Understanding the application of Multi-Objective Evolutionary Algorithms to the Inference of Fault Tree Models

BOGDAN COLȚA, University of Twente, The Netherlands
LISANDRO A. JIMENEZ-ROA, Formal Methods and Tools, University of Twente, The Netherlands
MARIËLLE STOELINGA, Formal Methods and Tools, University of Twente, The Netherlands

ABSTRACT

Fault Tree Analysis (FTA) is a recognized method in reliability engineering and risk assessment that manages systems by providing a structured depiction of how failures propagate and offering quantitative and qualitative metrics. Several challenges associated with FTA relate to model construction, which can be time-consuming and error-prone. Several algorithms have been proposed to address this for the automatic inference of Fault Trees.

Within the state-of-the-art algorithms is FT-MOEA, which utilizes multi-objective evolutionary algorithms to construct compact and efficient Fault Tree structures from failure datasets. However, a significant challenge FT-MOEA faces relates to scalability. The goal of this research is to further focus on investigating the influence of genetic operators on the convergence of the algorithm.

The paper proposes an extension to the algorithm's implementation that analyzes each step of the evolution process by collecting the metrics of each FT obtained and the genetic operators applied. Moreover, the paper suggests some analysis metrics that describe the performance and efficiency of genetic operators.

1 INTRODUCTION

Fault tree analysis (FTA) represents a systematic method in reliability engineering and risk analysis used to model an overview of a system and how it behaves in the event of failure. The main advantage of FTA is that it enables modeling complex systems by encoding and displaying logical relationships in a more intuitive visualization and also calculating the system and subsystem failure probabilities [1].

However, a major disadvantage of FTs is related to their construction, which domain experts generally conduct in a manual procedure, which can result in a costly and time-consuming task. Moreover, the presence of manual execution can manifest human bias, which might result in the presence of inconsistencies and incompleteness [2].

As this challenge has become the subject of ample research, it has been referred to in literature in different ways. In this paper, we refer to this as the automatic inference of Fault Tree models [6], which means the process of automatically creating an FT model from knowledge or data about a system.

TScIT 41, July 5, 2024, Enschede, The Netherlands

This research focuses on a data-driven model, analyzing each decision made inside an already established algorithm that uses a failure data set to infer a Fault Tree that correctly encodes the boolean logic that describes failure propagation through the system.

Approach. The research approach is collecting data about the input and output of each generation of a multi-objective evolutionary algorithm so that it can be presented systematically and analyzed. This is done by storing all the information in a graph structure, where each node contains information about a specific FT. Such a data structure allows an easy way to trace back each step of the evolution that a specific FT has passed through. Once the graph structure is completed, the stored data can be converted to a table structure and presented in a human-readable and systematic way. Further, this data is analyzed using statistics to determine the impact of each genetic operator.

Contributions. The primary contributions of the paper are as follows:

(1) Designing and implementing an algorithm that uses a graph structure and employs a Breadth-Search First traversal algorithm to store and describe the evolution of the FTs.
(2) An extension to the FT-MOEA-CM's implementation that stores information about each generation of the evolution and presents it systematically for further analysis.
(3) An analysis of the impact of each genetic operator on the evolutionary process using proposed analysis metrics.

Paper outline. In Section 2, technical concepts required to understand the paper are explained. Afterward, in Section 3, the research questions are presented, followed by a methodology section (Section 4) that shows the methods used during the research. Moreover, Section 5 presents and interprets the findings, followed by Section 6, where methods to analyze the collected data in the future are presented. In the end, Section 7 wraps up the paper.

Related Work. The literature discusses three main FT model inference approaches: data-driven, model-based, and knowledge-based [3]. Data-driven approaches use failure datasets to infer FTs. Model-based methods consist of converting already existing models, such as UML activity diagrams [4], into FTs. Knowledge-based methods are based on information provided by domain experts on components and their relationships.

Evolutionary algorithms have shown promise in improving the efficiency of automatic FT inference. Linard et al. [5] made the first attempt in this direction. The paper introduces a Genetic Algorithm for inferring FTs from failure data sets, using accuracy as its only objective. However, since size and other metrics were not considered, the convergence was slow and the resulting FT could be too large.

To tackle this problem, the paper by Jimenez-Roa [6] proposes using an MOEA to infer FTs from failure data sets. The algorithm uses multiple objectives simultaneously: MCS, accuracy, and size. The NSGA-II sorting algorithm [7] and the Crowding-Distance [8] are employed to determine the Pareto Front for additional metrics. As a result, the algorithm leads to smaller FTs and faster convergence.

As already mentioned, this research focuses on data-driven FT inference. In particular, we explore ways of improving the efficiency of FT-MOEA [6], which employs multi-objective evolutionary algorithms.

Despite the extensions on FT-MOEA and other automated FT inference methods, scalability has remained a significant issue for the algorithm's performance. With the increased development of modern systems, the continuously growing volumes of failure data sets will continue to become a more significant issue for the current algorithm over time.

Progress in this direction has been made through FT-MOEA-CM [11], and SymLearn [12]. For FT-MOEA-CM, the authors have worked on lowering the number of generations and the time the algorithm takes to converge. The results have shown that by replacing a Minimal Cut Sets metric with a Confusion Matrix metric, significant improvements in converging to the global optima can be achieved more consistently.

2 BACKGROUND

2.1 Fault Tree Analysis

Fault Tree Analysis (FTA) is a popular method in reliability engineering, which provides a representation of the decisions made in a system and how failure propagates through it. A Fault Tree (FT), Fig. 1, is formally defined as a 5-tuple of building blocks, which are the following:

(1) Basic Events (BE). These events have a probability of happening and are the leaves of the FT.
(2) Gates. These represent the logical operations that are applied to the BEs. For the scope of this research, only Or and And gates are considered.
(3) Intermediate events (IE). These are the output of the Gates.
(4) Top event (TE). It is the event to which all the BEs lead, and it is represented as the root of the FT.
(5) (FT) Element. It refers to a BE or a Gate.

