ADAPTIVE INTRUSION DETECTION SYSTEM BASED ON VARIABLE LENGTH MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION

By

AL DARRAJI MUSTAFA SABAH NOORI

Thesis Submitted to the School of Graduate Studies, University Putra Malaysia, in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

July 2024

FK 2024 11

DEDICATIONS

This thesis is dedicated to the most important people in my life. To my mother's loving soul, the epitome of kindness and love, and my beloved father, who both made countless sacrifices to raise me and help me reach this achievement. To my two sisters, for always being there to support and encourage me. To my wife, for her unwavering support and patience. I also extend my dedication to my extended family and my second home, Malaysia, as well as to my homeland, Iraq, the land of ancient civilizations.

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Doctor of Philosophy

ADAPTIVE INTRUSION DETECTION SYSTEM BASED ON VARIABLE LENGTH MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION

By

MUSTAFA SABAH NOORI

July 2024

Chairman : Associate Professor Ratna Kalos Zakiah binti Sahbudin, PhD
Faculty : Engineering

Intrusion Detection Systems (IDS) based on data stream encounter a unique phenomenon known as feature drift, where the significance of features evolves over time, potentially reducing the efficacy of IDSs or rendering them obsolete. Feature drift, along with other critical challenges such as high dimensionality, concept drift, and imbalanced datasets, necessitates the development of a framework that can adaptively and jointly handle these challenges to remain dynamically functional over time. Therefore, this thesis introduces an advanced IDS framework named Dynamic Feature Aware Genetic Programming Ensemble (DFA-GPE). DFA-GPE employs an incremental learning procedure to eliminate the need for storing incoming data or retraining the IDS, utilizes the Synthetic Minority Oversampling Technique (SMOTE) to address imbalanced datasets, and employs a Drift Detection Method (DDM) to detect and manage concept drift. A crucial component of the framework is the advanced Dynamic Feature Selection (DFS) method based on Multi-Objective Particle Swarm Optimization (MOPSO), utilized to handle feature drift and reduce the dimension simultaneously.

The development of MOPSO started with the introduction of an advanced algorithm, named Multi-Exemplar Particle Swarm Optimization with Local Awareness (MEPSOLA), designed to tackle conflicting optimization challenges. MEPSOLA integrates advanced strategies, including smart initialization, multi-exemplar, and conditional local search techniques, into the process of exemplar selection enhancement. MEPSOLA was evaluated using several standard mathematical benchmarks and the results compared against well-established benchmark Multi-Objective Optimization (MOO) algorithms in the literature.

Building on the foundational MEPSOLA, the algorithm was further refined and adapted for online DFS to operate in the DFA-GPE framework. Adjusted under the name Variable Length Multi-Objective Particle Swarm Optimization (VLMO-PSO), it dynamically manages feature drift. VLMO-PSO utilized a smart population initialization strategy and introduced a division-based search criterion that partitions the solution space into discrete segments or divisions. Additionally, it leverages a unique set of transfer functions that map mobility equation outcomes into the binary decision space and an innovative multi-objective exemplar selection method that balances exploration and exploitation. The final feature selection decisions within VLMO-PSO are performed by statistical analyses of feature weights, simultaneously enhancing accuracy and conserving memory. An evaluation of DFA-GPE on two benchmark datasets, namely HIKARI 2021 and TON_IoT 2020, compared its performance to different feature selection methods including Random Forest (RF), Logistic Regression (LR), Recursive Feature Elimination (RFE), and Non-dominated Sorting Genetic Algorithm (NSGA-II). Within the HIKARI dataset, DFA-GPE achieved the highest accuracy of 99.09%, surpassing NSGA-II of 98.79%, RF of

98.73%, LR of 98.62%, and RFE of 98.39%. Memory utilization showed a saving of 93.49%, reducing the number of features from 67 to just 5. Similarly, for the TON_IoT dataset, DFA-GPE led with an accuracy of 92.64%, outperforming NSGA-II of 92.18%, RF of 92.01%, RFE of 91.51%, and LR of 90.74%, reducing the feature count from 42 to 16 and achieving a memory saving of 62.93%.

Keyword: Data Stream classification, Dynamic Feature Selection, Feature Drift, Intrusion Detection Systems, Multi-Objective Particle Swarm Optimization.

SDG: GOAL 9: Industry, Innovation and Infrastructure

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

SISTEM PENGESANAN PENCEROBOHAN ADAPTIF BERASASKAN PENGOPTIMUMAN PANJANG PEMBOLEH UBAH PELBAGAI- OBJEKTIF PARTIKEL SWARM

Oleh

MUSTAFA SABAH NOORI

Julai 2024

Pengerusi : Profesor Madya Ratna Kalos Zakiah binti Sahbudin, PhD
Fakulti : Kejuruteraan

Sistem Pengesanan Pencerobohan (IDS) berasaskan aliran data menghadapi fenomena unik yang dikenali sebagai ciri hanyut, di mana kepentingan ciri berubah dari masa ke masa, yang berpotensi mengurangkan keberkesanan IDS sedia ada atau menjadikannya usang. Ciri hanyut, bersama-sama dengan cabaran kritikal lain seperti dimensi tinggi, konsep hanyut, dan set data tidak seimbang, memerlukan pembangunan rangka kerja yang boleh menyesuaikan diri dan menangani cabaran ini secara bersama-sama untuk kekal berfungsi secara dinamik dari masa ke masa. Oleh itu, tesis ini memperkenalkan rangka kerja IDS lanjutan yang dinamakan Dynamic Feature Aware Genetic Programming Ensemble (DFA-GPE). DFA-GPE menggunakan prosedur pembelajaran bertingkat untuk menghapuskan keperluan menyimpan data masuk atau melatih semula IDS, menggunakan Teknik Pensampelan Terlebih Minoriti Sintetik (SMOTE) untuk menangani set data tidak seimbang, dan menggunakan kaedah pengesanan hanyut (DDM) untuk mengesan dan menguruskan konsep hanyut. Komponen penting dalam rangka kerja ini ialah kaedah Pemilihan Ciri

Dinamik (DFS) lanjutan berdasarkan Pengoptimuman Pelbagai-objektif Partikel Swarm (MOPSO), yang digunakan untuk menangani ciri hanyut dan mengurangkan dimensi secara serentak.

Pembangunan MOPSO bermula dengan pengenalan algoritma lanjutan yang dinamakan Multi-Exemplar Particle Swarm Optimization with Local Awareness (MEPSOLA), yang direka untuk menangani cabaran pengoptimuman yang bercanggah. MEPSOLA mengintegrasikan strategi canggih, termasuk inisialisasi pintar, pelbagai-contoh, dan teknik pencarian lokal bersyarat, ke dalam proses penambahbaikan pemilihan contoh. MEPSOLA dinilai menggunakan beberapa penanda aras matematik standard dan hasilnya dibandingkan dengan algoritma Pengoptimuman Pelbagai Objektif (MOO) yang terkenal dalam literatur.

Berdasarkan asas MEPSOLA, algoritma ini telah ditingkatkan dan disesuaikan untuk DFS dalam talian untuk beroperasi dalam rangka kerja DFA-GPE. Diubah suai dengan nama Pengoptimuman Panjang Pemboleh Ubah Pelbagai- Objektif Partikel Swarm (VLMO-PSO), ia menguruskan ciri hanyut secara dinamik. VLMO-PSO menggunakan strategi inisialisasi populasi pintar dan memperkenalkan kriteria pencarian berasaskan pembahagian yang membahagikan ruang penyelesaian ke dalam segmen atau bahagian diskrit. Selain itu, ia memanfaatkan satu set fungsi pemindahan unik yang memetakan hasil persamaan mobiliti ke dalam ruang keputusan binari, dan kadah pemilihan contoh multi-objektif yang inovatif yang mengimbangi eksplorasi dan eksploitasi. Keputusan pemilihan ciri akhir dalam VLMO-PSO dilakukan melalui analisis statistik ciri berat, yang serentak meningkatkan ketepatan dan menjimatkan memori. Penilaian komprehensif DFA-GPE pada dua set data penanda aras, iaitu

HIKARI 2021 dan TON_IoT 2020, membandingkan prestasinya dengan kaedah pemilihan ciri yang berbeza termasuk Random Forest (RF), Logistic Regression (LR), Recursive Feature Elimination (RFE), dan Non-dominated Sorting Genetic Algorithm (NSGA-II). Dalam set data HIKARI, DFA-GPE mencapai ketepatan tertinggi sebanyak 99.09%, mengatasi NSGA-II sebanyak 98.79%, RF sebanyak 98.73%, LR sebanyak 98.62%, dan RFE sebanyak 98.39%. Penggunaan memori menunjukkan penjimatan sebanyak 93.49%, mengurangkan bilangan ciri daripada 67 kepada hanya 5. Begitu juga, untuk set data TON_IoT, DFA-GPE mendahului dengan ketepatan sebanyak 92.64%, mengatasi NSGA-II sebanyak 92.18%, RF sebanyak 92.01%, RFE sebanyak 91.51%, dan LR sebanyak 90.74%, mengurangkan bilangan ciri daripada 42 kepada 16 dan mencapai penjimatan memori sebanyak 62.93%.

Kata kunci: Ciri Hanyut, Pemilihan Ciri Dinamik, Pengelasan Aliran Data**,** Pengoptimuman Pelbagai-objektif Partikel Swarm, Sistem Pengesanan Pencerobohan.

SDG: MATLAMAT 9: Industri, Inovasi dan Infrastruktur

ACKNOWLEDGEMENTS

All praise to Allah, the Most Merciful, for His guidance and strength, and for blessing me with health and determination to complete this thesis. May His grace continue to guide us all.

I extend my heartfelt gratitude to my supervisor, Assoc. Prof. Dr. Ratna Kalos Zakiah Binti Sahbudin, for her invaluable guidance, encouragement, support, and care throughout this journey. Her expertise and insights have been instrumental in shaping this work, and her unwavering belief in my abilities has motivated me to strive for excellence. I am truly fortunate to have had her as a mentor.

I also wish to express my profound appreciation to my respectful supervisory committee, Professor Ir. Dr. Aduwati Binti Sali and Assoc. Prof. Dr. Fazirulhisyam Hashim, for their support, guidance, and invaluable help and direction, which have significantly contributed to my research. Their expertise and thoughtful advice have been crucial in navigating the complexities of my study. The dedication and insights of both my co-supervisors have greatly enriched my academic experience and research journey.

I extend my sincere gratitude to the staff at Universiti Putra Malaysia (UPM), especially those managing student affairs. Finally, your dedication to ensuring smooth administrative. Without your unwavering support, this achievement would not have been possible. Thank you.

This work is supported by Fundamental Research Grant Scheme (FRGS/1/2020/TK0/UPM/02/19).

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy. The members of the Supervisory Committee were as follows:

Ratna Kalos Zakiah binti Sahbudin, PhD
Associate Professor
Faculty of Engineering
Universiti Putra Malaysia
(Chairman)

Aduwati binti Sali, PhD
Professor, Ir.
Faculty of Engineering
Universiti Putra Malaysia
(Member)

Fazirulhisyam bin Hashim, PhD
Associate Professor
Faculty of Engineering
Universiti Putra Malaysia
(Member)

ZALILAH MOHD SHARIFF, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: 12 December 2024

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
AI	Artificial Intelligence
ANN	Artificial Neural Network
BBA	Binary Bat Algorithm
BOA	Butterfly Optimization Algorithm
BSA	Backtracking Search Algorithm
CCMGPSO	Cooperative Coevolutionary Multi-Guide Particle Swarm Optimization
CNN	Convolutional Neural Network
DDM	Drift Detection Method
DE	Differential Evolution
DE	Differential Evolution
DFA-GPE	Dynamic Feature Aware Genetic Programming Ensemble
DFS	Dynamic Feature Selection
FOA	Fruitfly Optimization Algorithm
FS	Feature Selection
GA	Genetic Algorithm
GP	Genetic Programming
GWO	Grey Wolf Optimizer
HHO	Harris Hawks Optimization
IDS	Intrusion Detection System
IoT	Internet of Things
KNN	k-Nearest Neighbors
LR	Logistic Regression
LSTM	Long Short-Term Memory

MEPSOLA	Multi-Exemplar Particle Swarm Optimization with Local Awareness
MOEA	Multi-Objective Evolutionary Algorithm
MOO	Multi-Objective Optimization
MOPSO	Multi-Objective Particle Swarm Optimization
NB	Naive Bayes
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
NSGA-III	Non-Dominated Sorting Genetic Algorithm III
OSELM	Online Sequential Extreme Learning Machine
PC	Learning Probability
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
Rep	Repository
RF	Random Forest
RFE	Recursive Feature Elimination
SA	Simulated Annealing
SAR	Sequential Adaptive Refinement
SMOTE	Synthetic Minority Over-sampling Technique
SSA	Salp Swarm Algorithm
SVM	Support Vector Machine
VLMO-PSO	Variable Length Multi-Objective Particle Swarm Optimization
VLO	Variable-Length Optimization

CHAPTER 1

INTRODUCTION

1.1 Background

In the modern era, our lives are increasingly surrounded with a complex web of interconnected networks. This includes the vast expanse of the Internet of Things (IoT), the proliferation of social networks, complex computer networks, and extensive sensor arrays, all integral to our daily operations. This extensive connectivity of diverse devices results in the generation of vast, high speed and dimensional data streams. The term high-dimensionality refers to datasets containing a large number of attributes or features, often due to the varied types and sources of information within these networks [1]. The rapid evolving nature of these high-dimensional data streams poses significant challenges to traditional data analysis techniques. This underscores the urgent need for adaptable methods that can effectively adapt to these changes, especially considering that the assumption of uniformly distributed data streams is becoming outdated [2].

Moreover, the vast network connectivity across different domains not only complicates the structure of data but also intensifies the risk of cyber threats and increases network vulnerability [3]. This case intensifies the need for a robust and advanced security measures to safeguard these networks against potential cyber threats [4]. In response to this critical need, Intrusion Detection Systems (IDS) have emerged as a key solution, whether implemented as software or hardware. IDSs are specifically designed to detect unauthorized activities and malicious attacks within networks or systems. IDS play a

crucial role in identifying potential threats by monitoring network traffic and system activities, serving as an essential line of defense in maintaining the integrity and security of these complex digital networks [5].

Historically, intrusion detection was a manual procedure undertaken by analysts, who visually monitored and examined the user's logs on fan-fold paper, investigating for any anomalies or unusual activities during employees' days off or weekends, this task was extremely extensive and exhausting. This approach experienced a significant change with the introduction of the first IDS by Anderson in 1980s. The IDS was initially designed for government and military purposes, it employed statistical analysis to examine patterns of users behavior, flagging significant deviations as potential security threats [6]. As cyber threats evolved over years, IDS also experienced significant transformations. The late 1990s and early 2000s marked a pivotal era when Artificial intelligence (AI) and Machine Learning (ML) began to play a crucial role in this field and contribute to IDS and enhance their effectiveness. During the current era, IDSs began to evolve from rule-based to more sophisticated algorithms capable of detecting complex patterns of malicious activity, marking a significant shift in the approach to digital security [5]. Today the complexity of data streams that IDSs should handle presents a formidable challenges due to their non-stationary nature [7].

In addition to facing the challenge of high dimensionality, IDS also confront the critical challenge of concept drift in data streams [8]. Concept drift involves changes in the underlying statistical properties of data distribution, which can arise due to several factors such as; shifting in user behavior, evolving attack tactics [9], or changes in the network environment [10]. This phenomenon results in a gradual or sudden loss of

accuracy for models trained on previous data distributions over time [9]. Within the scope of concept drift, feature drift stands out as a paramount challenge [11]. It occurs when the relevance of certain feature subsets for model predictions shifts over time, potentially rendering once-effective detection models obsolete. This shift significantly impacts the stability and reliability of threat detection models, leading to reduced accuracy. Adding to aforementioned challenges the problem of imbalanced datasets, which further complicates the challenge of IDS. This imbalance biases detection models toward the majority class, often at the expense of the minority class, which may represent more serious or rare threats patterns. Such bias leads to skewed performance and an underestimation of the minority class's risk [12]. To achieve robust performance in IDSs, IDS design must simultaneously address these interconnected challenges. The dimensionality reduction, concept drift adaptation, with a special emphasis on feature drift handling and addressing class imbalance, is essential for maintaining the efficacy and accuracy of the IDSs in dynamic data streams.

Feature Selection (FS) techniques are essential for IDS in addressing the challenge of high dimensionality data streams [13]. FS effectively reduce the dimensionality by selecting the most relevant features and eliminating irrelevant and redundant once, thereby enhancing learning efficiency, reducing complexity, and avoid overfitting risks [2]. FS is considered a combinatorial optimization problem accomplished by two primary objectives to improve the overall performance of IDS and to reduce memory consumption [14]. Many of the IDSs based on FS methods discussed in the literature are considered to employ static feature selection. These approaches ignore the assumption that the significance of the feature may change over time. By selecting a single subset of features to represent the entire dataset, such techniques may struggle in

dynamic environments as in IDS application [15], that may exhibit feature drift. Therefore, Dynamic Feature Selection (DFS) technique becomes important and should be adopted, as it continuously and automatically updates the selection of the most relevant features over time in response to expected changes in feature relevance [16].

