Expert Systems With Applications 215 (2023) 119345

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# Data-driven extraction and analysis of repairable fault trees from time series data

Parisa Niloofar [a,*], Sanja Lazarova-Molnar [b,a]

[a] Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Campusvej 55, Odense 5230, Denmark
[b] Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, Kaiserstr. 89, Karlsruhe 76133, Germany

ARTICLE INFO

*Keywords:*
Classification
Data-driven simulation
Fault tree analysis
Multi-state system
Proxel-based simulation
Reliability analysis

ABSTRACT

Fault tree analysis is a probability-based technique for estimating the risk of an undesired top event, typically a system failure. Traditionally, building a fault tree requires involvement of knowledgeable experts from different fields, relevant for the system under study. Nowadays' systems, however, integrate numerous Internet of Things (IoT) devices and are able to generate large amounts of data that can be utilized to extract fault trees that reflect the true fault-related behavior of the corresponding systems. This is especially relevant as systems typically change their behaviors during their lifetimes, rendering initial fault trees obsolete. For this reason, we are interested in extracting fault trees from data that is generated from systems during their lifetimes. We present DDFTAnb algorithm for learning fault trees of systems using time series data from observed faults, enhanced with Naïve Bayes classifiers for estimating the future fault-related behavior of the system for unobserved combinations of basic events, where the state of the top event is unknown. Our proposed algorithm extracts repairable fault trees from multinomial time series data, classifies the top event for the unseen combinations of basic events, and then uses proxel-based simulation to estimate the system's reliability. We, furthermore, assess the sensitivity of our algorithm to different percentages of data availabilities. Results indicate DDFTAnb's high performance for low levels of data availability, however, when there are sufficient or high amounts of data, there is no need for classifying the top event.

## 1. Introduction

Fault Tree Analysis (FTA) is a prominent method in analysing safety and reliability of systems (Vesely et al., 1981; Lee et al., 1985; Ruijters and Stoelinga, 2015). While in most of the real-world cases, it is necessary to consider both failures and repairs for components of a system, traditional fault trees do not consider repairable components. Repairable fault trees address this issue and consider information not only about failure times of basic components, but also about maintenance or repairs within a system.

Multi-state fault trees have the same structure as regular fault trees, but the components or the system may have more than two functioning levels. If the system and its components, either completely function or fail, reliability analysis for this system has a binary perspective. Nonetheless, there are systems that operate at various levels of performance, which usually yields more than two states associated with basic events (Lisnianski and Levitin, 2003). Studies have been dedicated to analyse these types of systems (Compare et al., 2017; Barlow and Heidtmann, 1984; Nadjafi et al., 2017; Caldarola, 1980).

Many extensions of fault trees have been proposed in the literature, each having their own variety of shortcomings and assumptions. However, even with the emerging availability of data through Internet of Thing (IoT) devices and all existing software tools, yet fault tree analysis requires a lot of manual effort and expert knowledge. Hence, the possibility to use data-driven methods to extract information about the status of a system under study has not yet been fully explored. Data-driven approaches are gaining attraction in many areas for their ability to analyse data from a system to derive the system's behaviour (Huang et al., 2021; Solomatine and Ostfeld, 2008). Big data are nowadays collected in a large portion of manufacturing systems, especially in non-safety–critical systems, where faults are more common occurrence and do not have associated catastrophic consequences. Data-driven fault detection, diagnosis or prediction are well-studied using machine learning and data mining methods (Dogan and Birant, 2021; Ayvaz and Alpay, 2021). However, completely ignoring human cognitive capabilities and expert knowledge causes a great loss of information, which might only be compensated by collecting large amounts of

* Corresponding author.
*E-mail addresses:* parni@mmmi.sdu.dk (P. Niloofar), sanja.lazarova-molnar@kit.edu (S. Lazarova-Molnar).

https://doi.org/10.1016/j.eswa.2022.119345
Received 2 December 2021; Received in revised form 16 September 2022; Accepted 22 November 2022
Available online 26 November 2022

**Nomenclature**

| | |
|---|---|
| AADL | Architecture Analysis & Design Language |
| ACC | Accuracy |
| AltaRica | Altarica Language and Its Semantics |
| BE | Basic Event |
| DAG | Directed Acyclic Graph |
| DDFTA | Data Driven Fault Tree Analysis |
| DDFTAnb | Data Driven Fault Tree Analysis enhanced with Naïve Bayes classifier |
| D*ij* | Disk number *ij* |
| FN | False Negative |
| FP | False Positive |
| FTA | Fault Tree Analysis |
| f/h | failures per hour |
| HiP-HOPS | Hierarchically Performed Hazard Origin & Propagation Studies |
| IE | Intermediate Event |
| IFT | Induction of Fault Trees |
| ILTA | Interpretable Logic Tree Analysis |
| IoT | Internet of Things |
| LIFT | Learning Fault Trees from observational data |
| MAP | Maximum a Posteriori |
| MBDA | Model Based Dependability Analysis |
| MCS | Minimal Cut Sets |
| M*i* | Memory number *i* |
| MILTA | Multi-Level Interpretable Logic Tree Analysis |
| MP | Multiprocessor |
| MTTF | Mean Time to Failure |
| MTTR | Mean Time to Repair |
| NB | Naïve Bayes |
| P*i* | Processor number *i* |
| Proxel | Probability Elements |
| PS | Power Supply |
| RBC | Radio Block Center |
| RMSE | Root Mean Square Error |
| SHyFTA | Stochastic Hybrid Fault Tree Automaton |
| T | Total time |
| TE | Top Event |
| TN | True Negative |
| TP | True Positive |
| $U_i$ | Unavailability at time step $i$ |
| $\widehat{U}_i$ | Estimated unavailability at time step $i$ |
| $\Delta t$ | Size of a time step |

data that is costly in many aspects. Hence, a systematic method for fusion of data and expert knowledge can increase the accuracy in reliability analysis of a system (Niloofar and Lazarova-Molnar, 2021).

Models extracted from observational data are often used to predict future behaviours of systems under study for unseen inputs of the models. When it comes to learning/extracting fault tree models from fault records of the components in a system, one also needs to know how unseen events/failures of components or combinations of events/failures of components impact the overall system, i.e., whether they lead to system failures or not. To address the issue of unavailability of data for unseen events or combinations of events, we present DDFTAnb that adds a classification functionality to the DDFTA algorithm introduced by Lazarova-Molnar et al. (2020), in an attempt to forecast system's behavior for unobserved combinations of basic events. Our proposed extended algorithm, DDFTAnb, first extracts repairable multi-state fault trees from observational multinomial time series data, then analyses the results to estimate reliability and maintainability distributions of basic events, and finally estimates the future behaviour of the system for the unobserved occurrences of combinations of basic events. Data-driven modelling can detect hidden causes of a system failure which are not evident utilizing solely expert knowledge. However, expert knowledge can be deployed as a prior information into the model and can also be supplemented for model validation. The highlights of the paper are:

1. Learning structures of repairable multi-state fault trees using only time series data from faults and other relevant basic events that contribute to a system failure
2. Working with reliability and maintainability distributions other than exponential
3. Estimating future fault-related behaviours of systems in terms of fault tree structures and systems' reliabilities
4. Simulating fault trees using proxel-based simulation, which is especially well suited for simulating complex stochastic dependencies and different probability distributions.

The rest of the paper is organized as follows. Section 2 presents the literature review. Section 3 provides the methodology where DDFTAnb with its classification module are described in details. In Section 4, two case studies are demonstrated, and finally, in Section 5, we conclude the paper.

## 2. Literature review

Classic FTA is primarily knowledge-driven, rather than data-driven. This can limit the capacity of the generated fault trees in depicting the true fault-related behaviours of the corresponding systems, especially as systems often evolve and change their behaviours during their lifetimes, rendering initial fault trees obsolete. Model building based on experts' knowledge alone is becoming outdated with evolving systems designs, data collection technologies, and blockchain-based data storage and access frameworks.

Automation of extracting dependability information from system models has led to the field of model-based dependability analysis (MBDA) (Sharvia et al., 2016; Aizpurua and Muxika, 2013; Kabir, 2017). Different tools and techniques have been developed as part of MBDA to automate the generation of dependability analysis artefacts such as fault trees (Papadopoulos and McDermid, 1999; Feiler et al., 2006; Arnold et al., 2000). Besides model-based approaches, statistical and artificial intelligence methods are another solution for automating or semi-automating the extraction of dependability information from systems (Jardine et al., 2006; Zhang et al., 2017).

While model-based methodologies need information about the physical characteristics of the system for the establishment of an explicit mathematical model, statistical models use historical data to represent and predict the future behaviour of a system. In addition, artificial intelligence techniques are suitable for addressing the complex and large-scale nonlinear problems that mostly requires no statistical assumptions about the data. Neural computation, evolutionary algorithms, and fuzzy computing as different categorizations of computational intelligence (which is a branch of artificial intelligence) have been applied for fault detection and classification (Chen et al., 2008; Zheng et al., 2017; Theodoropoulos et al., 2021; Brito et al., 2022). The applications of (explainable) artificial intelligence techniques for fault diagnosis and machinery monitoring is a subject based on the theory of signal processing and pattern recognition, and these techniques are mostly used for estimating the remaining useful life (Sikorska et al., 2011; Ayvaz and Alpay, 2021). However, some researchers combine artificial intelligence and statistical approaches with FTA, which we note in the following.

Lampis and Andrews (2009) applied Bayesian Belief networks to diagnose faults in a system. They first constructed fault trees to indicate how the component failures can combine to cause unexpected deviations in the variables monitored by the sensors, and then converted these fault trees into Bayesian networks for further analysis. Cai et al. (2015) addressed a case study of subsea pipe ram BOP system by proposing a new method for real-time reliability analysis through a combination of traditional and dynamic Bayesian networks. In this study, prior reliability knowledge of the system (failure distributions) is updated via dynamic Bayesian networks. In FTA, basic components are assumed to be independent and this is a strong assumption for some dynamic systems. Guo et al. (2021) proposed a reliability analysis model for dynamic systems with common cause failures based on discrete-time Bayesian networks. They applied their model for fault diagnosis of a digital safety-level distributed control system of nuclear power plants. These studies do not apply observational/historical data to build or to learn the fault tree structure, but some researchers use data to update or estimate the failure rates (Cai et al., 2015).

Observational data were used to generate FTs with the IFT (Induction of Fault Trees) algorithm (Nolan et al., 1994) based on standard decision-tree statistical learning. Later, Liggesmeyer and Rothfelder (1998) coined the term formal risk analysis and developed an approach for automatically generating a fault tree from finite state machine-based descriptions of a system where the generated fault tree is complete with respect to all failures assumed possible. Mukherjee and Chakraborty (2007) describe a technique to automatically generate fault trees using historical maintenance data in text form. Their technique relies on domain knowledge and linguistic analysis. Majdara and Wakabayashi (2009) represent a new system of modelling approach, composed of some components and different types of flows propagating through them, for computer-aided fault tree generation. Chiacchio et al. (2016) combined the Dynamic Fault Tree technique and the Stochastic Hybrid Automaton within the Simulink environment that represented an important step ahead for the delivering of a user-friendly computer-aided tool for the dynamic reliability. They also developed a library called Stochastic Hybrid Fault Tree Automaton (SHyFTA) that allows the accurate dependability analysis of repairable multi-state systems (Chiacchio et al., 2020). Nauta et al. (2018) introduced LIFT (Learning Fault Trees from observational data) to learn structures of static fault trees from untimed data bases with Boolean event variables, however, their method needs information about intermediate events. Linard et al. (2019) applied an evolutionary algorithm to learn fault trees from untimed Boolean basic event variables. Instead of the independence test in the LIFT algorithm, they used a score-based algorithm to extract fault trees. Furthermore, Waghen and Ouali (2019) proposes interpretable logic tree analysis (ILTA), which characterizes and quantifies event causality occurring in engineering systems with the minimum involvement of human experts. Their method is an integration of two concepts: knowledge discovery in database and fault tree analysis, which was improved to a multi-level interpretable logic tree (MILTA) (Waghen and Ouali, 2021). Qian et al. (2021), for the first time, applied association rule analysis to extract fault trees from overhead contact system of an electrified railway. They first transform the failure records of overhead contact system into transaction database, and then the extracted association rules from the data are converted to a fault tree. Lazarova-Molnar et al. (2020) introduce DDFTA algorithm that uses time series data of faults to extract repairable multi-state fault tree of a system.

