Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, eds.

DATA-DRIVEN MODELLING OF REPAIRABLE FAULT TREES FROM TIME SERIES DATA WITH MISSING INFORMATION

Parisa Niloofar
Sanja Lazarova-Molnar

Mærsk Mc-Kinney Møller Institute
University of Southern Denmark
Campusvej 55
Odense, 5230, DENMARK

ABSTRACT

Fault tree analysis is one of the most popular techniques for dependability analysis of a wide range of systems. True fault-related behavior of a system would be more accurately reflected if the system's fault tree is derived from a combination of observational data and expert knowledge, rather than expert knowledge alone. The concept of learning fault trees from data becomes more significant when systems change their behaviors during their lifetimes. We present an algorithm for learning fault trees of systems with missing information on fault occurrences of basic events. This algorithm extracts repairable fault trees from incomplete multinomial time series data, and then uses simulation to estimate the system's reliability measures. Our algorithm is not limited to exponential distributions or binary events. Furthermore, we assess the sensitivity of our algorithm to different percentages of missingness and amounts of available data.

1 INTRODUCTION

Fault Tree Analysis (FTA) investigates which components influence failures of a system, and, thus, fault trees model probabilistic causal chains of events that end in a global system failure (Vesely et al. 1981; Lee et al. 1985). Conventional fault trees consider only failure occurrences of basic events. However, in most of the real world cases, it is necessary to consider both the occurrence of a basic event (typically a fault in a basic component) and its corresponding withdrawal (typically a repair). This means that not only the information about failure times of basic components are needed, but it is also necessary to have a proper maintenance/repair policy. Repairable fault trees address this issue and consider both faults and repairs within a system.

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 yield more than two states (Lisnianski and Levitin 2003). Multi-state fault trees have the same structure of regular fault trees, but the components or the system may have more than two functioning levels. Although many extensions of fault trees have been proposed in the literature, they suffer from a variety of shortcomings. In particular, even with the emerging availability of data through Internet of Thing (IoT) devices and all existing software tools, fault tree analysis requires a lot of manual effort and expert knowledge.

The issue of missing data may arise in almost all studies and time series data of faults are not an exception. Missing data are defined as questions without answers or variables without observations. By far the most common approach to missing data is to simply omit those cases with the missing data and analyze

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

the remaining data (Rubin 1976; Little and Rubin 2019). This approach is known as the complete case (or available case) analysis or listwise deletion. Removing entries with missing values can cause bias in the estimation of model parameters and more importantly it can reduce the representativeness of the samples from the population of interest. Imputation, as another approach to cope with missing data, is the process of replacing missing data with estimated values. Instead of omitting entries with missing values, this approach preserves all cases by replacing missing data with probable values estimated by other available information. Once all missing values are imputed and the data set is complete, standard techniques can be applied to analyze the data.

In this paper, we extend DDFTA (Data-Driven Fault Tree Analysis) algorithm introduced by Lazarova-Molnar et al. (2020) to be able to learn fault trees of a system with missing information on fault occurrences of basic events. Extended DDFTA (DDFTAe) algorithm extracts repairable multi-state fault trees from incomplete multinomial time series data, and then analyzes the results to estimate the system's reliability measures. Deriving reliability models from data can be more insightful than utilizing solely expert knowledge as it is capable of capturing the true behavior of a system, after the system has been put in use. The data-driven models can, however, very well be supplemented by expert knowledge for validation and approval of extracted models.

The rest of the paper is organized as follows: Section 2 presents background and related work, Section 3 provides the methodology where DDFTAe is described in detail. In Section 4, three case studies are demonstrated, and finally, in Section 5 we conclude the paper.

2 BACKGROUND AND RELATED WORK

Over the past two decades, research has focused on simplifying dependability analysis by looking at how fault trees can be automatically derived from data. This has led to the field of model-based dependability analysis (MBDA) (Kabir 2017; Joshi et al. 2006). Many of MBDA techniques such as Hierarchically Performed Hazard Origin & Propagation Studies (HiP-HOPS) (Papadopoulos and McDermid 1999) and AltaRica (Arnold et al. 2000), use fault tree analysis as their primary means of system dependability analysis and automate the fault tree generation process.

