Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

PROXEL-BASED SIMULATION OF FAULT TREES IN R

Parisa Niloofar

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

Hossein Haghbin

Faculty of Intelligent Systems Engineering
and Data Science
Persian Gulf University
Bushehr, 7516913817, IRAN

Sanja Lazarova-Molnar

Institute of Applied Informatics and Formal Description Methods
Karlsruhe Institute of Technology
Kaiserstr. 89
Karlsruhe, 76133, GERMANY

ABSTRACT

Simulation is an advantageous solution to calculate reliability of complex systems when analytical methods are not able to deliver results in time or when we do not have the resources to apply analytical methods. Proxel-based simulation has shown to be very efficient in reliability analysis of complex systems. In this paper, we present a package called `ftaproxim` for proxel-based simulation of complex systems using fault trees, developed in the R programming language. `ftaproxim` can calculate and plot instantaneous unavailabilities of repairable events and the system as a whole.

1 INTRODUCTION

Proxel-based simulation was introduced by (Horton 2002) as a state-space-based analysis method for Stochastic Petri Nets, which are analyzed using Discrete-Time Markov Chains (DTMC) as underlying models. It computes the probabilities of all potential state changes in a deterministic way just like Markov chains. However, it is not restricted to the use of exponential distributions. Supplementary variables are used for the storage of the age information of enabled state transitions. The probability of state changes is computed by means of the instantaneous rate functions of the associated distributions. The age information is stored together with the discrete state and the probability, forming a Probability Element (short: proxel). The proxel-based simulation method generates and tracks all possible developments of the system behavior of the system for discrete steps over the simulation time. Rare events and the thereby reached system states are guaranteed to be considered. The method has shown to be applicable for the numerical simulation of stochastic Petri Nets, warranty models and Fault Tree Analysis (FTA) (Lazarova-Molnar et al. 2020; Lazarova-Molnar and Horton 2003; Niloofar and Lazarova-Molnar 2021).

Fault trees are a type of graphical models modeled as Directed Acyclic Graphs (DAGs) whose leaves model components failures and whose gates propagate failures (Vesely et al. 1981; Lee et al. 1985; Ruijters and Stoelinga 2015). FTA investigates how component failures lead to system failures and provides

methods and tools to compute a wide range of properties and measures. If the components of a system are repairable, the fault tree that models the system is a repairable fault tree. If the system and its components either completely function or fail, reliability analysis for this system has a binary perspective. 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 (Lisnianski and Levitin 2003; Niloofar and Lazarova-Molnar 2022).

In this paper we present `ftaproxim` for FTA based on proxel-based simulation in R. For this package, a fault tree is provided in terms of its minimal cut sets, along with reliability and maintainability distribution functions of the basic events. `ftaproxim` then calculates instantaneous unavailabilities of the system for every point in its life time. Considering tools developed in R programming language, `FaultTree` (Silkworth 2020) is the only other package for FTA developed in R. There are also other commercial tools for analyzing fault trees like CAFTA (Computer Aided Fault Tree Analysis) (EPRI 2013) and FaultTree+ (Isograph 1986) programmed in other languages. An overview of the capabilities and limitations of some of the tools for FTA are studied in Ruijters and Stoelinga (2015). Among these tools CAFTA can cope with various probability distributions, including normal and uniform distributions. Also several of these programs allow the analysis of dynamic FTs (Dugan et al. 1990). However, unlike `ftaproxim`, they are not able to compute complete transient solutions for repairable or multi-state systems.

The rest of the paper is organized as follows: Section 2 presents proxel-based simulation, Section 3 provides implementation of the methods in R and, in Section 4 we conclude the paper.

2 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. 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 (Figure 1), which is defined as the total time (T).

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

Assume that we are interested in obtaining the instantaneous unavailabilities for an event where the reliability distribution function is Exponential with $\lambda = 0.1$, and the maintainability distribution function is Normal with $\mu = 2$ and $\sigma^2 = 1$. The first step is to simulate the proxels at each time step based on the possible state changes and accordingly update the age intensity vector and calculate its probabilities. We use Δt to denote the size of the time step.

