Coping with recall and precision of soft error detectors

Leonardo Bautista-Gomez, Anne Benoit, Aurélien Cavelan, Yves Robert, Hongyang Sun

▶ To cite this version:

Leonardo Bautista-Gomez, Anne Benoit, Aurélien Cavelan, Yves Robert, Hongyang Sun. Coping with recall and precision of soft error detectors. Journal of Parallel and Distributed Computing, Elsevier, 2016, 98, pp.8-24. <10.1016/j.jpdc.2016.07.007>. <hal-01354888>

HAL Id: hal-01354888

https://hal.inria.fr/hal-01354888

Submitted on 19 Aug 2016

Coping with Recall and Precision of Soft Error Detectors

Leonardo Bautista-Gomez[a], Anne Benoit[b], Aurélien Cavelan[b], Saurabh K. Raina[c], Yves Robert[b,d], Hongyang Sun[b,*]

[a]*Argonne National Laboratory, USA*
[b]*Ecole Normale Superieure de Lyon & INRIA, France*
[c]*Jaypee Institute of Information Technology, India*
[d]*University of Tennessee Knoxville, USA*

Abstract

Many methods are available to detect silent errors in high-performance computing (HPC) applications. Each method comes with a cost, a recall (fraction of all errors that are actually detected, i.e., false negatives), and a precision (fraction of true errors amongst all detected errors, i.e., false positives). The main contribution of this paper is to characterize the optimal computing pattern for an application: which detector(s) to use, how many detectors of each type to use, together with the length of the work segment that precedes each of them. We first prove that detectors with imperfect precisions offer limited usefulness. Then we focus on detectors with perfect precision, and we conduct a comprehensive complexity analysis of this optimization problem, showing NP-completeness and designing an FPTAS (Fully Polynomial-Time Approximation Scheme). On the practical side, we provide a greedy algorithm, whose performance is shown to be close to the optimal for a realistic set of evaluation scenarios. Extensive simulations illustrate the usefulness of detectors with false negatives, which are available at a lower cost than the guaranteed detectors.

1. Introduction

Failures in high-performance computing (HPC) systems have become a major issue as the number of components proliferates. Indeed, future exascale platforms are expected to be composed of hundreds of thousands of computing nodes [25]. Even if each individual node provides an optimistic mean time between failures (MTBF) of, say 100 years, the whole platform will experience a failure around every few hours on average, which is shorter than the execution time of most HPC applications. Thus, effective resilient protocols will be essential to achieve efficiency.

The de-facto general-purpose error recovery technique in HPC is checkpointing and rollback recovery [18, 29]. Such protocols employ checkpoints to periodically save the state of a parallel application so that when an error strikes some process, the application can be restored to one of its former states. However, checkpoint/restart assumes instantaneous error detection, and therefore applies to fail-stop errors. Silent errors, a.k.a. silent data corruptions (SDC), constitute another source of failures in HPC, whose threat can no longer be ignored [38, 42, 36]. There are several causes of silent errors, such as cosmic radiation, packaging pollution, among others. In contrast to a fail-stop error whose detection is immediate, a silent error is identified only when the corrupted data leads to an unusual application behavior. Such detection latency raises a new challenge: if the error struck before the last checkpoint, and is detected after that checkpoint, then the checkpoint is corrupted and cannot be used for rollback.

In order to avoid corrupted checkpoints, an effective approach consists in employing some verification mechanism and combining it with checkpointing [19, 39, 1]. The simplest protocol with this approach would be to execute a verification procedure before taking each checkpoint. If the

[☆]A preliminary version of this paper has appeared in the Proceedings of the IEEE International Conference on High Performance Computing, December 2015.

[*]Corresponding author

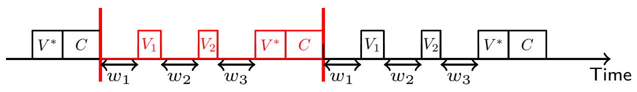


Figure 1: A periodic pattern (highlighted in red) with three segments, two partial verifications and a verified checkpoint.

verification succeeds, then one can safely store the checkpoint. Otherwise, it means that an error has struck since the last checkpoint, which was duly verified, and one can safely recover from that checkpoint to resume the execution of the application. Of course, more sophisticated protocols can be designed, by coupling multiple verifications with one checkpoint, or interleaving multiple checkpoints and verifications [1, 9]. The optimal parameter (e.g., number of verifications per checkpoint) in these protocols would be determined by the relative cost of executing a verification.

In practice, not all verification mechanisms are 100% accurate and at the same time admit fast implementations. In fact, guaranteeing accurate and efficient detection of silent errors for scientific applications is one of the hardest challenges towards extreme-scale computing [15, 16]. Indeed, thorough and general-purpose error detection is usually very costly, and often involves expensive techniques, such as replication [30] or even triplication [35]. Many applications have developed specific verification mechanisms that leverage detailed knowledge of the physics behind the simulation to determine whether the output of a simulation is corruption-free or not. While such application-specific mechanisms do not detect the totality of SDC affecting the hardware, they can guarantee to detect all the corruptions relevant for the end user, thus they can be called arguably *perfect detectors* or *guaranteed verifications*, at least from the user's perspective. For many parallel applications, alternative techniques exist that are capable of detecting silent errors but with lower accuracy. We call these techniques *partial verifications*. One example is the lightweight SDC detector based on data dynamic monitoring [3], designed to recognize anomalies in HPC datasets based on physical laws and spatial interpolation. Similar fault filters have also been designed to detect silent errors based on time series predictions [11]. Although not completely accurate, these partial verification techniques nevertheless cover a substantial number of silent errors, and more importantly, they incur very low overheads. These properties make them attractive candidates for designing more efficient resilient protocols.

Since checkpointing is often expensive in terms of both time and space required, to avoid saving corrupted data, we only keep *verified checkpoints* by placing a guaranteed verification right before each checkpoint. Such a combination ensures that the checkpoint contains valid data and can be safely written onto stable storage. The execution of the application is partitioned into *periodic patterns*, i.e., computational chunks that repeat over time, and that are delimited by verified checkpoints, possibly with a sequence of partial verifications in between. Figure 1 shows a periodic pattern with two partial verifications followed by a verified checkpoint.

The error detection accuracy of a partial verification can be characterized by two parameters: recall and precision. The *recall*, denoted by r, is the ratio between the number of detected errors and the total number of errors that occurred during a computation. The *precision*, denoted by p, is the ratio between the number of true errors and the total number of errors detected by the verification. For example, a basic spatial based SDC detector [3] has been shown to have a recall value around 0.5 and a precision value very close to 1, which means that it is capable of detecting half of the errors with almost no false alarm. A guaranteed verification can be considered as a special type of partial verification with recall $r^* = 1$ and precision $p^* = 1$. Each partial verification also has an associated *cost* V, which is typically much smaller than the cost V^* of a guaranteed verification. Note that precision and recall are conflicting objectives as they both are directly related to the allowed prediction error of the detector. If the prediction error is too small, then small changes in data behavior will produce false positives. On the other hand, if the allowed prediction error is too large, important corruption could be absorbed in the error corrupting the execution. Thus, one usually sets a target for one of them (e.g., precision = 0.999) and then measures the recall obtained with such a level of precision. Therefore, although it is hard to know in advance the precision and recall of a given detector for a particular application, it is possible to set a target for either one, and then quickly measure the complementary parameter.

An application can use several types of detectors with different overheads and accuracies. For instance, to detect silent errors in HPC datasets, one has the option of using either a detector based on time series prediction [11], or a detector using spatial multivariate interpolation [3]. The first one needs more data to make a prediction, hence comes at a higher cost. However, its accuracy is also better. In the example of Figure 1, the second verification may use a detector whose cost is lower than that of the first one, i.e., $V_2 < V_1$, but is expected to have a lower accuracy as well, i.e., $r_2 < r_1$ and/or $p_2 < p_1$. This is due to the fact that less accurate detectors perform a much simpler approximation, leading to more prediction errors.

In this paper, we assume that we have several detector types, whose costs and accuracies may differ. At the end of each segment inside the pattern, any detector can be used. The only constraint is to enforce a guaranteed verification after the last segment. Given the values of C (cost to checkpoint) and V^* (cost of guaranteed verification), as well as the cost $V^{(j)}$, recall $r^{(j)}$ and precision $p^{(j)}$ of each detector type $D^{(j)}$, the main question is which detector(s) to use? Note that we do not assume that all detectors perform equally on all applications, nor that their efficiency can be easily predicted for each type of application. The only requirement is that the accuracy and cost of those detectors can be measured in a relatively easy way. The objective is to find the optimal pattern that minimizes the expected execution time of the application. Intuitively, including more partial verifications in a pattern allows us to detect more errors earlier in the execution, thereby reducing the waste due to re-execution; but that comes at the price of additional overhead in an error-free execution, and in case of bad precision, of unnecessary rollbacks and recoveries. Therefore, an optimal strategy must seek a good tradeoff between error-induced waste and error-free overhead. The problem is intrinsically combinatorial, because there are many parameters to choose: the length of the pattern, the number of partial verifications, and the type and location of each partial verification within the pattern. Of course, the length of an optimal pattern will also depend on the platform MTBF μ.

Only very specific instances of the problem have received a solution yet. For example, when there is a single segment in the pattern without intermediate verification, the only thing to determine is the size of the segment. In the classical protocol for fail-stop errors (where verification is not needed), the optimal checkpointing period is known to be $\sqrt{2\mu C}$ (where C is the checkpoint time), as given by Young [41] and Daly [23]. A similar result is known for silent errors when using only verified checkpoints [9, 7], and in this case the optimal period is $\sqrt{\mu(V^* + C)}$. This latter case basically amounts to replacing the cost C of a checkpoint by the cost $V^* + C$ of a verified checkpoint. The factor of 2 difference is because a silent error is always detected at the end of the pattern, while a fail-stop error triggers an immediate recovery, which occurs on average in the middle of the pattern. These formulas provide first-order approximation to the length of the optimal pattern in the corresponding scenario, and are valid only if the resilience parameters satisfy $C, V^* \ll \mu$. To the best of our knowledge, the only analysis that includes partial verifications is the recent work [17], which deals with patterns that may include one or several detector(s) but all of the same type, and which considers detection recall only. While most applications accept several detector types, there has been no attempt to determine which and how many of these detectors should be used. This paper is the first to investigate the use of different types of partial detectors while taking both recall and precision into consideration.

As in those previous works, we apply first-order approximation to tackle the optimization problem. We first show that a partial detector with imperfect precision plays a limited role in the optimization of a pattern. Then we focus on detectors with perfect precision but imperfect recall, and we prove that the optimization problem is NP-complete. In this case, a detector is most useful when it offers the highest *accuracy-to-cost ratio*, defined as $\phi^{(j)} = \frac{a^{(j)}}{b^{(j)}}$, where $a^{(j)} = \frac{r^{(j)}}{2-r^{(j)}}$ denotes the accuracy of the detector and $b^{(j)} = \frac{V^{(j)}}{V^*+C}$ the relative cost. Finally, we propose a greedy algorithm and a fully polynomial-time approximation scheme (FPTAS) to solve the problem. Simulation results, based on a wide range of parameters from realistic detectors, corroborate the theoretical study by showing that the detector with the best accuracy-to-cost ratio should be favored. In some particular cases with close accuracy-to-cost ratios, an optimal pattern may use multiple detector types, but the greedy algorithm has been shown to work really well in these scenarios.

The rest of this paper is organized as follows. Section 2 surveys the related work. Section 3 introduces the model, notations and assumptions. Section 4 computes the expected execution time of a given pattern, based on which we derive some key properties of the optimal pattern in Section 5. Section 6 provides a comprehensive complexity analysis. While the optimization problem is shown to be NP-complete, a simple greedy algorithm is presented, and a fully polynomial-time approximation scheme is described. Simulation results are presented in Section 7. Finally, Section 8 provides concluding remarks and hints for future directions.

2. Related Work

Considerable efforts have been directed at detection techniques to reveal silent errors. Hardware mechanisms, such as ECC memory, can detect and even correct a fraction of errors. Unfortunately, future extreme scale systems are expected to observe an important increase in soft errors because of power constraints at increased system size. Most traditional resilient approaches maintain a single checkpoint. If the checkpoint file contains corrupted data, the application faces an irrecoverable failure and must restart from scratch. This is because error detection latency is ignored in traditional rollback and recovery schemes, which assume instantaneous error detection (therefore mainly targeting fail-stop errors) and are unable to accommodate SDC. This section describes some related work on detecting and handling silent errors.

2.1. Checkpoint Versioning

One approach to dealing with silent errors is by maintaining several checkpoints in memory [34]. This multiple-checkpoint approach, however, has three major drawbacks. First, it is very demanding in terms of stable storage: each checkpoint typically represents a copy of a large portion of the memory footprint of the application, which may well correspond to tens or even hundreds of terabytes. Second, the application cannot be recovered from fatal failures: suppose we keep k checkpoints in memory, and a silent error has struck before all of them. Then, all live checkpoints are corrupted, and one would have to re-execute the entire application from scratch. Third, even without memory constraints, we have to determine which checkpoint is the last valid one, which is needed to safely recover the application. However, due to the detection latency, we do not know when the silent error has occurred, hence we cannot identify the last valid checkpoint.

2.2. Process Replication

There are few methods that can guarantee a perfect detection recall. Process replication is one of them. The simplest technique is triple modular redundancy and voting [35]. Elliot et al. [28] propose combining partial redundancy and checkpointing, and confirm the benefit of dual and triple redundancy. Fiala et al. [30] apply process replication (each process is equipped with a replica, and messages are quadruplicated) in the RedMPI library for high-performance scientific applications. Ni et al. [37] use checkpointing and replication to detect and enable fast recovery of applications from both silent errors and hard errors. However, full process replication is too expensive to be used in extreme scale HPC systems and is usually avoided for this reason.

2.3. Application-Specific Techniques

Application-specific information can be very useful to enable ad-hoc solutions, which dramatically decrease the cost of detection. Algorithm-based fault tolerance (ABFT) [32, 12, 40] is a well-known technique, which uses checksums to detect up to a certain number of errors in linear algebra kernels. Unfortunately, ABFT can only protect datasets in linear algebra kernels, and it must be implemented for each different kernel, which incurs a large amount of work for large HPC applications. Other techniques have also been advocated. Benson, Schmit and Schreiber [10] compare the result of a higher-order scheme with that of a lower-order one to detect errors in the numerical analysis of ODEs and PDEs. Sao and Vuduc [39] investigate self-stabilizing corrections after error detection in the conjugate gradient method. Bridges et al. [13] propose linear solvers to tolerant soft faults using selective reliability. Elliot et al. [27] design a fault-tolerant GMRES capable of converging despite silent errors. Bronevetsky and de Supinski [14] provide a comparative study of detection costs for iterative methods.

