Graph Entropy on Fault Trees

EMIL ADAM, University of Twente, The Netherlands

A Fault Tree (FT) is a graphical model used in risk management to analyze systems. Graph entropy is a complexity measure that quantifies the structural information of graphical models. Given the limited availability of large, real-world Fault Trees, creating more realistic randomly generated Fault Trees is important for developing and testing dedicated quantitative analysis tools for Fault Trees. This study explores the applicability of graph entropy in capturing the structural complexity of Fault Trees, by comparing randomly generated Fault Trees and real-life Fault Trees from the FFORT dataset, which includes a diverse range of real-world fault trees and other risk models sourced from scientific literature and industrial reports. It is intended as a starting point for enhancing the methods of random Fault Trees. To analyze this, the graph entropy of the in-degree and out-degree distributions was computed and compared across multiple graph sizes. The results in this paper show that real-world Fault Trees show higher entropy values as the graph scale increases, suggesting real-world graphs become structurally more predictable, whereas random Fault Trees show lower entropy values and these tend to stabilize at higher graph sizes. The findings display differences between structures and suggest that current methods of generating Fault Trees do not fully mimic the predictability of real-world Fault Trees. This research highlights the need for improved algorithms for generating Fault Trees.

Additional Key Words and Phrases: Fault Tree, Graph Entropy, Random Graph.

1 Introduction

Risk analysis is crucial to guarantee the safe and reliable operation of critical systems. One of the most used techniques for risk modeling in complex systems is fault tree analysis. A Fault Tree (FT) is a graphical model that represents the various pathways through which failures can propagate within a system, ultimately leading to a critical system-level failure. There is currently a demand for specialized tools for quantitative analysis because of the size and complexity of fault trees. Creating such tools is an important topic of ongoing research [11].

A significant limitation in this field is the small amount of large, real-world FT available for public research. This challenge leaves researchers relying on randomly generated fault trees to test their quantitative analysis tools [6]. Not all randomly generated FTs can reflect the information contained in real-world FTs. The difference between random FTs and real-world FTs is under-researched, making it unclear to what extent random FTs are suitable for testing analysis tools. Thus, to further improve the quality of quantitative analysis tools it is important to realize random FTs.

Graph entropy associates a probability distribution with the elements of a graph and computes the entropy of this distribution using information theoretic formulas like Shannon entropy. The measures capture the amount of information needed to describe the structure of a graphical model. In reality, graph models rarely exhibit properties of random graphs. Therefore, the question arises if we can use graph entropy metrics in the study of random FTs to determine whether this given graph resembles the structure of FTs seen in real-life scenarios.

Author's address: Emil Adam, University of Twente, P.O. Box 217, Enschede, The Netherlands, 7500AE.

This research aims to investigate the applicability of graph entropy measures in capturing the structural complexity of FTs. The goal is to compare randomly generated FTs with real-world FTs and identify key differences that can improve the realism of randomly generated FTs. The results gained from this analysis are intended as a starting point for creating more advanced algorithms for generating random FTs. By computing and comparing these values, differences between the sets have been highlighted.

2 Research Questions

(1) What is the difference between the generated FTs graph entropy values and the values of the real-world FTs?
  a. How are the in-entropy values ($I_{d^-}(G)$) and out-entropy values ($I_{d^+}(G)$) related to the size ($|E(G)|$) and order ($|V(G)|$) of the graph ?
  b. What is the difference in the degree distributions of vertices between the real-world FTs and random FTs ?

(2) How effective is graph entropy at measuring the similarity between the structural complexities of randomly generated FTs and those of real-world FTs?

3 Preliminaries

3.1 Fault Trees

A FT is a directed acyclic graph (DAG) consisting of two types of nodes: events and gates. An event is an occurrence within the system, typically the failure of a (sub) system down to an individual component. The event at the top of the tree is called the top event and represents the system failure being analyzed in the FT. A gate represents how failures in subsystems can combine and cause a system failure [8]. Most commonly used gates in fault trees are **AND** gates, e.g G2 and G3 in Figure 1, and **OR** gates, e.g gate G1 and System in Figure 1. FDEP gates in FTs model the functional dependencies of a system. If the first child of a FDEP gate fails, called trigger, all other children fail with a certain probability. This type of gate is handled differently from the rest, when translating FTs to directed graphs and is discussed more in section 4. While more types of gates exist, this paper focuses on the structure of FTs as directed graphs and does not differentiate other gates based on their functionality.

