A Genetic and Graph-Guided Feature Learning Strategy for Improving Decision Tree Construction

Nour El Islem Karabadji, Abdelaziz Amara Korba, Ali Assi, Hassina Seridi, Mohamed Aimen, Yacine Ghamri-Doudane, Abdelghani Lakhdari, Mohamed Elati, Wajdi Dhifli

A Genetic and Graph-Guided Feature Learning Strategy for Improving Decision Tree Construction

Nour El Islem Karabadji[a,b,h] (n.karabadji@ensti-annaba.dz), Abdelaziz Amara Korba[c,d] (abdelaziz.amara_korba@univ-lr.fr), Ali Assi[e] (axacad5@rit.edu), Hassina Seridi[b](seridi@labged.net), Mohamed Aimen Karabadji[g] (ma.karabadji@insidjam.com), Yacine Ghamri-Doudane[d] (yacine.ghamri@univ-lr.fr), Abdelghani LAKHDARI[a] (a.lakhdari@ensti-annaba.dz), Mohamed Elati[f] (mohamed.elati@univ-lille.fr), Wajdi Dhifli[f] (wajdi.dhifli@univ-lille.fr).

[a] National Higher School of Technology and Engineering, Laboratoire De Technologies Des Systemes Energetiques (LTSE) E3360100, 23005, Annaba, Algeria.
[b] Electronic Document Management Laboratory (LabGED), Badji Mokhtar-Annaba University, P.O. Box 12 Annaba, Algeria.
[c] Networks and Systems Laboratory (LRS), Badji Mokhtar-Annaba University, P.O. Box 12 Annaba, Algeria.
[d] L3i, University of la Rochelle, Av. Michel Crépeau, 17042 La Rochelle, France.
[e] Rochester Institute of Technology, Dubai, United Arab Emirates.
[f] CANTHER, University of Lille, CNRS UMR 1277, Inserm U9020, 59045 Lille, France.
[g] Insidjam ERP Villa ELDER, Boumerdes, Algeria.
[h] Environmental Research Centre, Annaba, Algeria.

Corresponding Authors:
Nour El Islem Karabadji
National Higher School of Technology and Engineering, Laboratoire De Technologies Des Systemes Energetiques (LTSE) E3360100, 23005, Annaba, Algeria.
Tel: (+213) 38 43 84 02
Fax: +213 (0) 38 43 84 01
n.karabadji@esti-annaba.dz

Wajdi Dhifli
CANTHER, University of Lille, CNRS UMR 1277, Inserm U9020, 59045 Lille, France.
Tel.: +33 (0)3 20 96 40 40
Fax: +33 (0)3 20 95 90 09
wajdi.dhifli@univ-lille.fr

A Genetic and Graph-Guided Feature Learning Strategy for Improving Decision Tree Construction

Nour El Islem Karabadji[a,b,h,*], Abdelaziz Amara Korba[c,d], Ali Assi[e], Hassina Seridi[b], Mohamed Aimen Karabadji[g], Yacine Ghamri-Doudane[d], Abdelghani Lakhdari[a], Mohamed Elati[f] and Wajdi Dhifli[f,*]

[a]*National Higher School of Technology and Engineering, Laboratoire De Technologies Des Systemes Energetiques (LTSE) E3360100, 23005, Annaba, Algeria.*

[b]*Electronic Document Management Laboratory (LabGED), Badji Mokhtar-Annaba University, P.O. Box 12 Annaba, Algeria.*

[c]*Networks and Systems Laboratory (LRS), Badji Mokhtar-Annaba University, P.O. Box 12 Annaba, Algeria.*

[d]*L3i, University of la Rochelle, Av. Michel Crépeau, 17042 La Rochelle, France.*

[e]*Rochester Institute of Technology - RIT Dubai - United Arabe Emirates.*

[f]*CANTHER, University of Lille, CNRS UMR 1277, Inserm U9020, 59045 Lille, France.*

[g]*Insidjam ERP Villa ELDER, Boumerdes, Algeria.*

[h]*Environmental Research Centre, Annaba, Algeria.*

ARTICLE INFO

Keywords:
Decision Tree
Feature selection
Feature Construction
Genetic Algorithm
Internet of Vehicles

Abstract

Machine learning algorithms have offered unprecedented solutions for many real-world problems. These algorithms frequently involve using a large number of features. However, several of these features could not be very informative due to data uncertainties, such as noise and residual variation. Decision trees are among the most preferred classification models. This is due to their simplicity, explainability, and readability. However, data inaccuracies could impact the construction of decision trees and thus hinder their results. Feature selection and construction present promising research direction to enhance the performance of decision tree models. In this paper, we present a strategy that combines feature selection and construction where the construction of new features is performed by using the ones that were not chosen during the selection step. However, the search space of combinations of selected/constructed features is extremely large. To find the best solution, a genetic algorithm has been developed combined with a graph covering vertices set guided approach. The obtained results on a large number of datasets from the UCI Repository demonstrate that our approach outperforms both recent and classical decision tree construction techniques. We also present a successful use case of our approach in detecting Botnet traffic in the Internet of Vehicles.

1. Introduction

Classification is one of the most important predictive modeling problem. Out of the different existing classification algorithms, decision tree (DT) is considered to be a prominent and robust classifier. It has proven its efficiency as a standalone classifier (Karabadji, Khelf, Seridi, Aridhi, Remond and Dhifli, 2019) as well as a weak learner in ensemble methods (Karabadji, Korba, Assi, Seridi, Aridhi and Dhifli, 2023). Decision tree uses a set of rules to make decisions where the dataset is split repeatedly into subsets of samples. The splitting is based on the most significant features (splitting features) in the input samples, and this step is repeated recursively until all samples belonging to the same class (leaf) are isolated (or the decision tree reaches a predefined maximum depth). In general, the splitting step generates large complex DTs that may include data uncertainties. Pruning techniques are used to address this issue. They allow to discard one or more branches (sub-trees) from the DT including noisy or irrelevant data. However, pruning the overgrowing tree may lead to underfitting or overfitting. Therefore, finding the right balance between complexity and accuracy is a critical aspect of decision tree modeling (Quinlan, 1987).

*Corresponding author.

n.karabadji@ensti-annaba.dz (N.E.I. Karabadji); abdelaziz.amara.korba@univ-annaba.org (A. Amara Korba); axacad5@rit.edu (A. Assi); seridi@labged.net (H. Seridi); ma.karabadji@insidjam.com (M.A. Karabadji); yacine.ghamri@univ-lr.fr (Y. Ghamri-Doudane); a.lakhdari@ensti-annaba.dz (A. Lakhdari); mohamed.elati@univ-lille.fr (M. Elati); wajdi.dhifli@univ-lille.fr (W. Dhifli)

ORCID(s):

In classification, features that are irrelevant or redundant can hinder the accuracy, and increase the complexity of the classification model and the running time (Karabadji, Seridi, Bousetouane, Dhifli and Aridhi, 2017). The objective of feature selection is to select a subset of variables that effectively characterizes the input data, while also reducing the influence of noise or insignificant variables (Chandrashekar and Sahin, 2014; Tan, Gui and Qiu, 2024; SabbaghGol, Saadatfar and Khazaiepoor, 2024). Feature construction methods, on the other hand, are utilized to generate new, higher-level features from the original ones. This is done to reduce the dimensionality of the features, enhance the classification performance, and can reveal hidden/weak signals in the original features (Vouk, Guid and Robnik-Šikonja, 2023).

Feature selection and construction can be considered as alternative methods to tackle the complexity-accuracy balance (discussed earlier), and to improve the classification performances of DTs (Ma and Gao, 2020b). Typically, these techniques are used during the preprocessing as filters. However, although it is more complex, feature selection and construction can also be embedded and performed during the model construction.

Combining feature selection and construction can reduce feature dimensionality, leading to improved classification results (Tran, 2018). Therefore, adopting this solution will enable us to improve the construction of a compact decision tree with superior classification capabilities.

The main contributions of the paper are as follows:

- We present a novel method for improving the performances of decision trees by performing both feature selection and the construction of new features based on the ones that were not selected.
- As the search space is extremely large, we developed a genetic algorithm that employs a graph-guided strategy to discover the optimal solution among the numerous potential candidates.
- We used a wide range of datasets selected from the UCI machine learning repository to experimentally evaluate the proposed method against existing competitors. This evaluation shows the effectiveness of the proposed approach.
- We further evaluate our method on a real world application for detecting Botnet traffic in the Internet of Vehicles (IoV), using the experimental framework developed in (Rahal, Amara Korba, Ghoualmi-Zine, Challal and Ghamri-Doudane, 2022). The obtained results show the competitivity and high performance of our approach.

The remaining sections of the paper are organized as follows. In Section 2, we explore existing research on the construction of decision trees. In Section 3, we describe the preliminaries and used notations. In Section 4, we discuss the proposed decision tree construction method. Section 5 reports the benchmark datasets and the experimental setup configuration. The experimental results on benchmark datasets are presented in Section 6. In Section 7, we demonstrate the validity of the proposed approach on a real-world application for Botnet traffic detecting in the IoV. Finally, the conclusion of the paper is presented in Section 8.

2. Related Works

Many approaches have been proposed to develop effective ways of building optimized decision trees. In this context, evolutionary and meta-heuristic approaches have often been used (Fu, Golden, Lele, Raghavan and Wasil, 2006; Otero, Freitas and Johnson, 2012; Boryczka and Kozak, 2015; Karabadji et al., 2017, 2019; Liu, Lin, Lai and Miao, 2022). Based on swarm intelligence and using Ant-Colony Optimization, (Otero et al., 2012; Boryczka and Kozak, 2015) propose approaches to build trees. Ant-Tree-Miner (Otero et al., 2012) selects the nodes using the amount of pheromone. It uses the information gain ratio from the C4.5 approach as a heuristic function. ACDT (Boryczka and Kozak, 2015) follows the same principle as in Ant-Tree-Miner. It applies a modified decision criteria to select decision nodes where each ant creates a decision tree. (Karabadji et al., 2019) proposes an approach based on the particle swarm optimization (PSO). This approach identifies the optimal combination of learning samples and feature subsets to construct a decision tree model. This model offers the best generalization on a given training set and overcomes both tree size and overfitting problems.

In their study, the authors of (Fu et al., 2006) proposed a method that begins with an initial population of trees. Subsequently, genetic operations are applied to refine this population. The initial tree population, originally generated using the C4.5 method, undergoes a correction and pruning process.