Classical FTA is primarily knowledge-driven rather than data-driven. With evolving system designs, model building based on experts' knowledge becomes outdated. Ruijters and Stoelinga (2015) present a survey on standard fault tree analysis and its extensions, covering technical details of different types of fault trees and their analyses approaches. A literature review on different model based dependability analysis approaches is available in Sharvia et al. (2016) and Aizpurua and Muxika (2013). Kabir (2017) reviews the standard FTA, FTA's different extensions and describes its limitations. He also reviews different MBDA approaches where fault trees are used as an analysis technique.

Nauta et al. (2018) introduced LIFT (Learning Fault Trees) to learn the structure of static fault trees from untimed data bases with Boolean event variables. 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. Also, their databases did not need any information about intermediate events. Furthermore, 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.

The above-mentioned techniques also use data to construct fault trees, however, they do not use time series data nor do they consider the multi-state systems. Our method differs from those in the literature since we consider incomplete timed data for components with more than two states and provide a complete solution to calculate reliability measures from observational data following distributions other than exponential distribution.

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 as 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 $\{0,1,\dots,M\}$, where 0 indicates the completely failed state, M indicates the 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. Time series data of a system consists of a sequence of status change times for each basic event and the system state.

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 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 known, 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 Regression Modelling of Binary/Multinomial Time Series Data

Consider a binary time series $\{Y_t\}$ taking the values 0 or 1 (i.e., working, failure), and related covariate or auxiliary stochastic data represented by a column vector $\{Z_{1,t}, Z_{2,t}, \dots, Z_{m,t}\}$, $t = 1, 2, 3, \dots, T$, where T is the total time. The time-dependent random covariate vector process may include discrete or continuous variables that influence the evolution of the primary series of interest $\{Y_t\}$. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist (Cox and Snell 1989; Diggle et al. 2002; Fahrmeir and Kaufmann 1987). Suppose $\pi_t = P(Y_t = 1)$, then the generalized linear model that links the predictor variables to π_t is as follows:

$$logit(\pi_t) = \ln\left(\frac{\pi_t}{1-\pi_t}\right) = \beta_0 + \beta_1 Z_{1,t} + \beta_2 Z_{2,t} + \cdots + \beta_m Z_{m,t} \quad (1)$$

In this paper, to model binary basic events (typically associated with faults in basic components), we consider the logistic regression, where the predictor variables are times of status changes, system failure or other basic events. For multi-state systems where we have more than two states for components or the system, we apply the methods of multinomial logistic regression. Assume that Y_t has M functioning states such as $\pi_{t,c} = P(Y_t = c), c = 1, 2, \dots, M$, and $\sum_{c=1}^{M} \pi_{t,c} = 1$, then we have:

$$logit(\pi_{t,c}) = \ln\left(\frac{\pi_{t,c}}{1-\pi_{t,c}}\right) = \beta_{0,c} + \beta_{1,c} Z_{1,t} + \beta_{2,c} Z_{2,t} + \cdots + \beta_{m,c} Z_{m,t} \quad (2)$$

The logistic regression models the probability of output, here the probability of failure of a component, in terms of inputs. Hence, logistic regression can only predict the probability of failure and does not perform statistical classification. Though by choosing a cutoff value and classifying inputs with probability

greater than the cutoff as one class, below the cutoff as the other it can be used to make predictions on the response variable.

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 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).

3 METHODOLOGY FOR DATA-DRIVEN LEARNING OF REPAIRABLE FAULT TREES

In this section, we describe the methods and techniques that we developed to enable the data-based reliability modelling and analysis. DDFTA (as described in Algorithm 1) comprises of two main steps (Lazarova-Molnar et al. 2020): 1) structure learning, or the qualitative analysis part, and 2) quantitative analysis based on the output of the first step. In the structure learning step, we extract the minimal cut sets (MCS) from the time series data set, and then use Boolean algebra to build a fault tree that aims to be mathematically identical to the true fault tree of the system. In the second step, the distribution functions of the basic events are approximated from the time series data. These reliability and repair distribution functions of the basic events, along with the fault tree structure, are inputs to the proxel-based simulation, which is then used to calculate system's reliability measures, in form of transient complete solutions. This algorithm consists of three functions: STRUCTURE (lines 1-6) returns the skeleton of the fault tree, MCS (lines 7-20) which extracts the minimal cut sets is a sub function here, RELIABILITY function (lines 21-36) applies the output of the STRUCTURE function and returns the reliability measures of the system.
The process workflow of DDFTAe, the extended data-driven fault tree analysis (DDFTA) algorithm that can cope with incomplete time series data of faults is shown in Figure 1. We can see that DDFTAe uses time series data from faults' occurrences and the time it takes for them to be repaired, although we may miss some of the records.