The unavailability at each time step is the sum of the probabilities where the state of the proxel is Failure. Now for the above mentioned distribution functions and $\Delta t=0.1$, p_1, p_2 and p_3 in Figure 1, take the values in Equations (1)-(3), respectively.

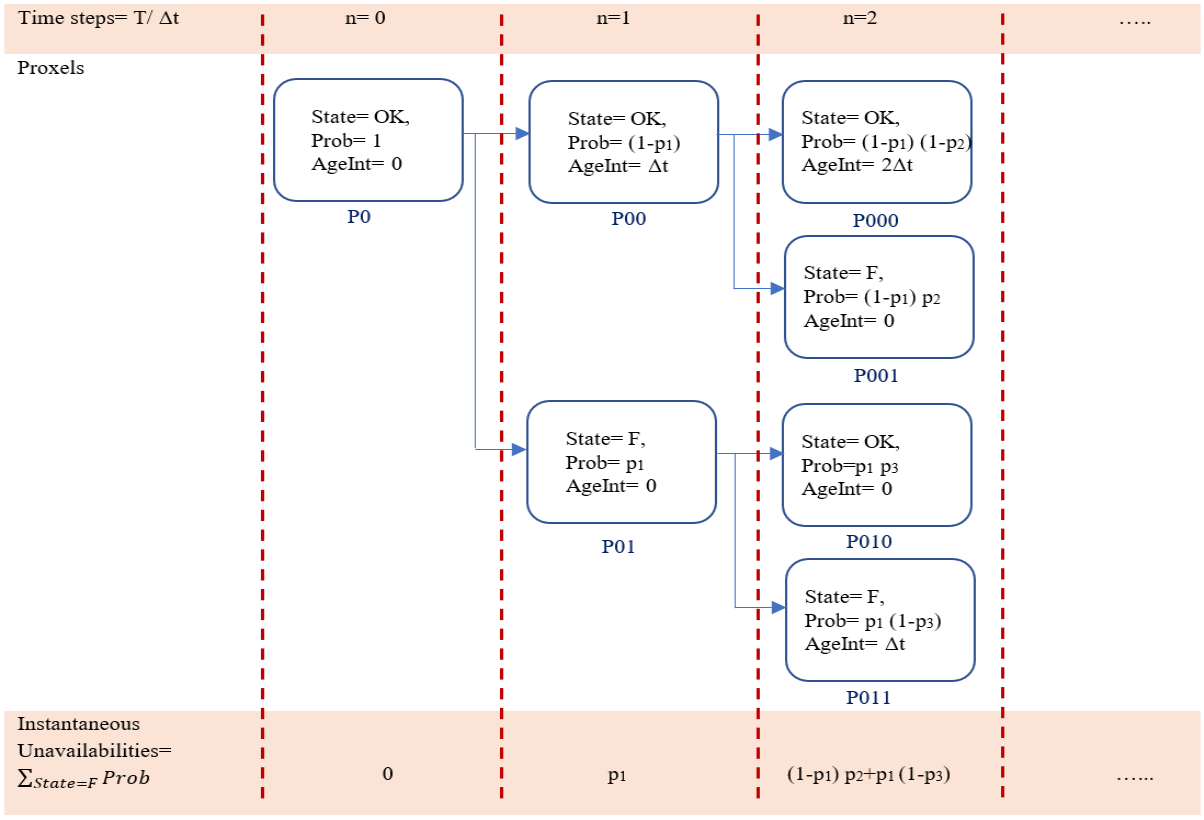


Figure 1: Proxel-based simulation framework.