Meta-heuristic techniques play a vital role in solving the complex problems of feature selection, especially in the context of high-dimensional data streams [17]. The necessity for advanced meta-heuristic techniques in IDS becomes evident when considering the complexity of obtaining an optimal subset of features, which is an NP-Hard problem (computational challenge). Where meta-heuristic algorithms excel in dealing with these combinatorial problems, outperforming exhaustive or greedy approaches [18]. However, meta-heuristic FS based IDS techniques must address the dynamic nature of feature drift within data streams. Feature drift requires continuous adaptation of detection models. Meta-heuristics employ two key strategies: exploration and exploitation. Exploitation focuses on searching areas surrounding the known solutions, while exploration explore into new search space and avoid premature convergence. A balance between them is to be enabled in order to prevent lacking of convergence due to exploration and sub-optimality due to exploitation [19]. Involve an affective technique to trade-off the balance is vital for managing feature drift in IDS, ensuring the system remains effective despite evolving data characteristics [20]. Meta-Heuristic Multi-Objective Optimization (MOO) algorithms like Multi-Objective Particle Swarm Optimization (MOPSO) [21] is compatible with handling FS's multiple objectives such as balancing accuracy with memory reduction [20]. MOPSO excels in simultaneously optimizing conflicting objectives, thereby facilitating the development of models adept at navigating the complexities of modern data environments [22]. However, direct

implementation of MOPSO on high-dimensional datasets may require a significant amount of memory and computation time. Additionally, it may suffer from premature convergence, especially in scenarios involving thousands of features, this highlighting the needs for incorporating several techniques to solve the a forementioned challenges [23]. Techniques such as the transfer function [24], to map the solutions into binary decision space and balancing between exploration and exploitation, and using variable length features representation to handle the problem of the high dimensions of numbers of features [23]. These two techniques could improve the searching process of the algorithm it also introduces practical challenges. Using transfer functions to map continuous variables into binary space introduces practical challenges, such as increased complexity in the transformation process, inefficiencies in optimization when relying on single shapes of transfer functions, and the potential for premature convergence. Selecting the appropriate transfer function requires careful tuning to balance exploration and exploitation, and maintaining population diversity can be difficult, particularly in high-dimensional spaces. Despite these challenges, effective management of transfer functions can streamline feature selection and improve model accuracy.

Variable-length representation also introduces practical challenges. Managing particles of different lengths increases computational complexity and consume resources, impacting processing time. Balancing exploration and exploitation with changing feature sets becomes more complex. Despite these challenges, variable-length representation reduces feature subset sizes and improves classification performance, making it valuable for high-dimensional data.

To adapt the MOPSO algorithm for real-time decision-making, the data should be divided into chunks, allowing the algorithm to handle continuous data streams by focusing on the most relevant features in each chunk [25]. The algorithm initializes solutions using a heuristic approach, with features ranked by symmetric uncertainty and allocated into divisions, ensuring a good starting point for the optimization process. Particles are guided by multiple exemplars, chosen based on performance and grid criteria [26]. This method enhances exploration and exploitation by dynamically adjusting the objective of exemplars every T time period. The exemplar selection process involves sorting solutions by grid occupancy and using a probability-based mechanism to ensure diverse guidance for the particles. Solutions are temporarily converted to a binary representation for movement, using mixed S-shaped and V-shaped transfer functions to map velocities to new solutions and avoid relaying on single shape [27]. This conversion helps in maintaining a fixed dimension for calculations while allowing the flexibility of variable-length feature representation. In addition to vary lengths of features, conditional mutation alters some elements in the solution to avoid local minima. Finally, a voting-based approach is used to select the best features from the Pareto front, ensuring a balanced trade-off between accuracy and feature reduction. This process performed automatically on each arriving chunk.

On the other hand, IDS is commonly considered as a classification problem [28], where many approaches utilized a single classifier for this task [29] [30]. While a single classifier may perform well in simple structured data flows, it can struggle in complex high-dimensional data streams. This challenge brings the ensemble classification technique to the surface. It combines multiple weak learners to create a stronger decision-making classifier, aiming to improve two key aspects: enhancing the accuracy

of the classifier and addressing the issue of class imbalance [31]. In an ensemble classifier, the most important component is the aggregation function or the combiner, especially in the domain a data streams where the properties of the data distribution may change. This necessitates retaining and adapting both the combiner and the classifiers to cope with the new data distribution. Some combiners, such as those used in the weighted voting method, may struggle in this context. Therefore, one powerful non-retrainable combiner is a Genetic Programming (GP)-combiner that enables using more than one classifier under a tree representation and provides various operators for aggregating the classifiers outcomes such as average, maximum and median, and it allows self-re-training of the aggregation when a drift happens in the data stream [32]. In addition to the non-stationary nature of the incoming data stream, this type of data flow can have infinite length and it could be endless, making it infeasible to store the data on many available memory storage systems [33]. In this scenario, a single-pass learning process or incremental learning is preferable rather than storing the incoming data flow for later processing. It is advisable to abandon the common assumption of data availability of representative training data during the training period to develop the classifier [34].

Therefore, based our knowledge this thesis proposed a novel IDS framework that adeptly handles DFS through a developed Variable-Length Multi-Objective PSO (VLMO-PSO) for handling the dynamic nature of feature evolutions. This achieved by two pivotal phases; first, the development of a standalone optimization algorithm for solving a complex problem with conflicted objectives nature integration of conditional local searching for exemplar selection and multi-exemplar strategy. Secondly, the adaptation and adjusting of this algorithm to effectively manage online decision-

making in PSO DFS for IDS. This approach addresses the critical gap identified in the literature, which lacks the integration of incremental learning ensemble learning based on genetic programming (GP) combiner for concept drift awareness and handling, alongside meta-heuristic methods for feature drift and high dimensionality awareness. By integrating these elements, the proposed framework aims to enhance the adaptability, accuracy and reduce memory consumption of IDS in non-stationary data streams and class imbalance handling, providing a robust and comprehensive defense against the evolving landscape of cyber threats.

1.2 Motivations

The panorama of cyber threats in network-based internet and large-scale networks is evolving rapidly, marked by increasing sophistication and variability. This progression is driven by the complexity and interconnectivity inherent in modern network structures, which present novel vulnerabilities and attack vectors. Cyber-attacks are becoming more complex, targeting not just traditional IT infrastructure but also encompassing a broad spectrum of networks and systems as well as servers. This escalation in threat complexity necessitates advanced, adaptive security measures capable of countering a diverse range of cyber threats. The evolving nature of these threats underscores the need for frequent innovation in cybersecurity strategies and technologies to protect these extensive and interconnected networks [35].

According to the national cyber security organizations, FBI and IMF [36], the global cost of cybercrime in the next three years is expected to rise from $14.6 trillion in 2024 to $23.84 trillion by 2027, as illustrated in Figure 1.1. Accordingly, in light of this

escalating threat and the demand for efficient real-time detection, IDS's fundamental design aims to achieve real-time detection and response by analyzing data streams data as they are received, allowing for the timely detection of intrusions and threats [37].

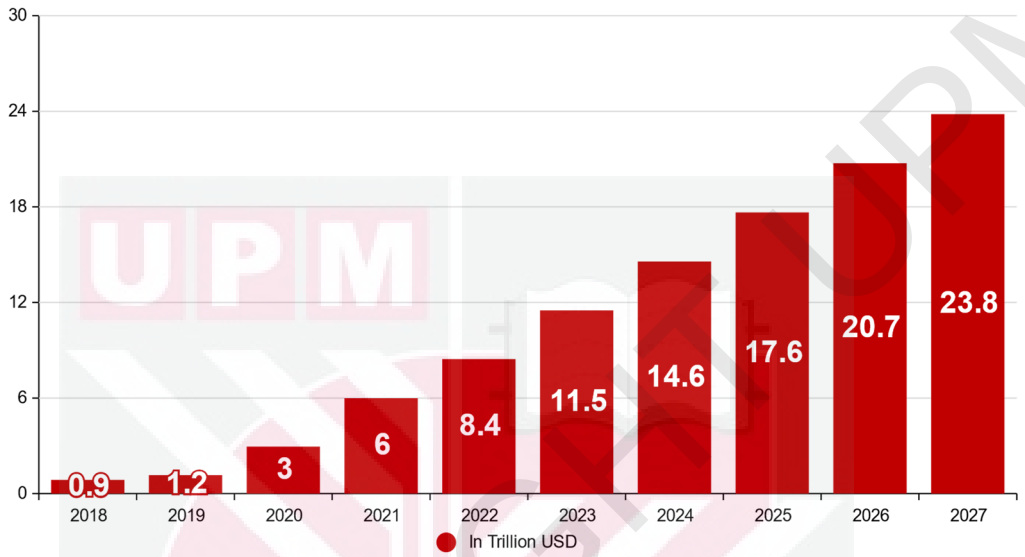


Figure 1.1: Anticipated global costs of cyber-attacks in the next years [36]

IDS in the current landscape of sophisticated cyber threats. It highlights the insufficiency of static models in evolving threat environments where the nature and tactics of attacks are constantly changing [9]. The traditional IDS, designed for more static and predictable environments, may struggle to adapt to the dynamic nature of modern cyber threats, making them less effective in detecting and responding to novel and complex attacks. This evolving scenario calls for more adaptive and advanced IDS solutions capable of handling the unpredictable nature of contemporary cyber threats. Concept drift and feature drift are critical phenomena in the domain of cybersecurity, impacting the effectiveness of any IDS based ML [1]. Concept drift could take forms such as sudden drift, gradual drift, incremental drift, recurring drifts, etc [38]. These drifts in some cases may reflect the dynamic nature of cyber threats, where attack

patterns may suddenly change [39], evolve gradually, or even recur after a period. Feature drift, on the other hand, occurs when a subset of features in the data becomes more or less relevant to the concept being learned. This change in feature relevance poses a significant challenge to traditional IDS, as the models must continuously adapt to these evolving features to maintain accuracy [11].

In cybersecurity, these drifts mean that attack and methods can evolve, rendering previously effective detection strategies less effective or even obsolete [8]. The attacks or threats can contribute in drift by manipulate the data stream by flipping labels or altering feature values or by changing class boundaries within the data or even features significance. Static models struggle in this evolving environment as they trained on historical data (outdated data), as they are often designed for a predefined set of features and threats. Therefore, the ability to detect and adapt to concept and feature drift is crucial for the continued effectiveness of IDS in the constantly changing landscape of cyber threats [40].

The significant impact of feature drift on IDS performance could be significate, for examples where existing ML methods may fall short, in research [11] offers empirical evidence from benchmarking various algorithms and a detailed case study involving the Spam Corpus dataset. These examples demonstrate how static models fail to maintain their accuracy and effectiveness over time due to evolving feature relevance and the dynamic nature of cyber threats. Empirical evidence from experiments that benchmark various algorithms under conditions of feature drift shows that algorithms not designed to handle feature drift experience significant drops in accuracy. For instance, decision trees and rule learners struggle to maintain performance when feature

drift occurs. The changing importance of specific features over time leads to decreased model performance if not adapted. This study highlights the dynamic nature of feature relevance in data streams and the consequent decline in accuracy for static models.

Additionally, an example from a study on the Spam Corpus dataset clearly illustrates the impact of feature drift on model accuracy [11]. The study observed that the importance of two features, "directed" and "info," changed significantly over time. Initially, the feature "directed" was highly relevant and played a substantial role in the model's decision-making process. However, as the nature of spam evolved, the relevance of "directed" decreased, while the importance of the feature "info" increased. This shift in feature relevance highlights the need for an IDS to have adaptive mechanisms that can continuously update the feature set. Without such adaptation, the model's accuracy declines because it continues to rely on outdated feature importance. This example clearly demonstrates the critical need for adaptive techniques to handle feature drift, ensuring that IDS can maintain its effectiveness in ever-changing environments.

The challenges posed by concept and feature drift in cybersecurity where attack patterns and feature relevance continually evolve directly relate to real-world scenarios like WannaCry attack. WannaCry ransomware outbreak in 2017 serves as a stark reminder of the evolving challenges in cybersecurity [41]. This devastating attack exploited vulnerabilities in older Windows operating systems causing widespread global disruption and highlighting the limitations of traditional IDS. Initially, IDS struggled to effectively detect WannaCry due to its novel use of payload and developed techniques, which were unlike previously known malware patterns. In the aftermath,

cybercriminals continued to develop and release variants of WannaCry, each with slight modifications that further evaded static IDS rules. This continuous evolution of the ransomware for an example underscored a critical weakness in conventional defense approaches, their inability to adapt quickly to new and evolved attacks. This incident clearly demonstrates the necessity for more dynamic and adaptive detection mechanisms in IDS to counter such sophisticated and evolving cyber threats effectively. Another recent example highlighting the need for adaptive IDS is the surge in phishing attacks during the COVID-19 pandemic. Attackers exploited public anxiety, using legitimate pandemic-related terms in their malicious campaigns, thereby overcome traditional keyword-based IDS methods [42]. In response, there is a call for a paradigmatic shift toward dynamic and adaptive IDS methodologies. Managing concept drift and feature drift in high dimensional data stream is crucial for the continued effectiveness of IDS in the constantly changing landscape of cyber threats.

1.3 Problem Statement

In the domain of vast networks, IDS confront significant challenges posed by high-dimensional data streams. These expansive networks generate data characterized by a plenty of features, complicating traditional data analysis methods and requiring innovative approaches for effective dimension management [1]. Additionally, these high-dimensional data streams often result in unequal data distribution, challenging the conventional IDS assumptions of uniformly distributed data flows [33]. One of the critical issues arising from this environment is concept drift, where the underlying statistical properties of data distribution change over time, affecting model accuracy and reliability [38]. However, an important and frequently overlooked challenge in

IDS is feature drift. Feature drift occurs when the relevance of specific features within the data stream shifts, potentially rendering existing detection models less effective [11]. This challenge is compounded by the lack of DFS strategies in many IDS, leaving them struggle to adapt to these rapid changes in data relevance. Additionally, IDS must contend with imbalanced datasets, where the skewness towards majority classes can lead to an underrepresentation of critical but less frequent threat patterns [43].

To further clarify the problem statements derived from our comprehensive review of IDS in data streams, we outline three critical problems:

1. **Genetic programming-based ensemble framework is useful for handling concept drift, however, it suffers from lacking of mechanisms of handling feature drift.**

The literature on IDS for non-stationary data streams reveals a range of approaches as shown in Tables 2.3 and 2.4, each addressing different aspects of the data stream problem. These approaches varied from Support Vector Machine (SVM)-based optimization algorithms to ensemble classifiers and deep learning methods, these studies demonstrate various techniques to tackle dimensionality reduction, concept drift, and data imbalance. However, a clear gap emerges in integrating ensemble learning based GP for concept drift awareness and handling, equipped with meta-heuristic methods for feature drift and high dimensionality reduction. Most approaches, such as those by [29] and [9], focus on dimensionality reduction through methods like Principal Component Analysis (PCA) or optimized SVM parameters while ignoring feature drift a ware. Similarly, methods like Long Short-Term Memory (LSTM) [44] and varying deep neural networks [45] targeting concept drift and data imbalance but lack a comprehensive approach to feature drift and high dimensionality.

Ensemble methods, as introduced by [46], show promise in addressing these challenges but still fall short of a holistic solution specially genetic programming ensemble learning by [32] for timely adapt to concept drifts by timely updating the ensemble model. This highlights the necessity for an integrated approach that combines the strengths of GP ensemble classifier for concept drift awareness and DFS techniques for feature drift and high dimensionality awareness. Such an approach would not only enhance the adaptability and accuracy of IDS in non-stationary data streams but also ensure a more robust and comprehensive defence against evolving cyber threats.

2. **Particle Swarm Optimization (PSO) has been used as multi-objective optimization algorithm for solving complex optimization problem, however, it suffers from the issue of lacking diversity because of using single exemplar and it might get stuck with local minima. Hence, incorporating conditional local search with multi-exemplar strategy is useful. Additionally, the use of intelligent initialization can overcome problems in existing algorithms that are sensitive to the initial population setup.**

The literature of MOO methods as shown in Table 2.5, particularly in the domain of PSO, reveals a diverse array of approaches and techniques. A key observation is the varying degree of emphasis on critical aspects such as smart initialization, trade-off between exploration and exploitation, different methods of solution selection, and some of handling of non-dominated solutions. For example, methods like the Modified Self-Adaptive PSO [47] and Fuzzy Multi-Objective PSO [48] demonstrate the integration of trade-off strategies and non-dominated solution techniques but lack smart initialization. On the other hand, the MOPSO with self-adjusting strategy [49] incorporates smart initialization, highlighting its potential in enhancing search efficiency and convergence but lack in local searching and multi-objectives exemplar strategy. However, there is a notable gap in simultaneously incorporating features like

smart initialization, trade-off exploration and exploitation, non-dominated solution technique with an effective solution selection mechanism with the addition of an efficient conditional local searching in exemplar selection method and multi-objectives exemplar process in PSO. Where local searching could provide more thorough exploration of the solution space avoiding premature convergence, while a multi-objectives exemplar process could offer diverse perspectives in optimizing multiple objectives, in addition smart initialization could offer diversity in population at the start of the search, fast convergence, avoid over presented solutions as well explore wider region from the solution space. This integrated approach would significantly enhance the efficacy of MOO methods, providing more effective and efficient solutions in complex, in high-dimensional, and dynamic problem spaces.

3. **Multi-objective meta-heuristic optimization has been used extensively for handling feature selection problem. However, it has not been used in an online mode for dynamic feature selection with support of selecting relevant features in real-time.**

The literature of MOO based methods for FS across various computational methods as shown in Table 2.6, such as Genetic Algorithm (GA) [50] , Butterfly Optimization Algorithm (BOA) [51] [52] and various methods of PSO as in [53], [54], [55] or [56], reveals a few of strategies addressing the problem of high-dimensional data. Despite the diversity and significance of these approaches, a significant gap is evident in addressing non-stationary data streams and absence of utilizing DFS, crucial for managing feature drift in data streams. Another notable observation is that while some methods employ binary representation, essential for discrete search problems like feature selection as it enables precise delineation of feature inclusion or exclusion, the comprehensive handling of concept drift and data imbalance is not extensively

covered. Furthermore, the benefit of utilizing variable-length feature selection representation is underexplored. Variable-length feature selection representation offers adaptability in evolving feature spaces, allowing the model to adjust its complexity based on the underlying data structure. This flexibility can lead to more accurate feature selection in dynamic environments, where the length of relevant features may change over time. Based on the literature an efficient approach in MOO-based FS would be a method that jointly addresses the a forementioned challenges [55], [57], [32] [43] of non-stationary data streams and DFS, particularly focusing on feature drift [11]. Such a method would ideally combine multiple trade-off strategies between exploration and exploitation, advanced solution selection techniques, and effectively tackle concept drift and data imbalance. Incorporating binary representation for discrete search problems and variable-length solutions representation would further enhance the method's adaptability and efficacy in complex, high-dimensional, and dynamic problem spaces.

