This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

Automatic inference of fault tree models via multi-objective evolutionary algorithms

Lisandro A. Jimenez-Roa, Tom Heskes, Tiedo Tinga, and Mariëlle Stoelinga

Abstract—Fault tree analysis is a well-known technique in reliability engineering and risk assessment, which supports decision-making processes and the management of complex systems. Traditionally, fault tree (FT) models are built manually together with domain experts, considered a time-consuming process prone to human errors. With Industry 4.0, there is an increasing availability of inspection and monitoring data, making techniques that enable knowledge extraction from large data sets relevant. Thus, our goal with this work is to propose a data-driven approach to infer efficient FT structures that achieve a complete representation of the failure mechanisms contained in the failure data set without human intervention. Our algorithm, the FT-MOEA, based on multi-objective evolutionary algorithms, enables the simultaneous optimization of different relevant metrics such as the FT size, the error computed based on the failure data set and the Minimal Cut Sets. Our results show that, for six case studies from the literature, our approach successfully achieved automatic, efficient, and consistent inference of the associated FT models. We also present the results of a parametric analysis that tests our algorithm for different relevant conditions that influence its performance, as well as an overview of the data-driven methods used to automatically infer FT models.

Index Terms—Fault tree analysis, evolutionary algorithms, multi-objective optimization, complex systems, model learning, parametric analysis.

List of Abbreviations

FTA: Fault Tree Analysis
FT: Fault Tree
MCS: Minimal Cut Set
BE: Basic Event
TE: Top Event
EA: Evolutionary Algorithm
MOEA: Multi-Objective Evolutionary Algorithm
m.o.f: Multi objective function

1 Introduction

Fault Tree Analysis (FTA) is a widely used method in reliability engineering and risk analysis, mainly because it enables modeling complex systems by encoding and displaying logical relationships that can be used, among others, to understand how a system might fail, trace the root cause of the failure, identify critical components, and calculate the system and subsystem failure probabilities.

Fault tree (FT) models exist since the 1960s and have been used in a wide range of domains, including the automotive, aerospace, and nuclear industries [1]. However, a major drawback of FTs is related to their construction, which is traditionally carried out in conjunction with domain expertise and in a hand-crafted manner, resulting in a tedious and time-consuming task. In the case of complex industrial systems, manual development of these models can lead to incompleteness, inconsistencies, and even errors [2].

The above challenge has been discussed since the 1970s, and it is referred to in the literature as *construction* [3], *synthesis* [4], or *induction* [5] of FTs. In this work, we refer to this as *automatic inference of FT models*, which in general, is the process that automatically (with limited human intervention) produces an FT model given compatible input information.

This problem shares some similarities with System Identification (SI), where the objective is to identify the mathematical model of a given system [6], although one difference we observe between SI and FTs inference is that for SI it is necessary to pre-define a model structure (e.g., based on laws of physics), which is not

- *Lisandro A. Jimenez-Roa is with the Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), University of Twente, Drienerlolaan 5, 7522 NB Enschede, The Netherlands. E-mail: l.jimenezroa@utwente.nl*
- *Tom Heskes is with the Institute for Computing and Information Sciences (iCIS), Radboud University Nijmegen, 6525 EC Nijmegen, The Netherlands. E-mail: Tom.Heskes@ru.nl*
- *Tiedo Tinga is with the Faculty of Engineering Technology (ET), University of Twente, Drienerlolaan 5, 7522 NB Enschede, The Netherlands. E-mail: t.tinga@utwente.nl*
- *Mariëlle Stoelinga is with the Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), University of Twente, and the Department of Software Science, Radboud University Nijmegen, The Netherlands. E-mail: m.i.a.stoelinga@utwente.nl*

Manuscript received September 24, 2021; revised -, 2021.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

possible in the case of FTs inference as this is a task of the inference process.

We identify FTs inference challenging because there are many possible FTs for a given failure data set, and finding the best match is not trivial. Existing methods fail as (i) they need too much human intervention to add assumptions e.g., to deal with complex dependencies between components; (ii) they do not scale adequately in real-world applications, especially algorithms that perform exhaustive search have exponential time complexity; (iii) they result in complex FT structures, (iv) it is unknown how reliable they are under noisy data.

We are interested in data-driven approaches, whose challenge is illustrated by the following example: Table 1 shows an toy input failure data set (Section 4.1). Suppose the associated system is composed of the components BE1, BE2 and BE3, where 0 and 1 are used as non-faulty and faulty states, respectively. Top corresponds to the system-level failure.

TABLE 1: Toy input failure data set.

BE1	BE2	BE3	Top
1	1	1	1
1	0	1	1
0	1	1	1
1	1	0	1
1	0	0	0
0	1	0	0
0	0	1	1
0	0	0	0

Thus, for this failure data set, we want to find the FT structure that best encodes the logic that describes the failure propagation in the system. Moreover, we are interested in the FT composed of a minimal amount of elements.

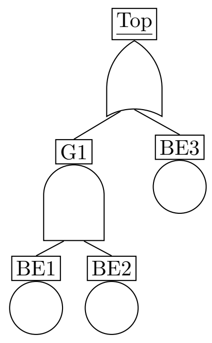

Fig. 1: Inferred FT.

The desired output is presented in Fig. 1 , where the inferred FT is composed of basic events BE1, BE2, connected to an And gate (G1), which together with BE3, is connected with an Or gate to the top event Top.

Here the gates, which connect the basic events and the top event, result from the inference process following the logic described by the failure data set. And although for this example the solution is rather simple, for larger failure data sets (i.e., with more basic events) the solution is not straightforward.

Our contributions with this paper are that (i) we show it is possible to achieve more consistent and efficient FT structures via *multi-objective evolutionary algorithms* (MOEAs), which simultaneously optimize various criteria in a multi-dimensional space, (ii) we propose a metric to compare FT structures via Minimal Cut Sets using the RV-coefficient, (iii) we carry out a parametric analysis that explains the performance of the algorithm under different assumptions, (iv) we show that maintaining compact FTs benefits scalability by enabling faster convergence. The implementation and all data are available at zenodo.org/record/5536431.

The remaining part of this paper is organized as follows. Section 2 summarizes the related work. Section 3 formally defines FTA and provides the technical background of MOEAs. Section 4 explains our methodology. Section 5 presents how we apply the NSGA-II (an MOEA) to infer FTs. Section 6 presents the results of a thorough parametric analysis. Section 7 discusses our findings and presents our conclusions.

2 Related Work

Different ways of inferring FTs have been discussed in the past. We identify three main groups namely *knowledge-based*, *model-based*, and *data-driven*. The main differences between these categories are that *knowledge-based* approaches mainly employ different heuristics for knowledge representation and domain expertise [7]; *model-based* approaches translate existing system models and/or graphs into FTs, and *data-driven* approaches have structured databases as the primary source of information, where the goal is to identify causal relationships present in a failure data set with minimal domain expertise and human intervention.

Carpignano & Poucet [8] thoroughly review several knowledge-based approaches. An example of a model-based approach can be found in Mhenni et al. [9] where the authors used SysML System Models as a base to obtain FT models. However, a major drawback of model-based approaches is the need for a pre-existing model [10].

As mentioned before, we focus our attention on *data-driven* approaches, where the applications of machine learning techniques and data analytics fall into this category. In Appendix A, Table 6 (divided into two parts) we summarise and compare relevant literature in data-driven methods for the automatic inference of FT models. Below we briefly discuss these methods.