2.4. Analytics-Based Corruption Detection

Recently, several SDC detectors based on data analytics have been proposed, showing promising results. These detectors use several interpolation techniques such as time series prediction [11] and spatial multivariate interpolation [3, 5, 6]. Such techniques have the benefit of offering large detection coverage for a negligible overhead. However, these detectors do not guarantee full coverage; they can detect only a certain percentage of corruptions (i.e., partial verification with an imperfect recall). Nonetheless, the accuracy-to-cost ratios of these detectors are high, which makes them interesting alternatives at large scale. Similar detectors have also been designed to detect SDCs in the temperature data of the Orbital Thermal Imaging Spectrometer (OTIS) [20]. Most of the research work done in this domain focuses on how to increase the error detection accuracy while keeping low overhead, but there has been no theoretical attempt to find the optimal protocol the applications should use when multiple verification techniques are offered by the runtime.

2.5. Optimal Strategies with Guaranteed Verifications

Theoretically, various protocols that couple verification and checkpointing have been studied. Aupy et al. [1] propose and analyze two simple patterns: one with k checkpoints and one verification, and the other with k verifications and one checkpoint. The latter pattern, which needs to maintain only one checkpoint, is also analyzed in [7] to accommodate both fail-stop and silent errors. Benoit et al. [9] extend the analysis of [1] by including p checkpoints and q verifications that are interleaved to form arbitrary patterns. All of these results assume the use of guaranteed verifications only.

As already mentioned, the only analysis that includes partial verifications in the pattern is the recent work of [17]. However, [17] restricts to a single type of partial verification, and it focuses on verifications with perfect precision. In this paper, we provide the first theoretical analysis that includes partial verifications of different types, and that considers verifications with imperfect precision.

3. Model

We consider divisible-load applications, where checkpoints and verifications can be inserted anywhere in the execution of the application. The occurrence of silent errors follows a Poisson process with arrival rate $\lambda = \frac{1}{\mu}$, where μ denotes the MTBF of the platform.

We enforce resilience through the use of a *pattern* that repeats periodically throughout the execution, as discussed in Section 1. When an error alarm is raised inside the pattern, either by a partial verification or by the guaranteed one, we roll back to the beginning of the pattern and recover from the last checkpoint (taken at the end of the execution of the previous pattern, or initial data for the first pattern). Since the last verification of the pattern is guaranteed, we need to maintain only one checkpoint at any time, and it is always valid. The objective is to find a pattern that minimizes the expected execution time of the application.

Let C denote the cost of checkpointing, R the cost of recovery and V^* the cost of guaranteed verification. Furthermore, there are k types of detectors available, and the detector type $D^{(j)}$, where $1 \leq j \leq k$, is characterized by its cost $V^{(j)}$, recall $r^{(j)}$ and precision $p^{(j)}$. For notational convenience, we also define $g^{(j)} = 1 - r^{(j)}$ (proportion of undetected errors) and let D^* be the guaranteed detector with cost V^*, recall $r^* = 1$ and precision $p^* = 1$.

A pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$ is defined by its total length W, the number n of segments in the pattern, a vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$ containing the proportions of the segment sizes, and a vector $\mathbf{D} = [D_1, D_2, \ldots, D_{n-1}, D^*]^T$ containing the detectors used at the end of each segment. We also define the vector of segment sizes $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$. Formally, for each segment i, where $1 \leq i \leq n$, w_i is the size of the segment, $\alpha_i = \frac{w_i}{W}$ is the proportion of the segment size in the whole pattern, and D_i is the detector used at the end of the segment. We have $\sum_{i=1}^{n} \alpha_i = 1$, and $\sum_{i=1}^{n} w_i = W$. If $i < n$, D_i has cost V_i, recall r_i and precision p_i (we have $D_i = D^{(j)}$ for some type j, $1 \leq j \leq k$), and $D_n = D^*$ with cost V^*, recall $r^* = 1$ and precision $p^* = 1$. Note that the same detector type $D^{(j)}$ may well be used at the end of several segments. Let $g_i = 1 - r_i$ denote the probability that the i-th detector of the pattern fails to detect an error (for $1 \leq i < n$), and let $g_{[i,j[} = \prod_{k=i}^{j-1} g_k$ be the probability that the error remains undetected by detectors D_i to D_{j-1} (for $1 \leq i < j < n$). Similarly, p_i represents the probability that the i-th detector does not raise a false alarm when there is no error, and let $p_{[i,j[} = \prod_{k=i}^{j-1} p_k$ denote the probability that no false alarm is raised by detectors D_i to D_{j-1}. In the example of Figure 1, we have $W = w_1 + w_2 + w_3$ and $n = 3$. The first partial verification has cost V_1, recall r_1 and precision p_1, and the second one has cost V_2, recall r_2 and precision p_2.

Let W_{base} denote the base time of an application without any overhead due to resilience techniques (without loss of generality, we assume unit-speed execution). Suppose the execution is divided into periodic patterns, defined by $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$. Let $\mathbb{E}(W)$ be the expected execution time of the pattern. Then, the expected makespan W_{final} of the application when taking silent errors into account can be bounded as follows:

$$\left\lfloor \frac{W_{\text{base}}}{W} \right\rfloor \times \mathbb{E}(W) \leq W_{\text{final}} \leq \left\lceil \frac{W_{\text{base}}}{W} \right\rceil \times \mathbb{E}(W).$$

This is because the execution involves $\lfloor \frac{W_{\text{base}}}{W} \rfloor$ full patterns, and terminates by a (possibly) incomplete one. For large jobs, we can approximate the execution time as

$$W_{\text{final}} \approx \frac{\mathbb{E}(W)}{W} \times W_{\text{base}}.$$

Let $H(W) = \frac{\mathbb{E}(W)}{W} - 1$ denote the execution *overhead* of the pattern. We obtain $W_{\text{final}} \approx W_{\text{base}} + H(W) \times W_{\text{base}}$. Thus, minimizing the expected makespan is equivalent to minimizing the pattern overhead $H(W)$.

We assume that errors only strike the computations, while verifications and I/O transfers (checkpointing and recovery) are protected and are thus error-free. It has been shown in [8] that removing this assumption does not affect the asymptotic behavior of a pattern. We also assume statistical independence of the detectors: if the same detector is applied twice, say at time steps t_1 and t_2, then its recall and precision are the same for both instances. This is because detectors actually detect the effect of an error on the resulting data, rather than the error itself. When an error is missed the first time at step t_1, either it dissipates and becomes harmless, or it propagates and corrupts more data. In the latter case, running the detector again after some iterations at step t_2 will actually detect the error within the precision and recall of the detector. Furthermore, to be on the safe side, we never use two detectors in a row. The idea is that after a chunk of computations, output data will be considered as random input when fed to the next detector. Understanding error propagation and correlation requires deep knowledge of the application. In this work, we provide a general-purpose solution, hence we have to rely on the independence hypothesis.

4. Expected Execution Time of a Pattern

In this section, we compute the expected execution time of a pattern by giving a closed-form formula that is exact up to second-order terms. This is a key result that will be used to derive properties of the optimal pattern in the subsequent analysis.

Consider a given pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$. The following proposition shows the expected execution time of this pattern.

Proposition 1. *The expected time to execute a pattern* $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$ *is*

$$\mathbb{E}(W) = \sum_{i=1}^{n} \frac{W\alpha_i + V_i}{p_{[i,n[}} + C + \left(\frac{1}{p_{[1,n[}} - 1\right) R + \lambda W \left(\frac{R}{p_{[1,n[}} + W\boldsymbol{\alpha}^T M \boldsymbol{\alpha} + \boldsymbol{\alpha}^T M \mathbf{v}\right) + o(\lambda), \quad (1)$$

where $\mathbf{v}$ *is an* $n \times 1$ *vector defined by* $\mathbf{v} = [V_1, V_2, \ldots, V_n]^T$*, and* M *is an* $n \times n$ *matrix defined as* $M_{ij} = \frac{1}{p_{[i,n[}}$ *for* $i \leq j$ *and* $M_{ij} = \frac{g_{[i,j[}}{p_{[j,n[}}$ *for* $i > j$*.*

Proof. Let q_i denote the probability that an error occurs in the execution of segment i. We can express the expected execution time of the pattern recursively as follows:

$$\mathbb{E}(W) = \left(\prod_{k=1}^{n}(1-q_k)\right) p_{[1,n[}C + \left(1 - \left(\prod_{k=1}^{n}(1-q_k)\right) p_{[1,n[}\right)(R + \mathbb{E}(W))$$
$$+ \sum_{i=1}^{n}\left(\sum_{j=1}^{i-1}\left(\prod_{k=1}^{j-1}(1-q_k)\right) p_{[1,j[}q_j g_{[j,i[} + \left(\prod_{k=1}^{i-1}(1-q_k)\right) p_{[1,i[}\right)(w_i + V_i). \qquad (2)$$

The first line shows that checkpointing will be taken only if no error has occurred in all the segments and no intermediate detector has raised a false alarm. This happens with probability

$$\left(\prod_{k=1}^{n}(1-q_k)\right)\left(\prod_{k=1}^{n-1}p_k\right) = \left(\prod_{k=1}^{n}(1-q_k)\right) p_{[1,n[}. \qquad (3)$$

In all the other cases, the application needs to recover from the last checkpoint and then re-computes the entire pattern. The second line shows the expected cost involved in the execution of each segment of the pattern and the associated verification. To better understand it, let us consider the third segment of size w_3 and the verification D_3 right after it, which will be executed only when the following events happen (with the probability of each event in brackets):

- There is a fault in the first segment (q_1), which is missed by the first verification ($1 - r_1 = g_1$) and again missed by the second verification ($1 - r_2 = g_2$).
- There is no fault in the first segment ($1 - q_1$), the first verification does not raise a false alarm (p_1), and there is a fault in the second segment (q_2), which is missed by the second verification ($1 - r_2 = g_2$).
- There is no fault in the first segment ($1 - q_1$), the first verification does not raise a false alarm (p_1), there is no fault in the second segment ($1 - q_2$), and the second verification does not raise a false alarm (p_2).

Thus, the expected cost involved in the execution of this segment is given by

$$\Big(q_1 g_1 g_2 + (1-q_1)p_1 q_2 g_2 + (1-q_1)p_1(1-q_2)p_2\Big)(w_3 + V_3)$$
$$= \Big(q_1 g_{[1,3[} + (1-q_1)p_{[1,2[}q_2 g_{[2,3[} + (1-q_1)(1-q_2)p_{[1,3[}\Big)(w_3 + V_3).$$

We can generalize this reasoning to express the expected cost to execute the i-th segment of the pattern, which leads to Equation (2).

Since errors arrive according to the Poisson process, by definition, we have $q_i = 1 - e^{-\lambda w_i}$. Substituting it into the recursive formula and solving for $\mathbb{E}(W)$, we obtain the expected execution time as

$$\mathbb{E}(W) = C + \left(\frac{e^{\lambda W}}{p_{[1,n[}} - 1\right) R + \sum_{i=1}^{n}\left(\sum_{j=1}^{i-1}\left(e^{\lambda W_{j,n}} - e^{\lambda W_{j+1,n}}\right)\frac{g_{[j,i[}}{p_{[j,n[}} + \frac{e^{\lambda W_{i,n}}}{p_{[i,n[}}\right)(w_i + V_i),$$

where $W_{i,j} = \sum_{k=i}^{j} w_k$. Approximating $e^{\lambda x} = 1 + \lambda x + o(\lambda)$ up to the first-order term, we can further simplify the expected execution time as

$$\begin{aligned}
\mathbb{E}(W) &= C + \left(\frac{1+\lambda W}{p_{[1,n[}} - 1\right) R + \sum_{i=1}^{n} \left(\sum_{j=1}^{i-1} \frac{\lambda w_j g_{[j,i[}}{p_{[j,n[}} + \frac{1+\lambda \sum_{j=i}^{n} w_j}{p_{[i,n[}} \right) (w_i + V_i) + o(\lambda) \\
&= \sum_{i=1}^{n} \frac{w_i + V_i}{p_{[i,n[}} + C + \left(\frac{1}{p_{[1,n[}} - 1\right) R \\
&\quad + \lambda W \frac{R}{p_{[1,n[}} + \lambda \sum_{i=1}^{n} \left(\sum_{j=1}^{i-1} \frac{w_j g_{[j,i[}}{p_{[j,n[}} + \sum_{j=i}^{n} \frac{w_j}{p_{[i,n[}} \right) (w_i + V_i) + o(\lambda).
\end{aligned}$$

Letting $F = \sum_{i=1}^{n} \left(\sum_{j=1}^{i-1} \frac{w_j g_{[j,i[}}{p_{[j,n[}} + \sum_{j=i}^{n} \frac{w_j}{p_{[i,n[}} \right) (w_i + V_i)$, we can express it in the following matrix form:

$$F = \mathbf{w}^T M \mathbf{w} + \mathbf{w}^T M \mathbf{v},$$

where M is the following $n \times n$ matrix:

$$M = \begin{bmatrix}
\frac{1}{p_{[1,n[}} & \frac{1}{p_{[1,n[}} & \frac{1}{p_{[1,n[}} & \cdots & \frac{1}{p_{[1,n[}} \\
\frac{g_{[1,2[}}{p_{[1,n[}} & \frac{1}{p_{[2,n[}} & \frac{1}{p_{[2,n[}} & \cdots & \frac{1}{p_{[2,n[}} \\
\frac{g_{[1,3[}}{p_{[1,n[}} & \frac{g_{[2,3[}}{p_{[2,n[}} & \frac{1}{p_{[3,n[}} & \cdots & \frac{1}{p_{[3,n[}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{g_{[1,n[}}{p_{[1,n[}} & \frac{g_{[2,n[}}{p_{[2,n[}} & \frac{g_{[3,n[}}{p_{[3,n[}} & \cdots & \frac{1}{p_{[n,n[}}
\end{bmatrix}.$$

For instance, when $n = 4$, we have:

$$M = \begin{bmatrix}
\frac{1}{p_1p_2p_3} & \frac{1}{p_1p_2p_3} & \frac{1}{p_1p_2p_3} & \frac{1}{p_1p_2p_3} \\
\frac{g_1}{p_1p_2p_3} & \frac{1}{p_2p_3} & \frac{1}{p_2p_3} & \frac{1}{p_2p_3} \\
\frac{g_1g_2}{p_1p_2p_3} & \frac{g_2}{p_2p_3} & \frac{1}{p_3} & \frac{1}{p_3} \\
\frac{g_1g_2g_3}{p_1p_2p_3} & \frac{g_2g_3}{p_2p_3} & \frac{g_3}{p_3} & 1
\end{bmatrix}.$$

Now, by using $\mathbf{w} = W\boldsymbol{\alpha}$, we obtain Equation (1), which completes the proof of the proposition. $\square$

5. Properties of Optimal Pattern

In this section, we characterize the properties of the optimal pattern. First, we derive the optimal length of a pattern (Section 5.1). Then, we show that the optimal pattern does not contain partial detectors with imperfect precision (Section 5.2). By focusing on detectors with perfect precision, we define two key parameters to characterize a pattern (Section 5.3). Finally, we obtain the optimal positions for a give set of partial verifications (Section 5.4).