3.1 DDFTAe Algorithm

DDFTAe algorithm (Figure 1) comprises of four main steps: 1) converting time series data of faults to a truth table with time steps, 2) regression fitting on the observed part and predicting/imputing missing values, 3) learning the structure and the parameters of the fault tree and 4) estimating reliability measures. The last two steps are the basic components of the DDFTAe algorithm. In this section, the regression modelling part is described in detail and for the last two parts we refer the reader to Lazarova-Molnar et al. (2020).

Suppose we have an incomplete truth table with time steps like in Figure 2. In order to apply the method of Algorithm 1, this truth table needs to be completed, or in other words, *imputed*. The imputation process fills in the missing values for each basic event until the data sets contains no more missing values. Assume we want to impute the missing values for the binary basic event BE1.

Algorithm 1 Data-Driven Fault Tree Analysis: DDFTA

procedure STRUCTURE(timeseries dataset)
    `binary dataset` ← *Binary*(`timeseries dataset`) ▷ Converts timeseries into binary data set
    MC ← *MCS*(`binary dataset`) ▷ Extract the minimal cut sets
    FTBooleanEquation ← *BooleanAlgebra*(*MC*) ▷ Shows the fault tree in terms of boolean equation by simplifying the MCS using Boolean algebra
    FaultTree ← *Graph*(*FTBooleanEquation*) ▷ returns the graphical structure of the Boolean equation
    return *FaultTree*
procedure MCS(binary dataset)
    data ← `binary dataset`
    t ← `number of rows(data)`
    CutSet ← 0
    for *i* ← `1 to t` **do** ▷ Identifying the cut sets
        if data[i, TopEvent]=1 **then**
            BESet ← `indices of Basic events with state 1 in the ith row`
        CutSet ← *CutSet* ∪ *BESet* ▷ Set of all cut sets
    CutSetUnique ← `Remove duplicate sets`(*CutSet*)
    c ← `number of sets`(*CutSetUnique*)
    for *i* ← `1 to c` **do** ▷ Refining the cut sets into MCS
        if ∃ $C \in CutSetUnique$ *s.t.* $C \subset C_i$ **then**
            `remove` C_i `from` *CutSetUnique*
    return *CutSetUnique*
procedure RELIABILITY(*FaultTree*, *timeseries dataset*) ▷ Calculates the reliability measures given a fault tree structure
    N ← `Number of Basic events`
    for *k* ← `1 to N` **do**
        $RelDist_k$ ← `DistributionFitt(time-to-failure for` $BasicEvent_k$`)`
        $MainDist_k$ ← `DistributionFitt(time-to-repair for` $BasicEvent_k$`)`
        $MTTF_k$ ← `average(time-to-failure for` $BasicEvent_k$`)`
        $MTTR_k$ ← `average(time-to-repair for` $BasicEvent_k$`)`
        $OpeAva_k$ ← `Operational availability`($BasicEvent_k$)
        $InsAva_k$ ← `Proxel`(*FaultTree*, $BasicEvent_k$)
    $RelDist_{Top}$ ← `DistributionFitt(time-to-failure for` *TopEvent*`)`
    $MainDist_{Top}$ ← `DistributionFitt(time-to-repair for` *TopEvent*`)`
    $MTTF_{Top}$ ← `average(time-to-failure for` *TopEvent*`)`
    $MTTR_{Top}$ ← `average(time-to-repair for` *TopEvent*`)`
    $OpeAva_{Top}$ ← `Operational availability`(*TopEvent*)
    $InsAva_{System}$ ← `Proxel`(*FaultTree*, *TopEvent*)
    return *RelDist, MainDist, MTTF, MTTR, OpeAva, InsAva*