1.4 Objectives

The ultimate goal of this thesis is to propose an approach for stream data-based intrusion detection that adeptly handles feature drift. This goal is realized through the following specific objectives:

1. To develop a Dynamic Feature Aware Genetic Programming Ensemble (DFA-GPE) framework for data stream-based Intrusion Detection Systems (IDS) that addresses both concept drift and feature drift effectively, ensuring continuous adaptability and high performance over time.
2. To design a Multi-Exemplar Particle Swarm Optimisation with Local Awareness (MEPSOLA) algorithm that incorporates advanced strategies including smart initialization, multi-exemplar by allowing particles to follow

multiple exemplars, and conditional local search for T period of time for solving conflicted Multi-Objective Optimisation problems.

3. To develop and adapt the Variable Length Multi-Objective Particle Swarm Optimisation (VLMO-PSO) for online dynamic feature selection within the DFA-GPE framework, incorporating multi-exemplar strategies by allowing particles to follow multiple exemplars to improve the algorithm diversity.

1.5 Scope

This thesis presents an approach to stream data-based intrusion detection, with a particular focus on addressing feature drift. It details the development and evaluation of a dynamic framework for stream-based IDS, designed to adapt to both concept drift and feature drift. The framework is tested on two datasets: HIKARI-2021 and TON_IoT Telemetry 2020. The DFS method is built upon a variable-length multi-objective optimization algorithm, utilizing PSO and specifically tailored for high-dimensional data spaces. This algorithm is integrated into the IDS framework, focusing on real-time feature selection. The proposed IDS framework is evaluated using these datasets, with performance measured through standard metrics such as accuracy, precision, recall, F1-score, and memory reduction. Additionally, the MOO algorithm is assessed using metrics like generational distance, set coverage, hypervolume, delta metric, and the number of non-dominated solutions. This evaluation provides a comprehensive benchmark against current state-of-the-art methodologies in intrusion detection. Table 1.1 illustrates the alignment between the research objectives and the scope of the study.

Table 1.1: Overview of Objectives and Associated Activities for IDS Framework Development

Objectives	Scope
Development of IDS Framework stream-based	Developing a framework for stream-based Intrusion Detection Systems adapt to both concept drift and feature drift.
Development of Multi-Objective Optimization Algorithm	Proposing and developing a multi-objective optimization algorithm tailored for high-dimensional data spaces, particularly emphasizing on multi exemplar and local search methods.
Adapt MOO for Dynamic Feature Selection	Integrating the developed optimization algorithm for dynamic feature selection within the IDS framework.

1.6 Research Contributions

To the best of our knowledge, this research is the first to introduce IDS based on ML that handles various challenges jointly, namely, feature and concept drifting, high dimensionality data, and data imbalance. This is enabled by leveraging the GP-combiner framework and integrating the DFS block within the framework.

To the best of our knowledge, this research introduces a pioneering MOPSO algorithm specifically designed for complex optimization problems. It features an innovative multi-exemplar selection criterion that finely tunes the balance between exploration and exploitation, ensuring a robust search strategy. Furthermore, the integration of a Tabu search mechanism provides a nuanced approach to search behaviour, granting the algorithm both global and local awareness while adeptly avoiding the trap of local minima.

To the best of our knowledge, this research introduces an innovative approach to online decision-making in PSO by proposing a multi-objective variable-length PSO framework for high dimensionality feature selection that effectively handles binary decision spaces. It incorporates advanced initialization and a mix of transfer functions

to guide the PSO algorithm, enabling fast convergence and a diverse search within the solution space. Through these mechanisms, the PSO algorithm becomes a dynamic decision-maker, adept at selecting optimal solutions in real-time for complex optimization problems data streaming.

The Figure 1.2, depicts the structured layout that connects the research gaps with the objectives to tackle them and the corresponding contributions from achieving these objectives.

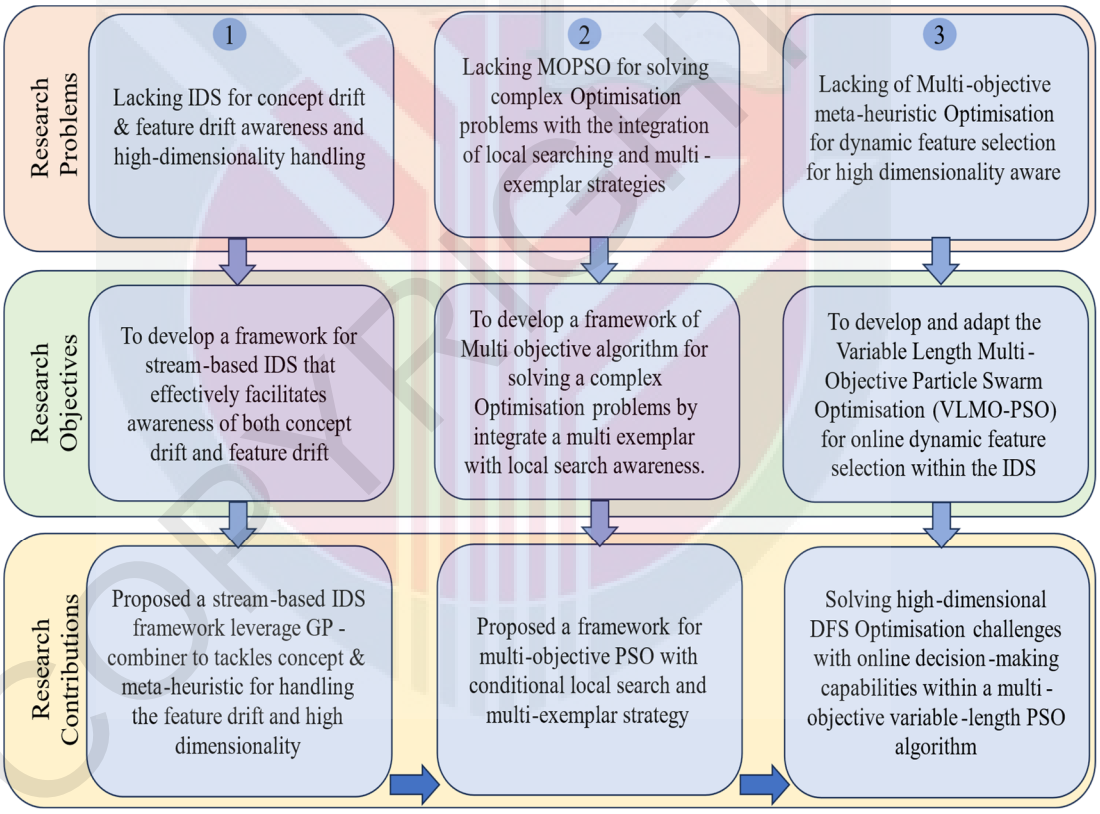


Figure 1.2: Alignment of Research Problems, Objectives, and Contributions of the research work

1.7 Thesis Organization

The organization of this thesis is methodically outlined as follows:

Chapter 1: explains the foundational aspects of the thesis are laid out, focusing on the critical challenges facing modern IDS, particularly in handling high-dimensional, non-stationary data streams. The problem statement highlights the limitations of existing IDS frameworks in adapting to concept drift and feature drift, which can significantly degrade performance over time. The objectives are clearly defined, aiming to develop a novel framework that integrates advanced meta-heuristic algorithms and dynamic feature selection techniques to enhance IDS adaptability and accuracy. The scope of work encompasses the design, implementation, and evaluation of these techniques within a real-time IDS framework, emphasizing the need for robust solutions to counter evolving cyber threats. This chapter establishes the direction of the research and sets the stage for the subsequent development and validation of the proposed methods.

Chapter 2 conducts an extensive literature review, exploring into key topics in IDS and the challenges posed by data streams. The chapter also provides a comprehensive overview of foundational concepts and the latest developments at the intersection of IDS and optimization algorithms.

Chapter 3 explores the research methodology, detailing the experimental design and methodologies for both MOPSO and the adaptation of the proposed MOPSO for online FS, along with the methodology for the novel IDS framework.

Chapter 4 presents the results and the comprehensive evaluation for both MOPSO and the IDS frameworks, followed by a discussion covering all results.

Chapter 5 introduces the thesis conclusion, highlights limitations, and outlines potential paths for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter presents a comprehensive literature review in the field of Intrusion Detection Systems (IDS), focusing on the challenges and methodologies associated with non-stationary data streams in cybersecurity, specifically the challenge of the phenomena of feature drift. It explores the evolution and principles of IDS, highlighting their significance in the rapidly changing landscape of network security. The chapter further explores various aspects such as the rise of high-dimensional challenge in data streams, the criticality of Feature Selection (FS) methods, and the distinction between static and Dynamic Feature Selection (DFS) techniques. It also discusses the importance of the variable length optimization. It also examines the integration of Multi-Objective Optimization (MOO) with Particle Swarm Optimization (PSO) in FS, underscoring their role in enhancing IDS in both objective accuracy and memory consumption. This review lays the groundwork for understanding the complexities of IDS in modern network environments, setting the stage for the subsequent chapters of the thesis.

2.2 Background

An IDS is a fundamental technique in cybersecurity, operate as a tool that monitors network traffic to identify and mitigate cyber threats [58]. IDSs leverage machine

learning techniques to classify the incoming data, mainly using ensemble learning. These techniques achieve enhanced classification accuracy and error reduction [59]. The evolving complexity of network traffic introduces challenges like high-dimensionality and dynamic nature or non-stationary data streams [60] [61]. In this domain, FS methodologies are pivotal. They aid in reducing dimensionality by selecting informative features and discarding irrelevant ones, thus optimizing IDS performance against constraints like memory usage [61]. Incorporating meta-heuristic algorithms in FS significantly enhances the efficiency of IDS [62]. By intelligently exploring the search space, meta-heuristic algorithms find optimal or near-optimal feature subsets, balancing IDS performance with constraints like memory usage [20]. The dynamic nature of data streams is critical, especially the challenge of feature drift [11], necessitating adaptive IDS strategies to effectively handle high-dimensional data in non-stationary network environments, where many conventional IDS techniques often struggle in such environments [32]. For more clarification about the literature review structure, refer to Figure 2.1.

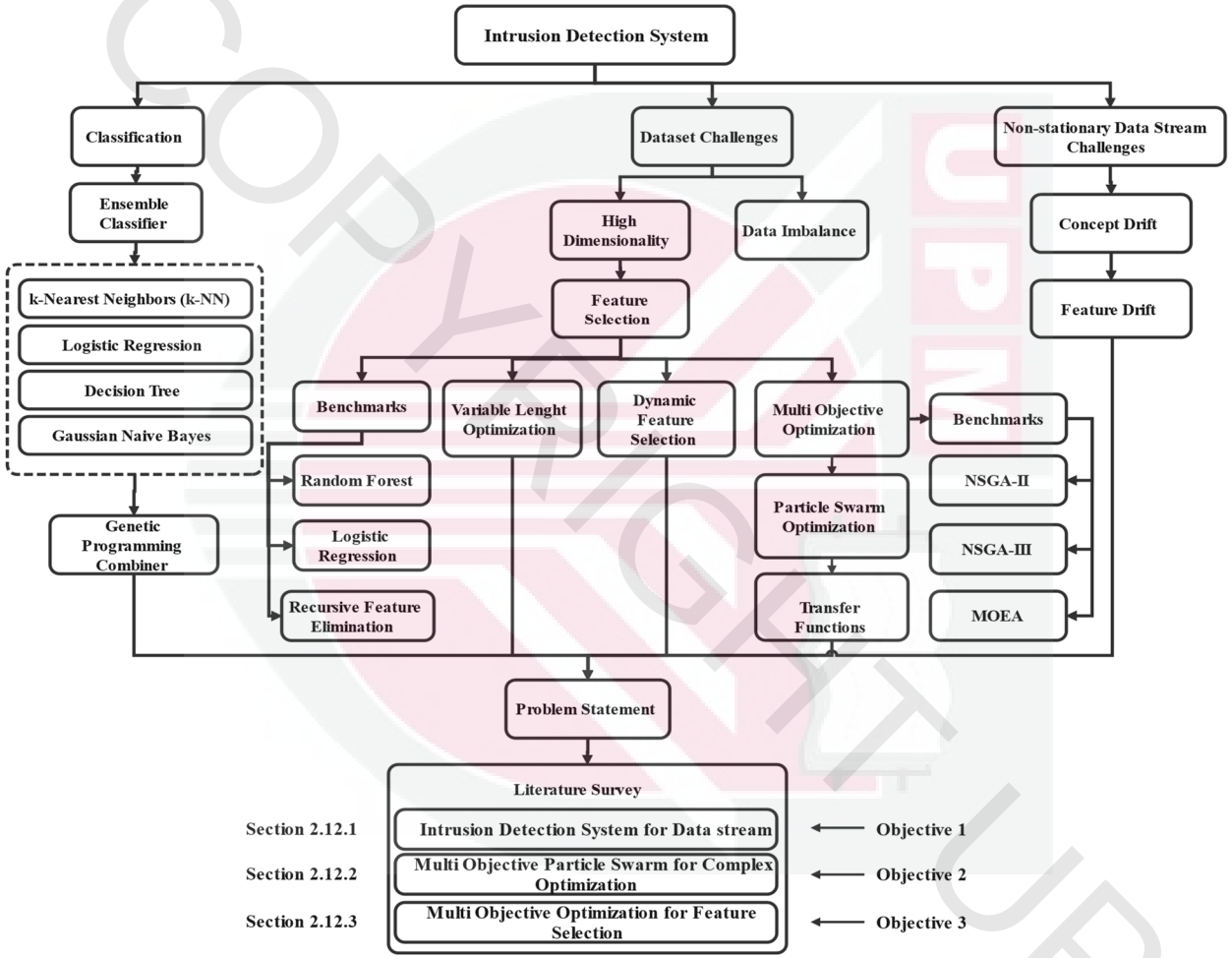


Figure 2.1: Literature Review Map for Intrusion Detection Systems in Dynamic Data Environments

2.3 Intrusion Detection System

IDS is a fundamental aspect of network security, and its history reflects a dynamic evolution in response to the continuously changing landscape of cybersecurity threats and network complexities. IDSs have undergone a long transformative journey from primitive processes to more complex tools, adapting to the ever-evolving landscape of threats and network complexity. In the past, administrative personnel performed intrusion detection manually by observing suspicious activities, such as unauthorized logins on local computers, printers, or employee device logins during off-hours. Systems administrators would then review the audit logs and investigate unusual activity. These logs were often printed on fan-folded paper, and the activities were observed manually, resulting in a time-consuming and exhausting task [63].

The field significantly jumped forward in 1980 when James Anderson introduced the IDS for governmental purposes [6]. Anderson posited that audit logs held valuable insights into inappropriate user behaviors or system misuse, encouraging notable advancements in systems and audit technologies. Later, in 1983, Dorothy Denning introduced further advanced innovation in the field by developing IDS for government purposes, focusing on analyzing mainframe computer audit trails and creating user behavior profiles [64]. With the advent of more affordable memory and storage in the early 1990s, audit logs transitioned to digital formats, although analysis remained a slow and resource-intensive process [65]. This limitation prompted researchers to develop real-time online IDS solutions, particularly to address the rising threat of malware attacks, such as spyware, worms, and botnets. These attacks, often undetected by users, are aimed at stealing information or damaging the data through unauthorized access [66].

The evolution of IDS mirrors the ongoing open war between cyber attackers and defenders. The recent surge in complex data traffic has challenged IDS-based machine learning with issues like high dimensionality and continuously changing data streams [67]. Consequently, FS has become a pivotal technique [68], with various approaches developed to streamline the data dimensions in IDS [69]. However, the need for automated FS processes has grown, addressing changes in data streaming, such as concept drift data distribution changes over time [70], and the emerging challenge of feature drift, where the relevance of features evolves [11].

The IDS field has witnessed a remarkable transformation from basic tools for log analysis tools or signature-based to more sophisticated, integrated artificial intelligence and multi-layered systems integrated with advanced security technologies [63]. This evolution has been driven by the escalating complexity of network attacks and the necessity to adhere to demanding security standards. Today, IDS utilize sophisticated machine learning algorithms to analyze network traffic and identify threats accurately [71]. They are capable of integrating with other advanced security technologies, offering a comprehensive, machine learning-driven security solution. In this ever-evolving landscape, the nature of data traffic, characterized by non-stationary streaming and increasing data volume, poses continual challenges [72]. The development of IDS remains an ongoing journey, marked by ongoing innovation and enhancements to keep pace with the evolving machine learning-driven threat landscape.

2.3.1 Intrusion Detection Systems Principles

IDS operate on fundamental and crucial principles for ensuring robust network security. These principles include constantly monitoring network or system activities

for prompt detection of threats or suspicious behavior. Real-time analysis is another essential principle wherein IDS systems rapidly assess incoming data to identify potential security threats, ensuring a rapid response [73]. Additionally, adaptive learning is fundamental, as IDS must evolve and adjust to evolving threats and changing network environments, maintaining their relevance and effectiveness. Collectively, these principles constitute the foundation of IDS functionality, which is crucial for modern cybersecurity strategies [13].

IDSs are classified into three main types based on deployment methods as shown in Figure 2.2. Network-based, Host-based, and Hybrid IDSs. Network-based IDSs monitor and analyze network traffic for intrusion signs in real time [74]. Host-based IDSs operate on a single host, analyzing system and application logs, configuration files, and system calls to detect intrusions [75]. Hybrid IDSs combine the strengths of both network-based and host-based systems, offering a more comprehensive intrusion detection approach [76]. Furthermore, IDS mechanisms are categorized into three types. Signature-based, Anomaly-based, and Hybrid IDSs. [77].

