Improving the Performance of Multi-Objective Evolutionary Algorithms for Fault Tree Inference

NICOLAE RUSNAC SUPERVISORS: MATTHIAS VOLK, LISANDRO A. JIMENEZ-ROA, University of Twente, The Netherlands

ABSTRACT

Fault trees (FTs) are models frequently used in risk management to identify potential failures and improve a system's reliability. They are typically created manually in consultation with domain experts, which is time-consuming and prone to human error. Existing multi-objective evolutionary algorithms optimise this process by using failure data sets generated by a system for inferring a FT. These algorithms are known to be inefficient and, in some cases unable to find an optimal solution. In this research, the focus has been placed on identifying better metrics for the existing genetic algorithm. Multiple metrics were identified and analyzed such as *random segmentation accuracy*, *basic event impact vector distance* and *confusion matrix metrics*. The results show that by adding metrics from the confusion matrix, significant improvements can be achieved in both the convergence time of the algorithm and the number of generations it takes to converge.

Keywords: Fault tree inference, data-driven,multi-objective evolutionary algorithms, optimization, risk management.

1 INTRODUCTION

Fault tree analysis [17] (FTA) is a systematic method for acquiring information about a system, which can lead to improved decision-making [17]. FTA provides several advantages, including a holistic overview of the system and the relationships that emerge within itself, enabling quantitative analysis by calculating the probabilities of the events and a more intuitive visual representation of the system. However, one of the drawbacks of FTs is related to their manual construction which makes the process slow and prone to bias. Additionally, FTA requires large data sets to create an accurate overview of the system. This makes them difficult to be up to date with a dynamic system. This paper will try to improve on the creation aspect of FTs.

This problem has been the subject of research in the past [10, 18]. Historically, three ways of automatic FT inference have been described: knowledge-based, model-based and data-driven [7]. *Knowledge-based* systems rely on knowledge about the components and the relationships between them. *Model-based* approaches focus on translating existing models, such as graphs, into FTs. *Data-driven* approaches use datasets for the inference of FTs.

In this paper, a data-driven approach will be the focus of the study. Precisely, it will focus on improving the performance of the FT-MOEA algorithm [7]. FT-MOEA is a multi-objective evolutionary algorithm (MOEA), inspired by biological evolution. These algorithms operate on principles such as survival of the fittest, mutation and crossover while trying to optimize multiple objectives (typically conflicting) in parallel [16]. In the context of FT-MOEA, it will be used to infer FTs from a failure data set.

The original FT-MOEA algorithm is able to infer FTs successfully with a low error rate while also prioritizing a small size for the FTs. The algorithm uses three objectives: *Error based on the failure data set* (ϕ_d), *Error based on the Minimal Cut Sets (MCS)* (ϕ_c), and *FT Size* (ϕ_s) [7]. The ϕ_c metric depends on the computation of MCSs, which is exponential in complexity [15] and restricts the algorithm from using noisy data [7].

Consequently, this research aims to remove this metric due to its deficits in usability and scalability and introduce others that could potentially improve convergence time and compensate for the quality improvements introduced by ϕ_c.

Typically in MOEAs problem specific knowledge is used to a large extent [13, 14]. In this paper, metrics related to the structure of the FT are considered, but also ones computed out of the *confusion matrix*.

The confusion matrix metrics are considered in spite of the fact that they are unrelated to FTs. A FT can be directly converted to a failure data set, making such metrics convenient. Using these metrics is useful in the context of comparing the similarity of datasets. Additionally, they have a low computational cost. They will be used in this research to design a new algorithm called FT-MOEA-CM [1]. Since designing an evolutionary algorithm is highly specific in nature, this research is exploratory in its nature.

The main contributions of this paper are:

1. Identification of metrics to be used in the algorithm
2. An analysis of the impact of the metrics on the performance of the algorithm
3. An improvement in both convergence time and number of generations till the convergence of the algorithm
4. Other insights about the nature of the algorithm and what factors affect its performance

In the following sections of this paper, the research questions are presented in Section 2 and then follows a background section (Section 3) where the relevant information for the understanding of the paper is explained. Afterwards, the method section (Section 4) explains the methods used during the research, followed by an experiments and results (Section 5) section where the findings are presented and interpreted. The related work (Section 6) will contain the main sources used in this paper, and the future work (Section 7) section explains additional future improvements to the performance of the algorithm. Finally, the conclusion section (Section 8) will wrap up the paper.

TScIT 39, July 7, 2023, Enschede, The Netherlands

[1]Repository with the implementation of the algorithm

2 RESEARCH QUESTIONS

The research questions of this study are:

- What new metrics can be identified for improving the FT-MOEA algorithm?
- How do the metrics affect the algorithm's performance in terms of convergence time, number of generations until convergence, size and accuracy w.r.t the dataset?
- Considering the newly identified metrics, what is the most optimal configuration?
- How does the newly found optimal configuration compare in terms of performance with the original algorithm?

3 BACKGROUND

3.1 Fault trees

Fault trees [17] are a graphical tool used in reliability and safety analysis, which provide a compact representation of the ways in which a system can fail. They consist of a few building blocks, which will be referenced throughout this paper:

(1) *Basic events* (BE). They are the leaf nodes of the FT and have a probability value representing the chance of them happening.
(2) *Gates.* These are operations performed on Basic Events or Intermediate events. For the scope of this research, only OR and AND gates will be used.
(3) *Intermediate events* (IE). These are vents that result from a gate operation.
(4) *Top event* (TE). This is the root of the FT. It represents the event towards which all the other events lead to.
(5) (FT) *Element.* This refers to either a Gate or a Basic Event.

In Fig. 1, an example of a FT is displayed. To offer a better explanation of what a FT is, its elements will be explained. For example, BE2, BE3 and BE4 are basic events. They are connected together to an OR gate (IE3), which is an intermediate event and BE1 is connected to an AND gate (IE1). At the top of the tree the top event TE can be seen.