5.1. Optimal Length of a Pattern

We first compute the optimal length W of a pattern PATTERN$(W, n, \boldsymbol{\alpha}, \mathbf{D})$ in order to minimize its execution overhead $H(W)$.

Theorem 1. *The execution overhead of a pattern* PATTERN$(W, n, \boldsymbol{\alpha}, \mathbf{D})$ *is minimized when its length is*

$$W^* = \sqrt{\frac{\sum_{i=1}^{n} \frac{V_i}{p_{[i,n[}} + C + \left(\frac{1}{p_{[1,n[}} - 1\right) R}{\lambda \boldsymbol{\alpha}^T M \boldsymbol{\alpha}}}. \quad (4)$$

In that case, the overhead is given by

$$H(W^*) = 2\sqrt{\lambda \boldsymbol{\alpha}^T M \boldsymbol{\alpha} \left(\sum_{i=1}^{n} \frac{V_i}{p_{[i,n[}} + C + \left(\frac{1}{p_{[1,n[}} - 1 \right) R \right)} + \sum_{i=1}^{n} \left(\frac{1}{p_{[i,n[}} - 1 \right) \alpha_i + o(\sqrt{\lambda}). \quad (5)$$

Proof. From the expected execution time of a pattern given in Equation (1), we can derive the overhead as follows:

$$H(W) = \frac{\mathbb{E}(W)}{W} - 1 = \frac{\sum_{i=1}^{n} \frac{V_i}{p_{[i,n[}} + C + \left(\frac{1}{p_{[1,n[}} - 1 \right) R}{W} + \lambda W \boldsymbol{\alpha}^T M \boldsymbol{\alpha} + \sum_{i=1}^{n} \left(\frac{1}{p_{[i,n[}} - 1 \right) \alpha_i + \lambda \left(\frac{R}{p_{[1,n[}} + \boldsymbol{\alpha}^T M \mathbf{v} \right) + o(\lambda). \quad (6)$$

The optimal pattern length that minimizes the execution overhead can now be computed by balancing the first two terms of the above equation, which gives rise to Equation (4). Now, substituting W^* back into Equation (6), we can obtain the execution overhead shown in Equation (5). Note that when the platform MTBF $\mu = 1/\lambda$ is large in front of the resilience parameters, the last two terms of Equation (6) become negligible compared to other dominating terms given in Equation (5), so they are absorbed into $o(\sqrt{\lambda})$. □

5.2. Usefulness of Imprecise Detectors

We now assess the usefulness of partial detectors with imperfect precision. We show that an imprecise partial verification (i.e., with $p < 1$) is not used in the optimal pattern. The result is valid when the platform MTBF $\mu = 1/\lambda$ is large in front of the resilience parameters, and when the precision values are constants and independent of the error rate λ.

Theorem 2. *The optimal pattern contains no detector with imprecise verification.*

Proof. We show that given any pattern containing imprecise verifications, we can transform it into one that does not use any imprecise verification and that has a better execution overhead.

Consider a given pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$ that contains imprecise verifications. Theorem 1 gives the optimal length of the pattern as well as the execution overhead in that case. From Equation (5), we observe that the overhead is dominated by the term $\sum_{i=1}^{n} \left(\frac{1}{p_{[i,n[}} - 1 \right) \alpha_i$, if the precisions of all detectors are constants and independent of the error rate λ. Assuming that the size of each segment in the pattern is also a constant fraction of the pattern length, we can improve the overhead by making α_i approach 0 for all segment i with $p_{[i,n[} < 1$. Suppose segment m is the first segment that satisfies $p_{[m,n[} = 1$. Then the execution overhead of the pattern becomes

$$H = 2\sqrt{\lambda \boldsymbol{\alpha}^T M \boldsymbol{\alpha} \left(\sum_{i=1}^{n} \frac{V_i}{p_{[i,n[}} + C + \left(\frac{1}{p_{[1,n[}} - 1 \right) R \right)} + o(\sqrt{\lambda}),$$

where $\boldsymbol{\alpha} = [0, \dots, 0, \alpha_m, \dots, \alpha_n]^T$. Now, by removing the first $m - 1$ detectors while keeping the relative sizes of the remaining segments unchanged, we get a new pattern whose overhead is

$$H' = 2\sqrt{\lambda \boldsymbol{\alpha}'^T M' \boldsymbol{\alpha}' \left(\sum_{i=m}^{n} V_i + C \right)} + o(\sqrt{\lambda}),$$

where $\boldsymbol{\alpha}' = [\alpha_m, \dots, \alpha_n]^T$ and M' is the submatrix of M by removing the first $m - 1$ rows and columns. Clearly, we have $H' < H$ since $\sum_{i=m}^{n} V_i + C < \sum_{i=1}^{n} \frac{V_i}{p_{[i,n[}} + C + \left(\frac{1}{p_{[1,n[}} - 1 \right) R$ and $\boldsymbol{\alpha}'^T M' \boldsymbol{\alpha}' = \boldsymbol{\alpha}^T M \boldsymbol{\alpha}$. □

Theorem 2 is valid up to first-order estimations and shows that an imprecise partial verification should not be used when the platform MTBF is large. Intuitively, this is because a low precision induces too much re-execution overhead when the error rate is small, making the verification unworthy. Again, we point out that this result holds when the precision can be considered as a constant, which is true in practice as the accuracy of a detector is independent of the error rate. In fact, many practical fault filters do have almost perfect precision under realistic settings [4, 20, 21]. Still, the result is striking, because it is the opposite of what is observed for predictors, for which recall matters more than precision [2].

In the rest of this paper, we will focus on partial verifications with perfect precision (i.e., $p = 1$) but imperfect recall (i.e., $r < 1$).

5.3. Two Key Parameters

For a pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$ and assuming that all detectors have perfect precision, the expected execution time of the pattern according to Proposition 1 is given by

$$\mathbb{E}(W) = W + \sum_{i=1}^{n} V_i + C + \lambda W \left(R + W \boldsymbol{\alpha}^T M \boldsymbol{\alpha} + \boldsymbol{\alpha}^T M \mathbf{v} \right) + o(\lambda),$$

where M is an $n \times n$ matrix defined by $M_{ij} = 1$ for $i \le j$ and $M_{ij} = g_{[i,j[}$ for $i > j$.

To characterize such as pattern, we introduce two key parameters in the following.

Definition 1. *The* fault-free overhead o_{ff} *of a pattern* $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$ *is*

$$o_{\mathit{ff}} = \sum_{i=1}^{n} V_i + C, \tag{7}$$

and the fraction of re-executed work in case of faults is

$$f_{re} = \boldsymbol{\alpha}^T M \boldsymbol{\alpha}. \tag{8}$$

According to Theorem 1, we can get the optimal pattern length and execution overhead as

$$W^* = \sqrt{\frac{o_{\text{ff}}}{\lambda f_{\text{re}}}},$$
$$H(W^*) = 2\sqrt{\lambda o_{\text{ff}} f_{\text{re}}} + o(\sqrt{\lambda}).$$

The equation above shows that when the platform MTBF $\mu = 1/\lambda$ is large in front of the resilience parameters, the expected execution overhead of the optimal pattern is dominated by $2\sqrt{\lambda o_{\text{ff}} f_{\text{re}}}$. The problem is then reduced to the minimization of the product $o_{\text{ff}} f_{\text{re}}$. Intuitively, this calls for a tradeoff between fault-free overhead and fault-induced re-execution, as a smaller fault-free overhead o_{ff} tends to induce a larger re-execution fraction f_{re}, and vice versa.

5.4. Optimal Positions of Verifications

To fully characterize an optimal pattern, we have to determine its number of segments, as well as the type and position of each partial verification. In this section, we consider a pattern whose number of segments is given together with the types of all partial verifications, that is, the value of o_{ff} (Equation (7)) is given. We show how to determine the optimal length of each segment (or equivalently, the optimal position of each verification), so as to minimize the value of f_{re} (Equation (8)). The following theorem shows the result. It is the most technically involved contribution of this paper, and its lengthy proof can be found in the appendix.

Theorem 3. *Consider a pattern* $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$ *where* W*,* n*, and* $\mathbf{D}$ *are given. The fraction of re-executed work* f_{re} *is minimized when* $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$*, where*

$$\alpha_k^* = \frac{1}{U_n} \cdot \frac{1 - g_{k-1} g_k}{(1 + g_{k-1})(1 + g_k)} \quad \text{for } 1 \le k \le n, \tag{9}$$

with $g_0 = g_n = 0$ and

$$U_n = 1 + \sum_{i=1}^{n-1} \frac{1-g_i}{1+g_i}. \tag{10}$$

In that case, the value of f_{re} is

$$f_{re}^* = \frac{1}{2}\left(1 + \frac{1}{U_n}\right). \tag{11}$$

Note that when all the partial verifications in the pattern have the same type, i.e., $g_k = g$ for all $1 \le k < n$, we retrieve the result of [17], obtaining $f_{\mathrm{re}}^* = \frac{1}{2}\left(1 + \frac{1+g}{n(1-g)+2g}\right)$ with

$$\alpha_k^* = \begin{cases} \frac{1}{n(1-g)+2g} & \text{for } k = 1, n \\ \frac{1-g}{n(1-g)+2g} & \text{for } k = 2, \dots, n-1 \end{cases}.$$

The result shows that when there is only one partial verification in the pattern, i.e., $n = 2$, the two resulting segments share the same length, i.e., $\alpha_1 = \alpha_2 = \frac{1}{2}$. With two or more partial verifications of the same type, the left-most and the right-most segments, each being adjacent to a guaranteed verification, are longer than all the intermediate segments, which have the same length.

Theorem 3 also shows that, for a given set of partial verifications in a pattern, the minimum value of f_{re} does not depend upon their ordering within the pattern.

Corollary 1. *For a given set of partial verifications within a pattern, the minimum fraction of re-executed work f_{re}^* is independent of their ordering.*

6. Complexity

This section builds upon the previous results to provide a comprehensive complexity analysis. We introduce the *accuracy-to-cost ratio* of a detector and show that it is the key parameter to compute the optimal rational solution (Section 6.1). Then we establish the NP-completeness to determine the optimal integer solution (Section 6.2). On the positive side, we design a simple greedy algorithm whose performance is guaranteed, and sketch the construction of an FPTAS for the problem (Section 6.3).

6.1. Accuracy-to-Cost Ratio and Rational Solution

Consider a pattern $\textsc{Pattern}(W, n, \boldsymbol{\alpha}, \mathbf{D})$. Let m_j denote the number of partial verifications using detector type $D^{(j)}$ in the pattern (the number of indices $i < n$ such that D_i is of type $D^{(j)}$), and define $\mathbf{m} = [m_1, m_2, \dots, m_k]$. Section 5.1 shows that minimizing the execution overhead of the pattern is equivalent to minimizing the product $o_{\mathrm{ff}} f_{\mathrm{re}}$. From Equations (7) and (11), we have $o_{\mathrm{ff}} f_{\mathrm{re}} = \frac{V^*+C}{2} f(\mathbf{m})$, where

$$f(\mathbf{m}) = \left(1 + \frac{1}{1 + \sum_{j=1}^{k} m_j a^{(j)}}\right)\left(1 + \sum_{j=1}^{k} m_j b^{(j)}\right). \tag{12}$$

In Equation (12), we define $a^{(j)} = \frac{1-g^{(j)}}{1+g^{(j)}}$ to be the *accuracy* of detector $D^{(j)}$ and define $b^{(j)} = \frac{V^{(j)}}{V^*+C}$ to be the *relative cost* of $D^{(j)}$. Furthermore, we define $\phi^{(j)} = \frac{a^{(j)}}{b^{(j)}}$ to be the *accuracy-to-cost ratio* of $D^{(j)}$. We will show that this ratio plays a key role in selecting the best detector(s).

Altogether, minimizing the pattern overhead amounts to finding the solution $\mathbf{m} = [m_1, m_2, \dots, m_k]$ that minimizes $f(\mathbf{m})$, with $m_j \in \mathbb{N}_0$ for all $1 \le j \le k$. Indeed, once $\mathbf{m}$ is given, Proposition 1 and Theorem 3 completely characterize the optimal pattern, giving its length W, the number of segments $n = \sum_{j=1}^{k} m_j + 1$, and the locations $\boldsymbol{\alpha}$ of all partial detectors (whose ordering does not matter).

We first derive the optimal solution if we relax the integer constraint on $\mathbf{m}$. A rational solution in this case is denoted by $\bar{\mathbf{m}} = [\bar{m}_1, \bar{m}_2, \dots, \bar{m}_k]$ with $\bar{m}_j \ge 0$ for all $1 \le j \le k$. The optimal value of $f(\bar{\mathbf{m}})$ is a lower bound on the optimal integer solution.