$$p_1 = \Delta t \times \frac{f(AgeInt_{P_0})}{1-F(AgeInt_{P_0})} = 0.1 \times \frac{0.1e^{-0.1\times 0}}{\int_0^{\infty} 0.1e^{-0.1x}dx} = 0.01, \quad (1)$$

$$p_2 = \Delta t \times \frac{f(AgeInt_{P00})}{1-F(AgeInt_{P00})} = 0.1 \times \frac{0.1e^{-0.1\times 0.1}}{\int_{0.1}^{\infty} 0.1e^{-0.1x}dx} = 0.01, \quad (2)$$

$$p_3 = \Delta t \times \frac{f(AgeInt_{P01})}{1-F(AgeInt_{P01})} = 0.1 \times \frac{\frac{1}{\sqrt{2\pi(1)^2}}e^{-\frac{(0-2)^2}{2(1)^2}}}{\int_0^{\infty}\frac{1}{\sqrt{2\pi(1)^2}}e^{-\frac{(x-2)^2}{2(1)^2}}dx} = 0.0055, \quad (3)$$

where $AgeInt_P$ is the age intensity of proxel P. The unavailability vector for this event is now equal to (0, p1, (1-p1)p2+p1(1-p3),…)= (0, 0.01, 0.0198,…).

Different choices for T and Δt, results in different estimated unavailabilities for the same basic event. For instance, for a basic event with reliability distribution function Exp(0.1) and maintainability distribution function Exp(1), unavailability is calculated as failure rate divided by (failure rate+ repair rate), that is 0.1/(0.1+1)=0.090909 (since both distributions are exponential, true unavailability value can be calculated analytically). Table 1 shows the results for different choices of T and Δt.

Table 1: Different choices for T and Δt affects the simulation results.

Total time	Best Δt	Best unavailability
1	0.01	0.06046
5	0.8, 0.9	0.09091
10	0.1, 0.2,…, 0.9	0.09091

3 IMPLEMENTATION IN R

In this section, we describe in detail how the proxel-based simulation of a given fault tree can be implemented in R (R Core Team 2021) using our developed R package called `ftaproxim`. This package is freely available on GitHub (Haghbin and Niloofar 2021). We begin by introducing and demonstrating the functions for a single basic event in a fault tree, in Section 3.1. In Section 3.2, we describe the implementation for a complete fault tree.

3.1 A Single Basic Event

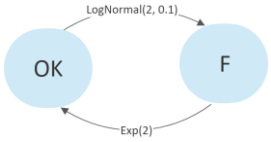
Assuming that R and RStudio are already installed, one can install `ftaproxim` from GitHub through loading `devtools` package (Wickham et al. 2021), running the following codes:

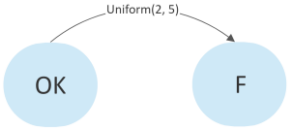
```
# install.packages("devtools")
devtools::install_github("parniSDU/ftaproxim")
library(ftaproxim)
```

Once the package is installed and loaded, the next step is to define the basic events of the system under study. Events can be binary or multi-state, and also repairable or non-repairable. Table 2 shows some typical examples, classified as binary-repairable, binary-nonrepairable, multistate-repairable and multistate-nonrepairable events with their transition probability distribution functions.

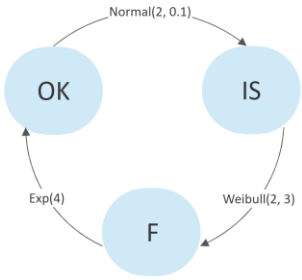
Table 2: Different types of events and the codes in R.

#	Event type	Graph structure	Matrix version	R code

1	Binary-repairable		$\begin{matrix} & \text{OK} & \text{F} \\ \text{OK} & NA & LogNormal(2,0.1) \\ \text{F} & Exp(2) & NA \end{matrix}$	`BE<-list(`
				`  states=c("OK", "F"),`
				`  G=rbind(c(NA, 1),`
				`  c(1,NA)),`
				`  dist=c("lnorm", "exp"),`
				`  param=list(c(2, 0.1), 2)`
				`)`

2	Binary-nonrepairable		$\begin{matrix} & \text{OK} & \text{F} \\ \text{OK} & NA & Uniform(2,5) \\ \text{F} & 0 & 1 \end{matrix}$	`BE<-list(`
				`  states=c("OK", "F"),`
				`  G=rbind(c(NA, 1),`
				`  c(0, 1)),`
				`  dist=c("unif"),`
				`  param=list(c(2, 5))`
				`)`