Similarly, in (Karabadji et al., 2017) the authors introduced an evolutionary meta-heuristic optimization-based approach for identifying the optimal settings during the construction of a decision tree. Their approach involves utilizing a genetic algorithm in conjunction with a multi-objective function to extract the most suitable decision tree. The objective function takes into account several factors, including precision with test samples, the reliability of constructing the decision tree with the smallest feasible training set and the largest possible testing set, as well as the choice of feature set.

Numerous studies have focused on the features selection and/or construction. The results obtained have demonstrated a significant improvement in the performance of decision trees. In (Tran, Xue and Zhang, 2019), the authors explored a variety of methods for generating multiple features and evaluated their efficiency and underlying characteristics. This analysis provided valuable insights about the process of constructing multiple features using genetic programming on high-dimensional data. (Ma and Gao, 2020b) proposes a hybrid approach that employs genetic programming based on feature construction and selection. They suggest creating multiple features before selecting the most effective ones based on combine two filter methods.

Two methods for constructing genetic programming based classifiers have been proposed in (Ma and Gao, 2020a) to mitigate the negative impact of irrelevant and redundant features. The first method (GPMO) is based on a multiple-objective fitness function that simultaneously minimizes the classification error rate and the number of selected features. The second method (FSGPMO) involves using a feature selection technique, such as linear forward selection (LFS), to eliminate irrelevant and redundant features before constructing classifiers using GPMO.

PSOFC (Xue, Zhang, Dai and Browne, 2013) is a PSO-based feature construction approach. It builds a single high-level feature from the original low-level features to directly solve binary classification problems. BFC-GA (Hammami, Bechikh, Louati, Makhlouf and Said, 2020) is a bi-level evolutionary method for features construction. An upper-level population is used to select features, and a lower-level population is used to find the best combinations of features. EFC (Explainable Feature Construction) (Vouk et al., 2023) reduces the large search space of constructive induction to a linear space of co-occurring features. This approach permits to construct various types of features including logical, relational, Cartesian and numerical operators. These new features are built from rule learning and threshold-based features.

3. Preliminaries and definitions

This section presents the preliminaries and definitions related to the Power-set system and graph minimum vertex cover.

Definition 1. ***(Power-set system)*** *Let E be a finite set. With the subset relation $\subseteq$, the power-set system $\mathcal{P}(E)$ is a partially ordered set, also known as poset, composed of all possible subsets of E, $(\mathcal{P}(\mathcal{E}), \subseteq)$.*

Given a set of features $\mathcal{AT}$ of size n. The power-set $\mathcal{P}(\mathcal{AT})$ has 2^n subsets of possible combinations of features. Each subset in $\mathcal{P}(\mathcal{AT})$ is denoted by $\mathcal{AT}_i$ where i is an integer in $[0,\ 2^n]$. This subset can be identified by its size and identifier denoted by $S_{\mathcal{AT}_i}$ and $ID_{\mathcal{AT}_i}$, respectively. $\mathcal{P}(\mathcal{AT})$ is a lattice such that $\forall \mathcal{AT}_i, \mathcal{AT}_j \in \mathcal{P}(\mathcal{AT})$, the least upper bound is $\mathcal{AT}_i \cup \mathcal{AT}_j \in \mathcal{P}(\mathcal{AT})$ and the greatest lower bound is $\mathcal{AT}_i \cap \mathcal{AT}_j \in \mathcal{P}(\mathcal{AT})$.

A lattice $\mathcal{P}_{AT}$ can be organized as follows:

- Every subset of features, *i.e.* $\mathcal{AT}_i$, is a node in $\mathcal{P}_{\mathcal{AT}}$.
- The bottom node corresponds to the empty set $\emptyset$ and the top node corresponds to the $\mathcal{AT}$ set. Thus, $\emptyset$ and $\mathcal{AT}$ are the least element and the greatest element of $\mathcal{P}_{\mathcal{AT}}$, respectively.
- $\mathcal{P}_{\mathcal{AT}}$ consists of $n+1$ ranges where $n = |\mathcal{AT}|$. Each range k, denoted by R_k where $k \in \{0, \dots, n\}$, consists of $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ features subsets of size k. For example, range R_0 includes the empty subset $\emptyset$, R_1 contains all the subsets with one feature, and so forth.

Definition 2. ***(Child node).*** *A node $\mathcal{AT}_x$ is a child of a node $\mathcal{AT}_y$ in a lattice $\mathcal{P}_{\mathcal{AT}}$, if the following properties hold:*

1. $\mathcal{AT}_x \subset \mathcal{AT}_y$
2. $\mathcal{AT}_x$ *and* $\mathcal{AT}_y$ *differ by exactly one feature.*

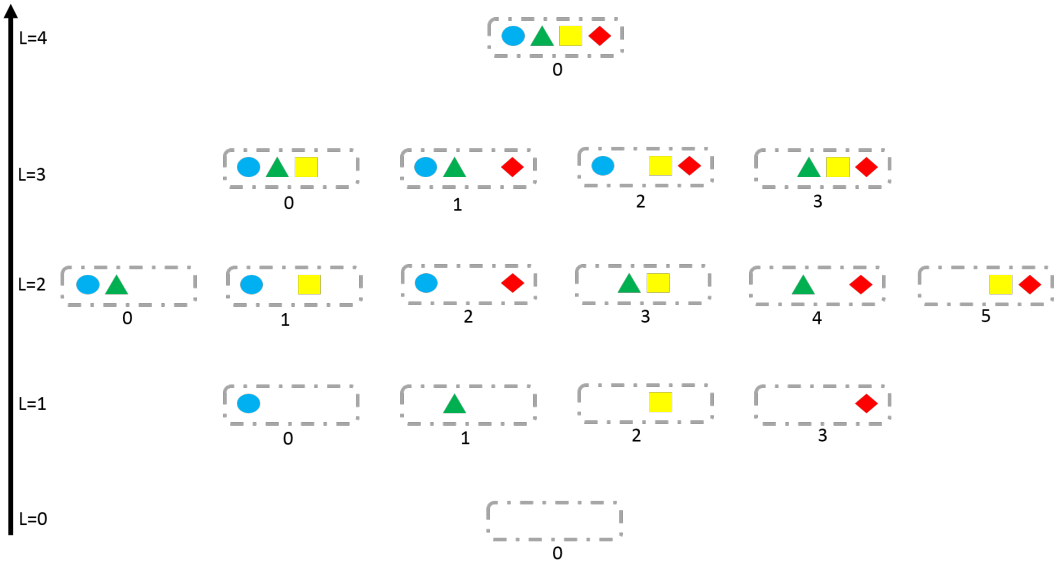

Figure 1: An illustrative example of a power-set system.

A child node $\mathcal{AT}_x$ is called more general (respectively more specific) than a node $\mathcal{AT}_y$, if $\mathcal{AT}_x$ and $\mathcal{AT}_y$ differs by more than one feature.

In addition to the $\subseteq$ order between subsets at different levels, two other linear orders are defined. The first one, denoted by $\preceq$, is defined between subsets at the same range of the power-set. This order is introduced as a lexicographic order between features in the same subset where $\mathcal{AT}_v \preceq \mathcal{AT}_w \Rightarrow v \leq w$. The second one, denoted by $\trianglelefteq$, introduces a linear lexicographic order between ordered feature sets having equivalent sizes. An ordered feature set $\mathcal{AT}_i = \{a_{i_0}, a_{i_1}, \ldots, a_{i_t}, \ldots, a_{i_n}\}$ is lexicographically more general than an ordered features set $\mathcal{AT}_j = \{a_{j_0}, a_{j_1}, \ldots, a_{j_t}, \ldots, a_{j_n}\}$, if and only if the equation 1 is true:

$$\exists t, 0 \leqslant t \leqslant n, a_{i_k} = a_{j_k} \text{ and } a_{i_t} \preceq a_{j_t}, \ \forall\, k < t \tag{1}$$

Example 1. *Let $\mathcal{AT} = \{$●,▲,■,◆$\}$ be a set of four features. Figure 1 illustrates the power-set $\mathcal{P}(\mathcal{AT})$ where:*

1. *The subsets $\mathcal{AT}_i$ are arranged by their sizes from 0 to $|\mathcal{AT}| = 4$;*
2. *At each level, subsets $\mathcal{AT}_i$ are indexed by their position at this level (i.e., chain).*

According to this formalization, the set $\{$●,▲$\}$, at the range 2, can be identified by its size $S_{\mathcal{AT}_2} = 2$ and position $ID_{\mathcal{AT}_2} = 0$.

In the following, graph related definitions will be presented. These definitions will be used during the feature construction (see Section 4.2).

Definition 3. ***(Vertex Cover).*** *A vertex cover, denoted by $\mathcal{VC}$, in a graph G is a set of vertices such that each edge has at least one of its two end points in this set.*

Definition 4. ***(Minimum Vertex Cover).*** *A minimum vertex cover in a graph G is a vertex cover that has the smallest number of vertices among all possible vertex covers.*

Example 2. *In Figure 2, the set $\{1, 2, 4, 8\}$ is a minimum vertex cover of the graph G since it is a vertex cover and there is no other vertex cover with fewer vertices.*

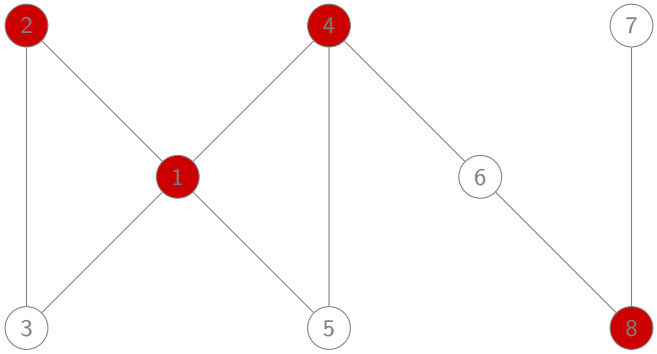

Figure 2: An undirected graph G. Red vertices represent a minimum vertex cover in G.

4. Evolutionary mining of optimal decision tree

This Section depicts our genetic algorithm-based approach for decision tree construction. This approach contains three stages. First, encoding and decoding operations are presented. Next, the feature construction process is introduced. Then, a fitness function is defined. Finally, a genetic optimization step is applied to build an optimal decision tree.

4.1. Encoding and decoding of chromosomes

This phase is a crucial step in the genetic algorithm. It allows to appropriately represent the candidate solutions in a format that can be processed by the genetic algorithm.

4.1.1. The encoding phase

The purpose of the encoding phase consists of defining an injective function that represents each candidate solution as a binary string of 0s and 1s.