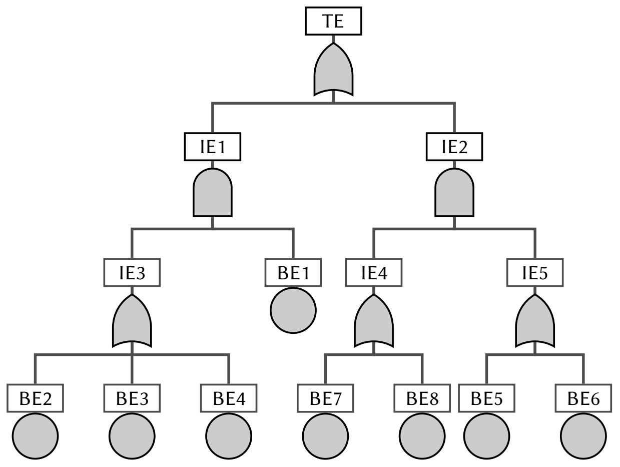


Fig. 1. Inferred FTs from the MPPS dataset using the improved version of the FT-MOEA algorithm.

Now that the concept of FTs has been explained, it is useful to explain an integral concept related to FTA: Minimal Cut Sets (MCS). MCS refer to the sets with a minimal number of basic events that lead to a system failure. In the example, BE1 and BE2 form an MCS, because their occurrence leads to the occurrence of the top event. IE3 is an OR gate, so it will have a true value since BE1 is true. IE1 will also be true because IE3 is true, and BE1 is true. TE is an OR gate, so it will be true since IE1 is true. The cut set is irreducible, meaning there is no BE we can remove from the set and still have the TE occur. The concept of MCS is important as it will be used multiple times in the metric analysis.

3.2 Failure data sets

The failure data sets used throughout this research paper consist of columns represented by basic events (BE), the value of the top event for that configuration (TE) and the number of times the event occurred (Count). Each basic event also has a probability value. However, it is not used within the scope of this research. The datasets are complete, meaning they have a row corresponding to any basic event configuration and are noise-free, meaning there is no corrupted data in the dataset. Table 1 contains an example of a randomly generated dataset for seven basic events, each with a probability of 0.5.

Table 1. Dataset example

Ob.	BE1	BE2	BE3	BE4	BE5	BE6	BE7	TE	Count
1	0	0	0	0	0	0	0	0	1,968
2	0	0	0	0	0	0	1	0	2,039
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
24	0	0	1	0	1	1	1	1	1,976
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
128	1	1	1	1	1	1	1	1	1,947
$p \approx$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	$N = 250,000$	

3.3 FT-MOEA

To explain the FT-MOEA algorithm, evolutionary algorithms should be described first. An evolutionary algorithm [6] mimics biological evolution. It has an initial population, mutation operations, a selection procedure, and a fitness function. The initial population represents the seed population used initially when the algorithm starts. Mutation operations are applied to the existing population to create a new, potentially improved population. To evaluate whether the offspring population is better, a fitness function considers different characteristics of individuals within the population. Finally, a selection procedure is a way in determining which members of the current generation move on to the next one.

The FT-MOEA algorithm [7] starts with an initial population of two FTs: an AND and an OR gate linked to all the basic events. During an evolutionary step, the FTs can be subject to seven genetic mutation operations:

(1) *Disconnect.* Disconnect a basic event from the tree.
(2) *Connect.* Connect a disconnected basic event and connect it to an existing gate.
(3) *Swap.* Move an event under a different gate.

(4) *Create.* A random gate is created under a random gate in the tree.
(5) *Delete.* A random gate is deleted, including its children.
(6) *Cross-over.* Randomly select two trees and exchange an element between them.
(7) *Change gate.* A random gate is changed to the opposite type.

After mutations occur, the metric computation takes place. The used metrics are: Error based on the failure data set (ϕ_d), Error based on the MCS (ϕ_c), and FT Size (ϕ_s). After the computation is done, NGSA-II[4], a sorting algorithm used in multi-objective evolutionary algorithms, is applied to select the population for the next generation.

There are also three important parameters used in the algorithm [7]:

- *population size (ps)*, which represents the population that will be selected to move on to the next generation
- *maximum generations (mg)*, which is used as a convergence criterion to prevent the algorithm from running indefinitely
- *unchanged generations (ug)*, which represent the number of generations during which the best FT has to remain unchanged in order for the algorithm to converge

4 METHOD

4.1 Problem definition

The problem attempted to be solved in this paper is the algorithm's performance. We define an improvement in performance the following way: Let (t_1, a_1) and (t_2, a_2) be the performance measures of convergence time and accuracy for two algorithms for the same FT. In this context, accuracy can be the error with respect to the total sum of the times a TE occurs divided by the total event count (N). We say that an algorithm with measure (t_1, a_1) is better than or equivalent to an algorithm with measure (t_2, a_2), denoted as $(t_1, a_1) \geq (t_2, a_2)$, if and only if:

$$(t_1 \leq t_2 \wedge a_1 \geq a_2) \vee a_1 > a_2$$

Where:
- t_1, t_2 represent the convergence times of the two algorithms.
- a_1, a_2 represent the accuracies of the two algorithms.

For the purposes of this study, this notation also applies for the time average and the accuracy average.

The problem that will be attempted to be solved in this research is to find metrics that lead to better performance in terms of the aforementioned metrics.

4.2 Experiment Design

For determining the performance of the algorithm, multiple tests have been conducted. These tests use the same parameters as in the original FT-MOEA paper [7]: population size - 400, maximum generations - 100, unchanged generations - 20. Additionally, the tests were performed on the HPC cluster of the University of Twente. The tests were run 5 times each. To conduct the analysis, the data results have been converted to charts using MATLAB scripts, from which conclusions were inferred. In total 4 datasets have been used to ensure the validity of the algorithm's results. Details about the failure datasets can be found in Table 2. The **MPPS** and **COVID-19** failure data sets are the same ones used in the original FT-MOEA paper [7]. **TS1** is based on a FT used in another paper [8] that focuses on improving the algorithm's performance. The **GPT dataset** is based on a random FT generated by Chat GPT[12]. For both TS1 and GPT the failure datasets have been randomly generated assuming a probability of 0.5 of occurring for each basic event.

Table 3. Confusion Matrix Properties

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

4.3 Performance evaluation metrics

Multiple performance metrics have been considered for the purpose of this research. In this section, each will be discussed, and the reasoning behind them. Their results will be further discussed in the Results section Section 5.

4.3.1 Confusion matrix. Using metrics based on the confusion matrix (Table 3) has numerous advantages. Compared to MCS, this metric is linear in complexity, so the metrics are much faster to compute. From the dataset of a FT, six different values can be calculated in a single iteration: P - positives, N - negatives, TP - true positives, TN - true negatives, FP - false positives, and FN - false negatives. Using these values, a large number of metrics can be computed. From the perspective of CPU cycles, these metrics provide an advantage, since once the confusion matrix properties are calculated all the other metrics can be computed without any additional overhead.