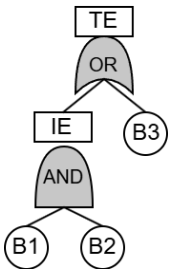

Fig. 1. Inferred FT from Table 1 dataset

2.2 Multi-Objective Evolutionary Algorithms

Multi-Objective Evolutionary Algorithms (MOEAs) are evolutionary algorithms that are based on the same idea of biological evolution [13]. This means they start with a population that is continuously subject to mutation operations, selection procedures, and fitness functions. Fig. 2 depicts an example of an evolutionary algorithm, FT-MOEA.

2.3 Failure Data Set

The failure data sets used as input for the MOEA consist of possible combinations of values for each BE and the resulting Top Event. Table 1 presents an example of a failure data set.

Table 1. Toy Input Failure Data Set

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

2.4 Mann-Whitney U Test

The Mann-Whitney U Test is a non-parametric test of the Null Hypothesis. It is generally used to determine whether there is a significant difference between the values of two sets. In the scope of this research, this test is used to determine whether there is a significant difference in the number of FTs passed to the next generation for each genetic operator.

2.5 Pearson Correlation Coefficient

Pearson correlation coefficient calculates the linear correlation between two data sets. In the scope of this research, this coefficient is used to measure the correlation between the number of genetic operators applied along each generation and the metrics obtained for each FT in the final generation.

2.6 Breadth-First Search Traversal Algorithm

Breadth-First Search (BFS) is an algorithm for searching a graph data structure. It starts with the graph's root and goes to all the nodes at the current level before moving on to the next level. This algorithm is used to traverse the data structure containing information about each FT, as each generation will represent a level inside the graph. The implementation will use this algorithm to generate all the data sets about the overall evolutionary process.

The graph data structure is only used in the implementation to reduce the program's complexity. Modifying the data sets during the run of FT-MOEA would become computationally expensive, as a look-up algorithm would be needed for this. Also, the data stored in the graph cannot be used directly for statistics computation, as it needs to be aggregated from different graph nodes.

The advantage of such an approach is that it allows us to trace back all the parents of an FT and the genetic operators applied to see how an FT evolved over the generations. Another advantage of using this algorithm is that it is possible to check the path from a source node to a destination node. This means that the steps of how one FT became another FT can be seen through the evolutionary process.

3 PROBLEM STATEMENT

The convergence time of the current implementation of FT-MOEA can increase fast, and scalability will become an issue when adding new basic events to the failure data sets [6]. Accordingly, there is a need for further research to address these issues.

An approach with the potential to aid scalability involves a thorough analysis of the behavior of multi-objective evolutionary algorithms (MOEAs). In the current implementation, genetic operators are applied randomly, leaving it unclear whether applying them differently increases the chance of finding the global optimum and reaching convergence faster.

A detailed analysis of FT-MOEA-CM's behavior can provide insights into the difference in the efficiency of the genetic operators. These insights could be applied in future research to enhance the algorithm's performance by guiding the application of genetic operators.

3.1 Research Questions

The research questions of this study are:

RQ1. To what extent do the genetic operators impact the convergence and the number of generations in multi-objective evolutionary algorithms to infer Fault Tree models from failure datasets?

RQ2. Considering there exists a pattern between genetic operators and convergence of the multi-objective evolutionary algorithms, which specific operators have the most impact and which have the least?

4 METHODOLOGY

4.1 Inferring Fault Tree models via Multi-Objective Evolutionary Algorithms

This section will describe the depicted steps in Fig. 2 that the FT-MOEA follows to infer FT models.

Step 1. The initial population is used as the first generation by the algorithm. The FT-MOEA is initialized with the two parent fault FTs created from the failure data sets, wherein all the events are connected to a single OR gate and another to an AND gate.

Step 2. The mutation operations are used on the current generation to create a new generation, which might have better or worse characteristics. In FT-MOEA, the genetic operators are applied randomly to the structure of the FTs.

Step 3. To decide whether the next generation exhibits better qualities, a fitness function is used to assess the characteristics of the individuals. A sorting algorithm is applied to FT-MOEA.

Step 4. The selection procedure represents a method to select which members of the current generation will move on to the next one by selecting the non-dominated individuals.

Step 5. This process will occur multiple times until the convergence criteria are met, and the last generation will contain the inferred FT.

The genetic operators used by the algorithm are the following:

- Disconnect BE. A basic event is disconnected from the tree.
- Connect BE. Connect to a gate a disconnected basic event.
- Change Gate Type. Take a random gate and change it to the other type.
- Create BE. Randomly creates a basic event under an existing gate.
- Create Gate. Randomly creates an And or Or gate under an existing gate.
- Delete BE. Randomly deletes a basic event.
- Delete Gate. Deletes a gate and its children from the tree.
- Move BE. Randomly takes one basic event and moves it under a different gate.
- Cross-over. Randomly choose two FTs in the offspring generation and exchange an element between them.
- Change gate. Randomly select a gate and change it to its opposite type.

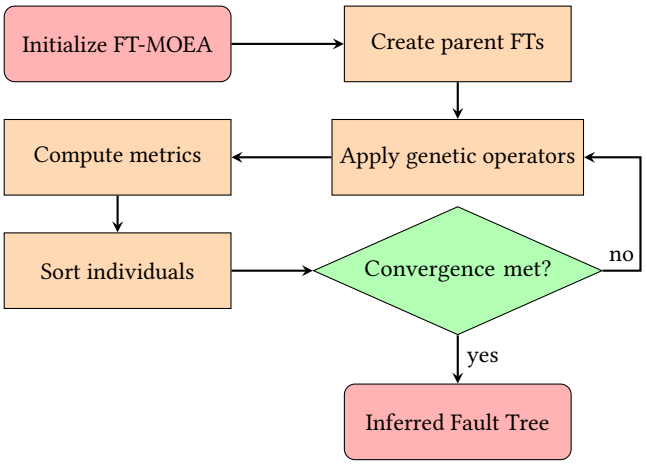

Fig. 2. Flow diagram describing the FT-MOEA process to infer FTs from a failure data set.