3.2 Graph Entropy

The entropy of a graph is interpreted as its structural information content and serves as a complexity measure [3]. These measures associate a probability distribution with the elements of a graph and compute the entropy of this distribution using information-theoretic formulas like Shannon entropy. In his work, Dehmer provides a generalized formula for graph entropy [2].

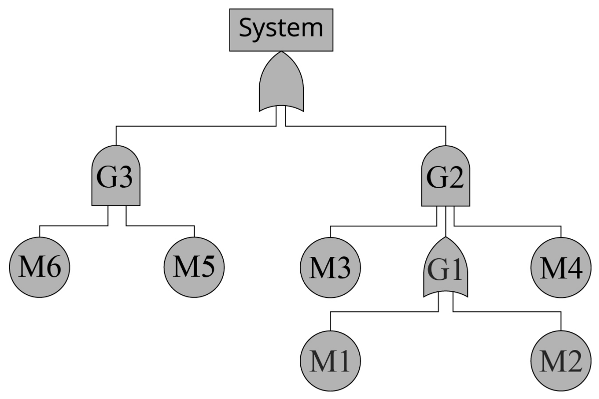

Fig. 1. Example FT

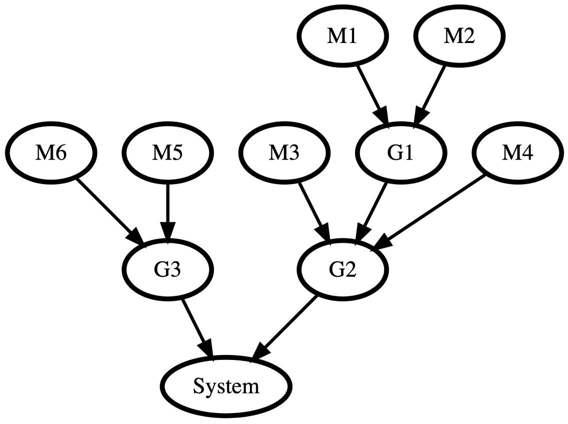

Fig. 2. FT from Figure 1 as a directed graph

Consider a graph $G = (V, E)$ with V as the set of vertices and E as the set of edges. Let f be an arbitrary information functional. The entropy of G with $V = \{v_1, v_2 \ldots, v_n\}$ is defined as follows:

$$I_f(G) = -\sum_{i=1}^{n} \frac{f(v_i)}{\sum_{j=1}^{n} f(v_j)} \log\left(\frac{f(v_i)}{\sum_{j=1}^{n} f(v_j)}\right)$$

where $n = |V|$ denotes the number of vertices, and v_i represents an individual vertex in V.

In this paper, we apply this formula specifically to the in-degree and out-degree of vertices. The in-degree $d^-(v_i)$ of a vertex is the number of edges that point to that vertex, starting from other vertices. The out-degree $d^+(v_i)$ of a vertex is the number of edges that originate at the given vertex and point to other vertices. For example, in Figure 2 the out-degree of G2 is 1, and the in-degree of G2 is 3.
Note that

$$\sum_{i=1}^{n} d^-(v_i) = \sum_{i=1}^{n} d^+(v_i) = m$$

where $m = |E(G)|$ represents the total number of edges in G. Therefore, the normalized sequence of degrees in a graph

$$\left[\frac{d^+(v_1)}{m}, \frac{d^+(v_2)}{m}, \ldots, \frac{d^+(v_n)}{m}\right]$$

is a probability distribution as its elements sum to 1. For f equal to the in-degree, the entropy formula becomes:

$$I_{d^-}(G) = -\sum_{i=1}^{n} \frac{d^-(v_i)}{m} \log_2\left(\frac{d^-(v_i)}{m}\right)$$

and similarly for out-degree entropy:

$$I_{d^+}(G) = -\sum_{i=1}^{n} \frac{d^+(v_i)}{m} \log_2\left(\frac{d^+(v_i)}{m}\right)$$

Due to $\log_2 0$ not being defined, the convention $0 \log_2 0 = 0$ was used to compute the in-degree and out-degree entropy. This ensures that vertices with degree 0 do not contribute to the sum of the entropy.