The above-mentioned techniques have different requirements; however, except for the work of Lazarova-Molnar et al. (2020), they cannot extract reliability models from time series data recorded from multi-state/repairable systems. Also, labelling the top event is not studied in the literature. Time series data of a system consists of a sequence of status change times for each basic event and the system state. In this study, we follow the work of Lazarova-Molnar et al. (2020) and add a classification module so that the algorithm does not only extract repairable multi-state fault trees from observational data, but also makes predictions on the future reliability state of the system. Being able to label the system state (classify the top event), becomes more important when the system contains rare events (or components with rare failures), which is the case for safety critical systems, or for systems composed of so many components that observing all the possibilities becomes unfeasible and non-realistic.

In the following, we provide background on the relevant concepts and methods that we refer to in this paper, i.e., repairable multi-state fault trees, Naïve Bayes classifier and proxel-based simulation.

### 2.1. Repairable multi-state fault trees

A fault tree is a Directed Acyclic Graph (DAG) whose leaves are the basic events (typically basic faults), and the root represents the top event, which is typically a system failure. The gates in a fault tree represent the propagation of failure through the tree (Ruijters and Stoelinga, 2015). Multi-state fault trees have the same structure of regular fault trees, except that the components or the system may have more than two functioning levels. In other words, the state space of the system and its components may be represented by $\{0, 1, \cdots, M\}$, where 0 indicates a completely failed state, $M$ indicates a perfectly working state, and the others are degraded states. Repairable fault trees consider both faults and repairs within a system. Hence, for each basic event that is typically associated with a fault, there are probability distributions that describe the fault's occurrences and repair times.

There are two essential analysis techniques for fault trees, qualitative analysis, and quantitative analysis. Qualitative analysis considers the structure of the fault tree, while the quantitative analysis computes failure probabilities, reliability, etc. of the system represented by the fault tree. The first step towards computing reliability of a system is to extract the structure of the system's underlying fault tree. When the structure of the fault tree is extracted, using the probability distribution functions of the basic events, we can calculate the reliability of the system, the likelihood of a top event occurrence, as well as those of the basic events that have caused the occurrence of the top event. The results of quantitative analysis give analysts an indication about system reliability and also help to determine which components or parts of the system are more critical so analysts can put more emphasis on the critical components or parts by taking necessary steps, e.g., including redundant components in the system model (Kabir, 2017).

### 2.2. Naïve Bayes classifier

The Naïve Bayes (NB) classifier is a probability-based supervised learning classification method which is well studied in the literature. NB is among the simplest Bayesian Network models and has received much attention due to its simple classification model and excellent classification performance. An early description can be found in Duda and Hart (1973). Domingos and Pazzani (1996) discuss its feature independence assumption and explain why Naïve Bayes performs well for classification even with such an over-simplified assumption.

In this paper, we apply NB to classify the state of the system for the unobserved combinations of basic events. Basic events, which are always considered independent, are the features in NB, and the top event is the class variable. We use observed data from fault occurrences related to basic events as a training set to fit the NB model, and we use the unobserved combinations of basic events as a testing set. NB first calculates the posterior for the top event and then applies the maximum a posteriori (MAP) decision rule: the label is the class with the maximum posterior. Those combinations of basic events in the testing set for which the top event is classified as "failed", along with those in the training set, where the state of the top event is "failed", are considered cut sets. These predicted cut sets are used to extract minimal cut sets that will construct the predicted behaviour of the system in terms of a fault tree.

Challenging point in data-driven modelling of faults for classification

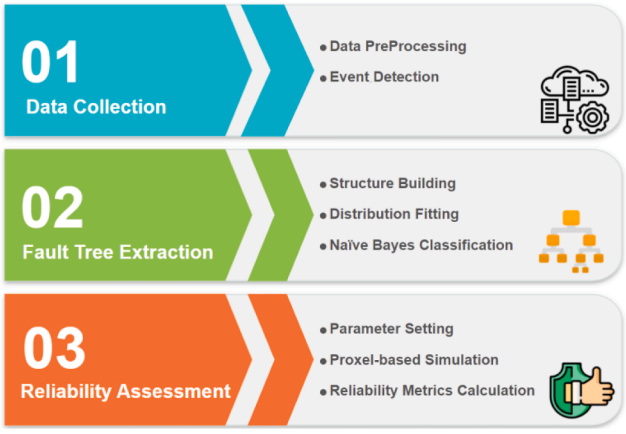



**Fig. 1.** Overall framework of DDFTAnb algorithm with the classification module.

tasks is the imbalanced proportion of classes as faults are rarely observed, especially for highly reliable systems. Hence, we are troubled with an imbalanced classification where one class of the dependent (response) variable (here, working state) outnumbers the other class (failed state) by a large proportion. There are many ways to combat this issue, where the very best is to accumulate more data. This, however, is not possible in our case. Another approach is to manually balance the classes. One common method of doing this is to upsample/oversample the minority class or undersample the majority class using resampling (bootstrapping) techniques. In this study we upsample the faulty state to balance the classes and apply bootstrapping techniques.

### 2.3. Proxel-based simulation

Proxel-based simulation is a state space-based simulation method to compute transient solutions for discrete stochastic systems. It relies on a user-definable discrete time step and computes the probability of all possible single state changes (and the case that no change happens at all) during a time step. The target states along with their probabilities are stored as so-called proxels (short for probability elements). To account

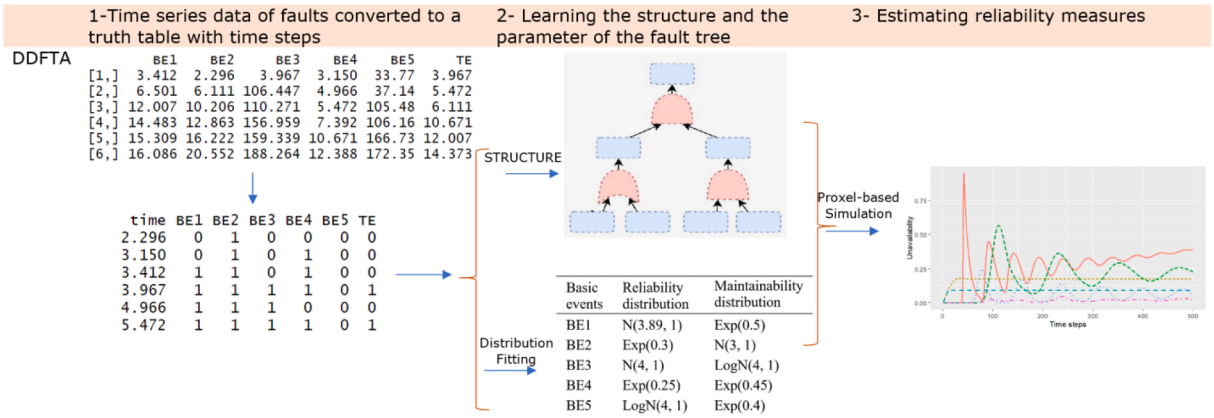



**Fig. 2.** The process workflow of the DDFTA algorithm.

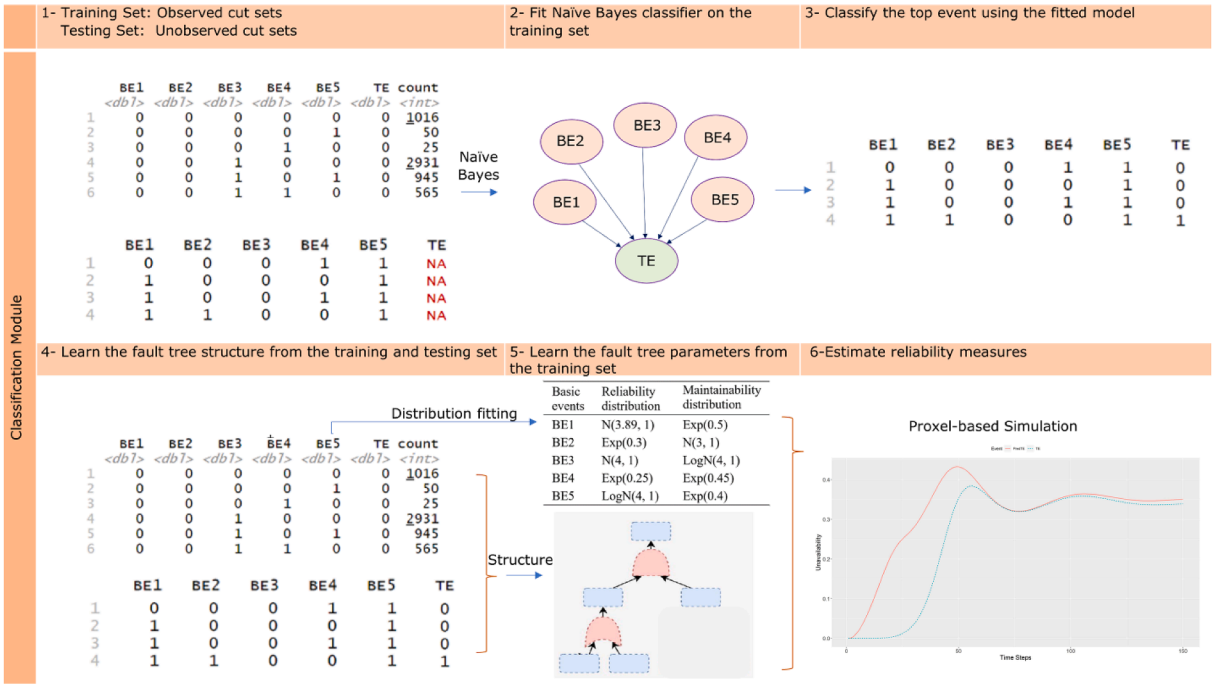



**Fig. 3.** Workflow of classification module for DDFTAnb algorithm.

**Table 1**
Time series data of faults converted into a truth table.

| Time | BE1 | BE2 | BE3 | BE4 | BE5 | TE |
|---|---|---|---|---|---|---|
| 17.96968 | 0 | 0 | 1 | 1 | 0 | 0 |
| 18.63438 | 0 | 1 | 1 | 1 | 0 | 1 |
| 20.1585 | 0 | 1 | 1 | 0 | 0 | 0 |
| 21.11844 | 1 | 1 | 1 | 0 | 0 | 0 |
| 21.52825 | 0 | 1 | 1 | 0 | 0 | 0 |
| 22.12907 | 0 | 0 | 1 | 0 | 0 | 0 |
| 23.07983 | 0 | 0 | 1 | 1 | 0 | 0 |
| 24.67361 | 0 | 0 | 1 | 0 | 0 | 0 |
| 24.74219 | 1 | 0 | 1 | 0 | 0 | 0 |
| 25.01376 | 1 | 0 | 1 | 1 | 0 | 1 |