To the best of our knowledge, the very first attempt to tackle the challenge in a data-driven manner was made by Madden & Nolan [5] with their IFT algorithm, which is based on Quinlan's ID3 algorithm to induce Decision Trees (DTs) [11]. The authors also continued with this work in the subsequent years [12], [13]. Mukherjee & Chakraborty [14] addressed this challenge via *linguistic analysis* and domain knowledge to identify the nature of the failure from short plain text descriptions of equipment faults, and from this generate an FT model. Roth et al. [15] propose a method that follows the Structural Complexity Management (StCM) methodology, where their main goal is to deduce dependencies that are later on used to infer the Boolean logic operators of the FT models. Inspired by *Causal Decision Trees* [16], in Nauta et al.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

[17] they proposed an approach based on the *Mantel-Haenszel* statistical test.

Based on *Knowledge Discovery in Database*, Waghen & Ouali [18] propose a method for hierarchical causality analysis called *Interpretable Logic Tree Analysis* (ILTA), that looks for patterns in a data set which are translated into *Interpretable Logic Trees*. Linard et al. [19] proposed a method that consists of learning a *Bayesian Network* graph, which is later translated into an FT model. Lazarova-Molnaretal [20] presents DDFTA, an approach based on time series of failure data, binarization techniques, Minimal Cut Sets, and Boolean algebra. This algorithm also manages to infer FT models with VoT gates. In Waghen & Ouali [21] the authors further extended their work to the *Multi-level Interpretable Logic Tree Analysis* (MILTA), which tackles the problem of multiple cause-and-effect sequences (the latter a limitation of their ILTA algorithm) by incorporating Bayesian probability rules.

The first attempt based on *evolutionary algorithms* was carried out by Linard et al. [22], here the authors created an algorithm to generate FT models from a labeled binary failure data set using a uni-dimensional cost function based on the *accuracy* i.e., the proportion of correctly predicted top events by a given FT.

A drawback of the above approach is that it focuses solely on achieving high accuracy without considering the FT structure. The latter has negative implications such as: running the algorithm twice for the same failure data set may yield considerably different FT structures; it is prone to complexity explosion, leading to massive FTs that are difficult to handle; long computational time, and bad convergence of the algorithm.

3 THEORETICAL BACKGROUND

3.1 Fault Tree Analysis

Fault Tree Analysis (FTA) is one of the most famous methods in reliability engineering that supports decisions in the design and maintenance of complex systems. FTA enables *qualitative* and *quantitative* analyses. Qualitative analysis is based on the FT structure and aims at finding the critical system components. Here, an important concept refers to the *Minimal Cut Sets* (MCSs), which are minimal combinations of component failures that lead to a system failure. Small minimal cut sets point to system vulnerabilities.

The quantitative analysis aims at computing various dependability metrics, such as system *Reliability*; *Availability*; and the *Mean-Time-to-Failure*. Estimation of these metrics requires that the FT leaves are equipped with failure probabilities.

A **Fault Tree** (FT) helps in understanding why a system fails by modeling how low-level failures propagate through the system and lead to the system-level failure. As similarly done by Ruijters and Stoelinga [23], to formally define an FT, we first define $\mathrm{GateType} = \{\mathrm{And}, \mathrm{Or}\} \cup \{\mathrm{VoT(k/N)} \mid \mathrm{k},\ \mathrm{N} \in \mathbb{N}^{\geq 1},\ \mathrm{k} \leq \mathrm{N}\}$. Then, an FT is a 5-tuple $\mathrm{F} = \langle \mathrm{BE}, \mathrm{G}, \mathrm{T}, \mathrm{I}, \mathrm{TE} \rangle$ where

- BE is a set of basic events, which may be annotated with a probability of occurrence $\mathrm{p_i}$.
- G is a set of logic gates, with $\mathrm{BE} \cap \mathrm{G} = \emptyset$.
- $\mathrm{E} = \mathrm{BE} \cup \mathrm{G}$ for the set of elements.
- $\mathrm{T} : \mathrm{G} \rightarrow \mathrm{GateTypes}$ is a function that describes the type of each gate.
- $\mathrm{I} : \mathrm{G} \rightarrow \mathrm{P(E)}$ describes the inputs of each gate. We require $\mathrm{I(g)} \neq \emptyset$ and that $|\mathrm{I(g)}| = \mathrm{N}$ if $\mathrm{T(g)} = \mathrm{VoT(k/N)}$.
- And gate is a tuple $\langle \mathrm{And}, \mathrm{I}, \mathrm{O} \rangle$ where O outputs *true*, if *every* $\mathrm{i} \in \mathrm{I}$ occurs.
- Or gate is a tuple $\langle \mathrm{Or}, \mathrm{I}, \mathrm{O} \rangle$ where O outputs *true*, if *at least one* $\mathrm{i} \in \mathrm{I}$ occurs.
- $\mathrm{VoT(k/N)}$ gate is a tuple $\langle \mathrm{VoT}, \mathrm{k}, \mathrm{I}, \mathrm{O} \rangle$ where O outputs *true*, if *at least* k of $\mathrm{i} \in \mathrm{I}$ occur.
- $\mathrm{TE} \in \mathrm{E}$ is a unique root called the top event, and it is reachable from all other nodes.

Importantly, the graph formed by (E, I) should be a directed acyclic graph. Fig. 2.(a) and 2.(b) depict respectively the *event* and *gate* symbols used to construct the FT model.

Fig. 2.(c) provides an example of an FT. This is the FT of a *Container Seal Design* adapted from [24]. In this FT, the top event, the *sealing function fails*, either occurs if a *common cause seal failure* occurs or if the *seals fail independently*. For the former, it is necessary that the *contamination tape fails*, and a *basic cause seal failure* occurs. For the latter, it is necessary for the *metal-to-metal seal*, the *fused plug*, and at least two of the three *compression seals* to fail.

3.2 Multi-Objective Evolutionary algorithms

Evolutionary algorithms (EAs) are population-based search strategies that use the fundamental principle of natural selection, where the best individuals are more likely to reproduce and, therefore, to pass on to the next generations [25]. When the EA deals with several conflicting objective functions to be simultaneously optimized in a multi-dimensional space [26], we call these *Multi-Objective Evolutionary Algorithms* (MOEAs). The result of MOEAs is a set of solutions with trade-offs, better known as *Pareto-optimal* solutions, from which the user can decide on the basis of higher-level qualitative considerations [27].

As a first step to tackle the challenge of automatically inferring FTs from a failure data set by simultaneously optimizing different metrics, we decided to use one of the most popular algorithms in multi-objective optimization called the *Elitist Non-dominated Sorting Genetic Algorithm* (NSGA-II) (Section 3.2.1) and the *Crowding-Distance* (Section 3.2.2).

