Automated Rare Event Simulation for Fault Tree Analysis via Minimal Cut Sets

Carlos E. Budde[1(✉)] and Mariëlle Stoelinga[1,2]

[1] Formal Methods and Tools, University of Twente, Enschede, The Netherlands
{c.e.budde,m.i.a.stoelinga}@utwente.nl
[2] Department of Software Science, Radboud University, Nijmegen, The Netherlands

Abstract. Monte Carlo simulation is a common technique to estimate dependability metrics for fault trees. A bottleneck in this technique is the number of samples needed, especially when the interesting events are rare and occur with low probability. Rare Event Simulation (RES) reduces the number of samples when analysing rare events. Importance splitting is a RES method that spawns more simulation runs from promising system states. How promising a state is, is indicated by an importance function, which concentrates the information that makes this method efficient. Importance functions are given by domain and RES experts. This hinders re-utilisation and involves decisions entailing potential human error. Focusing in (general) fault trees, in this paper we automatically derive importance functions based on the tree structure. For this we exploit a common fault tree concept, namely cut sets: the more elements from a cut set have failed, the higher the importance. We show that the cut-set-derived importance function is an easy-to-implement and simple concept, that can nonetheless compete against another (more involved) automatic importance function for RES.

Keywords: Minimal cut sets · Rare event simulation · Dynamic fault trees · Importance splitting · Fault tree analysis

1 Introduction

Classical Monte Carlo simulation (CMC) is a common technique to evaluate stochastic models. Its applications range from systems biology, climate models, and social interaction, to reliability analysis, performance evaluation, network security, and many more.

By taking a large number of random samples, CMC estimates the metric of interest, such as the average package loss in a network. While this technique is very flexible in the stochastic models it can handle, as well as the metrics it can analyse, it suffers from a major drawback: to get accurate estimates, a large

This work was partially funded by EU project 102112 (*SUCCESS*), and NS, ProRail, and NWO project 15474 (*SEQUOIA*).

H. Hermanns (Ed.): MMB 2020, LNCS 12040, pp. 259–277, 2020.
https://doi.org/10.1007/978-3-030-43024-5_16

number of samples is needed. This problem is exacerbated in case of rare events, i.e. events with low probability of occurrence.

Rare event simulation (RES [24]) is a scientific field dealing with simulation techniques that efficiently handle rare events. Various methods exist, including importance sampling [10,21], importance splitting [1,15], sequential Monte Carlo [6], etc. These methods typically tweak the probabilities in the systems, or the way samples are taken, to make the rare event less rare. The metric of interest is then also adjusted, to account for the changes in the simulations. However, RES also comes with a drawback: all these techniques depend on expert knowledge.

This paper presents a theory to deploy automatic RES for an important class of models, namely *fault trees* (FTs). Fault trees are a prominent model in reliability engineering, and are widely deployed in industry, by companies like Siemens, Honeywell, NASA and many others.

An FT is a graphical model that shows why a system fails, i.e. which failure modes and mechanisms exist that can cause a top-level system failure. Thus, the leaves of the tree model basic failures; while gates represent how basic failures propagate through the system. Static gates represent Boolean combinations of failures, such as AND and OR, while dynamic gates model dependability patters, such as spare components management and functional dependencies.

Typical metrics for FTs include system *reliability* (the probability that the system does not fail during its mission time) and *availability* (its average uptime).

Numerical analysis of fault trees is achievable for large models, but it is not feasible for fault trees that are complex, and model important features such as maintenance, interdependent interactions, or non-Markovian probability distributions. Such FT models are often analyzed using Monte Carlo simulation. Rare events are an issue here [29], since fault trees are typically used to model safety-critical systems (power plants, rockets) whose failure probability is low. Our technique for rare event simulation is based on importance splitting. The key idea is to generate more samples from a path that looks promising. More precisely, importance splitting relies on an *importance function* that assigns a higher *importance* (viz. weight) to more promising states in a trace. The main contribution of this paper is to assign to each fault tree an importance function that can be automatically derived from its structure, in a similar way to [3]. The goals is also to deploy functions with low computation overhead and low variability, so that the RES algorithms implemented from them hare highly-performant.

To achieve this, our importance functions is based on (minimal) cut sets. Cut sets are sets of basic events such that, if all elements in the cuts set fail, the tree fails—we note that cut sets are defined for static fault trees; we conservatively extend these to dynamic fault trees. Thus, the more elements in a cut set that have failed, the higher its importance. Since a fault tree usually has multiple cut sets, we take the maximum importance over all cut sets.

In fact, we propose several variants of this idea, where we also normalize the cut sets by their maximum weights, or prune them based on their cardinality or failure probability. Our experimental evaluation shows that our functions, tested on the standard HECS benchmark for FTA for a fixed simulation time budget,

can produce estimates as accurate as more involved approaches that consider the whole structure of the tree [3]. Moreover, the RES algorithms deployed by our functions show the highest stability in our tests. These results are a first step; elaborate experimental evaluation are an important topic for future research.

Outline. After reviewing related work, in Sect. 2 we present the basic theoretical concepts needed to understand our contribution. In Sect. 3 we introduce an heuristic importance function for RES, that uses the minimal cut sets of the FTs. Section 4 compares our approach—empirically—to other ways of analysing DFTs. This work concludes in Sect. 5, where we draw lines for future research.

Related Work. Much effort in RES has been dedicated to study highly reliable systems, which includes fault trees [29,30]. When the fail/repair times follow non-Markovian distributions, importance splitting is a usual choice. As long as a full system failure can be broken down into several smaller components failures, an importance splitting method can be devised. Of course, its efficiency relies heavily on the choice of importance function. This choice is typically done ad hoc for the model under study [17,31]. For instance, [29] defined a "state variable" (an importance function for the RESTART algorithm: $S(t) = \max_i\{c_i(t)\}$) for a specific system. In essence, viewing the system as a fault tree, $S(t)$ counts the max number of failed components in any MCS. This was generalised in [30] using cut set analysis to define the importance function $\Phi(t) = cl - oc(t)$. Unlike $S(t)$, $\Phi(t)$ does not require all cut sets to have the same cardinality. However, both functions are hindered when the branches of the fault tree have different failure probabilities. In this work we propose to alleviate this issue via *cut set pruning* and *importance normalisation* [3]. Regarding automation, [4,5,12,13] are among the first to attempt a heuristic derivation of all parameters required to implement splitting. In essence, here we extended [2,4,5], using the cut sets of the fault tree in conjunction with the structural function derived in [3].

2 Theoretical Framework

2.1 Fault Trees

A fault tree '△' is a directed acyclic graph that models how component failures propagate and eventually cause the full system to fail. The leaves are called *basic elements* (or basic events), and model the failure of elemental components. Other nodes called intermediate events are labelled with *gates*, and describe how combinations of lower failures propagate to upper levels. A full-system failure is called a *top level event* (TLE), and takes place when the root node of the tree fails.

In this work we consider the repairable dynamic fault trees (RFTs) presented in [8,19]. Thus, each basic element (BE) b is equipped with a failure distribution F_b that governs its failure probability as a function of time, and a repair distribution R_b that governs its repair time. Some BEs are used as spare components: these (SBEs) replace a primary component when it fails. The dormancy distribution D_b of an SBE b describes its failure while *dormant*, i.e. not in use. Only if b becomes active its failure distribution is given by F_b.