As a metric combination, errors with respect to the following metrics have been used: error w.r.t specificity (E_{spec}), error w.r.t sensitivity (E_{sens}), error w.r.t precision (E_{prec}), error w.r.t negative predictive value (E_{npv}) and error w.r.t accuracy (E_{acc}). These metrics are calculated according to the following formulas:

- Error w.r.t Sensitivity (True Positive Rate or Recall):

$$E_{\text{sens}} = 1 - \frac{TP}{TP + FN} \tag{1}$$

- Error w.r.t Specificity (True Negative Rate):

$$E_{\text{spec}} = 1 - \frac{TN}{TN + FP} \tag{2}$$

- Error w.r.t Precision (Positive Predictive Value):

$$E_{\text{prec}} = 1 - \frac{TP}{TP + FP} \tag{3}$$

- Error w.r.t Negative Predictive Value:

$$E_{\text{npv}} = 1 - \frac{TN}{TN + FN} \tag{4}$$

- Error w.r.t Accuracy:

$$E_{\text{acc}} = 1 - \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

In contrast to the (ϕ_d) metric, these metrics do not use the count of the occurrence of an event, but the occurrence of the event itself. This makes a rare, but existing event more likely to be a part of the resulting FT.

Table 2. The datasets used during the testing process and their configuration.

Dataset Name	#BE	#Gate	#AND	#OR	Rows in dataset
MPPS	8	6	2	4	256
COVID-19	9	4	2	2	512
TS1	10	8	3	5	1024
GPT	12	10	5	5	4096

These metrics have been chosen because they are the most commonly used that are computed from the confusion matrix. There are other metrics as well, but they use these metrics as their foundation.

These metrics should increase the search space of the algorithm. They each provide a different perspective of the FT since they don't perfectly correlate. The more conflicting objectives there are that don't correlate, the bigger the explored problem space becomes.

4.3.2 Random dataset segmentation. This technique randomly shuffles the rows of the failure dataset and splits it into equally sized segments. Then the accuracy with respect to the filled segments is calculated. Accuracy with respect to filled segments is defined as the number of filled segments divided by the total number of segments. A segment is considered filled if the accuracy within that respective segment is 100%. For the purpose of this study, segments of sizes 2,4,8 were analyzed. The reasoning behind this metric is to first make all data points equally important by removing the times the configuration of that specific data point occurs. Additionally, it makes the FT less conforming to the MCS. Since an MCS already process a true TE, the values of the other events are irrelevant. This means that a smaller MCS will correspond to a higher number of rows. By shuffling, we give the MCS with a lower number of rows a higher importance, making the tree move in their direction as well.

One interesting observation related to accuracy is that an improvement in the tree's structure does not necessarily correlate with increased accuracy. The segmentation prevents small improvements in accuracy from having a big impact.

4.3.3 Event impact vector distance. Suppose we have a dataset with N rows and let T be the top event. We introduce a new metric based on the Birnbaum importance measure [11]. This metric is aimed to calculate the impact of an event E on T using the following steps:

(1) Let $m(E)$ represent the number of occurrences of the top event T when event E occurs. Then,

$$m(E) = \sum_{i:E_i \text{ occurs}} T_i \tag{6}$$

where E_i and T_i are the instances of events E and T in the i-th row of the dataset.

(2) Let $m(\neg E)$ represent the number of occurrences of T when E does not occur. Then,

$$m(\neg E) = \sum_{i:E_i \text{ does not occur}} T_i \tag{7}$$

(3) The impact $I(E)$ of event E on T is computed as:

$$I(E) = \frac{m(E) - m(\neg E)}{N/2} \tag{8}$$

Here we divide by N/2 because a failure data set contains all the possible configurations of basic events. This means that for each configuration of BEs where a BE occurs, there is a similar configuration where the event does not occur.

(4) For every basic event E_j, we calculate $I(E_j)$ and create a vector $\mathbf{v} = [I(E_1), I(E_2), \ldots, I(E_n)]$ where n is the number of basic events.

(5) Compute the same vector for the original dataset to obtain $\mathbf{v}_{\text{dataset}}$.

(6) Normalize both vectors:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{||\mathbf{v}||} \quad \text{and} \quad \hat{\mathbf{v}}_{\text{dataset}} = \frac{\mathbf{v}_{\text{dataset}}}{||\mathbf{v}_{\text{dataset}}||} \tag{9}$$

(7) Calculate the Euclidean distance D between them:

$$D = ||\hat{\mathbf{v}} - \hat{\mathbf{v}}_{\text{dataset}}|| \tag{10}$$

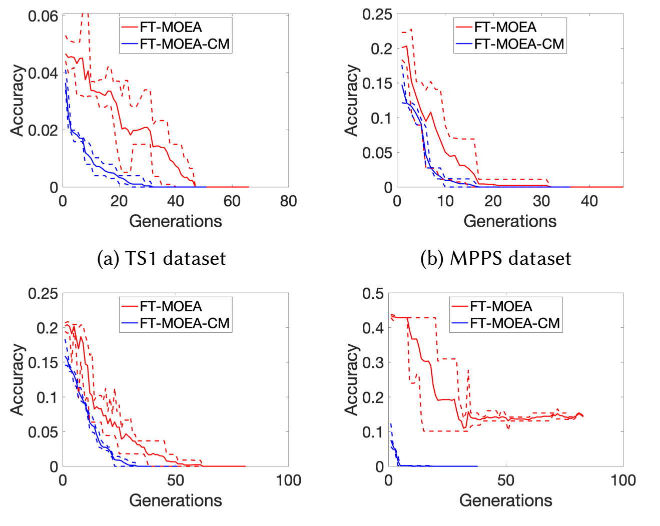
One potential limitation of this metric is that it compresses the impact of all basic events into a single scalar value, D, which may not fully represent the algorithm's progress. A decrease in impact difference for a highly important event could be overshadowed by minor deviations in less important events. A potentially more effective approach might be to consider the difference in impact for each event as a separate objective. This is beyond the scope of the present research as it would imply the use of MOEAs with dynamic objectives, which would lead to refactoring big parts of the existing codebase and significant time investment. Every BE would correspond to an objective, meaning the number of objectives would depend on the failure data set.