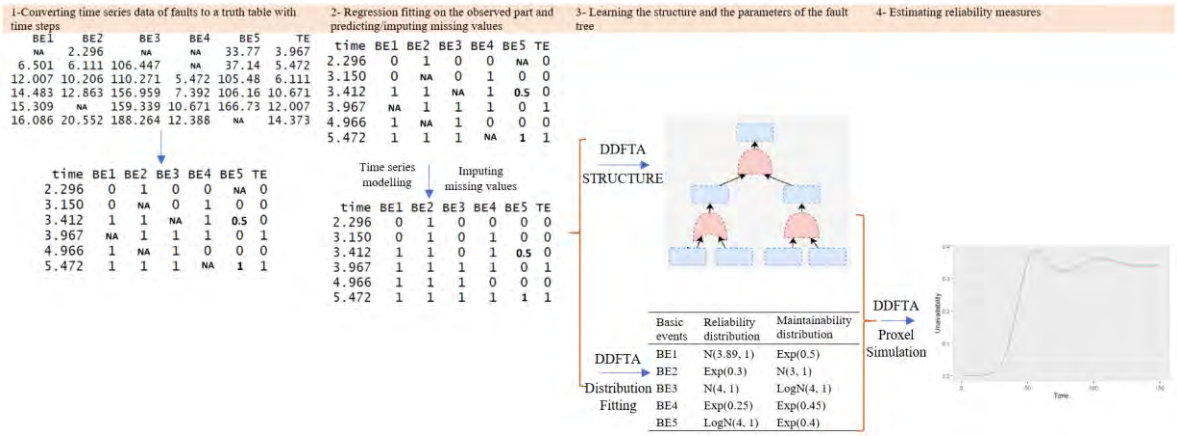


Figure 1: The process work flow of the DDFTAe algorithm.

time	BE1	BE2	BE3	BE4	BE5	TE
2.296	0	1	0	0	NA	0
3.150	0	NA	0	1	0	0
3.412	1	1	NA	1	0.5	0
3.967	NA	1	1	1	0	1
4.966	1	NA	1	0	0	0
5.472	1	1	1	NA	1	1

Figure 2: Incomplete truth table of a fault tree with five basic events.

Following the logistic regression model of Equation (1), BE1 is the response variable and all the other basic components along with time and the top event TE are the predictors in this model. This means that if $\pi = P(BE1 = 1)$, then:

$$logit(\pi) = \ln\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_2 BE2 + \beta_3 BE3 + \beta_4 BE4 + \beta_5 BE5 + \beta_6 TE + \beta_7 time$$

All the records where the response variable BE1 is observed are used to fit the model and estimate the model parameters β_i. Statistically insignificant (p-values > 0.05) predictors will be removed from the model and the best fitted model will be applied to make predictions for the records where we have no observation for BE1. The same process is utilized to complete all binary events. In Figure 2, we can see that BE5 is a multi-state event with {0, 0.5, 1} as its functioning levels. To fill in multi-state events, Equation (2) is the right model to choose for model fitting and making predictions. The standard *glm* function and *nnet* R package (Ripley et al. 2016) are used for fitting logistic regression and multinomial logistic regression models, respectively.

3.2 Performance Evaluation

To measure the performance of DDFTAe algorithm in depicting a system's behavior, we assume that the true behavior 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, the generated data set is artificially perturbed with different percentages of missingness. Since DDFTAe first imputes the data set, then learns the structure and finally computes the unavailability of the system, its performance needs to be evaluated in regards to these three aspects: imputation evaluation, structure learning evaluation, evaluation of reliability measures estimation.

3.2.1 Imputation Evaluation

To fill in the missing values for each basic event, first the data is divided into a train set (observed records) and a test set (unobserved or the records with missing values). Then a regression model is fitted to the train set and the best fitted model is applied to make predictions on the test set. Since the response variable is categorical, this is in fact a classification task that we evaluate like any other classifier. Classification accuracy (ACC) is a metric that summarizes the performance of a classification model as the number of correct predictions divided by the total number of predictions. In this paper ACC is calculated for both train and test sets.

3.2.2 Structure Learning Evaluation

To compare the reconstructed fault tree with the original fault tree, we use the 2*2 confusion matrix of Table 1, that depicts all four possible outcomes.

Table 1: 2*2 confusion matrix that depicts all four possible outcomes in predictions

	True fault tree	
Reconstructed fault tree	Identified	Not identified
Identified	True positive (TP)	False positive (FP)
Not identified	False negative (FN)	True negative (TN)

In this confusion matrix, true positive is the number of sets both in the MCS of the reconstructed fault tree and the true fault tree (correctly identified sets). False positive is the number of sets in the MCS of the predicted 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, accuracy (ACC) and F-measure:

$$Sensitivity = \frac{TP}{TP + FN}, \quad Specificity = \frac{TN}{TN + FP}, \quad ACC = \frac{TP + TN}{TP + TN + FP + FN}, \quad F - measure = \frac{2TP}{2TP + \mathrm{FP} + FN}$$

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

3.2.3 Reliability Measures Estimation

When the structure of the fault tree is extracted from the imputed 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 $\{\widehat{U}_i\}, i = 1, 2, \dots, n$, where n is the total number of time steps. For the original fault tree, there is also associated a vector of instantaneous unavailabilities: $\{U_i\}, i = 1, 2, \dots, n$. Root Mean Square Error (RMSE) is used to compare these vectors of unavailabilities:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(U_i - \widehat{U}_i)^2}{n}} \tag{3}$$

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.