Lemma 1. *Suppose there are k types of detectors sorted in non-increasing order of accuracy-to-cost ratio, i.e., $\phi^{(1)} \geq \phi^{(2)} \geq \cdots \geq \phi^{(k)}$. Then,*

$$f^*(\bar{\mathbf{m}}) = \begin{cases} \left(\sqrt{\frac{1}{\phi^{(1)}}} + \sqrt{1 - \frac{1}{\phi^{(1)}}}\right)^2 & \text{if } \phi^{(1)} > 2 \\ 2 & \text{otherwise} \end{cases}.$$

Proof. First, we prove that the optimal rational solution is achieved when only the detector with the largest accuracy-to-cost ratio $\phi^{(1)}$ is used. Specifically, given any rational solution $\bar{\mathbf{m}} = [\bar{m}_1, \bar{m}_2, \ldots, \bar{m}_k]$, we show that there exists a solution $\bar{\mathbf{m}}' = [\bar{m}'_1, 0, \ldots, 0]$, which satisfies $f(\bar{\mathbf{m}}') \leq f(\bar{\mathbf{m}})$. We have

$$\begin{aligned} f(\bar{\mathbf{m}}) &= \left(1 + \frac{1}{1 + \sum_{j=1}^{k} \bar{m}_j a^{(j)}}\right)\left(1 + \sum_{j=1}^{k} \bar{m}_j b^{(j)}\right) \\ &= \left(1 + \frac{1}{1 + a^{(1)} \sum_{j=1}^{k} \frac{\bar{m}_j a^{(j)}}{a^{(1)}}}\right)\left(1 + b^{(1)} \sum_{j=1}^{k} \frac{\bar{m}_j b^{(j)}}{b^{(1)}}\right). \end{aligned} \tag{13}$$

Let $\bar{m}'_1 = \sum_{j=1}^{k} \frac{\bar{m}_j a^{(j)}}{a^{(1)}}$ and $\bar{n}'_1 = \sum_{j=1}^{k} \frac{\bar{m}_j b^{(j)}}{b^{(1)}}$. Since $\frac{b^{(j)}}{b^{(1)}} \geq \frac{a^{(j)}}{a^{(1)}}$ for all $1 \leq j \leq k$, we get $\bar{n}'_1 = \sum_{j=1}^{k} \frac{\bar{m}_j b^{(j)}}{b^{(1)}} \geq \sum_{j=1}^{k} \frac{\bar{m}_j a^{(j)}}{a^{(1)}} = \bar{m}'_1$. Hence, Equation (13) can be written as

$$\begin{aligned} f(\bar{\mathbf{m}}) &= \left(1 + \frac{1}{1 + a^{(1)}\bar{m}'_1}\right)\left(1 + b^{(1)}\bar{n}'_1\right) \\ &= \left(1 + \frac{1}{1 + a^{(1)}\bar{m}'_1}\right)\left(1 + \frac{b^{(1)}\bar{n}'_1}{\bar{m}'_1} \cdot \bar{m}'_1\right) \\ &\geq \left(1 + \frac{1}{1 + a^{(1)}\bar{m}'_1}\right)\left(1 + b^{(1)}\bar{m}'_1\right) \\ &= f(\bar{\mathbf{m}}'). \end{aligned}$$

Now, define $f(\bar{m}) = \left(1 + \frac{1}{1 + a^{(1)}\bar{m}}\right)\left(1 + b^{(1)}\bar{m}\right)$. The following derives the minimum value of $f(\bar{m})$. Differentiating $f(\bar{m})$ with respect to $\bar{m}$ and solving $\frac{\partial f(\bar{m})}{\partial \bar{m}} = 0$, we get

$$\bar{m}^* = -\frac{1}{a^{(1)}} + \sqrt{\frac{1}{a^{(1)}}\left(\frac{1}{b^{(1)}} - \frac{1}{a^{(1)}}\right)}, \tag{14}$$

which is positive (hence a potential solution) if $\phi^{(1)} = \frac{a^{(1)}}{b^{(1)}} > 2$. Taking the second-order derivative of $f(\bar{m})$, we get

$$\frac{\partial^2 f(\bar{m})}{\partial \bar{m}^2} = \frac{2a^{(1)}(a^{(1)} - b^{(1)})}{(a^{(1)}\bar{m} + 1)^3},$$

which is positive (hence ensures that the solution is the unique minimum) for all $\bar{m} \in [0, \infty)$ if $\phi^{(1)} = \frac{a^{(1)}}{b^{(1)}} > 1$.

Thus, when $\phi^{(1)} > 2$, the optimal solution is obtained by substituting $\bar{m}^*$ into $f(\bar{m})$, and we get

$$\begin{aligned} f(\bar{m}^*) &= \left(1 + \frac{1}{1 + a^{(1)}\bar{m}^*}\right)\left(1 + b^{(1)}\bar{m}^*\right) \\ &= \left(1 + \frac{1}{\sqrt{\phi^{(1)} - 1}}\right)\left(1 - \frac{1}{\phi^{(1)}} + \sqrt{\frac{1}{\phi^{(1)}}\left(1 - \frac{1}{\phi^{(1)}}\right)}\right) \\ &= \frac{\phi^{(1)} - 1}{\phi^{(1)}} + 2\frac{\sqrt{\phi^{(1)} - 1}}{\phi^{(1)}} + \frac{1}{\phi^{(1)}} \\ &= \left(\sqrt{\frac{1}{\phi^{(1)}}} + \sqrt{1 - \frac{1}{\phi^{(1)}}}\right)^2. \end{aligned}$$

When $\phi^{(1)} \leq 2$, the minimum value of $f(\bar{m})$ is achieved at $\bar{m} = 0$, which gives $f(0) = 2$. □

Lemma 1 shows that the optimal rational solution is achieved with only one detector, namely, the one with the highest accuracy-to-cost ratio. The optimal integer solution, however, may use more than one detector. The following shows that finding the optimal integer solution is NP-complete.

6.2. NP-Completeness

We show that finding the optimal integer solution $\mathbf{m}$ is NP-complete, even when all detectors share the same accuracy-to-cost ratio. In particular, we consider the following decision problem.

Definition 2 (Multiple Partial Verifications (MPV))**.** *Given k detectors with the same accuracy-to-cost ratio ϕ, i.e., $\frac{a^{(j)}}{b^{(j)}} = \phi$ for all $1 \leq j \leq k$, and a bound K, is there a solution $\mathbf{m}$ that satisfies*

$$\left(1 + \frac{1}{1 + \sum_{j=1}^{k} m_j a^{(j)}}\right)\left(1 + \sum_{j=1}^{k} m_j b^{(j)}\right) \leq K? \tag{15}$$

Theorem 4. *The MPV problem is NP-complete.*

Proof. The MPV problem is obviously in NP. We prove the completeness by a reduction from the *Unbounded Subset Sum (USS)* problem, which is known to be NP-complete [31]. Given a multiset $S = \{s_1, s_2, \ldots, s_k\}$ of k positive integers and a positive integer I, the USS problem asks if there exists a subset $S' \subseteq S$ whose sum is exactly I, i.e., $\sum_{j=1}^{k} m_j s_j = I$, where $m_j \in \mathbb{N}_0$. We can further assume that I/s_j is not an integer for $1 \leq j \leq k$, since otherwise we would have a trivial solution.

Given an instance of the USS problem, we construct an instance of the MPV problem with k detectors. First, choose any $\phi \in \left(2, (I/s_{\max} + 1)^2 + 1\right)$, where $s_{\max} = \max_{j=1..k} s_j$. Then, let $\frac{a}{b} = \phi$ and $-\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)} = I$, so we can get $a = \frac{\sqrt{\phi-1}-1}{I}$ and $b = \frac{\sqrt{\phi-1}-1}{\phi I}$. For each $1 \leq j \leq k$, define $a^{(j)} = s_j a$ and $b^{(j)} = s_j b$. According to the range of ϕ, we have $a^{(j)} < 1$ and $b^{(j)} < 1$ for all $1 \leq j \leq k$. Finally, let $K = \left(\sqrt{\frac{1}{\phi}} + \sqrt{1 - \frac{1}{\phi}}\right)^2$.

If we use only one detector, say $D^{(j)}$, then Lemma 1 shows that Equation (15) is satisfied with the following unique solution:

$$\begin{aligned} m_j^* &= -\frac{1}{a^{(j)}} + \sqrt{\frac{1}{a^{(j)}}\left(\frac{1}{b^{(j)}} - \frac{1}{a^{(j)}}\right)} \\ &= \frac{1}{s_j}\left(-\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)}\right) = \frac{I}{s_j}, \end{aligned}$$

which is not an integer by hypothesis, but achieves the lower bound $\left(\sqrt{\frac{1}{\phi}} + \sqrt{1 - \frac{1}{\phi}}\right)^2 = K$. Now, we show that, by using multiple detectors, an integer solution to the MPV instance exists if and only if there is an integer solution to the USS instance.

($\Rightarrow$) Suppose there is an integer solution $\mathbf{m} = [m_1, m_2, \ldots, m_k]$ such that $\sum_{j=1}^{k} m_j s_j = I$.

Then, by employing m_j partial verifications of detector type $D^{(j)}$ for $1 \le j \le k$, we get

$$\begin{aligned}
&\left(1 + \frac{1}{1 + \sum_{j=1}^{k} m_j a^{(j)}}\right)\left(1 + \sum_{j=1}^{k} m_j b^{(j)}\right) \\
&= \left(1 + \frac{1}{1 + a\sum_{j=1}^{k} m_j s_j}\right)\left(1 + b\sum_{j=1}^{k} m_j s_j\right) \\
&= \left(1 + \frac{1}{1 + aI}\right)(1 + bI) \\
&= \left(\sqrt{\frac{1}{\phi}} + \sqrt{1 - \frac{1}{\phi}}\right)^2 = K.
\end{aligned}$$

($\Leftarrow$) Suppose there is an integer solution $\mathbf{m} = [m_1, m_2, \ldots, m_k]$ to the MPV instance such that

$$\left(1 + \frac{1}{1 + \sum_{j=1}^{k} m_j a^{(j)}}\right)\left(1 + \sum_{j=1}^{k} m_j b^{(j)}\right) = K.$$

This implies

$$\begin{aligned}
&\left(1 + \frac{1}{1 + a\sum_{j=1}^{k} m_j s_j}\right)\left(1 + b\sum_{j=1}^{k} m_j s_j\right) \\
&= 1 + 2\sqrt{\frac{1}{\phi}\left(1 - \frac{1}{\phi}\right)}.
\end{aligned}$$

Let $T = \sum_{j=1}^{k} m_j s_j$. Solving T from the equation above, we get the following unique solution:

$$T = -\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)} = I.$$

This completes the proof of the theorem. □

6.3. Greedy Algorithm and FPTAS

To cope with the NP-completeness of minimizing $o_{\text{ff}} f_{\text{re}}$, there is a simple and intuitive greedy algorithm. This greedy algorithm uses only the detector with the highest accuracy-to-cost ratio $\phi^{(1)}$. We compute the optimal rational number of partial verifications $\bar{m}^*$ (from Equation (14)) and then round it up if it is not an integer. In Section 7, we show that this algorithm performs quite well in practice.

Interestingly, we can guarantee the performance of this simple algorithm. From Lemma 1, we can assume $\phi^{(1)} = \frac{a^{(1)}}{b^{(1)}} > 2$. Since $a^{(1)} < 1$, we can get $b^{(1)} < 1/2$. If the optimal fractional solution $\bar{m}^*$ given in Equation (14) happens to be an integer, then we get the optimal solution. Otherwise, rounding it to $\lceil \bar{m}^* \rceil$ increases the objective function $f(\mathbf{m})$ shown in Equation (12) by at most a factor of $\delta = 1 + b^{(1)} < 3/2$. According to Equation (5), this gives a $\sqrt{3/2}$-approximation algorithm for minimizing the expected execution overhead (and hence the makespan).

In the following, we show that it is possible to have a fully polynomial-time approximation scheme (FPTAS), which ensures, for any $\epsilon > 0$, that the solution is within $1 + \epsilon$ times the optimal, and that the running time of the algorithm is polynomial in the input size and $1/\epsilon$. To develop the FPTAS, we perform the following transformations to the problem.

First, we convert all parameters in Equation (12) to integers. Since $a^{(j)} = \frac{1-g^{(j)}}{1+g^{(j)}} = \frac{r^{(j)}}{2-r^{(j)}} \le 1$ and $r^{(j)}$ is rational, we can write $a^{(j)} = \frac{p_j}{q_j}$, where p_j and q_j are positive integers with $p_j \le q_j$. We assume that C, V^* and all the $V^{(j)}$'s are also integers. Thus, minimizing $f(\mathbf{m})$ is equivalent to minimizing the following function:

$$F(\mathbf{m}) = \left(1 + \frac{L}{L + \sum_{j=1}^{k} m_j L^{(j)}}\right)\left(C + V^* + \sum_{j=1}^{k} m_j V^{(j)}\right),$$

where L denotes the least common multiple (LCM) of $q_1, q_2, \ldots, q_k$, and $L^{(j)} = \frac{p_j}{q_j} L \leq L$. Clearly, L and all the $L^{(j)}$'s can be represented by a polynomial function of the original input size.

Next, we compute an upper bound on the number of partial verifications. Observe that $F(\mathbf{0}) = 2(C + V^*)$ and $F(\mathbf{m}) \geq C + V^* + \sum_{j=1}^{k} m_j V^{(j)}$. This implies that the optimal solution must satisfy $m_j \leq \frac{C+V^*}{V^{(j)}}$ for all $1 \leq j \leq k$. Therefore, it follows that $\sum_{j=1}^{k} m_j V^{(j)} \leq k(C + V^*)$. The bound on m_j allows us to transform the unbounded problem to the 0-1 problem by providing $\lfloor \log m_j \rfloor$ additional copies of each item type j with doubling $V^{(j)}$ and $L^{(j)}$ values. This is a standard technique also used in transforming the bounded and unbounded knapsack problems to the 0-1 knapsack problem [33]. The total number of items becomes $K = \sum_{j=1}^{k} (1 + \lfloor \log m_j \rfloor) = O(k \log(C + V^*))$, which stays polynomial in the input size.

Define $\mathbf{x} = [x_1, x_2, \ldots, x_K]$, and let L_j and V_j be the value and cost of item j, respectively. We can now formulate the optimization problem as follows:

$$\begin{aligned} \text{minimize } & F(\mathbf{x}) = \left(1 + \frac{L}{L + \sum_{j=1}^{K} x_j L_j}\right)\left(C + V^* + \sum_{j=1}^{K} x_j V_j\right) \\ \text{subject to } & \sum_{j=1}^{K} x_j V_j \leq k(C + V^*) \\ & x_j \in \{0, 1\} \quad \forall j = 1, 2, \ldots, K \end{aligned}$$

and the size of all parameters is a polynomial function of the input size of the original problem. To find an FPTAS for the problem above, we adopt the technique used in [22] for designing an FPTAS for the *Maximum Density Knapsack (MDK)* problem described below.

Maximum Density Knapsack (MDK): Given a set $S = \{s_1, s_2, \ldots, s_K\}$ of K items, where each item $s_j \in S$ has a positive integer profit p_j and a positive integer weight w_j, a total capacity W, and an initial weight w_0, the MDK problem is formulated as:

$$\begin{aligned} \text{maximize } & \frac{\sum_{j=1}^{K} x_j p_j}{w_0 + \sum_{j=1}^{K} x_j w_j} \\ \text{subject to } & \sum_{j=1}^{K} x_j w_j \leq W \\ & x_j \in \{0, 1\} \quad \forall j = 1, 2, \ldots, K \end{aligned}$$

Cohen and Katzir [22] give an FPTAS for the MDK problem by using the existing FPTAS for the knapsack problem [33]. In particular, their algorithm relies on the property that, for every profit P, a minimum weight solution $\mathbf{x}$ is found such that $P(\mathbf{x}) = \sum_{j=1}^{K} x_j p_j \geq \lfloor \frac{P}{1+\epsilon'} \rfloor$, for any $\epsilon' > 0$. This immediately gives rise to an FPTAS for MDK.