4.2 Experiment setup

The current implementation of FT-MOEA-CM will be used as the basis for the experiments to determine the performance and impact of genetic operators. The algorithm is configured to use the following performance metrics: Matthews correlation coef., Specificity, Negative predictive value, Precision, Diagnostic odds ratio, FT Size, Accuracy. The enumerated metrics are used since they are the most informative [11].

Case Studies. The algorithm is evaluated on 3 FTs stemming from various application areas. The CSD [14] dataset was obtained from a Container Seal Design. The Data-driven Fault Tree (DDFT) [15]

was obtained from time series data. The COVID-19 FT [16] is used in infection risk management. Table 2 outlines for each case study the number of unique Basic Events (w), the number of total Basic Events (W), the number of Or gates (#Or), the number of And gates (#And) and the number of rows in the dataset (2^w).

Table 2. The data sets used during testing and their associated relevant information

Case Study	w	W	#Or	#And	O(2^w)
CSD [14]	6	6	2	2	64
DDFT [15]	8	8	1	3	256
COVID-19 [16]	9	21	3	9	512

The number of: unique BEs(w), total BEs(W), And gates(#And), Or gates(#Or) and complexity(O(2^w))

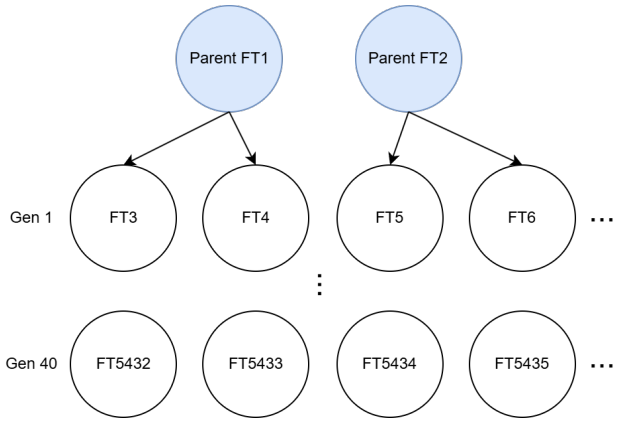

Fig. 3. Directed graph that stores the information about each FT of every generation

Implementation. An extension of the FT-MOEA-CM's implementation was designed to register every FT created in each generation, its metrics, and the operators applied. The implementation is available online [1] and complemented by some Jupyter Notebooks. These notebooks are used to create the plots from the Data Frames obtained after running the algorithm. Also, inside the repository, the data sets generated five times for each case study can be found.

Generation of Failure Data Set. Since access to real-life failure data is typically very limited, we evaluate each algorithm step on synthetic failure data sets generated from realistic reliability models, see Table 2. The already generated failure data sets from the FT-MOEA-CM repository were used for this. In the original research, these data sets were generated using the Monte Carlo method by evaluating all the unique combinations of BEs. This ensures the completeness of the failure data set [11].

The experiment will commence by testing the algorithm by running it with different operators each time and then comparing the input and output for each generation of the FT-MOEA. Table 3 depicts how the data collected from each generation will be stored and presented systematically.

In the implementation of the algorithm, the data will be stored in the form of a directed graph. Fig. 3 presents an example of how the data about each generation will be modeled as a data structure, where the node stores the FT, its metrics, and the operators used to obtain the FT as attributes.

4.3 Genetic operators Evaluation Metrics

Four metrics are proposed to test the effectiveness of the genetic operators and compare them with each other:

- The number of successful FTs obtained each generation after applying a genetic operator
- The mean improvement of metrics for successful FTs after applying a genetic operator
- Mann-Whitney U Test
- Pearson Correlation Coefficient

[1] https://github.com/bogdanColta/ft_moea_cm_analysis

A successful FT means an FT that has passed to the next generation. Also, the improvement of metrics refers to the difference in metrics between the parent FT and the child FT.

Normalization of Successful FTs and Definition Adjustments. The number of successful FTs will be normalized based on the size of the population. This is done since two genetic operators can result in the same FT, and in the context of the experiment, these two FTs are treated as two different entities. To avoid having different population sizes between generations when comparing the genetic operators, the ratio will be calculated using the population size of the algorithm. Also, an FT that is the same as the parent FT after applying a genetic operator and passes to the next generation is not considered a successful FT.

Fig. 4 depicts this research's methodology.

5 RESULTS

During the experiment, the data about the case studies mentioned in Table 2 has been collected five times per case study since the algorithm applies genetic operators in a stochastic way.

The implemented extension of FT-MOEA-CM collects data about each algorithm step by storing it inside a graph structure. This is done whenever a new FT is generated using a genetic operator. A new node containing the FT is added to the graph structure in that situation. The metrics and operators used to obtain the FT are added attributes of the respective node.

Also, when a node is added, an edge is drawn from the node representing the parent FT to the newly created node with the child FT. Applying this structure to the graph makes it possible to register all the evolutionary steps that every FT has followed until the algorithm has reached convergence.

When the algorithm converges, the extension generates data sets that describe the evolutionary steps and metrics of all the FTs. The data sets are created to provide a more human-readable and systematic way to present the data that is stored in the graph data structure. By creating these data sets, the data can be further used to statistically analyze the relationship between the application of genetic operators and the algorithm's convergence.