Figure 3 illustrates an example of a representation of a chromosome. This chromosome consists of 4 decision variables:

1. $gene_0$: represents the variable is_pruned.
2. $gene_1$: represents the variable $S_{\mathcal{AT}_i}$.
3. $gene_2$: represents the variable $ID_{\mathcal{AT}_i}$.
4. $gene_3$: represents the variable t_{cut}.

These four genes encode distinct pieces of information. $gene_0$ encodes in one bit (0 for unpruned or 1 for pruned), if the constructed decision tree will use pruning or not. $gene_1$ and $gene_2$ are used to represent the selected set of features. $gene_3$ is used to encode the cut-off threshold t_{cut} that will be used later to prune the similarity graph between the non selected features (detailed later). Each of these genes will be encoded using X bits. $S_{\mathcal{AT}_i}$ is an integer in the interval $\left[0, |\mathcal{AT}|\right]$. It represents the size of the selected feature set. $ID_{\mathcal{AT}_i}$ is another integer encoded in $gene_2$ in the interval $\left[0 \ldots \binom{|AT|}{S_{\mathcal{AT}_i}}\right]$. Based on both $ID_{\mathcal{AT}_i}$ and $S_{\mathcal{AT}_i}$, we can identify the corresponding selected subset of features $\mathcal{AT}_i$. The $gene_3$ encodes an integer in the interval [1, 9]. This integer represents a cut-off threshold that will be used to construct new variables from the set of the not selected features.

4.1.2. The decoding phase

In this step, the four encoded decision variables are transformed into is_pruned binary value, a pair of selected features set and a cut-off threshold. To achieve that, the following steps are applied. First, the identifiers of all the sets of features (*i.e.*, in the lattice) are determined. Then, the selected sets of features and the new constructed ones are computed.

In fact, we start by computing the different decision variables. Formally:

(P) Decoding $gene_0$ (i.e., the first bit) yields a binary value, either 0 or 1, which indicates whether a pruning phase will be undertaken or not.

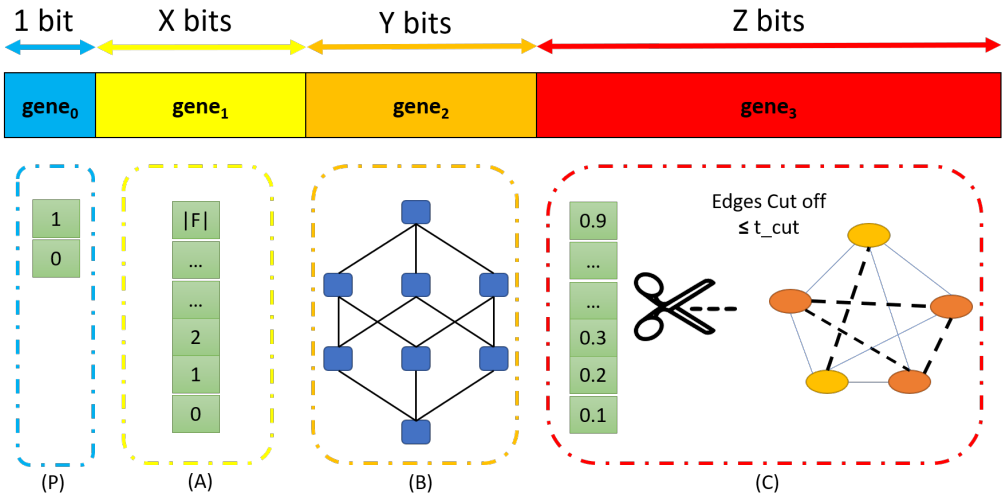

Figure 3: Representation of a chromosome.

(A) The size of the selected subset of features, represented by $gene_1$, is encoded as a binary string of X_1 bits. Decoding $gene_1$ produces an integer $S_{\mathcal{AT}_i} \in [0 \dots |\mathcal{AT}|]$ defined here by: $S_{\mathcal{AT}_i} = \text{Integer\_In}(X_1)\ modulo\ (|\mathcal{AT}|)$.

(B) The identifier of the subset of the selected features, represented by $gene_2$, is encoded using X_2 bits. Decoding $gene_2$ gives an integer $ID_{\mathcal{AT}_i} \in [0 \dots \binom{|\mathcal{AT}_i|}{S_{\mathcal{AT}_i}}]$ defined as follows: $ID_{\mathcal{AT}_i} = \text{Integer\_In}(X_2)\ modulo\ (\binom{|\mathcal{AT}_i|}{S_{\mathcal{AT}_i}})$.

(C) The value encoded in $gene_3$ consists of X_3 bits. Decoding $gene_3$ gives an integer in the interval $[0 \dots 9]$ that allows to define the cut-off threshold t_{cut} as follows: $t_{\text{cut}} = \frac{\text{Integer\_In}(X_3)\ modulo(10)}{10}$.

The second step focuses on 1) generating the selected features and 2) constructing the new features using the not selected *irrelevant* ones. In fact, using the integers $S_{\mathcal{AT}_i}$ and $ID_{\mathcal{AT}_i}$, the set of selected features is generated based on the Algorithm 1 and the equation (2). Formally:

$$ID_{\mathcal{AT}_0} + \sum_{x=1} \frac{(n - (\mathcal{AT}_i[a] + x))!}{(k-a)! * ((n - (\mathcal{AT}_i[a] + x)) - (k-a))!} \leq ID_{\mathcal{AT}_i} \tag{2}$$

where $ID_{\mathcal{AT}_0}$ is the identifier of the subset $\mathcal{AT}_0$. Starting from this first subset $\mathcal{AT}_0$, we can pick out all the elements from $\mathcal{AT}_i[1]$ to $\mathcal{AT}_i[k']$. Each element will be updated if it exists an $x > 0$ for which the equation 2 is valid, where $\mathcal{AT}_i[a] = \mathcal{AT}_i[a] + x$, ($a$ is the index of the array $\mathcal{AT}_i$). As a results, the set of the selected features $Selected\_AT\_set$ will be determined.

4.2. Feature construction

To generate new features from the non selected ones, an undirected graph $G = (V, E)$ will be constructed where V is the set of nodes and E is the set of edges. Each node $v \in V$ is a feature which does not belong to $Selected\_AT\_set$. Formally: $V = \mathcal{AT} - Selected\_AT\_set$. With each edge $e = (v_a, v_b) \in E$, we associate a real number w_e that represent its weight. This weight is computed using the Pearson similarity between the nodes (features) v_a and v_b. All the edges having weights less than t_{cut} will be pruned from G. Using this pruned graph, the minimum cover vertex set $\mathcal{VC}$ is then computed. Based on $\mathcal{VC}$, subsets of nodes are constructed from not selected features. Each subset is composed of a node from $\mathcal{VC}$ and its neighbor nodes within the graph G. Finally, each set of not selected features will be merged to generate a new feature by using a mean vector.

Example 3. *Given a feature set $\mathcal{AT}$ composed of* 20 *features. Figure 4 illustrates two binary sequences representing two chromosomes (a) and (b). Decoding (a) gives the triplet $L = 5$, $id = 5997$ and $t_{cut} = 0.2$. Similarly, decoding*

Algorithm 1: | Generation of subsets of features

Input: $\mathcal{AT}$, $ID_{\mathcal{AT}_i}$ and k. in
Output: An array of integers of length $k > 0$. out

$Selected\_AT\_set[k] \leftarrow \{1 \dots 1\}$;
$ID_{\mathcal{AT}_0} = 0$;
for $a = 1$ to k **do**
  $cID \leftarrow ID_{\mathcal{AT}_0}$;
  $x \leftarrow 0$;
  while $(cID <= ID_{\mathcal{AT}_i})$ **do**
    $ID_{\mathcal{AT}_0} \leftarrow cID$;
    $e \leftarrow \binom{\mathcal{AT}-(Selected\_AT\_set[a-1]+x)}{k-a}$;
    $cID \leftarrow cID + e$;
    if $(cID <= ID_{\mathcal{AT}_i})$ **then**
      $x \leftarrow x + 1$;
    end if
  end while
  $Selected\_AT\_set[a-1] \leftarrow Selected\_AT\_set[a-1] + x$;
  if $(a - 1 < k - 1)$ **then**
    $Selected\_AT\_set[a] \leftarrow Selected\_AT\_set[a-1] + 1$;
  end if
end for

return $Selected\_AT\_set$;

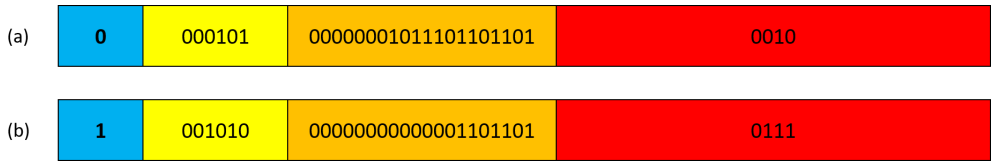

Figure 4: Example of two chromosomes.

the chromosome (b) gives the triplet $L = 10$, $id = 109$ *and* $t_{cut} = 0.7$. *Based on the Algorithm 1, the features sets selected from (a) and (b) are* [2, 7, 8, 17, 20] *and* [1, 2, 3, 4, 5, 6, 7, 9, 15, 19], *respectively. Thus, the not selected features sets are* [1, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19] *and* [8, 10, 11, 12, 13, 14, 16, 17, 18, 20], *respectively.*

Example 4. *Let us consider a set of* 5 *instances (see Figure 5). Each instance consists of four features* x_1, x_2, x_3, x_4 *and one feature* y *representing its label (i.e., class). In addition, suppose that the following subsets of features are computed to generate two new features:*

1. $\{x_1, x_2\}$
2. $\{x_1, x_3, x_4\}$

Thus, the mean vector a_1 *generated from* $\{x_1, x_2\}$ *is equal to* $(1 + 5)/2$, $(2 + 2)/2$, $(3 + 3)/2$, $\dots$, $(5 + 1)/2$. *Similarly, the mean vector* a_2 *is computed from* $\{x_1, x_3, x_4\}$ *and is equal to* $\frac{1+4+3}{3}$, $\frac{2+3+5}{3}$, $\frac{3+2+1}{3}$, $\dots$, $\frac{5+2+1}{3}$ *(see Figures 5 (b) and (c)).*

4.3. Evaluation of chromosomes

To evaluate the candidate solutions, an evaluation (fitness) function is defined. At each iteration, this function will be tested over all chromosomes. The chromosomes with the best fitness scores will be considered for the next iteration.

