical time complexity of $O(N \cdot M \cdot \log M)$ per test. However, due to the much higher cost of the expansion than the $E[M_f]$ calculation, Sequoia exhibits $O(N \cdot M)$ time complexity in practice.

4. THEORETICAL EVALUATION

In this section we analyze the performance of Sequoia on synthetically generated matrices A. The motivation for using synthetic data next to real-world data is the ability to study the effect of the various parameters in a controlled setting whereas real programs only represent a few parameter settings in the multi-dimensional parameter space. Furthermore, as we will show in Section 5, real data sets do not always provide the sample sizes needed in view of the high variances involved with diagnostic prioritization algorithms.

As explained in Section 1, prioritization performance is measured in terms of C_d as for equal testing effort (N) the cost reduction in Equation 1 is due to C_d.

4.1 Small Matrices

For small matrices we can still assess the optimality of Sequoia compared to IG, which has exponential complexity. The experiments were performed on randomly generated matrices ($N = 250$ tests, $M = 10$ components, coverage density $\rho = 0.6$, 750 samples to produce one average), seeded with random single and multiple faults with (uniform) health $h = \{0.1, 0.5, 0.9\}$.

Figure 3 shows the evolution of C_d/M vs. N for the random, IG, and Sequoia heuristics. For random and IG, C_d is computed using Bayesian diagnosis (with $p = 0.01$, corresponding to 10 faults per 1,000 components). The performance of Sequoia when internally using Bayesian diagnosis (seq-B) is also plotted to assess the performance degradation due to using the low-cost Sakura SC instead of Bayesian diagnosis in Sequoia's area heuristic.

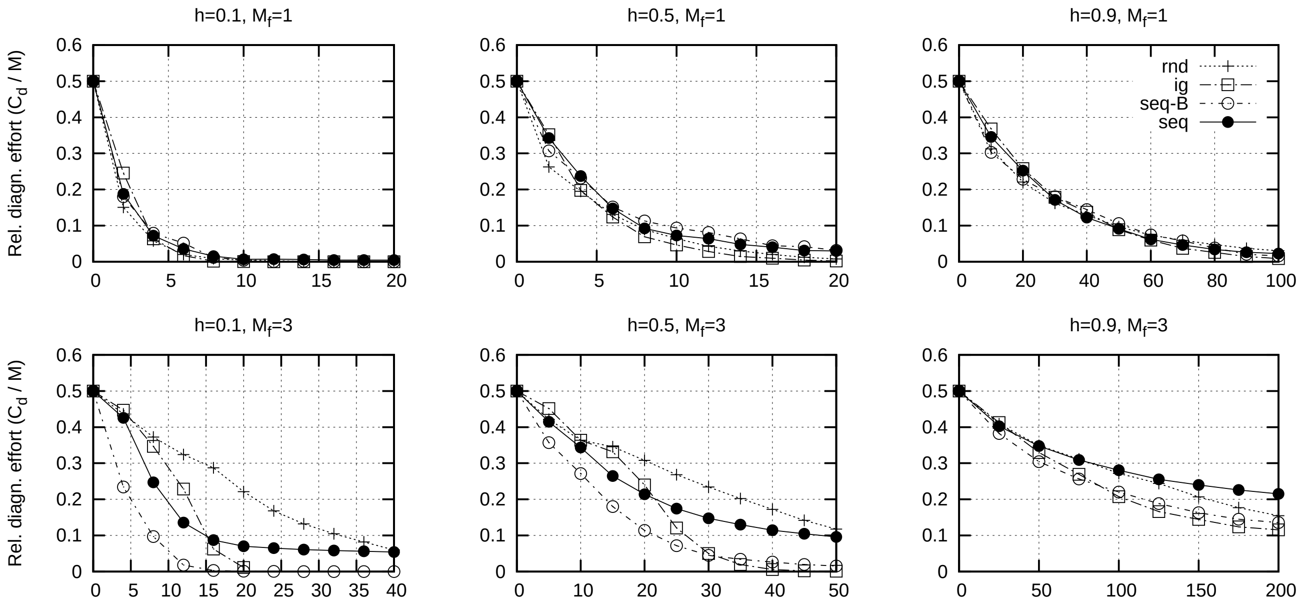


Figure 3: Performance of random, IG, and Sequoia small random matrices

All plots exhibit the typical, geometric decay in C_d where the diagnostic quality of R increases with the amount of observations. Better heuristics will exhibit greater decay rates.

For single faults the uniform coverage distribution in A prohibits the dynamic techniques (IG, SEQUOIA) to outperform random. Dynamic techniques select tests that yield optimal diagnostic information gain, i.e., have failure probability $\Pr(o_i = 1) = 0.5$, reducing diagnostic uncertainty by as much as 1 bit per (binary) test. To produce gains over random selection there must be the opportunity to select from tests that have widely varying failure probabilities. However, for the $M_f = 1$ case, A's uniformity results in every test having the same component coverage (on average, each component is involved in ρ tests and each test involves ρ suspect components). Hence, each test will effectively provide the same amount of information, which explains why random ordering is just as good.

For multiple faults the situation is quite different. Despite the uniformity of A, the tests have widely differing failure probabilities for different multiple-fault candidates [11]. Hence, on average, dynamic techniques are able to exploit much more diagnostic information (shown by the much more rapid decay of $C_d(N)$). In terms of area under the curve (i.e., average C_d) SEQUOIA outperforms the next best technique (ART) by 10%.

When fault exposure rates $1 - h$ become low, the failure probabilities (given by $\Pr(o_i = 1) = 1-(1-\rho\cdot(1-h))^{M_F}$ [1]) become too low (0.17 for $h = 0.9, M_F = 3$). As no tests with failure probability close to 0.5 are available, the additional information gain for dynamic techniques becomes negligible compared to random selection. Consequently, all techniques yield similar performance. Conversely, an excessive amount of faults can have the same effect as it will increase failure probability to levels where it is also not possible to obtain any effective information gain.

Note the better performance of the Bayesian implementation of SEQUOIA's area heuristic, compared to SAKURA, which is a low-cost estimation (5 ms vs. 240 ms per selected test for 250×10 size matrices). The performance difference tends to increase with h. The area heuristic even outperforms H (IG). This is due to the area heuristic not being affected (or being distracted) by the many subsumptions in D, and due to the heuristic being less sensitive to fault multiplicity than H. More details are given in [11].

4.2 Larger Matrices

Now that SEQUOIA's optimality with respect to IG is assessed in terms of M_f and h, in this section we compare SEQUOIA with respect to the classical prioritization heuristics presented in Section 2 for larger uniform matrices ($N = 500$ tests, $M = 50$ components, $\rho = 0.6$, 750 runs per average) where IG's complexity would explode. The results are shown in Figure 4, where we restrict ourselves to multiple-faults ($M_f = 5$) as it has previously been shown that for random (uniform) test matrices SEQUOIA only yields cost reduction for multiple faults. In all cases, SEQUOIA consistently provides the best performance, although the differences become smaller with increasing h. While SEQUOIA selects tests aiming at 50 percent failure probability, the FEP heuristic is designed to *maximize* failure probability (aiming at 100 percent), prohibiting any diagnostic information gain. As a consequence, FEP shows the worst diagnostic performance. ART exhibits intermediate performance between random and SEQUOIA. SInce ART is essentially a random technique, an increase in M_f will affect it to a greater extent than SEQUOIA, because its heuristic does not compensate for increasing failure probability. However, the fact that tests are chosen such that they maximize the average coverage distance with the previous tests, introduces some variability that helps ART improve over random.

4.3 Running Time

Execution time measurement confirms Section 3.2. In practice, the running time of SEQUOIA is $O(N \cdot M)$ and ranges (per selected test) from 4 ms for $N, M = 100$ to 380

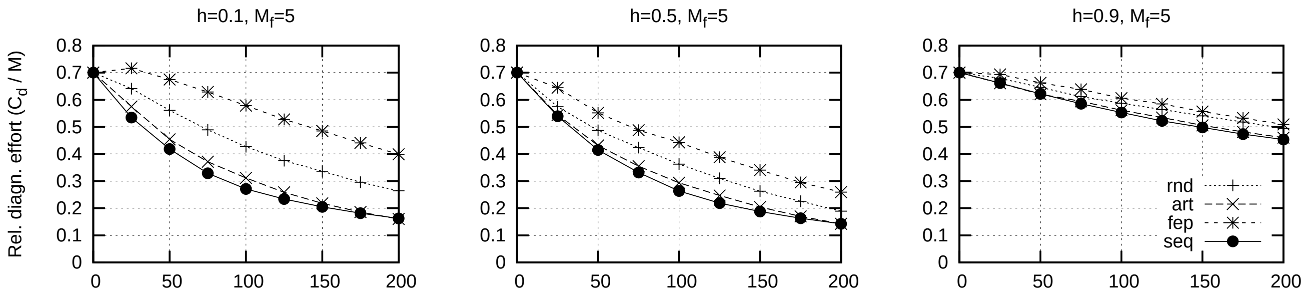


Figure 4: Performance of random, ART, FEP and Sequoia for medium-sized random matrices

ms for $N, M = 1000$ matrices. Despite its $O(MN)$ complexity, the running time of FEP is negligible when compared to ART and SEQUOIA. The average running time of ART is close to $O(N \cdot M)$ when using uniform matrices and is in the same range as SEQUOIA (within a factor of 2). The running times indicate that the overhead of SEQUOIA's online prioritization cannot always be ignored, especially for tests that have a small execution time. In the above 1000×1000 case, SEQUOIA starts becoming a bottleneck when a test takes less than 760 ms (since SEQUOIA can be run in parallel, speculating on both possible test outcomes). This implies that for SEQUOIA the application grain size must be well above the statement level. The cost of integration tests at Thales, our industrial partner, is in the range of seconds and even minutes per test and involves software (and hardware) components at the function level. In this case, SEQUOIA's execution cost plays no role.

4.4 Failure Detection Performance

With respect to failure detection performance (APFD), we observe the performance of SEQUOIA to be lower than FEP and ART, which are designed with failure detection in mind. This can be explained analytically in terms of how many test cases are needed until the first failure occurs, N_{ff}, which, like C_d, can be modeled by a geometric distribution with failure probability parameter $p = \Pr(o_i = 1)$ [11]. The objective of FEP is choosing tests with maximum failure probability, close to $p = 1.0$. On the other hand, diagnostic prioritization heuristics such as IG and SEQUOIA aim at $p = 0.5$, doubling N_{ff}. In practice, $\Pr(o_i = 1)$ is also affected by M_f and h_j [1]. The consequences can be seen in Table 2, rounded to the last significant decimal. For single faults, the factor of 2 mentioned above is indeed established. For multiple faults, however, the difference becomes much less.