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

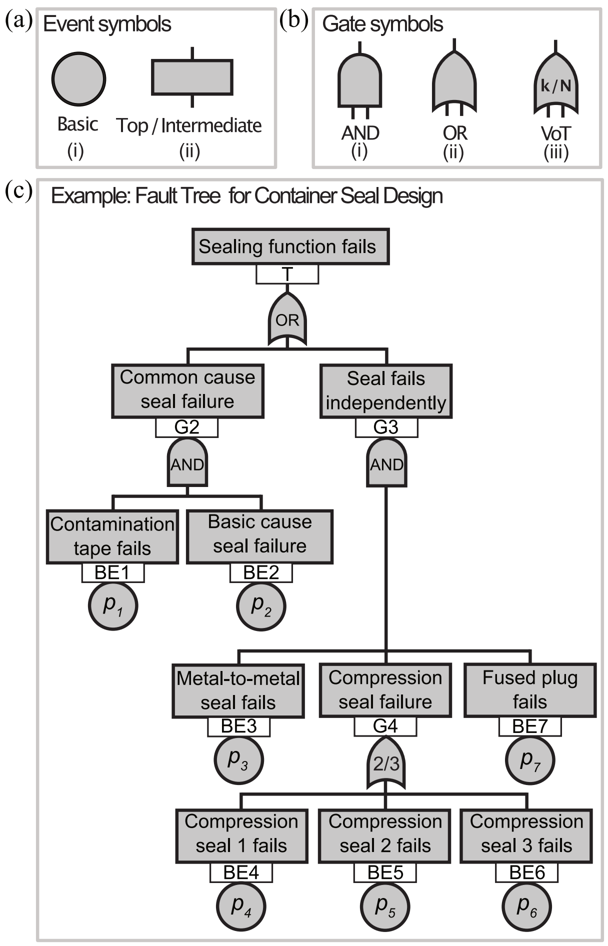

Fig. 2: Elements in an FT: (a) event symbols, (b) gate symbols, and (c) example of an FT, adapted from [24].

3.2.1 Elitist Non-dominated Sorting Genetic Algorithms (NSGA-II)

The *Elitist Non-dominated Sorting Genetic Algorithms* (NSGA-II) [28] aim at finding multiple Pareto-optimal solutions. NSGA-II uses the *elitist* principle which is a diversity preserving mechanism that emphasizes the non-dominated solutions [27]. Moreover, the elitist principle guarantees that the quality of the solution does not decrease, by letting the best individual(s) of the current generation pass to the next one.

Non-dominated MOEAs use the concept of *dominance*, where two solutions are compared to evaluate whether one dominates the other or not. Non-dominated sorting is a crucial step to uncovering elitist efficient solutions in MOEAs, but it is computationally expensive since it involves numerous comparisons [29]. A set of solutions that do not dominate each other is known as a *non-dominated front*. In Appendix B.1 we provide a detailed explanation of these concepts and exemplify how the NSGA-II is applied in the automatic inference of FTs.

3.2.2 Crowding-Distance

Since it may be the case that the last non-dominated front obtained through the NSGA-II does not fully accommodate the available slots to complete the new population, the *Crowding-Distance* is used to decide which individuals of the last front can pass to the next generation, acting as a mechanism that promotes diversity [30]. In Appendix B.2 we provide details on how the Crowding-Distance is applied in the automatic inference of FTs.

4 METHODOLOGY

Fig. 3 depicts the general methodology we followed in this paper. First, we selected some case studies of existing FTs (Section 6.2), these FTs act as ground truth. Then, we selected some parameters of interest (Section 6.4) to be evaluated in the parametric analysis. We used the Monte Carlo method (Section 6.1) to generate failure data sets (Section 4.1) based on selected case studies (Section 6.2). Then we used our FT-MOEA algorithm (Section 5) to infer the FT based on the provided failure data set. Finally, we compared the ground truth with the inferred FTs and evaluate the experiment (Section 6).

4.1 The failure data set

Our methodology needs the following assumptions for the input failure data set:

- *Labeled*: Our data set consists of combinations of BE and their corresponding TE.
- *Binary*: Both BE and TE are binary. This poses an advantage because we can make use of Boolean operations which makes the algorithm faster. In our case, 0 and 1 are used as non-faulty and faulty states, respectively.
- *Monotonic/consistent*: For a given set of BE, if a single BE changes from state 0 to 1, it is possible that TE changes from state 0 to 1, but it will never change from state 1 to 0.
- *Complete*: The total number of unique combinations of BE in the failure data set equals the space complexity $O(2^w)$, where w corresponds to the number of unique BE for a given FT.
- *Noise-free*: There is no corrupted information contained in the failure data set. In other words, the relation BE $\rightarrow$ TE is always true for a given FT.

Table 2 presents an example of the data set associated with the FT in Fig. 2. To generate this data set, we used the Monte Carlo method described in Section 6.1 for $\mathrm{N} = 250,000$ data points and using a failure rate for the BE of $p_i = 0.5$. *Ob.* refers to the *observation* associated with a unique combination of values of BE,

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

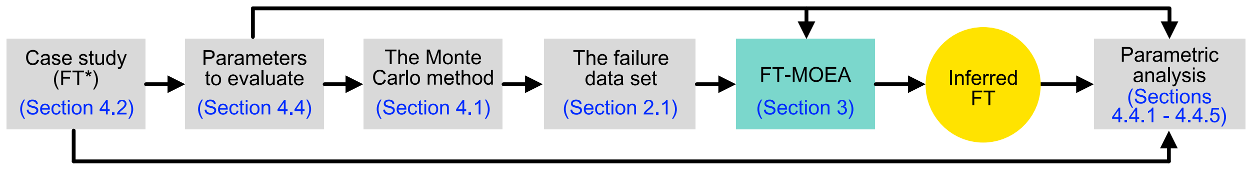
Case study (FT*) (Section 4.2)
Parameters to evaluate (Section 4.4)
The Monte Carlo method (Section 4.1)
The failure data set (Section 2.1)
FT-MOEA (Section 3)
Inferred FT
Parametric analysis (Sections 4.4.1 - 4.4.5)

Fig. 3: General methodology.

and the associated TE. The columns BE1, BE2, ..., BE7 show the *states* of the set of BE. TE corresponds to the *top event*. Finally, the last column refers to the *count* of each of the observations in the failure data set.

TABLE 2: Example of failure data set associated to the example in Fig. 2.

Ob.	BE1	BE2	BE3	BE4	BE5	BE6	BE7	TE	*Count*
1	0	0	0	0	0	0	0	0	1,968
2	0	0	0	0	0	0	1	0	2,039
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
24	0	0	1	0	1	1	1	1	1,976
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
128	1	1	1	1	1	1	1	1	1,947
$p \approx$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	$N =$	250,000

5 Inferring Fault Trees via Multi-Objective Evolutionary Algorithms (FT-MOEA)

The input of the algorithm is composed of the failure data set, described in Section 4.1, and the initial parameters of the MOEA, described in Section 5.1. The output of the algorithm is a string that describes the structure of the inferred FT (based solely on And and Or gates), and metrics of interest that tell about the error between the inferred FT and the failure data set. These metrics are explained in detail in Section 5.5.1. Fig. 4 provides an overview of our approach, consisting of the following five steps:

1) Step 1: Initialize the algorithm by loading the *failure data set* (Section 4.1) as well as the initial parameters (Section 5.1). An optional step is the extraction of the MCSs from the failure data set (Section 5.2), which is executed only when the objective function considers metrics based on MCSs.
2) Step 2: Initialize the population with the *parent fault tree(s)* (Section 5.3) on which the genetic operators are applied until the desired population size is reached.
3) Step 3: Apply the *genetic operators* (Section 5.4) to randomly modify the structure of the FTs. These genetic operators are recursively applied to a population of FTs until reaching at least the desired population size.
4) Step 4: For each FT in the offspring population, compute the *metrics* to be optimized (Section 5.5.1). Then, using the concept of *Pareto efficiency* through the *Elitist Non-Dominated Sorting Algorithm (NSGA-II)* and *Crowding-Distance* (Section 3.2), determine the population of FTs for the next generation.
5) Step 5: Check whether any of the convergence criteria (Section 5.6) are met. If not, the genetic operators (Step 3) are applied to the new population, and Steps 4 and 5 are applied recursively until at least one of the convergence criteria is met, outputting a Pareto set of inferred FTs where we chose the best individual i.e., the FT with the smallest size, and smallest error(s) within the first Pareto set.

5.1 Step 1 - Initialization

We have the following initial parameters.