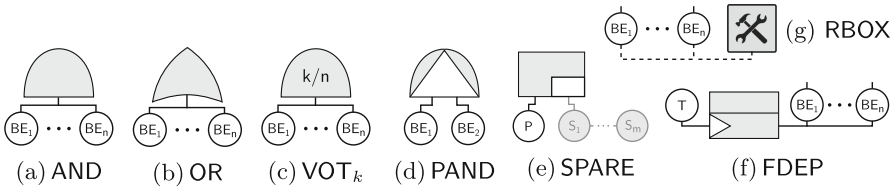


Fig. 1. Fault tree gates and the repair box [3]

RFTs have six types of gates. Their syntax is shown in Figs. 1a to f, and their meaning is as follows: the AND, OR, and VOT_k gates fail if respectively all, one, or k of their m children fail. The latter is called the *voting* or k out of m gate. The *priority-and gate* (PAND) is an AND gate that only fails if its children fail from left to right (or simultaneously). SPARE gates have one *primary* child and one or more *spare* children: spares replace the primary when it fails. The FDEP gate has an input *trigger* and several *dependent events*: all dependent events become unavailable when the trigger fails. Figure 2a shows an RFT without repairs.

RFTs handle repairs by means of *repair boxes* (RBOX [22]). BEs and SBEs can be connected to an RBOX, which determines which basic element is repaired next according to a given policy. Note that the repair distribution (and thus, the time-to-repair of failed components) is an attribute of the basic element.

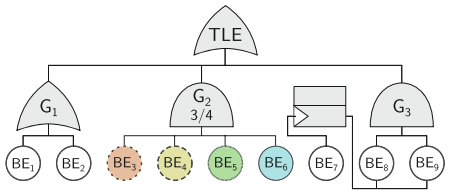
The semantics for (repairable) dynamic fault trees is given in terms of stochastic transition models, such as Markov automata, Petri nets, IOSA, etc. Following [19] we give semantics to RFT as Input/Output Stochastic Automata (IOSA), so that we can handle arbitrary probability distributions. Each state in the IOSA represents a system configuration, indicating which components are operational and which have failed. Transitions among states describe how the configuration changes when failures or repairs occur.

Dynamic fault trees may exhibit nondeterministic behaviour [7,14], for instance when two SPAREs have a single shared SBE: if all elements are failed, and the SBE is repaired first, the failure behaviour depends on which SPARE gets the SBE. Monte Carlo simulation cannot cope with nondeterminism. The theory from [8,19] overcomes this by imposing some mild syntactic conditions, to ensure that the IOSA semantics of an RFT is *weakly deterministic*. This means that all resolutions of nondeterministic choices lead to the same probability value. In particular (1) each BE must be connected to at most one SPARE gate, (2) BEs and SBEs connected to SPAREs are must not be connected to FDEPs, and (3) policies should be provided for RBOX and spare assignments. For full technical details the interested reader is referred to [19].

Fault Tree Analysis. An important goal in FTA is to compute relevant dependability metrics. A popular metrics is *system reliability*, which is the probability of observing no top level event before some mission time $T > 0$. This can be defined as $\mathrm{REL}_T = \mathit{Prob}\left(\forall_{t\in[0,T]} \,.\, X_t = 0\right)$, where X_t denotes the random variable that

represents the state of the top event at time t, which takes the value 1 if there is a TLE, and 0 otherwise. System *unreliability* is $\text{UNREL}_T = 1 - \text{REL}_T$.

Minimal Cut Sets. Cut sets are a well known qualitative technique in FTA for static FTs. A *cut set* is a set of BEs whose joint failure will cause a TLE. A *minimal cut set* (MCS) is a cut set of which no subset is a cut set. These concepts can be lifted to dynamic fault trees (and RFTs) in general, but this requires introducing an order to capture temporal dependencies, plus several other subtleties [14,25]. Nevertheless, RFTs as defined above rule out several issues raised by cut sets in DFTs, such as event simultaneity [14]. Furthermore, in this work we exclude order dependence by considering RFTs without PAND gates. This enables a conservative treatment of cut sets in RFTs, of which we give more details in Sect. 3.1.



(a) An RFT without PANDs

Familiy	Minimal cut sets included
$\{\text{MCS}_j\}$	$\{\text{BE}_1\}$ $\{\text{BE}_2\}$ $\{\text{BE}_7\}$ $\{\text{BE}_8,\text{BE}_9\}$ $\{\text{BE}_3,\text{BE}_4,\text{BE}_5\}$ $\{\text{BE}_3,\text{BE}_5,\text{BE}_6\}$ $\{\text{BE}_3,\text{BE}_4,\text{BE}_6\}$ $\{\text{BE}_4,\text{BE}_5,\text{BE}_6\}$
$\{\text{MCS}_j\}_{<3}$	$\{\text{BE}_1\}$ $\{\text{BE}_2\}$ $\{\text{BE}_7\}$ $\{\text{BE}_8,\text{BE}_9\}$
$\{\text{MCS}_j\}_{>\frac{1}{4}}$	$\{\text{BE}_2\}$ $\{\text{BE}_3,\text{BE}_4,\text{BE}_5\}$ $\{\text{BE}_3,\text{BE}_4,\text{BE}_6\}$ $\{\text{BE}_3,\text{BE}_5,\text{BE}_6\}$ $\{\text{BE}_4,\text{BE}_5,\text{BE}_6\}$

(b) Minimal cut sets of Fig. 2a

Fig. 2. Fault trees & minimal cut sets

For an illustration, Fig. 2b lists all minimal cut sets of the tree from Fig. 2a. We use the notation: $\{\text{MCS}_j\}$ for the family of all minimal cut sets; $\{\text{MCS}_j\}_{<N}$ for the subset of $\{\text{MCS}_j\}$ that excludes cut sets with N or more BEs (called *pruning* of order N); $\{\text{MCS}_j\}_{>\lambda}$ for the subset of $\{\text{MCS}_j\}$ that excludes cut sets where the product of the failure rate of the BEs is $\leqslant \lambda \in \mathbb{R}_{>0}$. The latter is well defined iff all BEs and SBEs in the RFT have Markovian failure and dormancy distributions. To obtain $\{\text{MCS}_j\}_{>\frac{1}{4}}$ in Fig. 2b, we make this the case with failure rates: $\frac{1}{4}$ for BE_1 and BE_7, $\frac{6}{20}$ for BE_2, $\frac{2}{3}$ for $\{\text{BE}_i\}_{i=3}^{6}$, and $\frac{1}{2}$ for BE_8 and BE_9.

Cut set pruning as in $\{\text{MCS}_j\}_{<N}$ and $\{\text{MCS}_j\}_{>\lambda}$ is a standard way to speed up FT analyses [28]. The goal is to ignore the most unlikely (and hard to compute) cut sets: pruning of order N assumes that the TLE will most likely occur by cut sets with less than N BEs; pruning by rate $\leqslant \lambda$ assumes that the TLE will occur first by cut sets where BEs have higher rates and thus fail faster. Choosing such N and λ to prune irrelevant MCS of a given tree depends on its structure and the BEs failure/dormancy/repair distributions.

2.2 Fault Tree Analysis via (Rare Event) Simulation

In this work, we analyse dependability metrics of RFTs using methods based on Monte Carlo simulation. This involves taking random samples from the (stochas-

tic) IOSA model that underlies the RFT. More specifically, *Monte Carlo simulation* means the discrete-event simulation process used to generate failures and repairs of the (IOSA components that model the) BEs of the tree. As described in [3,16], this process begins at simulation time 0 when all basic elements are operational. The next failure time of all these BEs and SBEs is randomly sampled, according to their failure/dormancy distributions, and stored in a heap with the smallest time T_1 at the top. Then, simulation time advances until time T_1, simulating a failure of the corresponding component. As soon as this happens, the repair time of that component is sampled and stored in the events-time heap. Next, simulation time advances until the smallest next-event time at the top of the heap, T_2, simulating either another failure, or a repair of the broken component. This process continues until the predefined end-of-simulation time T. The resulting sequence of fail/repair events in times $T_1 < T_2 < \cdots < T_N < T$ is called a *simulation trace*. The mathematical definition of a compositional semantics for repairable DFTs that allows this approach is introduced in [20]; moreover, [3] formally defines simulation traces in such IOSA semantics.