**Table 2**
Sets of cut sets and minimal cut sets for the truth table data of Table 1.

| Cut sets | Minimal cut sets |
|---|---|
| {BE2, BE3, BE4} | {BE2, BE3, BE4} |
| {BE1, BE3, BE4} | {BE1, BE3, BE4} |

**Table 3**
Constructing the fault tree based on the minimal cut sets of Table 2.

| Step | Boolean representation |
|---|---|
| 1 | TE=(BE1.BE3.BE4) +(BE2.BE3.BE4) |
| 2 | TE=(BE1 + BE2). (BE3.BE4) |
| 3 | TE = IE1.IE2 |

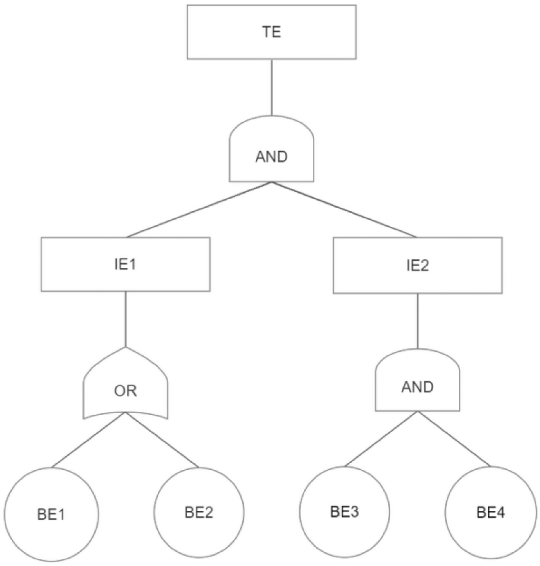



**Fig. 4.** Fault tree constructed from data of Table 1.

for aging (i.e., non-Markovian) transitions, proxels contain supplementary variables that keep track of the ages of all active and all race-age transitions. For each proxel created, the algorithm iteratively computes all successors for each time step. This results in a tree of proxels where all proxels having the same distance from the tree root belong to the same time step and all leaf proxels represent the possible states being reached at the end of the simulation.

Proxel-based simulation explores all possible future developments of the system each with a determined computable probability, based on the distribution functions which describe the events, as well as the time they have been pending, in discrete time steps. It determines all possible follow-up states and the rendering probability of the corresponding state transitions. The proxel-based simulation is well-known for its ability to cope with stiff models, as fault models are typically (Lazarova-Molnar and Horton, 2003; Lazarova-Molnar, 2005).

**Table 4**
Truth table with a multi-state event BE1 (up) turned into a truth table with binary events (down).

| Time | BE1 | BE2 | BE3 | TE |
|---|---|---|---|---|
| 17.96968 | 0 | 0 | 1 | 0 |
| 18.63438 | 0 | 1 | 1 | 1 |
| 20.1585 | 2 | 1 | 1 | 1 |
| 21.11844 | 1 | 1 | 1 | 0 |
| 21.52825 | 0 | 1 | 1 | 0 |
| 22.12907 | 0 | 0 | 1 | 0 |
| 23.07983 | 2 | 0 | 1 | 0 |
| 24.67361 | 0 | 0 | 1 | 0 |
| 24.74219 | 1 | 0 | 1 | 0 |
| 25.01376 | 1 | 0 | 1 | 1 |

| Time | BE1_1 | BE1_2 | BE2 | BE3 | TE |
|---|---|---|---|---|---|
| 17.96968 | 0 | 0 | 0 | 1 | 0 |
| 18.63438 | 0 | 0 | 1 | 1 | 1 |
| 20.1585 | 0 | 1 | 1 | 1 | 1 |
| 21.11844 | 1 | 0 | 1 | 1 | 0 |
| 21.52825 | 0 | 0 | 1 | 1 | 0 |
| 22.12907 | 0 | 0 | 0 | 1 | 0 |
| 23.07983 | 0 | 1 | 0 | 1 | 0 |
| 24.67361 | 0 | 0 | 0 | 1 | 0 |
| 24.74219 | 1 | 0 | 0 | 1 | 0 |
| 25.01376 | 1 | 0 | 0 | 1 | 1 |

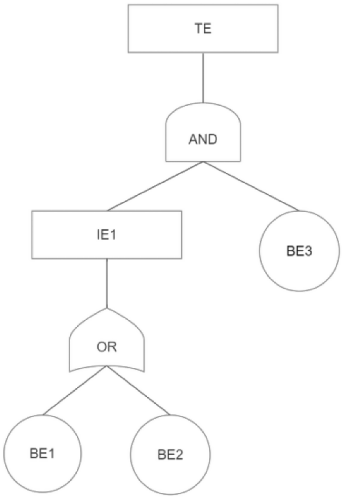



**Fig. 5.** Multi-state fault tree extracted from the data in Table 4.

## 3. Methodology

In this section, we describe the methods and techniques that we developed to enable the data-driven reliability modelling and analysis to extract repairable multi-state fault trees from observational data and to estimate the future reliability state of the system. The overall framework that describes the high-level workflow of DDFTAnb algorithm is shown in Fig. 1, and more detailed workflows are illustrated in Fig. 2 and Fig. 3.

DDFTA, as illustrated in Fig. 2, comprises of three steps (Lazarova-Molnar et al., 2020): 1) converting time series data of faults into a truth table with time steps, 2) structure learning and parameter learning of the fault tree, and 3) estimating reliability measures. To learn the structure of the fault tree, we extract the minimal cut sets (MCS) from the time

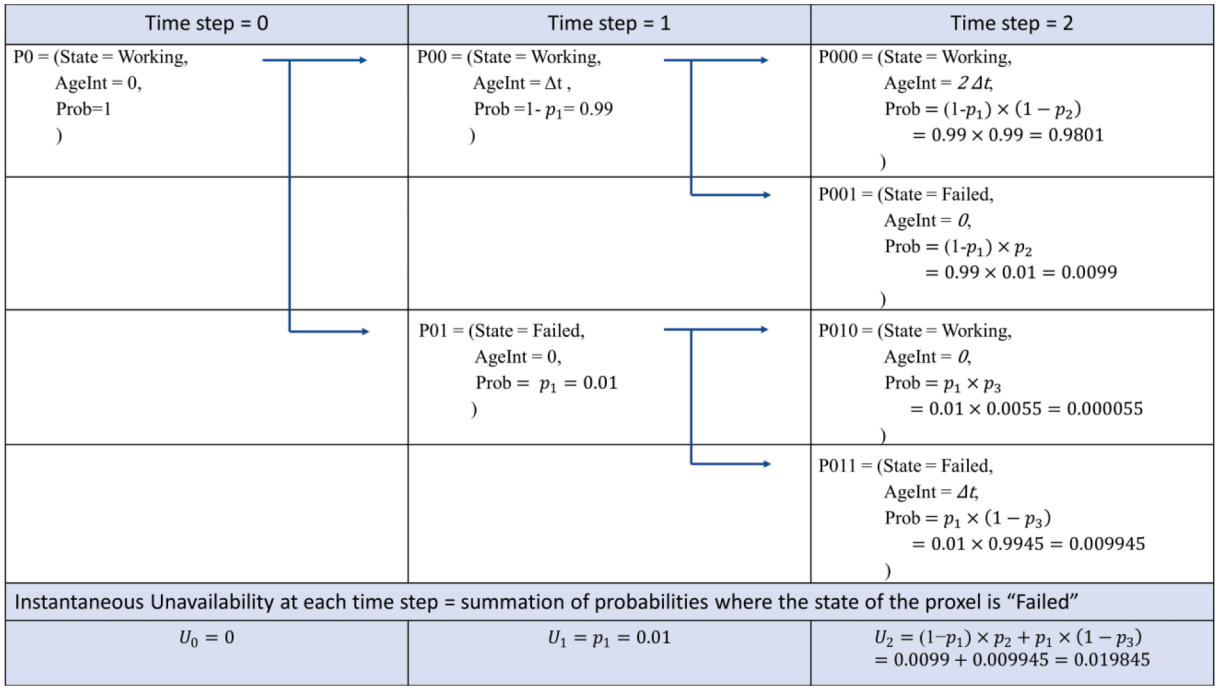



**Fig. 6.** First three time steps of proxel simulation.

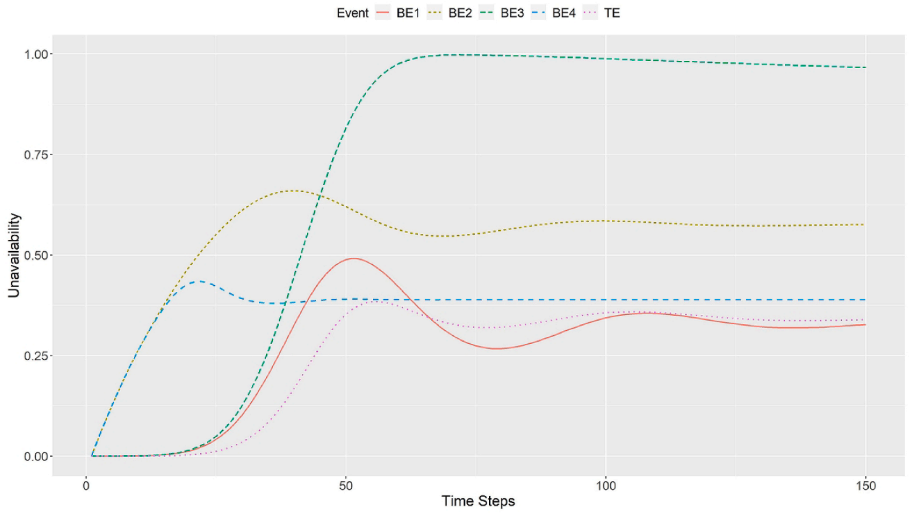



**Fig. 7.** Unavailabilities of basic events along with the top event for the fault tree from Fig. 4.

series data set, and then use Boolean algebra to build a fault tree that is aimed to be mathematically identical to the true fault tree of the system. For parameter learning, reliability and repair distribution functions of the basic events, along with the fault tree structure, are inputs to the proxel-based simulation in the final step, which is used to calculate the system's reliability measures, in form of complete transient solutions. The classification module of DDFTAnb algorithm is described in the Section 3.1.

### 3.1. Classification module for DDFTAnb algorithm

The basic DDFTA approach performs reliability analysis based on observed components' faults. DDFTA, however, does not provide a robust solution for very rare events or cases of small amounts of data with low resolution, where not all possible combinations of basic events have occurred and the corresponding top event statuses are unknown. The DDFTA approach (Fig. 2) begins by converting time series data of faults to a truth table with time steps. The next steps are structure learning and parameter learning, and the final step is estimating reliability measures. In this section, the classification module of the advanced DDFTAnb approach, as illustrated in Fig. 3, is described in detail.

The classification module for the DDFTAnb algorithm consists of six steps: 1) dividing the truth table with time steps into training set and testing sets, 2) fitting Naïve Bayes classifier to the training set, 3) classification of the top event for the testing set using the fitted Naïve Bayes model, 4) learning the fault tree structure from the combination of training and testing data set, 5) learning the fault tree parameters from the training set, and finally 6) estimate reliability measures. DDFTAnb with its classification module are better explained through an illustrative example in the next section.