- *Population size* (ps): Corresponds to the number of FTs within a generation. Only the best ps FTs can pass to the next generation.
- *Selection strategy*: For the NSGA-II algorithm we only use the *elitist* selection strategy.
- *Max. generations with unchanged best candidate* (uc): if after uc number of generations the best individual (i.e., the FT with the smallest size, and smallest error(s) within the best Pareto set) remains unchanged, then we assume the process has converged and is therefore terminated.
- *Max. number of generations* (ng): Terminates the optimization process if the number of generations exceeds ng and none of the other convergence criteria is met.

5.2 Step 1.2 - Extraction of MCSs from the failure data set (optional step)

As mentioned before, MCSs are a minimal combinations of component failures that lead to a system failure. This information is extremely valuable because it encodes the failure modes of the system. By considering this information in our optimization process we are adding additional criteria that can help our

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

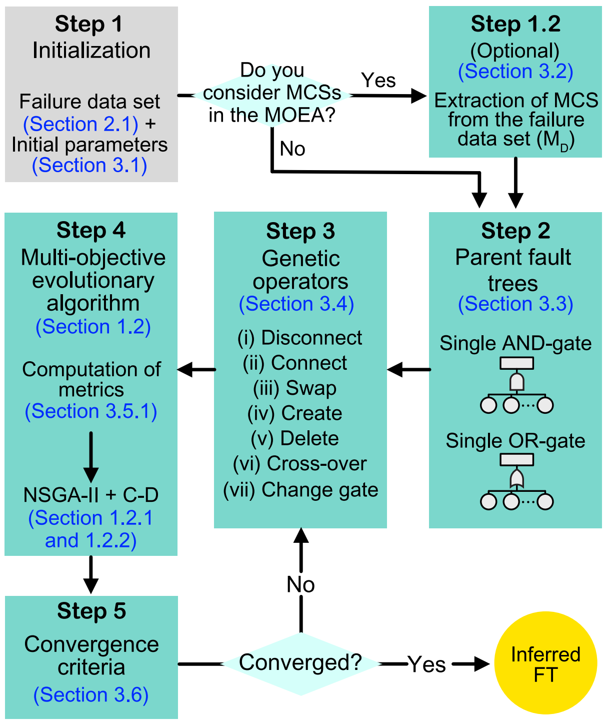

Fig. 4: General process of the FT-MOEA algorithm to infer FTs from a failure data set.

algorithm to find better solutions in a shorter amount of time. However, the user can decide whether to consider MCSs within the optimization process.

One should consider MCSs only if it is guaranteed that the failure data set is *noise-free*, and if the expected FT is not too complex (see Section 6.4.3 where we address the topic of complexity in FTs). The former because otherwise one cannot be sure whether MCSs are correct, and the latter because computing MCSs is computationally expensive (we will discuss this in Section 5.5.1) and for complex FTs, this could increase the convergence time.

The process of extracting MCSs from a failure data set is done as described by Lazarova-Molnar et al. [20] with the following four steps. *Step 1:* identify all the combinations of BE that output class 1 (i.e., $\mathrm{TE} = 1$). *Step 2:* from this sub-set, identify the one with the minimal order (where the order is defined as the number of *true* BE in an observation) and save this observation as part of the MCSs matrix computed from the failure data set ($\mathrm{M_D}$). *Step 3:* look in the sub-set for other observations that include the previously identified MCS. If any, delete them from the sub-set. *Step 4:* repeat Steps 2 and 3 until the sub-set is empty.

The obtained $\mathrm{M_D}$ can be used as an input argument to compute the accuracy based on the MCSs (ϕ_c) (see Eq. 3). Table 3 provides an example of $\mathrm{M_D}$ for the failure data set described in Table 2.

TABLE 3: Example of MCS matrix ($\mathrm{M_D}$) computed from the failure data set described in Table 2 associated with the example in Figure 2.

MCS	*BE1*	*BE2*	*BE3*	*BE4*	*BE5*	*BE6*	*BE7*
1	1	1	0	0	0	0	0
2	0	0	1	1	1	0	1
3	0	0	1	0	1	1	1
4	0	0	1	1	0	1	1

5.3 Step 2 - parent fault tree(s)

The *parent fault tree(s)* can be seen as that one (or those ones) from which the offspring population is generated when applying the *genetic operators* (Section 5.4). Defining the parent FT(s) is important because it determines how far away, from the global optimum, one starts the optimization process. In Linard et al. [22] two parent FTs are used to generate the offspring population, one in which the set of BE are connected to a single Or gate, and the other to a single And gate.

5.4 Step 3 - Genetic operators

The genetic operators are mathematical operations that seek to modify the structure of an FT. We use the seven genetic operators proposed by Linard et al. [22], which also gives the formal definitions of these operators. For completeness, we here provide a short description.

(i) G-create: randomly creates an And or Or gate under an existing gate in the set G for a given FT. *(ii) G-mutate*: randomly selects a gate in the set G and changes its type (i.e., Or $\rightarrow$ And, or And $\rightarrow$ Or). *(iii) G-delete*: from a given FT takes a gate in the set G and deletes it, including its children. *(iv) BE-disconnect*: from a given FT takes a basic event in the set BE and disconnects it. *(v) BE-connect*: from a given FT takes a disconnected basic event and randomly places it under a gate in the set G. *(vi) BE-swap*: from a given FT takes a basic event in the set BE and randomly moves it under a different parent gate in the set G. *(vii) Crossover*: randomly chooses two FTs in the offspring population, then an element in the set E of each FT is randomly selected and exchanged.

5.5 Step 4 - Multi-objective function

In Section 5.5.1 we define the metrics to be minimized, and in Section 5.5.2 we describe the different setups of our multi-objective optimization function.

5.5.1 *Computation of metrics*

We consider three metrics in our multi-objective function, namely the *fault tree size* (ϕ_s), the *error based on the failure data set* (ϕ_d), and the *error based on the MCSs* (ϕ_c).

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

- *Fault Tree Size* (ϕ_s): corresponds to the number of elements in the FT i.e., the value of E in the FT (Eq. 1).

$$\phi_s = |\mathrm{E}| = |\mathrm{BE}| + |\mathrm{G}| \tag{1}$$

Here note that $\phi_s \geq 2$, as each FT has at least one BE and one G (i.e., the TE).
- *Error based on the failure data set* (ϕ_d): First we need to compute a one-dimensional vector P with N values, where N is the number of data points. P contains the value of the TE of an FT for a given set BE. For the same set BE, we count with the value of the ground truth top event (TE*), which is in the failure data set (Section 4.1). Thus, ϕ_d is computed as follows

$$\phi_d = 1 - \frac{\sum_{i=1}^{\mathrm{N}} x_i}{\mathrm{N}} \begin{cases} x_i = 1, \textit{if } \mathrm{P_i} = \mathrm{TE_i^*}. \\ x_i = 0, \text{Otherwise}. \end{cases} \tag{2}$$