We can apply the same technique to construct an FPTAS for minimizing $F(\mathbf{x})$. Let $\mathbf{x}_{opt}$ denote the optimal solution. By considering V_j as weight and L_j as profit, we can run the FPTAS for knapsack and return in polynomial time a solution $\mathbf{x}$ that satisfies $P(\mathbf{x}) \geq \lfloor \frac{P(\mathbf{x}_{opt})}{1+\epsilon'} \rfloor$ and $W(\mathbf{x}) \leq W(\mathbf{x}_{opt})$. By setting carefully the value of ϵ' as a function of ϵ, the solution yields $F(\mathbf{x}) \leq (1 + \epsilon)F(\mathbf{x}_{opt})$. The detail is similar to the one presented in [22] and is omitted here.

7. Performance Evaluation

In this section, we assess the benefits of partial detectors and evaluate the performance improvement they can provide. Both Maple-based evaluations using the performance model and realistic simulations using fault-injection are conducted. We consider four scenarios. In the first scenario, we study the optimal algorithm using only a single detector type. In the second scenario, we study the impact of the number of partial verifications on the overhead and the optimal pattern length. The third scenario tackles applications with various datasets that expose a range of recall values for each detector rather than a single value. Finally, in the fourth scenario, we focus on the greedy algorithm and compare its performance with the optimal solution that uses more than one type of partial detectors. The simulator code is available for download at `http://graal.ens-lyon.fr/~yrobert/two-level.zip`, so that interested readers can experiment with it and build relevant scenarios of their choice.

7.1. Simulation Setup

We have chosen realistic parameters that depict a typical future exascale platform. The target platform consists of 10^5 nodes whose individual MTBF is 100 years, which amounts to a platform MTBF of $\mu = 31536$ seconds (i.e., about 8.7 hours). The global size of the memory for an exascale machine is expected to be between 32 PB and 64 PB; divided by the number of nodes (10^5), the memory size per node goes from 320 GB to 640 GB. Most HPC applications try to populate 90% of the node memory but only $10\% - 50\%$ of the memory is checkpointed. That makes the checkpoint size between 30 GB and 300 GB. At exascale, most checkpoints will be done in local non-volatile memory (NVM), which is known to be slower than DRAM. We assume checkpoint throughput between 0.5 GB/s and 1 GB/s. While the results presented in this paper are based on these given parameters, we encourage the readers to validate the model with different architecture characteristics.

Concerning the detectors, we assume that they have an almost perfect precision, otherwise we would not use them, as shown in Section 5.2. Thus, the detectors will be configured to adapt their prediction error in order to minimize the number of false positives (i.e., maximize precision) at the cost of some recall. Previous studies [24] have shown that such configuration can lead to different levels of recall depending on the prediction method and the dataset behavior. Nevertheless, this large study with over 20 different types of simulations showed some trends in performance and efficacy, and we base our simulation parameters in those results. The first detector $D^{(1)}$ has a throughput of about 200 MB/s/process and a recall of 0.5 [3, 6]. The second one $D^{(2)}$ has a throughput of about 20 MB/s/process and a recall of 0.95 [11]. If we assume 512 processes per node at exascale, then the node throughput of the detectors becomes 100 GB/s for $D^{(1)}$ and 10 GB/s for $D^{(2)}$. Finally, we assume a third detector $D^{(3)}$, which is an optimized version that combines the features of the first two detectors, achieving a recall of 0.8 and a throughput of 50 GB/s. Concerning the perfect detector D^*, we assume a throughput of 0.5 GB/s based on the fact that application-specific detectors are usually based on physical properties such as mass or energy conservation, which requires global communications and is therefore more expensive than purely local checks.

The simulator generates errors following an exponential distribution of parameter λ. An experiment goes as follows. We feed the simulator with the description of the platform consisting of the parameters described above. For each pattern, we derive the (near) optimal pattern by computing the pattern length W^*, and the optimal number m^* and positions $\boldsymbol{\alpha}^*$ of verifications, using the formulas from our model. The total amount of work for the application is then set to that of 1000 optimal patterns, i.e., $W_{\text{base}} = 1000W^*$. The simulator runs each experiment 1000 times, and the simulated overhead is obtained by averaging the results from the 1000 runs.

7.2. Scenario 1: Performance of Different Detectors

In the first scenario, we study the optimal algorithm when using a single detector type. Three detectors $D^{(1)}$, $D^{(2)}$ and $D^{(3)}$ are used separately, with respective costs and recall values $V^{(1)} = 3$ seconds, $V^{(2)} = 30$ seconds, $V^{(3)} = 6$ seconds and $r^{(1)} = 0.5$, $r^{(2)} = 0.95$, $r^{(3)} = 0.8$. The

Table 1: Characteristics of all detector types and the performance of the optimal pattern using each detector type alone.

	$D^{(1)}$	$D^{(2)}$	$D^{(3)}$	D^*
Cost V (seconds)	3	30	6	600
Recall r	0.5	0.95	0.8	1
Accuracy-to-cost ratio ϕ	133	36	133	2
Predicted overhead H^*	29.872%	31.798%	29.872%	39.014%
Optimal W^* (hours)	2.41	2.38	2.41	1.71
Optimal m^*	33	6	17	0
Simulated overhead	30.313%	32.537%	30.743%	40.414%
Ave. # checkpoints (per day)	7.28	7.23	7.25	9.50
Ave. # recoveries (per day)	2.26	2.25	2.33	1.94

checkpointing cost and the perfect detector cost with recall $r^* = 1$ are fixed at $C = V^* = 600$ seconds.

Table 1 summarizes the characteristics of all detector types including the perfect detector, and presents the predicted performance of the optimal pattern using each detector alone. Recall that the accuracy-to-cost ratio is defined as $\phi^{(j)} = \frac{a^{(j)}}{b^{(j)}}$, where $a^{(j)} = \frac{r^{(j)}}{2-r^{(j)}}$ denotes the accuracy of the detector and $b^{(j)} = \frac{V^{(j)}}{V^*+C}$ the relative cost. Thanks to the higher accuracy-to-cost ratios, the use of partial verifications yields much better performance compared to the baseline algorithm that uses only guaranteed verification. In particular, $D^{(1)}$ and $D^{(3)}$, which have the highest accuracy-to-cost ratio, offer about 10% improvement in the execution overhead. This translates to about 1 hour of saving for every 10 hours of execution, which is significant in terms of cost and resource usage. The optimal pattern also employs a larger number m^* of partial verifications, due to their lower costs, so that checkpoints can be taken less frequently (i.e., with a larger period W^*).

It is interesting to observe, for $D^{(1)}$ and $D^{(3)}$, that the product of cost and frequency (number of verifications to use in a pattern) is roughly equal. Indeed, for detectors with the same accuracy-to-cost ratio ϕ, our analysis shows that the cost-frequency product is in fact a constant, and it is given by $V\bar{m}^* = V\left(-\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)}\right) = (V^* + C)\left(-\frac{1}{\phi} + \sqrt{\frac{1}{\phi}\left(1 - \frac{1}{\phi}\right)}\right)$.

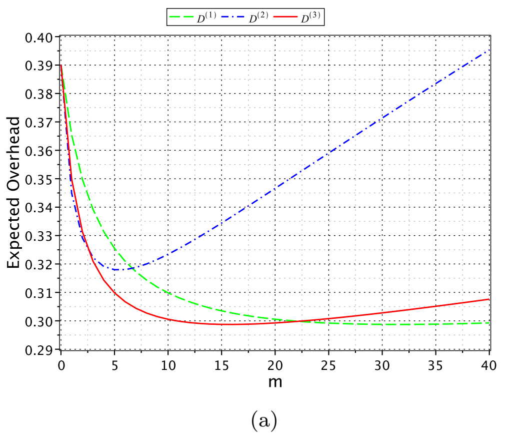
To validate the predicted performance, we have simulated the execution of the optimal patterns by injecting faults with the specified error rate. The last part of Table 1 shows the simulation results, obtained by averaging the values over 1000 runs for the respective patterns. We can see that the simulated overheads are within 1% of the predicted values for all patterns, which demonstrates the high accuracy of first-order approximation to the performance model. The results also confirm the low checkpointing frequency and high recovery rate of computing patterns that employ partial verifications. Intuitively, a higher recovery rate means that more errors are detected earlier in the execution. The results nicely corroborate the theoretical analysis, and demonstrate the benefit of using low-cost partial verifications for dealing with silent errors.

Since the results for realistic simulations with fault injections are very close to the model's predictions, we will focus on studying the model in the following experiments.

7.3. Scenario 2: Impact of Number of Partial Verifications

In the second scenario, we study the impact of the number of partial verifications on the execution overhead and pattern length of the optimal partial verification algorithm.

Figure 2 plots the overhead as well as the optimal pattern length as functions of the number of partial verifications m for each detector. The plots also show the overhead ($\approx 39\%$) and optimal pattern length (≈ 6156 seconds) of the baseline algorithm, represented by $m = 0$. We can see that the expected overhead is reduced for all three detectors by employing partial verifications. For each detector, the optimal overhead is attained for a particular value of m, corroborating the theoretical study. After this point, it starts rising again due to the fact that forcing too many verifications will eventually cause the error-free overhead to increase. The improvement in overhead over the



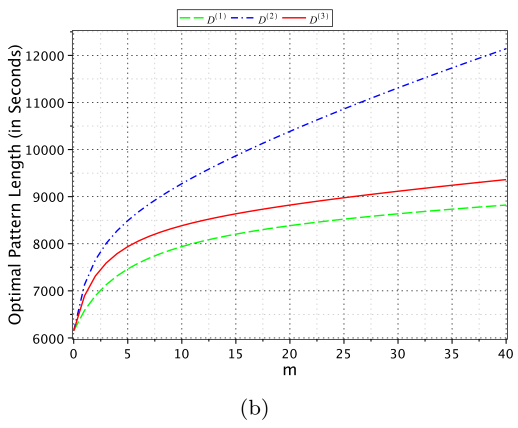


Figure 2: Expected overhead (a) and optimal length (b) of a pattern against the number of partial verifications when a single type of detector is used for three different detectors. The costs and recalls of the three detectors are $V^{(1)} = 3$ seconds, $V^{(2)} = 30$ seconds, $V^{(3)} = 6$ seconds, and $r^{(1)} = 0.5$, $r^{(2)} = 0.95$, $r^{(3)} = 0.8$. The costs of checkpointing and guaranteed verification are $C = V^* = 600$ seconds. The platform MTBF is $\mu = 31536$ seconds.

baseline algorithm is 9% for detectors $D^{(1)}$ and $D^{(3)}$ (optimal overhead for both is $\approx 30\%$), and 7% for detector $D^{(2)}$ (optimal overhead is $\approx 32\%$).

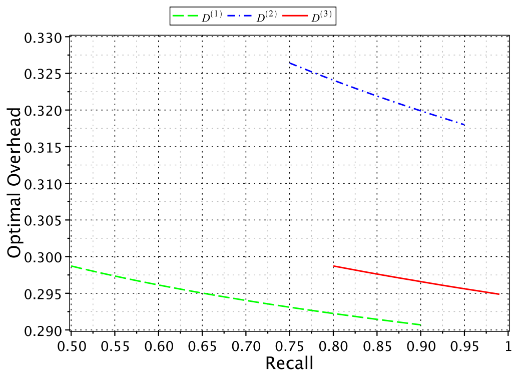
Also, the optimal pattern length increases as more partial verifications are employed inside the pattern. This is because the use of intermediate verifications allows silent errors to be detected earlier in the pattern and thus delays the checkpointing process. Interestingly, the optimal pattern lengths of all three detectors are around 8600 seconds when their respective optimal overheads are reached. This implies that an optimal pattern using partial verifications delays the taking of each checkpoint by ≈ 40 minutes, which corresponds to a saving of $\approx$4 checkpoints/day over the baseline algorithm. Concerning the performance of detectors, we can see that $D^{(1)}$ and $D^{(3)}$ are slightly better than $D^{(2)}$, due to their higher accuracy-to-cost ratios. However, for $m \leq 2$, $D^{(2)}$ is better due to its higher recall, while its performance degrades as more $D^{(2)}$ detectors are employed due to its high cost.

7.4. Scenario 3: Impact of Detector Recall

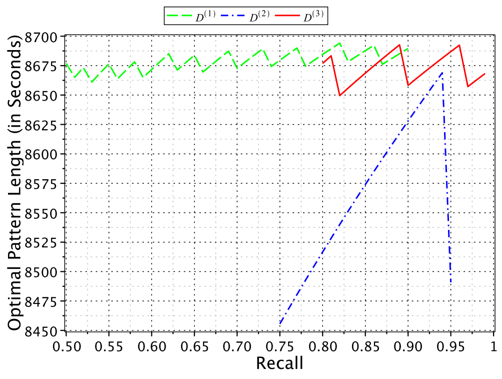
In the third scenario, we consider applications with various datasets that expose a change in the detection recall. Therefore, a range of recall values is possible for each detector rather than a single value.

According to [6, 5], the recall ranges of the three detectors are $r^{(1)} = [0.5, 0.9]$, $r^{(2)} = [0.75, 0.95]$, and $r^{(3)} = [0.8, 0.99]$, respectively. Given a dataset, we obtain a value of recall for each detector within the range. This is because different datasets might expose different levels of entropy and therefore the detectors might expose different prediction accuracies, hence different recalls. Note that although the recall might be different for different datasets, the work done, hence the detection cost, is the same. We rely upon four different metrics, namely, optimal overhead, optimal pattern length, optimal number of verifications, and accuracy-to-cost-ratio, to assess the impact of recall r on the optimal partial verification algorithm.