**Table 5**
The power set for the five binary basic events and the number of occurrences in parentheses.

| | BE1 | BE2 | BE3 | BE4 | BE5 | TE (#) | | BE1 | BE2 | BE3 | BE4 | BE5 | TE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 0 | 1 | 0 | 0 | 0 (1) | **17** | 1 | 0 | 0 | 0 | 0 | NA |
| **2** | 0 | 0 | 1 | 0 | 0 | 0 (2) | **18** | 1 | 0 | 0 | 0 | 1 | NA |
| **3** | 0 | 0 | 1 | 1 | 0 | 0 (2) | **19** | 1 | 0 | 0 | 1 | 0 | NA |
| **4** | 0 | 1 | 1 | 0 | 0 | 0 (2) | **20** | 1 | 0 | 0 | 1 | 1 | NA |
| **5** | 0 | 1 | 1 | 1 | 0 | 1 (1) | **21** | 0 | 0 | 0 | 1 | 0 | NA |
| **6** | 1 | 1 | 1 | 0 | 0 | 0 (1) | **22** | 1 | 0 | 1 | 0 | 1 | NA |
| **7** | 1 | 0 | 1 | 1 | 0 | 1 (1) | **23** | 0 | 1 | 0 | 0 | 0 | NA |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | **24** | 0 | 1 | 0 | 0 | 1 | NA |
| **9** | 1 | 1 | 1 | 1 | 1 | 1 | **25** | 1 | 1 | 0 | 0 | 0 | NA |
| **10** | 1 | 0 | 1 | 1 | 1 | 1 | **26** | 1 | 1 | 0 | 0 | 1 | NA |
| **11** | 1 | 1 | 1 | 1 | 0 | 1 | **27** | 1 | 1 | 0 | 1 | 0 | NA |
| **12** | 0 | 1 | 1 | 1 | 1 | 1 | **28** | 1 | 1 | 0 | 1 | 1 | NA |
| **13** | 0 | 0 | 0 | 1 | 1 | NA | **29** | 0 | 0 | 1 | 1 | 1 | NA |
| **14** | 0 | 0 | 0 | 0 | 1 | NA | **30** | 1 | 1 | 1 | 0 | 1 | NA |
| **15** | 0 | 1 | 0 | 1 | 0 | NA | **31** | 0 | 0 | 1 | 0 | 1 | NA |
| **16** | 0 | 1 | 0 | 1 | 1 | NA | **32** | 0 | 1 | 1 | 0 | 1 | NA |

**Table 6**
Classification results for the top event using Naïve Bayes classifier.

| | BE1 | BE2 | BE3 | BE4 | BE5 | Classified TE |
|---|---|---|---|---|---|---|
| 13 | 0 | 0 | 0 | 1 | 1 | 0 |
| 14 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 1 | 0 | 1 | 0 | 1 |
| 16 | 0 | 1 | 0 | 1 | 1 | 1 |
| 17 | 1 | 0 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 1 | 0 |
| 19 | 1 | 0 | 0 | 1 | 0 | 1 |
| 20 | 1 | 0 | 0 | 1 | 1 | 1 |
| 21 | 0 | 0 | 0 | 1 | 0 | 0 |
| 22 | 1 | 0 | 1 | 0 | 1 | 0 |
| 23 | 0 | 1 | 0 | 0 | 0 | 0 |
| 24 | 0 | 1 | 0 | 0 | 1 | 0 |
| 25 | 1 | 1 | 0 | 0 | 0 | 0 |
| 26 | 1 | 1 | 0 | 0 | 1 | 0 |
| 27 | 1 | 1 | 0 | 1 | 0 | 1 |
| 28 | 1 | 1 | 0 | 1 | 1 | 1 |
| 29 | 0 | 0 | 1 | 1 | 1 | 0 |
| 30 | 1 | 1 | 1 | 0 | 1 | 0 |
| 31 | 0 | 0 | 1 | 0 | 1 | 0 |
| 32 | 0 | 1 | 1 | 0 | 1 | 0 |

**Table 7**
Updated sets of cut sets and minimal cut sets based on the Classifications in Table 6.

| Cut sets | Minimal cut sets |
|---|---|
| {BE2, BE3, BE4} | {BE2, BE4} |
| {BE1, BE3, BE4} | {BE1, BE4} |
| {BE2, BE4} | |
| {BE2, BE4, BE5} | |
| {BE1, BE4} | |
| {BE1, BE4, BE5} | |
| {BE1, BE2, BE4} | |
| {BE1, BE2, BE4, BE5} | |

**Table 8**
Boolean representation of the fault tree based on the minimal cut sets of Table 7.

| Step | Boolean representation |
|---|---|
| 1 | TE=(BE1.BE4) +(BE2.BE4) |
| 2 | TE=(BE1 + BE2). (BE4) |
| 3 | TE = IE1. BE4 |

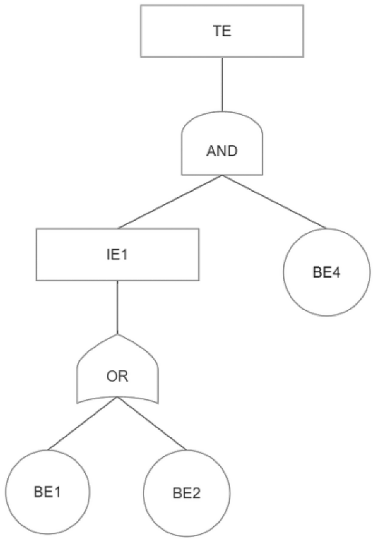



**Fig. 8.** Extracted fault tree based on the minimal cut sets of Table 7.

### 3.2. *Illustrative example*

Assume that time series data on faults for a system with five basic components (BEi, i = 1, 2, …, 5) and the top event (TE) are collected until a specific point in time. The goal is to use these observed time series data to assess the current reliability of the system and estimate the future structure of the system's fault tree as well as its reliability measures. For simplicity and without loss of generality, we assume that the observed data contain 10 records as in Table 1.

#### 3.2.1. *DDFTA algorithm*

According to DDFTA, the first step is to convert the time series data of faults to a truth table with time stamp. Table 1 shows the time-stamped truth table of the collected data, where 0 indicates working state and 1 shows failure state.

**Structure Learning**: To build the structure of the fault tree, we need to extract the minimal cut sets. Those rows in Table 1, where the system is failed (TE has label 1) indicate the cut sets (Table 2), and since they cannot be reduced to smaller cut sets, they are also minimal cut sets. These minimal cut sets build the structure of the fault tree (Table 3), which is also shown in Fig. 4.

To extract the minimal cut sets of a multi-state fault tree, the multi-state events with *m* (>2) number of states, are converted into *m-1* binary events. Assume a system with three basic events {BE1, BE2, BE3}, in

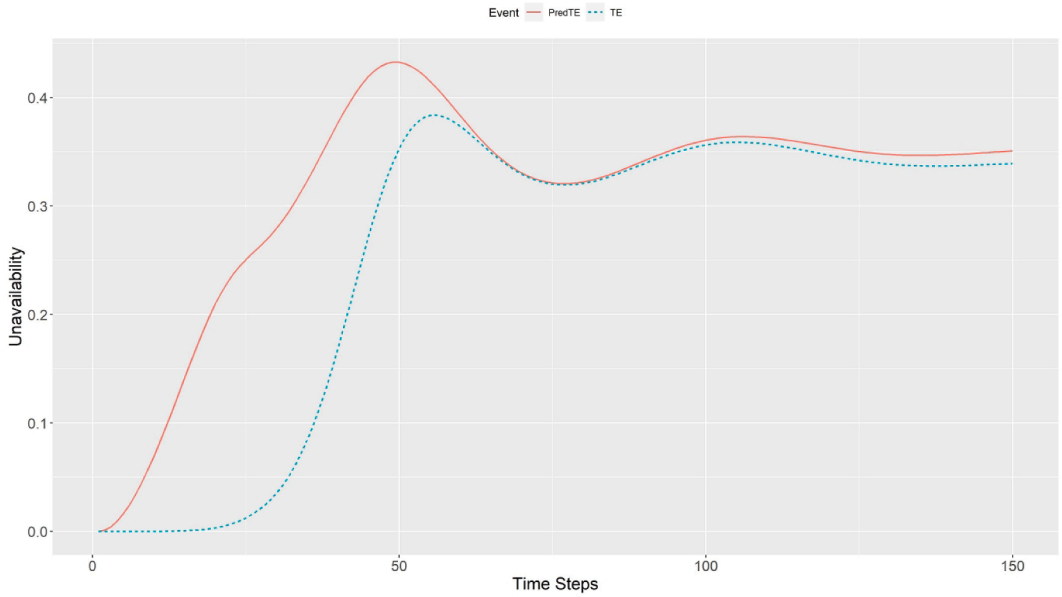



**Fig. 9.** Unavailability of the system changes by the new structure of the fault tree in Fig. 8.

**Table 9**
2*2 confusion matrix that depicts all four possible outcomes in classification.

| | True fault tree | |
|---|---|---|
| Reconstructed fault tree | Identified | Not identified |
| Identified | True Positive (TP) | False Positive (FP) |
| Not identified | False Negative (FN) | True Negative (TN) |

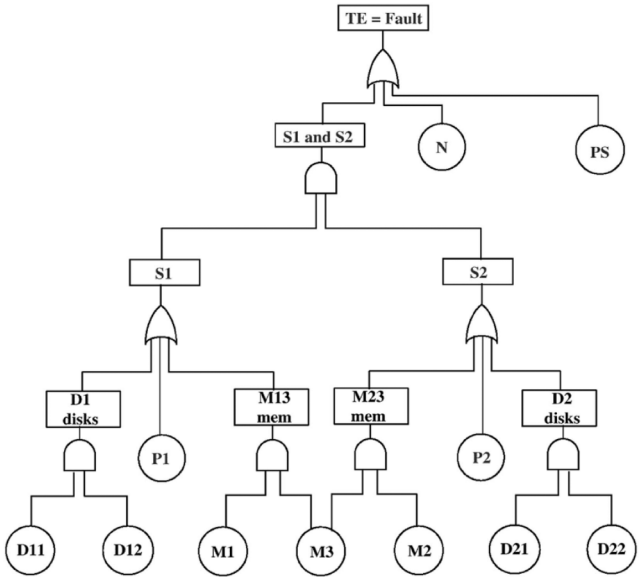
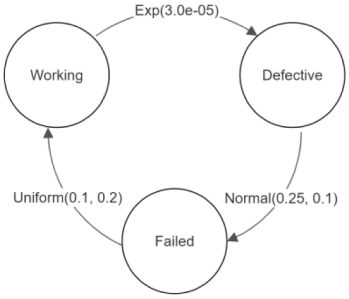



**Fig. 10.** A fault-tolerant multiprocessor system with a multi-state component PS.

which BE1 has three states: working (0), failed (1), idle (2) and the recorded data of Table 4. Here, CS={{BE2, BE3}, {BE1_1, BE2, BE3}, {BE1_1, BE3}}is the set of cut sets and hence the minimal cut sets are MCS={{BE2, BE3}, {BE1_1, BE3}}. Finally the fault tree equals TE= (BE2.BE3)+(BE1_1.BE3) = BE3.(BE2 + BE1_1), which is also illustrated in Fig. 5.