4.4 Optimization techniques

The parallelization of the FT-MOEA has been considered for improving its performance as an additional optimization. This is beneficial for computing larger trees and understanding the effect of metrics configuration on them. To determine whether the algorithm can benefit from such an improvement, some analysis has been done on the way in which the computation time scales with the growth in the number of basic events for each portion of the program.

The FT-MOEA consists of three parts which repeat every generation: creating a new population by applying genetic operations, computing the metrics of the generated population, and applying NSGA-II sorting for selecting the population for the next generation. The first two parts are the best candidates for this optimization. Generally speaking, programs that do many computations where their order does not matter are easily parallelizable.

Applying the genetic operations to the population can be done arbitrarily. It is important to remove duplicate trees, which can be done after all the parallel processes finish their work. The same



(a) TS1 dataset (b) MPPS dataset

(c) GPT dataset (d) COVID-19 dataset

Fig. 2. Visual representation of accuracy change across generations for different datasets (the dotted lines represent the minimum and maximum values used in the computation of the average). In this context, Accuracy can be considered equivalent to ϕ_d.

principle applies to the metric calculation section of the algorithm. The computations' order is unimportant, so they can be done in parallel.

5 RESULTS

5.1 Approach to algorithm improvement

Two primary methods were used to enhance the algorithm's performance: Literature research, involving the study of existing Multi-Objective Evolutionary Algorithms (MOEAs) which was unsuccessful due to their unique problem-specific metrics; and Exploratory research, which was centred on analyzing the specific problem to identify potential metrics.

5.2 Confusion Matrix Metrics vs FT-MOEA Metrics

The confusion matrix metrics (CMM) have been used as a candidate metric against the original one used in FT-MOEA. The goal was to find a configuration with an improved complexity compared to MCSs. Because of this, the CMM have not been evaluated in conjunction with ϕ_c. For the sake of simplicity, we will further refer to the algorithm with the original configuration (ϕ_c, ϕ_d, ϕ_s) as FT-MOEA. The algorithm using the configuration resulting from the aggregation of all the metrics described in the previous section Section 4.3.1 (E_{spec}, E_{sens}, E_{prec}, E_{npv}, E_{acc}) is FT-MOEA-CM.

5.2.1 Identifying the optimal configuration. To find the best configuration of metrics all possible combinations between (E_{spec}, E_{sens}, E_{prec}, E_{npv}) have been tested, in conjunction with E_{acc}. E_{acc} was needed to be used to have a metric that determines whether the optimal FT ($E_{\text{acc}} = 0$) has been reached. In the best cases, the subsets configurations would yield performance similar to or slightly worse than FT-MOEA-CM. In the worst cases, they would never reach an optimal FT.

In some cases, other configurations were faster in terms of computation time but did not reach the optimal FT. Considering the definition provided in the Method section (Section 4), this cannot be considered an improvement. Adding more confusion matrix metrics does not results in additional performance overhead, since the confusion matrix is already computed. Based on this information, it was concluded that the most optimal configuration from the confusion matrix was the one used in FT-MOEA-CM.

5.2.2 Accuracy analysis. Considering the evaluated datasets, FT-MOEA-CM reaches the optimal tree faster in all cases in terms of generations Figure 2. It is worth noting that the in the presented graphs, the curve representing the Confusion Matrix Metrics is below the one representing FT-MOEA in any recorded instance.

The distance between the slopes is the greatest in the case of COVID-19 (Figure 2d). When using FT-MOEA, the optimal tree is not reached. This is likely due to the fact that the algorithm gets stuck in local optima. Since it is only guided by three objectives, the genetic mutation operators might not be impactful enough to move out of the local optima towards a better solution. This shows that the Matrix Metrics explore the search space in a way that originally was not possible, which leads not only to faster convergence in terms of generations but also to previously unaccessible optimal trees.

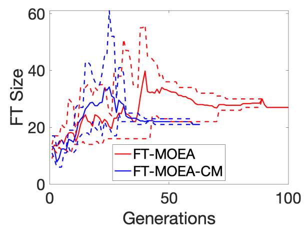
Another observation is that when the FT-MOEA-CM is used, the algorithm tends to deviate considerably less from the average. The value intervals in the case of the FT-MOEA are wider. This shows that the FT-MOEA-CM is more consistent, yielding a more predictable behaviour.

The faster drop towards the optimal solution in the case of FT-MOEA-CM is likely due to the number of metrics used and what they reveal about the problem. Having metrics that represent uncorrelated metrics is useful because each will cause the problem to consider new solutions that move towards the optimal tree. Highly correlated objectives will not add much value since they will not reveal new ways for achieving an optimal solution.

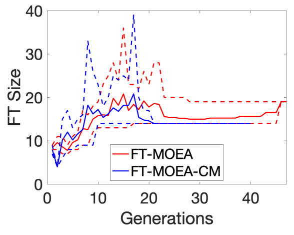
5.2.3 Size analysis. Unlike accuracy, variability in size is similar for both configurations (Figure 3). One common pattern that can be noticed is that the FTs tend to first quickly grow in size and decrease slowly over time until they reach the optimal tree.

In the case of FT-MOEA-CM, the algorithm starts exploring larger FTs quicker compared to FT-MOEA. This is best observed in the GPT dataset (Fig. 3c), where the explored FTs reach sizes over 100 in some cases.

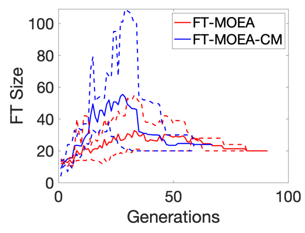
The quicker spike in size might be caused by the increase in objectives. Considering that in NSGA-II each objective has equal weight, adding more objectives decreases the influence of each objective. Size becomes less important, so the algorithm gets to explore larger FT. This is amplified by the fact that MOEAs are generally used to find solutions for conflicting objectives, but in this instance, we only have two real conflicting objectives: the size of the tree and how closely related the FT is to the dataset. The Matrix Metrics, for example, are somewhat correlated, since they all would have an error rate of 0 when the FT corresponds perfectly to the