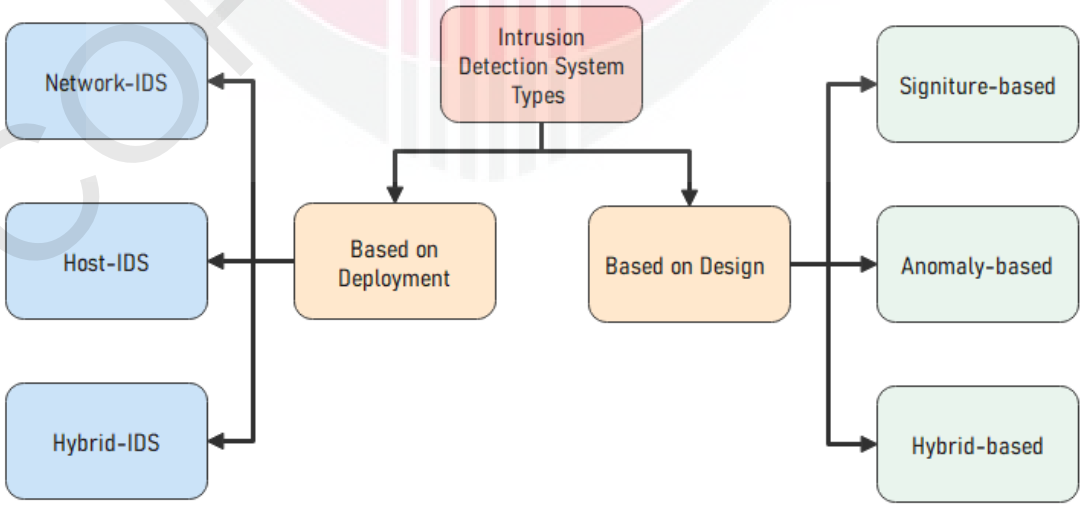


Figure 2.2: Intrusion Detection System Types [78]

2.3.2 Intrusion Detection System in Internet of Things (IoT)

The IoT represents a vast network of interconnected devices and applications designed to collect and exchange data between physical objects and the Internet through various communication methods. Gartner predicts a significant increase in IoT devices, expecting their number to reach 125 billion by 2030 [79].

The increasing reliance on IoT is evident in various sectors, including smart homes, industrial systems, medical fields and environmental monitoring. IoT networks include diverse devices, from simple sensors to more complex smartphones and computers [80]. This diversity makes IoT networks vulnerable to security challenges and attacks, such as Denial of Service DoS and Distributed Denial of Service DDoS attacks. A notable example is the 2015 attack on Ukraine's smart grid IoT [81]. In response to these risks, IDSs are employed to analyse data packets across different network layers, operating in challenging conditions with large data dimensions and heterogeneous environments [82]. Securing IoT platforms is critical for two primary reasons [83]. First, safety: IoT networks, used in vital sectors like the Internet of Medical Things (IoMT) and the Industrial Internet of Things (IIoT), directly impact human safety. Second, privacy: IoT platforms handle sensitive data like personal IDs and credit card information. Gaps in security allow unauthorized access, compromising user privacy.

The IoT architecture comprises three layers: the application, network, and perception layers [84], as illustrated in the Figure 2.3. In the perception layer, the bottom layer consists of the physical objects used to interact with environments and collect measurements and data. The network layer, or middle layer, is considered the most

important layer in the IoT because it gathers and processes the data from the perception layer and transmits it to the next layer. The last one is the application layer, which is considered the global management layer in IoT. Security challenges for IoT networks are similar to those in regular networks, such as Wireless Sensor Networks (WSN) or internet-based networks [85]. The security measurements cannot apply to the perception layer for two reasons: the limited computation resources and the high number of communicated devices [86].

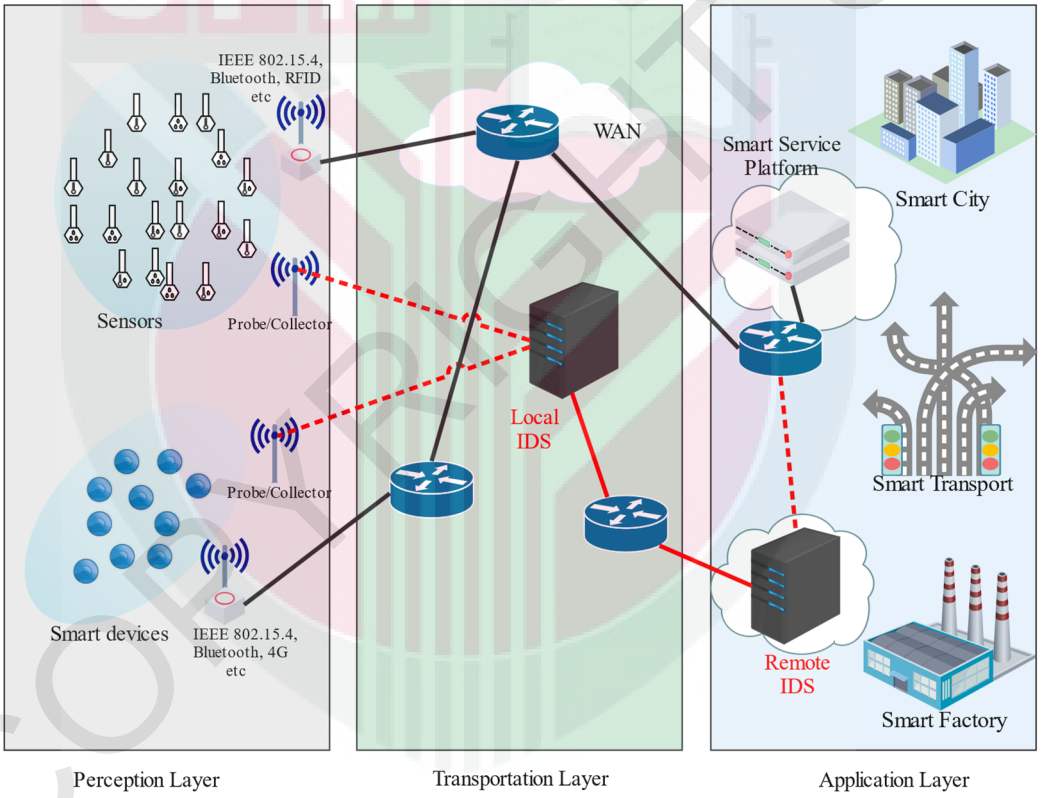


Figure 2.3: Common IDS Deployment Across IoT in Transportation and Application Layers [87]

Therefore, when designing any IDS for IoT environments, it's essential to consider their placement and architecture. IDS for IoT should ideally be situated on centralized nodes equipped with sufficient memory and computational resources. This approach

allows for effective implementation, especially given the inherent limitations in IoT related to power, memory, computational capacity, and specific design decisions. We can ensure a more scalable security solution by focusing the IoT-IDS on the network layer or incorporating it within the application layer [85] [88].

2.4 Data Stream

The vast networks and the widespread connectivity in today's digital landscape generate continuous data streams. These streams are sequences of digitally encoded signals updated in real time and derived from diverse sources. Their characteristics include high speed, real-time generation, and potentially unbounded nature. Unlike static datasets, these data streams represent a dynamic flow, necessitating advanced algorithms for immediate processing [89]. They typically include time-stamped records, allowing for chronological analysis. Crucially, in domains like financial domains, marketing, IoT device management, and security analytics, where timely data processing is vital, data streams also pose significant security challenges. Security analysis of these streams is essential to identify and mitigate potential threats, ensuring the integrity and confidentiality of the data in transit. The data stream can be presented by the following term:

$$D = \{(x_1, t_1), (x_2, t_2), \ldots., (x_\infty, t_\infty)\} \tag{2.1}$$

Where D is a data steam source of incoming tuples. Every $X_n = (x_n^1, x_n^2, \ldots \ldots x_n^d\}$ is the $D - dimensional$ vector, n denotes the number of data arrived at t time in the data stream, time index of each sample is vital in our thesis, and $Y = (y_1, y_2, \ldots., y_d)$ is the x corresponding label or attributes.

In data stream applications, especially those concerning security and real-time analytics, managing data streams is a significant challenge. Due to memory storage limitations, time sensitivity, and security requirements, storing these streams for later analysis often proves impractical [90]. This challenge stems from several factors. Firstly, stream data is sequential, necessitating real-time processing as control over instance arrival is limited. Secondly, the sheer volume of data streams demands efficient single-pass processing. Finally, the incoming rate of data streams may surpass system processing capacities [33].

In machine learning-IDS, incremental learning emerges as a promising solution to these challenges. This technique involves the continuous adaptation and updating of a learning model with each new instance, thereby eliminating the need to repeat the entire training process. This approach is convenient in scenarios where real-time processing, resource efficiency, and security are critical [91]. Incremental learning not only addresses the dynamic nature of data streams but also ensures that the model remains current with evolving data trends, making it an essential procedure in handling real-time, large-scale data stream applications.

2.4.1 Concept Drift

In data stream analysis, the traditional assumption of stationary, uniformly distributed data traffic is now outdated [92]. A significant challenge in this field is named concept drift, which denotes the expected changes in statistical properties of data distributions over time [8]. This phenomenon can significantly decrease the efficacy of machine learning models trained on historical data, leading to a loss of predictive accuracy for

the new incoming data. Concept drift can result from various factors, including external influences like seasonal variations or economic shifts, changes in user behavior, hardware ageing, software faults, and new cybersecurity threats. Therefore, addressing concept drift in IDS is a crucial step and necessary to maintain performance [93]. The representation of concept drift can be conceptualized as in following Equation (2.2) [94]:

Let's begin by defining a concept as the joint distribution, $P(X, Y)$. The concept drift can be observed between two specific time points, such as t_0 and t_1 if:

$$\exists X : P_{t_0}(X, Y) \neq P_{t_1}(X, Y) \tag{2.2}$$

Concept drift can manifest under two primary circumstances: First, changes in the prior probabilities of a class $P(X)$. Second, shifts in the conditional or posterior probabilities of $P(X \mid Y)$ [8].

Concept drift in data stream analysis is categorized into five distinct types based on the nature of its occurrence sudden, gradual, incremental, recurring, and blip [38], as shown in Figure 2.4.

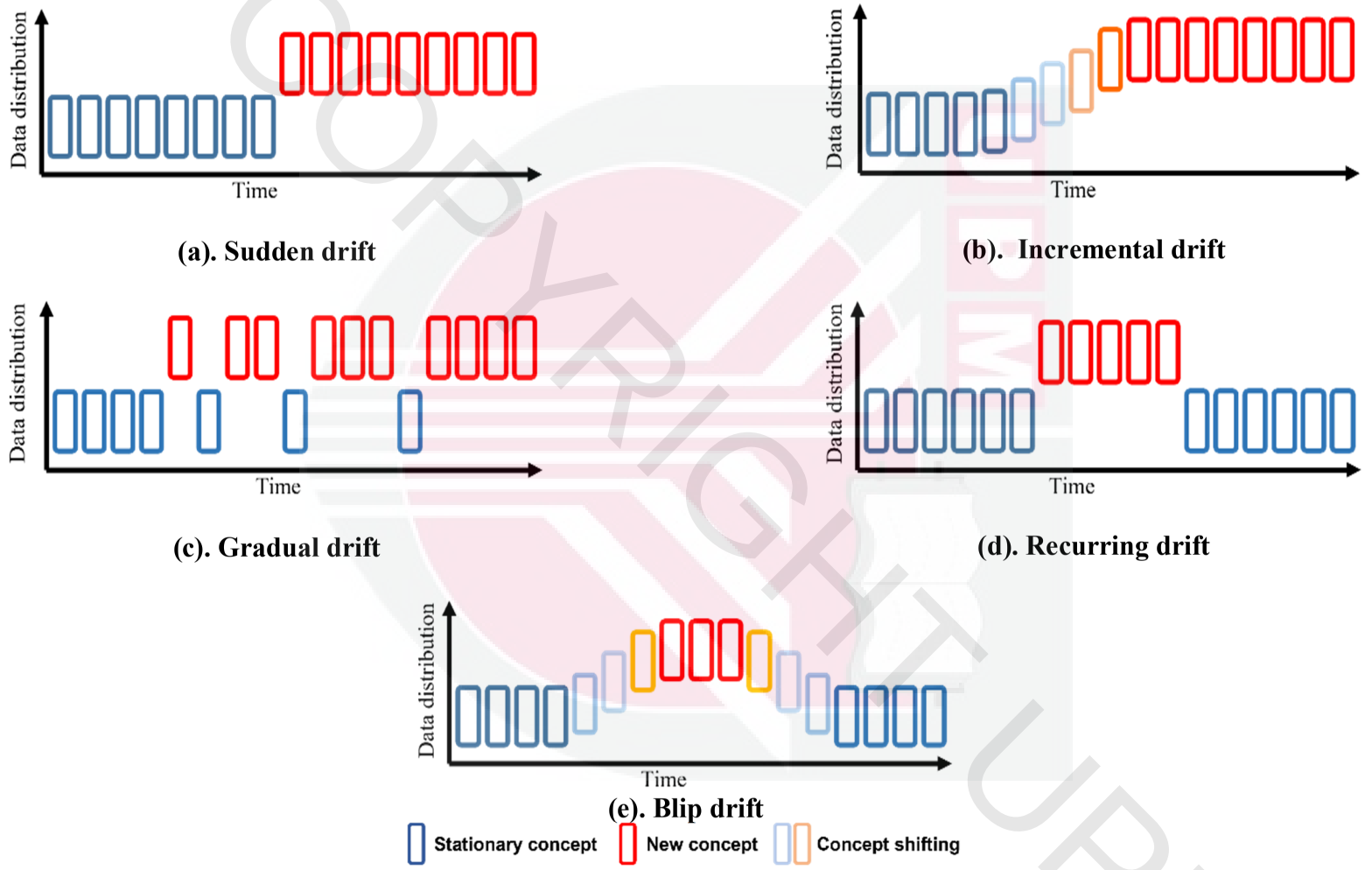


Figure 2.4: Common Types of Concept Drift [38]

In the current literature, methods to address concept drift are typically grouped into three main approaches [94]. The first, known as the error rate-based approach, focuses on tracking fluctuations in the online error rate of classification algorithms. This approach is well-illustrated by the Drift Detection Method (DDM) [46], a technique that focuses on monitoring error-based indicators to identify shifts in data distribution. The second approach, data distribution-based, measures the distances between newly arrived data and historical data using a distance function to assess their dissimilarity [60]. Lastly, the multiple hypothesis drift detection approach combines the strategies of the first two methods, either in parallel or hierarchically [95]. For our thesis, we utilized the DDM as illustrated the general diagram in Figure 2.5. DDM its proven efficacy in identifying significant shifts in data distributions. The DDM is one of the pioneering algorithms that introduced the concepts of warning and drift levels for identifying concept drift. The initial step uses a landmark time window to monitor the incoming data. As new data arrived, DDM checks if there's a notable rise in the online error rate within this window. If the change in error rate hits the warning threshold, DDM starts preparing a new predictive model, while the existing model continues to make predictions. However, if the error rate exceeds the drift threshold, the new model takes over the prediction tasks. The DDM requires a classifier to predict and thus convert the training data into a learning model, designated as the second stage, Data Modelling. The third stage involves using test statistics to establish the online error rate, while the fourth stage, the hypothesis test, determines the distribution of this error rate. This stage also sets the warning and drift thresholds.

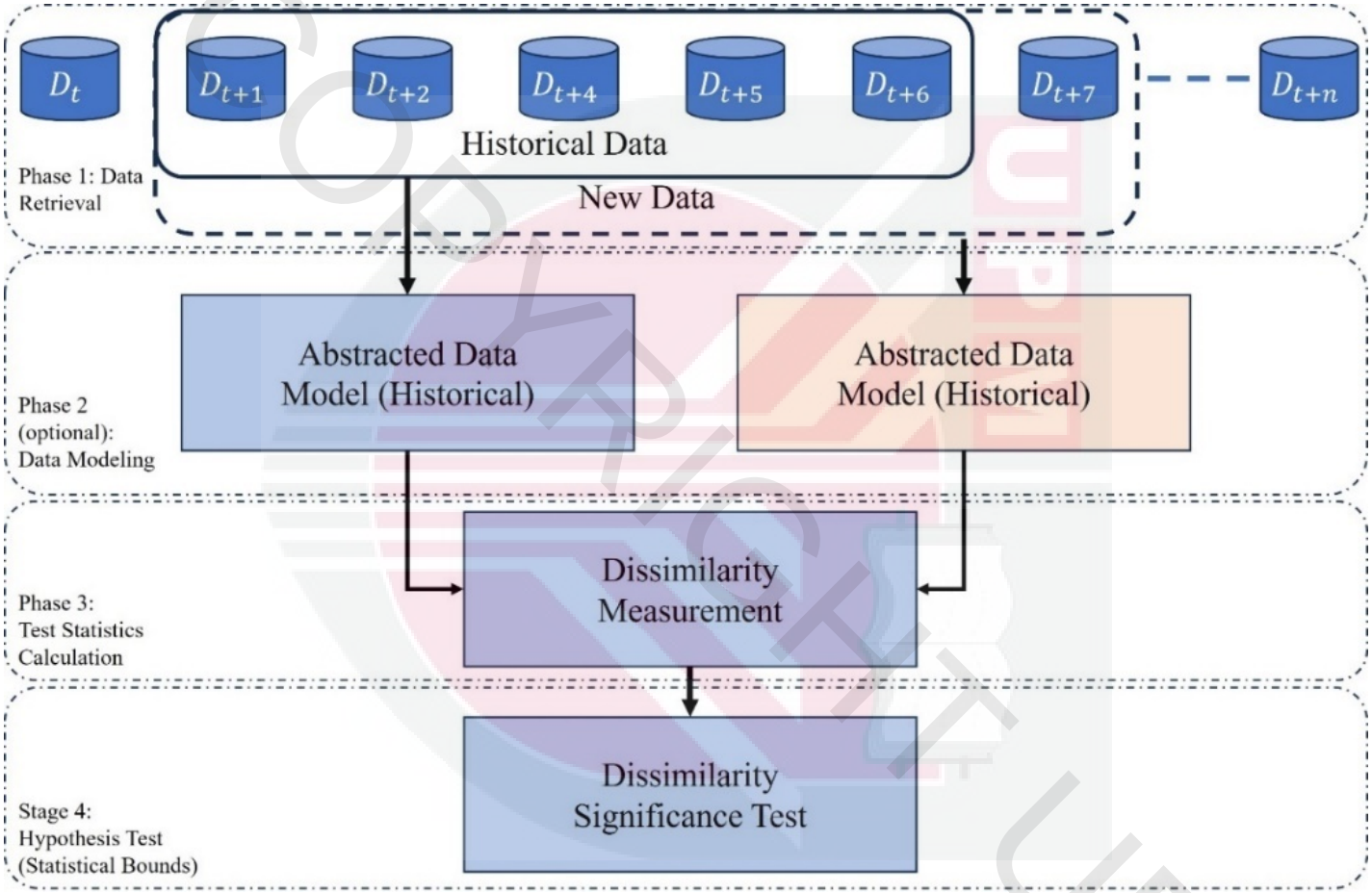


Figure 2.5: A General Framework for Drift Detection Method [94]

For each method identify in the literature with distinct strengths and limitations [94]. Error rate-based approaches, such as the DDM [96], monitor the performance of the learning model by tracking changes in the error rate over time. These methods are straightforward to implement and are effective for detecting sudden, large-scale changes in the data. However, they are sensitive to noise and may struggle to capture subtle or complex forms of drift.