Gen	Tree	Parent Tree	Metrics	Parent Metrics	Operators	Pass
0	2	0	[0.8958841587409293, 0.216796875, ...]	[0.6355945555932525, 0.18359375, ...]	Delete BE	1
0	3	0	[0.8958841587409293, 0.185546875, ...]	[0.7966048033906593, 0.185546875, ...]	Change Gate	0
0	4	0	[0.9062957428668363, 0.169921875, ...]	[0.6355945555932525, 0.169921875, ...]	Cross Over	1

Table 3. Example table of how the data about generations will be collected

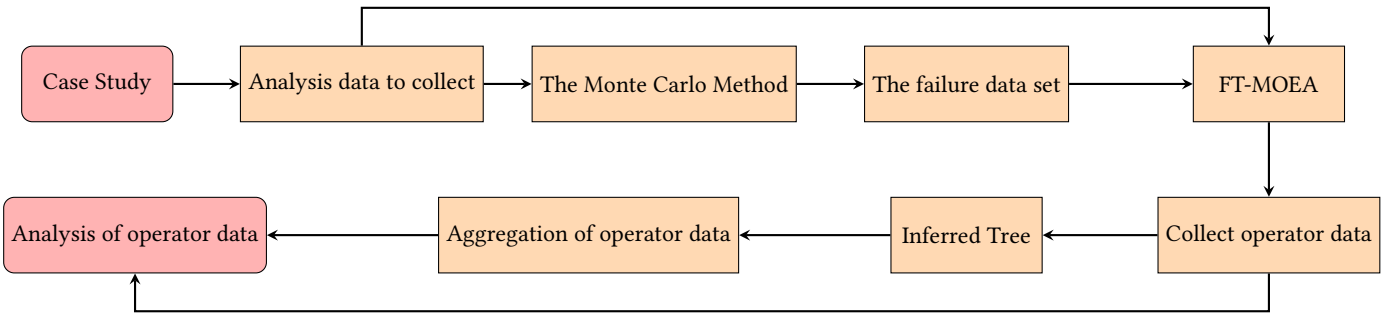

Fig. 4. General Methodology

5.1 P-dataset

Description. To store the metrics of the FTs that are part of the final generation, the extension of the algorithm will generate a performance dataset (P-dataset). This dataset contains two columns: the structure of the FT and a list containing the seven metrics. The number of rows in the data set is the same as the population size, and the FTs are ordered according to the Pareto sorting. Table 4 shows an example of 3 rows from the dataset. The generated datasets can be accessed online.

Implementation and algorithm. This dataset is created by going through the graph and reading the data stored on the last level, representing the last generation. The function's implementation uses a Breadth-First search algorithm that visits every node on a level-by-level basis.

Complexity. The time and space complexity of the algorithm is: $O(g * s)$ and $O(s)$ where g is the number of generations, and s is the population size.

5.2 D-dataset

Description. The decision dataset (D-dataset) is a generated dataset that stores all the evolutionary steps that an FT went through at each generation. This dataset contains two columns: the structure of the FT and a list that contains multiple lists for each generation. The list that describes the FT in that generation comprises three elements: the structure of the FT in that generation, the index of the generation, and a list of genetic operators that resulted in the creation of the FT. The list of genetic operators can also contain a descriptor that tells that the FT passed from the previous generation without being mutated after the Pareto sorting. The number of rows in the dataset is the same as the population size, and the FTs are ordered according to Pareto sorting. Table 5 shows an example of 3 rows from the dataset. The generated datasets can be accessed online.

Implementation and Algorithm. The dataset is created by going through the graph starting from the last level or generation. The implementation of the function employs a Bottom-Up Breadth-First Search algorithm. Such an approach is used since the graph already has a structure similar to a Tree. Besides this, since a node's children are all on the same level in our structure, and each node also stores its parent, a Bottom-Up approach is suitable for this case. The implementation works by starting with one node from the last generation and then visiting the current node's parent until the graph's root is reached. This is done for every node representing an FT from the last generation.

Complexity. The time and space complexity of the algorithm is $O(g * s)$ and $O(s)$, where g is the number of generations and s is the population size.

5.3 Tensor

Description. The extension of the algorithm also generates a tensor to represent the complex data structures derived from the graph analysis. The tensor used in this research is a multi-dimensional array designed to store the metrics of the FTs across different generations. More specifically, it captures the following dimensions:

- Position within generation: This dimension captures the position or index of a particular FT within a given generation. It ranges from 1 to the population size.
- Metrics: This dimension encompasses the various performance metrics associated with each FT. Each metric provides a different aspect of the FT's performance.
- Generation: This dimension represents the sequential generations in the evolutionary process. It ranges from the initial to the final generation, where convergence occurs.

Mathematically, the tensor can be represented as:

$$T[P][M][G]$$

where:

Table 4. Example of the data stored in the P-dataset

FT	Metrics: spec npv prec mcc acc s dor
AND(OR(BE6, BE5), OR(BE2, BE3, BE4))	[0.0, 0.0, 0.0, 0.0, 0.0, 8, 1.0]
AND(OR(BE6, BE5), OR(BE4, AND(BE5, BE6, BE1), BE2, BE3))	[0.0, 0.0, 0.0, 0.0, 0.0, 12, 1.0]
OR(AND(BE6, AND(BE5)), BE3)	[0.0, 0.4, 0.0, 0.4522774424948339, 0.2857142857142857, 6, 0.0]

Table 5. Example of the data stored in the D-dataset

FT	Evolution
AND(OR(BE3, BE4), BE5, BE6)	[[AND(OR(BE3, BE4), BE5, BE6), 30, [previous_generation, delete_gate]], ..., [AND(BE6, OR(BE5, BE1, BE3)), 0, [change_gate_type]]]
AND(OR(BE1, BE2, BE3, AND(BE5, BE6, BE4)), OR(BE5, BE6, BE3))	[[AND(OR(BE1, BE2, BE3, AND(BE5, BE6, BE4)), OR(BE5, BE6, BE3)), 30, [previous_generation, create_be, move_be, connect_be], ..., [OR(BE2, AND(BE5, BE1, BE6)), 0, [move_be]]]]
OR(AND(OR(BE4, BE6), BE5), BE3)	[[OR(AND(OR(BE4, BE6), BE5), BE3), 30, [previous_generation]], ..., [OR(BE3, AND(BE5, AND(BE2, BE4, BE6))), 0, [create_gate]]]

- P is the index of the FT within the generation.
- M is the name of the metric
- G is the index of the generation

Table 6 presents an example of some of the FTs in the tensor's space.