A main advantage of this approach when compared to purely combinatorial analyses is the capability to deal with non-Markovian distributions. Sampling discrete (random) events is a straightforward and efficient process with today's scientific libraries and computer power. For instance, to estimate the unreliability of an RFT one can sample N independent traces from its IOSA semantics as described above. Then, an unbiased statistical estimator for $p = \text{UNREL}_T$ is the proportion of traces observing a TLE, $\hat{p}$ [16]. The statistical error of $\hat{p}$ can be quantified with two numbers δ and ε s.t. $\hat{p} \in [p - \varepsilon, p + \varepsilon]$ with probability at least δ. The interval $\hat{p} \pm \varepsilon$ is called a *confidence interval* (CI) with coefficient δ and precision 2ε. Such procedures scale linearly with the number of tree nodes and can handle non-Markovian failure and repair PDFs. However, they find a bottleneck to estimate *rare events*: i.e. if $p \approx 0$, then very few traces observe the TLE. Increasing the number of traces alleviates this problem, but even standard CI settings—where ε is relative to p—require sampling an unacceptable number of traces [23]. Rare event simulation techniques solve this specific problem.

RES techniques [23] increase the amount of traces that observe the rare event. In particular, *importance splitting* RES [17] is very flexible w.r.t. the probability distributions it can handle, which makes it a perfect candidate to analyse RFTs. Importance splitting can be efficiently deployed as long as the rare event γ can be described as a nested sequence of less-rare events $\gamma = \gamma_M \subsetneq \gamma_{M-1} \subsetneq \cdots \subsetneq \gamma_0$. This decomposition allows to study the conditional probabilities $p_k = \mathit{Prob}(\gamma_{k+1} \,|\, \gamma_k)$ separately, to then compute $p = \mathit{Prob}(\gamma) = \prod_{k=0}^{M-1} \mathit{Prob}(\gamma_{k+1} \,|\, \gamma_k)$. Moreover, importance splitting requires all conditional probabilities p_k to be much greater than p, so that estimating each p_k can be done efficiently with classical Monte Carlo (CMC).

For this, importance splitting defines the γ_k via an *importance function* $\mathcal{I} \colon S \to \mathbb{N}$, that assigns an *importance* to each state in the system. For us, a state $s \in S$ is a configuration of failed/operational BEs in a tree, where the state space S includes all possible combinations of BE failures. The higher

the importance of a state s, the closer it is to the rare event γ_M. Event γ_k collects states with importance at least ℓ_k, for certain sequence of *threshold levels* $0 = \ell_0 < \ell_1 < \cdots < \ell_M$. Formally: $\gamma_k = \{s \in S \mid \mathcal{I}(s) \geqslant \ell_k\}$

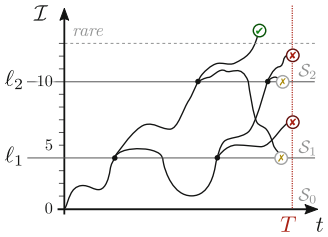


Fig. 3. RST_2 for UNREL_T

Importance splitting samples more (partial) traces from states with higher importance. Two well-known methods are Fixed Effort and RESTART. *Fixed Effort* [9] samples a predefined amount of traces in each region $S_k = \gamma_k \setminus \gamma_{k+1} = \{s \in S \mid \ell_{k+1} > \mathcal{I}(s) \geqslant \ell_k\}$. If the amount of sampled traces is the same in all regions, the *effort* $e \in \mathbb{N}$, we call this method FE_e. Thus, starting at γ_0, FE_e first estimates the proportion of traces that reach γ_1: $p_0 = \mathit{Prob}(\gamma_1 \mid \gamma_0) = \mathit{Prob}(S_0)$. Next, from the states that reached γ_1 new traces are generated to estimate $p_1 = \mathit{Prob}(S_1)$, and so on until p_M. *RESTART* (RST [32,33]) is another algorithm that starts one trace in γ_0 and monitors the importance of the states visited. If the importance of the trace up-crosses threshold ℓ_1, the first state visited in S_1 is saved and the trace is cloned, aka *split*—see Fig. 3. This mechanism rewards traces that get closer to the rare event. Each clone then evolves independently, and if one up-crosses threshold ℓ_2 the splitting mechanism is repeated. Instead, if a state with importance below ℓ_1 is visited, the trace is *truncated*. This penalises traces that move away from the rare event. To avoid truncating all traces, the one that spawned the clones in region S_k can go below importance ℓ_k. To deploy an unbiased estimator for p, RESTART measures how much split was required to visit a rare state [32]. If the same amount of splitting is used throughout, we call this method RST_e, to mean that $e - 1$ clones are spawned when a simulation up-crosses a threshold ℓ_i.

3 RES Using Minimal Cut Sets

The effectiveness of importance splitting relies heavily on the choice of the importance function [17]. Traditionally, this function is given by domain and/or RES experts, requiring domain knowledge both in the application and in the simulation techniques. In this section we introduce a series of importance functions that can be automatically derived from systems described as fault trees.

3.1 MCS Re-writing of Trees

The initial observation is that a fault tree—without temporal requirements such as PAND gates—can be re-written as a disjunction of its minimal cut sets. This is a well-known fact: the key insight is that the structure of the resulting tree can be readily exploited to derive importance functions automatically.

More in detail, given a fault tree $\triangle$ one can build a tree $\triangle^*$ which is equivalent w.r.t. the MCS of $\triangle$. Essentially, $\triangle^*$ is a re-writing in disjunctive normal form (DNF) of the logical formula represented by $\triangle$. The TLE of $\triangle^*$ is an OR gate, the children of that OR are AND gates (or BEs), and the children of the ANDs are BEs. The first-level nodes of $\triangle^*$, i.e. the direct children of the OR, are the conjunctions of the DNF. If a conjunction has a single literal, the AND is omitted and the corresponding BE is a direct input of the OR. Thus, each AND (or first-level BE) in $\triangle^*$ stands for an MCS of $\triangle$.

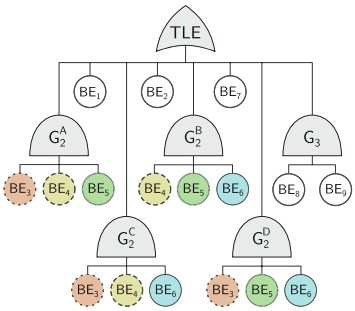


Fig. 4. MCS re-write of Fig. 2a

Figure 4 shows the result of applying this re-writing to the DFT from Fig. 2a. Although it is represented as a tree, note that the G_2^* gates in Fig. 4 share children: e.g. BE_4 and BE_5 are both children of $\mathsf{G}_2^{\mathsf{A}}$ and $\mathsf{G}_2^{\mathsf{B}}$.

For general FTA the scope of this technique is limited. First and foremost, the set $\{\mathrm{MCS}_j\}$ can be exponential in the size of the fault tree, e.g. when $\triangle$ has a conjunctive normal form structure. This is a general limitation of cut set analysis: in such cases the computation of all MCS can be very time/memory consuming, and pruning helps to alleviate the issue. Moreover, minimal cut sets are standard for the analysis of *static* fault trees. In *dynamic* fault trees, PAND gates introduce order dependencies that cannot be captured with MCS. Also, notions of simultaneity and causality of events must be considered to properly lift MCS for the general analysis of DFTs [14]. Note however that repairs do not affect MCS. Thus, in this work we consider the class of *repairable fault trees* (RFT) described in [20]—but excluding PAND gates.