M_f	1			5		
h_j	.1	.5	.9	.1	.5	.9
rnd	1.88	2.8	15.4	1.02	1.17	3.80
fep	1.36	2.4	11.9	1.00	1.08	3.03
art	1.86	3.2	17	1.02	1.20	3.9
seq	2.5	3.3	18	1.00	1.10	4.2

Table 2: Influence of M_f and h_j on N_{ff}

4.5 Parameter Sensitivity

The diagnostic precision of SAKURA depends on how well the h_j values are estimated (similar to the estimation problem in FEP). The average number of tests needed to exonerate a healthy component or to indict a faulty component is $1/(1 - h)$, which starts affecting diagnostic performance when h is close to 1.

Program	Faults	M	N	ρ	h	Description
`print_tokens`	4	539	4,130	0.39	0.70	Lexical Analyzer
`print_tokens2`	9	489	4,115	0.41	0.65	Lexical Analyzer
`replace`	23	507	5,542	0.41	0.90	Pattern Recognition
`schedule`	7	397	2,650	0.68	0.95	Priority Scheduler
`schedule2`	9	299	2,710	0.71	0.99	Priority Scheduler
`tcas`	35	174	1,608	0.37	0.96	Altitude Separation
`tot_info`	19	398	1,052	0.56	0.75	Information Measure
`space`	28	9,564	150	0.40	0.19	ADL Interpreter
`gzip-1.3`	7	5,680	210	0.34	0.30	Data compression
`sed-4.1.5`	5	14,427	370	0.35	0.30	Textual manipulator

Table 3: The subject programs

We performed a simple experiment by adding random noise to h with an error of $\sigma = 5, 10, 25\%$, and $M_f = 3$, averaged over 750 runs. For $h = 0.1$ the error of the area under C_d is low, but significantly different for all techniques, which means that the difference between techniques is partially due to the test choice. As h does not influence test choice, random ordering is the least affected with a maximum error of 3%, on the other hand, SEQUOIA is the most affected with a maximum error of 19%. On the other hand, for $h = 0.5, 0.9$ the error is higher (up to 56%), but not significantly different between the techniques. This means that for intermediate and high healths, the error that SEQUOIA or any other prioritization technique makes (which affects how fast C_d decreases) is much lower when compared with the error in the diagnostic (the final C_d value), which is due to the diagnostic algorithm. More details can be found in [11].

5. EMPIRICAL EVALUATION

In this section we compare the performance of SEQUOIA to random, FEP and ART, in terms of C_d as measured by SAKURA, for the well-known Siemens benchmark set [15] as well as `gzip`, `sed` and `space` (obtained from SIR [7]).

Table 3 provides more information about the programs used in the experiments, where M corresponds to the number of lines of code (components at statement granularity). For our experiments, we have extended the subject programs with program versions for which we can activate arbitrary combinations of *multiple* faults. For this purpose, we limit ourselves to a selection of 146 out of the 183 faults, based on criteria such as faults being attributable to a single line of code, to enable unambiguous evaluation.

As each program suite includes a correct version, we use the output of the correct version as oracle. Again, we compare in terms of relative diagnostic cost C_d/M' but in this case we modify M to $M' = M - W$ to remove white space and static code (i.e., code not instrumented by tt gcov) bias (W) from the comparisons. Further, to avoid plots we aggregated the $C_d(N)$ curves to numbers by computing the area under the curve (i.e., average diagnostic cost over $N = 1, \dots, 250$ tests).

5.1 Real Faults

In the first experiments the real faults provided with each program are used. The number of faults and possible fault combinations varies per program between 4 and 1,448. We diagnosed each fault on 5 matrices of 250 tests each, to smooth artifacts caused by random sampling and to remove any possible test suite construction bias. A shortcoming of test sequencing is that it requires knowledge of the value of h_j. In our $M_f = 1$ experiments the health value of the faults could be easily estimated as the test suite is seeded with single faults. For $M_f > 1$, the maximum h value was used as input to SEQUOIA to minimize estimation errors as discussed in Section 4.5. The average h_j for each program is shown in Table 3. Notice the significant difference between the average h_j of the hand-seeded faults of the Siemens programs and the real faults of `gzip`, `sed` and `space`, which suggests that the faults in Siemens are not representative of real software, in accordance with results from Briand [4]. Based on an ANOVA analysis of the results, we were unable to reject the null hypothesis that there is a significant difference between techniques in many cases ($\alpha = 0.05$). Table 4 shows the results for those cases where a significant difference was found. In the single-fault case, only `schedule` provided significant results, be it with a high sampling error ($\sigma = 25\%$), which reinforces our suspicion that the amount and location of faults in the Siemens set are far from statistically appropriate for diagnostic prioritization studies. In the cases where the results are different, post-hoc multiple comparison analysis was performed using the Bonferroni Lowest Significant Difference (LSD) method. The *best* row contains the heuristics (best to worst) whose difference with the best one is lower than LSD.

With the exception of FEP, which was designed to maximize test failure probability, the rest of heuristics cannot be considered statistically different in half of the cases. SEQUOIA comes on top only in 1 case and ties with ART in 3 cases. The high performance of FEP for `tot_info` can be explained by the fact that most faults are in areas of the code covered by very few tests [25]. By maximizing the failure probability, those areas will be covered faster than by ART or SEQUOIA, which work under the assumption that faults are uniformly spread through the code. However, as we will see in the next section, a more uniform distribution of faults will reverse the situation.

	prnt_tok2		replace		schedule			sched2		tot_info		gzip
M_f	3	5	3	5	1	3	5	3	5	3	5	5
rnd	0.7	0.93	0.62	0.78	0.2	0.5	0.6	0.6	0.6	0.56	0.65	0.8
fep	0.8	0.97	0.67	0.82	0.3	0.6	0.8	0.69	0.76	0.52	0.60	0.9
art	0.7	0.93	0.61	0.78	0.2	0.5	0.5	0.6	0.6	0.53	0.63	0.7
seq	0.7	0.93	0.60	0.76	0.1	0.4	0.5	0.5	0.6	0.55	0.64	0.7
LSD	0.1	0.01	0.01	0.01	0.1	0.1	0.1	0.0	0.0	0.01	0.01	0.1
best	s,a,r	a,r,s	s,a	s	s,a,r	s,a	s,a	s,a,r	s,r,a	f,a	f	s,a,r

Table 4: Average diagnostic cost, real faults

With respect to failure detection capability as measured by APFD, again, the performance of SEQUOIA is (somewhat) worse than FEP, with a median score of 0.92 vs. 0.94. However, is better than RND (0.85) and ART (0.88), as the occurrence of the first fault depends heavily on the densities of the matrices, which is lower than $\rho = 0.5$ in most cases.

5.2 Simulated Faults

As mentioned earlier, the faults included in the test suites gave little opportunity to construct experiments with statistical significance, given the fact that dynamic prioritization techniques inherently exhibit a much larger variance than static techniques. Therefore, we extended our performance assessment with experiments using the original coverage matrices of the programs, but seeding them with random faults. Compared to the synthetic experiments in Section 4 we now deal with real matrices. To eliminate bias induced by suite composition, we sampled randomly the original suites to create 25 matrices of 250 tests each. For each of them, we simulated 50 combinations of M_f faults. This gives a total of 1,250 runs per program, per (M_f, h_j) setting.

Table 5 presents a summary of the results for each program where the ANOVA test did show significant differences between techniques with at least 95% probability. In accordance with our synthetic experiments, SEQUOIA and ART clearly outperform random, although the improvement is usually smaller than in the synthetic case. SEQUOIA performs significantly the best in 12 cases, ART in 3 cases, whereas FEP and random are never the best techniques.

The cases of `schedule`, `schedule2`, are the most favorable for SEQUOIA providing up to a 15% improvement over the next-best (ART) and up to 19% over random prioritization, as the high density of their matrices partially compensates the high h_j. In these cases, ART's effectiveness is impaired by the fact that the redundancy in the tests reduces the variability that the distance function can introduce. The high number of cases (18) where SEQUOIA and ART tie, are most likely due to the uniformity of p_j and h_j as discussed in Section 5.3.

On the other hand, sparse test matrices like the ones of `print_tokens`, `print_tokens2` and especially `gzip`, `space` (where program functionalities are tested individually or in small bundles) provide little advantage as a diagnostic algorithm such as SEQUOIA is unable to *cleverly* combine multiple components per test [11]. Consequently, the improvement over random prioritization is negligible. This can be observed in the 8 cases where SEQUOIA, ART and random prioritization are tied. However, as M_f increases this effect tends to disappear, as can be seen in the ($h = 0.5, M_f = 5$) case of `print_tokens` and `print_tokens2`. Furthermore, for $M_f = 10$, `space` shows significant differences between techniques, with an 10% improvement of SEQUOIA and ART over random prioritization.

5.3 Threats to Validity

Our results show that the benchmark suites are not very suitable for our comparisons where large fault samples are required for statistical significance. Furthermore, faults in Siemens are often too hard to detect [4] and too hard to reach [25], which makes it questionable to what extent the Siemens set is representative. In view of the above, the role of (semi) synthetic coverage matrices becomes important, as this also offers more insight into the various (controllable) parameters. However, we only consider random matrices A with only two parameters (ρ,h). Although much insight was gained, the real programs showed that real A are far

	print_tokens						print_tokens2						replace						schedule					
M_f	1		3		5		1		3		5		1		3		5		1		3		5	
h_j	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9
rnd	0.08	0.14	0.31	0.48	0.51	0.69	0.08	0.13	0.28	0.46	0.48	0.65	0.09	0.14	0.29	0.46	0.49	0.66	0.09	0.18	0.37	0.58	0.62	0.76
fep	0.13	0.16	0.36	0.49	0.56	0.71	0.10	0.15	0.33	0.48	0.51	0.66	0.08	0.15	0.29	0.47	0.48	0.67	0.10	0.23	0.39	0.61	0.61	0.78
art	0.07	0.13	0.29	0.47	0.50	0.68	0.07	0.13	0.27	0.45	0.46	0.64	0.07	0.13	0.27	0.45	0.46	0.64	0.08	0.17	0.35	0.55	0.59	0.74
seq	0.07	0.13	0.31	0.47	0.52	0.68	0.06	0.14	0.27	0.45	0.46	0.63	0.07	0.13	0.28	0.44	0.47	0.64	0.07	0.14	0.32	0.52	0.54	0.72
LSD	0.01	0.01	0.02	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
best	s,a	a,s,r	a,s,r	a,s,r	a	s,a	s,a	a,r,s	s,a,r	s,a,r	a,s	s,a	s,a	a,s	a,s	s,a	a,s	a,s	s,a	s	s	s	s	s

	schedule2						tcas						tot_info						gzip				space
M_f	1		3		5		1		3		5		1		3		5		1		3		10
h_j	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5	0.9	0.5
rnd	0.10	0.19	0.39	0.58	0.64	0.76	0.09	0.14	0.41	0.55	0.66	0.75	0.12	0.16	0.38	0.54	0.61	0.74	0.14	0.20	0.40	0.54	0.30
fep	0.16	0.26	0.46	0.62	0.68	0.78	0.10	0.15	0.43	0.60	0.68	0.79	0.17	0.19	0.41	0.55	0.64	0.75	0.10	0.17	0.34	0.51	0.28
art	0.10	0.18	0.37	0.56	0.63	0.75	0.08	0.13	0.39	0.55	0.64	0.76	0.11	0.15	0.35	0.52	0.58	0.72	0.11	0.18	0.36	0.51	0.27
seq	0.08	0.16	0.35	0.54	0.58	0.73	0.08	0.14	0.43	0.56	0.67	0.75	0.10	0.14	0.36	0.51	0.58	0.72	0.09	0.18	0.36	0.51	0.27
LSD	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
best	s	s	s	s	s	s	s,a	a	a,r	a,r,s	a	s,r,a	s	s,a	a,s	s,a	a,s	s,a	s,f	f,a	f,s,a	s,f,a	a,s

Table 5: Average diagnostic cost, simulated faults ($\alpha = 0.05$)

from uniform, and that this affects achievable diagnostic cost reduction. Instead of synthesizing random faults, mutation may offer an alternative way of obtaining a larger sample of faults that are close to real faults [4], improving external validity.