Data distribution-based approaches, such as those utilizing Kullback-Leibler divergence [97], focus on detecting changes in the underlying data distribution. These methods offer a more direct measure of concept drift, providing a nuanced understanding of when and how drift occurs. While they can be more accurate, they are often computationally expensive and require careful tuning of parameters, such as window sizes, to be effective.

Multiple hypothesis test methods provide a robust detection mechanism by performing statistical tests across multiple stages, particularly useful in noisy environments. These methods improve detection accuracy but add complexity to the process, which can be a trade-off depending on the application's needs [8].

Hierarchical or parallel drift detection methods [98], which involve an initial detection followed by further validation, reduce false positives and enhance overall accuracy, albeit at the cost of increased computational complexity. For example, error rate-based methods might be ideal for scenarios with sudden drift, while data distribution-based methods could be more suitable for gradual changes affecting specific features. Multiple hypothesis tests and hierarchical methods may be preferred in noisy or highly dynamic environments where robustness is key [94].

The comparative analysis highlights that the choice of method should be driven by the specific characteristics of the data stream, the type of drift expected, and the computational resources available. Each method has its domain of applicability, and understanding these nuances is critical for effective concept drift detection [99].

Table 2.1: Comparison of Concept Drift Detection Methods: Advantages and Disadvantages

Method	Advantages	Disadvantages
Error rate-based	Ideal for scenarios where computational simplicity and quick detection of sudden drifts are priorities	Not be suitable for noisy environments or cases where gradual drifts are more common.
Data distribution-based	Suited for complex scenarios where subtle changes in the data distribution need to be monitored	Higher computational cost and complexity.
Multiple hypothesis test methods	Provide a middle ground with robust detection in noisy environments	Additional computational cost and potentially slower detection times.
Hierarchical methods	Useful in environments where both the frequency of drift and the need to minimize false positives are concerns.	Require careful management of computational resources due to their complexity.

2.4.2 Feature Drift

In data stream classification, the ultimate goal is to classify incoming data accurately into predefined classes or labels by map the relevancy of the input features to their associated classes. However, the relevancy of these features may evolve, giving rise to a phenomenon known as feature drift [11]. Feature drift, considered a specific type of concept drift, occurs when features once considered significant to the learning process become redundant or irrelevant or when there are shifts in the statistical properties of those features. Unfortunately, many learning applications, particularly in IDS, often overlook the implications of feature drift, which can gradually decrease their predictive accuracy over time [100]. To conceptualize this:

Consider a feature space denoted by $\mathcal{F}$, where $\dot{\mathrm{F}}_{\mathrm{t}} \subseteq \mathcal{F}$ represents the relevant features at time t. If for two consecutive time points e and q and observed that $\dot{\mathrm{F}}_e \neq \dot{\mathrm{F}}_q$, then it's evident that a feature drift has occurred.

Feature drift in IDS arises from several factors, such as the evolution of network protocols, modifications in network architectures, or even due to the adaptability of cyber attackers. As cyber attackers refine their techniques to bypass conventional security defenses, they introduce traffic patterns that may initially go unnoticed. For example, the ever-evolving malware attacks, where attackers consistently adapt in numbers and sophistication, occasionally mimicking legitimate traffic, could lead to feature drift. These attacks can manipulate features or labels by altering or modifying feature values. Such tactics target the feature distribution of the model through instance-based poisoning attacks. Additionally, they may attempt to change the class boundaries by employing class-based poisoning attacks, aiming to modify the conditional probability distribution $P(Y|X)$ component of the concept drift. These deliberate actions can significantly obstruct the IDS's ability to adapt to actual concept drifts, misleading the drift detector into adapting to changes that are not natural occurrences within the network environment [40].

Conclusively, addressing feature drift is a critical challenge that any data stream analysis technique [101], especially IDS, must tackle. The dynamic nature of cyber threats, coupled with the adaptability of attackers, necessitates the development of robust IDS mechanisms capable of recognizing and adapting to feature drift. Ensuring the efficacy of IDS in the face of such challenges is paramount for maintaining network security and integrity in an ever-evolving digital landscape.

2.4.3 Class Imbalance

In the field of IDS, data imbalance refers to the unequal distribution of data across various classes. Typically, normal traffic constitutes the majority class, while the minority of instances could be rare patterns or, in the context of IDS, considered intrusions. Mathematically, this imbalance can be represented by the ratio of the number of instances in the majority class to the number of instances in the minority class [102].

Ensemble learning [103] and Synthetic Minority Over-sampling Technique (SMOTE) [104] are two methodologies effectively adopted to mitigate the issue of data imbalance. Ensemble learning leverages multiple classifiers whose outputs are combined based on their performance, enhancing the robustness of the system against imbalanced datasets [43]. Meanwhile, SMOTE addresses the imbalance by generating synthetic samples for the minority class by selecting a sample of the feature space for each target class and its nearest neighbors. The method involves interpolating these samples to create new, synthetic samples of the minority class. For further clarification about SMOTE, refer to Figure 2.6, thus balancing the class distribution and improving the classifier's ability to detect rare intrusion events. This approach is preferred over random oversampling because it effectively reduces the risk of overfitting, particularly in datasets with high dimensionality. These methods are integral in refining the accuracy of intrusion detection where class distribution is skewed.

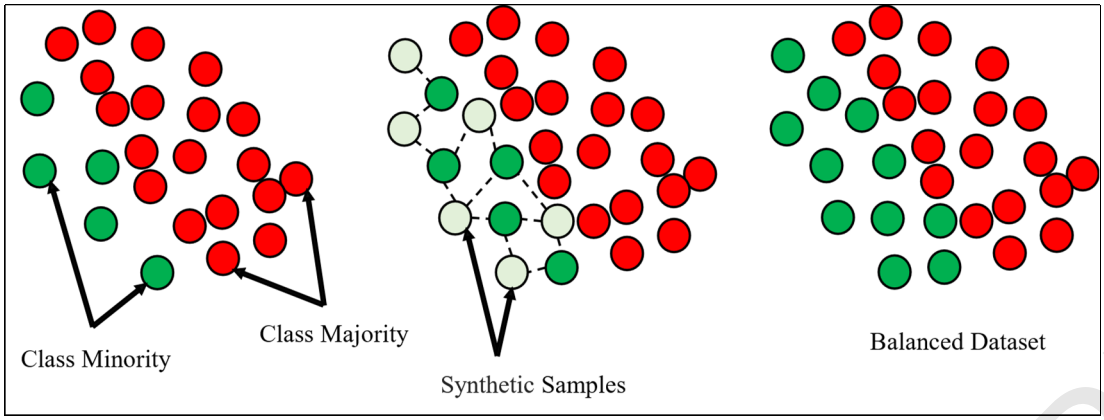


Figure 2.6: The adopted SMOTE technique to handle data imbalance [105]

The implementation of SMOTE method procedure can be described in the following points:

1. Randomly select a data point x_i from the minority class.
2. Select the k nearest neighbors from the minority class sample x_i, where $k = 5$.
3. Compute the distance between the selected sample x_i and its chosen neighbors.
4. Randomly select one of its closet's neighbors points let's assume z_i.
5. Synthetically generating samples x_{new} in the line segment between x_i and z_i.

The process iterates until a balance is achieved by blending the genuine minority class data with the synthetically created samples. The balanced dataset thus obtained serves as the foundation to train the model.

In data streams, class imbalance is common and poses a significant challenge as it can skew the performance of the IDS towards the majority class, leading to a high number of false negatives for intrusions. Hence, it is essential to incorporate strategies that effectively address class imbalance to ensure the reliability and accuracy of IDS in real-time data stream environments.

2.4.4 High-Dimensionality

IDS, when deployed in extensive network infrastructures, encounter a significant challenge known as high-dimensional data streams. Within the field of IDS, high dimensionality refers to datasets that encompass a vast number of features or attributes [106]. Managing high-dimensional data poses significant challenges due to increased complexity and the intensive computational demands associated with processing and analyzing such extensive datasets, where time efficiency becomes a critical consideration [107]. Specifically, convolutional methods in IDS may find it challenging to cope with high dimensionality, with the potential of running into NP-hard problems, particularly as the number of features grows exponentially, a situation mathematically expressed as 2^n where n denotes the number of dimensions or attributes.

The complexity of data analysis escalates in direct proportion to the number of dimensions, necessitating the adoption of more sophisticated data preprocessing techniques. Datasets expand not only in the number of dimensions, n, but also in the number of samples, m. A dataset with m samples and n dimensions is described as $n - dimensional$. Typically, the term 'high dimensional' is applied when there is a significant increase in the number of dimensions, n, giving rise to the so-called curse of dimensionality [1]. Figure 2.7 demonstrates the challenges in interpreting data within high-dimensional spaces. As the number of dimensions grows, the task of selecting the relevant features progressively becomes more complex.

Further, to elucidate the concept of high dimensionality, consider a high-dimensional data stream represented as $D = \{\langle X_1, t_1\rangle, \langle X_2, t_2\rangle, \ldots, \langle X_\infty, t_\infty\rangle\}$, where D represents an endless source of data stream tuples. Each data element $X_d = (x_d^1, \ldots, x_d^n)$ is a vector in $d - dimensions$, and each vector X is associated with a corresponding label set $F = \langle F_1, F_2, \ldots, F_d\rangle$, where d signifies the number of samples received at time t.

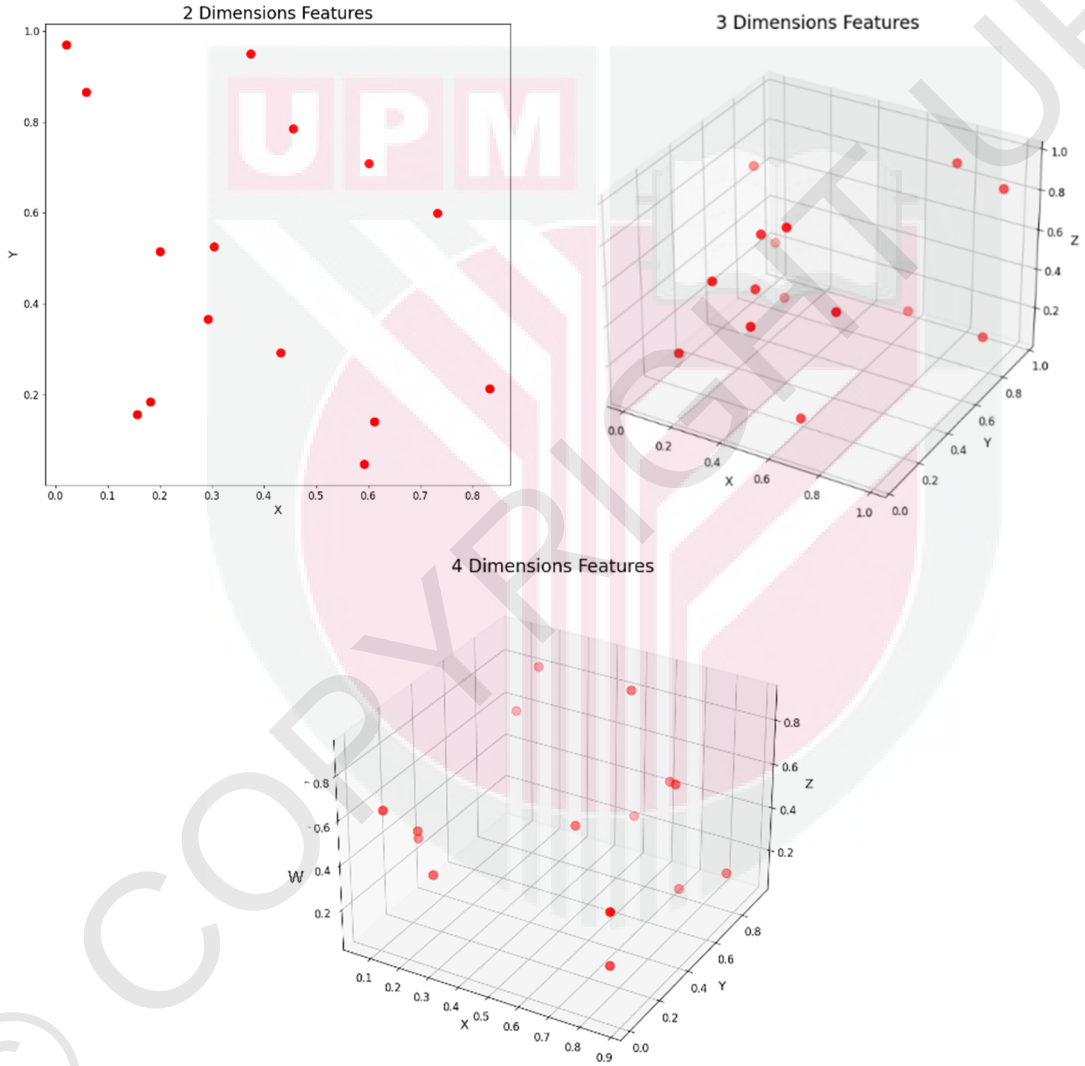


Figure 2.7: The Impact of The Number of Dimensions in The Feature Space [14]

Dimensionality reduction, a crucial process in data analysis methods, aims to reduce data dimensions by eliminating redundant and irrelevant features and presenting a

more informative dataset. It can be commonly divided into two methods: feature extraction and FS [108]. Feature extraction creates a new, lower-dimensional representation of the data by merging features from the original set. Conversely, FS techniques decrease data dimensionality by choosing a smaller subset of features directly from the original set [61]. Both methods attempt to minimize storage requirements, reduce computational complexity, and enhance model generalization. However, they differ in approach. Feature extraction can make it challenging to map a direct connection between the newly combined features and those in the original space. In contrast, FS involves directly selecting subsets of features from the original space without any transformation [109].

FS is subdivided into wrapper, filter, and embedded methods. Wrapper methods select features based on their impact on classifier performance [110]. Filter-based methods rank features by their relevance to the target concept independently of any classifier [111]. Embedded methods combine the strengths of both wrapper and filter approaches [112], integrating FS as part of the model training process.

In the literature of IDS, various studies have proposed FS methods to reduce dimensionality, such as in [113], [9], [73], [114], [115], and [62]. However, a noticeable gap exists in current research. The majority of these studies concentrate on static FS methods, which are not inherently designed to accommodate the dynamic nature of data streams. In essence, while numerous techniques have been developed to reduce feature counts in IDS, thereby facilitating more manageable computations, they often overlook the critical aspect that the significance of these features may shift over time. This oversight is particularly significant considering the dynamic and ever-changing landscape of network security threats.

2.5 Feature Selection

The exponential growth of digital networks and applications has significantly increased the volume of high-dimensional data streams, characterized by their huge volume and rapid flow[89]. This presents challenges for IDS, particularly due to memory storage constraints, processing time and limited computational capacity [90], and the need for immediate security responses [116], alongside the dynamic, complex nature of data streams that often contain redundant and irrelevant features. As a result, FS has emerged as a critical step in developing machine-learning-based IDS, aiming to ensure effectiveness and efficiency in security protocols. FS techniques prioritize the most relevant features, enhancing machine learning algorithms' performance by eliminating superfluous data, thereby minimizing overfitting risks. This process simplifies data, reduces dimensionality, and enables algorithms to deliver accurate, interpretable results more efficiently.

FS is categorized into three main types[117]: Supervised, Unsupervised, and Semi-Supervised FS, each serving distinct purposes based on the availability of labeled data. Supervised FS operates with labeled data, making it a preferred choice for tasks with well-defined outcomes, leveraging the labels to identify relevant features. Unsupervised FS, in contrast, is applied when labels are absent, focusing on uncovering the inherent structure of the dataset to find features that reveal underlying patterns such as clustering. Semi-Supervised FS, on the other hand, is utilized in situations with limited labeled data, combining the advantages of both approaches. It selects features that are significant for the target variable by using labeled data and enhances the selection with insights drawn from unlabeled data, effectively bridging the gap between supervised and unsupervised learning techniques.

The traditional methods of FS in IDS, as discussed in various studies such as in [113], [9] ,[73], [114], [115], and [62] have mainly been designed for stationary data flows. These methods typically involve using the entire dataset during the selection process, which overlooks the specific nuances associated with different instance groups. Consequently, the same attribute subset is applied across all instances, potentially limiting the effectiveness of the FS process. Additionally, these traditional approaches often rely on a single evaluation criterion to assess attribute relevance, necessitating significant effort to determine the optimal subset of features [16]. However, this approach encounters limitations in the context of feature drift in non-stationary data stream, where the relevance of features changes over time. Such evolving data environments necessitate an adaptive of FS strategies in IDS [118] [72]. These strategies must adapt to dynamic data characteristics, a challenge not sufficiently addressed by traditional methods. Therefore, the integration of adaptable FS methods using optimization algorithm, such as PSO, becomes crucial in ensuring that IDS remain effective against the constantly evolving landscape of network security threats. PSO in FS for IDS is highly effective due to its ability to navigate large search spaces and adapt to various data types [119]. Its balance of exploration and exploitation helps identify optimal feature combinations, reducing complexity and computation time. Integrating MOO with PSO is particularly beneficial in IDS, addressing simultaneous goals such as maximizing detection accuracy and minimizing features numbers. This blend is crucial in IDS, where optimizing performance without overloading computational resources is essential, making MOO within PSO frameworks ideal for the complex demands of IDS environments. [120].

2.5.1 Random Forest Feature Selection

Random Forest (RF) FS uses the RF algorithm to rank the importance of features in a dataset. It builds multiple decision trees and evaluates each feature based on its contribution to reducing impurity in these trees. This method is effective for large, high-dimensional datasets and helps in identifying the most significant features [121].