**Table 10**
Reliability and maintainability distribution functions of the basic events in Fig. 10.

| | Basic events | Reliability distribution (rate in f/h) | Maintainability distribution |
|---|---|---|---|
| 1 | Disk Dij | Exp(8.0e-05) | Weibull(5, 0.75) |
| 2 | Proc Pi | Exp(5.0e-07) | Exp(0.25) |
| 3 | Mem Mj | Exp(3.0e-08) | Weibull(5, 20) |
| 4 | Bus N | Exp(2.0e-09) | Exp(0.006) |
| 5 | Power supply PS |  | |



- **Parameter Learning**: Once the structure of the fault tree is extracted from data, we use it to calculate the reliability metrics of the constructed fault tree (here is the fault tree of Fig. 4). The first step to the quantitative analysis is to estimate reliability and maintainability probability distribution functions of the basic events, based on the time series data. Suppose we are interested in estimating the reliability distribution of BE1. We calculate the times to failures by looking at the points in time where the state of the basic event changes from working (label = 0) to failed (label = 1). For example, the first two times to failures for BE1 are:

$r_1 = 21.11844 - 17.96968 = 3.14876$, $r_2 = 24.74219 - 21.52825 = 3.21394$.

Also, times to repairs are calculated by looking at the points in time where the state of the basic event changes from failed (label = 1) to working (label = 0). Hence, $m_1 = 21.52825–21.11844 = 0.40981$. $r_i$'s and $m_i$'s are then used to estimate not only the parameters of the reliability and maintainability distributions, but also types of the

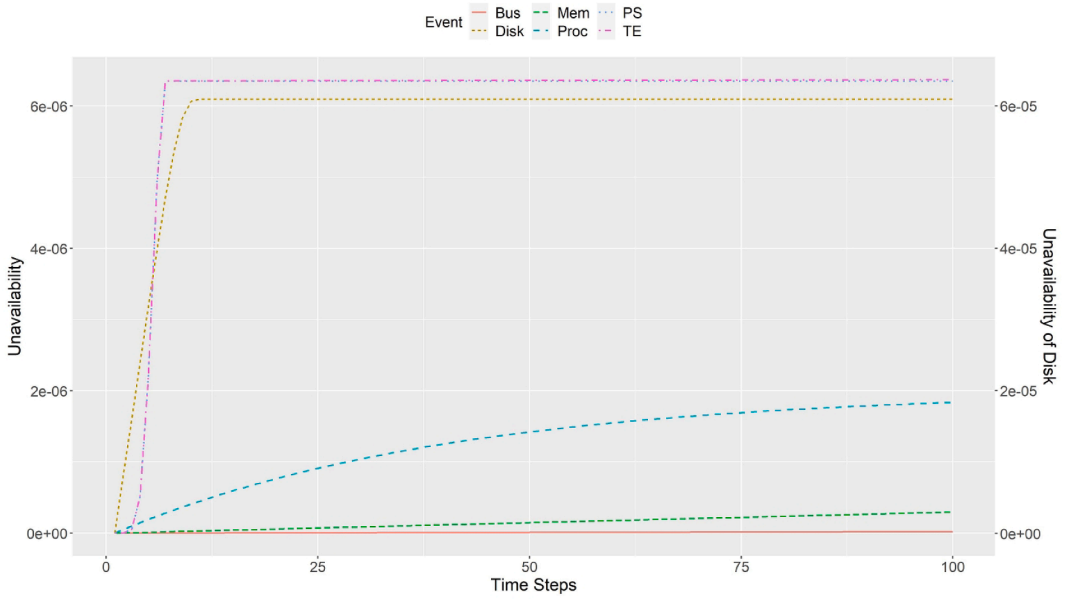



**Fig. 11.** Instantaneous unavailabilities for the fault-tolerant multiprocessor of Fig. 10.

distributions themselves, because our algorithm can cope with distributions other than the common exponential distribution.

The packages in R, gamlss (Rigby and Stasinopoulos, 2005) and fitdistrplus (Delignette-Muller and Dutang, 2015), cover a wide range of probability distributions supported on the interval [0,∞). Hence, we applied these R packages for the distribution fitting part. MTTF and MTTR for each basic event are the means of the reliability and maintainability distributions, respectively.

For exponential distributions, unavailability of an event is MTTR/(MTTR + MTTF), but for non-exponential distributions we need more advanced methods to calculate the unavailability of the system. Unavailability is the probability that the component or system is not operational. Proxel-based simulation (Lazarova-Molnar and Horton, 2003; Niloofar and Lazarova-Molnar, 2022) is not limited to exponential distributions, and can be used to determine the instantaneous unavailability of basic components with nonexponential distributions and multi-state events.

Assume a binary repairable basic event where the reliability distribution is estimated as an Exponential distribution function with rate 0.1 and the estimated repair distribution function is Normal with mean 2 and the standard deviation of 1. Fig. 6 illustrates the first three-time steps of the proxel simulation process for this basic event. Each proxel is a vector with three elements: State, Age intensity (which tracks the time that each of the possible state changes has been pending) and Probability.

Assuming that $\Delta t = 0.1$, the detailed calculation of $p_1$, $p_2$ and $p_3$ in Fig. 6 are as follows:

$$p_1 = \Delta t \times \frac{f(AgeInt_{P0})}{1 - F(AgeInt_{P0})} = 0.1 \times \frac{0.1e^{-0.1\times 0}}{\int_0^{\infty} 0.1e^{-0.1x}dx} = 0.01$$

$$p_2 = \Delta t \times \frac{f(AgeInt_{P00})}{1 - F(AgeInt_{P00})} = 0.1 \times \frac{0.1e^{-0.1\times 0.1}}{\int_{0.1}^{\infty} 0.1e^{-0.1x}dx} = 0.01$$

$$p_3 = \Delta t \times \frac{f(AgeInt_{P01})}{1 - F(AgeInt_{P01})} = 0.1 \times \frac{\frac{1}{\sqrt{2\pi(1)^2}}e^{-\frac{(0-2)^2}{2(1)^2}}}{\int_0^{\infty} \frac{1}{\sqrt{2\pi(1)^2}}e^{-\frac{(x-2)^2}{2(1)^2}}dx} = 0.0055$$

We, then, propagate the unavailability of each individual component through the fault tree to calculate the unavailability of the system. Fig. 7 shows the unavailability related to the basic events and the top event, for the fault tree from Fig. 4.

### 3.2.2. DDFTAnb algorithm

The system's fault tree along with the corresponding reliability measures are extracted from the observed time series data of faults using DDFTA algorithm. The observed data in Table 1 is only a portion of what can happen in a system with five components, and not all possibilities can be considered in fault tree analysis of the system. For example, the state of the system is unknown when only basic events 4 and 5 occur and other components are working perfectly (row 13 in Table 5). In DDFTAnb's classification module, we address the problem of the unobserved combination of basic events.

*Step 1*: Obtaining training and testing sets is the first step of the classification module. To obtain the training and testing sets, we need the power set for the 5 binary basic events. The power set for {BE1, BE2, BE3, BE4, BE5} or the set of all possible subsets of these basic events has $2^5 = 32$ elements, as shown in Table 5.

Combinations shown in rows 1 to 7 in Table 5 are observed and the state of the system (TE) for these combination of basic events can be extracted from the truth table in Table 1. Also, the number of occurrences for each row is indicated in the parenthesis in the TE column. For example, row 2 belongs to the case where all components are working, except for the one linked to the basic event BE3. We see this combination in Table 1 at times 22.12907 and 24.67361, along with the state of TE as working. Obviously, the state of TE when all basic events are working and when all of them are failed (rows 8 and 9 of Table 5) is 0 and 1, respectively. The top event also occurs in rows 10 to 12 because minimal cut sets {BE1, BE3, BE4} and {BE2, BE3, BE4} (Table 2) are subsets of these rows. The state of the top event is unknown for rows 13 to 32, because we have no information on these combinations of basic events. It is worth noticing that at this stage we have the highest percentage of missing values for TE, as not enough data is collected from the system yet. We take rows 1 to 12 as the training set with TE as the class variable, and rows 13–32 with missing information on TE belong to the testing set.

*Step 2 and 3*: In the next two steps, Naïve Bayes classifier as a supervised machine learning algorithm is fitted to the training set and the state of TE is classified by applying the fitted model to the testing set. Once we label the values of TE for these rows, we apply the method explained in Section 2.2.1, to build an updated structure of the fault tree. The extracted fault tree at this stage is most probably not reliable

**Table 11**
Results of the DDFTA and DDFTAnb algorithms for the multiprocessor fault tree of Fig. 10 considering different levels of data availability.

| | | | Data Availability | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 10 % | 20 % | 30 % | 40 % | 50 % | 60 % | 70 % | 80 % | 90 % | 100 % |
| DDFTA | Structure learning measures | Sen | 0.1091 ± 0.0238 | 0.1909 ± 0.0178 | 0.2636 ± 0.0178 | 0.3273 ± 0.0394 | 0.4000 ± 0.0713 | 0.7182 ± 0.0178 | 0.6566 ± 0.0376 | 0.8889 ± 0.0248 | 0.9182 ± 0.0178 | 1.0000 ± 0.0000 |
| | | Spe | 0.9904 ± 0.0004 | 0.9895 ± 0.0011 | 0.9911 ± 0.0004 | 0.9926 ± 0.0011 | 0.9941 ± 0.0007 | 0.9965 ± 0.0004 | 0.9957 ± 0.0007 | 0.9987 ± 0.0006 | 0.9985 ± 0.0005 | 1.0000 ± 0.0000 |
| | | ACC | 0.9881 ± 0.0003 | 0.9874 ± 0.0011 | 0.9891 ± 0.0005 | 0.9908 ± 0.0012 | 0.9925 ± 0.0009 | 0.9956 ± 0.0004 | 0.9948 ± 0.0008 | 0.9984 ± 0.0007 | 0.9983 ± 0.0005 | 1.0000 ± 0.0000 |
| | Parameter Learning measures | $\widehat{U}_n$ | 3.2673 e-10 ± 1.03 e-10 | 6.35589 e-07 ± 1.24 e-06 | 5.0887 e-06 ± 1.65 e-06 | 2.6121 e-06 ± 2.03 e-06 | 6.4228 e-06 ± 1.54 e-15 | 5.1527 e-06 ± 1.65 e-06 | 6.4228 e-06 ± 8.27 e-13 | 5.7172 e-06 ± 1.31 e-06 | 6.4228 e-06 ± 1.19 e-12 | 6.4228 2,649,198 e-06 ± 0.0000 |
| | | RMSE | 6.3411 e-06 ± 1.02 e-10 | 5.7111 e-06 ± 1.23 e-06 | 1.2983 e-06 ± 1.64 e-06 | 3.7822 e-06 ± 2.01 e-06 | 3.5645 e-12 ± 1.41 e-15 | 1.2606 e-06 ± 1.64 e-06 | 3.1690 e-12 ± 7.36 e-13 | 7.0034 e-07 ± 1.30 e-06 | 2.4905 e-12 ± 1.06 e-12 | 0.0000 ± 0.0000 |
| DDFTAnb | Structure learning measures | Sen | 0.4182 ± 0.0606 | 0.5545 ± 0.0675 | 0.4545 ± 0.0000 | 0.5727 ± 0.0597 | 0.5545 ± 0.0675 | 0.7545 ± 0.0380 | 0.6967 ± 0.0398 | 0.9091 ± 0.0000 | 0.9818 ± 0.0238 | ——— |
| | | Spe | 0.9928 ± 0.0012 | 0.9922 ± 0.0013 | 0.9890 ± 0.0005 | 0.9892 ± 0.0007 | 0.9894 ± 0.0014 | 0.9926 ± 0.0006 | 0.9914 ± 0.0010 | 0.9962 ± 0.0003 | 0.9977 ± 0.0003 | ——— |
| | | ACC | 0.9912 ± 0.0013 | 0.9910 ± 0.0015 | 0.9876 ± 0.0005 | 0.9881 ± 0.0007 | 0.9882 ± 0.0016 | 0.9920 ± 0.0007 | 0.9906 ± 0.0010 | 0.9960 ± 0.0003 | 0.9977 ± 0.0003 | ——— |
| | Parameter learning measures | $\widehat{U}_n$ | 6.4229 090,742 e-06 ± 3.52 e-11 | 6.4228 257,002 e-06 ± 1.05 e-12 | 6.4228 261,012 e-06 ± 7.82 e-13 | 6.4228 248,979 e-06 ± 1.28 e-12 | 6.4228 252,955 e-06 ± 1.20 e-12 | 6.4228 232,938 e-06 ± 1.05 e-12 | 6.4228 229,310 e-06 ± 8.28 e-13 | 6.4228 264,911 e-06 ± 1.52 e-15 | 6.4228 264,919 e-06 ± 1.96 e-17 | ——— |
| | | RMSE | 7.6858 e-11 ± 3.18 e-11 | 7.2144 e-13 ± 9.29 e-13 | 3.6145 e-13 ± 6.93 e-13 | 1.4266 e-12 ± 1.13 e-12 | 1.0729 e-12 ± 1.06 e-12 | 2.8470 e-12 ± 9.29 e-13 | 3.1686 e-12 ± 7.36 e-13 | 7.6663 e-16 ± 1.40 e-15 | 1.9424 e-17 ± 1.57 e-17 | ——— |

enough, because it is estimated using 12/32 = 37.5 % of the data. Classification results of the top event for rows 13–32 can be found in Table 6.