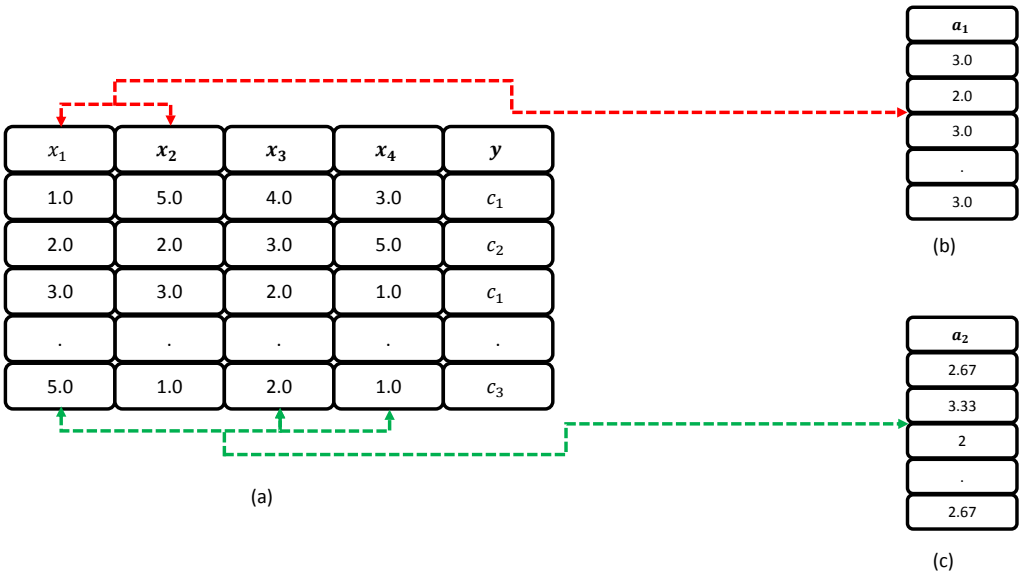

Figure 5: An illustrative example of the creation of two new features.

More precisely, each chromosome ch encodes a candidate solution $\mathcal{S}$ (i.e., the selected and constructed features). It will be evaluated in terms of its ability to generate an accurate decision tree $\mathcal{DT}$.

The fitness function is defined as the average score of the 10-fold cross-validation accuracy ($AvgACC$) of the decision tree (constructed based on $\mathcal{S}$) over only the transformed training samples $\mathcal{TR}'$. Formally, the fitness of a chromosome is defined as:

$$fitness(ch) = AvgACC_{10-CV}(\mathcal{DT}(\mathcal{TR}')) \tag{3}$$

4.4. Genetic operators

This section will describe the crossover and mutation operators used in our approach.

Crossover. This operator is applied on two chromosomes (parents) randomly picked from the population. This leads to the generation of a new chromosome. This latter is formed by the displacement of a portion of the considered (parents) chromosomes. For the next generation, two offsprings will be generated. Among the different types of crossover, single point crossover will be applied in our approach.

Mutation. This operation is implemented on an individual by altering one or more randomly chosen genes. It is one of the operators that facilitate the introduction of diversity in the population. The altered genes are randomly selected from the parent chromosomes to create a new offspring. The mutation percentage defines the likelihood of substituting one bit with another (changing 0 to 1 or 1 to 0) in a random manner, without any interaction with other chromosomes.

4.5. The convergence criteria

The genetic algorithm will converge when the fittest individual of the newly generated population is an optimal or a near-optimal solution. To avoid the cases of the high computation time, the genetic algorithm can be terminated after a predefined maximum number of generations. In our case, This number is set to 2000.

5. Experimental setup

Software. All experiments are carried out on a 2.9 GHz Intel Core i5 dual-core PC with 12 GB of RAM. Our approach, termed FSC_GA_DT, is implemented in Java using the frameworks Weka (Witten, Frank, Hall, Pal and DATA, 2005) and Mosek (ApS, 2019).

Table 1
Experimental datasets.

Datasets	$\#Classes$	$\#Features$	$\#Instances$	$Majority\_Class\_Probability$ (in %)
Artificial-characters (AC)	10	7	10218	11.72
Wisconsin-Breast-Cancer (WBC)	2	9	699	65.52
Column2C (C2)	2	6	310	67.74
Column3C (C3)	3	6	310	48.38
CNAE-9 (CN)	9	856	1080	11.11
Dermatology (DE)	6	34	366	30.60
Diabetes (DI)	2	8	768	65.10
Heart (HE)	2	13	270	55.55
Hepatitis (HEP)	2	19	155	79.35
Ionosphere (IO)	2	34	351	35.89
Letter (LE)	26	16	20000	04.06
Liver Disorders (LI)	2	6	345	57.97
Mammographic Mass (MA)	2	5	961	53.69
Optdigits (OP)	10	64	5620	10.17
Parkinsons (PA)	2	22	195	75.38
Pendigits (PE)	10	16	10992	10.40
QSAR biodegradation (QS)	2	41	1055	66.25
Segment (SE)	7	19	2310	14.28
Sonar (SO)	2	61	208	46.63
Spambase (SP)	2	57	4601	60.59
Spectf heart (SPE)	2	44	349	72.77
Vehicle (VE)	4	18	846	74.23
Waveform-5000 (WA)	3	40	5000	66.16

Datasets. We experiment on 23 real-world benchmark datasets from UCI (Dua and Graff, 2017). Their statistics are summarized in Table 1 where $\#Classes$ represents the number of classes, $\#Features$ represents the number of features, and $\#Instances$ is the number of samples (instances) in each dataset. Moreover, for each dataset, we show the $Majority\_Class\_Probability$ which refers to the obtained accuracy (in %) if the most frequent class is always predicted. The sizes of the used datasets range from 195 up to 20000 instances, and the number of attributes ranges from 4 to 856. The used dataset also encompass both binary and multiclass classifications, and 9 of them are of balanced data. This variation of characteristics ensures a deliberate and unbiased selection of databases for our analysis.

Parameter settings. To evaluate the performance of our approach, we perform a 10-fold cross validation classification on the benchmark datasets. Changes are made between randomly selected pairs of chromosomes (*i.e.*, crossover) using a probability value of 0.8, a mutation probability value of 0.05, 20 chromosomes, and 2000 iterations.

Baseline models. We compare FSC_GA_DT with:

- C4.5 algorithm (i.e., J48), implemented in Weka, with and without pruning;
- EFC (Vouk et al., 2023) with and without the feature selection. We use the authors implementation with the default parameters.

6. Results and analysis

Our approach FSC_GA_DT will be evaluated in two scenarios. The first one involves comparing our approach to C4.5 and EFC in terms of accuracy, precision, recall, F-measure, tree size, and number of generated attributes. Note that the EFC method does not provide precision, recall, F-measure, or a confusion matrix, making it impossible to compute these metrics in this specific case. The second scenario is a real-world application designed to evaluate the effectiveness of our approach in solving a practical problem (presented in the next Section).

Table 2: Precision, recall and F-measure (in %) of our approach compared to C4.5.

Datasets	Metrics	C4.5 with pruning	C4.5 without pruning	FSC_GA_DT
AC	**PR(%)**	74.00	**74.29**	74.17
	RE(%)	73.65	73.98	**74.61**
	F1(%)	73.61	73.93	**74.16**
WBC	**PR(%)**	94.69	93.78	**96.45**
	RE(%)	94.56	93.70	**96.61**
	F1(%)	94.57	93.68	**96.42**
C2	**PR(%)**	80.92	80.92	**84.21**
	RE(%)	80.64	80.64	**82.90**
	F1(%)	79.75	79.75	**82.63**
C3	**PR(%)**	***81.39***	81.09	79.92
	RE(%)	**80.96**	80.64	79.35
	F1(%)	**80.84**	80.51	78.68
CN	**PR(%)**	**90.76**	88.63	89.75
	RE(%)	**88.79**	87.87	88.60
	F1(%)	**89.00**	87.79	88.60
DE	**PR(%)**	93.72	92.61	**97.95**
	RE(%)	93.71	92.62	**97.56**
	F1(%)	93.70	92.61	**97.51**
DI	**PR(%)**	74.19	**74.20**	72.58
	RE(%)	74.34	**74.47**	72.24
	F1(%)	73.64	**73.95**	70.57
HE	**PR(%)**	79.29	75.67	**81.86**
	RE(%)	78.51	74.81	**80.76**
	F1(%)	78.25	74.64	**80.44**
HEP	**PR(%)**	75.90	76.48	**82.92**
	RE(%)	76.79	77.41	**83.17**
	F1(%)	75.82	76.18	**81.71**
IO	**PR(%)**	89.69	89.97	**92.56**
	RE(%)	89.45	89.73	**92.03**
	F1(%)	89.25	89.56	**91.98**
LE	**PR(%)**	87.99	88.07	**88.18**
	RE(%)	87.79	87.85	**88.01**
	F1(%)	87.81	87.87	**88.02**
LI	**PR(%)**	67.35	67.12	**67.79**
	RE(%)	67.21	66.94	**69.86**
	F1(%)	66.59	66.029	**68.98**
MA	**PR(%)**	81.68	80.78	**81.97**
	RE(%)	81.68	80.54	**82.24**
	F1(%)	81.64	80.45	**82.01**
OP	**PR(%)**	90.67	90.50	**91.06**
	RE(%)	90.42	90.26	**91.27**
	F1(%)	90.43	90.26	**91.05**
PA	**PR(%)**	81.47	88.53	**89.52**
	RE(%)	80.50	87.26	**89.33**
	F1(%)	79.92	86.92	**89.24**
PE	**PR(%)**	**96.60**	96.56	96.48
	RE(%)	**96.57**	96.53	96.43
	F1(%)	**96.57**	96.53	96.42

	PR(%)	83.34	81.94	**85.07**
QS	**RE**(%)	83.50	82.08	**85.12**
	F1(%)	83.39	82.00	**85.01**
	PR(%)	96.39	96.37	**96.53**
SE	**RE**(%)	96.32	96.32	**96.42**
	F1(%)	96.31	96.31	**96.41**
	PR(%)	68.31	68.31	**74.53**
SO	**RE**(%)	67.78	67.78	**73.08**
	F1(%)	67.54	67.54	**72.40**
	PR(%)	92.72	92.64	**92.80**
SP	**RE**(%)	92.71	92.61	**92.79**
	F1(%)	92.70	92.60	**92.78**
SPE	**PR**(%)	86.57	86.96	**88.16**
	RE(%)	84.50	84.78	**87.68**
	F1(%)	84.85	85.22	**87.69**
VE	**PR**(%)	70.62	71.36	**72.53**
	RE(%)	70.33	70.57	**72.22**
	F1(%)	69.60	70.07	**71.58**
WA	**PR**(%)	74.38	74.10	**81.87**
	RE(%)	74.30	74.02	**81.73**
	F1(%)	74.29	74.01	**81.72**

Table 2 illustrates the obtained results in terms of precision, recall, and F-measure between FSC_GA_DT and C4.5. The results clearly show the superiority of FSC_GA_DT where it outperforms C4.5 (with and without pruning) in 19 out of the 23 benchmark datasets.