Example For the DAG in Figure 2, has in-degree sequence [2, 2, 3, 0, 2, 0, 0, 0]. Substituting in the in-degree formula:

$$I_{d^-}(G) = -\left(\frac{2}{9}\log_2\frac{2}{9} + \frac{2}{9}\log_2\frac{2}{9} + \frac{3}{9}\log_2\frac{3}{9} + \frac{2}{9}\log_2\frac{2}{9}\right) \approx 1.968 \text{ bits}$$

It is important to note that the value of degree entropy is maximized for a fixed m when the sequence of degrees is uniform, each vertex having the same degree.

4 Methodology

This section will detail the steps and tools used to complete the research.

4.1 Tools

For conducting the experiments Python was used, mainly for its wide choice of easy-to-use libraries. To easily read and manipulate data in directed graphs, the module graph.py [1] was used. One modified version of a script from the paper [1] project artifact was used for reading and parsing fault tree files to our data structure. For creating the plots, the libraries maplotlib [5] and numpy [4] were used.

4.2 Fault Tree Instance

4.2.1 Real-World FT instances The real-world FT instances used in the experiments were taken from the dataset known as FFORT [9]. FFORT is a collection comprising various risk models including Fault Trees, Attack Trees, and BDMPs sourced from scientific literature and open industrial reports. Additionally, some FT instances were found in literature. This is the collection of real-world FTs used for the comparison:

- 98 FT instances from the FFORT dataset
- 3 industrial FT models for components of a lock used in water navigation

[1]Module for working with directed and undirected multigraphs. Version: 29-01-2015 by P. Bonsma. Version: 01-02-2017, P. Bos, T. Bontekoe

- 3 FT models of wet-pipe fire sprinkler systems used in Australian shopping centers [7]

The order of the graphs in the set ranges from 11 to 414 vertices. Instances with comparable orders were grouped to represent the real-world FT set. The data sets obtained contain graphs with approximately order 10, 15, 25, 35, 45, 100, 150, 200, 250, 300.

4.2.2 Random FT instances Initially, a set of 128 random FT instances was used from the [1] project's artifact. These instances have sizes ranging from approximately 220 vertices to 270. Due to the differences between the graph sizes of the real-world instance sets, these were not enough to perform a comparison. A Python program was created to generate random instances for this experiment. As this paper focuses on fault trees as directed graphs, the algorithm outputs randomly generated rooted directed acyclic graphs (DAG). The function has 4 parameters: the size of the resulting graph, the maximum possible number of children for the root vertex, the maximum possible number of children for the rest of the vertices, and the maximum possible out-degree. The algorithm iterates over the depth levels and vertices of the graph starting with the root, generates a number depending on the range given in the parameters, and creates the same amount of children vertices for this given vertex. After the first loop, it iterates again and creates a randomly generated number of edges between neighboring levels without creating cycles. The graphs generated by this algorithm are stored in DOT format. To find adequate parameters values for the generation algorithm, the same metrics where computed for the real-world sets. Following the graph size of the real-world FTs groups, 1000 random graphs with the same number of vertices were generated using as parameters the values observed in the their real-world counterparts.

4.3 Converting real-world Fault Trees to Directed Graphs

The graphs from the set of real-world FTs are structured in standard Galileo Format [10]. The authors of [1] have created a program that defines a grammar based on the Galileo Format, parses the FT files and creates a mapping, with gates as keys and their children (BE or gates) as values. For this research, the program was modified to build a graph object using the graph.py module. For BEs and gates, vertex objects are created. Afterward, edge objects are created according to the mappings in the dictionary, with the key gate as the tail, and each value element as the head of the edge. Gates of type FDEP are treated differently from the other types of gates. These have as the first element in the mapping their trigger event, followed by states failures can propagate to next. Therefore, an edge connecting the trigger event is added to the fdep vertex. The rest of the states left in the values of the mapping are treated the same as the other types of gate.

4.4 Calculating Degree Entropy

For calculating the in-degree entropy and out-degree entropy a fairly simple Python program was written following the entropy formula. A pseudocode of the calculation can be seen in Algorithm 1.

Algorithm 1 Calculate In-Degree Entropy of a Graph

```
1:  procedure InDegreeEntropy($Graph$)
2:      $m \leftarrow$ length of $Graph.edges$
3:      $entropy \leftarrow 0$
4:      for $vertex$ in $Graph.vertices$ do
5:          $degree\_fraction \leftarrow vertex.in\_degree/m$
6:          if $degree\_fraction > 0$ then
7:              $entropy \leftarrow entropy - (degree\_fraction \times \log_2(degree\_fraction))$
8:          end if
9:      end for
10: end procedure
```

Similarly, the algorithm for the out-entropy has the same logic. Furthermore, the sequences of degrees computed were used to plot the degree distributions of vertices. The average entropy was computed to plot the relation between the size of the graphs and the degree entropy values per order group of graphs.