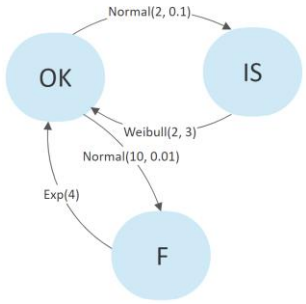
3 Multistate-repairable



$$\begin{array}{c} \\ OK \\ IS \\ F \end{array} \begin{array}{c} \begin{array}{ccc} OK & IS & F \end{array} \\ \begin{bmatrix} NA & Normal(2,0.1) & 0 \\ 0 & NA & Weibull(2,3) \\ Exp(4) & 0 & NA \end{bmatrix} \end{array}$$

```
BE<-list(
  states=c("OK","IS","F"),
  G=rbind(c(NA,1,0),
          c(0,NA,1),
          c(1,0,NA)),
  dist=c("norm", "weibull","exp"),
  param=list(c(2, 0.1), c(2, 3), 4)
)
```

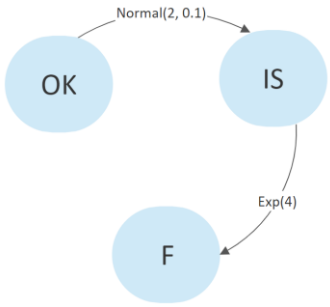
4 Multistate-repairable



$$\begin{array}{c} \\ OK \\ IS \\ F \end{array} \begin{array}{c} \begin{array}{ccc} OK & IS & F \end{array} \\ \begin{bmatrix} NA & Normal(2,0.1) & Normal(10,0.01) \\ Weibull(2,3) & NA & 0 \\ Exp(4) & 0 & NA \end{bmatrix} \end{array}$$

```
BE<-list(
  states=c("OK","IS","F"),
  G=rbind(c(NA,1,1),
          c(1,NA,0),
          c(1,0,NA)),
  dist=c("norm", "norm",
         "weibull","exp"),
  param=list(c(2, 0.1), c(10, 0.01),
             c(2, 3), 4)
)
```

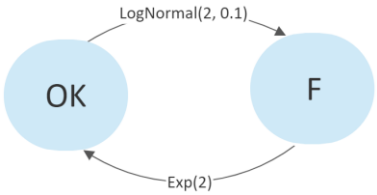
5 Multistate-nonrepairable



$$\begin{array}{c} \\ Working \\ Idle \\ Failed \end{array} \begin{array}{c} \begin{array}{ccc} Working & Idle & Failed \end{array} \\ \begin{bmatrix} NA & Normal(2,0.1) & 0 \\ 0 & NA & Exp(4) \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

```
BE<-list(
  states=c("OK","IS","F"),
  G=rbind(c(NA,1,0),
          c(0,NA,1),
          c(0,0,1)),
  dist=c("norm","exp"),
  param=list(c(2, 0.1), 4)
)
```

Each basic event, denoted as `BE`, is a list of four elements: `states`, `G`, `dist` and `param`, detailed as follows. `states` is a vector of labels indicating the states of the `BE`. The syntax in R for vectors is `c(x, y, …)`. For a binary basic event (i.e., rows 1 and 2 in Table 2) `"OK"` stands for working or functioning state, and `"F"` is for a failed or not functioning state. For multistate events (i.e., rows 3-5 in Table 2), an intermediate state `"IS"` can be added to the states' vector. `G` defines the transition matrix of `BE`. For a basic event with *n* number of states, `G` is a *n* by *n* matrix. The syntax for matrix in R is `rbind(first row, second row,…)` or `matrix(c(x, y, …), nrow=n)`. In this matrix, `0` indicates that no transition is possible, `1` shows that the transition is possible, and the distribution is provided, `NA` means that the transition is possible, but the probability is calculated based on other given distributions in the same row. For example, assume the binary-repairable basic event displayed in Figure 2.