2.5.2 Logistic Regression Feature Selection

The Logistic Regression (LR) FS method utilizes Logistic LR for binary classification tasks, leveraging the magnitude of the model's coefficients to determine the importance of features. In this approach, features associated with larger coefficients are deemed to have a more significant influence on the prediction outcome. This method offers a clear and interpretable means of pinpointing crucial predictors within the dataset [122].

2.5.3 Recursive Feature Elimination Feature Selection

The Recursive Feature Elimination (RFE) systematically removes the least important features from a model to identify the most relevant ones. It starts with all features, eliminates the least significant ones iteratively, and retrains the model until only the most influential features remain. RFE is versatile and enhances model performance by focusing on essential features [123].

2.6 Multi-Objective Optimization

MOO is a form of optimization that involves making trade-offs between two or more conflicting objectives. It is a decision-making process focused on finding optimal solutions while considering the balance between these conflicting objectives, acknowledging that improvement in one objective may lead to a compromise in another [124]. Mathematically, MOO problems are characterized by multiple objective functions that need to be minimized or maximized, with the aim of finding a set of Pareto-optimal solutions. The Pareto front represents a set of optimal solutions that are not dominated by any other solution in terms of all the objectives. An instance of trade-off between two objectives resulting in a set of non-dominated solutions as presented in Figure 2.8.

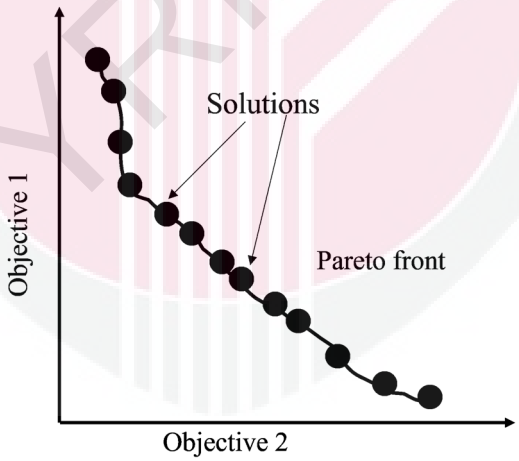


Figure 2.8: Pareto-Front in MOO Represent Trade-off Between Two Objectives [120]

The mathematical formulation of a MOO problem is typically presented as follows [125]:

Minimize

$$F(x) = [f_1(x), f_2(x), \dots, f_k(x)] \quad (2.3)$$

Subject to

$$g_i(x) \leq 0, \qquad i = 1,2,3, \dots, m \quad (2.4)$$

$$h_i(x) = 0, \; j = 1,2,3, \dots, l \quad (2.5)$$

Where x represents the vector of decision variable of $f_i(x)$ is the objective function and k is number of objectives to be optimized. $g_i(x)$ and $h_j(x)$ are the constrained function. The effectiveness of a MOO algorithm is gauged by maintains the balance among the conflicting objectives.

For example, the problem is to minimize the objective i by balancing between two conflicted goals, where e and f , if the conditions achieved, conclude that, e dominates f as follows [126]:

$$\forall i: f_i(e) \leq f_i\,(f) and\; \exists j :\; f_i(e) < f_i(f) \quad (2.6)$$

Furthermore, FS in MOO algorithms is an approach to FS that considers multiple conflicting objectives simultaneously. In this approach, the goal is to find a trade-off between multiple conflicting objectives, For example, a MOO algorithm could be used to find a set of features that maximize accuracy while minimizing the number of features [48].

2.6.1 Multi Objective Particle Swarm Optimization (MOPSO)

The MOPSO an advances the PSO [127] heuristic, initially inspired by the social behavior observed in bird flocking or fish schooling, to address the complexities of MOO challenges. This method capitalizes on the principles of Pareto dominance to navigate the search space, navigation the velocity of each particle in the swarm. Integral to the MOPSO algorithm is the establishment of a global repository, or an external archive, which retains previously discovered nondominated vectors. This archive is pivotal, functioning as a collective memory that informs the swarm's navigational decisions by cataloging the most successful solutions encountered. Contributions to this repository are made by individual particles following each iteration of their search, ensuring continuous refinement of the solution space. The repository's update mechanism is refined and methodical, taking into account a geographically inspired system that is delineated by the objective function evaluations of the particles, thereby ensuring a comprehensive and diverse exploration of potential solutions [128]. The MOPSO algorithm operates as follows:

1. Initialization: The algorithm begins by initializing a population of particles with random positions and velocities.
2. Evaluation: The fitness of each particle is evaluated based on the objectives of the problem. The best position encountered by each particle (personal best) is recorded.
3. Repository Update: A repository of non-dominated solutions (best solutions found so far) is maintained. It is updated regularly with new non-dominated solutions from the particle evaluations.
4. Leader Selection: In each iteration, leaders are selected from the repository to guide the swarm. This selection is crucial for directing the search process and maintaining diversity in solutions.
5. Particle Update: Particles update their velocities in Equation (2.7) and positions based on their personal bests and the leaders.

6. Loop: Steps 2 to 5 are repeated iteratively. In each iteration, particles explore the search space, guided by their own experience and the influence of the leaders.

7. Convergence Check: The algorithm checks for convergence criteria, such as a maximum number of iterations or satisfactory fitness level. Once convergence criteria are met, the algorithm terminates.

During operation, in particle update step computes their velocity using an Equation (2.7) [21], that includes the particle's best-known position, the position from the repository that guides the particle, and random factors that ensure exploration and exploitation in the search space. Particles then update their positions based on the new velocity. The search space is continuously updated with currently nondominated locations, while dominated locations are removed from the repository. There is also a mechanism to prioritize particles in less crowded areas of the objective space, promoting diversity.

$$Vel_{(i)} = W \times Vel_{(i)} + r_1 \times Pbest_i - Pop_{(i)} + r_2 \times \left(Rep_{(h)} - Pop_{(i)}\right) \quad (2.7)$$

Where, $Vel_{(i)}$ is the velocity of particle i, and W is the inertia weight, r_1 and r_2 are random numbers within [0, 1], $Pbest_i$ is the best position particle i has achieved, $Rep_{(h)}$ is a position taken from the archive, $Pop_{(i)}$ is the current position of particle i. A key component of updating a particle's position is the application of Pareto dominance. If a particle's current position is better than the one in its memory (i.e., it is not dominated by the position in memory), then the current position is updated; otherwise, the position in memory is retained. If neither position dominates the other, one is selected randomly. MOPSO represents an effective approach to handling MOO problems by incorporating concepts of Pareto dominance and utilizing a memory repository to guide the search process. It is important to note that while PSO is

inherently an unconstrained search technique, mechanisms for dealing with constrained MOO problems are being developed.

In this thesis, the integration of several advanced techniques within the MOPSO algorithm plays a pivotal role in enhancing its performance, particularly for online decision-making environments. The Multi-Exemplar MOPSO introduces an innovative approach where each particle learns from multiple exemplars, specifically from various non-dominated solutions within the archive, rather than relying on a single reference point. This strategy enhances the diversity of the swarm and improves convergence by enabling a more comprehensive exploration of the search space [26].

Additionally, the variable-length feature representation in MOPSO technique involves a variable-length features representation, allowing the algorithm to effectively handle high-dimensional spaces by dividing the feature space based on the importance of features. This method focuses the search within relevant subspaces, effectively balancing the trade-off between exploration and exploitation, which is crucial for maintaining diversity while ensuring convergence towards optimal solutions [23].

The Division-Based MOPSO technique further enhances the algorithm's capability by dividing the feature space based on the importance of features, which facilitates a more focused search within relevant subspaces. This method effectively balances the trade-off between exploration and exploitation, crucial for maintaining diversity while ensuring convergence towards optimal solutions [55].

Finally, an Improved Initialization Phase is incorporated to ensure that the initial population is not only diverse but also strategically positioned within the search space, which significantly enhances the algorithm's overall performance. This step is essential

for avoiding premature convergence and ensuring a robust exploration of potential solutions from the very beginning [129].

2.6.2 Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

The NSGA-II is a popular evolutionary algorithm used for solving MOO problems [130]. Specifically tailored to handle the complexities associated with multiple, often conflicting objectives. NSGA-II operates by generating a population of potential solutions and iteratively evolving them through the processes of selection, crossover, and mutation. The algorithm employs a fast non-dominated sorting approach to classify the solutions into different levels of non-domination. Solutions in the first front are considered the best as they are not dominated by any other solution, with subsequent fronts representing progressively dominated solutions. NSGA-II also incorporates a crowding distance mechanism, which ensures diversity in the population by preserving individuals that provide widespread coverage of the Pareto front.

The chose NSGA-II for the comparative analysis because it is a well-established benchmark in the multi-objective optimization field, renowned for its efficient exploration and exploitation capabilities. Its consistent performance and robustness across various applications make it a reliable standard for evaluating the proposed algorithms.

2.6.3 The Non-Dominated Sorting Genetic Algorithm III (NSGA-III)

The NSGA-III is an advanced evolutionary algorithm designed to solve MOO problems [131]. Building on the strengths of NSGA-II, this algorithm introduces a

reference-point-based approach for maintaining diversity in a higher-dimensional objective space. NSGA-III begins with a population of candidate solutions and iteratively improves them using genetic operators like crossover and mutation. It employs a non-dominated sorting mechanism to categorize solutions into various levels based on dominance.

A key feature of NSGA-III is its use of a set of predefined reference points that are systematically distributed in the objective space. These reference points guide the selection process to ensure a diverse and well-distributed Pareto front, even in many-objective spaces.

the chose NSGA-II for the comparative analysis because it is a well-established benchmark in the multi-objective optimization.

The comparative analysis to evaluate the performance of MOPSO [120] and NSGA-II [130] within the context of IDS and real-world applications. Both algorithms are widely used in MOO, but they offer distinct advantages and limitations when applied to the dynamic and complex environments typical of real-world applications. This analysis highlights their respective strengths and weaknesses [120] [132], providing insights into their suitability for real-world IDS scenarios, refer to the Table 2.2.

Table 2.2: Comparative Analysis for MOSPO and NSGA-II Advantages and Disadvantages

Aspect	**MOPSO** [120]	**NSGA-II** [132]
Convergence Speed	MOPSO generally converges faster due to its simpler structure and fewer parameters and required few iterations to find the optimal solutions.	NSGA-II may converge slower due to its complex operations like crossover and mutation. slower convergence in real-time applications where quick responses is critical such as in IDS.
Exploration vs Exploitation	In high dimensions decision space MOPSO may suffer from premature convergence due to insufficient exploration.	NSGA-II maintains a better balance between exploration and exploitation due to its use of genetic diversity mechanisms and crowding distance.
Diversity Maintenance	MOPSO maintaining diversity in the Pareto front through utilizing an external archive accumulating the solutions over iterations. Where its better in real time applications when required to accumulating the knowledge over time	NSGA-II uses crowding distance to maintain diversity among solutions on the Pareto front.
Parameter Sensitivity	MOPSO has fewer parameters, making it easier to tune and implementing.	NSGA-II requires careful tuning of multiple parameters, such as mutation and crossover rates, to achieve optimal performance.
Scalability	MOPSO may not scale well with large and complex problems. Further techniques required to incorporate to solve this.	NSGA-II has been shown to scale better with problem complexity and size, particularly in large-scale applications.
Computational Efficiency	MOPSO is computationally more efficient due to its simpler operations. So, it's more desirable for real time limited constrain resources such as in IDS.	NSGA-II can be computationally expensive due to the need for sorting and maintaining diversity in large populations and high number for iterations. Make it more applicable in offline applications and availability of recourses.

In real-time applications like IDS, where quick response and computational efficiency are paramount, MOPSO emerges as the more suitable algorithm. It's faster convergence, ease of implementation, and lower computational demands make it ideal for environments with limited resources and the need for rapid decision-making. While NSGA-II offers advantages in exploration and scalability, its slower convergence and higher computational requirements make it better suited for offline applications or scenarios where a more thorough exploration of the solution space is needed.

2.6.4 Multi-Objective Evolutionary Algorithm (MOEA)

The algorithm introduced in 2007 by [133], the MOEA has gained popularity as an effective method for addressing MOO challenges. This algorithm approaches a multi-objective problem by breaking it down into numerous scalar subproblems that are solved concurrently through a cooperative search approach. This strategy promotes both efficient exploration of the solution space and convergence towards a comprehensive set of Pareto-optimal solutions. Additionally, the MOEA incorporates a neighborhood-based search technique, which enhances information exchange between subproblems, thereby aiding in faster convergence. Due to these features, the MOEA has been widely and successfully implemented in a variety of real-world optimization scenarios, showcasing its adaptability and efficiency in multi-objective problem-solving.

2.7 Static and Dynamic Feature Selection

In exploring the literature of FS techniques for IDS, it's crucial to distinguish between static and dynamic methods. Most FS techniques in the current literature are static for

three primary reasons [16]. Firstly, many authors apply FS to the entire dataset, failing to account the relevancy of features for each group of instances that may change over time. This oversight is particularly problematic in data streams, where the relevance of features can shift over time from relevant to irrelevant status and vice versa, necessitating an automated selection method for each arriving group of instances. Secondly, static methods often utilize a single evaluation criterion to measure feature relevance. This approach may demand significant effort and may not always yield the most suitable criterion for a dataset. Lastly, the inherent non-stationary nature of data streams, often characterized by phenomena feature drift, requires classifiers to periodically adjust to the changes in newly arriving data groups [100]. This highlights the growing necessity for DFS methods in IDS that can adapt to evolving data characteristics.

To clarify the distinction between static and DFS, static feature selection is the traditional approach where the FS process is performed once on the entire dataset. The selected features are then used to train a machine learning model, which is then used to make predictions on the new data. This approach assumes that the statistical properties of the features remain constant over time [134]. In contrast, in DFS, the feature selection process is performed continuously over each group of instances that arrive. This allows the FS to adapt to changes in the features importance over time [16], which can result in improved performance for models trained on dynamic data [100].

DFS represents the process of selecting the relevant features at a certain point in a certain time interval. More formally, assuming that DFS is a process of FS, and $D =$

$\{(x_t, y_t)\}$ is a stream data, then $DFS_{t_1,t_2}(D) = \mathbb{F}$ where $\mathbb{F} \subseteq$ X and $\mathbb{F}$ provide the most relevant features in X throughout t_1, t_2 considering that $t_2 > t_1$ and $t_2 - t_1 = \Delta$ represents the least period that does not involve feature changings in the data stream.

The evaluation of FS techniques reveals a significant shift from traditional static methods to dynamic approaches. Static methods, while useful in stationary environments, fail to adapt to the evolving nature of data streams, leading to potential inaccuracies in feature relevance over time. DFS, in contrast, offers continuous adaptation to the changing importance of features, enhancing the performance of models trained on dynamic data. This adaptability of DFS is paramount in addressing the challenges posed by feature drift in data streams, positioning it as an essential tool in the development of robust and responsive IDS systems in the face of rapidly changing network security landscapes.

2.8 Fix vs Variable Length Optimization

Fixed optimization and Variable-Length Optimization (VLO) are two pivotal concepts in optimization theory. In fixed optimization focuses on minimizing a function within a fixed-dimensional space $\mathbb{R}^n$, where solution size remains constant throughout the process. In contrast, VLO allows solutions of varying dimensions, suitable for scenarios where the ideal solution size may change. This type of optimization adapts to a broader range of problems by permitting solutions of different lengths, making it ideal for complex scenarios where the optimal solution structure is not known beforehand. The flexibility of VLO is especially valuable in dynamic environments where solution requirements may evolve [23].

In the optimization process, minimize $f(x,t) = \{f_1(x,t), f_2(x,t), \dots, f_m(x,t)\}$ where always $x \in \mathbb{R}^n$, then the optimization is termed a fixed optimization problem. Conversely, when $x \in X$, with $X \subseteq \bigcup_{i=1}^{n} \mathbb{R}^i$, the optimization is termed as variable-length optimization.

The adoption of VLO in this thesis is identified as crucial, particularly with its focus on non-stationary data streams where feature relevance and concepts may shift unpredictably. This optimization technique is highlighted for providing the flexibility necessary for adapting to these arbitrary changes in data characteristics. By permitting solutions of varying dimensions, VLO facilitates continuous adjustments, ensuring the research method remains effective in dynamic environments. Its adaptability is deemed essential for addressing the complexities of non-stationary data streams, maintaining the research's relevance and efficiency amidst the fluctuating nature of the data [55].

2.9 Transfer Functions for Binary Searching

In the domain of data mining and machine learning, MOO using PSO is gaining importance, especially for FS tasks. The importance of FS lies in its dual objectives: enhancing classification accuracy by retaining the most relevant features, and enhancing the memory consumption by removing redundant or irrelevant features, thereby reducing computational complexity. Recognizing FS as a binary problem, each feature is represented as a binary variable within a discrete binary search space, where '0' denotes an excluded feature and '1' an included feature, aligning with the feature's dimension in the dataset [120].

Transfer functions play a pivotal role in converting continuous PSO to a binary version suitable for FS, defining the probability of feature inclusion or exclusion. V-shaped and S-shaped transfer functions, in particular, have been instrumental in improving PSO search efficacy by modulating this probability with their respective mathematical characteristics. Transfer functions are integral to the binary variant of PSO, enhancing its utility in FS by improving classification accuracy and the efficiency of selecting features. Their incorporation into PSO is a testament to their adaptability, offering a refined approach to optimization that advances beyond traditional methods [135] [57].

Furthermore, the choice between S-shaped and V-shaped transfer functions influences the search behavior of the PSO, with S-shaped functions generally providing a gradual probability change suitable for exploration, and V-shaped functions offering a more abrupt probability change conducive to exploitation, thereby substantiating their potential to fine-tune the FS process to achieve high accuracy and memory efficiency [27]. The use of transfer functions in converting continuous PSO to its binary counterpart for FS is a significant development. These functions, especially the V-shaped and S-shaped types, play a crucial role in determining the probability of feature inclusion or exclusion. Their effectiveness in enhances PSO's search efficiency, balancing exploration and exploitation. This advancement in PSO application for FS tasks in data mining and machine learning demonstrates a promising path towards achieving higher classification accuracy and efficient memory usage in complex data environments.