Here ϕ_d varies in the closed interval $[0, 1]$, where 0 means that an FT manages to perfectly map the given set BE in the failure data set into the corresponding TE in the failure data set.
- *Error based on the MCSs* (ϕ_c): we propose to compute the error of an FT based on MCSs by means of the *RV-coefficient* [31]. The RV-coefficient is a generalization of the *squared Pearson correlation coefficient* and measures the similarity between the MCS matrix computed from the failure data set ($\mathrm{M_D}$) (see Section 5.2) and the MCS matrix of a given FT ($\mathrm{M_F}$).
$\mathrm{M_F}$ is computed based on the *disjunctive normal form* (DNF), which in Boolean logic is also known as an Or of Ands. Thus, we transform a given FT into its DNF and identify from it the MCSs to construct $\mathrm{M_F}$. However, this transformation showed to be computationally expensive for large FT sizes.
In our context, $\mathrm{M_D}$ is a $p \times w$ matrix, $\mathrm{M_F}$ is a $q \times w$ matrix, w corresponds to the number of unique BEs considered in the problem and p and q are the number of MCSs in the failure data set in an FT, respectively. The computation of ϕ_c is defined as

$$\phi_c = 1 - \frac{\mathrm{tr}(\mathrm{M_D M_F^T M_F M_D^T})}{\sqrt{\mathrm{tr}(\mathrm{M_D M_D^T})^2 \mathrm{tr}(\mathrm{M_F M_F^T})^2}} \tag{3}$$

Here $\mathrm{tr}(.)$ is the *trace*, and ϕ_c varies in the closed interval $[0, 1]$, where 0 indicates perfect correlation or similarity between $\mathrm{M_D}$ and $\mathrm{M_F}$. We choose to use the RV-coefficient as a means to compute the error based on MCSs because, for a given problem, the number of unique BEs always remains the same, but the FTs within a population for a given generation often have different number of MCSs with respect the ones found in the failure data set (i.e., $p \neq q$).

5.5.2 Setups of the multi-objective functions

Since our multi-objective function (m.o.f.) has three arguments, we can play with different setups and assess their influence (see Section 6.4.2 for the results of the parametric analysis). For example, if we want to minimize only the error based on the MCSs (ϕ_c), we can "turn off" ϕ_s and ϕ_d by assigning constant values (i.e., $\phi_s = \phi_d = 1$). To differentiate between the different configurations in the m.o.f., we propose the nomenclature presented in Table 4. Here 'x' refers to whether a metric is being considered (or active) in the m.o.f.

TABLE 4: Different setups of the m.o.f.

m.o.f.	ϕ_s	ϕ_d	ϕ_c
sdc	x	x	x
dc		x	x
sc	x		x
sd	x	x	
c			x
d		x	

5.6 Step 5 - Convergence criterion

Our convergence criterion is based on two initial parameters namely the *max. number of generations* (ng), and the *max. generations with unchanged best candidate* (see Section 5.1). Additionally, we terminate the convergence process if $\phi_c = 0$ or $\phi_d = 0$ when the minimization of the FT size is "turned off" i.e., for the m.o.f.'s *cd*, *c*, or *d*.

6 Experimental Evaluation

For our experimental evaluation, we first selected six case studies from the literature (Section 6.2), then we made the implementation of our FT-MOEA algorithm in Python, whose source code and data are available at zenodo.org/record/5536431. We evaluate our algorithm using synthetic failure data sets (Section 6.1). Section 6.3 compares FT-MOEA with FT-EA and shows details on convergence. Our parametric analysis is presented in Section 6.4.

6.1 The Monte Carlo method

We use the Monte Carlo method to generate synthetic failure data sets using the case studies presented in Section 6.2 and keeping the same properties of the input data set described in Section 4.1. To generate the synthetic dataset (i) we randomly generate (*N*) data points, by drawing the BE *independently* from a *binomial distribution* with a probability of success equal to p_i, where i corresponds to a basic event,

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

TABLE 5: Case studies and associated relevant information. The number of: unique BEs (w), total BEs (W), And gates (#And), Or gates (#Or), VoT gates (#VoT), Minimal Cut Sets (#MCS), order of MCSs (O-MCSs), and space complexity ($O(2^w)$).

Case study	*Ref.*	w	W	*#And*	*#Or*	*#VoT*	FT_{size}	*#MCSs*	*O-MCSs*	$O(2^w)$
CSD$^{(a)}$	[24]	6	6	2	2	0	10	3	{2,3,3}	64
PT$^{(b)}$	[24]	6	6	1	4	0	11	5	{1,1,2,2,2}	64
COVID-19$^{(c)}$	[32]	9	21	9	3	0	33	6	{3,4,4,4,4,4}	512
ddFT$^{(d)}$	[20]	8	8	3	1	1	13	6	{3,4,4,5,6,6}	256
MPPS$^{(e)}$	[24]	8	12	3	8	0	23	7	{2,2,2,2,2,2,2}	256
SMS$^{(f)}$	[33]	13	17	0	8	0	25	13	{1,1,1,1,1,1,1,1,1,1,1,1,1}	8,192

$^{(a)}$**CSD**: Container Seal Design; $^{(b)}$**PT**: Pressure Tank ; $^{(c)}$**COVID-19**: COVID-19 infection risk; $^{(d)}$**ddFT**: Data-driven Fault Tree; $^{(d)}$**MPPS**: Mono-propellant propulsion system; $^{(d)}$**SMS**: Spread Monitoring System.

and (ii) computing the corresponding TE by following the logical rules of the given FT (e.g., case studies in Section 6.2). As one of the main requirements for the failure data set is to be *complete* (see Section 4.1) we satisfied this condition for each case study by drawing enough data points from the Monte Carlo simulation ensuring that the number of unique observations of BE equals the space complexity $O(2^w)$, where w corresponds to the number of unique BE for a give FT.

6.2 Case studies

In order to establish a sensible ground truth, we used existing FTs in the literature with different applications. Our selection criteria were the number of elements in the FT as well as the number of MCSs and their orders. Table 5 presents the case studies we selected together information on their number of BE, the total number of And, Or, and VoT gates, the number of MCSs, their orders, and the space complexity which is measured as $O(2^w)$. Since we work only with complete data sets, the space complexity in Table 5 also indicates the size of the failure data set. We make a distinction between the number of unique BE (w) and the total number of BE (W) because some FTs have shared BE.

6.3 Key findings of the FT-MOEA algorithm

To illustrate our findings and main contributions, we will use the case study Mono-propellant propulsion system (MPPS) from Table 5. We first generate the failure data set as described in Section 6.1, with $N = 250.000$ data points. Then, we used this failure data set as part of the input data of the FT-MOEA algorithm, together with the following initial parameters: $ps = 400$, $ng = 100$, and $uc = 20$.

We first compare the evolutionary process between generations for two m.o.f.'s d and *sdc*. In this way we can respectively compare the approach by Linard et al. [22] (FT-EA, only minimizing ϕ_d) and our multi-objective optimization process (FT-MOEA, minimizing ϕ_s, ϕ_d, and ϕ_c).

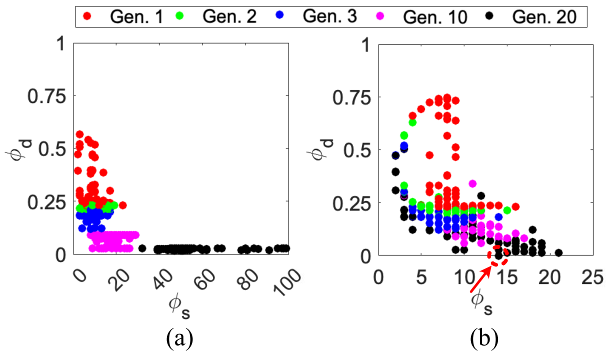

Fig. 5: Evolution of metrics over generations. In (a) using the m.o.f. d, in (b) using the m.o.f. *sdc*, for the MPPS case study ($ps = 400$, $ng = 100$, $uc = 20$). The red dashed circle with the arrow at the bottom of (b) indicates the global optimum.