(a)

$$\begin{array}{c} \\ OK \\ F \end{array} \begin{array}{c} \begin{array}{cc} OK & F \end{array} \\ \begin{bmatrix} NA & LogNormal(2,0.1) \\ Exp(2) & NA \end{bmatrix} \end{array}$$

(b)

Figure 2: (a) A binary-repairable basic event with LogNormal failure probability distribution and the repair rate following an exponential distribution (b) Matrix version of the binary-repairable basic event in (a).

For this basic event, the probability of staying in the working state, which is the transition probability from "OK" to "OK" in the next time step (indicated by NA in the first row and first column of the matrix version in Figure 2.(b)) is calculated as in Equation (4).

$$P(state_t = OK | state_{t-1} = OK) = 1 - P(state_t = F | state_{t-1} = OK) \quad (4)$$

$$= 1 - \left(\Delta t \times \frac{f(AgeInt)}{1 - F(AgeInt)} \right),$$

where f and F are the probability density and the cumulative probability functions for the failure probability distribution function LogNormal (transition from "OK" to "F"), respectively. The `dist` element in `BE` is a vector of probability distribution functions, and, finally, `param` is a list of parameters for the probability distributions defined in `dist`. Once an event is defined and the state transitions are specified, instantaneous unavailabilities of the events can be computed using `ProxelBE` function in `ftaproxim`. The user specifies the total time (`totaltime`), step size (`delta`) and tolerance level (`tol`) for the simulation. Proxels with probabilities smaller than the given tolerance level are ignored. The output of `ProxelBE` (here stored in `UnavailabilityBE`) is a numeric vector of length `totaltime` divided by `delta`. Below is the example code for calculating instantaneous unavailabilities of the basic event `BE` in row 1 of Table 2.

```
BE<-list(
  states=c("OK", "F"),
  G=rbind(c(NA, 1),
          c(1,NA)),
  dist=c("lnorm", "exp"),
  param=list(c(2, 0.1), 2)
)

# ProxelBE gains a numeric vector of length totaltime/delta
UnavailabilityBE<-ProxelBE(BE, "F", totaltime=50, delta=0.1, tol=1e-7)
```

For multistate events, the second input, "F", in `ProxelBE` can be replaced by "IS" to obtain the instantaneous probabilities of the intermediate state. Figure 3 illustrates the plot of instantaneous failure probabilities for `BE`. Failure probability for this event at time step 500 equals 0.0578.

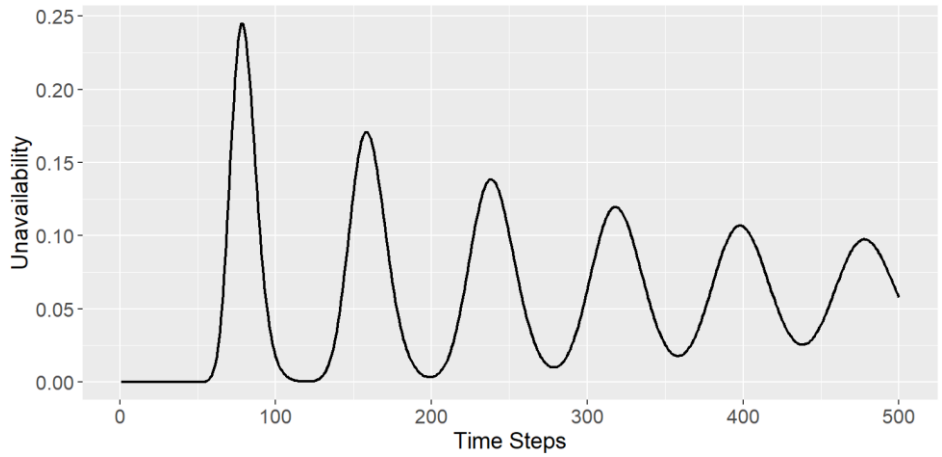


Figure 3: Instantaneous failure probabilities for BE in the first row of Table 2 using ggplot.

The unavailability vector of the multistate-repairable event in row 3 of Table 2, is also illustrated in Figure 4. The unavailability of this event at time step 200 equals 0.0654.