4 CASE STUDIES

We assess the performance of our algorithm using three repairable fault trees: 1) A simple fault tree with five basic events and three types of gates shown in Figure 3; 2) A multi-state repairable fault tree illustrated in Figure 5; and 3) Radio Block Center (RBC) fault tree of Figure 7 explained in Galileo textual format (Sullivan and Dugan 1996). The general steps in the experiments are as follows:

1. Generate times series data from the basic events of each original fault tree.
2. Build the truth tables based on each generated time series.
3. Randomly perturb the data with 5%, 15% and 40% percentages of missing values, each 100 times.
4. Use the methods of Section 2.2 to impute missing values.
5. Learn the fault trees from the imputed truth tables using Algorithm 1.
6. Compare the MCS of the reconstructed fault tree with that of the original fault tree using sensitivity, specificity, accuracy (ACC), F-measure.
7. 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.

8. Report the evaluation measures in terms of 95% confidence intervals.

4.1 A Simple Fault Tree

In the fault tree configuration shown in Figure 3, we use abbreviations BE*n* and TE to denote Basic Event *n* and the Top Event, respectively. Time series data are generated from this fault tree with the reliability and maintainability distributions shown in Table 2.

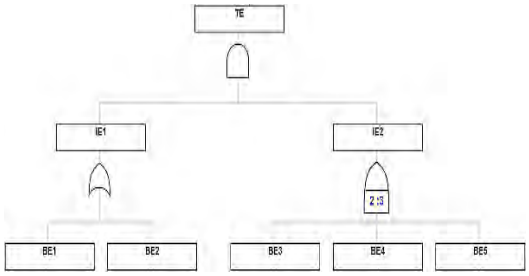


Figure 3: A simple fault tree with 5 basic events and 3 types of gates.

Table 2: Reliability and maintainability distribution functions of the basic events in Figure 3

Basic events	Reliability distribution	Maintainability distribution
BE1	N(4, 0.1)	Exp(1)
BE2	Exp(0.1)	N(2, 1)
BE3	N(10, 1)	LogN(0.5, 1)
BE4	Exp(0.1)	Weibull(5, 1)
BE5	LogN(2, 0.1)	Exp(2)

True unavailability values calculated using proxel-based simulation for the basic events and the system (top event) are illustrated in Figure 4, and the system unavailability (U_n) is 0.0216. Results of DDFTAe algorithm for the fault tree of Figure 3 considering 5%, 15% and 40% missingness percentages are shown in Table 3. As the percentages of missing values increase and less data are available, ACC on train sets become closer to those of the test sets, due to the model becoming overfit. As we lose more data points, unavailability and RMSE values deviate more from the true value 0.0216 and the ideal value 0, respectively. And naturally, the best structure learning performance belongs to 5% missingness and worsens as the missingness percentages increase.