Fig. 5.(a) shows the results of different generations minimizing solely ϕ_d. One can observe that within the first generations there is a rapid decrease in ϕ_d but from the 10th generation onwards, there is a rapid growth in the size of the FTs (ϕ_s) (up to $\phi_s = 100$) without decreasing ϕ_d, whereas the ground truth FT is $\phi_s = 23$. On the other hand, by using the FT-MOEA (Fig. 5.(b)) we observe a smoother decrease in all directions. Additionally, we observe that the FT-MOEA found the global optimum (i.e., $\phi_d = \phi_c = 0.0$) in the 20th generation with $\phi_s = 14$ (red dashed circle with the arrow at the bottom of 5.(b)) i.e., a compressed version of the ground truth.

In Fig. 6 we compare both m.o.f.'s across generations using only the metrics of the best FT per generation (i.e., that one in the first Pareto front that has the smallest error ϕ_d and ϕ_c). In Fig. 6.(a) we analyze ϕ_d across the generations for both objective functions, we can see that the m.o.f. d more rapidly minimize ϕ_d compared to the m.o.f. *sdc*. However, the latter m.o.f. managed to achieve the global optimum in the 20th generation, whereas the former m.o.f. did

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

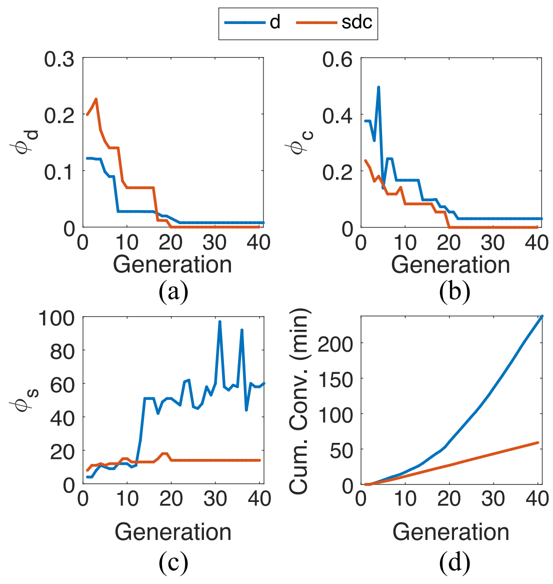

Fig. 6: Metrics over generations for the best FTs using the m.o.f.'s d and sdc. Convergence of (a) ϕ_d, (b) ϕ_c, (c) ϕ_s, (d) cumulative convergence time. Using the MPPS case study ($ps = 400$, $ng = 100$, $uc = 20$).

not find the global optimum. Fig. 6.(b) compares ϕ_c, with pretty similar results.

Fig. 6.(c) depicts the variation of ϕ_s over the generations. One can observe that our m.o.f. keeps smaller FT structures, and although the size of the ground truth FT is 23, the FT-MOEA found one of $\phi_s = 14$, i.e., a compressed version of the original one (for more details see Appendix C, Fig. 11)

Fig. 6.(d) depicts the cumulative time to convergence (t) for both m.o.f.'s. We can observe that our algorithm manages to find the optimal solution in about 20 minutes. On the other hand, by just minimizing ϕ_d the process takes about 4 hours without finding the global optimum. In Appendix D we provide details on the convergence of metrics over the generations for the whole population.

6.4 Parametric analysis

We consider in our parametric analysis the population size, the multi-objective functions, the FT complexity, and the influence of superfluous variables. Additionally, we evaluate the influence of varying the parent FT in the Appendix E. We decided to explore these parameters to understand their impact on the computational time and convergence.

We generate the failure data set as described in Section 6.1. Since the evolutionary algorithm is a stochastic process, we run our algorithm five times per combination of parameters until convergence, and by using *box charts* in Matlab (e.g., Fig. 7) we depict the groups of numerical data through their quartiles.

6.4.1 Population size

Fig. 7 presents the results of the parametric analysis when varying the population size for the m.o.f.'s d and sdc. Here we use the MPPS case study (Table 5).

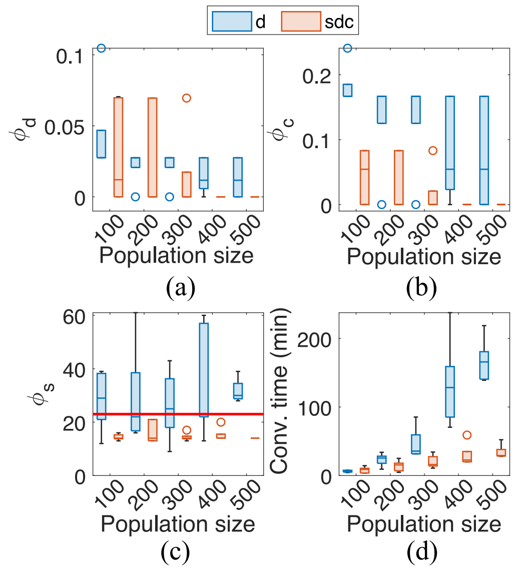

Fig. 7: Influence of population size (ps) on (a) ϕ_d, (b) ϕ_c, (c) ϕ_s, and (d) convergence time. For the m.o.f's sdc and d, for the case study MPPS ($ps = 400$, $ng = 100$, $uc = 20$).

From Fig. 7.(a) and 7.(b), the m.o.f. sdc is more consistent when the population size is larger, also both errors (ϕ_c and ϕ_d) tend to decrease with larger population sizes. On the other hand, when using the m.o.f. d, it seems that the errors decrease for larger population sizes, but with less consistency.

Fig. 7.(c) shows that the m.o.f. sdc always retrieves smaller FTs compared with the m.o.f. d, even smaller than the ground truth (i.e., $\phi_s \leq 23$) indicated by the horizontal red line.

Fig. 7.(d) shows that larger population sizes exponentially increases the computational time for both m.o.f.'s. However, the m.o.f. sdc is consistently faster.

6.4.2 Multi-objective functions

We evaluate all Setups of our m.o.f. (Table 4), for this, we use the case studies in Table 5, and keep the input parameters $ps = 400$, $ng = 100$, $uc = 20$ fixed. Fig. 8 presents the results for the case studies COVID-19, MPPS, and ddFT. Fig. 15 (Appendix F) presents the results of the case studies CSD, PT, and SMS.

Fig. 8.(a) shows the error based on the failure data set (ϕ_d). We observe different behaviors per objective function. The m.o.f. dc achieves the exact solution for all the cases, whereas the other m.o.f.'s failed in at least one of the cases to find the global optimum.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

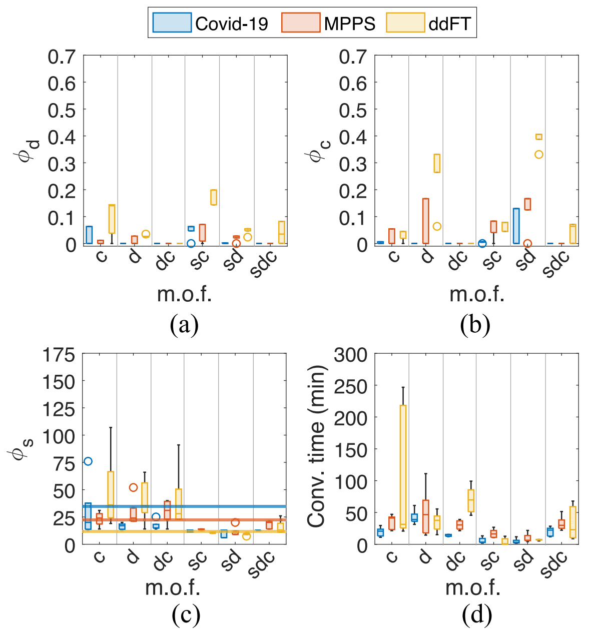

Fig. 8: Comparing the performance of m.o.f.'s for all the case studies in Table 5, using $ps = 400$, $ng = 100$, $uc = 20$. In (a) error based on the failure data set (ϕ_d), (b) error based on the MCSs (ϕ_c), (c) fault tree size (ϕ_s), and (d) convergence time.