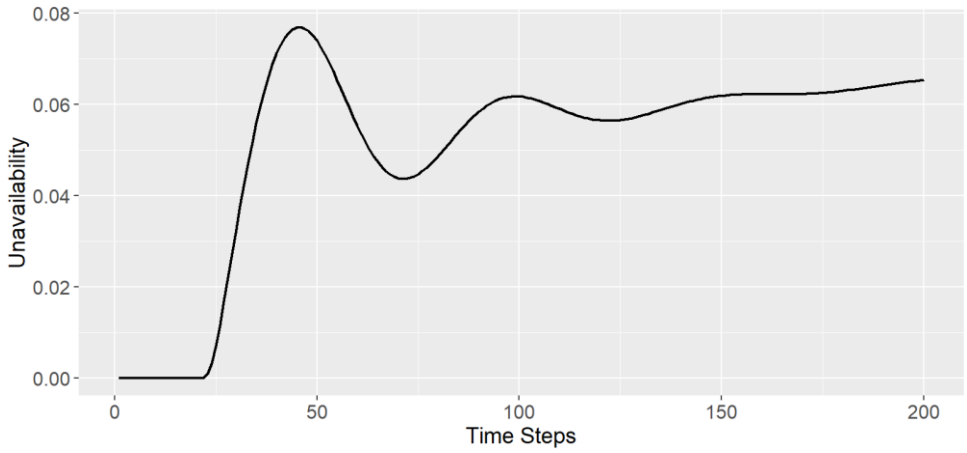


Figure 4: Instantaneous failure probabilities for the multistate-repairable component in row 3 of Table 2.

3.2 A Fault Tree

A fault tree consists of two parts: qualitative part, including the structure of the fault tree in form of a DAG (where no circles are allowed and all the edges are directed); and quantitative part, also known as parameters, which concerns the failure and repair distribution functions of the leaves in the tree.

One way of formulating a fault tree structure is through its minimal cut sets. Cut sets indicate which combinations of event failures lead to system failures. A minimal cut set is a cut set which has no subset that is a cut set. Figure 4 displays a fault tree with 7 basic events and 3 types of gates. The minimal cut sets for this fault tree are: {E, F}, {E, G}, {F, G}, {B, D}, C and A. The logical expression of the fault tree of Figure 5 as a function of its minimal cut sets is given by the following expression:

$$TE= E.F+E.G+F.G+B.D+C+A$$

We assume that A is a multistate-repairable event with the transition probabilities of the event in row 4 of Table 2, and other basic events' transition probabilities follow the distributions in Table 3.

Table 3: Reliability and maintainability distribution functions of the basic events in Figure 5.

Basic events	Reliability distribution	Maintainability distribution
B	Normal(0.1, 1)	Gamma(1, 10)
C	Uniform(0.1, 0.5)	Exp(10)
D	Normal(0.1, 1)	Gamma(1, 10)
E	Exp(0.1)	Weibull(5, 2)
F	Exp(0.1)	Weibull(5, 2)
G	Exp(0.1)	Weibull(5, 2)