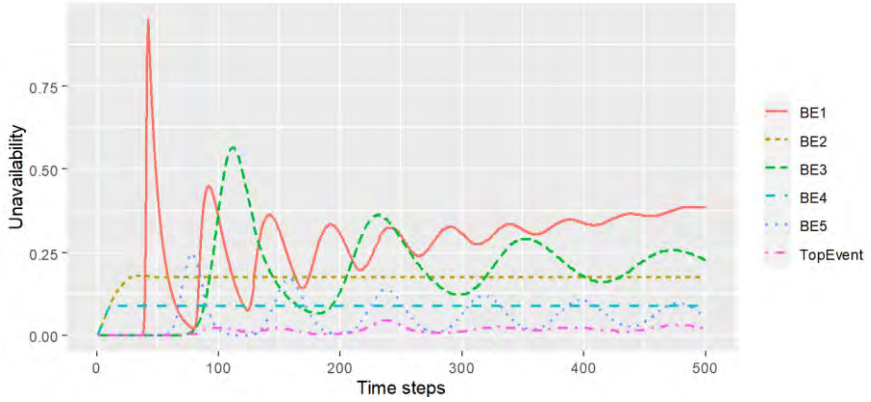


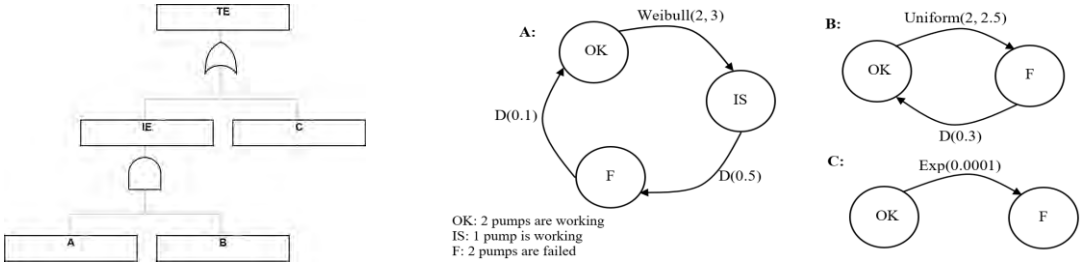
Figure 4: Instantaneous unavailability for the fault tree of Figure 3

Table 3: Results of the DDFTAe algorithm for the fault tree of Figure 3 considering different percentages of missingness

		Percentages of missingness		
		5%	15%	40%
Structure learning measures	Sensitivity	0.7441 ±0.0426	0.4980±0.0411	0.4383±0.0332
	Specificity	0.9576 ±0.0065	0.9167±0.0057	0.9096±0.0061
	ACC	0.9162 ±0.0129	0.8357±0.0114	0.8184±0.0104
	F-measure	0.7659 ±0.0375	0.5285±0.0354	0.4777±0.0312
Reliability measures	Unavailability	0.0274 ±0.0011	0.0344±0.0012	0.0383±0.0012
	RMSE	0.0067 ±0.0012	0.0142±0.0012	0.0185±0.0011
Imputation measures	Train set ACC	0.8131 ±0.0086	0.8133±0.0086	0.8136±0.0087
	Test set ACC	0.8111±0.0092	0.8125±0.0088	0.8136±0.0086

4.2 Multi-State Fault Tree

Our second case study is a multi-state repairable system (Figure 5) to assess the performance of the DDFTAe algorithm in learning multistate repairable fault trees from incomplete data sets. Basic event A consists of two pumps working in parallel, hence A is fully functioning if two pumps are working (OK), it is in an intermediate state (IS) if one of the pumps fails, and A fails completely (F) if both of the pumps fail. Basic events B and C have two functioning states, and C is not repairable. Figure 6 shows the system unavailability along with those of the basic evens. System unavailability equals 1.3492e-04. Performance evaluation measures are reported in Table 4. Although most of the measures indicate lower performances for higher percentages of missingness, they are not dramatically worsened.



(a) (b)

Figure 5: (a) A multi-state fault tree, and (b) State changes diagrams.

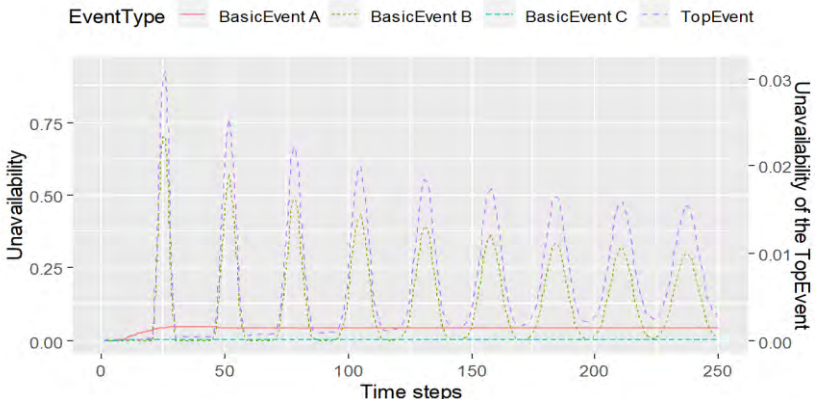


Figure 6: Unavailability of the basic events and the top event of the fault tree in Figure 5

Table 4: Results of the DDFTAe algorithm for the fault tree of Figure 5 considering different percentages of missingness

		Percentages of missingness		
		5%	15%	40%
Structure learning measures	Sensitivity	0.9096±0.0379	0.8269±0.0469	0.6071±0.0558
	Specificity	0.9745±0.0132	0.9513±0.0169	0.8571±0.0246
	ACC	0.9559±0.0188	0.9157±0.0232	0.7857±0.0320
	F-measure	0.9181±0.0342	0.8432±0.0419	0.6190±0.0553
Reliability measures	Unavailability	0.0155±0.0093	0.0265±0.0111	0.0415±0.0169
	RMSE	0.0130±0.0084	0.0234±0.0099	0.0374±0.0149
Imputation measures	Train set ACC	0.7558±0.0359	0.7558±0.0359	0.7562±0.0480
	Test set ACC	0.7540±0.0368	0.7554±0.0362	0.7548±0.0483