2.10 Ensemble Learning Based Intrusion Detection

The success of an IDS depends on its ability to achieve high detection accuracy in classify the attacks, maintain a low false rate, and adapt to diverse environments. Achieving these two objectives in a singular trained classifier remains challenging [136]. Therefore, ensemble-based intrusion detection approaches offer a promising solution by integrating multiple classifiers into a unified framework to produce a strong classifier [137] and blends the results of each classifier using various methods (such as average, maximum, median, voting majority and others) [73] to achieve a high detection rate (DR) and a low false positive rate (FPR) [31]. For more clarification refer to Figure 2.9.

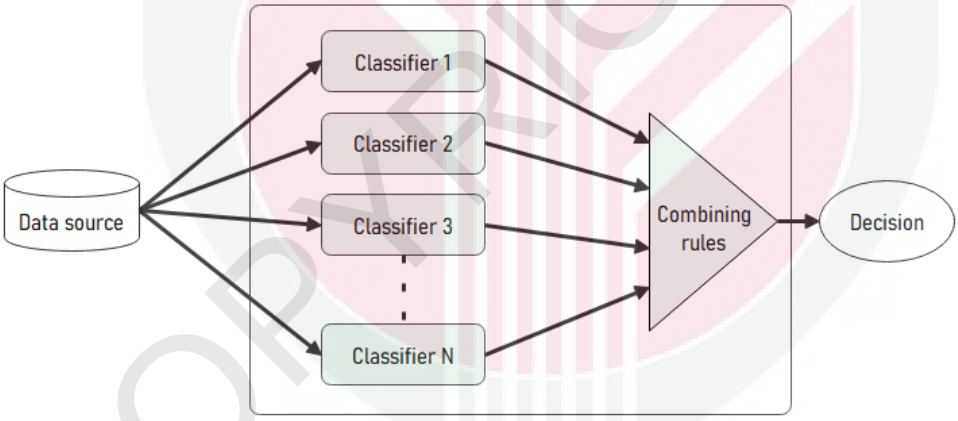


Figure 2.9: General Diagram of Ensemble Classifier [33]

Ensemble classifiers typically function within a tree-like architecture, where each 'leaf' or node represents a classifier whose decision contributes to the 'branches' or aggregated output [116]. By integrating diverse predictive models, ensemble classifiers benefit from the "wisdom of the crowd" effect, where the collective judgments result in superior accuracy and resilience against errors than any single

classifier could achieve [138]. This approach is especially valuable in IDS, where the dynamic landscape of cybersecurity threats requires robust and flexible solutions to stay ahead of malicious actors.

Several reasons behind choosing the ensemble learning for this research can be summarized as: Ensemble learning is particularly effective in addressing class imbalance issues, which is a common challenge in real-world datasets. Single classifier methods often struggle losing important information by bias toward the majority class, but ensemble methods overcome these challenges by combining multiple classifiers. This approach ensures that the minority class is better represented, leading to more robust and stable predictions [43]. Moreover, ensemble learning is highly advantageous in non-stationary environments where the data distribution may change over time. Unlike single classifiers, which may struggle to adapt quickly to these changes by retrain or reparametrize, ensemble methods can continuously adapt by replacing or updating classifiers within the ensemble as new data comes in. Furthermore, the ensemble approach inherently improves predictive accuracy by combining the strengths of multiple classifiers. Each classifier in the ensemble may capture different aspects of the data, and their combined decision-making process typically results in more accurate and reliable predictions than any single classifier could achieve. Ensemble methods also offer greater flexibility and scalability compared to single classifiers. They can be adapted to various data stream scenarios, whether the data is stationary or non-stationary. The ability to adjust how classifiers are updated, replaced, or combined allows the ensemble to remain effective across different datasets and problem domains. Additionally, the success of ensemble methods often hinges on the diversity and complementarity of the classifiers in the

ensemble. An ideal ensemble consists of classifiers that are diverse yet complementary, meaning that they make different errors and, therefore, can correct each other when combined. This increases the likelihood that the ensemble will perform well on various parts of the data, leading to overall better performance than a single classifier that might be more prone to specific errors [33].

2.11 Genetic Programming Combiner

A vital component in constructing an ensemble classification model is choosing the combiner function, which merges the predictions of base classifiers. Adaptive, data-driven combiners have been found to outperform standard static aggregation methods like average predictions or weighted voting. Nonetheless, for dynamic intrusion detection data, this method may be impractical due to the potential need for frequent retraining of parts of the ensemble [116].

Ensemble classifiers often employ a combiner to aggregate the predictions from multiple classifiers literature review for the recent studies illustrated in Table 2.1. Genetic combiners, utilizing genetic algorithms, have gained prominence in data stream classification. While conventional combiners might struggle with the dynamic nature of data streams, where majority voting or averaging might not be agile enough to adapt to these changes, especially if the base classifiers are trained on outdated data. Older classifiers in the ensemble can potentially provide incorrect votes, leading to an overall wrong decision [139], while in genetic combiners adaptively merge classifier decisions, making them a promising solution to address the inherent challenges of continuously evolving data streams [32].

The GP combiner is generated through a GP-based evolutionary computation scheme, focusing on the optimization of ensemble classifier predictions. Each candidate GP tree, representing a potential combiner function, is structured with leaves corresponding to base classifiers and internal nodes linked to non-trainable aggregation functions. The module limits the complexity of these GP trees to manage the search space efficiently. The fitness of each GP tree is assessed based on prediction accuracy over a validation set. This fitness calculation employs a weighted scheme for misclassified tuples, particularly useful in unbalanced datasets common in intrusion detection scenarios. The unique aspect of this GP combiner is its ability to adaptively combine classifier outputs, enhancing the overall accuracy and responsiveness of the IDS in dynamically changing data environments. The review of GP-combiner literature in ensemble classifiers, as presented in Table 2.3, highlights a important gap, there is a critical need for methodologies that address feature drift while incorporating strategies for handling high-dimensional data, DFS, and MOO to improve ensemble classifiers in dynamic data environments.

Table 2.3: Literature review of methods for ensemble classifiers utilizing GP-combiner

Reference	Field	High dimension	DFS	MOO	Feature drift	Concept drift	Imbalance	Year	Type
[140]	Network IDS	×	×	×	×	√	√	2023	Article
[141]	Data stream	×	×	×	×	√	√	2023	Article
[142]	Data stream	×	×	×	×	√	×	2021	Article
[143]	Data stream	×	×	×	×	√	×	2020	Article
[144]	Data stream	×	×	×	×	√	√	2020	conference
[145]	Data stream	×	×	×	×	√	×	2019	Article
[146]	Cyber Security	×	×	×	×	√	√	2016	conference
Proposed method	Data stream	√	√	√	√	√	√	-	-

The GP Combiner represents a significant advancement in the field of ensemble classification, particularly suited for non-stationary data streams. Its adaptive nature ensures that classifiers within an ensemble can effectively respond to concept drifts, maintaining high levels of accuracy and robustness in dynamic data environments.

2.11.1 k-Nearest Neighbors (k-NN)

The Nearest Neighbor (NN) classification is a type of supervised machine learning algorithm. It uses NN search methodologies to tackle classification challenges. It operates by comparing a new, unclassified sample (query) to a training dataset of pre-labelled samples, typically situated in a metric space. The NN search method identifies the closest sample in the training data to the query, and the class label of this nearest neighbor is then assigned to the query. The NN algorithm uses distance metrics, like Euclidean or Manhattan distance, to determine the similarity between samples. This method is known for its simplicity and effectiveness in various classification challenges[147]. In the context of classification [148]:

$$x_{nn} = arg\ min_{x_i \in D} d(x, x_i) \tag{2.8}$$

$$y = y_{nn} \tag{2.9}$$

Where x is the sample that attend to classify, D refer to the training dataset, x_i represents the data sample in D. $d(x, x_i)$ is to calculate the distance between x and every x_i samples. By minimize the distance y is the label that will be assigned to the query point x, where y_{nn} is the label of the nearest neighbor to the query point $\boldsymbol{x}$.

2.11.2 Logistic Regression

Logistic regression is a supervised learning used for binary classification problems and even for multi class classification. Logistic regression is a simple and more efficient method for binary and linear classification problems. Logistic regression's range is bounded between 0 and 1. In the context of binary classification problems, if the output is above a specified threshold (commonly 0.5), the predicted class label might be set to 1, otherwise 0. One of the key advantages of logistic regression is its interpretability; the coefficients of the independent variables can be used to ascertain the relative influence of each predictor on the outcome. Moreover, logistic regression can be extended to handle multi-class classification tasks through techniques like one-vs-all [149].

$$Logistic\ function = \frac{1}{1+e^{-x}} \tag{2.10}$$

Where x is the input to the function, which is the linear combination of the features (independent variables) and their weights (coefficients), represented as:

$$x = \beta_0 + \beta_1 x_1 + \cdots . + \beta_n x_n \tag{2.11}$$

β_0 is the bias, and $\beta_1, \beta_2, \ldots . , \beta_n$, are the weights or coefficients of the features $x_1, x_2, \ldots x_n$.

2.11.3 Decision Tree

Decision tree also known as classification tree as shown in the Figure 2.10, is a supervised learning algorithm used to predict a target variable based on several input

features. A decision tree is visually represented as tree structure. It comprises nodes that represent the features, branches that represent the decision rules, and leaves that represent the outcome or target variable or class label. The root of the tree refers as the starting point that later expand to tree to form on the bias of subtrees to answer the questions. Decision tree well known with easy in interpretability and can capture non-linear relationships between features and target [150].

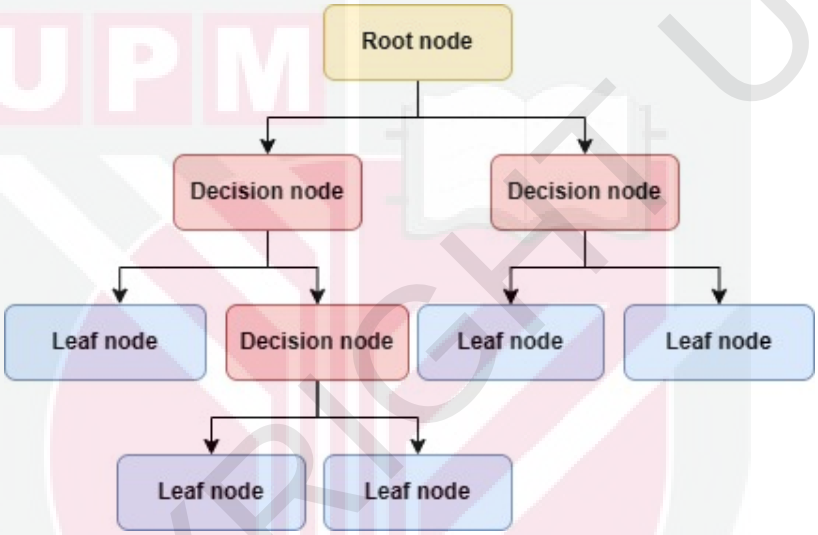


Figure 2.10: The General Diagram of Decision Tree [150]

2.11.4 Gaussian Naive Bayes

The GaussianNB It's a specific type of Naive Bayes classifier that assumes each feature is normally distributed within each class. In the case of the Gaussian Naive Bayes, it is assumed that the continuous values associated with each class are distributed according to a Gaussian (normal) distribution. The advantages of the Gaussian NB can be summarized as simple and easy to implement, quickly training and performs well with high-dimensional datasets.

The likelihood of the features is assumed to be Gaussian, hence, the likelihood of a feature given a class $P = (x_i \mid y)$ can be determined using the following equation [151]:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}} \quad (2.12)$$

Where, μ_y is the mean of the features of class y, and σ_y^2 is the variance of the features of the class y.

2.12 Literature Survey

In this section, conducts a comprehensive literature survey encompassing three pivotal domains: IDSs for Data Streams, MOPSO for solving complex problems, and MOO algorithms for feature selection. The review begins by exploring the dynamic challenges and advanced methodologies in IDSs, emphasizing the need for real-time data stream handling and adaptability. It then transitions to examining MOPSO, delving into its evolution and application in complex optimization, multifaceted problem-solving scenarios. Finally, the section focuses on the critical role of MOO algorithms in feature selection, highlighting their impact in enhancing IDS performance through optimal feature identification from high-dimensional data. This unified overview aims to bridge the gaps and pinpoint future research directions in these rapidly evolving fields.

2.12.1 Intrusion Detection System for Data stream

This section provides a literature review on IDS within data streams, as outlined in the Table 2.4. It focuses on key challenges such as feature drift, dimensionality reduction, concept drift, data imbalance, and specific dataset utilized. The review critically evaluates a variety of methodologies applied in data stream environments. This literature analysis has been pivotal in pinpointing existing gaps in the field.

Table 2.4: Literature survey of IDS for non-stationary data streams

Article	Approach	Feature drift	Dimensionality Reduction	Concept drift	Data Imbalance	Dataset
[29]	SVM-based Parameters Optimization Algorithm	×	×	√	×	Network flow traffic
[9]	Principal Component Analysis (PCA)	×	√	√	×	IoT dataset
[44]	Long Short-Term Memory	×	×	√	√	Real data, no details
[45]	Ensemble Of Varying Deep Neural Network	×	√	√	×	DS2OS
[90]	Principal Component Analysis (PCA)	×	√	√	×	Network flow traffic
[46]	Ensemble of classifiers based on self-training online learning and dynamic classifier selection	×	×	√	×	LUXEMBOURG/ KDDCUP99/ Generators
[32]	Ensemble framework based on genetic program and CAGE Tool with supporting stream	×	×	√	√	Hyperplane / ISCX IDS datasets
[152]	Multiple algorithms to make combined detection, confidence-guided anomaly detection model for anti-concept drift in dynamic logs	×	×	√	√	HDFS benchmark dataset, Oil industry
[153]	K-means clustering for data imbalance and dimensionality, RF and LR classifiers at level 0 to handle concept drift, and SVM used at level 1	×	√	√	√	NSL-KDD/ CIDDS-2017 dataset/Testbed dataset
[154]	Combine four classifiers for FS and Ensemble learning IDS	×	√	√	√	AWID/ NSL-KDD
[140]	Ensemble learning based on the Genetic Programming Combiner	×	×	√	√	KDD Cup '99/ CICIDS-2017/ CSE-CIC-IDS-2018/ ISCX2012
[155]	Online Sequential Extreme Learning Machine (OSELM)	×	√	√	×	KDD-99 / NSL-KDD-99
[30]	SVM classification and K-Means clustering	×	√	√	√	NSL-KDD/IDDS-2017
[156]	Density-aware and Feature-deviated Active Intrusion Detection and various classifiers	×	×	√	×	CIC-IDS2017 and ISCX-2012
[157]	Convolutional Neural Network (CNN), a Long Short-Term Memory (LSTM) network, and a self-attention mechanism network	×	√	×	×	CSE-CIC-IDS2018/CIC-IDS2017/ UNSW-NB15
[158]	Node-based feature extraction and unsupervised deep learning and supervised multi-class classifier	×	×	×	√	Traffic data IoT devices
[119]	Random forest and Particle Swarm Optimization (PSO) for feature selection and multiple classifiers	×	√	×	×	NSL-KDD
Proposed method	Particle Swarm Optimization FS based GP ensemble classifier	√	√	√	√	IoT/Network traffic

The table produce a numerous studies of an advance intrusion detection methods. A notable example is the study referenced as [158], which addressed intrusion detection in IoT dataflows with limited hardware constraints. This study introduced a machine learning-driven multi-class classifier that can identify six types of attacks as well as benign traffic. Its approach incorporated node-based feature extraction and detection techniques, enabling the pinpointing of attacker locations through the analysis of traffic characteristics across a sliding time window. Another study in [119], tackled the issue of FS in the presence of noisy data in data streams. This was achieved through FS using a RF algorithm, thereby enhancing the efficiency of the intrusion detection process. A comparative evaluation using various classifiers such as k-NN, SVM, LR, DT, and NB was performed to gauge IDS metrics. The application of PSO to the FS led to a reduction in false alarm rates and improvements in detection rate.

The study in [157] developed a network intrusion detection method using a hierarchical model and attention mechanism to enhance feature representation and classification accuracy in complex networks. The research [156] introduced an active detection framework countering new attack classes and concept drift with mask density scores and robust clustering to improve accuracy. The study [45] addressed data and concept drift in IoT with data quality improvements and an adaptable deep neural network model for stable learning. The study in [30] employed error rate and data distribution-based concept drift detection, using K-Means clustering and SVM for efficient anomaly detection, showing accuracy gains and fewer false alarms. The research, [9] explored data and concept drift detection in IoT with PCA and an adaptive online deep neural network, enhancing intrusion detection stability compared to static models.

The study in [153] emphasized the growing necessity for innovative approaches in network attack detection due to the evolution of network-based attacks. This study delved into distributed machine learning and ensemble techniques to detect concept drift and network attacks. Employing classifiers like RF, LR, and SVM, combined with K-means clustering for concept drift. Another study [90] explored an online PCA method for network intrusion detection, focusing on addressing concept drift in large data streams. This method, using online eigenvector transformation, showed improved recall and F1-measure compared to offline methods. The study in [32] proposed an ensemble-based framework for online intrusion detection, employing a GP method to derive a combiner function. This approach manages concept drifts and integrates functionalities like drift detection and base classifier induction.

The research in [46] introduced a semi-supervised drift detector utilizing an ensemble of classifiers based on self-training online learning and dynamic classifier selection. This method is crafted to identify drifts and update the decision model, diminishing the reliance on labeled data. The study in [154] presented a machine learning-based NIDS featuring a two-phased hybrid ensemble learning and automatic FS. This framework boosts attack detection capability in network applications by employing multiple classifiers for FS and using an ensemble learning algorithm. The research in [155] proposed framework for IDS, integrating MOO searching algorithms. This method augments the accuracy and F-measure of IDS by periodically employing NSGA-II, which selects solutions using mean and median types. Study in [44] proposes an AI-based Improved Long Short-Term Memory (I-LSTM) neural network for anomaly detection in smart city services. The I-LSTM enhances multi-classification performance by adding time factors and an activation function.