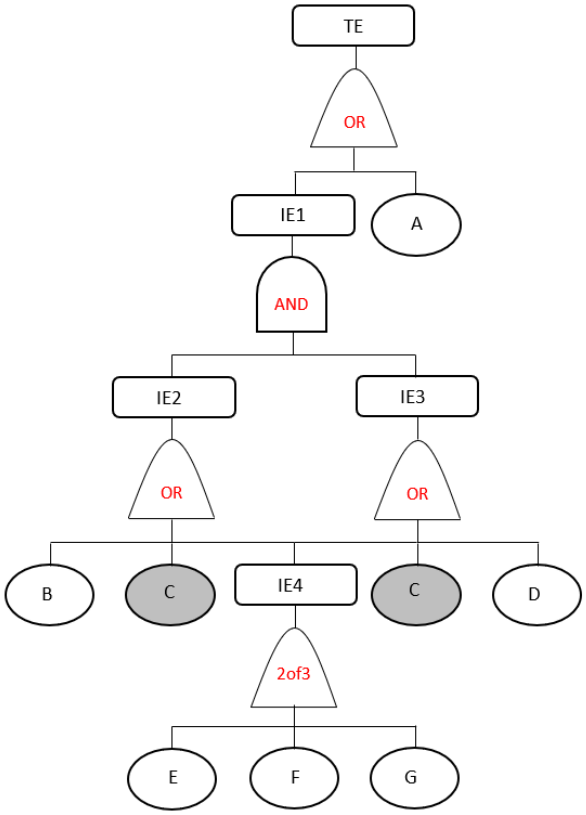


Figure 5: A fault tree with 7 basic events, 4 intermediate events (IE) and 3 types of gates, the basic event C is shaded to indicate that both leaves correspond to the same event.

Once the basic events are defined as instructed in Section 3.1, instantaneous unavailabilities of the basic events along with the top event (system), and their plot can be obtained using `FTUna` function. `FTUna` function returns a list where the first element is a data frame of unavailabilities and the second element is the plot for this data frame. Each column of the data frame belongs to an event, and the number of time steps indicate the number of rows in the data frame. R codes to compute the unavailabilities and their plot, for the fault tree of Figure 5 and the transition probabilities of Table 3, are as follows:

```
BElist<-list(A, B, C, D, E, F, G)
names(BElist)<-c("A", "B", "C", "D", "E", "F", "G")
MCS<-list(c("E", "F"), c("E", "G"), c("F", "G"),
          c("B", "D"), "C", "A")

x<-FTUna(BElist, MCS, totaltime= 10, delta= 0.1, tol= 1e-07)

# Data frame of unavailabilities
x$Unavailability

# Plots using ggplot
x$Plot
```

Figure 6 displays the plot element of the output returned by `FTUna` function. Unavailability of this system is calculated as 0.2718. Computation times (in seconds) on a work station with 16GB RAM and processor Core i7 2.8GHz for the this fault tree with T=10 and Δt =0.1 is 58.07 and for T=5 and Δt =0.1 is 12.68.

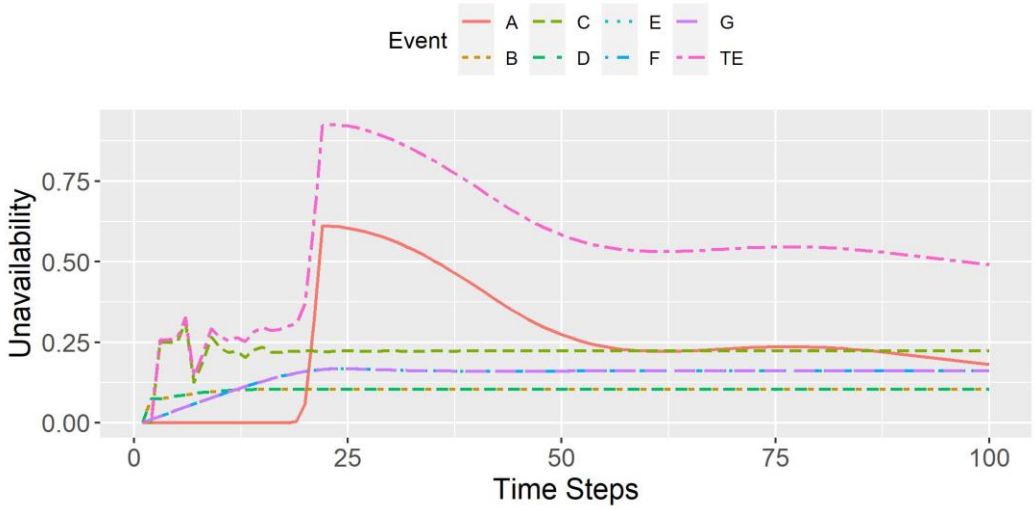


Figure 6: Instantaneous unavailabilities of the basic events and the top event for the fault tree of Figure 5.