Implementation and algorithm. The tensor is created by going through the graph, starting with the first two nodes representing the initial population of two FTs. The function's implementation also uses the Breadth-First Search algorithm by visiting all the nodes in the same generation level-by-level. For each metric stored in a node, a new entry is created for the tensor along with the value of that specific metric.

Complexity. The algorithm's space and time complexity of generating the tensor is $O(g * m * s)$ where g is the number of generations, m is the number of metrics used, and s is the population size.

Usage. The tensor allows for various analyses. Examining how specific metrics evolve across generations can provide insights into the algorithm's convergence behavior. Moreover, individual FTs' performance can be tracked across different generations. By correlating metric changes with the results from the P- and D-datasets, the effectiveness of different genetic operators can be assessed.

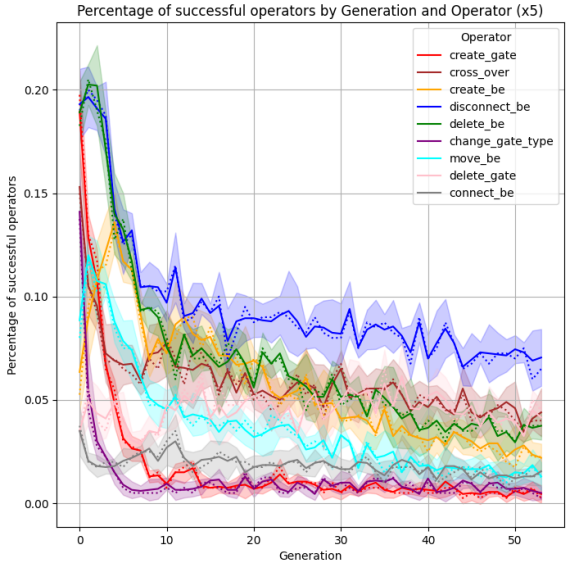

Fig. 5. Number of successful FTs per operator for the COVID-19 case study

5.4 Number of successful FTs per genetic operator

This metric has been calculated to see the difference between the genetic operators regarding how many FTs they created have passed to the next generation. Fig. 5 represents the plots for the mean, median, and 95% uncertainty bounds of the data collected across the five times run of the algorithm for the COVID-19 case study.

Conclusion. As can be seen from the plot, there are differences between the genetic operators. Still, it does not provide enough evidence to conclude which operators are more useful and which are less. Despite this, this metric still offers some valuable insights. From the plots, it is revealed the following phenomena:

- From the 10 and onward generation, the Change Gate Type operator becomes less useful than in the first generations.
- From the 20 and onward generation, the Create Gate operator produces noticeably less successful FTs.
- From the 30 and onward generation, the Connect BE operator also produces less successful FTs.
- The Disconnect BE and Delete BE produce more successful FTs than the other operators, but not by a lot. Also, both

Table 6. Example of the tensor data

FT	Position in the generation	Metric Name	Generation	Value
OR(BE5, BE1, AND(BE3, BE2))	0	npv	0	0.400000
OR(BE6, BE5, BE2, AND(BE3))	3	mcc	0	0.741801
AND(BE4, BE1, BE3)	4	acc	1	0.285714
OR(BE6, BE5, BE4, BE2, BE1)	7	prec	1	0.000000

operators are responsible for removing BEs from the FTs, meaning they are expected to perform similarly.

As explained above, a similar trend can be noticed in the other case studies, see Appendix A.

5.5 The mean difference in metrics for successful FTs per genetic operator

The metric is calculated to observe the difference between the genetic operators in the change of metrics of an FT after mutation. After analyzing the given metric, the data did not exhibit any pattern between multiple runs of the same case study. Even for a single run, the data was too inconsistent across multiple generations as the uncertainty bounds for the mean were very large, reflecting significant variability in the data, see Appendix B. This variability may be the result of the algorithm's stochastic nature.

5.6 Survival rate after applying a genetic operator per generation

This metric is similar to the one described in Section 5.4 with the difference that the number of successful FTs is not normalized based on the population size. The metric for each genetic operator is calculated as the ratio of the number of successful FTs created by the genetic operator to the total number of fault trees created by that operator. This difference is important as we want to compare genetic operators between each other in the context of each generation and not within the context of the overall evolutionary process.

Mann-Whitney U Test. The data for each genetic operator were collected as data sets containing the survival rate per generation. These data sets were then used to perform a pairwise comparison between each operator using the Mann-Whitney U Test. The following setup was used to perform the U Test:

- Null Hypothesis (H0): The two operators have no significant difference in the survival rates.
- Alternative Hypothesis (H1): There is a significant difference in the survival rates between the two operators.
- The significance level used is $\alpha = 0.05$. If the obtained p-value from the U Test is $\geq \alpha$, H0 is accepted. Else, H1 is accepted, and H0 is rejected.

Score system. After performing the pairwise U Test between each operator, if the Null Hypothesis is rejected, there is a significant difference between the two genetic operators. In this case, the mean value of the data sets will contribute to the score of both operators. The difference between the means will be added to the operator's score. Thus, the operator with the higher mean will increase its score, and the operator with a lower mean will decrease its score. Even though the U Tests are performed in the context of each generation, the calculated score is the mean of all generations, which means that it describes the genetic operator in the context of the overall evolutionary process.

Results. Table 7 represents the mean scores and 95% uncertainty bounds obtained for the data collected across the five times run of the algorithm for the COVID-19 case study. The table shows the genetic operators sorted according to the mean score.

Conclusion. As can be seen in the table, there are differences between the genetic operators. This analysis reveals that Disconnect BE, Delete BE, Cross-Over, Create BE, and Delete Gate are the most successful operators. This score only describes the efficiency of a genetic operator for its overall use across all generations of evolution. So, it cannot be fully concluded for which generation a genetic operator has shown a good score. Even though there are four genetic operators with a low overall score, it can be seen in Section 5.4 that these genetic operators still have an impact in the first generations and then become less useful very early in the evolution, leading to a low overall score. A similar pattern can be noticed for the other case studies, see Appendix C.