In essence, RFTs are DFTs enriched with RBOX elements, where the propagation of events from children to parents is instantaneous. Potential sources of nondeterminism (e.g. an SBE claimed by two SPAREs) are proven weakly-deterministic, as long as the RFT adheres to the syntactic rules described in Sect. 2. Thus, SPARE gates are essentially ANDs. Moreover, FDEPs in RFTs are non-destructive: if a failed trigger is repaired, the BEs affected by the gate become *available* (not necessarily operational) once again [20]. This can be modelled with OR gates: in Fig. 2a, the FDEP and gate G_3 are equivalent to $\mathsf{OR}(\mathsf{BE}_7, \mathsf{G}_3)$. Therefore, an RFT without PANDs can be seen as a static fault tree whose BEs can be repaired. This means that the MCS re-write of an RFT as $\triangle^*$, gives a faithful description of the sources of top-level failures in the original tree $\triangle$.

To exploit this for the automation of RES analysis of fault trees, we combine this fact with the theory deployed in [3]. As explained in Sect. 2.2, the key ingredient to implement importance splitting is the importance function $\mathcal{I}$. [3] introduces a recursive construction for $\mathcal{I}$ based on the structure of the RFT $\triangle$, which begins from its BEs and ends in the top level gate. The essential contribution of the current work is to apply this same construction to the MCS re-write $\triangle^*$.

3.2 Importance Functions from Minimal Cut Sets

The compositional importance function of [3] is defined per gate (and BE) type. The key concept is that, in general, importance should reflect proximity to the rare event. For fault trees this means that *the importance of a gate should increase as the gate approaches its own failure*. Note that the type of a gate defines how, as its children fail, the gate approaches its own failure. This is used in [3] to define a local importance function for each gate, which assigns an importance to the gate based on its type and on the state of its children.

For instance, AND gates fail when all its children fail, so the importance of an AND should increase with the failure of every child. Thus, the local importance function of ANDs is a summation *of the importance* of its children. By the same argument, the importance of an OR is the max importance among its children. Basic events are the base case of this recursion: BEs and SBEs essentially have a binary state, failed or not,[1] which are assigned importance 1 and 0 resp.

In [3] this recursive construction starts from the leaves of the tree and ends in the top gate, thus deriving a function $\mathcal{I}_{\mathsf{FT}}$ that considers every single gate in the original FT $\triangle$. Instead, here we work on the MCS re-write $\triangle^*$, for which the three cases described above suffice. $\triangle^*$ consists of the top OR, potentially some ANDs, and the base BEs and SBEs: the importance function assigned to such tree is a max (OR) over the summation (ANDs) of every basic event in an MCS.

Table 1 gives the mathematical expression of this formula, which we denote $\mathcal{I}_{\mathsf{MCS}}$. Note that each AND of the MCS re-write $\triangle^*$ represents a minimal cut set of the original tree, viz. an element of $\{\mathrm{MCS}_j\}$. In the mathematical expressions of Table 1, BE_i stands for the elementary importance function described above, which takes the value 1 if the BE is failed and 0 otherwise.

We further consider pruned variants of $\mathcal{I}_{\mathsf{MCS}}$, that discard cut sets based on their cardinality ($\mathcal{I}_{\mathsf{MCS\text{-}P}}$) or, if BE failures are exponentially distributed, based on the product of the failure rates of the BEs in the cut set ($\mathcal{I}_{\mathsf{MCS\text{-}PR}}$). The only difference between these functions and $\mathcal{I}_{\mathsf{MCS}}$ is the range of the max, which reflects the pruning of some minimal cut sets.

Another concept introduced in [3] is *importance normalisation*. The intention is to level the importance values that a parent gate reads from its children. This has the following motivation: take a binary AND gate whose left child is an AND of 2 BEs, and whose right child is an AND of 6 BEs. The top AND fails iff both AND children fail. Thus, having *one* BE of the left AND fail, is just as important as having *three* BEs of the right AND fail, because in both cases one child of the top AND is half-failed. To achieve this, [3] divides the importance of the children of a gate by its maximum possible value—which corresponds to a failed child. In this way, the importance of a gate uses the "percentage of failure" of its children.

We also experiment with this concept: in Table 1, $\mathcal{I}_{\mathsf{MCS\text{-}N}}$ stands for the normalised version if $\mathcal{I}_{\mathsf{MCS}}$. Note that j_n is the number of children of an AND that represents an MCS in the original tree. The importance function of this AND is the summation if its children, all of which are BEs whose max importance is

[1] This is given more thorough semantics in [3] via an output function $\boldsymbol{z}_i$.

Table 1. Importance functions for automatic RES in fault trees

Name	Expression	Description
$\mathcal{I}_{\mathsf{FT}}$	[3, Table 2]	Based on the full FT structure, this importance function considers all nodes recursively, defining local functions for BEs and moving up until the TLE
$\mathcal{I}_{\mathsf{MCS}}$	$\max_{\{\mathsf{BE}_1,\ldots,\mathsf{BE}_{j_n}\}\in\{\mathrm{MCS}_j\}} \left\{ \sum_{i=1}^{j_n} \mathsf{BE}_i \right\}$	For each MCS of the tree, $\mathcal{I}_{\mathsf{MCS}}$ counts the number of BEs that have failed. The importance of the current state of the tree is the max among these counts
$\mathcal{I}_{\mathsf{MCS\text{-}P}}$	$\max_{\{\mathsf{BE}_1,\ldots,\mathsf{BE}_{j_n}\}\in\{\mathrm{MCS}_j\}<N} \left\{ \sum_{i=1}^{j_n} \mathsf{BE}_i \right\}$	$\mathcal{I}_{\mathsf{MCS\text{-}P}}$ operates similarly to function $\mathcal{I}_{\mathsf{MCS}}$ above, but here the max ranges over a *pruned* set of MCS, discarding cut sets with N or more BEs
$\mathcal{I}_{\mathsf{MCS\text{-}PR}}$	$\max_{\{\mathsf{BE}_1,\ldots,\mathsf{BE}_{j_n}\}\in\{\mathrm{MCS}_j\}>\lambda} \left\{ \sum_{i=1}^{j_n} \mathsf{BE}_i \right\}$	Similar to $\mathcal{I}_{\mathsf{MCS\text{-}P}}$ but using the failure *rates* for pruning, $\mathcal{I}_{\mathsf{MCS\text{-}PR}}$ considers only MCS where the product of the failure rate of all BEs is greater than λ
$\mathcal{I}_{\mathsf{MCS\text{-}N}}$	$\max_{\{\mathsf{BE}_1,\ldots,\mathsf{BE}_{j_n}\}\in\{\mathrm{MCS}_j\}} \left\{ \mathrm{lcm} \cdot \sum_{i=1}^{j_n} \frac{\mathsf{BE}_i}{j_n} \right\}$	$\mathcal{I}_{\mathsf{MCS\text{-}N}}$ is a normalised version of $\mathcal{I}_{\mathsf{MCS}}$ (see *normalisation of the importance functions* in [3]) based on the number of BEs of each cut set in $\{\mathrm{MCS}_j\}$

- $\mathsf{BE}_i = 1$ if the i-th BE is in a failed state, and $\mathsf{BE}_i = 0$ otherwise.
- lcm is the least common multiple of the cardinality of every MCS in range.

1. By dividing this summation by its max possible value, j_n, we compute the percentage of failure of the AND. The scaling factor lcm ensures that the resulting value is an integer, as required by our tooling framework. Finally, although omitted in Table 1, we further define the functions $\mathcal{I}_{\mathsf{MCS\text{-}P\text{-}N}}$ and $\mathcal{I}_{\mathsf{MCS\text{-}PR\text{-}N}}$ as the normalised versions of the functions $\mathcal{I}_{\mathsf{MCS\text{-}P}}$ and $\mathcal{I}_{\mathsf{MCS\text{-}PR}}$.