In our experiments we assumed identical fault probabilities p_j and component healths h_j which is not likely for real faults. This threatens the external validity of the results with simulated faults, especially given the close tie between ART and SEQUOIA. However, while SEQUOIA automatically exploits additional information on p_j, h_j, ART has no mechanism to adapt its test selection. Hence, SEQUOIA's performance should be seen as a lower bound, and the close tie between SEQUOIA and ART as less realistic.

As already mentioned, SEQUOIA takes as input the fault probability p_j and health parameter h_j of each component. In our controlled experiments this value was available. In practice, however, both must be estimated. From our sensitivity analysis it is clear that the performance of SEQUOIA (and, e.g., FEP) is affected by the estimation quality. Hence, in *this* sense our results should be seen as upper bounds.

6. RELATED WORK

The influence of test-suite extension, reduction, modification, and prioritization on fault detection and diagnosis has received considerable attention [5, 13, 31]. In particular, Jiang *et al.* [17] show how some prioritization techniques are worse than random orders and that those which are better, do not provide a significant improvement. In contrast to the works cited above, which use coverage-based heuristics to reduce the size of the test suite, SEQUOIA employs an heuristic to select the best test case to optimize diagnostic accuracy.

Test case prioritization is a mature and active area of research whose most common goal is to increase failure detection rate, which dates back to [14, 30]. The failure detection effectiveness of different coverage-based prioritization techniques was studied by Rothermel *et al.* [9], who also proposed the FEP heuristic [25] that has already been discussed throughout the paper. Other work on prioritization includes [8, 20, 25]. Jiang *et al.* [16] propose a hybrid random and coverage-based prioritization technique (ART) which has already been discussed throughout this paper. SEQUOIA fundamentally differs from traditional prioritization techniques in that its main goal is not to optimize the rate of failure detection, but to minimize debugging cost.

Automated fault-localization techniques also aim at minimizing diagnostic cost when failures occur during the testing phase. Statistical approaches include [2, 19, 21, 22, 24, 29]. A recent, probabilistic approach of acceptable complexity is [3]. Unlike the above diagnostic approaches, SEQUOIA addresses the *order* in which tests are applied. Furthermore, the novel SAKURA SC used in SEQUOIA differs from SCs like Tarantula or Ochiai in that it is more tuned to a prioritization algorithm where optimum diagnostic precision based on only a few tests is critical.

Sequential diagnosis aims at finding the test sequence that optimizes diagnostic performance based on the current test outcomes. Sequential diagnosis is often applied to hardware systems where tests can produce false negatives (intermittent faults) [23]. Typically, only one fault is assumed present in the system. If the system can have multiple faults, sequential diagnosis complexity becomes exponential. In [26] an approximation is described for systems with permanent faults ($h = 0$), an assumption which does not hold for most software systems. SEQUOIA differs in that it employs a sequential approach for *multiple, intermittent* faults, at polynomial complexity due to our low-cost heuristic instead of the Bayesian approach. A preliminary version of our work is described in [12], that reveals the high potential of (IG-based) dynamic prioritization. However, to avoid the combinatorial explosion of Bayesian diagnosis, in that work we assumed single faults, which is unrealistic for large systems.

7. CONCLUSIONS & FUTURE WORK

In this paper we presented a dynamic test prioritization algorithm, SEQUOIA, to minimize the subsequent diagnostic cost after testing revealed the existence of faults.

Our results show that diagnostic prioritization can significantly reduce diagnosis cost, compared to the case where no information is exploited (random), at a minimal expense in terms of APFD. Unlike the large cost reductions in our earlier work for the specific, single-fault case (more than 50% [12]), in the more realistic case, where any amount of faults may be present, and where tests can have false negatives, the reduction by SEQUOIA appears highly dependent on the coverage matrix A, component health h, and fault density M_f. When A comprises tests that cover multiple components (e.g., integration tests instead of unit tests) without excessive component health, SEQUOIA is able to exploit test information to reduce diagnostic cost. Our measurements show that SEQUOIA outperforms random sequencing, FEP, and ART, up to 10% to next-best for synthetic suites, and up to 15% to next-best for the Siemens set (`schedule` and `schedule2`) and 20% to random orderings. Although significant reduction cannot always be achieved, the cost reduction increases with the amount of faults, an attractive prospect for large codebases. Although FEP and ART are designed to optimize failure probability instead of diagnostic information (which requires 50% failure probability per test), ART also yields diagnostic cost reduction, often outperforming random sequencing and FEP. However, ART has no mechanism to adapt test choices to more realistic cases with non-identical fault probabilities and component healths. ART is unable to exploit this information, unlike SEQUOIA, making it a less robust method.

Future work includes (1) extending our study to non-uniform test costs, fault probabilities and healths, (2) studying the use of mutation techniques to better fault sampling and for estimating h as done in [25], (3) extending the synthetic models to (non-uniform) coverage matrices that better characterize real test suites.

8. REFERENCES

[1] R. Abreu, P. Zoeteweij, and A. van Gemund. An observation-based model for fault localization. In *Proc. WODA'08*.

[2] R. Abreu, P. Zoeteweij, and A. van Gemund. On the accuracy of spectrum-based fault localization. In *Proc. TAIC PART'07*.

[3] R. Abreu, P. Zoeteweij, and A. van Gemund. Spectrum-based multiple fault localization. In *Proc. ASE'09*.

[4] J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments? In *Proc. ICSE'05*.

[5] B. Baudry, F. Fleurey, and Y. L. Traon. Improving test suites for efficient fault localization. In *Proc. ICSE'06*.

[6] J. de Kleer. Diagnosing multiple persistent and intermittent faults. In *Proc. IJCAI'09*.

[7] H. Do, S. G. Elbaum, and G. Rothermel. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Emp. Soft. Eng. J.*, 10(4), 2005.

[8] S. Elbaum, G. Rothermel, S. Kanduri, and A. G. Malishevsky. Selecting a cost-effective test case prioritization technique. *Soft. Quality Control*, 2004.

[9] S. G. Elbaum, A. G. Malishevsky, and G. Rothermel. Test case prioritization: A family of empirical studies. *IEEE TSE*, 28(2), 2002.

[10] A. Feldman, G. M. Provan, and A. van Gemund. Fractal: Efficient fault isolation using active testing. In *Proc. IJCAI'09*.

[11] A. Gonzalez, R. Abreu, H.-G. Gross, and A. van Gemund. A diagnostic approach to test prioritization (technical report). Technical report, Softw. Eng. Grp., Delft University of Technology, 2010.

[12] A. Gonzalez, E. Piel, H.-G. Gross, and A. van Gemund. Prioritizing tests for software fault localization. Technical report, Software Engineering Research Group, Delft University of Technology, 2010. Under review for QSIC'10.

[13] D. Hao, L. Zhang, H. Zhong, H. Mei, and J. Sun. Eliminating harmful redundancy for testing-based fault localization using test suite reduction: An experimental study. In *Proc. ICSM'05*.

[14] M. J. Harrold, R. Gupta, and M. L. Soffa. A methodology for controlling the size of a test suite. *ACM TSEM*, 2(3), 1993.

[15] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proc. ICSE '94*.

[16] B. Jiang, Z. Zhang, W. Chan, and T. Tse. Adaptive random test case prioritization. In *Proc. ASE'09*.

[17] B. Jiang, Z. Zhang, T. H. Tse, and T. Y. Chen. How well do test case prioritization techniques support statistical fault localization. In *Proc. COMPSAC'09*.

[18] R. Johnson. An information theory approach to diagnosis. In *Symposium on Reliability and Quality Control*, 1960.

[19] J. A. Jones, M. J. Harrold, and J. Stasko. Visualization of test information to assist fault localization. In *Proc. ICSE'02*.

[20] Z. Li, M. Harman, and R. M. Hierons. Search algorithms for regression test case prioritization. *IEEE TSE*, 33(4), 2007.

[21] B. Liblit. Cooperative debugging with five hundred million test cases. In *Proc. ISSTA'08*.

[22] C. Liu, X. Yan, L. Fei, J. Han, and S. P. Midkiff. Sober: Statistical model-based bug localization. In *Proc. ESEC/FSE-13*.

[23] V. Raghavan, M. Shakeri, and K. R. Pattipati. Test sequencing algorithms with unreliable tests. *IEEE TSMC*, 29(4), 1999.

[24] M. Renieris and S. P. Reiss. Fault localization with nearest neighbor queries. In *Proc. ASE'03*.

[25] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Prioritizing test cases for regression testing. *IEEE TSE*, 27(10), 2001.

[26] M. Shakeri, V. Raghavan, K. R. Pattipati, and A. Patterson-Hine. Sequential testing algorithms for multiple fault diagnosis. *IEEE TSMC*, 2000.

[27] A. M. Smith and G. M. Kapfhammer. An empirical study of incorporating cost into test suite reduction and prioritization. In *Proc. SAC'09*.

[28] J. M. Voas. Pie: A dynamic failure-based technique. *IEEE TSE*, 18(8):717–727, 1992.

[29] W. Wong, T. Wei, Y. Qi, and L. Zhao. A crosstab-based statistical method for effective fault localization. In *Proc. ICST'08*.

[30] W. E. Wong, J. R. Horgan, S. London, and H. Agrawal. A study of effective regression testing in practice. In *Proc. ISSRE'97*.

[31] Y. Yu, J. A. Jones, and M. J. Harrold. An empirical study of the effects of test-suite reduction on fault localization. In *Proc. ICSE'08*.