*Step 4*: the rows in Table 6 where "classified TE" has label 1, construct the new cut sets that should be added to the ones in Table 2. Table 7 shows that the new cut sets impose a great change in the minimal cuts sets which consequently affects the constructed fault tree as can be seen in Table 8 and Fig. 8.

*Step 5 and 6*: Since the results of the top event using Naïve Bayes model are not time-stamped, they cannot be used to update the estimates for the reliability and maintainability distributions. Hence, we use the estimated distribution functions of the DDFTA algorithm and the extracted fault tree of Fig. 8 to estimate unavailability of the system through proxel-based simulation. The unavailability of the system changes by the new structure of the fault tree and this change is depicted in Fig. 9.

As more data are recorded, newly observed data can be added to the training set to update the fault tree analysis and increase the classification accuracy.

### 3.3. Performance evaluation

To measure the performance of DDFTA in depicting a system's behaviour, we assume that the true behaviour of that system follows a repairable fault tree with a set of reliability and maintainability distributions as its parameters. We call this fault tree the original fault tree, and in the first simulation step, time series data are fabricated from this model. In the second step, truth table of the generated data set with time steps is used as an input to DDFTA algorithm. The structure of the fault tree is learnt and the unavailability of the system is computed. Finally, using DDFTAnb, future fault tree of the system and its unavailabilities are estimated. Hence, the performance of the presented method needs to be evaluated in regard to three aspects: structure learning evaluation, evaluation of reliability measures estimation and classification evaluation.

#### 3.3.1. Structure learning evaluation

To compare the reconstructed fault tree with the original fault tree, we use the 2*2 confusion matrix of Table 9, that depicts all four possible outcomes.

In this confusion matrix, true positive represents the number of sets that are both in the MCS of the reconstructed fault tree and the true fault

```
Toplevel "System";
"System" or "Power" "WANinterface" "SystemBUS" "GSMRinterface" "TMR";
"Power" and "PowerSupply1" "PowerSupply2" "PowerSupply3";
"WANinterface" and "WANcard1" "WANcard2";
"SystemBUS" and "BUS1" "BUS2";
"GSMRinterface" and "GSMRCard1" "GSMRCard2";
"TMR" or "CPUcore" "voter";
"CPUcore" 2of3 "CPUboard1" "CPUboard2" "CPUboard3";
"voter" and "FPGA1" "FPGA2";
1 "BUS1" lambda=4.4444e-6 repair=4;
  "BUS2" lambda=4.4444e-6 repair=4;
2 "FPGA1" lambda=3.003e-9 repair=4;
  "FPGA2" lambda=3.003e-9 repair=4;
3 "PowerSupply1" lambda=1.8182e-5 repair=6;
  "PowerSupply2" lambda=1.8182e-5 repair=6;
  "PowerSupply3" lambda=1.8182e-5 repair=6;
4 "WANcard1" lambda=2.5e-6 repair=6;
  "WANcard2" lambda=2.5e-6 repair=6;
5 "GSMRCard1" lambda=5.7078e-6 repair=6;
  "GSMRCard2" lambda=5.7078e-6 repair=6;
6 "CPUboard1" lambda=7.4074e-6 repair=6;
  "CPUboard2" lambda=7.4074e-6 repair=6;
  "CPUboard3" lambda=7.4074e-6 repair=6;
```

**Fig. 12.** Radio Block Center fault tree with six different types of basic events.

tree (correctly identified sets). False positive is the number of sets in the MCS of the extracted fault tree which are not in the MCS of the true fault tree (incorrectly identified sets). False negative is the number of incorrectly rejected sets and finally, true negative is the number of correctly rejected sets. Using the confusion matrix, we calculate the sensitivity, specificity, and accuracy (ACC):

$$Sensitivity = \frac{TP}{TP+FN}, \quad Specificity = \frac{TN}{TN+FP}, \quad ACC = \frac{TP+TN}{TP+TN+FP+FN}$$

Larger values of above-mentioned measures indicate higher performance in structure learning.

### 3.3.2. Reliability measures estimation

When the structure of the fault tree is extracted from the data set, the unavailability of the system can be calculated using proxel-based simulation. Since unavailabilities are calculated as transient solutions for each time step, we have a vector of instantaneous unavailabilities calculated for the extracted fault tree $\{\widehat{U}_i\}, i = 1, 2, \cdots, n$, where $n$ is the total number of time steps. For the original fault tree, there is also an associated vector of instantaneous unavailabilities: $\{U_i\}, i = 1, 2, \cdots, n$. Root Mean Square Error (RMSE) is used to compare these vectors of unavailabilities:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (U_i - \widehat{U}_i)^2}{n}} \tag{1}$$

Better estimation of unavailability leads to a smaller distance between $\{\widehat{U}_i\}$ and $\{U_i\}$, hence smaller values of RMSE. We also report $\widehat{U}_n$ and $U_n$ as the final stable unavailability values.

### 3.3.3. Classification evaluation

In the classification module, first the training set (observed cut sets) and the testing set (unobserved cut sets) are prepared. Then, a Naïve Bayes model is fitted to the training set and the fitted model is applied to classify the top event in the test set. Since the classification module includes extracting the structure and the unavailability of the learnt fault tree, it is evaluated in regard to structure learning and estimation of the reliability measures. Hence, the methods of Sections 3.3.1 and 3.3.2 are applied for the classification module as well.

We assess the performance of our algorithm using two repairable fault trees: 1) A fault-tolerant multiprocessor system shown in Fig. 10 (Malhotra and Trivedi, 1995); 2) Radio Block Center (RBC) fault tree (Fig. 12) explained in Galileo textual format (Sullivan and Dugan, 1996). The general steps in the experiments are as follows:

1. Generate time series data from the basic events of each original fault tree.
2. Build the timely truth tables based on each generated time series.
3. Obtain training and testing sets using the truth table with time stamps.
4. Learn the fault tree (structure and parameters) from the observed data set using DDFTAnb algorithm.
5. Compare the MCS of the reconstructed fault tree with that of the original fault tree in terms of sensitivity, specificity, accuracy (ACC).

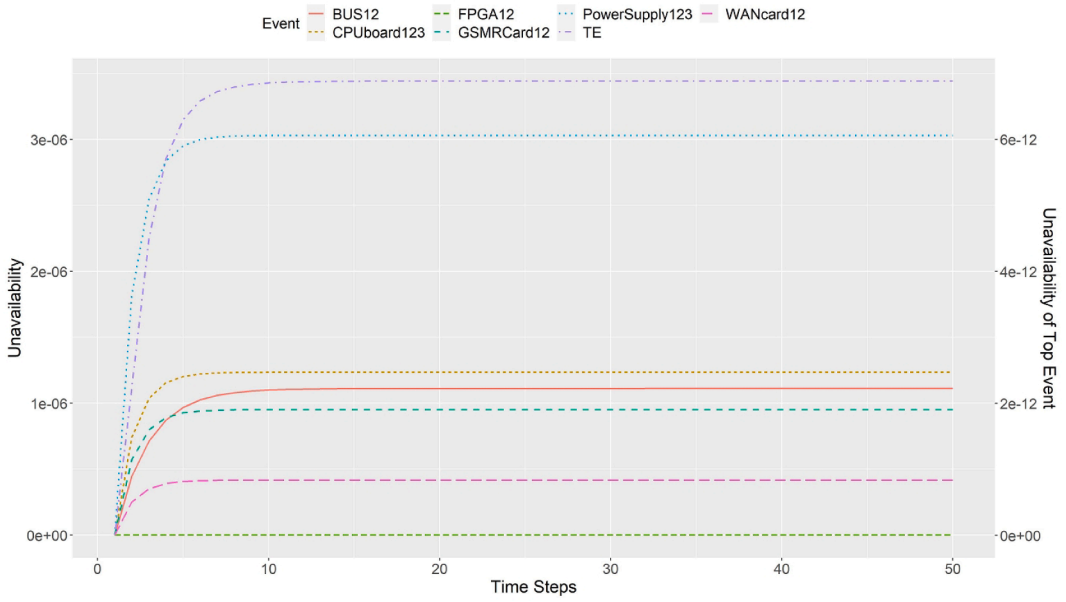



**Fig. 13.** Unavailability values for the RBC fault tree in Fig. 12.

**Table 12**
Results of the DDFTA and DDFTAnb algorithms for the RBC fault tree considering different percentages of data availabilities.