Similarly, Fig. 8.(b) shows the error based on MCSs (ϕ_c). As expected, we observe that the m.o.f. *dc* achieves $\phi_c = 0.0$ for all the cases. Nevertheless, notice that a low ϕ_d does not imply an optimal FT, as an example, compare ϕ_d and ϕ_c for the case MPPS for the m.o.f.'s d and sd.

We note that both errors ϕ_d and ϕ_c (especially for the ddFT case study) appear to increase when ϕ_s is minimized, (i.e., m.o.f.s *sc*, *sd*, and *sdc*), this is because FT-MOEA may converge to a FT with a slightly larger error, but smaller size.

Fig. 8.(c) shows the sizes of the inferred FTs (ϕ_s). Here we can clearly observe the influence of minimizing ϕ_s. When accounted, ϕ_s for most of the cases is equal or less than the ground truth, indicated with the horizontal lines for the different case studies (see Appendix C for examples and details). When not accounted for, in some cases, ϕ_s may be significantly larger than the ground truth.

6.4.3 Fault tree complexity

Fig. 8.(d) depicts the convergence time. We observe that in general, for all the m.o.f.'s, sorting the case studies based on the convergence time from the longest to the shortest, we have *ddFT*, *MPPS*, *COVID-19*, *CSD*, *PT*, and *SMS*. This indicates that there is a relationship between the *complexity* of the underlying FT model of a failure data set and the time that takes the algorithm to find it.

We believe this complexity is ruled by the amount of MCSs and their orders (see *O-MCSs* in Table 5). For example, the case study *ddFT* has six MCSs and with orders between 3 and 6, the FT-MOEA generally took the longest time to converge. In contrast, the *SMS* case study has 13 MCSs but all have order 1, here the FT-MOEA converged almost immediately. Thus, the greater the number of MCSs and their orders, the longer will take the algorithm to retrieve the global optimum. We believe that further research is needed to better quantify this type of complexity.

6.4.4 Influence of superfluous variables

Real-world data sets may consider a different number of BEs where possibly not all of them contribute to the failure of the system. In other words, regardless of the state of superfluous BEs, they will not have any effect on the TE. We call these *superfluous variables* (ρ). We assess ρ ranging from 0 to 6 using the MPPS case study with $ps = 400$, $ng = 100$, and $uc = 20$, and the m.o.f.'s sdc and d. The results are presented in Fig 9.

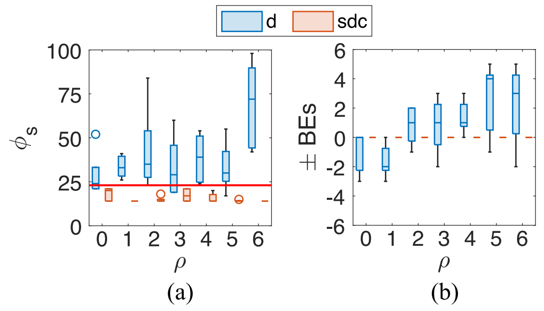

Fig. 9: Influence of *superfluous variables* (ρ) on (a) size of the resulting FT (ϕ_s), (b) additional or missing number of BEs ($\pm$ BEs). Using the m.o.f's sdc and d, and the MPPS case study ($ps = 400$, $ng = 100$, $uc = 20$).

Fig. 9.(a) presents ϕ_s for different values of ρ. Here ϕ_s is smaller than the ground truth (indicated with the dashed horizontal red line) when using the m.o.f. sdc. On the other hand, when using the m.o.f. d, ϕ_s seems to increase for a larger number of ρ.

Fig. 9.(b) shows for different values of ρ the additional or missing number of BEs ($\pm$ BEs). Recall that the case study MPPS has 7 unique BEs. Thus, we subtract this number from the unique number of BEs per each inferred FT. As we can observe, the m.o.f. sdc, despite ρ, it always outputted an FT with 7 BEs (i.e., $\pm$ BEs $= 0$), in other words, the superfluous variables were removed in the optimization process. On the other hand, the m.o.f. d shows inconsistency for different values of ρ, and it seems to perform poorly for larger values of ρ.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

7 Discussion and Conclusions

We demonstrate that it is possible to infer efficient and interpretable FT structures by implementing multi-objective evolutionary algorithms. However, there are several aspects that still need to be addressed before thinking about real-world applications.

Even though our algorithm fares better than alternative approaches, its *scalability* will become an issue for FTs with many BEs. A plausible way to overcome this drawback is by means of *"guided" multi-objective evolutionary algorithms*. This concept consists of helping the optimization process by injecting additional knowledge. One way to address this is by searching out for *patterns* (e.g., Waghen & Ouali [21]) in the failure data set that enables identifying parts of the FT and then assembling them.

Another way is by *guiding* the application of the *genetic operators*. In our current procedure, these are randomly applied, whereas one may be able to increase the chances of finding the global optimum by targeting parts of the FT that most likely need to be modified. The latter might be possible to achieve by implementing Bayesian optimization. A more challenging path is exploring deep learning-based approaches like the one by Cranmer et al. [34] to derive symbolic rules from a Graph Neural Networks.

We found that the inclusion of MCSs in the multi-objective optimization function greatly improves the optimization process. However, the FT-MOEA computes the MCSs based on the disjunctive normal form, which is computationally expensive for large FTs. Moreover, MCSs cannot be computed in noisy data (see some preliminary results regarding the effects of noise in the Appendix G). Thus, alternatives to overcome these issues are worth exploring. Finally, some spare but equally interesting challenges are enlisted next.

- Having noise-free, balanced and complete failure data sets for complex engineering systems is virtually impossible. Thus, a more thorough evaluation of the performance of our algorithm under incomplete, noisy, and unbalanced failure data sets is necessary.
- Real-world problems often contain symmetries, e.g., when two basic events are known to be fully exchangeable. This property could be used as an advantage, because we expect to reduce the solution space, leading to faster convergence. Thus, research in this direction is needed.
- In order to obtain more compact and efficient FT structures, exploring alternatives that enable the inference of more sophisticated gates (e.g., VoT gates) is necessary.
- We believe that a methodology similar to the one used in this paper could be applied for the inference of other reliability models, such as reliability block diagrams, as well as Boolean circuits.
- System identification techniques may be applicable to the inference of FTs, and research in this direction is needed.
- Further research is needed to better understand and quantify the complexity in the inference of FT models. In addition, guidelines and metrics to properly identify the practical capabilities of FT inference algorithms remain an open problem.

Our novel algorithm, the FT-MOEA, has in general a better performance than its predecessor, the FT-EA, by converging faster, inferring more compact FT structures, achieving lower error levels, better removing superfluous variables, and being consistent.

Acknowledgments

Thanks to Matthias Volk for providing us with useful advice. This research has been partially funded by NWO under the grant PrimaVera (https://primavera-project.com) number NWA.1160.18.238.

References

[1] S. Kabir, "An overview of fault tree analysis and its application in model based dependability analysis," *Expert Systems with Applications*, vol. 77, pp. 114–135, 2017.

[2] J.-P. Signoret, A. Leroy *et al.*, "Automated fault tree building," *Springer Series in Reliability Engineering*, pp. 423–426, 2021.