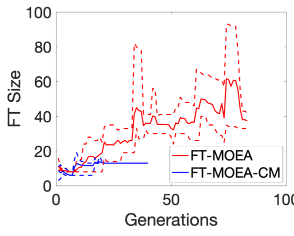
(a) TS1 dataset



(b) MPPS dataset

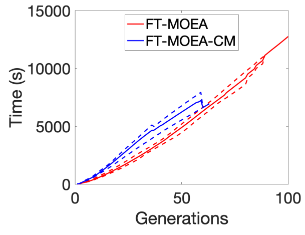


(c) GPT dataset

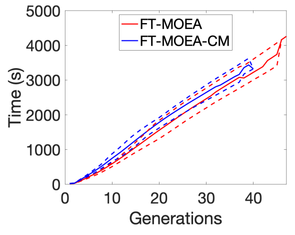


(d) COVID-19 dataset

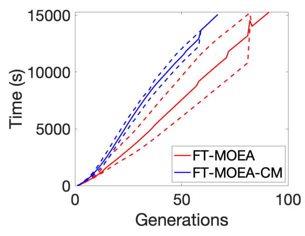
Fig. 3. Visual representation of size change across generations for different datasets (the dotted lines represent the minimum and maximum value used in the computation of the average)



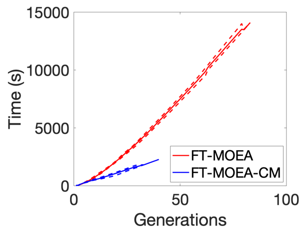
(a) TS1 dataset



(b) MPPS dataset



(c) GPT dataset



(d) COVID-19 dataset

Fig. 4. Visual representation of time across generations for different datasets (the dotted lines represent the minimum and maximum values used in the computation of the average)

dataset. So, in this sense, the weight of the objectives related to the dataset is higher than the size.

Exploring larger FTs comes at the cost of an increase in metrics computation time. This creates a trade-off between exploring larger trees and keeping the algorithm efficient. Since in this case size becomes less important, the algorithm can choose to follow a route with larger FTs instead of one with fewer trees which also leads to convergence in a comparable number of generations.

We can notice that in all cases, FT-MOEA-CM results in a smaller FT, except for the GPT dataset (Fig. 3c). The exploration of larger trees increases the number of generations it takes for the algorithm to reduce its size.

5.2.4 *Time analysis.*

In all the studied cases, the convergence time was faster than the original algorithm (Figure 4). An interesting observation is that in all cases, except for COVID-19, the time per generation is larger in FT-MOEA-CM. As mentioned in the previous subsection (Section 5.2.3), this is likely caused by the larger subtrees explored. The inverse of this is noticeable in the COVID-19 dataset, where FT-MOEA explores larger trees, resulting in a larger time spent per generation.

Looking at the MPPS size chart (Figure 3b), we can see that in both cases a pretty similar pattern is followed resulting in also a similar pattern in terms of time spent. This might indicate that, at least for smaller FTs ϕ_c is not a very time-consuming metric. The precise effects of the metric should be studied further to gather a better understanding of its impact on the performance of the algorithm.

5.2.5 *Analysis conclusion.*

To conclude the analysis of this metric, it is important to understand whether this metric led to an improvement in the algorithm. Earlier, in the problem definition, it was stated that the algorithm would yield an improvement if and only if, either the accuracy of the Matrix Metrics configuration would result in better accuracy than FT-MOEA or if it would result in a better time with no worse accuracy. Considering that in all cases, the algorithm converged faster, the changes in metrics can be considered an improvement to the algorithm.

5.3 Scalability

The old version of the algorithm did not scale very well. It relies on the computation of MCSs for reaching the global optima ($\phi_d = 0$ and $\phi_c = 0$). The problem is that computing MCSs has exponential complexity in the number of BEs, which makes it difficult for the algorithm to handle larger cases.

This problem should be reduced when using CMM. However, during the analysis in Section 5.2, it became clear that in order to reach solutions faster or to reach some solutions at all, larger FTs need to be explored. To calculate a metric, like accuracy, for example, for each row in the data set, it is necessary to waste a number of cycles the size of a tree to find the value a particular configuration of basic events would yield. So if a dataset has 2^{14}, it would take approximately 2^{14} operations to iterate through the dataset. To calculate accuracy, for example, $2^{14} * TreeSize$ would have to be calculated.

Perhaps a solution to this problem to be tried in the future is to save FTs in memory and only compute the confusion matrix for the rows that have been subjected to change based on the events that were changed in the structure of the FT. We can find the rows which affect these events using a similar process to the one described in the Event impact vector distance subsection. It is also likely that multiple FTs move from one generation to the next one. For those, it is also possible to save the metrics in memory to not compute them again.

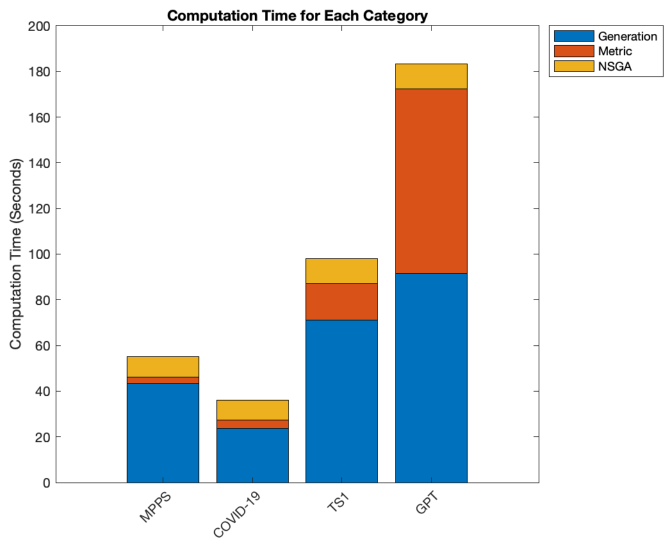


Fig. 5. Computational resources wasted in one iteration of FT-MOEA-CM

Another important problem hindering larger trees' computability is the population generation process. Currently, many similar trees are generated on which metrics are computed, which only wastes computing power. In its current state, the algorithm checks for duplicate identical trees only. This, however, overlooks many identical, slightly different trees. For example, if we consider the following trees T1 = AND(BE1, BE2), T2 = AND(BE1, BE2) and T3 = AND(BE2, BE1), where AND represents a gate to which basic events are connected, then the algorithm would consider T1 to be identical to T2 but not to T3.

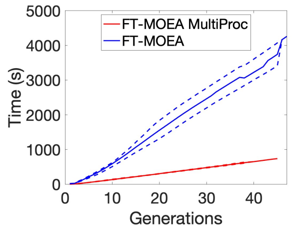
The best solution to scale, however, which would improve the performance most significantly, is the spreading of the computation tasks across multiple CPUs. In the next section, this type of optimization will be discussed.