| | | | Data Availability | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 10 % | 20 % | 30 % | 40 % | 50 % | 60 % | 70 % | 80 % | 90 % | 100 % |
| DDFTA | Structure learning measures | Sen | 0.1250 ± 0.0000 | 0.2750 ± 0.1200 | 0.3250 ± 0.0980 | 0.4500 ± 0.1660 | 0.4750 ± 0.1430 | 0.5000 ± 0.1550 | 0.6500 ± 0.1630 | 0.7500 ± 0.1340 | 0.7750 ± 0.1429 | 1.0000 ± 0.0000 |
| | | Spe | 0.9972 ± 3e-04 | 0.997 ± 5e-04 | 0.9974 ± 5e-04 | 0.9981 ± 7e-04 | 0.9983 ± 7e-04 | 0.9983 ± 7e-04 | 0.9990 ± 8e-04 | 0.9995 ± 6e-04 | 0.9995 ± 6e-04 | 1.0000 ± 0.0000 |
| | | ACC | 0.9967 ± 3e-04 | 0.9966 ± 6e-04 | 0.9971 ± 6e-04 | 0.9978 ± 8e-04 | 0.9980 ± 7e-04 | 0.9981 ± 8e-04 | 0.9988 ± 8e-04 | 0.9994 ± 6e-04 | 0.9994 ± 7e-04 | 1.0000 ± 0.0000 |
| | Parameter learning measures | $\widehat{U}_n$ | 1.80e-13 ± 3.54e-13 | 1.76e-12 ± 4.82e-13 | 2.61e-12 ± 1.01e-12 | 3.71e-12 ± 1.16e-12 | 4.39e-12 ± 1.29e-12 | 4.20e-12 ± 1.87e-12 | 5.29e-12 ± 8.59e-13 | 5.78e-12 ± 1.19e-12 | 6.09e-12 ± 1.19e-12 | 6.86e-12 ± 0.0000 |
| | | RMSE | 6.48e-12 ± 3.44e-13 | 4.94e-12 ± 4.59e-13 | 4.12e-12 ± 9.74e-13 | 3.06e-12 ± 1.12e-12 | 2.41e-12 ± 1.25e-12 | 2.59e-12 ± 1.81e-12 | 1.53e-12 ± 8.29e-13 | 1.06e-12 ± 1.16e-12 | 7.60e-13 ± 1.16e-12 | 0.0000 ± 0.0000 |
| DDFTAnb | Structure learning measures | Sen | 0.3500 ± 0.1800 | 0.3500 ± 0.2616 | 0.5000 ± 0.1550 | 0.6250 ± 0.1096 | 0.6562 ± 0.1049 | 0.6500 ± 0.1200 | 0.7250 ± 0.1429 | 0.8250 ± 0.0600 | 0.8250 ± 0.0600 | ——— |
| | | Spe | 0.9989 ± 4e-04 | 0.9981 ± 7e-04 | 0.9982 ± 4e-04 | 0.9983 ± 5e-04 | 0.9982 ± 7e-04 | 0.9983 ± 6e-04 | 0.9987 ± 7e-04 | 0.9994 ± 3e-04 | 0.9995 ± 3e-04 | ——— |
| | | ACC | 0.9985 ± 3e-04 | 0.9978 ± 8e-04 | 0.9979 ± 5e-04 | 0.9981 ± 5e-04 | 0.9981 ± 8e-04 | 0.9982 ± 6e-04 | 0.9985 ± 8e-04 | 0.9993 ± 4e-04 | 0.9994 ± 3e-04 | ——— |
| | Parameter learning measures | $\widehat{U}_n$ | 1.48e-06 ± 1.78e-06 | 7.40e-07 ± 9.67e-07 | 2.47e-07 ± 4.84e-07 | 9.47e-12 ± 2.46e-12 | 7.64e-12 ± 2.57e-12 | 7.89e-12 ± 3.00e-12 | 7.28e-12 ± 1.32e-12 | 6.51e-12 ± 4.33e-13 | 6.51e-12 ± 4.33e-13 | ——— |
| | | RMSE | 1.45e-06 ± 1.74e-06 | 7.24e-07 ± 9.47e-07 | 2.41e-07 ± 4.73e-07 | 2.87e-12 ± 1.94e-12 | 2.36e-12 ± 9.93e-13 | 2.15e-12 ± 2.20e-12 | 1.16e-12 ± 7.00e-13 | 3.51e-13 ± 4.21e-13 | 3.51e-13 ± 4.21e-13 | ——— |

6. Use the reconstructed fault tree and the reliability and maintainability distributions to obtain the reliability measures of the top event as well as those of the basic events.
7. Estimate the structure and reliability measures of the system for the unobserved combinations of the basic events using DDFTAnb's classification module.
8. Report the evaluation measures in terms of 95 % confidence intervals.

### 3.4. A fault-tolerant multiprocessor system

Fig. 10 shows the fault tree of a fault-tolerant multiprocessor system which consists of two processors $P_i$ (i = 1, 2) with private memories $M_i$ (i = 1, 2) and $M_3$ as a shared one. A processor and a memory form a processing unit. Each processing unit is connected to a mirrored disk system $D_{ij}$ (i = 1, 2 and j = 1, 2), forming a processing subsystem. Both the processing subsystems and $M_3$ are connected via an interconnection Bus N. (Bobbio et al., 2001) refine the description of the multiprocessor system by adding the component power supply (PS) such that, when failing, it causes a system failure. The PS is modelled with three possible modes: working, defective and failed, where the first corresponds to a nominal behaviour, the second to a defective working mode with abnormal voltage provided, while the last mode (failed) corresponds to a situation where the PS cannot work at all. As anticipated, the failed mode causes the whole system to be down. According to the literature, the failure distribution of all components (except for the PS) is assumed to be exponential with failure rates given in Table 10, expressed in failures per hour (f/h). State changes diagram for PS, is also illustrated in Table 10, where it has exponential probability distribution with the rate of 3.0e-05 (Exp(3.0e-05)) as the transition probability from working to defective state. PS fails with a rate following Normal(0.25, 0.1) distribution function, and it is repaired again with Uniform(0.1, 0.2) transition proability. For binary events, we add individual repair distributions that are not limited to exponential distribution to highlight the ability of our algorithm to cope with non-exponential probability distribution functions, as well as repairable and multi-state components.

Unavailability values, calculated using proxel-based simulation for the basic events and the system (top event), are illustrated in Fig. 11, and the system unavailability ($U_n$) is 6.422826e-06. The results of the DDFTA algorithm for the fault tree in Fig. 10 considering 10 % to 100 % data availabilities are shown in Table 11. As we observe more data points, unavailability and RMSE values converge to the true value 6.422826e-06 and the ideal value of 0, respectively. As expected, the best structure learning performance occurs with the highest data availabilities and worsens as the data availability decreases. As can be seen, Naïve Bayes classifier, indicated by NB, performs relatively well for


**Fig. 14.** ACC mean values for MP and RBC fault trees, considering different levels of data availability, for DDFTA and DDFTAnb.


**Fig. 15.** $\widehat{U}_n$ means of MP and RBC fault trees, considering different levels of data availability, for DDFTA and DDFTAnb.

small amounts of training data because it has a low propensity to overfit.

### 3.5. Radio block center

Radio Block Center (RBC) is the most important subsystem of The European Railway Traffic Management System / European Train Control System (Flammini et al., 2005). It is responsible for guaranteeing a safe outdistancing between trains by managing the information received from the onboard subsystem and from the interlocking subsystem. In the RBC fault tree illustrated in Fig. 12, "BUS1" lambda = 4.4444e-6 repair = 4 means that the reliability and maintainability distribution of the basic event "BUS1" are exponential with a failure rate of 4.4444e-6 and a repair rate of 4, respectively. Estimated unavailability of the system is 6.8699e-12 and the instantaneous unavailabilities are illustrated in Fig. 13. Results shown in Table 12, demonstrate that this fault tree has been affected by loss of data more than the other two examples, because even with 90 % of data availability, DDFTA's sensitivity is 0.775. We suspect that the reason for this is that the fault events are rare, and the system is highly reliable.

## 4. Discussion

In this paper, we investigate two fault trees as case studies, a fault-tolerant multiprocessor (MP) and the radio block center (RBC). MP has a lower reliability measure than RBC since the unavailability value for MP is 6.4228e-06 and that of the RBC equals 6.86e-12. Furthermore, MP has a multi-state event (PS) and repair rates that follow distributions other than exponential.

In terms of structure learning, comparing the accuracies of the two fault trees indicate that for lower data availability applying the classification module is highly promising (Fig. 14). For RBC, as a highly reliable system, average ACC values when applying the classification

**Table 13**
Computational times (in seconds) for RBC and MP fault trees.

| Parameter Learning | | | | |
|---|---|---|---|---|
| Fault Tree | # different types of basic events | # time steps | | |
| | | 50 (T = 5, Δt = 0.1) | 100 (T = 5, Δt = 0.05) | 500 (T = 5, Δt = 0.01) |
| MP | 5 | 12.68 | 58.07 | 1529 |
| RBC | 6 | 22.73 | 94.17 | 2067.95 |
| **Structure Learning** | | | | |
| Fault Tree | # combinations of basic events | # minimal cut sets | | |
| MP | $3 \times 2^{10} = 3{,}072$ | 11 | 78.18 | |
| RBC | $2^{14} = 16{,}384$ | 8 | 3380.05 | |

module (NB) are higher even for 60 percentage of data availability. Accuracy values of DDFTA with classification module for MP are higher only for very low levels of data availabilities. In general, DDFTA with and without classification module has a higher accuracy for RBC fault tree.

Considering parameter learning, as illustrated in Fig. 15, estimated unavailability values converge to the true unavailability values as data availability increases. For both fault trees, estimated unavailabilities using DDFTA are lower than the true unavailability value, as opposed to DDFTAnb where the estimated unavailabilities are higher than the true unavailability value. The reason is that the sets of minimal cut sets predicted using DDFTA are always subsets of the real set of minimal cut sets. Unavailability values calculated using DDFTAnb are always higher (for low data availabilities) or equal (full data availability) to the true unavailability value which makes this algorithm more conservative since it estimates the reliability of the system lower than it really is. DDFTA calculates lower (for low data availability) or the same (for full data availability) unavailability values compared to the true unavailability value, which is risky since it shows the system as more reliable than it truly is.

DDFTAnb is affected by a set of experimental parameters. The structure learning step is affected by the number of basic events, whether basic events are multistate or binary, repairable/nonrepairable events and the number of minimal cut sets. Numbers of basic events and multistate or binary events affect the size of the truth table. For example, a system with 7 binary basic events has $2^7 = 128$ possible combinations of basic events, whereas a system with 13 binary events and an event with three states has $2^{13} \times 3^1 = 24{,}576$ number of combinations of basic events. The quantitative analysis part of the DDFTAnb is responsive to repairable/nonrepairable events, rare events, size of the time step and the total simulation time. If the total simulation time is 5 years and the size of the time step is 1 day, then the total number of time steps are $5 \times 365 = 1{,}825$. For the same total simulation time of 5 years, if we take time steps of one month then we only have $5 \times 12 = 60$ time steps to calculate instantaneous unavailabilities.

For a single fault tree, all the above parameters are fixed and the experimenter cannot change them, except for the total simulation time and the size of the time step. DDFTAnb's results are not very sensitive to these parameters in general. However, for rare events DDFTAnb may obtain different unavailabilities for a fixed fault tree if we choose different total simulation time and size of the time step. Rare events may require larger total time and smaller time steps, hence larger number of time steps are necessary. Also, state changes of repairable events define the number of proxels that need to be calculated at each time step.

Table 13 summarizes the computation time on a workstation with 16 GB RAM and processor Core i7 2.8 GHz for the MP and RBC case studies. For the parameter learning step, the number of different types of basic events and the number of time steps are considered, and for the structure learning, we access the number of combinations of basic events and the number of minimal cut sets for both MP and RBC (computation times are reported in seconds).

According to Table 13, DDFTA and DDFTAnb algorithms are not very efficient for complex systems with rare events. The main drawback is that as the number of cut sets, or the number of basic components increases with the size of the system, the presented algorithm becomes slower. Also, some types of basic events need larger T (total time) with smaller Δt (size of the time steps). Hence, the quantitative analysis becomes more time-consuming. One way to overcome this difficulty is to divide the whole system into its major subsystems and use parallel computing methods to overcome these issues.

## 5. Conclusion

We presented DDFTAnb algorithm, an efficient and novel algorithm for extracting repairable fault trees from incomplete multinomial time series data to extract the future fault-related behaviour of a system in terms of a fault tree and estimate the system's reliability measures. We extended the work of Lazarova-Molnar et al. (2020) by providing classification capability to address the issue of missing or unobserved cut sets in fault occurrences of basic events. Classifying the top event for unseen combinations of events becomes more critical when the system of interest is highly reliable with significantly rare events, or when it is composed of significantly many basic components. We demonstrated that our approach has clear benefits through two case studies.