A Addendum to the Theory

A.1 Bayesian Diagnostic

After each test case t_i, the probability of each diagnostic explanation $d_k \in D$ is updated depending on the outcome o_i of the test, following Bayes' rule:

$$\Pr(d_k|o_i, o_{i-1}, \ldots) = \frac{\Pr(o_i|d_k)}{\Pr(o_i)} \cdot \Pr(d_k|o_{i-1}, \ldots) \tag{1}$$

In this equation, $\Pr(d_k|o_{i-1}, \ldots)$ represents the prior probability of explanation d_k before the test is executed. Before any test is executed, the probability of each explanation is:

$$\Pr(d_k) = \prod_{c_j \in d_k} p_j \cdot \prod_{c_{j'} \notin d_k} (1 - p_{j'}) \tag{2}$$

$\Pr(o_i|d_k)$ represents the probability of the observed outcome o_i produced by a test t_i, if that diagnostic explanation d_k was the correct one. This depends on the health h_j of the faulty components involved (given by A and d_k) according to

$$\Pr(o_i = 0|d_k) = 1 - \Pr(o_i = 1|d_k) = \prod_{c_j \in d_k \wedge a_{ij} = 1} h_j \tag{3}$$

$\Pr(o_i)$ is the probability of the observed outcome, independent of which diagnostic explanation is the correct one.

$$\Pr(o_i) = \sum_{d_k \in D} \Pr(o_i|d_k) \cdot \Pr(d_k|o_{i-1}, \ldots) \tag{4}$$

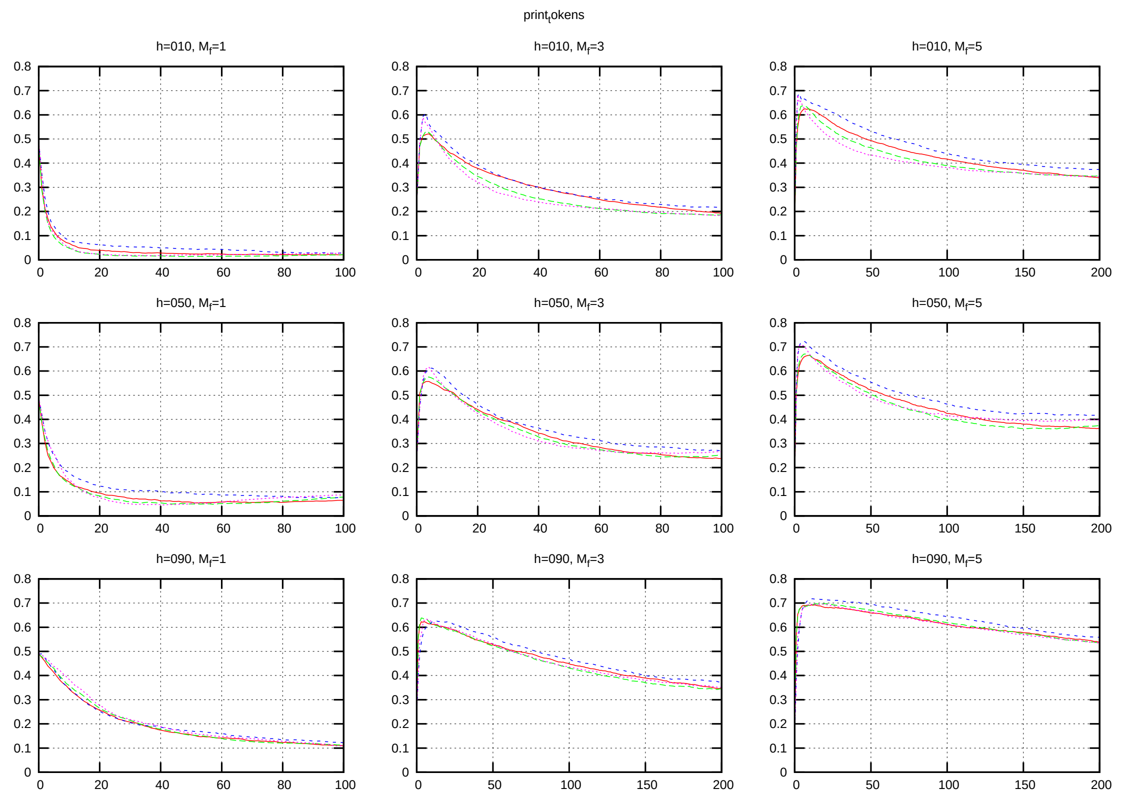
A.2 Matrix density and multiple faults

When a test matrix is uniform, every test case provides the same information gain in the $M_f = 1$ case. It is easy to see why if we transform A to a matrix where columns correspond to combiantions of components instead of to individual components.

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1
\end{bmatrix} \tag{5}$$

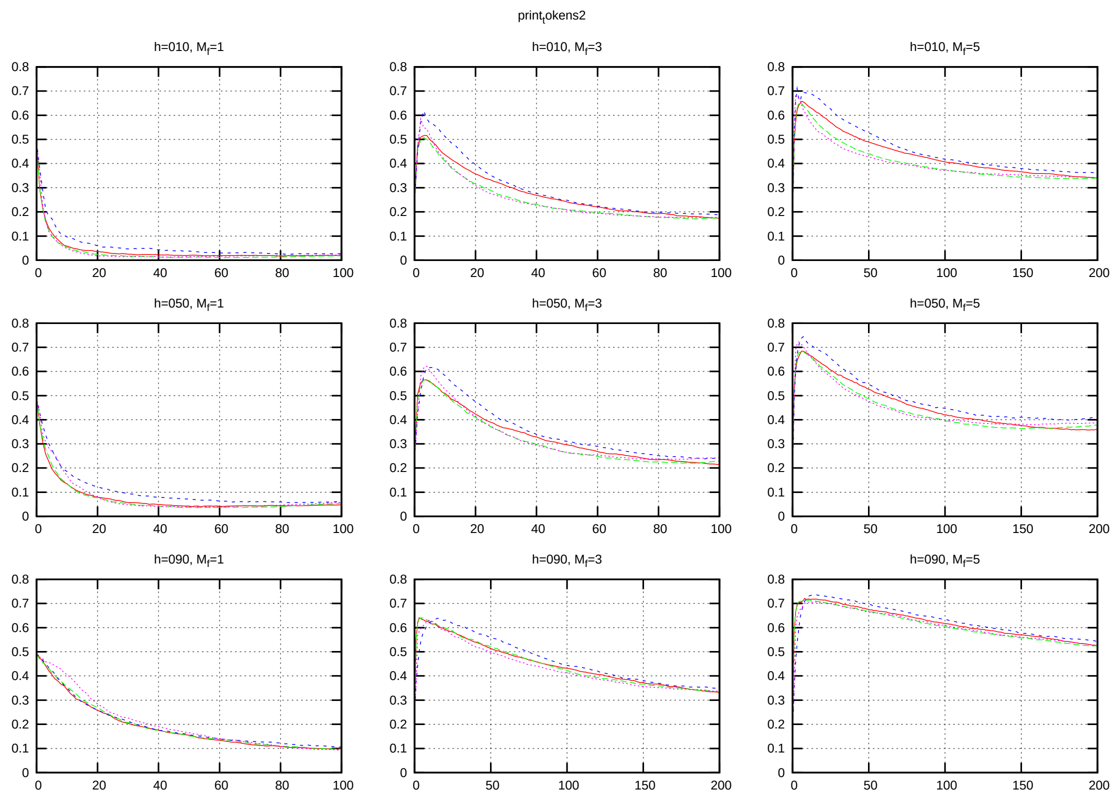
B Detailed Empirical Results

B.1 Plots and Statistical Analysis of Simulated Faults



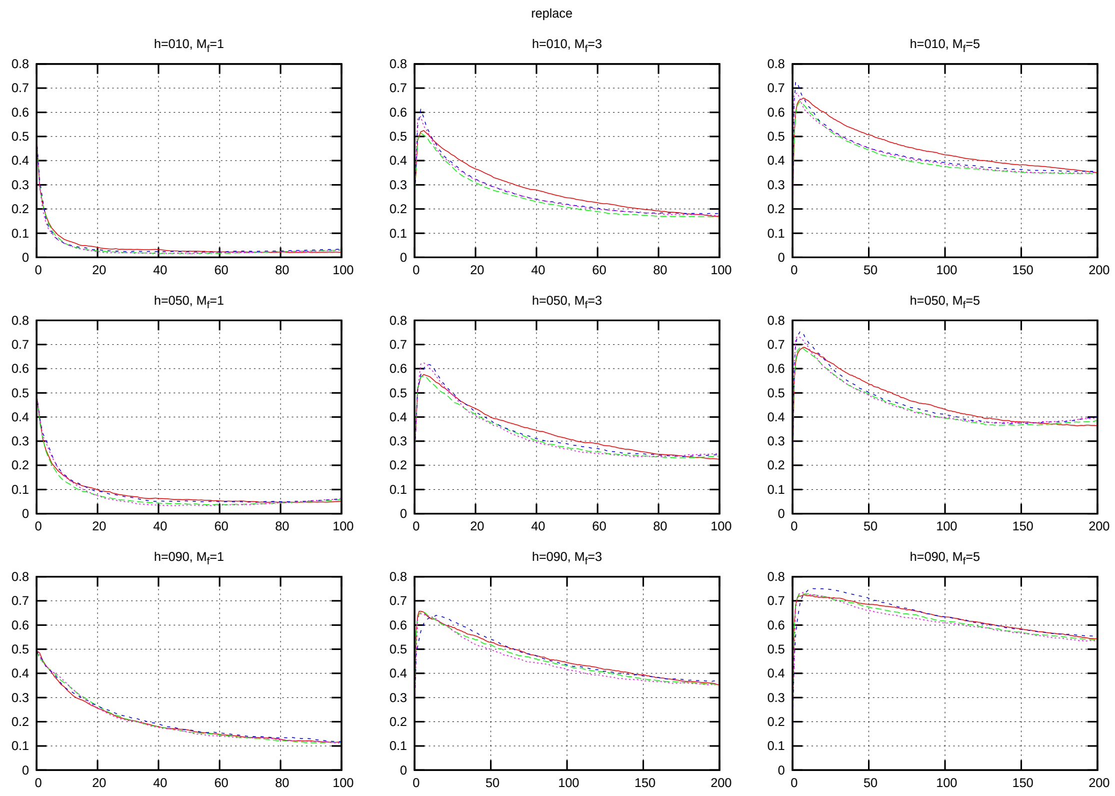
```
print_tokens   C1  050
0.0842+-0.0033 0.1137+-0.0043 0.0806+-0.0034 0.0877+-0.0040  F=15.99 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.009 T= 1.7559 { art rnd seq }
     rnd - fep :    -0.0294793698135 True
     rnd - art :    0.00366397243434 False
     rnd - seq :    -0.00343006677648 False
     fep - art :    0.0331433422478 True
     fep - seq :    0.026049303037 True
     art - seq :    -0.00709403921081 False

print_tokens   C1  090
0.1918+-0.0040 0.2010+-0.0043 0.1950+-0.0043 0.2038+-0.0043  F=1.70 p=0.1649

print_tokens   C3  050
0.3391+-0.0051 0.3646+-0.0056 0.3340+-0.0052 0.3332+-0.0056  F=7.51 p=0.0001
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { seq art rnd }
     rnd - fep :    -0.0254816730577 True
     rnd - art :    0.00515393002816 False
     rnd - seq :    0.00590597429454 False
     fep - art :    0.0306356030859 True
     fep - seq :    0.0313876473523 True
     art - seq :    0.000752044266377 False

print_tokens   C3  090
0.4595+-0.0050 0.4761+-0.0054 0.4491+-0.0051 0.4541+-0.0053  F=5.06 p=0.0017
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { art seq rnd }
```

```
     rnd - fep :    -0.0165225436731 True
     rnd - art :    0.01040697191 False
     rnd - seq :    0.00545616813017 False
     fep - art :    0.026929515583 True
     fep - seq :    0.0219787118032 True
     art - seq :    -0.0049508037798 False

print_tokens   C5  050
0.4558+-0.0051 0.4952+-0.0055 0.4414+-0.0053 0.4547+-0.0058  F=18.44 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { art seq }
     rnd - fep :    -0.0394198611214 True
     rnd - art :    0.014442199891 True
     rnd - seq :    0.00112631540247 False
     fep - art :    0.0538620610123 True
     fep - seq :    0.0405461765238 True
     art - seq :    -0.0133158844885 False

print_tokens   C5  090
0.5917+-0.0041 0.6140+-0.0046 0.5947+-0.0041 0.6077+-0.0043  F=6.10 p=0.0004
Bonferroni means: ddof= 4996 MSD= 0.011 T= 1.7559 { rnd art }
     rnd - fep :    -0.0223008575249 True
     rnd - art :    -0.0030553734446 False
     rnd - seq :    -0.016016436983 True
     fep - art :    0.0192454840803 True
     fep - seq :    0.00628442054185 False
     art - seq :    -0.0129610635384 True
```