4.3 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 Figure 7, "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 Figure 8. Results shown in Table 5, demonstrate that this fault tree has been affected by loss of data more than other two examples, which we suspect the reason is that the events are rare and the system is highly reliable.

```
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";
"BUS1" lambda=4.4444e-6 repair=4;
"BUS2" lambda=4.4444e-6 repair=4;
"FPGA1" lambda=3.003e-9 repair=4;
"FPGA2" lambda=3.003e-9 repair=4;
"PowerSupply1" lambda=1.8182e-5 repair=6;
"PowerSupply2" lambda=1.8182e-5 repair=6;
"PowerSupply3" lambda=1.8182e-5 repair=6;
"WANcard1" lambda=2.5e-6 repair=6;
"WANcard2" lambda=2.5e-6 repair=6;
"GSMRCard1" lambda=5.7078e-6 repair=6;
"GSMRCard2" lambda=5.7078e-6 repair=6;
"CPUboard1" lambda=7.4074e-6 repair=6;
"CPUboard2" lambda=7.4074e-6 repair=6;
"CPUboard3" lambda=7.4074e-6 repair=6;
```

Figure 7: Radio Block Center fault tree

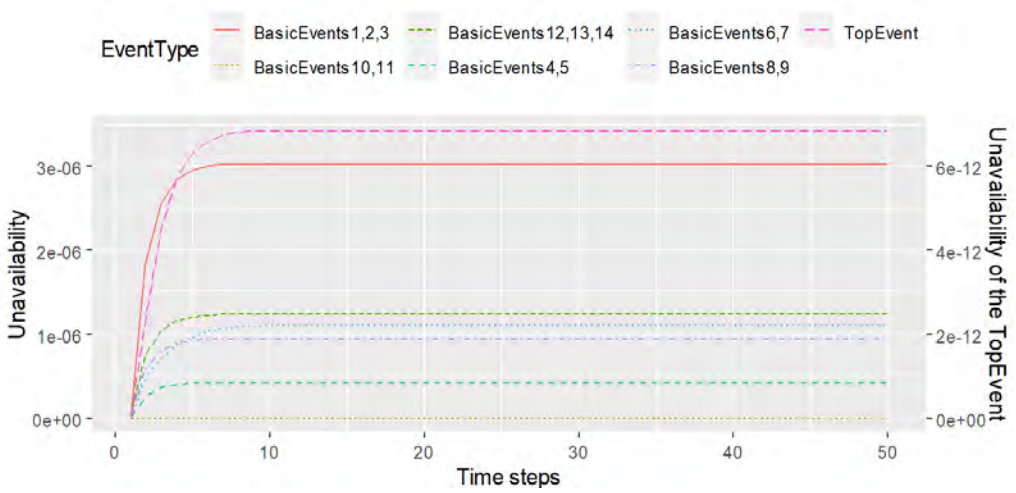


Figure 8: Unavailability values for the RBC fault tree of Figure 7

Table 5: Results of the DDFTAe algorithm for the RBC fault tree considering different percentages of missingness

		Percentages of missingness		
		5%	15%	40%
Structure learning measures	Sensitivity	0.7137±0.0153	0.5662±0.0275	0.2740±0.0029
	Specificity	0.9999±0.0000	0.9999±0.0000	0.9998±0.0000
	ACC	0.9998±0.0000	0.9997±0.0000	0.9995±0.0000
	F-measure	0.8061±0.0173	0.6386±0.0287	0.3266±0.0313
Reliability measures	Unavailability	1.79e-07±8.96e-08	1.06e-07±1.84e-07	2.60e-06±3.29e-07
	RMSE	1.75e-07±8.74e-08	1.03e-06±1.80e-07	2.24e-06±3.40e-07
Imputation measures	Train set ACC	0.9631±0.0003	0.9631±0.0003	0.9632±0.0003
	Test set ACC	0.9631±0.0010	0.9630±0.0006	0.9630±0.0004

5 CONCLUSION

We have presented DDFTAe algorithm, an efficient and novel algorithm for extracting repairable fault trees from incomplete multinomial time series data, that analyses the results to estimate the system's reliability measures. We followed the work of (Lazarova-Molnar et al. 2020) providing flexibility and extensibility to address the issue of missing values in fault occurrences of basic events. We have demonstrated through three case studies that our approach has clear benefits: DDFTAe can extract and analyze multi-state repairable fault trees, compute reliability metrics for distributions other than the usual exponential distribution and handles incomplete binary/multinomial time series data of faults. As future work, we intend to extend the tool to predict the future failure times of the basic components as well as the system, allowing the reliability analysis of systems beyond the historical or real time data sets.