4 CONCLUSION

We presented `ftaproxim` library in R programming language that is freely available on GitHub. This library computes instantaneous unavailabilities of basic events and the system using proxel simulation. `ftaproxim` allows users to spot trends in the system's unavailability since it calculates instantaneous unavailabilities of the system for every point in its life time. Proxel-based simulation is not limited to exponential distribution, and neither to binary events. Using `ProxelBE` function, the unavailability vector of a single basic event can be obtained. To compute the same reliability measure for a fault tree where the minimal cut sets are known, `FTUna` function can be utilized. This function also provides a plot of the unavailability vectors for the system and its events. Reliability measures obtained using `ftaproxim` were validated and compared with some of the fault trees studied in the literature while developing the package was under process (Niloofar and Lazarova-Molnar 2021, 2022). For instance, for the Radio Block Center fault tree (Flammini et al. 2005) where the basic events are binary-repairable following exponential distributions, `ftaproxim` results confirm the ones reported in the literature. In general, since complete

transient solutions cannot be calculated using other existing tools, there is no benchmark for the results obtained from `ftaproxim. ftaproxim` can be improved to accommodate dynamic fault trees and parallel computing for fast computation of reliability measures .

REFERENCES

Dugan, J. B., S. J. Bavuso, and M. A. Boyd. 1990. "Fault Trees and Sequence Dependencies". In *Annual Proceedings on Reliability and Maintainability Symposium*, January 23rd-25th, Los Angeles, United States, 286-293.

EPRI. 2013. Cafta (Computer Aided Fault Tree Analysis). http://www.epri.com/abstracts/Pages/ProductAbstract.aspx?ProductId=000000000001015514, accessed 3rd July.

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.

Haghbin, H., and P. Niloofar. 2021. Ftaproxim: Fault Tree Analysis Based on Proxel Simulation. https://github.com/parniSDU/ftaproxim.git, accessed 22nd April.

Horton, G. 2002. "A New Paradigm for the Numerical Simulation of Stochastic Petri Nets with General Firing Times". In *Proceedings of the 14th European Simulation Symposium*, October 23rd-26th, Dresden, Germany, 129-136.

Isograph. 1986. Faulttree+. www.isograph.com/software/reliability-workbench/fault-tree-analysis/, accessed 3rd July.

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 and15th 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-a Review". *IEEE Transactions on Reliability* 34 (3):194-203.

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

Niloofar, P., and S. Lazarova-Molnar. 2021. "Data-Driven Modelling of Repairable Fault Trees from Time Series Data with Missing Information". In *2021 Winter Simulation Conference (WSC)*, December 13th-15th, JW Marriott Desert Ridge, Phoenix, Arizona, United States, 1-12.

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.

R Core Team. 2021. R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing. https://www.R-project.org/, accessed 22nd April.

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.

Silkworth, D. 2020. Faulttree: Fault Trees for Risk and Reliability Analysis. R Package Version 0.99.5. https://CRAN.R-project.org/package=FaultTree, accessed 2nd July.

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.

Wickham, H., J. a. J. Hester, W. Chang, and J. Bryan. 2021. Devtools: Tools to Make Developing R Packages Easier. https://CRAN.R-project.org/package=devtools, accessed 22nd April.

AUTHOR BIOGRAPHIES

PARISA NILOOFAR is an Assistant Professor 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.

HOSSEIN HAGHBIN is an Assistant Professor with the Faculty of Intelligent Systems Engineering and Data Science at the Persian Gulf University in Iran. His current research interests include Data Science, Computational Statistics and Functional data analysis. Hossein Haghbin obtained her PhD in Statistics, in 2012, specializing in the area of Functional time series. His email address is haghbin@pgu.ac.ir.

SANJA LAZAROVA-MOLNAR is a Professor at the Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology. She is also a Professor at the University of Southern Denmark, where she leads the research group Modelling, Simulation and Data Analytics. She is a Senior Member of The Institute of Electrical and Electronics Engineers (IEEE), and currently serving as Director-at-Large on the Board of Directors of The Society for Modeling & Simulation International (SCS). Furthermore, she is Chair of IEEE Denmark and Vice-Chair of IEEE Denmark Women in Engineering Affinity Group. Her email address is sanja.lazarova-molnar@kit.edu.