Tool Automation. The theory described in this section is piggybacked in the tool chain of [3], where RFTs are input in an extended version of the Galileo textual format [26,27]. A Java converter builds its IOSA semantics following [20], as well as the compositional importance function of [3]. Model and function are then fed to the FIG tool, which implements various flavours of (importance splitting) RES. FIG output is a confidence interval that estimates the answer to

a quantitative user query, such as system reliability for a given time horizon. Minimal cut sets, required to implement the importance functions in Table 1 other than $\mathcal{I}_{\mathsf{FT}}$, can be computed using the classical top-down algorithms; more efficient methods employ Binary Decision Diagrams [18,25].

4 Empirical Evaluation

To assess the efficiency of our approach, we analyse the reliability of a classic benchmark in FTA: the Hypothetical Example Computer System (HECS [28]).

4.1 Case Study and Experimental Setting

We study a parameterised version of HECS deployed in [3]. The parameter p in HECS_p determines the number of spare processors and parallel buses: HECS_p features p spare processors (PS_i) and $2p$ buses (B_j). Moreover, the system analysed in [3] is repairable and defines one independent RBOX per subsystem (Memory, Interface, Bus, and Processor). Figure 5 shows HECS_3. The full DFT described in (extended) Galileo can be found in Appendix A.

Note that the repair times of BEs are given by PDFs with non-Markovian distributions, but their failure (and dormant-failure) times are all exponential. This allows us to experiment with the pruned importance functions $\mathcal{I}_{\mathsf{MCS\text{-}P}}$ and $\mathcal{I}_{\mathsf{MCS\text{-}PR}}$. Furthermore HECS has cold spares: the SBEs PS_i must not fail while inactive. To encode this and analyse REL_T in our tool chain, we select a dormant-failure time beyond T for the PS_i, e.g. the dormancy distribution $\text{Dirac}(T + \varepsilon)$.

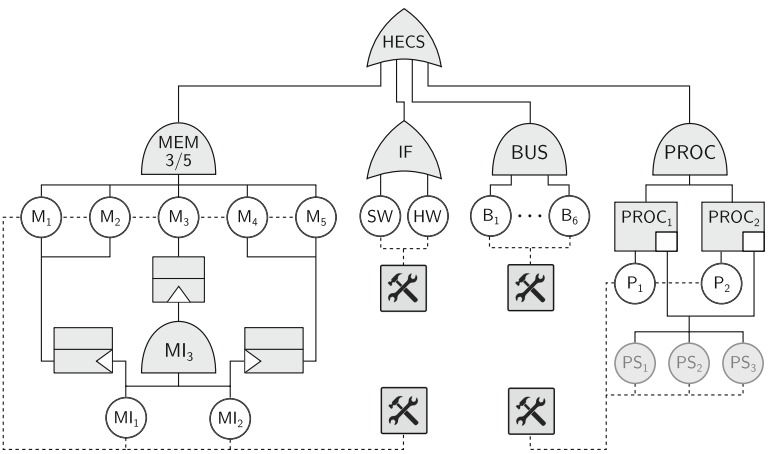


Fig. 5. Repairable DFT for the HECS_3 case study

As in [3] we estimate UNREL_{1000}, hereby called φ. We analyse the systems $\{\text{HECS}_p\}_{p=3}^{5}$ to measure how, as p grows and the corresponding value of φ

decreases, RES analysis can yield increasingly narrower CIs for a fixed simulation time budget of 25 min. We also consider variants of HECS without repairs for which we estimate UNREL_{750}, hereby called ψ.

On the one hand we compare RES vs. CMC, and show that classical Monte Carlo simulation cannot yield useful results when $\varphi < 2.0 \times 10^{-6}$. On the other hand we compare the efficiency of the functions presented in Table 1. For this, on each model HECS_p and for different RES algorithms, we estimate φ, ψ with FIG running RES for 25 min. The best importance functions are those with which FIG can implement RES and produce the narrowest confidence intervals. Thus, the goal is to identify whether any importance function can consistently produce the narrowest CIs, with most RES algorithms, in all HECS_p models.

Importance functions $\mathcal{I}_{\text{MCS-P}}$ and $\mathcal{I}_{\text{MCS-PR}}$ (and their normalised versions) require a pruning criterion. We selected $N = 4$ for the former and $\lambda = 1 \times 18^{-18}$ for the latter. Thus, for all models, $\mathcal{I}_{\text{MCS-P}}$ discards the cut sets corresponding to the Bus and Processor subsystems. Instead, $\mathcal{I}_{\text{MCS-PR}}$ discards all cut sets corresponding to the Memory subsystem (with the sole exception of $\{\text{MI}_1, \text{MI}_2\}$) for HECS_3 and HECS_4. For HECS_5, $\mathcal{I}_{\text{MCS-PR}}$ also discards the cut set of Processors. All functions for HECS_3 are shown in Appendix B.

4.2 Results and Discussion

Figure 7 shows the results of our experiments on HECS without repairs, where we estimated $\psi = \text{UNREL}_{750}$. Figure 8 shows the results for HECS with repairs, where we estimated $\varphi = \text{UNREL}_{1000}$. We ran FIG on a computer with a CPU Intel® Xeon® E7-8890 v4 @ 2.20 GHz and 2 TB of RAM DDR4 @ 1866 MHz, running Linux x64 (Ubuntu, kernel 3.13.0-168).

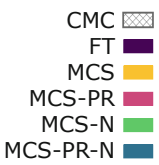


Fig. 6. Color legend of bar plots

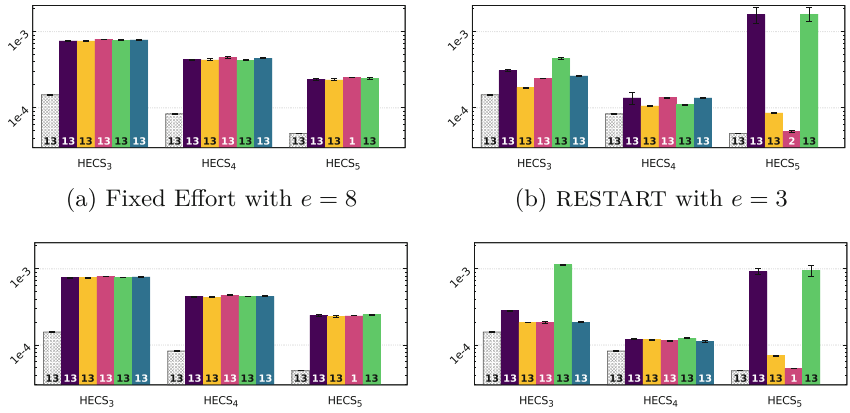
We use whisker-bar plots to show the width of the 95% CIs estimated for each instance. An *instance* is a combination of a RES algorithm, a model, and an importance function—e.g. $\text{FE}_{e=8}$, HECS_3, and $\mathcal{I}_{\text{MCS}}$—represented in Figs. 7 and 8 by one bar in one plot. Each instance was repeated 13 times. The height of a bar represents the resulting average CI width: achieved by that algorithm, in that model, via that importance function (we removed outliers using a $\text{Z-score}_{m=2}$ [11]). The whiskers on top of the bar represent the variance of these widths. The number at the base of a bar indicates how many out of the 13 repetitions of that instance yielded valid results—if no rare event is observed, FIG outputs the "null CI" $[0, 0]$ to indicate an invalid estimation.

We use the same colour of bar-instance to identify the importance functions across plots. The colour legend is shown in Fig. 6, where CMC stands for classical Monte Carlo (i.e. not an importance function), FT stands for the $\mathcal{I}_{\text{FT}}$ importance function from [3], MCS stands for $\mathcal{I}_{\text{MCS}}$ from Table 1, and so on. Therefore, to assess an importance function, we must see how each colour fared in the following orthogonal criteria: *bar height*, where shorter means narrower CIs and is thus

better, *whisker length*, where shorter means less variance and is thus better, and *validity count*, where 13 is best and 0 (or absence of a bar) is worst.