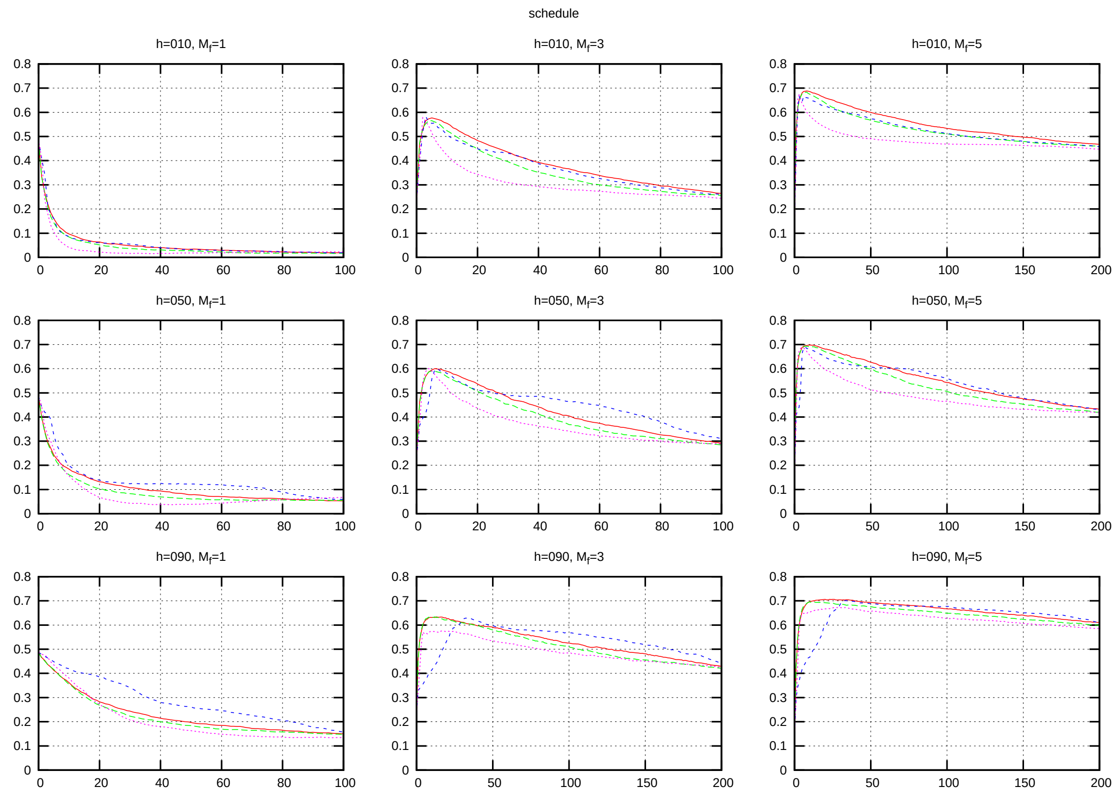


```
print_tokens2  C1  050
0.0713+-0.0027 0.1005+-0.0037 0.0687+-0.0027 0.0764+-0.0029  F=22.99 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.008 T= 1.7559 { art rnd }
     rnd - fep :    -0.0291689975248 True
     rnd - art :    0.00258917079208 False
     rnd - seq :    -0.00507883663366 False
     fep - art :    0.0317581683168 True
     fep - seq :    0.0240901608911 True
     art - seq :    -0.00766800742574 True

print_tokens2  C1  090
0.1851+-0.0040 0.1927+-0.0042 0.1888+-0.0042 0.2006+-0.0040  F=2.62 p=0.0489
Bonferroni means: ddof= 4996 MSD= 0.010 T= 1.7559 { rnd art fep }
     rnd - fep :    -0.00754412128713 False
     rnd - art :    -0.0037192759901 False
     rnd - seq :    -0.0154552289604 True
     fep - art :    0.00382484529703 False
     fep - seq :    -0.00791110767327 False
     art - seq :    -0.0117359529703 True

print_tokens2  C3  050
0.3239+-0.0049 0.3498+-0.0050 0.3086+-0.0049 0.3172+-0.0050  F=12.88 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { art seq }
     rnd - fep :    -0.0259496699692 True
     rnd - art :    0.0152301258239 True
     rnd - seq :    0.00670463077925 False
     fep - art :    0.0411797957931 True
     fep - seq :    0.0326543007484 True
     art - seq :    -0.00852549504467 False
```

```
print_tokens2  C3  090
0.4464+-0.0051 0.4627+-0.0051 0.4443+-0.0050 0.4323+-0.0049  F=6.23 p=0.0003
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { seq art }
     rnd - fep :    -0.0162927445977 True
     rnd - art :    0.0021683043116 False
     rnd - seq :    0.0141665319219 True
     fep - art :    0.0184610489093 True
     fep - seq :    0.0304592765196 True
     art - seq :    0.0119982276103 False

print_tokens2  C5  050
0.4549+-0.0049 0.4844+-0.0050 0.4376+-0.0049 0.4419+-0.0053  F=17.66 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { art seq }
     rnd - fep :    -0.0294814365721 True
     rnd - art :    0.0173067900023 True
     rnd - seq :    0.0130235167874 True
     fep - art :    0.0467882265744 True
     fep - seq :    0.0425049533595 True
     art - seq :    -0.00428327321489 False

print_tokens2  C5  090
0.6016+-0.0041 0.6271+-0.0046 0.6070+-0.0044 0.6072+-0.0043  F=6.64 p=0.0002
Bonferroni means: ddof= 4996 MSD= 0.011 T= 1.7559 { rnd art seq }
     rnd - fep :    -0.0255092931047 True
     rnd - art :    -0.00534675165425 False
     rnd - seq :    -0.00559128938356 False
     fep - art :    0.0201625414504 True
     fep - seq :    0.0199180037211 True
     art - seq :    -0.000244537729312 False
```

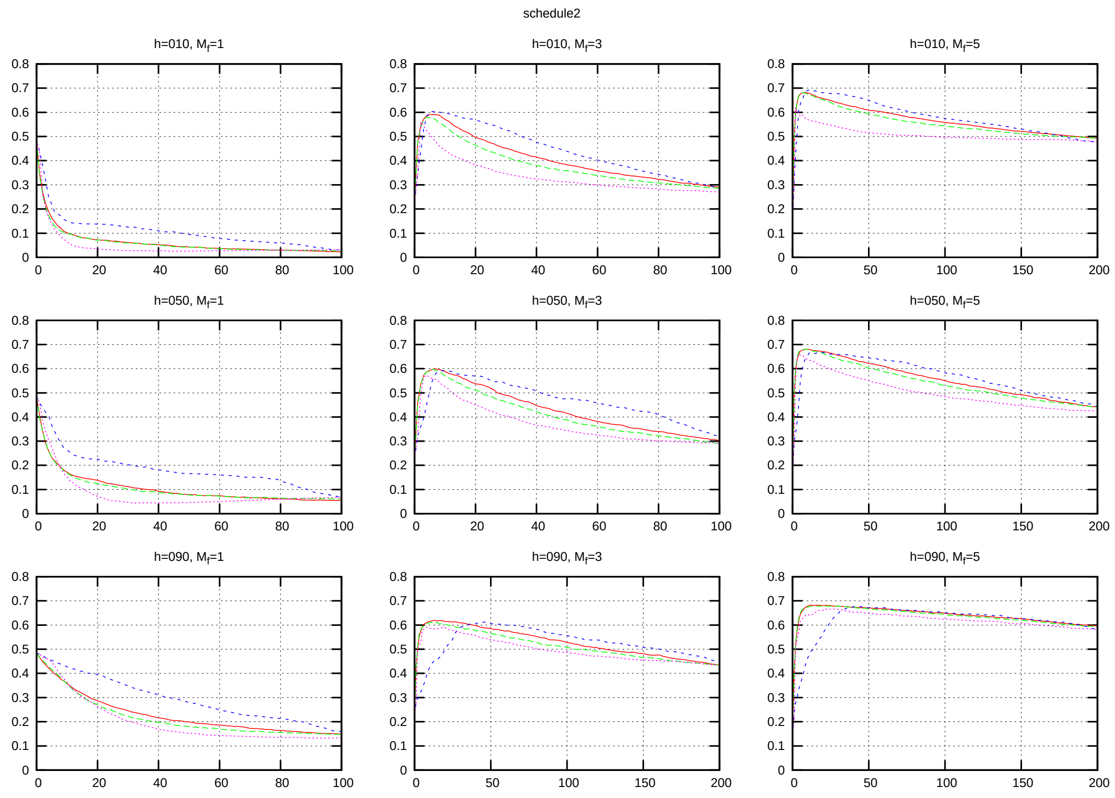


```
replace        C1  050
0.0818+-0.0028 0.0819+-0.0029 0.0697+-0.0026 0.0708+-0.0027  F=5.90 p=0.0005
Bonferroni means: ddof= 4996 MSD= 0.007 T= 1.7559 { art seq }
     rnd - fep :    -8.8303558812e-05 False
     rnd - art :    0.0120750953495 True
     rnd - seq :    0.0109729086638 True
     fep - art :    0.0121633989083 True
     fep - seq :    0.0110612122226 True
     art - seq :    -0.0011021866857 False

replace        C1  090
0.1941+-0.0042 0.2003+-0.0043 0.1943+-0.0043 0.1936+-0.0043  F=0.55 p=0.6452

replace        C3  050
0.3368+-0.0050 0.3280+-0.0050 0.3150+-0.0049 0.3182+-0.0049  F=4.04 p=0.0071
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { art seq }
     rnd - fep :    0.00883722490657 False
     rnd - art :    0.0217985638747 True
     rnd - seq :    0.0186461611762 True
     fep - art :    0.0129613389681 True
     fep - seq :    0.00980893626958 False
     art - seq :    -0.00315240269849 False

replace        C3  090
0.4626+-0.0051 0.4625+-0.0050 0.4513+-0.0050 0.4411+-0.0051  F=4.16 p=0.0059
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { seq art }
     rnd - fep :    0.000167692705393 False
     rnd - art :    0.0113125544772 False
     rnd - seq :    0.021556510508 True
     fep - art :    0.0111448617718 False
     fep - seq :    0.0213888178026 True
     art - seq :    0.0102439560308 False

replace        C5  050
0.4633+-0.0048 0.4546+-0.0049 0.4404+-0.0048 0.4460+-0.0050  F=4.20 p=0.0056
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { art seq }
     rnd - fep :    0.00874164127286 False
     rnd - art :    0.0229205912252 True
     rnd - seq :    0.0172943772352 True
     fep - art :    0.0141789499523 True
     fep - seq :    0.00855273596235 False
     art - seq :    -0.00562621398995 False

replace        C5  090
0.6085+-0.0041 0.6141+-0.0044 0.5974+-0.0043 0.5790+-0.0044  F=13.07 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.011 T= 1.7559 { seq }
     rnd - fep :    -0.00563267037341 False
     rnd - art :    0.0111058466143 True
     rnd - seq :    0.0294799669342 True
     fep - art :    0.0167385169877 True
     fep - seq :    0.0351126373077 True
     art - seq :    0.0183741203199 True
```