5.4 Multi-processing

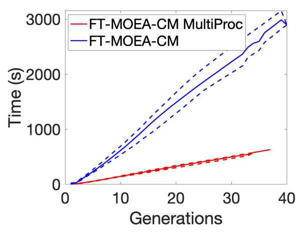
To get a better understanding of how multi-processing might impact the algorithm, for each dataset, for one emulation for each main section of the algorithm: the generations of the population, the metric computation and the NSGA-II sorting, the average time per generation will be calculated with the metric configuration used in the original algorithm. The times have been measured for all the datasets described in Table 2. The bars are organized from left to right in increasing order regarding the number of BEs.

Analyzing Figure 5, the time spent in NSGA-II sorting is likely constant; however, a positive correlation between the number of BEs and elapsed time can be noticed in the population generation and metric computation phases.

Based on these findings, the implementation of the algorithm has been changed to using multiple processes in these two parts of the code. This has been done using the multiprocessing library in Python.



(a) Time performance using FT-MOEA metric configuration



(b) Time performance using FT-MOEA-CM metric configuration

Fig. 6. Multiprocessing time analysis (the dotted lines represent the minimum and maximum values used in the computation of the average)

Figure 5 presents the algorithm's behaviour when multiprocessing with 16 CPUs. A significant improvement in performance in both the original configuration of the FT-MOEA algorithm and FT-MOEA-CM can be noticed. Additionally, it can be seen that using multiprocessing yields more consistent results as the deviation between runs in terms of time becomes smaller.

These results are promising. However, more work is needed to understand the full effects of this optimization better. It would be useful to understand better the relation between the computation time of the population generation phase in the algorithm and the size of the FT.

We can see that the results presented in Figure 6 are conformed to the bar chart presented in Figure 5. If we consider the time performance when multiprocessing in both cases presented in Figure 6. the time difference is negligible. This is because the metric computation represents a smaller time portion in this dataset.

5.5 Other metrics

The other metrics that were discussed in Section 4.3.2 and Section 4.3.3 did not show a significant improvement. When used instead of ϕ_c the results were far worse. In some cases, there was some improvement when used with it. The following subsections will explain the reasoning behind the lack of improvement.

5.5.1 *Random dataset segmentation.*

This metric has shown limited improvements when used with FT-MOEA. The results, however, were inconsistent and only noticeable when using a segment size of 4. When using a segment size of 2, there was essentially no change in performance, likely since this metric would resemble ϕ_d . When using a segment size of 8 there would also be no change in performance. This is probably caused by the fact that the genetic operators used do not change the FT enough for it to increase under such conditions.

The positive results in the case of using a segment size of 4 might be attributed to the reasons stated in the method section (Section 4). It is also likely that its presence lowered the influence of the size objective, resulting in better performance.

Additionally, the value of this metric might yield different results for the same FT. This is problematic because, in some cases, worse FTs might be selected over better FTs due to the random nature of the algorithm.

5.5.2 *Event impact vector distance.*

This metric had no improvement in performance at all. This is likely due to the reasons stated in the methods section (Section 4), precisely that a scalar value is not a good fit for calculating differences between many vectors with the goal of improving them.

6 RELATED WORK

The area of FT inference using MOEAs is relatively unexplored, posing a unique challenge when seeking direct literature precedents. Few studies delve into the details of fitness functions in genetic algorithms, key areas of this research.

Generally speaking, there are different approaches to inferring FTs: knowledge-based, model-based and data-driven. Carpiagno and Poucet [1] discuss multiple approaches to knowledge-based fault tree inference. They focus on representing what is known about the system through knowledge representation models, later used to infer the FT. Dickerson et al[5] discuss a model-based approach to inferring fault trees from an existing model, precisely UML diagrams. This approach focuses on essentially translating an existing model to an FT. In this study the focus will be placed on data-driven approaches, precisely those that use evolutionary algorithms.

In the context of evolutionary algorithms, the first attempt was carried by Linard et al. [9]. The paper describes a genetic algorithm for inferring FTs using a one-dimensional fitness function based on accuracy based on failure data sets. The main drawback of the algorithm is that it can grow into arbitrarily large trees, because size is not considered. Additionally, since it does not use other metrics, it converges slower.

The paper by Jimenz-Roa et al. [7] introduces the use of MOEA to infer FTs from failure data sets. The algorithm presented in the paper uses the NSGA-II sorting method. Additionally, it uses additional objectives related to MCSs, and accuracy in size. Compared to the algorithm presented by Linard et al. [9], it leads to faster convergence and smaller FTs. The main drawbacks of the algorithm are the slow computation of MCS and its inability to handle noisy data.

This research [8] introduced the use of symmetries and modules to optimise the inference process of FTs when using MOEAs. It focuses on identifying similar parts in the FT structure for faster computation of the FT. It essentially uses a similar algorithm to the one used by Jimenz-Roa et al. [7], adding heuristics for faster computation. However, it does not directly provide any insights about improving evolutionary algorithms in this context.

7 FUTURE WORK

Although this paper has brought some contributions to the initial algorithms, there are still multiple aspects of the algorithm to study that could improve it.

- *Further Confusion Matrix Metrics Analysis* Although some of the more basic metrics have been used in this paper, other interesting metrics must be studied. Metrics such as the Matthews correlation coefficient and Balanced accuracy could improve the algorithm's performance. According to the relevant literature, they might be better compared to accuracy [2, 3].
- *Noisy Data* In the original FT-MOEA paper, there was testing done on noisy data. Originally, it was not possible to calculate the MCSs under such conditions. FT-MOEA with the metrics presented here should deal better with noisy data, but further research is needed.
- *Deeper Algorithm Performance Analysis* Some interesting results have emerged from using multiprocessing. There still are odd behaviours, like only using confusion matrix metrics taking longer to compute compared to the initial configuration of the algorithm. An in-depth analysis in this regard could yield performance improvements. Also, it would be useful to further understand in which way parts of the algorithm affect its performance.
- *Caching Metric Data* Other improvements could be made from the perspective of the software. Likely, using some sort of caching to compute metrics in the FT could show positive results. Computing the metrics more smartly is likely possible. Considering a mutation operation in most cases results in the change of a single event or a gate, it is not necessary to recompute the metric for the entire dataset, but only for the affected portion of the dataset by that change.
- *Improved Equivalence Comparison in FTs* Creating a more powerful filter that would prevent equal FTs from being evaluated twice will likely result in better performance.