[3] S. L. Salem, G. Apostolakis, and D. Okrent, "Computer-oriented approach to fault-tree construction," California Univ., Tech. Rep., 1976.

[4] A. Hunt, B. Kelly, J. Mullhi, F. Lees, and A. Rushton, "The propagation of faults in process plants: 6, overview of, and modelling for, fault tree synthesis," *Reliability Engineering & System Safety*, vol. 39, no. 2, pp. 173–194, 1993.

[5] M. G. Madden and P. J. Nolan, "Generation of fault trees from simulated incipient fault case data," *WIT Transactions on Information and Communication Technologies*, vol. 6, 1994.

[6] T. Johnson and P. Husbands, "System identification using genetic algorithms," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1990, pp. 85–89.

[7] G. Latif-Shabgahi, "Comparing selected knowledge-based fault tree construction tools," in *Proc. IASTED Int. Conf. Intell. Syst. Control*, 2002.

[8] A. Carpignano and A. Poucet, "Computer assisted fault tree construction: a review of methods and concerns," *Reliability Engineering & System Safety*, vol. 44, no. 3, pp. 265–278, 1994.

[9] F. Mhenni, N. Nguyen, and J.-Y. Choley, "Automatic fault tree generation from sysml system models," in *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 2014, pp. 715–720.

[10] C. E. Dickerson, R. Roslan, and S. Ji, "A formal transformation method for automated fault tree generation from a uml activity model," *IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 1219–1236, 2018.

[11] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[12] M. G. Madden, "Hierarchically structured inductive learning for fault diagnosis," *WIT Transactions on Information and Communication Technologies*, vol. 20, 1998.

[13] M. G. Madden and P. J. Nolan, "Monitoring and diagnosis of multiple incipient faults using fault tree induction," *IEE Proceedings-Control Theory and Applications*, vol. 146, no. 2, pp. 204–212, 1999.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3203805

[14] S. Mukherjee and A. Chakraborty, "Automated fault tree generation: bridging reliability with text mining," in *2007 Annual Reliability and Maintainability Symposium*. IEEE, 2007, pp. 83–88.
[15] M. Roth, M. Wolf, and U. Lindemann, "Integrated matrix-based fault tree generation and evaluation," *Procedia Computer Science*, vol. 44, pp. 599–608, 2015.
[16] J. Li, S. Ma, T. Le, L. Liu, and J. Liu, "Causal decision trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 257–271, 2016.
[17] M. Nauta, D. Bucur, and M. Stoelinga, "Lift: Learning fault trees from observational data," in *International Conference on Quantitative Evaluation of Systems*. Springer, 2018, pp. 306–322.
[18] K. Waghen and M.-S. Ouali, "Interpretable logic tree analysis: A data-driven fault tree methodology for causality analysis," *Expert Systems with Applications*, vol. 136, pp. 376–391, 2019.
[19] A. Linard, M. L. Bueno, D. Bucur, and M. Stoelinga, "Induction of fault trees through bayesian networks," 09 2019.
[20] S. Lazarova-Molnar, P. Niloofar, and G. K. Barta, "Data-driven fault tree modeling for reliability assessment of cyber-physical systems," in *Proceedings of the 2020 Winter Simulation Conference*, 2020.
[21] K. Waghen and M.-S. Ouali, "Multi-level interpretable logic tree analysis: A data-driven approach for hierarchical causality analysis," *Expert Systems with Applications*, vol. 178, p. 115035, 2021.
[22] A. Linard, D. Bucur, and M. Stoelinga, "Fault trees from data: Efficient learning with an evolutionary algorithm," in *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*. Springer, 2019, pp. 19–37.
[23] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools," *Computer science review*, vol. 15, pp. 29–62, 2015.
[24] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback, "Fault tree handbook with aerospace applications," 2002.
[25] M. Ojha, K. P. Singh, P. Chakraborty, and S. Verma, "A review of multi-objective optimisation and decision making using evolutionary algorithms," *International Journal of Bio-Inspired Computation*, vol. 14, no. 2, pp. 69–84, 2019.
[26] K. Deb, "Multi-objective optimisation using evolutionary algorithms: an introduction," in *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, 2011, pp. 3–34.
[27] ——, "Multi-objective optimization," in *Search methodologies*. Springer, 2014, pp. 403–449.
[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.
[29] Q. Long, X. Wu, and C. Wu, "Non-dominated sorting methods for multi-objective optimization: Review and numerical comparison," *Journal of Industrial & Management Optimization*, vol. 17, no. 2, p. 1001, 2021.
[30] L. Marti, E. Segredo, E. Hart *et al.*, "Impact of selection methods on the diversity of many-objective pareto set approximations," *Procedia computer science*, vol. 112, pp. 844–853, 2017.
[31] P. Robert and Y. Escoufier, "A unifying tool for linear multivariate statistical methods: The rv-coefficient," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 25, no. 3, pp. 257–265, 1976.
[32] T. Bakeli, A. A. Hafidi *et al.*, "Covid-19 infection risk management during construction activities: An approach based on fault tree analysis (fta)," *Journal of Emergency Management*, vol. 18, no. 7, pp. 161–176, 2020.
[33] A. Mentes and I. H. Helvacioglu, "An application of fuzzy fault tree analysis for spread mooring systems," *Ocean Engineering*, vol. 38, no. 2-3, pp. 285–294, 2011.
[34] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, "Discovering symbolic models from deep learning with inductive biases," *arXiv preprint arXiv:2006.11287*, 2020.

Lisandro A. Jimenez-Roa is a doctorate candidate in Computer Science at the University of Twente, The Netherlands. His background is in civil engineering and has worked on a variety of projects in the fields of structural health monitoring, finite element modeling, and damage detection via data analytics and machine learning. Currently, he is conducting research on system-level prognostics of complex engineering systems within the PrimaVera project (https://primavera-project.com), with a particular interest in hybrid integration of domain expertise, physics-informed, and data-driven approaches.

Tom Heskes is full professor of Artificial Intelligence. After receiving his Ph.D. on neural networks, he worked as a postdoc at the Beckman Institute in Champaign-Urbana, Illinois. Back in the Netherlands, he joined SNN, the Foundation for Neural Networks, and later the Institute for Computing and Information Sciences at Radboud University. Tom Heskes is the former Editor-in-Chief of Neurocomputing and has been (co-)leading various national and European projects. Heskes' research concerns the development, understanding, and application of machine learning methods, currently particular deep learning and causal inference. He works on applications in other scientific disciplines, as well as in industry, among others through his spin-off company Machine2Learn.

Tiedo Tinga is a full professor in dynamics based maintenance at the University of Twente since 2012 and full professor of Life Cycle Management at the Netherlands Defence Academy since 2016. He received his Ph.D. degree in mechanics of materials from Eindhoven University in 2009. He is chairing the smart maintenance knowledge center and leads a number of research projects on developing predictive maintenance concepts, mainly based on the physics of failure models, but also following data-driven approaches.

Mariëlle Stoelinga is a full professor of risk analysis for high-tech systems, both at the University of Twente and Radboud University, the Netherlands. She holds a Master's degree in Mathematics & Computer Science, and a Ph.D. in Computer Science. After her Ph.D., she has been a postdoctoral researcher at the University of California at Santa Cruz, USA. Prof. Stoelinga leads the executive Master of Risk Management at the University of Twente, a part-time program for risk professionals. She also leads various research projects, including a large national consortium on Predictive Maintenance and an ERC consolidator grant on safety and security interactions.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF TWENTE.. Downloaded on March 27,2023 at 13:48:06 UTC from IEEE Xplore. Restrictions apply.