Operator	Mean Score	Lower B.	Upper B.
Disconnect BE	0.40942	0.38603	0.43280
Delete BE	0.19492	0.14905	0.24080
Cross-Over	0.08972	0.07155	0.10789
Create BE	0.07299	0.03218	0.11381
Delete Gate	0.02914	-0.00841	0.06670
Move BE	-0.07621	-0.10747	-0.04495
Connect BE	-0.21484	-0.23395	-0.19572
Create Gate	-0.2374	-0.24544	-0.22936
Change Gate Type	-0.26775	-0.28913	-0.24637

Table 7. The scores obtained for each genetic operator for the COVID-19 case study

5.7 Pearson Correlation Coefficient

This metric is calculated to find the correlation between the genetic operators used in each generation and the metrics of the final FTs obtained. P- and D- datasets were used to calculate the coefficients. Table 8 shows the mean and 95% uncertainty bounds of the data collected across five times run of the algorithm for the COVID-19 case study.

Interpretation of results. The Pearson correlation coefficient can have a value between the $[-1, 1]$ interval. A value closer to -1 shows a negative correlation between the variables, meaning that by decreasing one variable, the other one increases. A value closer to 1

Table 8. The mean values and 95% uncertainty bounds for the Pearson correlation coefficient for three genetic operators (COVID-19 case study)

Operator	**spec**	**npv**	**prec**	**mcc**	**acc**	**s**	**dor**
Change Gate	0.015 [−0.078, 0.109]	-0.015, [−0.111, 0.08]	-0.001, [−0.094, 0.093]	-0.001, [−0.086, 0.084]	-0.014, [−0.104, 0.075]	0.056, [−0.008, 0.121]	-0.026, [−0.172, 0.12]
Create BE	-0.276 [−0.344, −0.208]	-0.166 [−0.222, −0.111]	-0.244 [−0.31, −0.178]	-0.302 [−0.343, −0.262]	-0.259 [−0.298, −0.22]	0.467 [0.394, 0.541]	-0.126 [−0.184, −0.067]
Delete Gate	0.183 [0.117, 0.249]	-0.016 [−0.085, 0.051]	0.187 [0.13, 0.244]	0.141 [0.035, 0.247]	0.044 [−0.033, 0.122]	-0.174 [−0.216, −0.131]	0.061 [0.039, 0.083]

indicates a positive correlation between the variables, and by increasing one variable, the other one also tends to increase. A value closer to 0 shows no correlation between the variables. By applying this test, a negative correlation is better as it shows that by applying a genetic operator more, a specific metric can be decreased. The same logic applies to a null or positive value, as it indicates that there is no correlation or that the operator is increasing the value of the metrics.

Conclusions. The following conclusions were drawn from the calculated coefficients for the COVID-19 case study:

- There is a small negative correlation (≥ -0.2) between the Create BE operator and the spec, npv, prec, mcc, acc metrics.
- The Delete Gate shows a small positive correlation (≥ 0.1) for the spec and prec metrics. For the acc metric, it also shows some positive correlation, but the uncertainty bounds are too large to draw a certain conclusion.

The correlation coefficients do not draw other conclusions because either their values are too close to 0 or the uncertainty bounds are too large due to the algorithm's stochastic nature. All the test results can be found inside the Jupyter notebooks provided.

Moreover, a pattern across different case studies cannot be noticed since the conclusions are different for each. Appendix D presents some coefficients from which conclusions can be drawn for other case studies.

6 FUTURE WORK

Although this research has contributed to the further development of the FT-MOEA-CM, multiple aspects of the algorithm still need to be studied to find a correlation between the genetic operators and the evolution of an FT.

Analyze new metrics for genetic operators comparison. This paper has discussed the structure of the P-dataset, the D-dataset, and a Tensor, which stores information about the metrics of the FTs in each generation. These datasets are introduced with the scope of later being used to study the correlation between the final metrics of the FT and the genetic operators that were applied in each generation.

Guided genetic operators using Bayesian Optimization. Suppose ongoing research in this direction reveals a stronger correlation between specific genetic operators and the metrics of an FT. In that case, further measures should be implemented to guide these genetic operators accordingly. For this, Bayesian Optimizations could be implemented to apply operators based on the current structure or metrics of the FT.

7 CONCLUSION

This paper introduced an extension of the FT-MOEA-CM's implementation, specifically designed to analyze the generations and the genetic operators applied inside a multi-objective evolutionary algorithm. The research focused on understanding the genetic operators' influence on the algorithm's convergence and which genetic operators are more useful for finding the global optima and which are less.

Multiple datasets that describe the algorithm's evolution have been proposed to address this issue. The D-dataset and the P-dataset have been proposed to find a correlation between the genetic operators and each generation's FT by applying a correlation coefficient. Also, a tensor that describes a multi-linear relationship between the FTs, generation, and each metric has been proposed. The tensor provides a much clearer way to access and visualize how each FT was mutated during each generation and which specific metrics were affected.

Moreover, this research also mentions the introduction of four analysis metrics: the number of successful FTs and the mean improvement of metrics for successful FTs after applying a genetic operator, the Mann-Whitney U Test between the survival rates of each operator, and the Pearson Correlation.

The first metric shows us that a pattern might exist across different case studies but does not provide enough insight to conclude which genetic operators are better. The second metric did not show any noticeable insight, as the data was inconsistent across multiple runs of the same case study. In most instances, the uncertainty bounds of the calculated values were too large to reveal the presence of a pattern. The Mann-Whitney U Test shows some insight into the genetic operators' differences. Still, the differences described only relate to the overall genetic evolution, meaning that it cannot be used to conclude for which generation a genetic operator shows a more significant improvement. The last analysis metric proposed is the Pearson correlation coefficient. This metric only shows results for specific cases and is unreliable due to the large uncertainty bounds of the obtained values.

REFERENCES

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

[2] J.-P. Signoret and A. Leroy, *Automated fault tree building, Rel. Assessment Safety Prod. Syst.*, pp. 423–426, 2021.

[3] S. L. Salem, G. Apostolakis, and D. Okrent, *Computer-oriented approach to fault-tree construction*, California Univ., Tech. Rep. EPRI-NP-288, 1976.