8 CONCLUSION

The problem tackled in this research is improving the performance of the FT-MOEA algorithm. The focus was placed on identifying metrics that would replace a metric based on MCS, considering it was problematic from a complexity perspective and would prevent the algorithm from working with noisy data.

Several new metrics were identified, such as the Random segmentation metric, Event impact vector metric and Confusion Matrix matrix. Their performance was evaluated in relation to accuracy, time, size and number of generations until the algorithm's convergence. It was concluded that the first two metrics did not show much improvement. Still, the algorithm based on the confusion matrix has shown significant improvements in terms of time, accuracy and generations until convergence. The new metric configuration performs consistently across various failure data sets varying in BE count.

The findings have notable implications in the field of multi-objective evolutionary algorithms that focus on the data-driven inference of models, especially in the case of FTs, by outlining the potential of the use of confusion matrix metric in improving algorithm performance.

REFERENCES

[1] A. Carpignano and A. Poucet. 1994. Computer Assisted Fault Tree Construction: A review of methods and concerns. *Reliability Engineering amp;amp; System Safety* 44, 3 (1994), 265–278. https://doi.org/10.1016/0951-8320(94)90018-3

[2] Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 21, 1 (Jan 2020). https://doi.org/10.1186/s12864-019-6413-7

[3] Davide Chicco, Niklas Tötsch, and Giuseppe Jurman. 2021. The Matthews Correlation Coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData Mining* 14, 1 (Feb 2021). https://doi.org/10.1186/s13040-021-00244-z

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. https://doi.org/10.1109/4235.996017

[5] Charles E. Dickerson, Rosmira Roslan, and Siyuan Ji. 2018. A formal transformation method for automated fault tree generation from a UML activity model. *IEEE Transactions on Reliability* 67, 3 (2018), 1219–1236. https://doi.org/10.1109/tr.2018.2849013

[6] A.E. Eiben and J.E. Smith. 2015. *Introduction to evolutionary computing*. Springer. 25–48 pages.

[7] Lisandro A. Jimenez-Roa, Tom Heskes, Tiedo Tinga, and Marielle Stoelinga. 2023. Automatic inference of fault tree models via multi-objective evolutionary algorithms. *IEEE Transactions on Dependable and Secure Computing* (2023), 1–12. https://doi.org/10.1109/tdsc.2022.3203805

[8] Lisandro Arturo Jimenez-Roa, Matthias Volk, and Mariëlle Stoelinga. 2022. Data-driven inference of fault tree models exploiting symmetry and modularization. *Lecture Notes in Computer Science* (2022), 46–61. https://doi.org/10.1007/978-3-031-14835-4_4

[9] Alexis Linard, Doina Bucur, and Mariëlle Stoelinga. 2019. Fault trees from data: Efficient Learning with an evolutionary algorithm. *Dependable Software Engineering. Theories, Tools, and Applications* (2019), 19–37. https://doi.org/10.1007/978-3-030-35540-1_2

[10] Faida Mhenni, Nga Nguyen, and Jean-Yves Choley. 2014. Automatic fault tree generation from SysML system models. *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (2014). https://doi.org/10.1109/aim.2014.6878163

[11] Patryk Miziula and Jorge Navarro. 2019. Birnbaum importance measure for reliability systems with dependent components. *IEEE Transactions on Reliability* 68, 2 (2019), 439–450. https://doi.org/10.1109/tr.2019.2895400

[12] OpenAI. 2023. GPT-4. https://openai.com/research/gpt-4.

[13] Pedro J. Ponce de León, José M. Iñesta, Jorge Calvo-Zaragoza, and David Rizo. 2016. Data-based melody generation through multi-objective evolutionary computation. *Journal of Mathematics and Music* 10, 2 (may 3 2016), 173–192. [Online; accessed 2023-06-09].

[14] T. Devi Prasad and Nam-Sik Park. 2004. Multiobjective genetic algorithms for design of water distribution networks. *Journal of Water Resources Planning and Management* 130, 1 (1 2004), 73–82. [Online; accessed 2023-06-09].

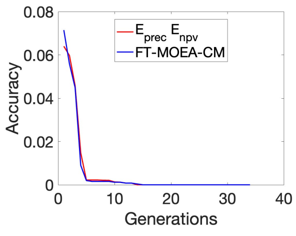
[15] A. Rauzy. 2001. Mathematical foundations of minimal cutsets. *IEEE Transactions on Reliability* 50, 4 (2001), 389–396. [Online; accessed 2023-06-09].

[16] Tomasz G. Smolinski. 2014. *Multi-objective evolutionary algorithms*. Springer New York, New York, NY, 1–4. [Online; accessed 2023-06-09].

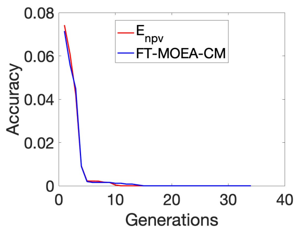
[17] W. E. Vesely. 1981. *Fault Tree Handbook*. Nuclear Regulatory Commission. 8–22 pages. [Online; accessed 2023-06-09].

[18] Jianwen Xiang, Kazuo Yanoo, Yoshiharu Maeno, and Kumiko Tadano. 2011. Automatic synthesis of static fault trees from System Models. *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement* (2011). https://doi.org/10.1109/ssiri.2011.32