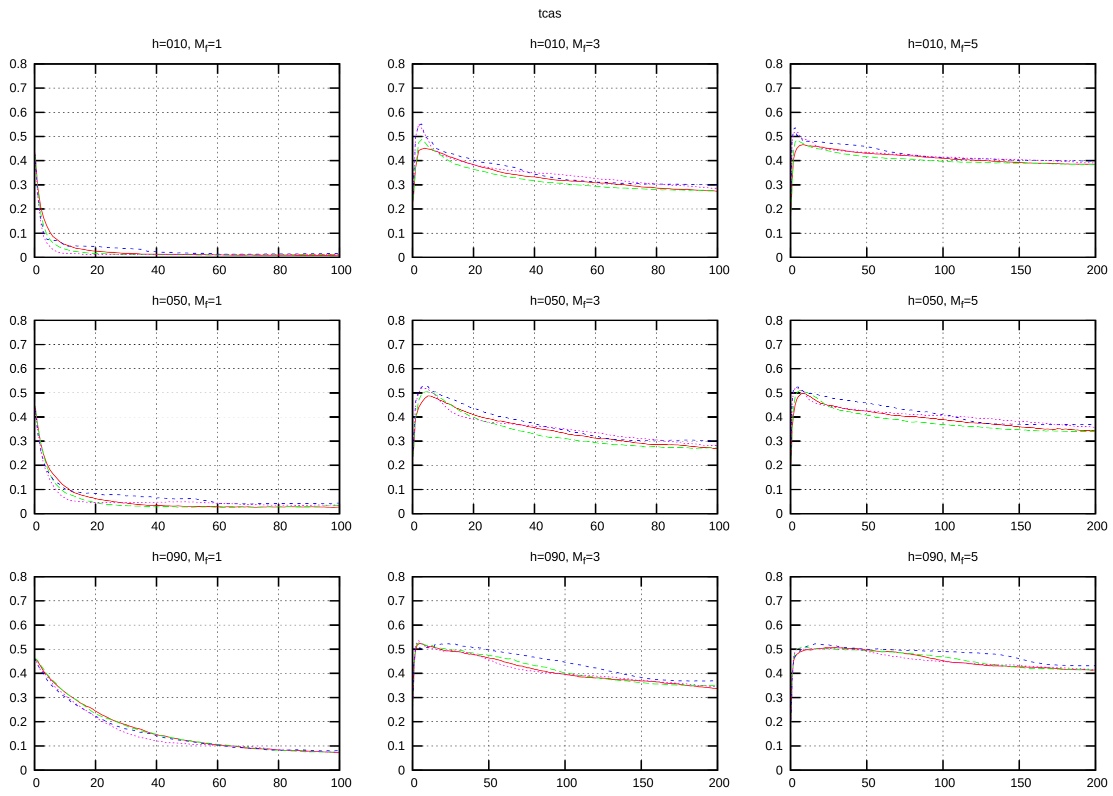
```
schedule       C1  050
0.1040+-0.0032 0.1336+-0.0039 0.0896+-0.0032 0.0781+-0.0030  F=51.93 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.008 T= 1.7559 { seq }
     rnd - fep :    -0.0295680693069 True
     rnd - art :    0.0144158910891 True
     rnd - seq :    0.0259038613861 True
     fep - art :    0.043983960396 True
     fep - seq :    0.0554719306931 True
     art - seq :    0.011487970297 True

schedule       C1  090
0.2259+-0.0052 0.2830+-0.0050 0.2158+-0.0050 0.2047+-0.0047  F=49.50 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { seq art }
     rnd - fep :    -0.0570578217822 True
     rnd - art :    0.0101339108911 False
     rnd - seq :    0.0212476732673 True
     fep - art :    0.0671917326733 True
     fep - seq :    0.0783054950495 True
     art - seq :    0.0111137623762 False

schedule       C3  050
0.4174+-0.0051 0.4428+-0.0053 0.3947+-0.0052 0.3640+-0.0051  F=41.89 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { seq }
     rnd - fep :    -0.0253728712857 True
     rnd - art :    0.0226839273906 True
     rnd - seq :    0.0533929042863 True
     fep - art :    0.0480567986763 True
     fep - seq :    0.078765775572 True
     art - seq :    0.0307089768957 True
```

```
schedule       C3  090
0.5266+-0.0055 0.5324+-0.0055 0.5125+-0.0054 0.4881+-0.0052  F=13.26 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { seq }
     rnd - fep :    -0.0057872388035 False
     rnd - art :    0.0140580762825 True
     rnd - seq :    0.0385116003368 True
     fep - art :    0.019845315086 True
     fep - seq :    0.0442988391403 True
     art - seq :    0.0244535240543 True

schedule       C5  050
0.5465+-0.0045 0.5421+-0.0047 0.5225+-0.0046 0.4819+-0.0050  F=39.30 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { seq }
     rnd - fep :    0.00444543445556 False
     rnd - art :    0.0240623200621 True
     rnd - seq :    0.0646712752721 True
     fep - art :    0.0196168856066 True
     fep - seq :    0.0602258408165 True
     art - seq :    0.0406089552099 True

schedule       C5  090
0.6568+-0.0043 0.6395+-0.0047 0.6421+-0.0044 0.5973+-0.0042  F=33.94 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.011 T= 1.7559 { seq }
     rnd - fep :    0.0172720115634 True
     rnd - art :    0.0146461210656 True
     rnd - seq :    0.0594991359775 True
     fep - art :    -0.00262589049787 False
     fep - seq :    0.0422271244141 True
     art - seq :    0.044853014912 True
```

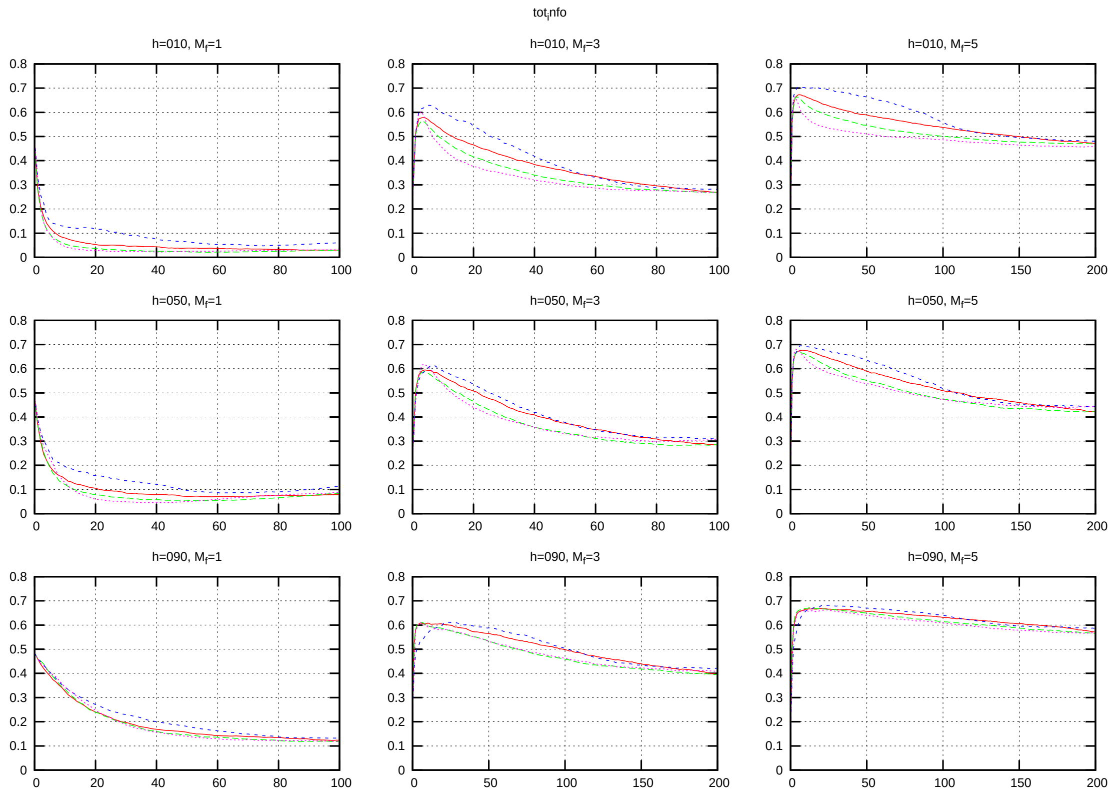


```
schedule2      C1  050
0.1038+-0.0033 0.1824+-0.0046 0.1016+-0.0034 0.0816+-0.0031  F=147.78 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.009 T= 1.7559 { seq }
     rnd - fep :    -0.0785587157029 True
     rnd - art :    0.00220810087774 False
     rnd - seq :    0.0222075011063 True
     fep - art :    0.0807668165806 True
     fep - seq :    0.100766216809 True
     art - seq :    0.0199994002286 True

schedule2      C1  090
0.2262+-0.0053 0.2928+-0.0051 0.2144+-0.0051 0.1988+-0.0048  F=65.84 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { seq }
     rnd - fep :    -0.0665596161209 True
     rnd - art :    0.0118182418681 False
     rnd - seq :    0.0274163217685 True
     fep - art :    0.078377857989 True
     fep - seq :    0.0939759378894 True
     art - seq :    0.0155980799004 True

schedule2      C3  050
0.4251+-0.0052 0.4636+-0.0052 0.4042+-0.0051 0.3669+-0.0055  F=59.19 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { seq }
     rnd - fep :    -0.038597460043 True
     rnd - art :    0.020836503629 True
     rnd - seq :    0.0581146316824 True
     fep - art :    0.059433963672 True
     fep - seq :    0.0967120917254 True
     art - seq :    0.0372781280533 True
```

```
schedule2      C3  090
0.5255+-0.0053 0.5247+-0.0051 0.5107+-0.0053 0.4933+-0.0052  F=8.31 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { seq }
     rnd - fep :    0.000831529233015 False
     rnd - art :    0.0148221217588 True
     rnd - seq :    0.0321702801367 True
     fep - art :    0.0139905925258 True
     fep - seq :    0.0313387509037 True
     art - seq :    0.0173481583779 True

schedule2      C5  050
0.5516+-0.0046 0.5647+-0.0044 0.5390+-0.0047 0.4978+-0.0050  F=37.92 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { seq }
     rnd - fep :    -0.0130707414521 True
     rnd - art :    0.0125864736574 True
     rnd - seq :    0.0537973892987 True
     fep - art :    0.0256572151095 True
     fep - seq :    0.0668681307507 True
     art - seq :    0.0412109156413 True

schedule2      C5  090
0.6380+-0.0043 0.6159+-0.0044 0.6342+-0.0041 0.5915+-0.0040  F=25.61 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.010 T= 1.7559 { seq }
     rnd - fep :    0.0221215378539 True
     rnd - art :    0.0038109371392 False
     rnd - seq :    0.046477689134 True
     fep - art :    -0.0183106007147 True
     fep - seq :    0.0243561512801 True
     art - seq :    0.0426667519947 True
```