5 Results

5.1 Comparison of Entropy Values

This section will present and discuss the results for each research question. The code used for this research can be found at : https://gitlab.utwente.nl/s2811731/graph-entropy-on-fault-trees.
Figures 3, 4, 5, and 6 present the average in-degree and out-degree entropy values across the real world and randomly generated FT instances with various graph orders and sizes. The entropy curves in the figures resemble logarithmic growth, showing a rapid increase at first, then gradually slowing down. In Fig 4 and Fig 6, it is observed that the out-entropy value on randomly generated graphs stabilizes approximately 5-6 bits after a size of approximately 150-200 edges. The same pattern is observed for the in-entropy. On the other hand, real-world FTs entropy values continue to increase with size. Furthermore, in Fig. 3 and Fig. 5, a higher magnitude is observed in the in-degree and out-degree entropy values as the graph grows. Overall, real-world FTs exhibit higher in-degree and out-degree entropy values on big graphs. This indicates that the sequence of degrees in real-world FTs more uniform.

Figures 7, 8, 9, and 10 present the distributions of vertices based on their in-degree and out-degree for a specific real-world FT, and one comparable randomly generated graph, further highlighting differences between their structures. The distribution based on in-degree on real FTs, shown in Figure 7, is highly skewed, most vertices having low in-degree values and few with higher in-degree values. In contrast, Fig 9. shows a more even distribution of in-degrees across vertices on random graphs. Note that vertices with degree 0 do not contribute to the value of the degree entropy, hence, they are not relevant in this comparison. These results reinforce the observations that FTs show predictability in their growth.
The out-degree distribution on real-world FTs, shown in Figure 8, displays an even more pronounced skewness, with the majority of vertices having an out-degree of one. Since gates in Fault Trees, except FDEP, have only one out-degree, these results indicate that

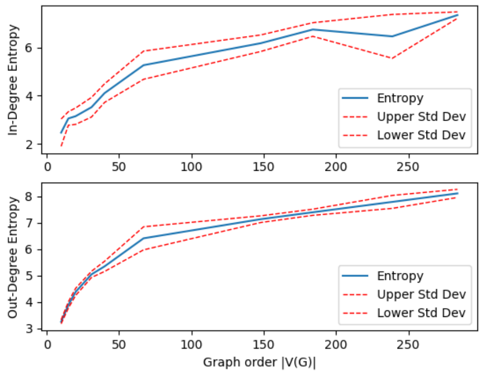

Fig. 3. Relation between (I_{d^-}) and (I_{d^+}) and graph order (n) on real-world FTs

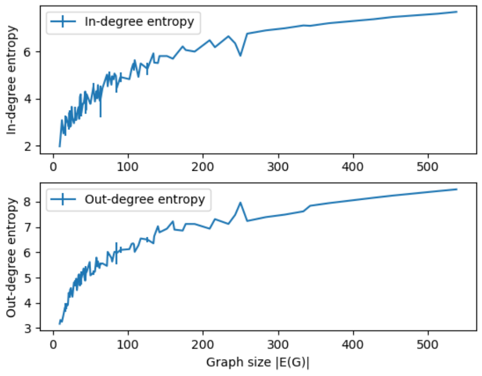

Fig. 5. Relation between (I_{d^-}) and (I_{d^+}) and graph size (m) on real-world FTs

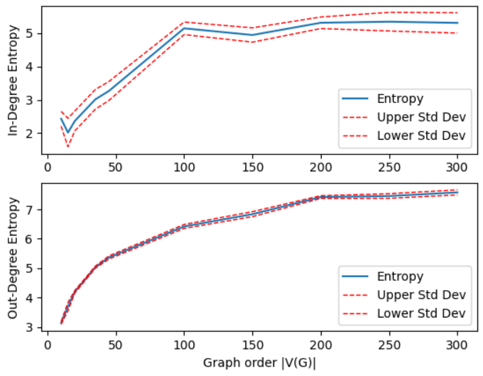

Fig. 4. Relation between (I_{d^-}) and (I_{d^+}) and graph order (n) on randomly generated DAGs