REFERENCES

Aizpurua, J. I., and E. Muxika. 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., A. Griffault, G. Point, and A. Rauzy. 2000. "The Altarica Language and Its Semantics". *Fundamenta Informaticae* 34 (2-3):109-124.

Cox, D. R., and E. J. Snell. 1989. *Analysis of Binary Data*. Florida: CRC press.

Diggle, P. J., P. J. Heagerty, K.-Y. Liang, and S. Zeger. 2002. *Analysis of Longitudinal Data*. Oxford: Oxford University Press.

Fahrmeir, L., and H. Kaufmann. 1987. "Regression Models for Non‐Stationary Categorical Time Series". *Journal of Time Series Analysis* 8 (2):147-160.

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.

Joshi, A., M. P. Heimdahl, S. P. Miller, and M. W. Whalen. 2006. "Model-Based Safety Analysis".Technical Report *NASA/CR-2006-213953*, Advanced Technology Center, Rockwell Collins, Inc., Cedar Rapids, Los Angeles, USA.

Kabir, S. 2017. "An Overview of Fault Tree Analysis and Its Application in Model Based Dependability Analysis". *Expert Systems with Applications* 77:114-135.

Lazarova-Molnar, S. 2005. *The Proxel-Based Method-Formalisation, Analysis and Applications*, Ph.D. thesis, Otto-von-Guericke-University, Magdeburg, Germany.

Lazarova-Molnar, S., and G. Horton. 2003. "Proxel-Based Simulation of Stochastic Petri Nets Containing Immediate Transitions". *Electronic Notes in Theoretical Computer Science* 85 (4):203-217.

Lazarova-Molnar, S., P. Niloofar, and G. K. Barta. 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, Venice, Italy, 1805-1812.

Lee, W.-S., D. L. Grosh, F. A. Tillman, and C. H. Lie. 1985. "Fault Tree Analysis, Methods, and Applications ď a Review". *IEEE Transactions on Reliability* 34 (3):194-203.

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., and G. Levitin. 2003. *Multi-State System Reliability: Assessment, Optimization and Applications.* Singapore, Singapore: World scientific.

Little, R. J., and D. B. Rubin. 2019. *Statistical Analysis with Missing Data.* Hoboken, New Jersey: John Wiley & Sons.

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.

Nauta, M., D. Bucur, and M. Stoelinga. 2018. "Lift: Learning Fault Trees from Observational Data". In *15th International Conference on Quantitative Evaluation of Systems,* September 4th-7th, Beijing, China, 306-322.

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.

Ripley, B., W. Venables, and M. B. Ripley. 2016. "Package 'Nnet'". *R package version* 7:3-12.

Rubin, D. B. 1976. "Inference and Missing Data". *Biometrika* 63 (3):581-592.

Ruijters, E., and M. Stoelinga. 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., S. Kabir, M. Walker, and Y. Papadopoulos. 2016. "Model-Based Dependability Analysis: State-of-the-Art, Challenges, and Future Outlook". *Software Quality Assurance*:251-278.

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 28th March 2021

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.

AUTHOR BIOGRAPHIES

PARISA NILOOFAR is a Postdoctoral Researcher with the Faculty of Engineering at the University of Southern Denmark. Her current research interests include Bayesian networks, simulation and modelling and reliability modeling. Parisa Niloofar obtained her PhD in Statistics, in 2013, specializing in the area of Graphical modeling. Her email address is parni@mmmi.sdu.dk.

SANJA LAZAROVA-MOLNAR is a Professor with the SDU Software Engineering Section at the Faculty of Engineering, University of Southern Denmark, researching in the areas of Modelling, Simulation and Data Analytics, in various decision support contexts. She is a Senior Member of IEEE, and currently serving as Director-at-Large on the Board of Directors of The Society for Modeling & Simulation International (SCS). Her email address is slmo@mmmi.sdu.dk