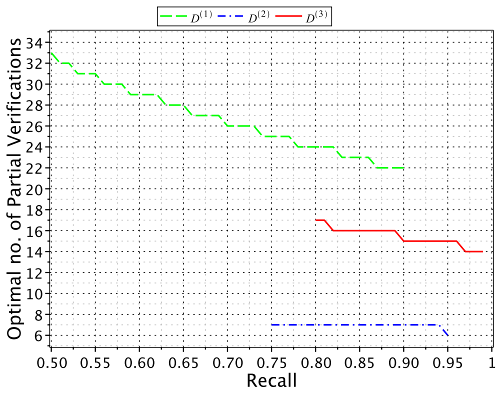
Figure 3 compares the performance of the three detectors through the four metrics when there is a change in the detection recall for each detector in its recall range. The plots in Figure 3(a) show variations in the optimal overheads with increasing recall values. As expected, the optimal overheads are reduced for all three detectors, since a higher recall value for the same cost (and same number) of verification reduces the fault-induced re-execution cost (f_{re}), while keeping the fault-free overhead (o_{ff}) constant, thus minimizes the product $o_{\text{ff}} f_{\text{re}}$ (see Section 5.1). This reduction in overhead can also be explained through the plots in Figure 3(d), which show an increase in the accuracy-to-cost ratio of each detector with higher recall values. The detectors $D^{(1)}$ and $D^{(3)}$ have the highest accuracy-to-cost ratio, and when used alone inside the pattern, produce the lowest



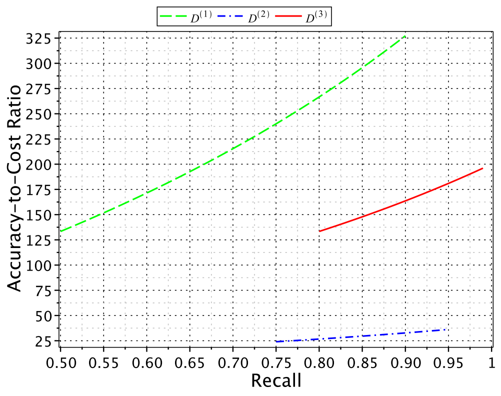
(a)



(b)



(c)



(d)

Figure 3: Optimal overhead (a), optimal pattern length (b), optimal number of partial verifications (c), and accuracy-to-cost ratio (d) for three different detectors as functions of recall in their respective recall ranges ($r^{(1)} = [0.5, 0.9]$, $r^{(2)} = [0.75, 0.95]$ and $r^{(3)} = [0.8, 0.99]$).

optimal overheads for their respective recall ranges. This substantiates the theoretical analysis of Lemma 1 in Section 6.1. The detector $D^{(2)}$, being an expensive verification, has a much lower ratio and thus incurs a higher optimal overhead.

Figure 3(b) shows oscillations in the curves representing the optimal pattern length for varying recall values. This can be understood by observing the plot in Figure 3(c), where the optimal number of partial verifications m^* for all three detectors follows a staircase function. For example, the optimal m^* for detector $D^{(1)}$ goes from 33 to 22 as the recall value increases in the range. This is due to the fact that verifications with higher recalls (or accuracy-to-cost ratios) allow us to achieve lower optimal overhead (as in Figure 3(a)) with fewer verifications. This step-wise reduction in the number of verifications leads to minor oscillations in the optimal pattern length. In particular, it is interesting to observe that in case of detector $D^{(2)}$, by fixing its recall at $r^{(2)} = 0.94$ and $r^{(2)} = 0.95$, the optimal overheads are 31.83% and 31.79% respectively, and the optimal pattern lengths are 8668 and 8490 seconds respectively. Thus, approximately 3 minutes of more execution per pattern can be done by compromising 0.04% of overhead. The reduction in the optimal pattern length for a higher recall value is due to a decrement in m^*. Note that both oscillations and staircase effects would disappear if m^* was allowed to take rational values.

7.5. Scenario 4: Performance of Greedy Algorithm

Finally, in the last scenario, we focus on the greedy algorithm presented in Section 6.3 and compare its performance with the optimal solution that uses more than one type of partial detector with different datasets, while keeping the same values for C, V^* and μ.

Table 2: Performance comparison of the greedy algorithm and the optimal solution. In all scenarios, $C = V^* = 600$ seconds, $V^{(1)} = 3$ seconds, $V^{(3)} = 6$ seconds.

	m	overhead H	diff. from opt.
Scenario 1: $r^{(1)} = 0.51$, $r^{(3)} = 0.82$, $\phi^{(1)} \approx 137$, $\phi^{(3)} \approx 139$			
Optimal solution	(1, 15)	29.828%	0%
Greedy with $D^{(3)}$	(0, 16)	29.829%	0.001%
Scenario 2: $r^{(1)} = 0.58$, $r^{(3)} = 0.9$, $\phi^{(1)} \approx 163$, $\phi^{(3)} \approx 164$			
Optimal solution	(1, 14)	29.659%	0%
Greedy with $D^{(3)}$	(0, 15)	29.661%	0.002%
Scenario 3: $r^{(1)} = 0.64$, $r^{(3)} = 0.97$, $\phi^{(1)} \approx 188$, $\phi^{(3)} \approx 188$			
Optimal solution	(1, 13)	29.523%	0%
Greedy with $D^{(1)}$	(27, 0)	29.524%	0.001%
Greedy with $D^{(3)}$	(0, 14)	29.525%	0.002%

As in the previous experiment, the recall of each detector is given a range of possible values, and its actual value depends on the dataset. As shown in Figure 3(d), even with the recall ranges, $D^{(2)}$ always has a lower accuracy-to-cost ratio compared to $D^{(1)}$ and $D^{(3)}$, which share similar ratios. Table 2 presents three scenarios that we have identified, where a combination of $D^{(1)}$ and $D^{(3)}$ constitutes the optimal pattern. In all these scenarios, the greedy algorithm, which uses only the detector with the highest accuracy-to-cost ratio, performs within 0.002% of the optimal solution. The results show that the greedy algorithm performs extremely well under these practical settings, even though the optimal pattern may employ both $D^{(1)}$ and $D^{(3)}$ in the solution.

8. Conclusion and Future Work

In this paper, we provided a comprehensive analysis of computing patterns that employ different types of partial verifications for detecting silent errors in HPC applications. We demonstrated that detectors with imperfect precision should not be used in such computing patterns. When considering detectors with imperfect recall, we showed that the optimization problem is NP-complete in general, and we proposed both a greedy algorithm and an FPTAS for choosing the number of detectors to be used, as well as their types and locations in the pattern. Extensive simulations based on realistic detector settings showed that the greedy algorithm works well in practice, and confirmed the usefulness of partial detectors to cope with silent errors in exascale systems.

In future work, we plan to investigate detectors with imperfect recall (and possibly imperfect precision) for an application consisting in a set of tasks with precedence constraints, where a detector can be employed only at the end of a task, hence reducing the flexibility in the computational scenario. Also, it would be interesting to combine the use of such detectors with a reasonable use of replication: one may prefer to replicate a small task rather than to use a costly detector for detecting silent data corruptions.

Acknowledgment

We would like to thank the reviewers for their valuable comments and suggestions, which greatly helped improve the quality of this paper. This research was funded in part by the European project SCoRPiO, by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program “Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR), by the PIA ELCI project, by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research Program, under Contract DE-AC02-06CH11357, by the INRIA-Illinois-ANL-BSC Joint Laboratory on Extreme Scale Computing, and by the Center for Exascale Simulation of Advanced Reactors at Argonne. Yves Robert is with Institut Universitaire de France.

References

[1] G. Aupy, A. Benoit, T. Hérault, Y. Robert, F. Vivien, and D. Zaidouni. On the combination of silent error detection and checkpointing. In *Proceedings of the 19th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 11–20, 2013.

[2] G. Aupy, Y. Robert, F. Vivien, and D. Zaidouni. Checkpointing algorithms and fault prediction. *J. Parallel and Distributed Computing*, 74(2):2048–2064, 2014.

[3] L. Bautista Gomez and F. Cappello. Detecting silent data corruption through data dynamic monitoring for scientific applications. In *Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '14, pages 381–382, New York, NY, USA, 2014. ACM.

[4] L. Bautista Gomez and F. Cappello. Detecting silent data corruption through data dynamic monitoring for scientific applications. *SIGPLAN Notices*, 49(8):381–382, 2014.

[5] L. Bautista Gomez and F. Cappello. Detecting and correcting data corruption in stencil applications through multivariate interpolation. In *Proceedings of the 1st International Workshop on Fault Tolerant Systems*, FTS'15. IEEE, 2015.

[6] L. Bautista Gomez and F. Cappello. Exploiting Spatial Smoothness in HPC Applications to Detect Silent Data Corruption. In *Proceedings of the 17th IEEE International Conference on High Performance Computing and Communications*, HPCC'15. IEEE, 2015.

[7] A. Benoit, A. Cavelan, Y. Robert, and H. Sun. Assessing general-purpose algorithms to cope with fail-stop and silent errors. In *Proceedings of the 5th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pages 215–236, 2014.

[8] A. Benoit, A. Cavelan, Y. Robert, and H. Sun. Optimal resilience patterns to cope with fail-stop and silent errors. Research report RR-8786, INRIA, 2015. Available at `graal.ens-lyon.fr/~yrobert/rr8786.pdf`.

[9] A. Benoit, Y. Robert, and S. K. Raina. Efficient checkpoint/verification patterns. *Int. J. of High Performance Computing Applications*, DOI: 10.1177/1094342015594531, Published online, July 2015. Available as ICL Research report RR-1403.

[10] A. R. Benson, S. Schmit, and R. Schreiber. Silent error detection in numerical time-stepping schemes. *Int. J. High Performance Computing Applications*, DOI: 10.1177/1094342014532297, 2014.

[11] E. Berrocal, L. Bautista-Gomez, S. Di, Z. Lan, and F. Cappello. Lightweight silent data corruption detection based on runtime data analysis for HPC applications. In *Proceedings of The ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, HPDC '15, 2015.

[12] G. Bosilca, R. Delmas, J. Dongarra, and J. Langou. Algorithm-based fault tolerance applied to high performance computing. *J. Parallel Distrib. Comput.*, 69(4):410–416, 2009.

[13] P. G. Bridges, K. B. Ferreira, M. A. Heroux, and M. Hoemmen. Fault-tolerant iterative methods via selective reliability. *ArXiv e-prints*, June 2012.

[14] G. Bronevetsky and B. de Supinski. Soft error vulnerability of iterative linear algebra methods. In *Proceedings of the International Conference on Supercomputing (ICS)*, pages 155–164, 2008.

[15] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir. Toward exascale resilience. *Int. Journal of High Performance Computing Applications*, 23(4):374–388, 2009.

[16] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir. Toward exascale resilience: 2014 update. *Supercomputing frontiers and innovations*, 1(1), 2014.

[17] A. Cavelan, S. K. Raina, Y. Robert, and H. Sun. Assessing the impact of partial verifications against silent data corruptions. In *Proceedings of the 44th Annual International Conference on Parallel Processing (ICPP)*, 2015.

[18] K. M. Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, 1985.

[19] Z. Chen. Online-ABFT: An online algorithm based fault tolerance scheme for soft error detection in iterative methods. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 167–176, 2013.

[20] E. Ciocca, I. Koren, Z. Koren, C. M. Krishna, and D. S. Katz. Application-level fault tolerance in the orbital thermal imaging spectrometer. In *Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04)*, pages 43–48, 2004.

[21] E. Ciocca, I. Koren, and C. M. Krishna. Determining acceptance tests for application-level fault detection. In *Proceedings of the 2nd ASPLOS EASY Workshop*, pages 47–53, 2002.

[22] R. Cohen and L. Katzir. The generalized maximum coverage problem. *Inf. Process. Lett.*, 108(1):15–22, 2008.

[23] J. T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Comp. Syst.*, 22(3):303–312, 2006.

[24] S. Di and F. Cappello. Adaptive impact-driven detection of silent data corruption for HPC applications. *IEEE Trans. Parallel Distributed Systems*, 2016.

[25] J. Dongarra, P. Beckman, P. Aerts, F. Cappello, T. Lippert, S. Matsuoka, P. Messina, T. Moore, R. Stevens, A. Trefethen, and M. Valero. The international exascale software project: a call to cooperative action by the global high-performance community. *IJHPCA*, 23(4):309–322, 2009.

[26] M. Dow. Explicit inverses of Toeplitz and associated matrices. *ANZIAM J.*, 44(E):E185–E215, Jan. 2003.

[27] J. Elliott, M. Hoemmen, and F. Mueller. Evaluating the impact of SDC on the GMRES iterative solver. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, IPDPS '14, pages 1193–1202, 2014.

[28] J. Elliott, K. Kharbas, D. Fiala, F. Mueller, K. Ferreira, and C. Engelmann. Combining partial redundancy and checkpointing for HPC. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 615–626, 2012.

[29] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Survey*, 34:375–408, 2002.

[30] D. Fiala, F. Mueller, C. Engelmann, R. Riesen, K. Ferreira, and R. Brightwell. Detection and correction of silent data corruption for large-scale high-performance computing. In *Proc. SC'12*, page 78, 2012.

[31] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[32] K.-H. Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Comput.*, 33(6):518–528, 1984.

[33] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[34] G. Lu, Z. Zheng, and A. A. Chien. When is multi-version checkpointing needed? In *Proc. 3rd Workshop on Fault-tolerance for HPC at extreme scale (FTXS)*, pages 49–56, 2013.

[35] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM J. Res. Dev.*, 6(2):200–209, 1962.

[36] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *Proc. of the ACM/IEEE SC Conf.*, pages 1–11, 2010.

[37] X. Ni, E. Meneses, N. Jain, and L. V. Kalé. ACR: Automatic checkpoint/restart for soft and hard error protection. In *Proc. SC'13*. ACM, 2013.

[38] T. O'Gorman. The effect of cosmic rays on the soft error rate of a DRAM at ground level. *IEEE Trans. Electron Devices*, 41(4):553–557, 1994.

[39] P. Sao and R. Vuduc. Self-stabilizing iterative solvers. In *Proceedings of the Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*, 2013.

[40] M. Shantharam, S. Srinivasmurthy, and P. Raghavan. Fault tolerant preconditioned conjugate gradient for sparse linear system solution. In *Proceedings of the ACM International Conference on Supercomputing (ICS)*, pages 69–78, 2012.

[41] J. W. Young. A first order approximation to the optimum checkpoint interval. *Comm. of the ACM*, 17(9):530–531, 1974.

[42] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, and B. Chin. IBM experiments in soft fails in computer electronics. *IBM J. Res. Dev.*, 40(1):3–18, 1996.