Table 3 shows the obtained accuracy results for our approach compared to EFC and C4.5. The results show that FSC_GA_DT demonstrates impressive performance, often outperforming the competing methods C4.5 (with and without pruning) and EFC (with and without feature selection). More specifically, regarding the number of classes, we have 10 datasets ranging from 3 to 26 classes. FSC_GA_DT proves to be more effective than the competitive approaches in terms of precision with a score of 6 out of 10. Similarly, for binary classification datasets, it achieves a higher score in 8 out of the 13 datasets. Based on the number of features in the datasets, the results show that FSC_GA_DT is significantly better than the competitive approaches in 10 out of the 16 datasets having more than 10 features and in 4 out of the 7 datasets with fewer than 10 features. Regarding the number of instances, FSC_GA_DT dominates all the other approaches in 7 out of 9 of the datasets with a number of instances exceeding 1000. It also scores best in 7 out of the 14 datasets with less than 1000 instances. It is worth noting that the benchmark datasets contain 10 balanced and 13 unbalanced. Our approach scored best in 7 out of the 10 balanced and 7 from the 13 unbalanced datasets. Notably, even on the datasets where FSC_GA_DT did not outperform the best competing approach, the difference between its accuracy and the best score is very small.

Table 4 shows the average size of decision trees constructed in 10-cv. A cross examination of the results in both Table 3 and 4 shows that FSC_GA_DT has a better ability to effectively balance model complexity and performance, than both C4.5 and EFC. This allows our approach to generate simpler yet more efficient decision tree models. In the same context, we also show the number of constructed features for FSC_GA_DT and EFC (with and without feature selection) in Table 5 as well as the frequency of pruning per dataset (only for FSC_GA_DT). A cross examination of the results in the Tables 3, 4, 5 and the Figure 6 demonstrates a notable efficiency of FSC_GA_DT in feature reduction across various dataset. For instance, for the dataset WA, FSC_GA_DT was able to make a notable increase in accuracy while having a much smaller decision tree model than the competing methods and with only 25.20 averagely constructed features. The same observation could be noticed with multiple other datasets such as WBC and MA, which endorses the effective strategy of our approach in achieving model simplicity without sacrificing accuracy. Finally, we also notice that, given the nature of decision trees, particularly the pruning phase, we can observe that the two algorithms, FSC_GA_DT and EFC, can construct unnecessary attributes that will be pruned later, such as VE for EFC and CN for FSC_GA_DT.

Table 3
Classification accuracy (in %) of our approach compared to C4.5 and EFC.

Datasets	C4.5 with pruning	C4.5 without pruning	EFC without feature selection	EFC with feature selection	FSC_GA_DT
AC	73.65	73.99	-	-	**74.17**
WBC	94.56	93.70	94.42	95.42	**96.42**
C2	80.65	80.65	81.94	**84.19**	82.90
C3	**80.97**	80.65	80.32	80.00	79.35
CN	88.80	87.87	**88.98**	**88.98**	88.61
DE	93.72	92.62	94.80	95.08	**97.56**
DI	74.34	**74.47**	73.70	73.96	72.27
HE	78.52	74.81	78.15	79.63	**80.74**
HEP	76.79	77.42	81.21	79.88	**83.17**
IO	89.45	89.74	92.31	**92.88**	92.04
LE	87.79	87.85	-	-	**88.00**
LI	67.22	66.95	66.12	66.12	**68.98**
MA	81.69	80.54	81.06	81.27	**82.00**
OP	90.43	90.27	87.51	90.50	**91.05**
PA	80.50	87.26	**89.79**	89.29	89.34
PE	**96.58**	96.53	-	-	96.43
QS	83.51	82.09	83.23	83.13	**85.12**
SE	96.32	96.32	95.24	95.93	**96.41**
SO	67.79	67.79	76.45	**76.90**	73.07
SP	92.72	92.61	92.61	92.39	**92.78**
SPE	84.50	84.79	86.52	85.67	**87.67**
VE	70.33	70.57	72.71	**73.41**	72.23
WA	74.30	74.02	76.96	75.40	**81.74**

7. Application on the detection of Botnet traffic in Internet of Vehicles

This section addresses the critical issue of vehicular bot malware within the realm of Internet of Vehicles (IoV) technology. Our emphasis is on the pressing need to enhance IoV security by developing an intrusion detection system capable of identifying and mitigating the impact of vehicular bot malware on driver privacy and safety. To this end, we propose a flow-based detection approach utilizing FSC_GA_DT to identify bots within the vehicle network traffic. Furthermore, we provide an overview of the experiments conducted to evaluate the effectiveness of this approach.

7.1. Botnet in Internet of Vehicles

The IoV is a technology that enables information exchange between vehicles and related entities, aiming to reduce accidents, ease traffic congestion, and provide other information services (Wen, Zhang, Zhang, Cui, Cai and Chen, 2024). With its heterogeneous network architecture and compatibility with various communication devices, IoV enables the sharing of big data and reliable communication services, expanding the application range of automotive communication. However, with the increase in security threats, enhancing the IoV security is necessary to develop trust among vehicles. One of the most dangerous cybersecurity threats is vehicular bot malware, which involves the remote exploitation of a connected vehicle's onboard computer by an attacker. This cyber threat is particularly concerning as it can be used to execute a variety of malicious tasks remotely, including distributed network attacks (DDoS), violating driver's privacy (e.g., tracking their location, eavesdropping on conversations), controlling the vehicle remotely (e.g., opening doors, starting the engine, disabling brakes), and misleading the driver with false information about the vehicle state.

Despite the serious impact of vehicular bot malware on driver privacy and safety, research on this topic is limited. Only a few studies have investigated this issue. A recent work (Rahal et al., 2022) proposed a multilevel behavior-based framework called AntibotV to detect vehicular botnets, which can detect DDoS, and in-vehicle attacks. The framework monitors vehicle activity at the network and in-vehicle levels by training a decision tree with a set of features extracted from network traffic data of legitimate and malicious applications and in-vehicle data. (Garip, Lin, Reiher and Gerla, 2019) proposed SHIELDNET, a machine learning-based botnets detection mechanism. SHIELDNET

Table 4
Average size of the constructed decision tree (10-cv).

Datasets	C4.5 with pruning	C4.5 without pruning	EFC without feature selection	EFC with feature selection	FSC_GA_DT
AC	2268	2444.4	-	-	2366.4
WBC	20.8	40.2	25.00	19.90	12.6
C2	19.2	19.4	25.2	21.2	17.2
C3	22	23.6	24.6	23.4	25.8
CN	115.6	161.4	115.80	116.20	130.4
DE	37	45	24.20	31.00	13.6
DI	42.6	52.6	102.00	64.60	29.6
HE	37.2	64	27.30	28.90	34
HEP	16.6	28.8	16.80	19.60	24.4
IO	25.4	26	15.00	14.80	24.4
LE	2344.2	2562.2	-	-	2467.4
LI	49.4	59.2	45.2	23.5	43.6
MA	21	57.8	21.90	22.60	16.6
OP	401	436.4	435.80	430.50	384.8
PA	18.8	20.2	12.00	12.20	12
PE	375.2	410.4	-	-	390.4
QS	129	179	91.7	92.1	119.4
SE	80.2	87.4	91.60	85.40	81.4
SO	28.4	28.4	23.80	24.20	28.8
SP	200.6	339.6	207.50	211.20	202
SPE	41.4	43.4	32.80	33.60	43
VE	128.2	143.2	173.50	162.10	132.4
WA	594.2	609.8	706.10	732.50	330.4

specifically detects the use of GHOST (Garip, Reiher and Gerla, 2016), a botnet communication protocol that disguises its communication over the control channel using Basic Safety Messages (BSMs). The researchers identify vehicular botnet communication by detecting abnormal values of specific BSM fields. However, SHIELDNET's effectiveness relies on the use of GHOST protocol, so it may not be effective if the botmaster changes the communication protocol.

This paper aims to analyze and model vehicle network traffic to effectively detect bots. Among the various available intrusion detection techniques, decision trees have proven to be highly effective due to their ability to create rules that classify network traffic and identify potential intrusions based on traffic features and patterns. Furthermore, the high level of interpretability offered by decision trees is a significant advantage, allowing security analysts to comprehend the decision-making process and rules employed by the model. This facilitates the identification of false positives and false negatives and assists analysts in developing more efficient countermeasures while gaining a deeper understanding of the threat. Therefore, in this paper, we propose a novel flow-based detection approach that utilizes the FSC_GA_DT decision tree algorithm.

In the following, we will provide an overview of the experiment that was conducted. Firstly, we will discuss the process of creating the dataset. Next, we will outline the steps involved in the network flow processing. Finally, we will present the results obtained from the experiment.

7.2. Experimental environment

To test the proposed approach, we used the experimental environment built in (Rahal et al., 2022). To simulate realistic benign vehicular network traffic, we implemented 17 applications based on safety, convenience, and commercial criteria. It is recommended to refer to (Rahal et al., 2022) for learning more about the simulated applications and their descriptions. To ensure diversity and realistic traffic representation, we considered various factors such as physical-layer channel, transfer protocols, message Time-To-Live (TTL), routing protocols, trigger conditions, and communication technologies.

We simulated four bot attacks to generate malicious network traffic related to bot malware activity. The first attack is the Wave Short Message Protocol (WSMP) Flood, where the hacker sends WSM packets with unknown Packet

Table 5
Average numbers of constructed features (10-cv).