Functions $\mathcal{I}_{\text{MCS-P}}$ and $\mathcal{I}_{\text{MCS-P-N}}$ are not shown in Figs. 7 and 8 because they only yielded null CIs. This was expected: the Processors cut set is a main cause of TLEs due to the failure rates involved—see Code 1 in Appendix A. Pruning that cut set makes importance insensitive to the relevant failures of the system. Thus, cloning and truncating simulation traces is a futile overhead, which makes RES fare even worse than CMC. This also explains why $\mathcal{I}_{\text{MCS-PR}}$ is competitive for $\text{HECS}_{3,4}$ but never for HECS_5, where it prunes the Processors cut sets.

Regarding CMC, Fig. 7 shows that RES does not pay off in general to estimate metrics like ψ when these are above 2×10^{-6}. This changes in Fig. 8, where CMC lost to almost every RES implementation for HECS_4, and for HECS_5 (where $\varphi \approx 3 \times 10^{-7}$) it could not produce CIs narrower than 3×10^{-6}, thus including 0.



(a) Fixed Effort with $e = 8$ (b) RESTART with $e = 3$

(c) Fixed Effort with $e = 16$ (d) RESTART with $e = 4$

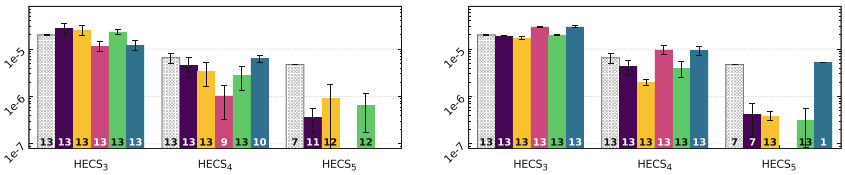
Fig. 7. CI precision for UNREL_{750} of HECS without repairs: four RES algorithms

Going back to the results for HECS without repairs (Fig. 7) we see that CMC consistently outperforms RES. Here we estimated $\psi = \text{UNREL}_{750}$ instead of UNREL_{1000}. Our importance functions are oblivious to the simulation time, and therefore a smaller time horizon hinders importance splitting w.r.t. CMC. This is because simulation runs are cloned closer to the (truncating) time limit, incurring computation overhead that may yield no rare event observations. Automatic importance functions sensitive to simulation time are an interesting line of research: we addressed this again in the conclusions. On the other hand, our method performs best when operating with large time horizons, such that the rarity of UNREL_T is only lightly influenced by T[2].

[2] This complements standard model checking, where time-bounded properties with large time bounds entail memory problems [25].

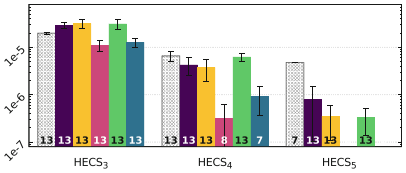
Also remarkable is the fact that $\mathcal{I}_{\mathsf{FT}}$ and $\mathcal{I}_{\mathsf{MCS\text{-}N}}$ performed very bad for RST_3 and RST_4 in HECS_5 without repairs—Figs. 7b and d. We repeated our experiments and the same behaviour was observed. For these two cases, it was found that the expression of the importance functions had large scaling factors (lcm in Table 1) due to importance normalisation, namely 210. For other functions this factor ranges from 2 to 30. We suspect that the variability of importance during RES resulted in over-splitting and truncation, to which RESTART is more sensitive than Fixed Effort. This is also observed (to a lesser degree) in HECS_3, where again $\mathcal{I}_{\mathsf{FT}}$ and $\mathcal{I}_{\mathsf{MCS\text{-}N}}$ have the largest scaling factor (30)—see Table 2.

Focusing now in Fig. 8 and to our surprise, we see that $\mathcal{I}_{\mathsf{MCS}}$ fared quite well, many times even outperforming $\mathcal{I}_{\mathsf{FT}}$—see e.g. Fig. 8b and d, in particular for HECS_5. We had expected that the absence of importance normalisation would unbalance importance computations, making RES implementations from $\mathcal{I}_{\mathsf{MCS}}$ loose on performance, which was clearly not the case. This however could be a result that most system failures originate in the Processors subsystem, so considering other cut sets in HECS may not pay off. In any case, the normalised version of this function, viz. $\mathcal{I}_{\mathsf{MCS\text{-}N}}$, almost always performed as well as $\mathcal{I}_{\mathsf{MCS}}$, and similarly with $\mathcal{I}_{\mathsf{FT}}$. For RFTs with several (three or more) MCS that can equally likely lead to a TLE, functions such as $\mathcal{I}_{\mathsf{MCS\text{-}N}}$ and $\mathcal{I}_{\mathsf{FT}}$ should outperform un-normalised variants like $\mathcal{I}_{\mathsf{MCS}}$ and $\mathcal{I}_{\mathsf{MCS\text{-}PR}}$.

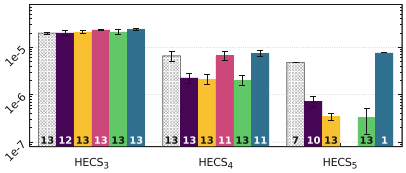


(a) Fixed Effort with $e = 8$

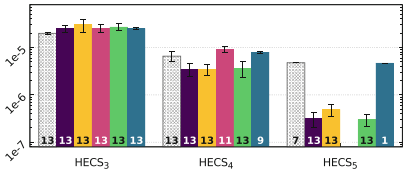
(b) RESTART with $e = 3$



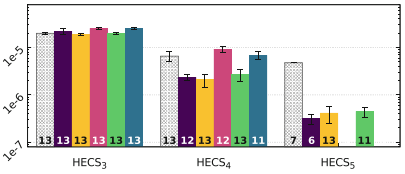
(c) Fixed Effort with $e = 16$



(d) RESTART with $e = 4$



(e) Fixed Effort with $e = 32$



(f) RESTART with $e = 5$

Fig. 8. CI precision for UNREL_{1000} of HECS with repairs: six RES algorithms

As mentioned in Sect. 1, $\mathcal{I}_{\mathsf{MCS}}$ is equivalent to the importance function $\Phi(t)$ from [30], essentially counting the number of failed BEs in the MCS with the largest such number. A simpler approach would be to count the number of failed elements in the tree, i.e. disregarding cut sets. This function was named AC_1 in [5], and applied to a small tree representing a database system. There, the function AC_4 represents what we here call $\mathcal{I}_{\mathsf{MCS}}$. Both functions AC_1 and AC_4 performed similarly, which was explained by the balanced nature of the RFT: the failure of any one branch (cut set) was similar to the rest. This was further corroborated in [2] with another system representing an oil-refinery pipeline.

A final insight is given by the near absence of valid results by $\mathcal{I}_{\mathsf{MCS\text{-}PR\text{-}N}}$ for HECS_5, most likely due to its pruning of the Processors cut set. This suggests that cut set pruning is non-trivial for relatively complex repairable fault trees, even when the failure of all BEs is given in terms of exponential rates.

5 Conclusions

In this work we have defined importance functions for RES analysis of fault trees. These functions can be automatically derived from the tree structure, by re-writing the tree as a disjunction of its minimal cut sets. This can be applied to repairable dynamic fault trees without PAND gates, that are given semantics as IOSA. Our method exploits a simple concept, and yet we have demonstrated that it can outperform classical Monte Carlo analysis, and even compete against other automatically derived importance functions for RES analysis of RFTs.