Appendix. Proof of Theorem 3

The goal is to minimize $f_{\text{re}} = \boldsymbol{\alpha}^T M \boldsymbol{\alpha}$ (Equation (8)) subject to the constraint $\sum_{k=1}^{n} \alpha_k = 1$, which we rewrite as $\mathbf{c}^T\boldsymbol{\alpha} = 1$ with $\mathbf{c} = [1, 1, \dots, 1]^T$. Hence, we have a quadratic minimization problem under a linear constraint. For convenience, let us replace M by $A = \frac{M+M^T}{2}$, which gives the same value for f_{re}, and we obtain the symmetric matrix A defined as $A_{ij} = \frac{1+g_{[i,j[}}{2}$ for $i \le j$. For instance, when $n = 4$, we have:

$$A = \frac{1}{2}\begin{bmatrix} 2 & 1+g_1 & 1+g_1g_2 & 1+g_1g_2g_3 \\ 1+g_1 & 2 & 1+g_2 & 1+g_2g_3 \\ 1+g_1g_2 & 1+g_2 & 2 & 1+g_3 \\ 1+g_1g_2g_3 & 1+g_2g_3 & 1+g_3 & 2 \end{bmatrix}.$$

When A is symmetric positive definite (SPD), which we will show later in the proof, there is a unique solution

$$f_{\text{re}}^{\text{opt}} = \frac{1}{\mathbf{c}^T A^{-1}\mathbf{c}}, \tag{16}$$

obtained for

$$\boldsymbol{\alpha}^{\text{opt}} = \frac{A^{-1}\mathbf{c}}{\mathbf{c}^T A^{-1}\mathbf{c}}. \tag{17}$$

This result is shown as follows. Let a *valid* vector $\boldsymbol{\alpha}$ be a vector such that $\mathbf{c}^T\boldsymbol{\alpha} = 1$. We have $\mathbf{c}^T\boldsymbol{\alpha}^{\text{opt}} = f_{\text{re}}^{\text{opt}}(\mathbf{c}^T A^{-1}\mathbf{c}) = 1$, hence $\boldsymbol{\alpha}^{\text{opt}}$ is indeed a valid vector. Then, because A is SPD, we have $X = (\boldsymbol{\alpha} - \boldsymbol{\alpha}^{\text{opt}})^T A(\boldsymbol{\alpha} - \boldsymbol{\alpha}^{\text{opt}}) \ge 0$ for any valid vector $\boldsymbol{\alpha}$, and $X = 0$ if and only if $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{\text{opt}}$. Developing X, we get

$$X = \boldsymbol{\alpha}^T A\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T A\boldsymbol{\alpha}^{\text{opt}} + (\boldsymbol{\alpha}^{\text{opt}})^T A\boldsymbol{\alpha}^{\text{opt}}.$$

We have $\boldsymbol{\alpha}^T A\boldsymbol{\alpha}^{\text{opt}} = f_{\text{re}}^{\text{opt}}\boldsymbol{\alpha}^T\mathbf{c} = f_{\text{re}}^{\text{opt}}$ because $\mathbf{c}^T\boldsymbol{\alpha} = 1$. Similarly, we get $(\boldsymbol{\alpha}^{\text{opt}})^T A\boldsymbol{\alpha}^{\text{opt}} = f_{\text{re}}^{\text{opt}}$. Hence, we derive that $X = \boldsymbol{\alpha}^T A\boldsymbol{\alpha} - f_{\text{re}}^{\text{opt}} \geq 0$, with equality if and only if $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{\text{opt}}$. Hence the optimal value of f_{re} is achieved for $\boldsymbol{\alpha}^{\text{opt}}$, and is equal to $f_{\text{re}}^{\text{opt}}$.

In the following, we prove that A is symmetric positive definite (SPD), and that $\boldsymbol{\alpha}^{\text{opt}} = \boldsymbol{\alpha}^*$ and $f_{\text{re}}^{\text{opt}} = f_{\text{re}}^*$. To avoid ambiguity, we use superscripts like $A^{(n)}$ whenever needed to identify the problem size n (the number of work segments).

From the definition of matrix A, we can rewrite $A^{(n)}$ as:

$$A^{(n)} = \frac{1}{2}\left(J^{(n)} + B^{(n)}\right),$$

where $J^{(n)}$ is the $n \times n$ matrix whose entries are all 1, and $B^{(n)}$ is the $n \times n$ matrix defined by $B_{ij}^{(n)} = g_{[i,j[}$ for $i \leq j$.

We start by proving two properties of $\boldsymbol{\alpha}^*$.

Lemma 2. *$\boldsymbol{\alpha}^*$ is a valid vector, i.e., $\sum_{k=1}^{n} \alpha_k^* = 1$.*

Proof. The proof is by induction on n. First, for $n = 1$ we do have $\sum_{k=1}^{1} \alpha_k^{*(1)} = 1$. For $n = 2$, we have $\sum_{k=1}^{2} \alpha_k^{*(2)} = \frac{1+g_1}{2}\left(\frac{1}{1+g_1} + \frac{1}{1+g_1}\right) = 1$, which is also correct. Assume that this result holds up to $n-1$. We can express $\boldsymbol{\alpha}^{*(n)}$ as:

$$\boldsymbol{\alpha}^{*(n)} = \frac{U_{n-1}}{U_n}\begin{bmatrix} \alpha_1^{*(n-1)} \\ \alpha_2^{*(n-1)} \\ \vdots \\ \alpha_{n-2}^{*(n-1)} \\ \alpha_{n-1}^{*(n-1)} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -g_{n-1}\alpha_n^{*(n)} \\ \alpha_n^{*(n)} \end{bmatrix}. \tag{18}$$

Therefore, we have:

$$\begin{aligned}
\sum_{k=1}^{n} \alpha_k^{*(n)} &= \sum_{k=1}^{n-2} \alpha_k^{*(n)} + \alpha_{n-1}^{*(n)} + \alpha_n^{*(n)} \\
&= \frac{U_{n-1}}{U_n}\sum_{k=1}^{n-2} \alpha_k^{*(n-1)} + \frac{U_{n-1}}{U_n}\alpha_{n-1}^{*(n-1)} - g_{n-1}\alpha_n^{*(n)} + \alpha_n^{*(n)} \\
&= \frac{U_{n-1}}{U_n}\sum_{k=1}^{n-1} \alpha_k^{*(n-1)} + \alpha_n^{*(n)}(1 - g_{n-1}) \\
&= \frac{U_{n-1}}{U_n}\sum_{k=1}^{n-1} \alpha_k^{*(n-1)} + \frac{1}{U_n} \cdot \frac{1 - g_{n-1}}{1 + g_{n-1}}.
\end{aligned}$$

Now, using the inductive hypothesis that $\sum_{k=1}^{n-1} \alpha_k^{*(n-1)} = 1$, we get:

$$\begin{aligned}
\sum_{k=1}^{n} \alpha_k^{*(n)} &= \frac{1}{U_n}\left(U_{n-1} + \frac{1 - g_{n-1}}{1 + g_{n-1}}\right) \\
&= \frac{1}{U_n} \cdot U_n \\
&= 1,
\end{aligned}$$

which concludes the proof. □

Lemma 3. *$A\boldsymbol{\alpha}^* = f_{re}^*\mathbf{c}$.*

Proof. We have $A = \frac{1}{2}(J + B)$ and from Lemma 2 $J\boldsymbol{\alpha}^* = \left(\sum_{k=1}^{n} \alpha_k^*\right)\mathbf{c} = \mathbf{c}$. The result will follow if we show

$$B\boldsymbol{\alpha}^* = \frac{\mathbf{c}}{U_n}, \tag{19}$$

for all $n \geq 1$. Equivalently, letting $\boldsymbol{\gamma} = U_n\boldsymbol{\alpha}^*$, we prove by induction on n that $B^{(n)}\boldsymbol{\gamma}^{(n)} = \mathbf{c}^{(n)}$. First, for $n = 1$ we have $B^{(1)}\boldsymbol{\gamma}^{(1)} = 1$, and for $n = 2$ we get:

$$B^{(2)}\boldsymbol{\gamma}^{(2)} = \begin{bmatrix} 1 & g_1 \\ g_1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{1+g_1} \\ \frac{1}{1+g_1} \end{bmatrix} = \begin{bmatrix} \frac{1}{1+g_1} + \frac{g_1}{1+g_1} \\ \frac{g_1}{1+g_1} + \frac{1}{1+g_1} \end{bmatrix} = \mathbf{c}^{(2)}.$$

Now, suppose the result holds up to $n - 1$. We can write:

$$\begin{aligned} B^{(n)}\boldsymbol{\gamma}^{(n)} &= \begin{bmatrix} B^{(n-1)} & \mathbf{x}^{(n-1)} \\ \left(\mathbf{x}^{(n-1)}\right)^T & 1 \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\gamma}}^{(n-1)} \\ \gamma_n^{(n)} \end{bmatrix} \\ &= \begin{bmatrix} B^{(n-1)}\bar{\boldsymbol{\gamma}}^{(n-1)} + \mathbf{x}^{(n-1)}\gamma_n^{(n)} \\ \left(\mathbf{x}^{(n-1)}\right)^T \bar{\boldsymbol{\gamma}}^{(n-1)} + \gamma_n^{(n)} \end{bmatrix}, \end{aligned} \tag{20}$$

where $\bar{\boldsymbol{\gamma}}^{(n-1)}$ is the $(n-1)\times 1$ truncated vector containing the first $n-1$ elements of $\boldsymbol{\gamma}^{(n)}$ (for a problem of size n), and $\mathbf{x}^{(n-1)}$ is an $(n-1)\times 1$ vector defined as $\mathbf{x}^{(n-1)} = \begin{bmatrix} g_{[1,n-1[} & g_{[2,n-1[} & \cdots & g_{n-1} \end{bmatrix}^T$. For instance, for $n = 4$ we have $\mathbf{x}^{(3)} = \begin{bmatrix} g_1g_2g_3 & g_2g_3 & g_3 \end{bmatrix}^T$. Then the goal is to show $B^{(n-1)}\bar{\boldsymbol{\gamma}}^{(n-1)} + \mathbf{x}^{(n-1)}\gamma_n^{(n)} = \mathbf{c}^{(n-1)}$ and $\left(\mathbf{x}^{(n-1)}\right)^T \bar{\boldsymbol{\gamma}}^{(n-1)} + \gamma_n^{(n)} = 1$. From Equation (18), we can derive:

$$\begin{aligned} &B^{(n-1)}\bar{\boldsymbol{\gamma}}^{(n-1)} \\ &= B^{(n-1)}\left(\boldsymbol{\gamma}^{(n-1)} + \begin{bmatrix} \mathbf{0}^{(n-2)} \\ -g_{n-1}\gamma_n^{(n)} \end{bmatrix}\right) \\ &= B^{(n-1)}\boldsymbol{\gamma}^{(n-1)} + \begin{bmatrix} B^{(n-2)} & \mathbf{x}^{(n-2)} \\ \left(\mathbf{x}^{(n-2)}\right)^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{0}^{(n-2)} \\ -g_{n-1}\gamma_n^{(n)} \end{bmatrix} \\ &= B^{(n-1)}\boldsymbol{\gamma}^{(n-1)} + \begin{bmatrix} -\mathbf{x}^{(n-2)}g_{n-1}\gamma_n^{(n)} \\ -g_{n-1}\gamma_n^{(n)} \end{bmatrix} \\ &= \mathbf{c}^{(n-1)} - \mathbf{x}^{(n-1)}\gamma_n^{(n)}. \end{aligned}$$

The last line applies the inductive hypothesis $B^{(n-1)}\boldsymbol{\gamma}^{(n-1)} = \mathbf{c}^{(n-1)}$ as well as the property that $\begin{bmatrix} \mathbf{x}^{(n-2)} \\ 1 \end{bmatrix} g_{n-1} = \mathbf{x}^{(n-1)}$. Putting this result back into Equation (20), we derive that

$$\begin{aligned} &B^{(n-1)}\bar{\boldsymbol{\gamma}}^{(n-1)} + \mathbf{x}^{(n-1)}\gamma_n^{(n)} \\ &\quad = \mathbf{c}^{(n-1)} - \mathbf{x}^{(n-1)}\gamma_n^{(n)} + \mathbf{x}^{(n-1)}\gamma_n^{(n)} = \mathbf{c}^{(n-1)}. \end{aligned}$$

Using the property $\left(\mathbf{x}^{(n-1)}\right)^T = \begin{bmatrix} \left(\mathbf{x}^{(n-2)}\right)^T & 1 \end{bmatrix} g_{n-1}$, we can write:

$$\begin{aligned} &\left(\mathbf{x}^{(n-1)}\right)^T \bar{\boldsymbol{\gamma}}^{(n-1)} + \gamma_n^{(n)} \\ &= \begin{bmatrix} \left(\mathbf{x}^{(n-2)}\right)^T & 1 \end{bmatrix} g_{n-1} \left(\boldsymbol{\gamma}^{(n-1)} + \begin{bmatrix} \mathbf{0}^{(n-2)} \\ -g_{n-1}\gamma_n^{(n)} \end{bmatrix}\right) + \gamma_n^{(n)} \\ &= \begin{bmatrix} \left(\mathbf{x}^{(n-2)}\right)^T & 1 \end{bmatrix} \boldsymbol{\gamma}^{(n-1)} g_{n-1} - g_{n-1}^2\gamma_n^{(n)} + \gamma_n^{(n)}. \end{aligned}$$

Notice that $\left[\left(\mathbf{x}^{(n-2)}\right)^T \quad 1\right] \boldsymbol{\gamma}^{(n-1)}$ is actually the last row of the product $B^{(n-1)}\boldsymbol{\gamma}^{(n-1)}$, which, by induction, is 1. Therefore we get:

$$\begin{aligned}
&\left(\mathbf{x}^{(n-1)}\right)^T \bar{\boldsymbol{\gamma}}^{(n-1)} + \gamma_n^{(n)} \\
&\quad = g_{n-1} + \gamma_n^{(n)}(1 - g_{n-1}^2) \\
&\quad = g_{n-1} + \frac{(1+g_{n-1})(1-g_{n-1})}{1+g_{n-1}} \\
&\quad = g_{n-1} + 1 - g_{n-1} \\
&\quad = 1.
\end{aligned}$$

This concludes the proof. □

We now prove that A is SPD. This requires several intermediate steps.