Datasets	EFC without feature selection	EFC with feature selection	FSC_GA_DT
AC	-	-	4.22
WBC	22.4	16.8	1.50
C2	17.4	12.7	3.33
C3	16.3	14.2	2.00
CN	59.7	58.7	232.30
DE	11.8	12.00	18.90
DI	71.3	40.8	3.60
HE	24.4	22.8	4.10
HEP	11.3	12.00	10.75
IO	13.9	12.4	7.40
LE	-	-	9.50
LI	23.6	23.6	5.70
MA	17.00	13.5	1.90
OP	397.50	249.20	4.20
PA	11.00	10.1	2.00
PE	-	-	7.30
QS	76.90	64.40	10.10
SE	70.6	57.00	6.30
SO	20.3	19.5	14.30
SP	160.8	134.1	18.50
SPE	26.6	25.3	4.56
VE	137.7	119.00	3.44
WA	718.8	521.3	25.20

Service Identifier (PSID) field values to the targeted vehicle, causing it to exhaust its resources by attempting to check the PSID of numerous WSM packets simultaneously. The second attack, the Geographic WSMP Flood, is a variant of the WSMP Flood attack that operates on a larger scale. In this attack, the hacker broadcasts forged WSMP messages within a specific geographic area, affecting all neighboring vehicles in that region, causing a significant impact on multiple vehicles' bandwidth and resources. The third attack is GPS tracking, where hackers can exploit vulnerabilities to track the real-time location of a vehicle, retrieve its complete trajectory, or check its location. This paper focuses on the real-time GPS tracking scenario, where the bot vehicle sends its latitude, longitude, speed, time, and direction coordination in a continuous streaming mode to the botmaster. The fourth attack is Eavesdropping, where hackers can use virtual assistant tools, such as Siri, to embed malicious commands in innocuous-seeming speech, recorded music, or low-powered lasers. Once activated, Siri records conversations using the vehicle's microphones and sends them to the hacker's remote server at regular intervals. We recommend referring to (Rahal et al., 2022) for detailed descriptions of the four bot attacks.

We simulated various scenarios for both benign and malicious traffic based on nodes ID. We used the Network Simulator version 3 (NS3) to conduct the simulation of vehicular traffic, while the Simulation of Urban MObility (SUMO) package provided realistic vehicular traffic simulation. The experiment used different types of nodes and generated traffic dataset as summarized in Table 6. Figure 7 shows the distribution of the network traffic dataset, where green and red colors represent benign and malicious traffic, respectively.

After collecting network traffic data generated during the simulation and storing them as Packet Capture (PCAP) files, we extracted flows and calculated features from the raw traffic. To achieve this, we utilized several scripts developed using the popular traffic exporter CICFlowMeter (Lashkari, Draper-Gil, Mamun, Ghorbani et al., 2017). We present the resulting list of features, along with their descriptions, in Table 7. The features under consideration can be categorized into three main categories: time-based, bytes-based, and packets-based.

7.3. Results and discussion

In the context of Network-based Intrusion Detection (NIDS), the choice of performance metrics should focus on the model's ability to accurately detect and classify Bot activity while minimizing false positives. Commonly used

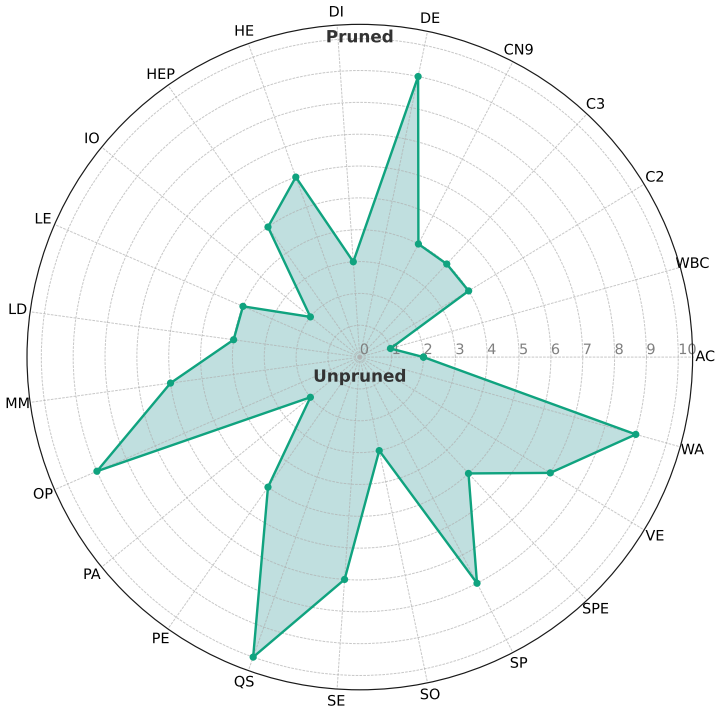

Figure 6: A radar chart depicting the adoption of a pruning phase within the 10 cross-validation (CV) runs.

Table 6
Simulation parameters.

Network Simulator	NS3
Traffic Generator	SUMO
Simulation Area	Manhattan Map
Simulation Time	500 seconds
Number of Nodes	40
Max Speed	20 m/s
MAC/PHY Standard	IEEE802.11p
Traffic Type	WSMP, IP
Bandwidth Channel	CCH, SCH
Propagation Model	Two-ray ground-reflection model
Transmission Power	20 dBm
Packets Size	Depend on the application and protocol
Packets Data Rate	Depend on the application

metrics include True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall, F1-measure, and Area Under the Receiver Operating Characteristic Curve (AUC). These metrics provide a comprehensive evaluation of the NIDS's performance in terms of its ability to detect malicious traffic while minimizing false alerts. Table 8 shows the results of a 10-fold cross-validation (cv) evaluation of FSC_GA_DT using the aforementioned metrics. The results suggest that our approach efficiently detects bot's network traffic with high TPR, low FPR, and high precision and F-measure. The AUC is also high, indicating that our approach is able to distinguish between legitimate and malicious network

Table 7
List of features.

Features	Description
Protocol	Protocol type
Duration	Duration of flow in micro second
total Fwd/bwd Packet	Total packets in the forward/backward direction
total Length of Fwd/Bwd Packet	Total size of packet in forward/backward direction
Fwd/Bwd Packet Length Min/Max/Mean/Std	Min/Max/Mean/Std size of packet in forward/backward direction
Flow Byte/s, Packets/s	Number of flow packets/Byte per second
Flow IAT Min/Max/Mean/Std	Min/Max/Mean/Std time between two packets sent in the flow
Fwd/Bwd IAT Min/Max/Mean/Std/Total	Min/Max/Mean/Std/Total time between two packets sent in the forward/backward direction
SYN Flag Count	Number of packets with SYN
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Fwd/Bwd Header Length	Total bytes used for headers in the forward/backward direction
Fwd/bwd Packets/s	Number of forward/backward packets per second
Min/Max/Mean/Std Packet Length	Min/Max/Mean/std length of a packet
down/Up Ratio	Download and upload ratio
Avg Fwd/Bwd Segment Size	Average size observed in the forward/backward direction
Fwd Header Length	Length of the forward packet header
Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
Avg Packet Size	Average size of packet
Fwd /Bwd AVG Bulk Rate	Average number of bulk rate in the forward/backward direction
Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
Bwd AVG Packet/Bulk	Average number of packets bulk rate in the backward direction
Init_Win_bytes_forward/Backward	The total number of bytes sent in initial window in the forward/backward direction
Subflow Fw/Bwd Packets	The average number of packets in a sub flow in the Forward/backward direction
Active Min/Mean/Max/Std	Min/Max/Mean/Std time a flow was active before becoming idle
Idle Min/Max/Mean/Std	Min/Max/Mean/Std time a flow was idle before becoming active
Subflow Fwd/bwd Packets	The average number of packets in a sub flow in the forward/backward direction
Act_data_pkt_forward	Count of packets with at least 1 byte of TCP data payload in the forward direction
Subflow Fwd/bwd Bytes	The average number of bytes in a sub flow in the forward/backward direction

Table 8
Predictive performances for 10-cv.

CV	TPR	FPR	Precision	F-Measure	AUC	ACC
CV_01	100.00%	0.00%	100.00%	100.00%	100.00%	100.00%
CV_02	99.60%	0.40%	99.60%	99.60%	99.90%	99.58%
CV_03	100.00%	0.00%	100.00%	100.00%	100.00%	100.00%
CV_04	98.40%	0.20%	98.50%	98.40%	99.50%	98.35%
CV_05	99.60%	0.40%	99.60%	99.60%	99.60%	99.58%
CV_06	100.00%	0.00%	100.00%	100.00%	100.00%	100.00%
CV_07	99.20%	0.40%	99.20%	99.10%	99.60%	99.17%
CV_08	99.60%	0.00%	99.60%	99.60%	99.80%	99.58%
CV_09	99.20%	0.10%	99.30%	99.20%	99.60%	99.17%
CV_10	100.00%	0.00%	100.00%	100.00%	100.00%	100.00%
Average	99.56%	0.15%	99.58%	99.55%	99.80%	99.55%

flows effectively. The high accuracy further supports the performance of the approach.

We conducted an extensive investigation of the model's performance, specifically focusing on its ability to detect each type of bot traffic. The detailed results are presented in Table 9. Overall, the obtained results indicate exceptional performance across all classes and metrics. The TPR is impressively high for all classes, indicating that the approach is capable of correctly identifying bot traffic. The FPR is also low, indicating that the model does not classify too many legitimate network flows as bot traffic. Compared to other traffics, detecting geographic WSMP flood traffic is very challenging since this class has a very low number of samples (see Figure 7) which makes the dataset imbalanced. Despite this difficulty, our approach is able to capture geographic WSMP flood traffic with a very good TPR at over 96%.