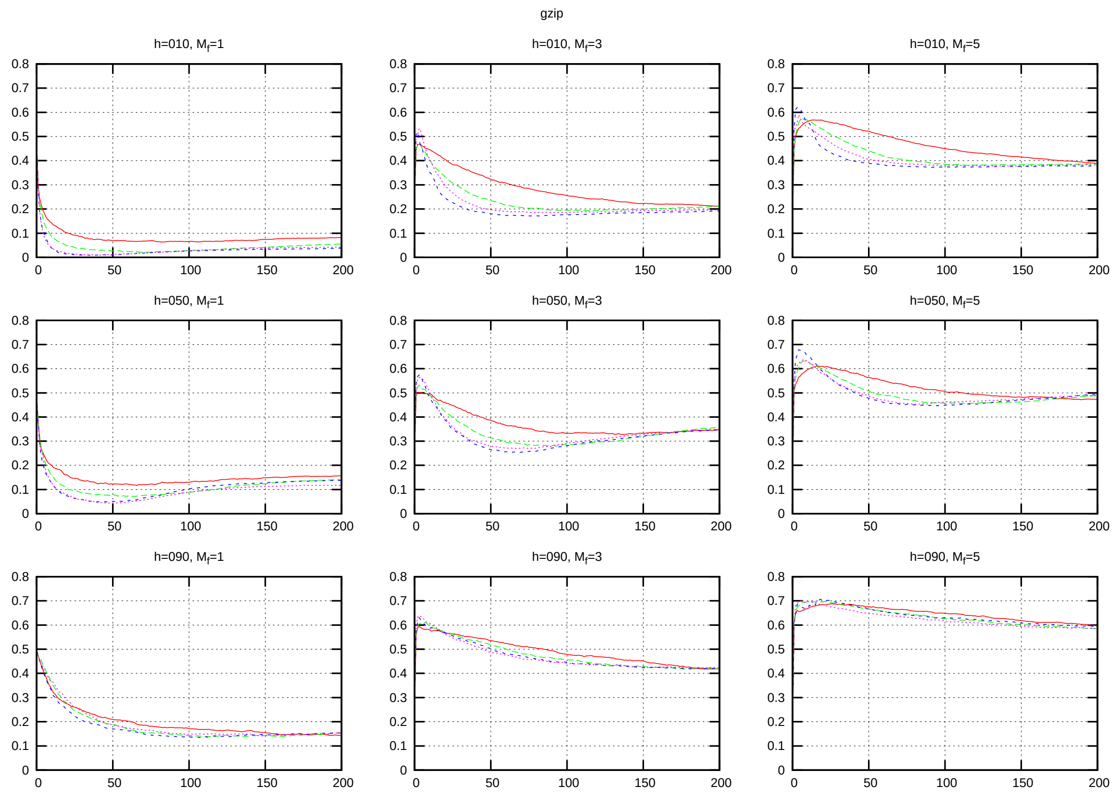


```
tcas           C1  050
0.0552+-0.0020 0.0711+-0.0032 0.0497+-0.0022 0.0555+-0.0027  F=12.95 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.006 T= 1.7559 { art rnd seq }
     rnd - fep :    -0.0159373407251 True
     rnd - art :    0.00555275832337 False
     rnd - seq :    -0.000248938726574 False
     fep - art :    0.0214900990485 True
     fep - seq :    0.0156884019986 True
     art - seq :    -0.00580169704994 False

tcas           C1  090
0.1618+-0.0036 0.1548+-0.0035 0.1611+-0.0037 0.1509+-0.0037  F=2.05 p=0.1043

tcas           C3  050
0.3438+-0.0046 0.3632+-0.0053 0.3319+-0.0046 0.3522+-0.0051  F=7.14 p=0.0001
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { art rnd }
     rnd - fep :    -0.0193864212278 True
     rnd - art :    0.0118493636447 False
     rnd - seq :    -0.00844309286844 False
     fep - art :    0.0312357848725 True
     fep - seq :    0.0109433283593 False
     art - seq :    -0.0202924565131 True

tcas           C3  090
0.4103+-0.0047 0.4393+-0.0051 0.4148+-0.0049 0.4105+-0.0048  F=8.10 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { rnd seq art }
     rnd - fep :    -0.029076213936 True
     rnd - art :    -0.004524212301 False
     rnd - seq :    -0.000262757674667 False
     fep - art :    0.024552001635 True
     fep - seq :    0.0288134562613 True
     art - seq :    0.00426145462633 False

tcas           C5  050
0.3920+-0.0035 0.4139+-0.0035 0.3811+-0.0037 0.4053+-0.0037  F=16.19 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.009 T= 1.7559 { art }
     rnd - fep :    -0.0219272730593 True
     rnd - art :    0.0108550343882 True
     rnd - seq :    -0.013312532601 True
     fep - art :    0.0327823074474 True
     fep - seq :    0.00861474045827 False
     art - seq :    -0.0241675669892 True

tcas           C5  090
0.4550+-0.0032 0.4765+-0.0032 0.4571+-0.0033 0.4533+-0.0032  F=11.24 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.008 T= 1.7559 { seq rnd art }
     rnd - fep :    -0.021515143165 True
     rnd - art :    -0.00209329535725 False
     rnd - seq :    0.00169036234659 False
     fep - art :    0.0194218478077 True
     fep - seq :    0.0232055055116 True
     art - seq :    0.00378365770384 False
```



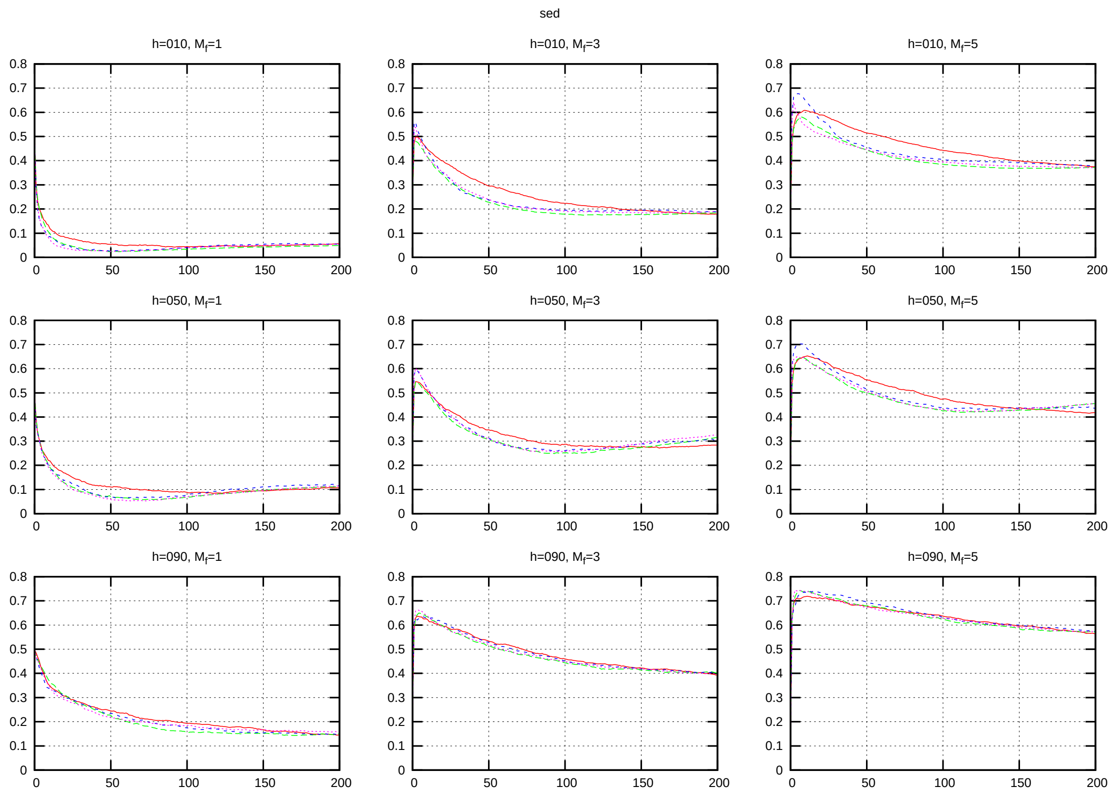
```
tot_info       C1  050
0.0963+-0.0034 0.1293+-0.0042 0.0819+-0.0031 0.0829+-0.0035  F=38.64 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.009 T= 1.7559 { art seq }
     rnd - fep :    -0.0329722167363 True
     rnd - art :    0.0144471648386 True
     rnd - seq :    0.0134382242187 True
     fep - art :    0.0474193815749 True
     fep - seq :    0.046410440955 True
     art - seq :    -0.00100894061988 False

tot_info       C1  090
0.1896+-0.0043 0.2079+-0.0046 0.1842+-0.0045 0.1842+-0.0044  F=6.37 p=0.0003
Bonferroni means: ddof= 4996 MSD= 0.011 T= 1.7559 { seq art rnd }
     rnd - fep :    -0.018315691332 True
     rnd - art :    0.00543846399422 False
     rnd - seq :    0.00544926513529 False
     fep - art :    0.0237541553263 True
     fep - seq :    0.0237649564673 True
     art - seq :    1.08011410696e-05 False

tot_info       C3  050
0.3956+-0.0051 0.4075+-0.0051 0.3651+-0.0049 0.3674+-0.0050  F=17.45 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { art seq }
     rnd - fep :    -0.0118671467926 False
     rnd - art :    0.0305642766954 True
     rnd - seq :    0.0282660269088 True
     fep - art :    0.042431423488 True
     fep - seq :    0.0401331737014 True
     art - seq :    -0.00229824978661 False
```

```
tot_info       C3  090
0.4980+-0.0052 0.5014+-0.0053 0.4726+-0.0052 0.4759+-0.0052  F=8.03 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { art seq }
     rnd - fep :    -0.00342742840549 False
     rnd - art :    0.0254042517862 True
     rnd - seq :    0.0220612798991 True
     fep - art :    0.0288316801917 True
     fep - seq :    0.0254887083046 True
     art - seq :    -0.00334297188716 False

tot_info       C5  050
0.5237+-0.0041 0.5388+-0.0044 0.4957+-0.0043 0.4949+-0.0046  F=25.14 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.011 T= 1.7559 { seq art }
     rnd - fep :    -0.0151507355046 True
     rnd - art :    0.0280302490222 True
     rnd - seq :    0.0288042477178 True
     fep - art :    0.0431809845268 True
     fep - seq :    0.0439549832224 True
     art - seq :    0.000773998695642 False

tot_info       C5  090
0.6033+-0.0037 0.6257+-0.0042 0.5818+-0.0039 0.5859+-0.0040  F=25.59 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.010 T= 1.7559 { art seq }
     rnd - fep :    -0.0223927491422 True
     rnd - art :    0.0214723936782 True
     rnd - seq :    0.0174168957601 True
     fep - art :    0.0438651428204 True
     fep - seq :    0.0398096449022 True
     art - seq :    -0.00405549791815 False
```