Our empirical evaluation was sound yet arguably modest in size. HECS is a classical FTA benchmark that features several gates an heterogeneous sub-tree structures, but it remains a single case study. It is important to exercise our techniques in further systems, e.g. to determine whether mildly-balanced trees (where most MCS have roughly similar contributions to the TLE) makes a distinction between $\mathcal{I}_{\mathsf{MCS}}$ and its normalised counterpart $\mathcal{I}_{\mathsf{MCS\text{-}N}}$.

An interesting line of future research would be to weigh the cut sets based on their failure probability or failure rates. Moreover, the study of REL_{750} for HECS without repairs raised the issue of time-awareness for importance computation. If the rarity of the dependability metric is based on the shortness of the time horizon, our approach will most likely not perform well, since it is oblivious to this dimension. Penalising importance based on the remaining time may be a way to approach this issue.

A Galileo of the HECS_3 DFT

```
1 toplevel "HECS";
2 "HECS" or "IF" "MEM" "BUS" "PROC";
3
4 "IF" or "SW" "HW";
5 "SW"  lambda=4.5e-12  EXT_repairPDF=uniform(28,56);
6 "HW"  lambda=1.0e-10  EXT_repairPDF=uniform(28,56);
7
```

```
"MEM" 3of5 "M1" "M2" "M3" "M4" "M5";
"MEM1" fdep "MI1" "M1" "M2";
"MEM2" fdep "MI2" "M4" "M5";
"MEM3" fdep "MI3" "M3";
"MI3" and "MI1" "MI2";
"MI2" lambda=5.0e-9  EXT_repairPDF=uniform(21,28);
"MI1" lambda=5.0e-9  EXT_repairPDF=uniform(21,28);
"M1"  lambda=6.0e-8  EXT_repairPDF=uniform(21,28);
"M2"  lambda=6.0e-8  EXT_repairPDF=uniform(21,28);
"M3"  lambda=6.0e-8  EXT_repairPDF=uniform(21,28);
"M4"  lambda=6.0e-8  EXT_repairPDF=uniform(21,28);
"M5"  lambda=6.0e-8  EXT_repairPDF=uniform(21,28);

"BUS" and "B1" "B2" "B3" "B4" "B5" "B6";
"B1"  lambda=8.7e-4  EXT_repairPDF=lognormal(4.45,0.24);
"B2"  lambda=8.7e-4  EXT_repairPDF=lognormal(4.45,0.24);
"B3"  lambda=8.7e-4  EXT_repairPDF=lognormal(4.45,0.24);
"B4"  lambda=8.7e-4  EXT_repairPDF=lognormal(4.45,0.24);
"B5"  lambda=8.7e-4  EXT_repairPDF=lognormal(4.45,0.24);
"B6"  lambda=8.7e-4  EXT_repairPDF=lognormal(4.45,0.24);

"PROC" and "PROC1" "PROC2";
"PROC1" wsp "P1" "PS1" "PS2" "PS3";
"PROC2" wsp "P2" "PS1" "PS2" "PS3";
"P1"  lambda=1.0e-3                        EXT_repairPDF=lognormal(4.45,0.24);
"P2"  lambda=1.0e-3                        EXT_repairPDF=lognormal(4.45,0.24);
"PS1" lambda=1.5e-3 EXT_dormPDF=dirac(1e4) EXT_repairPDF=lognormal(4.45,0.24);
"PS2" lambda=1.5e-3 EXT_dormPDF=dirac(1e4) EXT_repairPDF=lognormal(4.45,0.24);
"PS3" lambda=1.5e-3 EXT_dormPDF=dirac(1e4) EXT_repairPDF=lognormal(4.45,0.24);

"RB_I" repairbox_priority "HW" "SW";
"RB_M" repairbox_priority "MI1" "MI2" "M1" "M2" "M3" "M4" "M5";
"RB_B" repairbox_priority "B1" "B2" "B3" "B4" "B5" "B6";
"RB_P" repairbox_priority "P1" "P2" "PS1" "PS2" "PS3";
```

Code 1. Description of HECS_3 in (extended) Galileo

B Importance Functions Used for HECS Experiments

In Table 2 we give the importance functions used in Sect. 4, for experimentation with the case study HECS_3, which are oblivious of repairs. The arithmetic expressions use the names of the IOSA modules that give semantics to the BEs of the DFT. For instance nodes SW, HW, MI1, and MI2, are all BEs. In the IOSA semantics these BEs correspond to modules named BE_0, BE_1, BE_3, and BE_8 resp. Since the $\{\mathrm{MCS}_j\}$ family of HECS_3 includes $\{\mathtt{SW}\}$, $\{\mathtt{HW}\}$, and $\{\mathtt{MI1}, \mathtt{MI2}\}$, therefore in Table 2 the max of function $\mathcal{I}_{\mathsf{MCS}}$ ranges (among others) over the following three summations: BE_0, BE_1, and BE_3+BE_8.

We highlight that the importance function $\mathcal{I}_{\mathsf{FT}}$ from [3] is more complex, but not necessarily larger than the functions introduced in the current work. This is mainly a consequence of the VOT gate in the Memory subsystem, a 3of5, which yields 10 minimal cut sets. The FDEPs give even further TLE possibilities. Indeed, from the 21 minimal cut sets of HECS, 17 come from failure combinations in the Memory subsystem.

Table 2. Importance functions used for HECS_3

IFUN	Expression in terms of the IOSA semantic model
$\mathcal{I}_{\text{FT}}$	`max(30*(max(BE_0,BE_1)),5*(summax(3,2*(max(BE_3,BE_4)),2*(max(BE_3,BE_6)),max(BE_3+BE_8,2*(BE_10)),2*(max(BE_8,BE_12)),2*(max(BE_8,BE_14)))),5*(BE_17+BE_18+BE_19+BE_20+BE_21+BE_22),3*(max(BE_24+BE_25+BE_29+BE_31,(5.0*(SPARE_26==9?1:0)))+max(BE_33+BE_25+BE_29+BE_31,(5.0*(SPARE_27==9?1:0)))));0;30`
$\mathcal{I}_{\text{MCS}}$	`max(BE_0,BE_1,BE_4+BE_6+BE_10,BE_4+BE_6+BE_12,BE_4+BE_6+BE_14,BE_4+BE_10+BE_12,BE_4+BE_10+BE_14,BE_4+BE_12+BE_14,BE_6+BE_10+BE_12,BE_6+BE_10+BE_14,BE_6+BE_12+BE_14,BE_10+BE_12+BE_14,BE_3+BE_10,BE_3+BE_12,BE_3+BE_14,BE_8+BE_4,BE_8+BE_6,BE_8+BE_10,BE_3+BE_8,BE_24+BE_33+BE_25+BE_29+BE_31,BE_17+BE_18+BE_19+BE_20+BE_21+BE_22);0;6`
$\mathcal{I}_{\text{MCS-PR}}$	`max(BE_0,BE_1,BE_3+BE_10,BE_3+BE_12,BE_3+BE_14,BE_8+BE_4,BE_8+BE_6,BE_8+BE_10,BE_3+BE_8,BE_24+BE_33+BE_25+BE_29+BE_31);0;5`
$\mathcal{I}_{\text{MCS-N}}$	`max(30*(BE_0),30*(BE_1),10*(BE_4+BE_6+BE_10),10*(BE_4+BE_6+BE_12),10*(BE_4+BE_6+BE_14),10*(BE_4+BE_10+BE_12),10*(BE_4+BE_10+BE_14),10*(BE_4+BE_12+BE_14),10*(BE_6+BE_10+BE_12),10*(BE_6+BE_10+BE_14),10*(BE_6+BE_12+BE_14),10*(BE_10+BE_12+BE_14),15*(BE_3+BE_10),15*(BE_3+BE_12),15*(BE_3+BE_14),15*(BE_8+BE_4),15*(BE_8+BE_6),15*(BE_8+BE_10),15*(BE_3+BE_8),6*(BE_24+BE_33+BE_25+BE_29+BE_31),5*(BE_17+BE_18+BE_19+BE_20+BE_21+BE_22));0;30`
$\mathcal{I}_{\text{MCS-PR-N}}$	`max(10*(BE_0),10*(BE_1),5*(BE_3+BE_10),5*(BE_3+BE_12),5*(BE_3+BE_14),5*(BE_8+BE_4),5*(BE_8+BE_6),5*(BE_8+BE_10),5*(BE_3+BE_8),2*(BE_24+BE_33+BE_25+BE_29+BE_31));0;10`