Lemma 4. *B is nonsingular and $\boldsymbol{\alpha}^* = \frac{1}{U_n} B^{-1}\mathbf{c}$.*

Proof. To prove that B is nonsingular, we prove by induction on n that $B^{(n)}\mathbf{y}^{(n)} = \mathbf{0}^{(n)}$ has only one solution $\mathbf{y}^{(n)} = \mathbf{0}^{(n)}$. First, for $n = 1$ we have $y_1^{(1)} = 0$, which is correct. Then, for $n = 2$ we have the following equation:

$$\begin{bmatrix} 1 & g_1 \\ g_1 & 1 \end{bmatrix} \begin{bmatrix} y_1^{(2)} \\ y_2^{(2)} \end{bmatrix} = \mathbf{0}^{(2)},$$

from which we derive $y_1^{(2)}(1 - g_1^2) = 0$ and $y_2^{(2)}(1 - g_1^2) = 0$, hence $\mathbf{y}^{(2)} = \mathbf{0}^{(2)}$, which is also correct. Now, assume that the result holds up to $n-1$. We want to solve the general equation:

$$\begin{bmatrix} B^{(n-1)} & \mathbf{x}^{(n-1)} \\ \left(\mathbf{x}^{(n-1)}\right)^T & 1 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{y}}^{(n-1)} \\ y_n^{(n)} \end{bmatrix} = \mathbf{0}^{(n)}, \tag{21}$$

which is equivalent to:

$$\begin{bmatrix} B^{(n-2)} & \mathbf{x}^{(n-2)} & \mathbf{x}^{(n-2)} g_{n-1} \\ \left(\mathbf{x}^{(n-2)}\right)^T & 1 & g_{n-1} \\ \left(\mathbf{x}^{(n-2)}\right)^T g_{n-1} & g_{n-1} & 1 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{y}}^{(n-2)} \\ y_{n-1}^{(n)} \\ y_n^{(n)} \end{bmatrix} = \mathbf{0}^{(n)}, \tag{22}$$

where $\bar{\mathbf{y}}^{(n-1)}$ and $\bar{\mathbf{y}}^{(n-2)}$ are the truncated vectors containing respectively the first $n-1$ and $n-2$ elements of $\mathbf{y}^{(n)}$ (for a problem of size n). First, let us expand Equation (22) and consider only the last two equations of the system:

$$\begin{aligned}
\left(\mathbf{x}^{(n-2)}\right)^T \bar{\mathbf{y}}^{(n-2)} + y_{n-1}^{(n)} + g_{n-1} y_n^{(n)} &= 0 \\
g_{n-1}\left(\mathbf{x}^{(n-2)}\right)^T \bar{\mathbf{y}}^{(n-2)} + g_{n-1} y_{n-1}^{(n)} + y_n^{(n)} &= 0.
\end{aligned}$$

We can derive that $y_n^{(n)}(1 - g_{n-1}^2) = 0$, hence $y_n^{(n)} = 0$. Then, plugging $y_n^{(n)} = 0$ back into Equation (21), we derive that:

$$B^{(n-1)}\bar{\mathbf{y}}^{(n-1)} = \mathbf{0}^{(n-1)}.$$

Using the induction hypothesis for $B^{(n-1)}\bar{\mathbf{y}}^{(n-1)} = \mathbf{0}^{(n-1)}$, we have $\bar{\mathbf{y}}^{(n-1)} = \mathbf{0}^{(n-1)}$ and thus $\mathbf{y}^{(n)} = \mathbf{0}^{(n)}$, which implies that $B^{(n)}$ is nonsingular. Hence, from Equation (19), we can get:

$$\boldsymbol{\alpha}^* = \frac{1}{U_n} B^{-1}\mathbf{c},$$

which concludes the proof. □

Lemma 5. *A is nonsingular.*

Proof. To prove that A is nonsingular, we solve $A\mathbf{y} = \mathbf{0}$ and show that $\mathbf{y} = \mathbf{0}$. First, we can write:

$$J\mathbf{y} + B\mathbf{y} = 0,$$
$$B\mathbf{y} = -J\mathbf{y} = -\left(\sum_{i=1}^{n} y_i\right)\mathbf{c}.$$

From Lemma 4, we know that B is nonsingular and $B^{-1}\mathbf{c} = U_n\boldsymbol{\alpha}^*$. Therefore, we get:

$$\mathbf{y} = -U_n\left(\sum_{i=1}^{n} y_i\right)\boldsymbol{\alpha}^*. \tag{23}$$

Summing the components of both sides of Equation (23), we obtain:

$$\left(\sum_{i=1}^{n} y_i\right) = -U_n\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} \alpha_i^*\right).$$

Since $\sum_{i=1}^{n} \alpha_i^* = 1$ from Lemma 2, we have:

$$\left(\sum_{i=1}^{n} y_i\right)(1 + U_n) = 0,$$
$$\sum_{i=1}^{n} y_i = 0,$$

which implies $\mathbf{y} = \mathbf{0}$ from Equation (23), and this concludes the proof that A is nonsingular. □

Lemma 6. *The last column of B^{-1} is given by:*

$$\mathbf{z} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ -g_{n-1}z_n \\ z_n \end{bmatrix}, \textit{ with } z_n = \frac{1}{1 - {g_{n-1}}^2}.$$

Proof. Because we do not need the whole inverse of B, we solve $B\mathbf{z} = \mathbf{d}$, where $\mathbf{d} = \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix}^T$, hence $\mathbf{z}$ will be the last column of B^{-1}. We can write:

$$\begin{bmatrix} B^{(n-2)} & \mathbf{x}^{(n-2)} & \mathbf{x}^{(n-2)}g_{n-1} \\ \left(\mathbf{x}^{(n-2)}\right)^T & 1 & g_{n-1} \\ \left(\mathbf{x}^{(n-2)}\right)^T g_{n-1} & g_{n-1} & 1 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}}^{(n-2)} \\ z_{n-1}^{(n)} \\ z_n^{(n)} \end{bmatrix} = \mathbf{d}^{(n)},$$

where $\bar{\mathbf{z}}^{(n-2)}$ is the truncated vector containing the first $n-2$ elements of $\mathbf{z}^{(n)}$. Expanding the product, we get the following system of equations:

$$B^{(n-2)}\bar{\mathbf{z}}^{(n-2)} + \mathbf{x}^{(n-2)}z_{n-1}^{(n)} + \mathbf{x}^{(n-2)}g_{n-1}z_n^{(n)} = \mathbf{0}^{(n-2)},$$
$$\left(\mathbf{x}^{(n-2)}\right)^T \bar{\mathbf{z}}^{(n-2)} + z_{n-1}^{(n)} + g_{n-1}z_n^{(n)} = 0,$$
$$\left(\mathbf{x}^{(n-2)}\right)^T g_{n-1}\bar{\mathbf{z}}^{(n-2)} + g_{n-1}z_{n-1}^{(n)} + z_n^{(n)} = 1.$$

Since $B^{(n)}$ is nonsingular, there is a unique solution. We can check that $\bar{\mathbf{z}}^{(n-2)} = \mathbf{0}^{(n-2)}$, $z_{n-1}^{(n)} = \frac{-g_{n-1}}{1-{g_{n-1}}^2}$ and $z_n^{(n)} = \frac{1}{1-{g_{n-1}}^2}$ is indeed a solution, which concludes the proof. □

Remark. The matrix B is an extension of the famous KMS symmetric matrix K [26], where $K_{ij} = g^{j-i}$ for $i \leq j$ (recall that $B_{ij} = g_{[i,j[}$). The inverse of B turns out to be tridiagonal, just as that of K, and we get:

$$B^{-1}_{ij} = \begin{cases} -\frac{g_j}{1-g_j^2} & \text{if } i = j+1 \\ -\frac{g_i}{1-g_i^2} & \text{if } i = j-1 \\ \frac{1-g_{i-1}^2 g_i^2}{(1-g_{i-1}^2)(1-g_i^2)} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

For instance, when $n = 4$ we have

$$B^{-1} = \begin{bmatrix} \frac{1}{1-g_1^2} & -\frac{g_1}{1-g_1^2} & 0 & 0 \\ -\frac{g_1}{1-g_1^2} & \frac{1-g_1^2 g_2^2}{(1-g_1^2)(1-g_2^2)} & -\frac{g_2}{1-g_2^2} & 0 \\ 0 & -\frac{g_2}{1-g_2^2} & \frac{1-g_2^2 g_3^2}{(1-g_2^2)(1-g_3^2)} & -\frac{g_3}{1-g_3^2} \\ 0 & 0 & -\frac{g_3}{1-g_3^2} & \frac{1}{1-g_3^2} \end{bmatrix}.$$

The proof of this result is very similar to the proof of Lemma 6.

Lemma 7. $A^{-1}_{nn} = 2\frac{U_n(1+g_{n-1})+2g_{n-1}}{(U_n+1)(1-g_{n-1})(1+g_{n-1})^2}$.

Proof. As in the proof of Lemma 6, we compute the last column of A^{-1}, which we call $\boldsymbol{\beta}$, by solving $A\boldsymbol{\beta} = \mathbf{d}$. Because we already solved $B\mathbf{z} = \mathbf{d}$, we have:

$$\begin{aligned} A\boldsymbol{\beta} &= B\mathbf{z} = \mathbf{d}, \\ \frac{1}{2}(J+B)\boldsymbol{\beta} &= B\mathbf{z}, \\ J\boldsymbol{\beta} &= B(2\mathbf{z} - \boldsymbol{\beta}). \end{aligned}$$

Remember that J is the matrix whose entries are all 1. Hence, we have $J\boldsymbol{\beta} = \left(\sum_{i=1}^n \beta_i\right)\mathbf{c}$. Also, from Lemma 4, we have $B\boldsymbol{\alpha}^* = \frac{\mathbf{c}}{U_n}$. Therefore, we can derive:

$$2\mathbf{z} - \boldsymbol{\beta} = \left(\sum_{i=1}^n \beta_i\right) U_n \boldsymbol{\alpha}^*. \tag{24}$$

Summing the components of both sides of Equation (24), we get $2\sum_{i=1}^n z_i - \sum_{i=1}^n \beta_i = \left(\sum_{i=1}^n \beta_i\right) U_n \left(\sum_{i=1}^n \alpha_i^*\right)$. Since $\sum_{i=1}^n \alpha_i^* = 1$ from Lemma 2, we get

$$\sum_{i=1}^n \beta_i = \frac{2}{U_n+1}\sum_{i=1}^n z_i.$$

From Lemma 6, we can easily compute $\sum_{i=1}^n z_i = -g_{n-1}z_n + z_n = \frac{1}{1+g_{n-1}}$. Hence, we have

$$\sum_{i=1}^n \beta_i = \frac{2}{(U_n+1)(1+g_{n-1})}.$$

Finally, from Equation (24), we derive that

$$\begin{aligned} \beta_n &= 2z_n - \left(\sum_{i=1}^n \beta_i\right) U_n \alpha_n^* \\ &= \frac{2}{1-g_{n-1}{}^2} - \frac{2}{(U_n+1)(1+g_{n-1})^2} \\ &= 2\frac{U_n(1+g_{n-1})+2g_{n-1}}{(U_n+1)(1-g_{n-1})(1+g_{n-1})^2}, \end{aligned}$$

which concludes the proof. □

Lemma 8. *A is symmetric positive definite (SPD).*

Proof. Note that by construction, A, J and B are all symmetric matrices. To show that A is positive definite, we show that all its principal minors are strictly positive. Recall that the principal minor of order k of $A^{(n)}$ is the determinant of the submatrix of size k that consists of the first k rows and columns of $A^{(n)}$. But this submatrix is exactly $A^{(k)}$, the matrix for the problem of size k, so the result will follow if we show that $\det\left(A^{(n)}\right) > 0$ for all $n \geq 1$. We prove by induction on n that

$$\det\left(A^{(n)}\right) = \frac{U_n + 1}{2^n} \prod_{k=1}^{n-1} \left(1 - g_k^2\right) > 0. \tag{25}$$

For $n = 1$, Equation (25) gives $\det\left(A^{(1)}\right) = 1$, which is correct. Suppose the result holds up to $n-1$. Since $A^{(n)}$ is nonsingular, using the co-factor method, we get that

$$\left(A^{(n)}\right)^{-1}_{nn} = \frac{\det\left(A^{(n-1)}\right)}{\det\left(A^{(n)}\right)}.$$

Therefore, using the definition of U_n and the induction hypothesis for $\det\left(A^{(n-1)}\right)$, we can get:

$$\begin{aligned}
\det\left(A^{(n)}\right) &= \frac{\det\left(A^{(n-1)}\right)}{\left(A^{(n)}\right)^{-1}_{nn}} \\
&= \frac{1}{\left(A^{(n)}\right)^{-1}_{nn}} \cdot \frac{U_{n-1} + 1}{2^{n-1}} \prod_{k=1}^{n-2} (1 - g_k^2) \\
&= \frac{1}{\left(A^{(n)}\right)^{-1}_{nn}} \cdot \frac{1}{2^{n-1}} \left(U_n - \frac{1 - g_{n-1}}{1 + g_{n-1}} + 1\right) \prod_{k=1}^{n-2} (1 - g_k^2) \\
&= \frac{1}{\left(A^{(n)}\right)^{-1}_{nn}} \cdot \frac{1}{2^{n-1}} \cdot \frac{U_n(1 + g_{n-1}) + 2g_{n-1}}{1 + g_{n-1}} \prod_{k=1}^{n-2} (1 - g_k^2). \qquad (26)
\end{aligned}$$

Now, plugging $\left(A^{(n)}\right)^{-1}_{nn}$ from Lemma 7 into Equation (26), we get:

$$\begin{aligned}
\det\left(A^{(n)}\right) &= \frac{1}{2^n} \cdot \frac{(U_n + 1)(1 - g_{n-1})(1 + g_{n-1})^2}{U_n(1 + g_{n-1}) + 2g_{n-1}} \cdot \frac{U_n(1 + g_{n-1}) + 2g_{n-1}}{1 + g_{n-1}} \prod_{k=1}^{n-2} (1 - g_k^2) \\
&= \frac{U_n + 1}{2^n} (1 - g_{n-1}^2) \prod_{k=1}^{n-2} (1 - g_k^2) \\
&= \frac{U_n + 1}{2^n} \prod_{k=1}^{n-1} (1 - g_k^2),
\end{aligned}$$

which shows that Equation (25) holds for $\det\left(A^{(n)}\right)$ and completes the proof that $A^{(n)}$ is SPD. $\square$

We are almost done! There remains to show that $\boldsymbol{\alpha}^{\text{opt}} = \boldsymbol{\alpha}^*$ and $f_{\text{re}}^{\text{opt}} = f_{\text{re}}^*$. But Lemma 3 shows that $A\boldsymbol{\alpha}^* = f_{\text{re}}^* \mathbf{c}$, hence $\boldsymbol{\alpha}^* = f_{\text{re}}^* A^{-1}\mathbf{c}$ and $1 = \mathbf{c}^T \boldsymbol{\alpha}^* = f_{\text{re}}^* (\mathbf{c}^T A^{-1} \mathbf{c})$, which leads to $f_{\text{re}}^{\text{opt}} = f_{\text{re}}^*$, and finally $\boldsymbol{\alpha}^{\text{opt}} = \boldsymbol{\alpha}^*$. This concludes the proof of Theorem 3.