[4] Charles Dickerson, Rosmira Roslan, Siyuan Ji, *A Formal Transformation Method for Automated Fault Tree Generation from a UML Activity Model*, *IEEE Transactions on Reliability 67, 3 (2018), 1219–1236*

[5] Linard A., Bucur D., Stoelinga M., *Fault Trees from Data: Efficient Learning with an Evolutionary Algorithm*, *Guan, N., Katoen, JP., Sun, J. (eds) Dependable Software Engineering. Theories, Tools, and Applications. SETTA 2019. Lecture Notes in Computer Science(), vol 11951. Springer, Cham.*

[6] L. A. Jimenez-Roa, T. Heskes, T. Tinga and M. Stoelinga, *Automatic Inference of Fault Tree Models Via Multi-Objective Evolutionary Algorithms*, *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 3317–3327, 1 July-Aug. 2023.

[7] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002*

[8] Mart´ı, L., Segredo, E., Pi, N.S., Hart, E., *Impact of selection methods on the diversity of many-objective Pareto set approximations, In: KES. Procedia Computer Science, vol. 112, pp. 844–853. Elsevier (2017)*

[9] Dorfhuber, F., Eisentraut, J., Kret´ınsk´y, J., *Learning attack trees by genetic algorithms, In: ICTAC. Lecture Notes in Computer Science, vol. 14446, pp. 55–73*

[10] Nicu Rusnac, Matthias Volk, L. A. Jimenez-Roa, *Improving the Performance of Multi-Objective Evolutionary Algorithms for Fault Tree Inference*, 7 July-Aug. 2023.

[11] Lisandro A. Jimenez-Roa, Nicolae Rusnac, Matthias Volk, and Marielle Stoelinga, *Fault Tree Inference using Multi-Objective Evolutionary Algorithms and Confusion Matrix-based Metrics*

[12] Lisandro A. Jimenez-Roa, Matthias Volk, and Marielle Stoelinga, *Data-Driven Inference of Fault Tree Models Exploiting Symmetry and Modularization Trapp, M., Saglietti, F., Spisländer, M., Bitsch, F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2022.*

[13] A.E. Eiben and J.E. Smith, *Introduction to evolutionary computing. Springer. 25–48 pages.*

[14] M. Stamatelatos, W. Vesely, J. B. Dugan, J. Fragola, J. Minarick, and J. Railsback, *Fault Tree Handbook with Aerospace Applications Office of safety and mission assurance NASA headquarters, 2002.*

[15] S. Lazarova-Molnar, P. Niloofar, and G. K. Barta, *Data-driven fault tree modeling for reliability assessment of cyber-physical systems in Proc. Winter Simul. Conf., 2020, pp. 2719–2730.*

[16] T. Bakeli et al., *Covid-19 infection risk management during construction activities: An approach based on fault tree analysis (FTA), J. Emerg. Manage., vol. 18, no. 7, pp. 161–176, 2020.*

APPENDICES

A APPENDIX A: PLOTS FOR THE NUMBER OF SUCCESSFUL FTS PER GENETIC OPERATORS

In this Appendix, the analysis metric is plotted for each genetic operator across multiple generations in Fig. 6 and 7, for the other case studies mentioned in the paper. The data was collected five times for the same case study, and the mean was plotted with 95% uncertainty bounds along with the median.

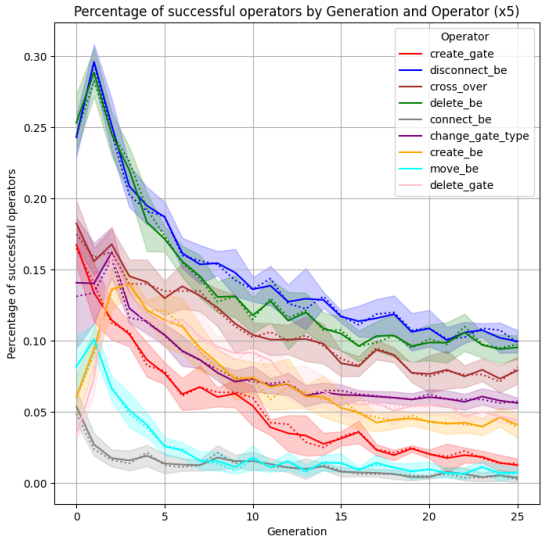

Fig. 6. The number of successful FTs per genetic operators for the CSD case study

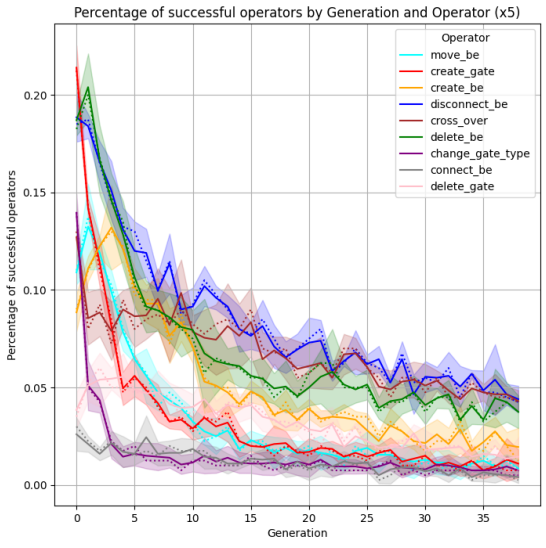

Fig. 7. The number of successful FTs per genetic operators for the DDFT case study

B APPENDIX B: PLOTS FOR THE MEAN DIFFERENCE IN METRICS FOR SUCCESSFUL FTS PER GENETIC OPERATOR

In this Appendix, the analysis metric is plotted for the accuracy of the FT for each genetic operator across multiple generations for the COVID-19 case study in Fig. 8. In Fig. 9, the 95% uncertainty bounds and median are plotted for the Cross Over operator. It can be observed that the uncertainty bounds are very large, and there is a high variability between the mean difference even in the same generation, which can be attributed to the algorithm's stochastic nature.