References

1. Bayes, A.J.: Statistical techniques for simulation models. Aust. Comput. J. **2**(4), 180–184 (1970)
2. Budde, C.E.: Automation of importance splitting techniques for rare event simulation. Ph.D. thesis, Universidad Nacional de Córdoba, Córdoba, Argentina (2017)
3. Budde, C.E., Biagi, M., Monti, R.E., D'Argenio, P.R., Stoelinga, M.: Rare event simulation for non-Markovian repairable fault trees. In: TACAS 2020 (to appear)
4. Budde, C.E., D'Argenio, P.R., Hermanns, H.: Rare event simulation with fully automated importance splitting. In: Beltrán, M., Knottenbelt, W., Bradley, J. (eds.) EPEW 2015. LNCS, vol. 9272, pp. 275–290. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23267-6_18
5. Budde, C.E., D'Argenio, P.R., Monti, R.E.: Compositional construction of importance functions in fully automated importance splitting. In: VALUETOOLS. ICST (2016). https://doi.org/10.4108/eai.25-10-2016.2266501
6. Cérou, F., Del Moral, P., Furon, T., Guyader, A.: Sequential Monte Carlo for rare event estimation. Stat. Comput. **22**(3), 795–808 (2012). https://doi.org/10.1007/s11222-011-9231-6
7. Crouzen, P., Boudali, H., Stoelinga, M.: Dynamic fault tree analysis using input/output interactive Markov chains. In: DSN 2007, pp. 708–717 (2007). https://doi.org/10.1109/DSN.2007.37
8. D'Argenio, P.R., Monti, R.E.: Input/output stochastic automata with urgency: confluence and weak determinism. In: Fischer, B., Uustalu, T. (eds.) ICTAC 2018. LNCS, vol. 11187, pp. 132–152. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02508-3_8
9. Garvels, M.J.J.: The splitting method in rare event simulation. Ph.D. thesis, University of Twente, Enschede, The Netherlands (2000)

10. Heidelberger, P.: Fast simulation of rare events in queueing and reliability models. ACM Trans. Model. Comput. Simul. **5**(1), 43–85 (1995). https://doi.org/10.1145/203091.203094
11. Iglewicz, B., Hoaglin, D.: How to Detect and Handle Outliers. ASQC Basic References in Quality Control. ASQC Quality Press, Milwaukee (1993)
12. Jegourel, C., Legay, A., Sedwards, S.: Importance splitting for statistical model checking rare properties. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 576–591. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_38
13. Jégourel, C., Legay, A., Sedwards, S., Traonouez, L.M.: Distributed verification of rare properties using importance splitting observers. In: ECEASST, vol. 72 (2015). https://doi.org/10.14279/tuj.eceasst.72.1024
14. Junges, S., Guck, D., Katoen, J., Stoelinga, M.: Uncovering dynamic fault trees. In: DSN 2016, pp. 299–310. IEEE (2016). https://doi.org/10.1109/DSN.2016.35
15. Kahn, H., Harris, T.E.: Estimation of particle transmission by random sampling. Natl. Bur. Stand. Appl. Math. Ser. **12**, 27–30 (1951)
16. Law, A.M.: Simulation Modeling and Analysis. McGraw-Hill Education, New York (2014)
17. L'Ecuyer, P., Le Gland, F., Lezaud, P., Tuffin, B.: Splitting techniques. In: Rubino and Tuffin [24], pp. 39–61. https://doi.org/10.1002/9780470745403.ch3
18. Lee, W., Grosh, D., Tillman, F., Lie, C.: Fault tree analysis, methods, and applications–a review. IEEE Trans. Reliab. **R–34**(3), 194–203 (1985). https://doi.org/10.1109/TR.1985.5222114
19. Monti, R.E.: Stochastic automata for fault tolerant concurrent systems. Ph.D. thesis, Universidad Nacional de Córdoba, Argentina (2018)
20. Monti, R.E., D'Argenio, P.R., Budde, C.E.: A compositional semantics for repairable fault trees with general distributions. arXiv e-prints arXiv:1910.10507 (2019)
21. Nicola, V.F., Shahabuddin, P., Nakayama, M.K.: Techniques for fast simulation of models of highly dependable systems. IEEE Trans. Reliab. **50**(3), 246–264 (2001). https://doi.org/10.1109/24.974122
22. Raiteri, D., Iacono, M., Franceschinis, G., Vittorini, V.: Repairable fault tree for the automatic evaluation of repair policies. In: DSN 2004, pp. 659–668 (2004). https://doi.org/10.1109/DSN.2004.1311936
23. Rubino, G., Tuffin, B.: Introduction to rare event simulation. In: Rubino and Tuffin [24], pp. 1–13. https://doi.org/10.1002/9780470745403.ch1
24. Rubino, G., Tuffin, B. (eds.): Rare Event Simulation Using Monte Carlo Methods. Wiley, New York (2009). https://doi.org/10.1002/9780470745403
25. Ruijters, E., Stoelinga, M.: Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. Comput. Sci. Rev. **15–16**, 29–62 (2015). https://doi.org/10.1016/j.cosrev.2015.03.001
26. Sullivan, K., Dugan, J.: Galileo user's manual & design overview (1998). v2.1-alpha. https://www.cse.msu.edu/~cse870/Materials/FaultTolerant/manual-galileo.htm
27. Sullivan, K., Dugan, J., Coppit, D.: The Galileo fault tree analysis tool. In: 29th Annual International Symposium on Fault-Tolerant Computing (Cat. No. 99CB36352), pp. 232–235 (1999). https://doi.org/10.1109/FTCS.1999.781056
28. Vesely, W., Stamatelatos, M., Dugan, J., Fragola, J., Minarick, J., Railsback, J.: Fault tree handbook with aerospace applications. NASA Office of Safety and Mission Assurance, version 1.1 (2002)
29. Villén-Altamirano, J.: RESTART method for the case where rare events can occur in retrials from any threshold. Int. J. Electron. Commun. **52**, 183–189 (1998)

30. Villén-Altamirano, J.: Importance functions for RESTART simulation of highly-dependable systems. Simulation **83**(12), 821–828 (2007). https://doi.org/10.1177/0037549707081257
31. Villén-Altamirano, J.: RESTART vs splitting: a comparative study. Perform. Eval. **121**–**122**, 38–47 (2018). https://doi.org/10.1016/j.peva.2018.02.002
32. Villén-Altamirano, M., Martínez-Marrón, A., Gamo, J., Fernández-Cuesta, F.: Enhancement of the accelerated simulation method RESTART by considering multiple thresholds. In: Proceedings of the 14th International Teletraffic Congress. Teletraffic Science and Engineering, vol. 1, pp. 797–810. Elsevier (1994). https://doi.org/10.1016/B978-0-444-82031-0.50084-6
33. Villén-Altamirano, M., Villén-Altamirano, J.: RESTART: a method for accelerating rare event simulations. In: Queueing, Performance and Control in ATM (ITC-13), pp. 71–76. Elsevier (1991)