By analyzing the relevance of features used by FSC_GA_DT, we can also gain insights on how these features impact the accuracy of the model. The algorithm's use of features varies across different cvs. In 90% of the cvs,

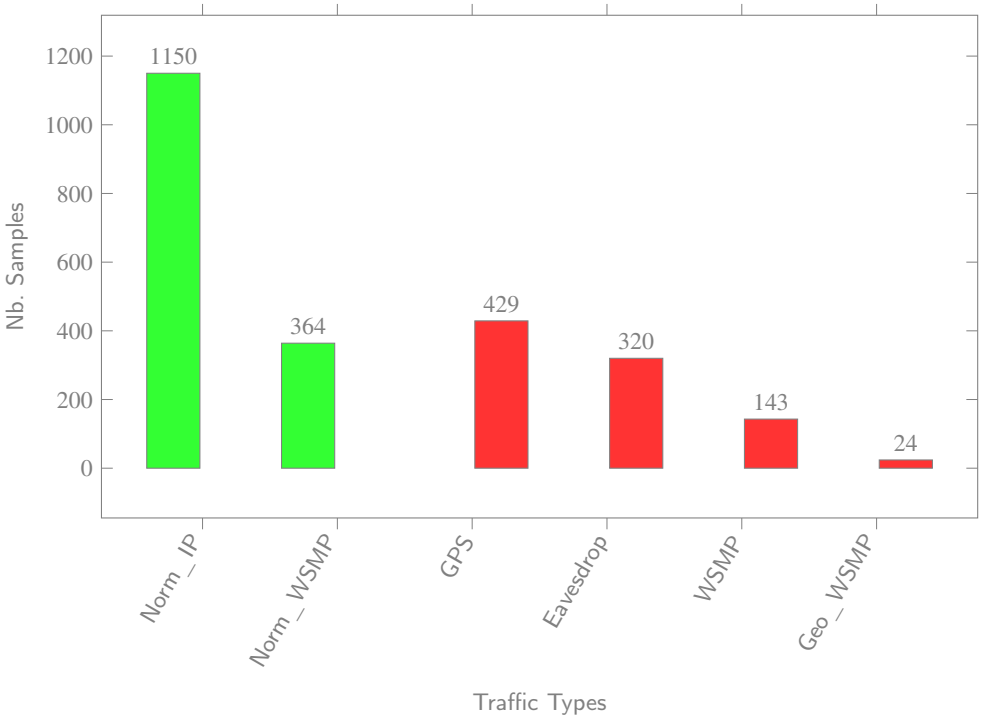

Figure 7: Number of samples per traffic type.

Table 9
Detection performances per traffic type.

	Traffic Type	TPR	FPR	Precision	F-Measure	AUC
Benign	WSMP Traffic	98.07%	0.10%	99.44%	98.74%	99.36%
	IP Traffic	99.74%	0.24%	99.73%	99.75%	99.78%
Malicious	GPS Tracking	100.00%	0.05%	99.77%	99.89%	99.98%
	Eavesdropping	100.00%	0.00%	100.00%	100.00%	100.00%
	WSMP Flood	00.00%	0.22%	96.83%	98.32%	99.94%
	Geo WSMP Flood	96.67%	0.00%	100.00%	98.00%	99.81%

FSC_GA_DT creates more new features than it retains (from the initial features set). Figure 8 illustrates the percentage decrease in the number of features for each cv. Remarkably, in one cv, FSC_GA_DT achieved 100% accuracy while reducing the number of features by 72.22%. In this case, FSC_GA_DT preserved two original features and introduced three new ones. On average, FSC_GA_DT reduces the number of features by 52.78%. Figure 9 displays the frequency of use of the features used in this experiment. This suggests that the FSC_GA_DT classifier can achieve impressive predictive performance by relying on time-based and packet-based features without the need for byte-based features. The high frequency of use of features related to inter-arrival time between packets, both in the forward direction and for the overall flow, highlights their importance in the classification process. The three most frequently selected features across the 10-cv are “Flow Duration”, “Flow Pkts/s” and “Fwd IAT Mean”.

The selection of time-based and packet-based features as the most pertinent for the classifier can be explained by their effectiveness in distinguishing different types of cyberattacks. For volume-based attacks, such as Denial of Service (DoS), time-based features are crucial in revealing anomalies, exemplified by the rapid influx of packets. This is complemented by packet-based features, which accentuate the sheer number of packets, a hallmark of these attacks. On the other hand, targeted attacks, such as GPS tracking and eavesdropping, present more subtle indications. In these scenarios, time-based features are instrumental in detecting irregular traffic patterns. Meanwhile, packet-based

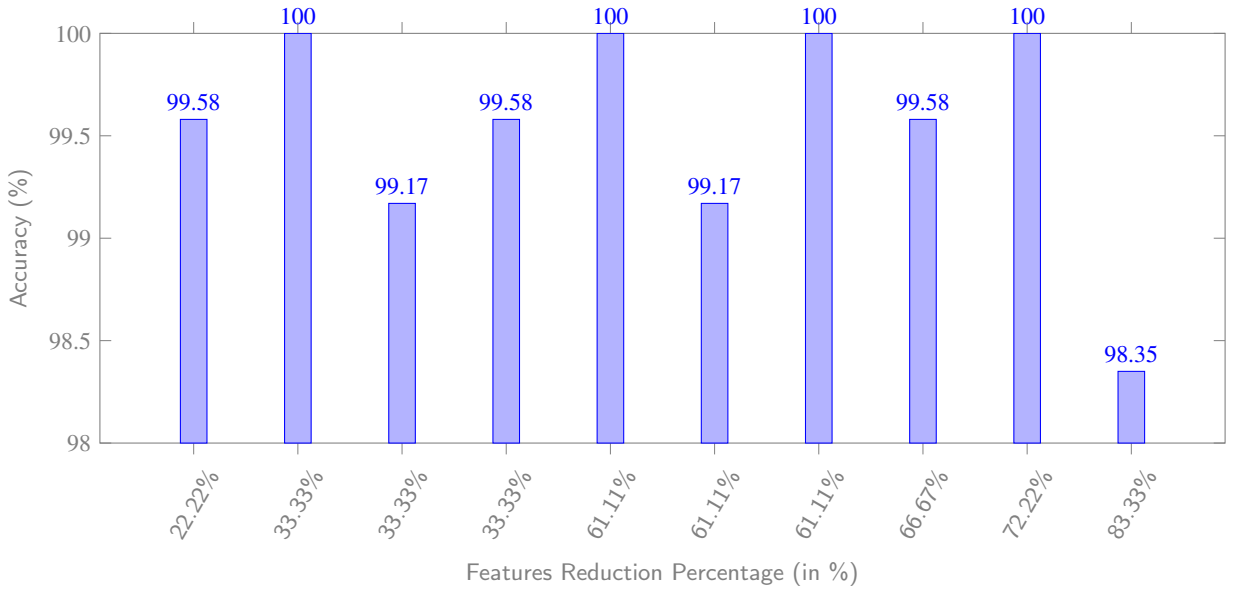

Figure 8: Accuracy and features reduction percentage of our approach on each of 10-cv.

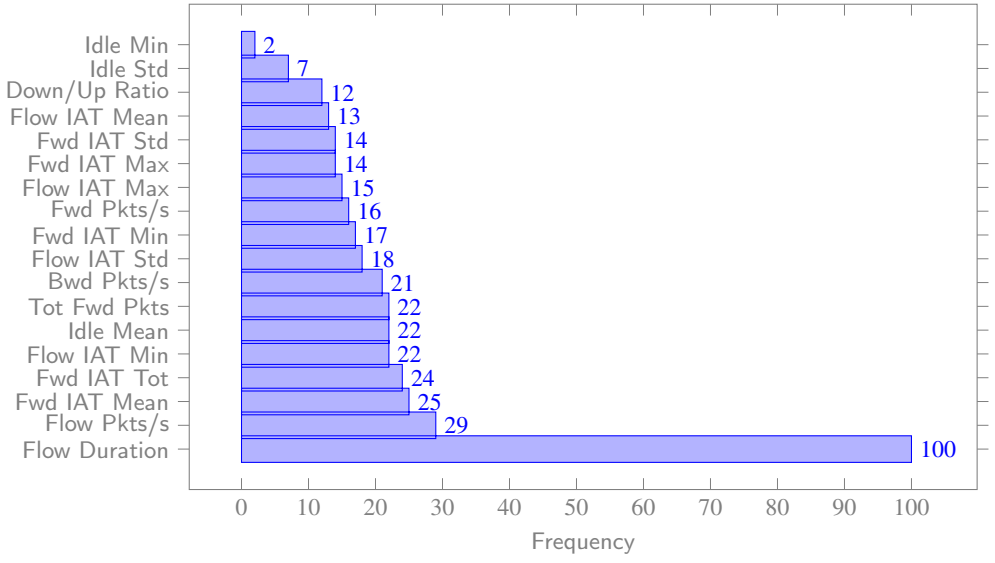

Figure 9: Feature usage frequency across the 10-cv.

features shift the focus to the specific nature and alterations of packets, rather than sheer volume. This sophisticated understanding and application of features enable a tailored and effective approach in differentiating and responding to both the widespread, volume-based attacks and the more nuanced, targeted cyber threats.

To assess the effectiveness of our approach, we compare it with Antibot (Rahal et al., 2022). We chose Antibot (Rahal et al., 2022) for comparison due to its significant relevance and alignment with our research objectives. Notably, this study represents the most recent significant development in the current state-of-the-art. Like our work, it employs a decision tree-based methodology. Importantly, both our study and Antibot utilize the same dataset and adhere to a comparable validation schema. This ensures a robust and consistent foundation for our comparative analysis. We consider the True Positive Rate (TPR) and the False Positive Rate (FPR) because these are the two widely used

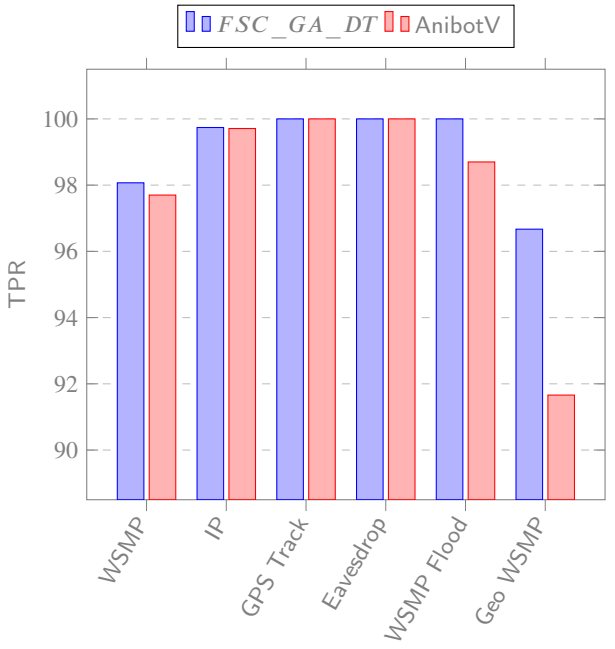

Figure 10: Comparison of TPR per traffic type.

metrics that provide the most insight into the performance of an IDS. A high TPR ensures effective detection of real cyber threats, while a low FPR avoids excessive false alarms, thereby maintaining the reliability and operational effectiveness of the IDS. Together, these metrics are crucial in assessing an IDS's ability to provide reliable security without unnecessarily burdening security teams.