DDFTAnb can extract and analyse multi-state repairable fault trees, compute reliability metrics for probability distributions other than the usual exponential probability distribution and estimate the future reliability of the system. In addition, DDFTAnb is highly recommended in cases when there is insufficient amount of data. In terms of our case studies, we observed the following: for 10 % of data availability, accuracies of DDFTAnb (DDFTA) are 0.9912 (0.9881) and 0.9985 (0.9967) for MP and RBC, respectively. However, in cases when there are sufficient or high amounts of data, DDFTA alone has a high performance: for 90 % of data availability, accuracies of DDFTAnb (DDFTA) are 0.9977 (0.9983) and 0.9994 (0.9994) for MP and RBC, respectively. Moreover, the reliability measure calculated by DDFTA for a system of interest is higher than the system's true reliability value, while DDFTAnb calculates a lower reliability measure than the system's true reliability.

DDFTAnb can be used to analyse any system where its fault tree can be expressed in terms of its minimal cut sets and has no limitations in this regard. The main limitation of the presented algorithm is that as the number of cut sets, or the number of basic components increase with the size of the system, it becomes slower and parallel computing can be considered as a solution. As future work, we intend to improve the classification performance of the presented algorithm and extend the tool to model and extract dynamic fault trees with more types of gates.

## CRediT authorship contribution statement

**Parisa Niloofar:** Conceptualization, Methodology, Software, Visualization, Validation, Writing – original draft, Writing – review & editing. **Sanja Lazarova-Molnar:** Supervision, Writing – review & editing, Conceptualization.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References


Aizpurua, J. I., & Muxika, E. (2013). Model-based design of dependable systems: Limitations and evolution of analysis and verification approaches. *International Journal on Advances in Security, 6*(1), 12–31.

Arnold, A., Griffault, A., Point, G., & Rauzy, A. (2000). The Altarica language and its semantics. *Fundamenta Informaticae, 34*(2–3), 109–124.

Ayvaz, S., & Alpay, K. (2021). Predictive maintenance system for production lines in manufacturing: A machine learning approach using Iot data in real-time. *Expert Systems with Applications, 173*, Article 114598.

Barlow, R. E., & Heidtmann, K. D. (1984). Computing K-out-of-N system reliability. *IEEE Transactions on Reliability, 33*(4), 322–323.

Bobbio, A., Portinale, L., Minichino, M., & Ciancamerla, E. (2001). Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Engineering & System Safety, 71*(3), 249–260.

Brito, L. C., Susto, G. A., Brito, J. N., & Duarte, M. A. (2022). An Explainable Artificial Intelligence Approach for Unsupervised Fault Detection and Diagnosis in Rotating Machinery. *Mechanical Systems and Signal Processing, 163*, Article 108105.

Cai, B., Liu, Y., Ma, Y., Liu, Z., Zhou, Y., & Sun, J. (2015). Real-time reliability evaluation methodology based on dynamic Bayesian Networks: A case study of a subsea Pipe Ram Bop System. *ISA transactions, 58*, 595–604.

Caldarola, L. (1980). Fault tree analysis with multistate components. In G. Apostolakis, S. Garribba, & G. Volta (Eds.), *Synthesis and analysis methods for safety and reliability studies*. Boston, MA: Springer, US.

Chen, J., Roberts, C., & Weston, P. (2008). Fault detection and diagnosis for railway track circuits using Neuro-Fuzzy systems. *Control Engineering Practice, 16*(5), 585–596.

Chiacchio, F., Aizpurua, J. I., Compagno, L., & D'Urso, D. (2020). Shyftoo, an object-oriented Monte Carlo simulation library for the modeling of stochastic hybrid fault tree automaton. *Expert Systems with Applications, 146*, Article 113139.

Chiacchio, F., D'Urso, D., Compagno, L., Pennisi, M., Pappalardo, F., & Manno, G. (2016). Shyfta, a stochastic hybrid fault tree automaton for the modelling and simulation of dynamic reliability problems. *Expert Systems with Applications, 47*, 42–57.

Compare, M., Baraldi, P., Bani, I., Zio, E., & Mc Donnell, D. (2017). Development of a bayesian multi-state degradation model for up-to-date reliability estimations of working industrial components. *Reliability Engineering & System Safety, 166*, 25–40.

Delignette-Muller, M. L., & Dutang, C. (2015). Fitdistrplus: An R package for fitting distributions. *Journal of statistical software, 64*(4), 1–34.

Dogan, A., & Birant, D. (2021). Machine learning and data mining in manufacturing. *Expert Systems with Applications, 166*, Article 114060.

Domingos, P., and M. Pazzani. 1996. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classi Er. In *Proc. 13th Intl. Conf. Machine Learning,* edited 105-112: Citeseer.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Feiler, P. H., Lewis, B. A., & Vestal, S. (2006). The Sae architecture analysis & design language (Aadl) a standard for engineering performance critical systems. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, October 4th-6th* (pp. 1206–1211). Germany: Munich.

Flammini, F., N. Mazzocca, M. Iacono, and S. Marrone. 2005. Using repairable fault trees for the evaluation of design choices for critical repairable systems. In *Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05),* October 12th-14th, Heidelberg, Germany, 163-172.

Guo, Y., Zhong, M., Gao, C., Wang, H., Liang, X., & Yi, H. (2021). A discrete-time Bayesian network approach for reliability analysis of dynamic systems with common cause failures. *Reliability Engineering and System Safety, 216*, Article 108028.

Huang, W., Zhang, Y., Yu, Y., Xu, Y., Xu, M., Zhang, R., … Liu, Z. (2021). Historical data-driven risk assessment of railway dangerous goods transportation system: comparisons between entropy weight method and scatter degree method. *Reliability Engineering & System Safety, 205*, Article 107236.

Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing, 20*(7), 1483–1510.

Kabir, S. (2017). An overview of fault tree analysis and its application in model based dependability analysis. *Expert Systems with Applications, 77*, 114–135.

Lampis, M., & Andrews, J. (2009). Bayesian belief networks for system fault diagnostics. *Quality and Reliability Engineering International, 25*(4), 409–426.

Lazarova-Molnar, S. (2005). *The Proxel-based method-formalisation, analysis and applications*. Magdeburg, Germany: Otto-von-Guericke-University. Ph.D. thesis.

Lazarova-Molnar, S., & Horton, G. (2003). Proxel-based simulation of stochastic petri nets containing immediate transitions. *Electronic Notes in Theoretical Computer Science, 85*(4), 203–217.

Lazarova-Molnar, S., Niloofar, P., & Barta, G. K. (2020). Automating reliability analysis: Data-driven learning and analysis of multi-state fault trees. In *30th European Safety and Reliability Conference and 15th Probabilistic Safety Assessment and Management Conference, November 1st-5th* (pp. 1805–1812). Italy: Venice.

Lee, W.-S., Grosh, D. L., Tillman, F. A., & Lie, C. H. (1985). Fault tree analysis, methods, and applications-a review. *IEEE Transactions on Reliability, 34*(3), 194–203.

Liggesmeyer, P., & Rothfelder, M. (1998). Improving System Reliability with Automatic Fault Tree Generation. *Digest of Papers. Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing (Cat No. 98CB36224)*. IEEE. edited 90–99.

Linard, A., D. Bucur, and M. Stoelinga. 2019. Fault trees from data: Efficient learning with an evolutionary algorithm. In *5th International Symposium on Dependable Software Engineering: Theories, Tools, and Applications,* November 27th–29th, Shanghai, China, 19-37.

Lisnianski, A., & Levitin, G. (2003). *Multi-state system reliability: Assessment, optimization and applications*. Singapore, Singapore: World scientific.

Majdara, A., & Wakabayashi, T. (2009). Component-based modeling of systems for automated fault tree generation. *Reliability Engineering & System Safety, 94*(6), 1076–1086.

Malhotra, M., & Trivedi, K. S. (1995). Dependability modeling using petri-nets. *IEEE Transactions on Reliability, 44*(3), 428–440.

Mukherjee, S., and A. Chakraborty. 2007. "Automated Fault Tree Generation: Bridging Reliability with Text Mining". In *2007 Annual Reliability and Maintainability Symposium,* January 22nd-25th, Orlando, USA, 83–88.

Nadjafi, M., Farsi, M. A., Jabbari, H., & Management. (2017). Reliability analysis of multi-state emergency detection system using simulation approach based on fuzzy failure rate. *International Journal of System Assurance Engineering, 8*(3), 532–541.

Nauta, M., Bucur, D., & Stoelinga, M. (2018). Lift: Learning fault trees from observational data. In *15th International Conference on Quantitative Evaluation of Systems, September 4th-7th* (pp. 306–322). China: Beijing.

Niloofar, P., & Lazarova-Molnar, S. (2021). Fusion of data and expert knowledge for fault tree reliability analysis of cyber-physical systems. In *2021 5Th International Conference on System Reliability and Safety (ICSRS), November 24th-26th* (pp. 92–97). Italy: Palermo.

Niloofar, P., and S. Lazarova-Molnar. 2022. Collaborative data-driven reliability analysis of multi-state fault trees. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*:1748006X221076290.

Nolan, P. J., M. G. Madden, and P. Muldoon. 1994. Diagnosis using fault trees induced from simulated incipient fault case data. In *Second International Conference on Intelligent Systems Engineering,* edited 304–309.

Papadopoulos, Y., and J. A. McDermid. 1999. Hierarchically performed hazard origin and propagation studies. In *International Conference on Computer Safety, Reliability, and Security,* September 27th-29th, Toulouse, France, 139-152.

Qian, K., Yu, L., & Gao, S. (2021). Fault tree construction model based on association analysis for railway overhead contact system. *International Journal of Computational Intelligence Systems, 14*(1), 96–105.

Rigby, R. A., & Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C, 54*(3), 507–554.

Ruijters, E., & Stoelinga, M. (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review, 15*, 29–62.

Sharvia, S., Kabir, S., Walker, M., & Papadopoulos, Y. (2016). Model-based dependability analysis: State-of-the-art, challenges, and future outlook. *Software Quality Assurance*, 251–278.

Sikorska, J. Z., Hodkiewicz, M., & Ma, L. (2011). Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing, 25*(5), 1803–1836.

Solomatine, D. P., & Ostfeld, A. (2008). Data-driven modelling: Some past experiences and new approaches. *Journal of Hydroinformatics, 10*(1), 3–22.

Sullivan, K., and J. B. Dugan. 1996. Galileo User's Manual & Design Overview, University of Virginia. https://www.cse.msu.edu/~cse870/Materials/FaultTolerant/manual-galileo.htm, accessed 28 March 2021.

Theodoropoulos, P., Spandonidis, C. C., Giannopoulos, F., & Fassois, S. (2021). A deep learning-based fault detection model for optimization of shipping operations and enhancement of maritime safety. *Sensors, 21*(16), 5658.

Vesely, W. E., F. F. Goldberg, N. H. Roberts, and D. F. Haasl. 1981. Fault Tree Handbook. Technical Report *NUREG-0492*, Nuclear Regulatory Commission Washington DC, USA.

Waghen, K., & Ouali, M.-S. (2019). Interpretable logic tree analysis: A data-driven fault tree methodology for causality analysis. *Expert Systems with Applications, 136*, 376–391.

Waghen, K., & Ouali, M.-S. (2021). Multi-level interpretable logic tree analysis: A data-driven approach for hierarchical causality analysis. *Expert Systems with Applications, 178*, Article 115035.

Zhang, W., Jia, M.-P., Zhu, L., & Yan, X.-A. (2017). Comprehensive overview on computational intelligence techniques for machinery condition monitoring and fault diagnosis. *Chinese Journal of Mechanical Engineering (English Edition), 30*(4), 782–795.

Zheng, J., Pan, H., & Cheng, J. (2017). Rolling bearing fault detection and diagnosis based on composite multiscale fuzzy entropy and ensemble support vector machines. *Mechanical Systems and Signal Processing, 85*, 746–759.