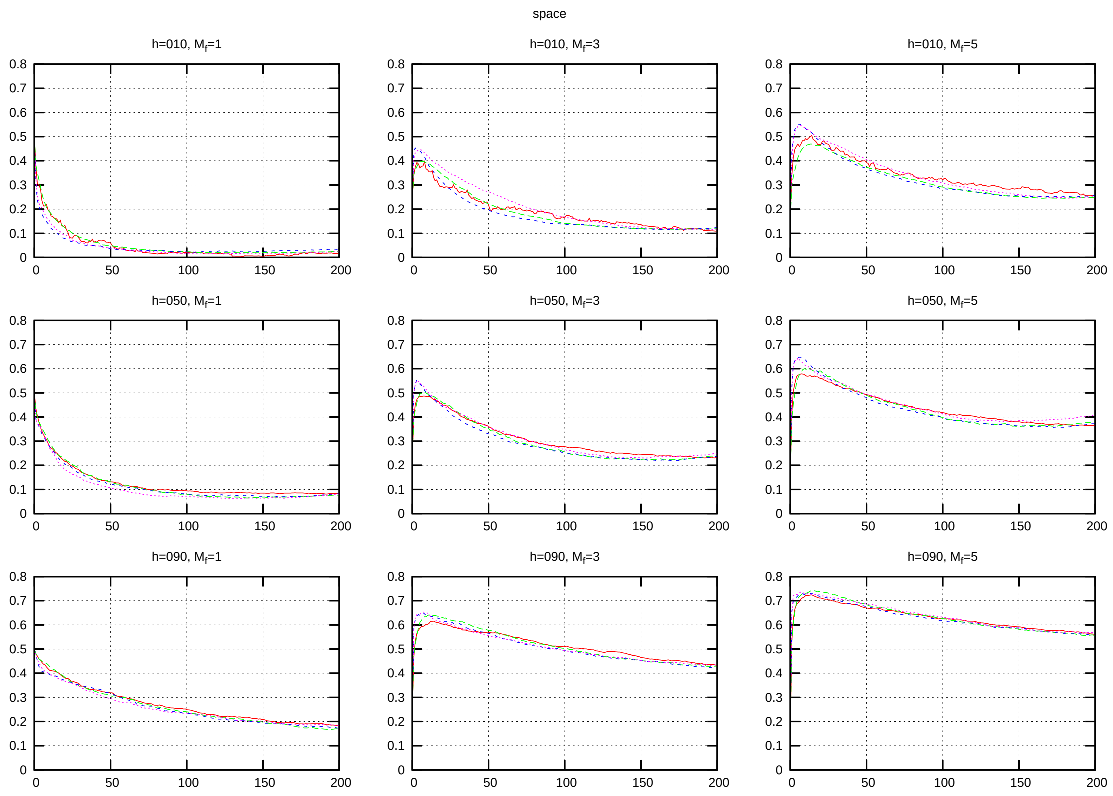
```
gzip           C1  050
0.1460+-0.0048 0.1044+-0.0048 0.1111+-0.0045 0.0946+-0.0045  F=22.89 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { seq fep }
     rnd - fep :    0.0416154296687 True
     rnd - art :    0.034880274546 True
     rnd - seq :    0.0513869241935 True
     fep - art :    -0.00673515512271 False
     fep - seq :    0.00977149452484 False
     art - seq :    0.0165066496475 True

gzip           C1  090
0.1941+-0.0036 0.1731+-0.0039 0.1802+-0.0038 0.1854+-0.0040  F=5.37 p=0.0011
Bonferroni means: ddof= 4996 MSD= 0.009 T= 1.7559 { fep art }
     rnd - fep :    0.0210106368127 True
     rnd - art :    0.0139661129904 True
     rnd - seq :    0.00870643942358 False
     fep - art :    -0.00704452382227 False
     fep - seq :    -0.0123041973891 True
     art - seq :    -0.00525967356683 False

gzip           C3  050
0.3654+-0.0057 0.3184+-0.0057 0.3327+-0.0056 0.3267+-0.0056  F=13.25 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.014 T= 1.7559 { fep seq }
     rnd - fep :    0.0470536659549 True
     rnd - art :    0.0327698876664 True
     rnd - seq :    0.0387195289947 True
     fep - art :    -0.0142837782886 True
     fep - seq :    -0.00833413696026 False
     art - seq :    0.0059496413283 False
```

```
gzip           C3  090
0.4881+-0.0053 0.4669+-0.0054 0.4732+-0.0054 0.4657+-0.0054  F=3.67 p=0.0117
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { seq fep art }
     rnd - fep :    0.0212364795325 True
     rnd - art :    0.0149133673935 True
     rnd - seq :    0.0224262069736 True
     fep - art :    -0.00632311213906 False
     fep - seq :    0.00118972744107 False
     art - seq :    0.00751283958014 False

gzip           C5  050
0.5189+-0.0047 0.4872+-0.0050 0.4922+-0.0049 0.4895+-0.0048  F=9.28 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.012 T= 1.7559 { fep seq art }
     rnd - fep :    0.031756394515 True
     rnd - art :    0.026760636209 True
     rnd - seq :    0.0294403017159 True
     fep - art :    -0.00499575830597 False
     fep - seq :    -0.00231609279906 False
     art - seq :    0.00267966550691 False

gzip           C5  090
0.6268+-0.0040 0.6325+-0.0043 0.6016+-0.0042 0.5905+-0.0043  F=22.88 p=0.0000
Bonferroni means: ddof= 4996 MSD= 0.010 T= 1.7559 { seq }
     rnd - fep :    -0.0056854261083 False
     rnd - art :    0.0252069726043 True
     rnd - seq :    0.0362726875009 True
     fep - art :    0.0308923987126 True
     fep - seq :    0.0419581136092 True
     art - seq :    0.0110657148967 True
```



```
sed              C1   050
0.1150+-0.0045 0.1063+-0.0047 0.0977+-0.0043 0.0950+-0.0046  F=4.07 p=0.0067
Bonferroni means: ddof= 4996 MSD= 0.011 T= 1.7559 { seq art }
     rnd - fep :    0.00873636571668 False
     rnd - art :    0.0172932480092 True
     rnd - seq :    0.019992339607 True
     fep - art :    0.00855688229252 False
     fep - seq :    0.0112559738903 True
     art - seq :    0.00269909159781 False

sed              C1   090
0.2143+-0.0048 0.2035+-0.0048 0.1970+-0.0046 0.2051+-0.0048  F=2.24 p=0.0810

sed              C3   050
0.3223+-0.0055 0.3150+-0.0058 0.3053+-0.0057 0.3167+-0.0058  F=1.55 p=0.1994

sed              C3   090
0.4792+-0.0052 0.4764+-0.0053 0.4683+-0.0052 0.4714+-0.0053  F=0.88 p=0.4498

sed              C5   050
0.4963+-0.0053 0.4845+-0.0054 0.4708+-0.0054 0.4746+-0.0054  F=4.49 p=0.0037
Bonferroni means: ddof= 4996 MSD= 0.013 T= 1.7559 { art seq }
     rnd - fep :    0.0117548486688 False
     rnd - art :    0.0255091382272 True
     rnd - seq :    0.0216747476088 True
     fep - art :    0.0137542895584 True
     fep - seq :    0.00991989893996 False
     art - seq :    -0.00383439061847 False

sed              C5   090
```

```
0.5965+-0.0042 0.6138+-0.0043 0.5967+-0.0042 0.5915+-0.0043  F=5.22 p=0.0013
Bonferroni means: ddof= 4996 MSD= 0.011 T= 1.7559 { seq rnd art }
     rnd - fep :    -0.0173165591338 True
     rnd - art :    -0.00021061108159 False
     rnd - seq :    0.00495240361391 False
     fep - art :    0.0171059480522 True
     fep - seq :    0.0222689627477 True
     art - seq :    0.0051630146955 False
```



```
space          C1  050
0.1236+-0.0042 0.1134+-0.0044 0.1159+-0.0036 0.1028+-0.0037  F=4.73 p=0.0027
Bonferroni means: ddof= 4996 MSD= 0.010 T= 1.7559 { seq }
     rnd - fep :     0.0102497613204 True
     rnd - art :     0.00772266524533 False
     rnd - seq :     0.0208224383826 True
     fep - art :     -0.0025270960751 False
     fep - seq :     0.0105726770622 True
     art - seq :     0.0130997731373 True

space          C1  090
0.2665+-0.0053 0.2561+-0.0056 0.2592+-0.0052 0.2543+-0.0054  F=0.99 p=0.3946

space          C3  050
0.3060+-0.0055 0.2911+-0.0057 0.2948+-0.0052 0.3056+-0.0051  F=2.01 p=0.1103

space          C3  090
0.5091+-0.0052 0.5042+-0.0054 0.5073+-0.0050 0.5051+-0.0053  F=0.18 p=0.9082

space          C5  050
0.4360+-0.0054 0.4300+-0.0056 0.4301+-0.0052 0.4455+-0.0054  F=1.84 p=0.1381

space          C5  090
0.6020+-0.0043 0.6001+-0.0045 0.6034+-0.0041 0.6072+-0.0044  F=0.49 p=0.6907
```

TUD-SERG-2010-007
ISSN 1872-5392