Looking at the TPR results in the Figure 10, we can see that FSC_GA_DT has a higher TPR than AnibotV in all bot traffics. For WSMP Traffic, IP Traffic, GPS Tracking, and Eavesdropping, both classifiers have very high TPR, with FSC_GA_DT having a higher rate in all cases. However, for WSMP Flood and Geo WSMP Flood, there is a notable difference between the TPR of the two classifiers. FSC_GA_DT has a perfect TPR for WSMP flood and high TPR of 96.67% for Geo WSMP flood, while AnibotV has a lower rate of 98.70% and 91.66%, respectively. This shows that FSC_GA_DT is better suited for detecting of these types of bot attacks than AnibotV.

By referring to Figure 11, it is apparent that FSC_GA_DT exhibits a substantially lower FPR in almost all categories when compared to AnibotV. Specifically, our approach surpasses AntibotV in terms of legitimate traffic, as it shows a lower FPR. Furthermore, FSC_GA_DT demonstrates superior performance in mitigating GPS Track and Eavesdrop attacks, with lower FPR rates compared to AntibotV. However, it is worth mentioning that AnibotV exhibits a slightly lower FPR for WSMP Flood.

8. Conclusion

The popularity of decision trees can be attributed to their simplicity and their close resemblance to human reasoning. Nonetheless, the effectiveness of the resulting trees is significantly influenced by the specificities of the dataset under consideration. Usually, after the generation phase, decision trees undergo a pruning process to reduce their complexity and size. However, in numerous applications, this pruning step comes at the cost of reduced accuracy in the models.

Feature selection and construction have proven to be very efficient methods to emphasize regularities in the data and to boost the performances of predictive approaches especially in the presence of noise and uncertainties in the data.

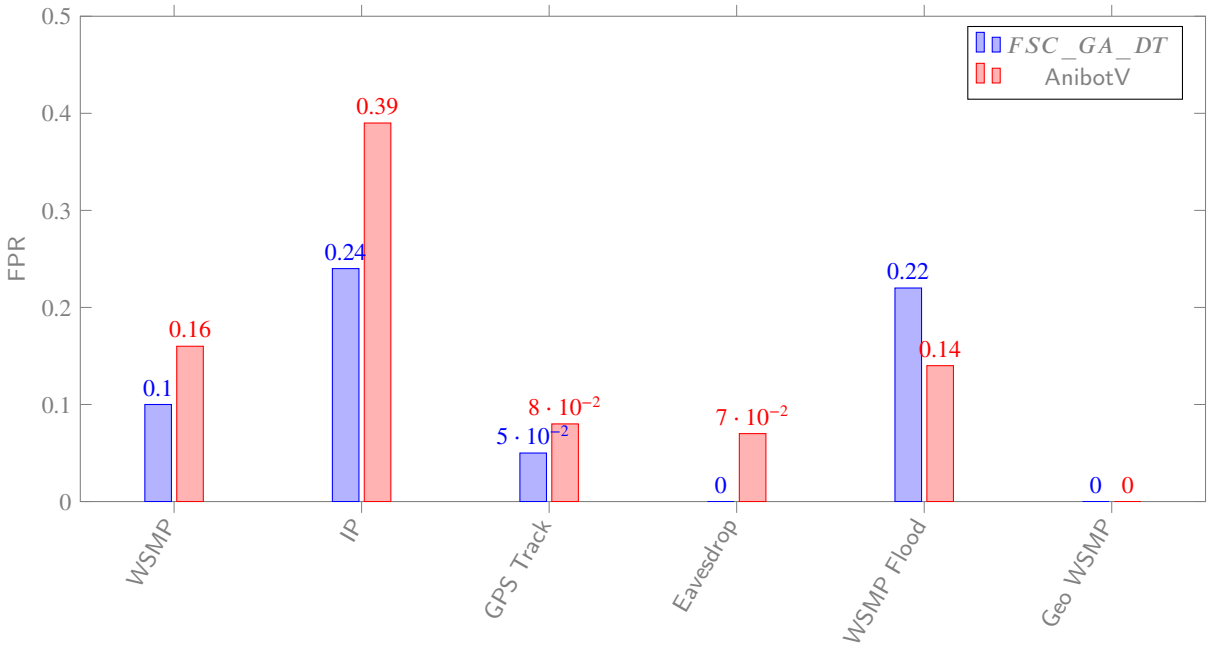

Figure 11: Comparison of FPR per traffic type

This paper introduces *FSC_GA_DT*, a graph-based approach that combines both feature selection and construction. It uses a genetic algorithm and a lattice-based representation to find the best features to select from the dataset as well as the best parameters to use for the feature construction on the non-selected features. This feature construction relies on a similarity graph-based representation between the non-selected features and covering set algorithm to find the best combinations of attributes.

Experiments on 20 datasets show the competitiveness of *FSC_GA_DT* and how it allows to obtain decision trees of higher performances than existing state-of-the-art approaches. Additionally, a real-world cybersecurity application was used to evaluate the proposed approach, specifically for detecting botnet traffic in IoV. The obtained results demonstrate the efficiency of our approach in accurately detecting botnet traffic in an IoV environment.

Acknowledgment

This work also benefitted from the PRFU project A14N01EP230220230001, funded in Algeria by The Directorate-General for Scientific Research and Technological Development (DGRSDT). Moreover, this research was supported by the 5G-INSIGHT bilateral project (ID: 14891397) / (ANR-20-CE25-0015-16), jointly funded by the Luxembourg National Research Fund (FNR) and the French National Research Agency (ANR). Additional funding came from the FEDER MISMAR, Région Nouvelle-Aquitaine B4IoT.

References

ApS, M., 2019. The MOSEK optimization toolbox for MATLAB manual. Version 9.0. URL: http://docs.mosek.com/9.0/toolbox/index.html.

Boryczka, U., Kozak, J., 2015. Enhancing the effectiveness of ant colony decision tree algorithms by co-learning. Applied Soft Computing 30, 166–178.

Chandrashekar, G., Sahin, F., 2014. A survey on feature selection methods. Computers Electrical Engineering 40, 16–28. URL: https://www.sciencedirect.com/science/article/pii/S0045790613003066, doi:https://doi.org/10.1016/j.compeleceng.2013.11.024. 40th-year commemorative issue.

Dua, D., Graff, C., 2017. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Fu, Z., Golden, B.L., Lele, S., Raghavan, S., Wasil, E., 2006. Diversification for better classification trees. Comput. Oper. Res. 33, 3185–3202. URL: http://dx.doi.org/10.1016/j.cor.2005.02.035, doi:10.1016/j.cor.2005.02.035.

Garip, M.T., Lin, J., Reiher, P., Gerla, M., 2019. Shieldnet: An adaptive detection mechanism against vehicular botnets in vanets, in: 2019 IEEE Vehicular Networking Conference (VNC), IEEE. pp. 1–7.

Garip, M.T., Reiher, P., Gerla, M., 2016. Ghost: Concealing vehicular botnet communication in the vanet control channel, in: 2016 International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE. pp. 1–6.

Hammami, M., Bechikh, S., Louati, A., Makhlouf, M., Said, L.B., 2020. Feature construction as a bi-level optimization problem. Neural Computing and Applications 32, 13783–13804.

Karabadji, N.E.I., Khelf, I., Seridi, H., Aridhi, S., Remond, D., Dhifli, W., 2019. A data sampling and attribute selection strategy for improving decision tree construction. Expert Systems with Applications 129, 84–96.

Karabadji, N.E.I., Korba, A.A., Assi, A., Seridi, H., Aridhi, S., Dhifli, W., 2023. Accuracy and diversity-aware multi-objective approach for random forest construction. Expert Systems with Applications , 120138.

Karabadji, N.E.I., Seridi, H., Bousetouane, F., Dhifli, W., Aridhi, S., 2017. An evolutionary scheme for decision tree construction. Knowledge-Based Systems 119, 166 – 177. URL: http://www.sciencedirect.com/science/article/pii/S0950705116305056, doi:https://doi.org/10.1016/j.knosys.2016.12.011.

Lashkari, A.H., Draper-Gil, G., Mamun, M.S.I., Ghorbani, A.A., et al., 2017. Characterization of tor traffic using time based features., in: ICISSp, pp. 253–262.

Liu, C., Lin, B., Lai, J., Miao, D., 2022. An improved decision tree algorithm based on variable precision neighborhood similarity. Information Sciences 615, 152–166.

Ma, J., Gao, X., 2020a. Designing genetic programming classifiers with feature selection and feature construction. Applied Soft Computing 97, 106826.

Ma, J., Gao, X., 2020b. A filter-based feature construction and feature selection approach for classification using genetic programming. Knowledge-Based Systems 196, 105806.

Otero, F.E., Freitas, A.A., Johnson, C.G., 2012. Inducing decision trees with an ant colony optimization algorithm. Applied Soft Computing 12, 3615 – 3626. URL: http://www.sciencedirect.com/science/article/pii/S1568494612002864, doi:https://doi.org/10.1016/j.asoc.2012.05.028.

Quinlan, J.R., 1987. Simplifying decision trees. International journal of man-machine studies 27, 221–234.

Rahal, R., Amara Korba, A., Ghoualmi-Zine, N., Challal, Y., Ghamri-Doudane, M.Y., 2022. Antibotv: A multilevel behaviour-based framework for botnets detection in vehicular networks. Journal of Network and Systems Management 30, 1–40.

SabbaghGol, H., Saadatfar, H., Khazaiepoor, M., 2024. Evolution of the random subset feature selection algorithm for classification problem. Knowledge-Based Systems 285, 111352.

Tan, J., Gui, N., Qiu, Z., 2024. Gaefs: Self-supervised graph auto-encoder enhanced feature selection. Knowledge-Based Systems , 111523.

Tran, B., Xue, B., Zhang, M., 2019. Genetic programming for multiple-feature construction on high-dimensional classification. Pattern Recognition 93, 404–417.

Tran, B.N., 2018. Evolutionary computation for feature manipulation in classification on high-dimensional data .

Vouk, B., Guid, M., Robnik-Šikonja, M., 2023. Feature construction using explanations of individual predictions. Engineering Applications of Artificial Intelligence 120, 105823.

Wen, J., Zhang, J., Zhang, Z., Cui, Z., Cai, X., Chen, J., 2024. Resource-aware multi-criteria vehicle participation for federated learning in internet of vehicles. Information Sciences , 120344.

Witten, I.H., Frank, E., Hall, M.A., Pal, C.J., DATA, M., 2005. Practical machine learning tools and techniques, in: Data Mining.

Xue, B., Zhang, M., Dai, Y., Browne, W.N., 2013. Pso for feature construction and binary classification, in: Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp. 137–144.