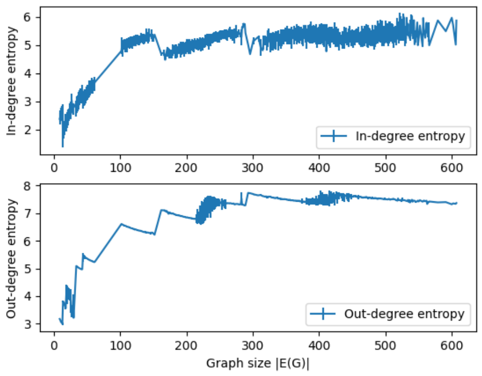

Fig. 6. Relation between (I_{d^-}) and (I_{d^+}) and graph size (m) on randomly generated DAGs

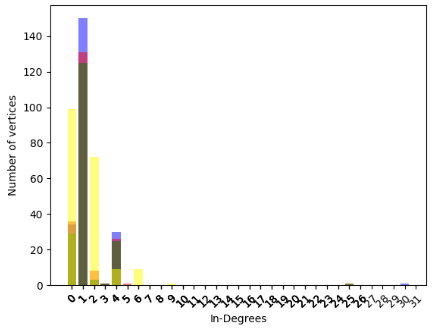

Fig. 7. Distribution of vertices based on their in-degree in real-world FTs with $n = 200$

occurrences of Basic Events with multiple out-degrees are rare in real-world FTs.

These patterns are consistently observed across larger real-world FT instances, implying that FTs become more predictable as their size increases.

6 Conclusion

This paper has analyzed the entropy values of in-degree and out-degree distributions for real-world FTs and randomly generated FTs. It found that real-world FTs and randomly generated are in terms of predictability in their structural complexity. Real FTs generally display higher entropy values and faster growth as graph size increases, suggesting that their degree sequence becomes more uniform and their growth is predictable. In-entropy and out-entropy values on randomly generated graphs show a similar logarithmic growth, however, they stabilize at the approximate value of m = 200 and are lower than their real counterparts indicating a high variety of different in-degrees and out-degrees. The consistent patterns

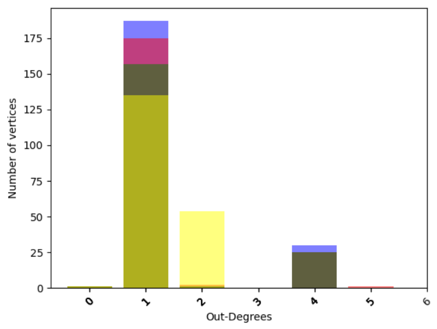

Fig. 8. Distribution of vertices based on their out-degree on real-world FTs with $n = 200$

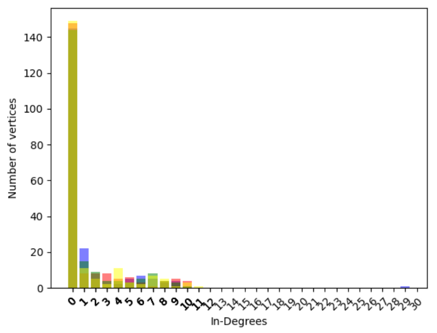

Fig. 9. Distribution of vertices based on their in-degree on 4 randomly generated DAGs with $n = 200$

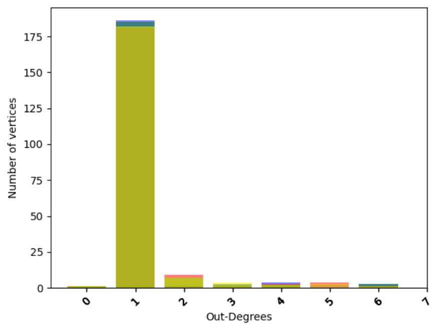

Fig. 10. Distribution of vertices based on their out-degree on 4 randomly generated DAGs with $n = 200$

observed in the real-world FT's entropy values demonstrate that graph entropy can be effective at differentiating random graphs with higher variability in in-degrees and out-degrees from real FTs.

7 Future Work

The generation algorithm can be improved to include a target in-degree and out-degree value of the resulting graph. The calculation of the current entropy values can be done every iteration Different weighted probabilities could be used to select the number of new vertices and edges added to the graph, effectively steering the entropy value to the desired range. Moreover, it can include more structural constraints observed in real-world FTs, such as the ratio of certain types of gates or the number of gates and basic events. Future studies may look into other entropy-based metrics that capture different aspects of graph complexity. These measures could provide a better understanding of real-world FTs structures.