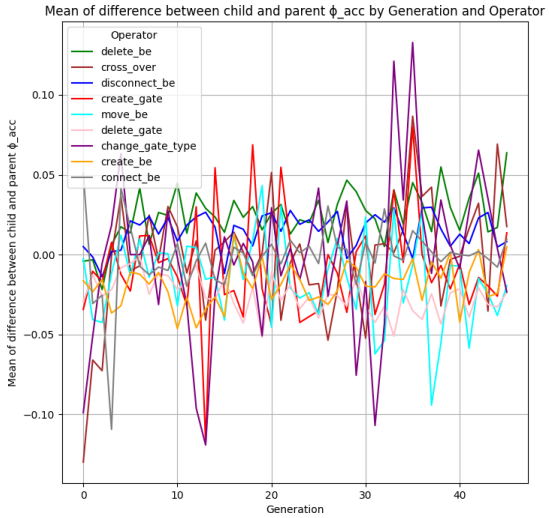

Fig. 8. Mean difference between the accuracy of the child and parent FT after applying a genetic operator for the COVID-19 case study

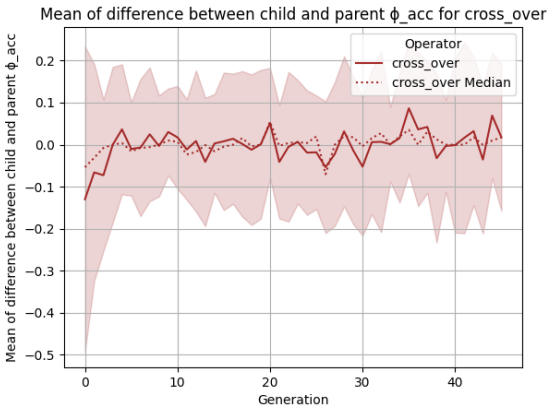

Fig. 9. Mean difference between the accuracy of the child and parent FT after applying the cross-over operator for the COVID-19 case study along with the 95% uncertainty bounds and median

C APPENDIX C: SCORES OF GENETIC OPERATORS ACCORDING TO MANN-WHITNEY U COMPARISON

In this Appendix, the analysis metric is presented in Table 9 and 10 for the other case studies mentioned in the paper. The data was collected five times per case study, and the mean was calculated along with 95% uncertainty bounds.

Operator	Mean Score	Lower B.	Upper B.
Disconnect BE	0.53216	0.50630	0.55802
Delete BE	0.47412	0.44462	0.50362
Cross Over	0.28496	0.24905	0.32087
Change Gate Type	0.02272	0.00686	0.03858
Delete Gate	-0.01968	-0.03714	-0.00222
Create BE	-0.04364	-0.06081	-0.02647
Create Gate	-0.24022	-0.26133	-0.21911
Move BE	-0.46846	-0.49723	-0.43968
Connect BE	-0.54195	-0.56762	-0.51629

Table 9. The scores obtained for each genetic operator for the CSD case study

Operator	Mean Score	Lower B.	Upper B.
Disconnect BE	0.32626	0.31218	0.34035
Cross Over	0.21060	0.19396	0.22724
Delete BE	0.20907	0.18668	0.23145
Create BE	0.04105	0.02611	0.05599
Delete Gate	-0.08477	-0.10688	-0.06267
Create Gate	-0.11638	-0.13454	-0.09823
Move BE	-0.11847	-0.13000	-0.10694
Change Gate Type	-0.22512	-0.24468	-0.20556
Connect BE	-0.24223	-0.25593	-0.22853

Table 10. The scores obtained for each genetic operator for the DDFT case study

D APPENDIX D: PEARSON CORRELATION COEFFICIENT

In this Appendix, the Pearson correlation coefficient is presented in Table 11 and 12 for the other case studies mentioned in the paper. The data was collected five times per case study, and the mean was calculated along with 95% uncertainty bounds.

Table 11. The mean values and 95% uncertainty bounds for the Pearson correlation coefficient for three genetic operators (CSD case study)

Operator	spec	npv	prec	mcc	acc	s	dor
Delete BE	0.151 [0.132, 0.17]	0.099 [0.072, 0.126]	0.114 [0.093, 0.135]	0.207 [0.182, 0.233]	0.167 [0.139, 0.195]	-0.466 [−0.484, −0.447]	0.046 [0.022, 0.069]
Create BE	-0.075 [−0.09, −0.06]	-0.075 [−0.113, −0.037]	-0.071 [−0.088, −0.055]	-0.176 [−0.2, −0.153]	-0.119 [−0.16, −0.078]	0.552 [0.51, 0.594]	-0.025 [−0.061, 0.011]
Disconnect BE	0.128 [0.085, 0.17]	0.105 [0.09, 0.12]	0.087 [0.045, 0.13]	0.184 [0.15, 0.219]	0.161 [0.132, 0.189]	-0.414 [−0.442, −0.385]	0.024 [0.007, 0.042]

Table 12. The mean values and 95% uncertainty bounds for the Pearson correlation coefficient for three genetic operators (DDFT case study)

Operator	spec	npv	prec	mcc	acc	s	dor
Delete Gate	0.258 [0.202, 0.314]	0.073 [0.03, 0.115]	0.259 [0.203, 0.315]	0.32 [0.287, 0.353]	0.207 [0.185, 0.229]	-0.33 [−0.438, −0.222]	0.191 [0.152, 0.231]
Create BE	-0.037 [−0.068, −0.006]	-0.307 [−0.344, −0.271]	-0.01 [−0.04, 0.018]	-0.327 [−0.366, −0.289]	-0.348 [−0.395, −0.301]	0.512 [0.469, 0.555]	0.03 [−0.011, 0.072]
Delete BE	-0.079 [−0.103, −0.056]	0.202 [0.157, 0.246]	-0.114 [−0.139, −0.089]	0.098 [0.042, 0.154]	0.17 [0.114, 0.225]	-0.24 [−0.33, −0.151]	-0.066 [−0.109, −0.023]