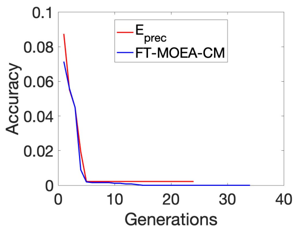
Appendices



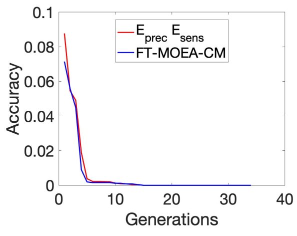
(a) E_{prec}+E_{npv} vs. FT-MOEA-CM



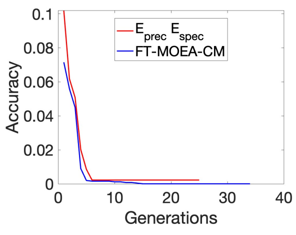
(b) E_{npv} vs. FT-MOEA-CM



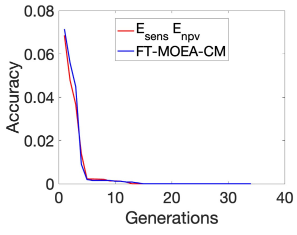
(c) E_{prec} vs. FT-MOEA-CM



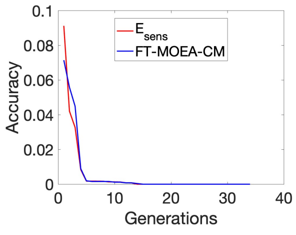
(d) E_{prec}+E_{sens} vs. FT-MOEA-CM



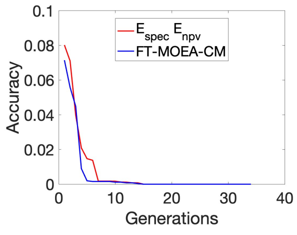
(e) E_{prec}+E_{spec} vs. FT-MOEA-CM



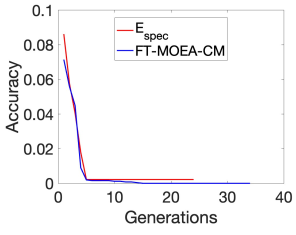
(f) E_{prec} vs. FT-MOEA-CM



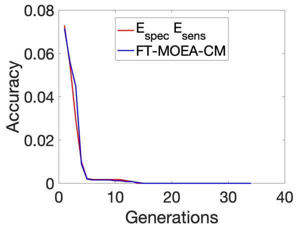
(g) E_{sens} vs. FT-MOEA-CM



(h) E_{spec}+E_{npv} vs. FT-MOEA-CM

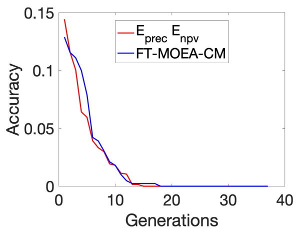


(i) E_{spec} vs. FT-MOEA-CM

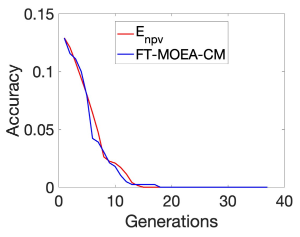


(j) E_{spec}+E_{sens} vs. FT-MOEA-CM

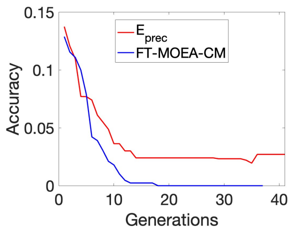
Fig. 7. Visual representation of accuracy change across generations for different configurations evaluated on dataset COVID-19. In this context, Accuracy can be considered equivalent to ϕ_d.



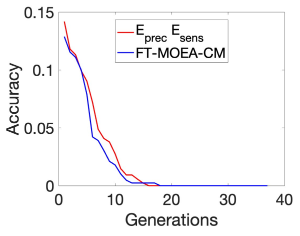
(a) E_{prec}+E_{npv} vs. FT-MOEA-CM



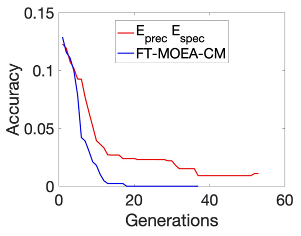
(b) E_{npv} vs. FT-MOEA-CM



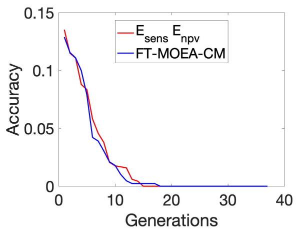
(c) E_{prec} vs. FT-MOEA-CM



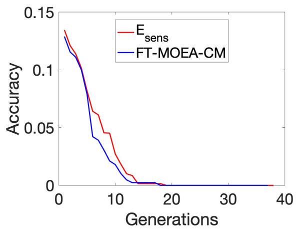
(d) E_{prec}+E_{sens} vs. FT-MOEA-CM



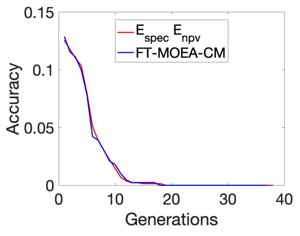
(e) E_{prec}+E_{spec} vs. FT-MOEA-CM



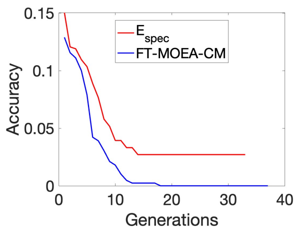
(f) E_{prec} vs. FT-MOEA-CM



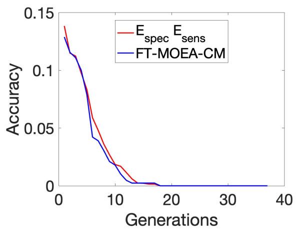
(g) E_{sens} vs. FT-MOEA-CM



(h) E_{spec}+E_{npv} vs. FT-MOEA-CM

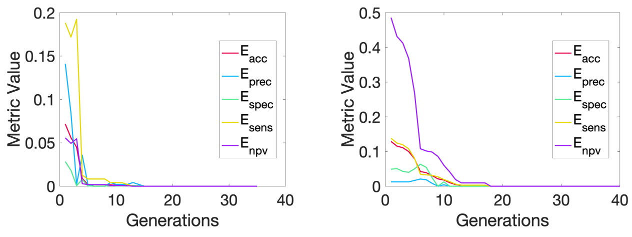


(i) E_{spec} vs. FT-MOEA-CM



(j) E_{spec}+E_{sens} vs. FT-MOEA-CM

Fig. 8. Visual representation of accuracy change across generations for different configurations evaluated on dataset MPPS. In this context, Accuracy can be considered equivalent to ϕ_d.



(a) COVID-19 dataset (b) MPPS dataset

Fig. 9. Metric behaviour in MPPS and COVID-19 datasets across generations in FT-MOEA-CM.

A PLOTS FOR DIFFERENT CONFIGURATIONS OF CONFUSION MATRIX METRIX

In this appendix, different configurations of the confusion matrix metrics will be presented in comparison with FT-MOEA-CM in Figure 7 and Figure 8. E_{acc} was used in conjunction with the metric combinations specified in all cases. In Figure 9, the behaviour of different metrics is presented as the algorithm approaches the optimal solution. These measurements were used to find the optimal configuration of the algorithm and gain a better understanding of how it works.