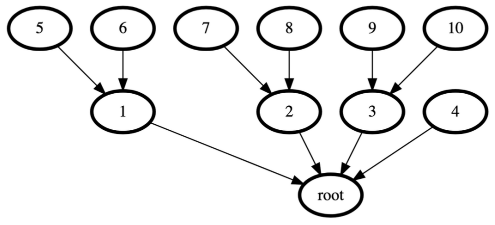

Fig. 11. Example directed graph

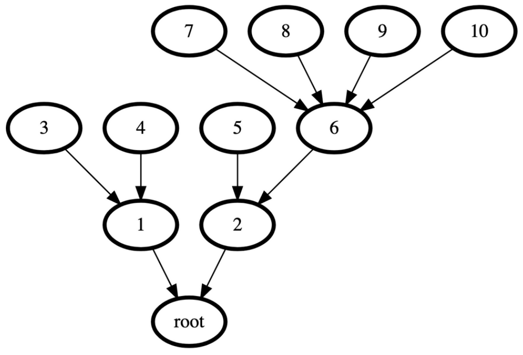

Fig. 12. Example directed graph

8 Limitations

One significant limitation of this research is the small amount of big real-world FTs in the dataset. This scarcity can impact the conclusions drawn on the structural features of FTs. Furthermore, differentiating specific FT structures can be difficult only with entropy. For example, many graphs can have the same sequence of in-degrees and out-degrees but be structurally significantly different. One such example is shown in Figure 11 and Figure 12. Naturally, these 2 graphs will have the same in-degree and out-degree entropy values and are interpreted the same. This is to show that entropy metrics may not capture all the important structural differences in FTs.

References

[1] D. Basgöze, M. Volk, J.P. Katoen, S. Khan, and M. Stoelinga. *BDDs Strike Back: Efficient Analysis of Static and Dynamic Fault Trees*, pages 713–732. Lecture notes in computer science. Springer, Germany, May 2022. Funding Information: This work has been partially funded by NWO under the grant PrimaVera number NWA.1160.18.238, European Union's Horizon 2020 research and innovation programme under the Marie Sk lodowska-Curie grant agreement No. 101008233 (Mission), and the ERC Consolidator Grant 864075 (CAESAR). Khan is funded by a HEC-DAAD stipend. Publisher Copyright: © 2022, Springer Nature Switzerland AG.; 14th International Symposium NASA Formal Methods, NFM 2022, NFM 2022 ; Conference date: 24-05-2022 Through 27-05-2022.

[2] M. Dehmer. Information processing in complex networks: Graph entropy and information functionals. *Applied Mathematics and Computation*, 201(1):82–94, 2008.

[3] M. Dehmer and A. Mowshowitz. A history of graph entropy measures. *Information Sciences*, 181(1):57–78, 2011.

[4] C. R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[5] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[6] M. Lopuhaä-Zwakenberg. Fault tree reliability analysis via squarefree polynomials. In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering - MODELSWARD*, pages 39–49. INSTICC, SciTePress, 2024.

[7] K. A.M. Moinuddin, J. Innocent, and K. Keshavarz. Reliability of sprinkler system in australian shopping centres –a fault tree analysis. *Fire Safety Journal*, 105:204–215, 2019.

[8] E. Ruijters and M. Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15-16:29–62, 2015.

[9] E. J.J. Ruijters, C. E. Budde, Muhammad C. N., M. I.A. Stoelinga, D. Bucur, D. Hiemstra, and S. Schivo. Ffort: A benchmark suite for fault tree analysis. In M. Beer and E. Zio, editors, *ESREL 2019: Proceedings of the 29th European Safety and Reliability Conference*, pages 878–885. Research Publishing, 2019. 29th European Safety and Reliability Conference, ESREL 2019, ESREL 2019 ; Conference date: 22-09-2019 Through 26-09-2019.

[10] K. Sullivan, J. Dugan, and D. Coppit. The galileo fault tree analysis tool. pages 232–235, 02 1999.

[11] M. Volk, F. Sher, JP. Katoen, and M. Stoelinga. Safest: Fault tree analysis via probabilistic model checking. 2024.

9 Appendix

During the preparation of this work the author(s) used ChatGPT to refresh knowledge on usage of different python libraries like matplot, numpy etc. After using this tool/service, the author(s) reviewed and edited the content as needed and takes full responsibility for the content of the work.