The study in [29] revealed a framework to train SVM for intrusion detection by dynamically optimizing its hyperparameters C and γ, utilizing Moth-Flame Optimization (MFO) as the primary algorithm, augmented by Lightweight MFO and knowledge-based search for initializing solutions. This framework effectively re-trains the model in response to data changes, as indicated by a drift detection module. The research in [152] developed Multi-CAD, a confidence-guided anomaly detection model for dynamic logs. It employs a statistical value, p_value, to guide anomaly detection, demonstrating its effectiveness in dynamic log analysis. Lastly, the study in [140] develops an extended variant of the GP combiner classification. It integrates online sequential extreme learning machine components and strategies to handle different types of concept drift.

The literature of IDS for non-stationary data streams, demonstrates a broad spectrum of methodologies targeting various challenges such as concept drift, dimensionality reduction, and data imbalance. However, a notable gap persists in addressing feature drift, a critical and often overlooked aspect. Studies such as [153] and [154] have utilized feature selection techniques to minimize data dimensionality and have deployed various machine learning algorithms for intrusion detection. Conversely, research like [158], [32], and [152] has concentrated on addressing concept drift in non-stationary data stream. Despite these advancements, the issue of feature drift has been relatively overlooked, indicating a crucial area for further exploration within the field. Feature drift, involving changes in feature distribution over time, is crucial for maintaining the efficacy of IDS in dynamic environments. This oversight highlights an essential area for future research, where the development of adaptive models capable of recognizing and adjusting to evolving feature landscapes is imperative.

Bridging this gap is key to advancing IDS capabilities, ensuring robust and resilient security measures in the face of constantly evolving network threats and data characteristics. Furthermore, it's imperative that IDS should jointly handle all the aforementioned critical challenges feature drift, concept drift, dimensionality reduction, and data imbalance to provide a comprehensive solution for dynamic network environments.

In the review of literature for the GP-based ensemble framework, as outlined in Table 2.3, the focus has largely been on the framework's effectiveness in addressing concept drift within IDS and data streams. However, the analysis reveals a significant gap in the existing methodologies regarding their capability to manage feature drift, which is critical for maintaining the relevance and accuracy of model over time. The problem emphasizes that while GP-based ensembles are adept at adapting to concept drift where the underlying data distribution changes over time, they fall short in dynamically adjusting to the varying importance of features.

For instance, the study [140] is a robust approach to concept drift within Network IDS, yet it does not incorporate DFS, Similarly, the methodologies applied in [141], [144], and [146] focus mainly on adapting to concept drift in data streams but neglect the critical aspect of feature drift reduce the dimensionality and the multi objective nature of FS. This oversight suggests that the current GP-based ensemble frameworks are not fully equipped to handle the data stream critical challenges simultaneously.

Moreover, the second table, Table 2.4, reviewing IDS methodologies for non-stationary data streams, further underscores this gap. The reviewed approaches,

including techniques like PCA in [9] and [90] , LSTM [44], and various ensemble learning strategies [32], [45], [46], [154], and [140], primarily target dimensionality reduction and concept drift and class imbalance without a comprehensive strategy for addressing feature drift, but they lack the necessary components to adapt to feature drift, thereby limiting their applicability in scenarios where the importance of features changes over time.

The proposed IDS DFA-GPE framework effectively integrates the techniques highlighted in both tables to significantly enhance performance in data streams. It comprehensively addresses the challenge of feature significance shifting by dynamically adapting to feature drift while maintaining the ability to handle concept drift. Additionally, the framework incorporates a multi-objective optimization approach for feature selection, ensuring a well-balanced trade-off between accuracy and memory usage. This holistic approach allows the DFA-GPE to achieve superior adaptability and robustness in real-time IDS applications, making it well-suited for dynamic and evolving data environments.

2.12.2 Multi Objective Particle Swarm for Complex Problems Optimization

The recent literature review on MOPSO, as highlighted in Table 2.5, showcases a diverse range of advancements in optimization techniques. It delves into the MOPSO method, examining various aspects such as initialization techniques, balancing trade-offs between exploration and exploitation, the employment of non-dominated solutions, solution selection methods, local searching, and multi-exemplar strategies. The study [115] significantly advances PSO application in autonomously optimizing

CNN architectures by managing variable-length structures and speeding up the process through evaluation on smaller data subsets. The research [47] introduces a multi-objective optimizer that combines PSO with evolutionary game theory, employing a self-adaptive rule driven by evolutionary strategies for optimal parameter tuning. The work [159] explores an improved PSO integrated with a Pareto archive for multi-objective reactive power optimization, using a local search greedy strategy to effectively balance search scopes and optimize solution selection.

The study [160] addresses PSO's limitations, like diversity loss and local optima entrapment, by combining PSO with the Backtracking Search Optimization Algorithm (BSA). This hybrid approach enhances convergence and precision through modified mutation and crossover operations. In [161], a multi-objective PSO algorithm simplifies exploration and exploitation, supported by a qualitative analysis of similar operations in genetic algorithms and PSO. Study [162] tackles PSO challenges related to archive size and solution diversity by introducing a MOO-based PSO with a novel archive update mechanism and strategic refreshes for non-dominated solutions, improving convergence and diversity.

In work [49], a multi-modal MOO-based PSO with a self-adjusting strategy is proposed to counteract convergence deterioration. This method employs a multi-swarm optimization approach, dividing particles into sub-swarms for diverse solution exploration. The search within these sub-swarms is guided by diversity feedback, and a balancing strategy ensures equitable progress across various solution spaces. The study by [163] analyses the swarm connection topology in MOO PSO, focusing on the number of connections between particles. A version of the Speed-constrained MOPSO

is adapted to improve sensitivity to swarm topologies, and its performance is evaluated using machine learning metrics on regular graphs of varying degrees. The work [164], the authors address the issue of optimal solutions in MOO methods being too loosely distributed due to inadequate leader selection. They proposed a method that enhances evolutionary performance by detecting the evolutionary state, adapting leader selection for different evolutionary states, and refining performance based on each particle's dominance relationship.

The studies in [165] and [125] explore dynamic MOO in environments with shifting objectives. In the first propose a double search strategy in PSO for both rapid convergence and population diversity, augmented by an archive set prediction and a piecewise search strategy. This approach ensures optimal solution distribution in changing scenarios. The second study address dynamic multi-objective problems by initially positioning solutions in a single space, then dividing the swarm into sub-swarms using the Pareto ranking operator upon detecting objective changes. This method enhances exploitation and employs a dynamic strategy with random detectors for objective function changes, allowing for re-evaluation and replacement of solutions as needed.

The paper [166] introduce a two-stage population cooperation PSO algorithm to address challenges of high dimensionality and balance between convergence and diversity. The first stage enhances convergence through auxiliary population cooperation, while the second promotes a balance using Sub-Population Cooperation. In [167], a two-archive MOO PSO method is designed for optimizing convergence and maintaining diversity, using separate archives for each and integrating genetic

techniques with an adaptive parameter system for improved solution quality. Meanwhile, the author in [168] presents an innovative global best solution selection strategy based on a virtual generational distance indicator from an elite archive, optimizing solution suitability.

The work referenced in [169] focuses on advancing MOO for high-dimensional challenges through the innovative Cooperative Coevolutionary Multi-Guide Particle Swarm Optimization (CCMGPSO). This approach integrates cooperative coevolution with the MGPSO framework to enhance scalability and computational efficiency. Its superiority, particularly in navigating the complexities of high-dimensional environments, is validated through empirical comparisons against leading optimization algorithms, highlighting its significant contributions to the field.

Finally, in the research of [26], an advancement has been made with the Multi-Exemplar Particle Swarm Optimization (MEPSO) algorithm. To solve the issue that traditional PSO often suffer from, such as insufficient population diversity and premature convergence due to their reliance on global best and personal best positions, MEPSO addresses these problems by having each particle select both the global best and the best companion particle as exemplars, thus enhancing diversity. It also introduces influence coefficients inspired by mechanics and a variable-scale search mechanism, leading to improved convergence. However, MEPSO lacks a multi-objective-aware criterion for exemplar selection, a Tabu search mechanism for providing local and global awareness, and an improved initialization phase.

Table 2.5: Literature surveys existing methods of multi-objective optimization

Ref	MOO Method	Smart Initialization	Trade-off Exploration and Exploitation	Non-Dominated Solution	Solution Selection	Local Searching	Multi-exemplar
[47]	Modified Self-Adaptive PSO	x	√	√	Roulette wheel / density estimation	x	x
[159]	Particle Swarm Optimization and Pareto Archive	x	√	√	Minimum sum between objective function and optimal solution	x	x
[160]	Particle Swarm Optimization & search optimization algorithm	x	√	x	Greedy selection	√	x
[170]	Particle Swarm Optimization & variable-length CNN	x	x	x	Global Best Selection and Decoding	x	x
[161]	Multi-objective Particle Swarm Optimization	x	√	√	Elitism operator	x	x
[162]	PSO with archive updating mechanism	x	√	√	Distance	x	x
[49]	Multi multi-objective particle swarm Self-Adjusting Strategy	√	√	√	crowding distance/ dominance rank	x	x
[163]	Speed-constrained Multi-objective Particle Swarm Optimizer	x	√	√	Crowding Distance	x	x
[164]	PSO/ adaptive multiple selection strategy	x	√	√	Sparse degree and rank	x	x
[169]	Cooperative coevolutionary multi-guide particle swarm optimization	x	√	√	Crowding distance-based archive	x	x
[165]	Dynamic multi-objective optimization problems PSO	x	√	√	Crowding distance-based ranking	x	x
[166]	Population Cooperation based Particle Swarm Optimization	√	√	√	Crowding distance	x	x
[167]	Multi-objective particle swarm optimization algorithm based on two-archive mechanism	x	√	√	Euclidean distance	x	x
[168]	Multi-objective particle swarm optimization	x	√	√	Roulette wheel selection	x	x
[125]	Dynamic Pareto bi-level Multi-Objective Particle Swarm Optimization	x	√	√	Crowding distance and the Pareto ranking	x	x
[26]	Multi-Exemplar Particle Swarm Optimization	x	√	x	Update the global and personal best based on fitness evaluations	x	√
Proposed method	Multi-Exemplar PSO with Local Awareness	√	√	√	Grid based selection	√	√

MEPSOLA shares several key features with some existing MOO methods, particularly in its approach to balancing trade-offs between exploration and exploitation, as well as its ability to identify non-dominated solutions. These are common across several methods such as those referenced in [161], [162], [164], and [165]. However, MEPSOLA distinguished in several important ways that contribute to its superior performance.

Firstly, MEPSOLA incorporate an improved initialization strategy, which ensures that the optimization process begins with a well-distributed and representative initial population. This is critical in effectively exploring the solution space from the start. Unlike methods in [161], [162], [164], and [164] which do not emphasize this feature, MEPSOLA ensures that the algorithm does not start from a biased or poorly distributed initial set of solutions. When it comes to solution selection, MEPSOLA uses a grid-based selection mechanism, which is more advanced compared to the other methods. For instance, while [162], [164], and [165] rely on crowding distance, and [161] Elitism operator to select the solutions, MEPSOLA's grid-based selection method ensures a more uniform and diverse set of solutions. This is particularly important in maintaining a balance between exploration and exploitation throughout the optimization process.

Furthermore, MEPSOLA is unique in its inclusion of local search capabilities and a multi-exemplar strategy, features not found in the other methods listed. The local search component allows MEPSOLA to refine solutions more effectively within promising regions, thereby improving the algorithm's convergence to the true Pareto front. This capability is crucial for achieving high-quality solutions, especially in

complex optimization problems. The multi-exemplar strategy employed by MEPSOLA further enhances the diversity of solutions and reduces the risk of premature convergence, ensuring that the algorithm does not get stuck in suboptimal regions of the solution space.

The reviewed literature on MOPSO highlights several advancements that directly address the challenges identified in the problem statement, particularly the issues of diversity loss and the risk of convergence to local minima. These critical advancements is the introduction of an improved initialization techniques, as seen in studies like [49] and [166] , which strategically position the initial population within the search space to mitigate the sensitivity of traditional PSO algorithms to initial conditions. This enhancement ensures that the optimization process begins with a diverse and well-distributed population, reducing the likelihood of premature convergence. Moreover, the balance between exploration and exploitation as a key concern in maintaining diversity while ensuring convergence, is effectively managed in approaches like those discussed in [160], [161], and [162], where hybrid algorithms integrate methods such as the Backtracking Search Optimization Algorithm (BSA) and qualitative analysis and a novel archive update mechanism . These methods allow for a broader exploration of the solution space while concentrating the search in areas with high potential, thereby addressing the problem of local optima.

Furthermore, the introduction of the multi-exemplar strategy, particularly in the Multi-Exemplar Particle Swarm Optimization discussed in [26], to enhance swarm diversity and avoid reliance on a single global best solution. Additionally, the incorporation of local search mechanisms, such as those mentioned in [160] , plays a pivotal role in

refining solutions within specific regions and adaptively exploring new areas when current searches are suboptimal.

Finally, as highlighted in the table, the propose MEPSOLA algorithm represents a significant advancement over previous methods by jointly considering and improved initialization, trade-off exploration and exploitation, non-dominated solutions, solution selection, local searching, and multi-exemplar strategies. By integrating these techniques, MEPSOLA enhances the algorithm's overall performance, particularly in complex optimization scenarios. This approach ensures not only a balanced exploration of the search space but also improved convergence rates and solution quality, making it well-suited for high-dimensional and dynamic environments. The comprehensive incorporation of these techniques in MEPSOLA demonstrates its potential to effectively address the challenges of dynamic feature selection and real-time decision-making, particularly in applications like IDS, where adaptability and performance are crucial.

2.12.3 Multi Objective Optimization Algorithms for Feature Selection

Feature selection has considered as optimize a problem with multi-criteria nature. Therefore, researchers have favored algorithms of multi-objective meta-heuristic searching [171]. One of FS applications is IDS where FS technique helps to reduce the computation time and complexity as well as improve the model performance and reducing the memory consumption [172], [50]. Many challenges are faced in FS for IDS, including high dimensionality, feature drift, concept drift, and data imbalance. Furthermore, the literature in Table 2.6, investigates important aspects of FS

techniques, such as the use of DFS, MOO FS nature, and the existence of variable length representations. Additionally, it explores the employment of transfer functions for binary searching.

The literature review for existing methods of MOO-based for FS, Highlights a method for handling the high dimensionality problem using MOO algorithm was tackled by variable length algorithms for instance in the work [23], by enabling particles to have different lengths, which defines smaller search space. While in the work of [55], a variable-size PSO algorithm for FS was proposed. The algorithm employed the idea of divide and conquer. First, a space division strategy based on the feature importance is presented, where classify relevant features into the same subspace with a low computational cost. Similarly the author in [54], proposed a strategy of combining filter and wrapper approaches in a single evolutionary process in order to achieve smaller feature subsets with maintaining the classification performance in a shorter time. Next, MOO based evolutionary algorithm used for elevating the feature drift occurrence and effect in data streaming environments was proposed in [100], the algorithm called dynamic filter-based feature selection. It determines the time of feature drift occurrence in the stream by using one-time random initialization (NP1) mechanism and it selects a relatively small subset of relevant features from the original feature space. Then classification performed using Artificial Neural Network (ANN). Some authors identify balancing exploration and exploitation as a key challenge [19] [54]. At this point, in the study of [173], introduces a multi-objective FS algorithm named Binary Differential Evolution with self-learning, designed to overcome Differential Evolution (DE) limitations in multi-objective FS, particularly in exploration. It incorporates three techniques: binary mutation to enhance search, a one-

bit purifying technique for selecting optimal individuals, and a non-dominated sorting operator with crowding distance to streamline selection complexity in DE.

In [174], a hybrid algorithm combining the binary versions of Grey Wolf Optimization (GWO) and PSO is proposed for FS, leveraging both algorithms to balance exploration and exploitation for improved solution spaces and outcomes, incorporating a K-nearest neighbors classifier. Work in [51] introduces a hybrid FS method merging the Butterfly Optimization Algorithm (BOA) with a Feature Interaction Maximization (FIM) scheme for optimal feature subset selection, enhancing exploration and reducing redundancy. Studies [175] and [176] employ MOO for FS using the Salp Swarm Algorithm (SSA) and GWO, respectively, to balance exploration and exploitation and prevent local optima entrapment. In the same content, the work of, [56] presents a PSO-based method to address high-dimensionality challenges in FS, using a graph representation model for initial features, calculating feature centralities, and finalizing with a PSO search process.

In the research [177] developed three solutions: Harris Hawks Optimization, Fruitfly Optimization Algorithm, and their hybrid method, demonstrating some promising results against established algorithms like MOPSO and NSGA-II on various datasets. Study [52] proposed the Information Gain binary BOA to enhance FS, focusing on maximizing classification accuracy and the mean mutual information between features and class labels. This method underwent a three-phase process, showing effectiveness in selecting optimal feature subsets on UCI datasets. Paper [53] introduced an algorithm named SFE, targeting high-dimensional datasets with a two-phased approach for FS. This was further enhanced with a hybrid SFE-PSO algorithm, showcasing superior performance in selecting features compared to other algorithms.

The analysis for methods of MOO-based FS, Table 2.6 summarizes various techniques that address critical challenges such as high-dimensional data handling, dynamic feature selection, fixed versus variable-length representations, and the balance between exploration and exploitation. These methods have been instrumental in advancing feature selection, mainly in offline or static environments. However, when examined in the context of the problem statement, which emphasizes the need for a dynamic, real-time feature selection process capable of adapting to concept drift and feature drift, a significant gap in the current literature becomes evident.

Many methods listed in the literature, such as the GA [50]and the BOA [51], [52] and PSO [53] [54] [55] [56] [23] , are adept at managing high-dimensional data by effectively reducing the number of features. However, these methods typically operate in static environments and do not dynamically adjust to changes in feature relevance, which is essential for real-time decision-making.

The problem statement specifically calls for DFS, which is crucial in environments where feature importance can shift over time due to feature drift. Among the methods summarized, the method [100] stands out as one of the few that addresses this challenge by dynamically selecting relevant features in real-time. However, its lake capability for variable length features representation, transfer function, and handling class imbalance. Variable length necessary to reduce the complexity of FS while the transfer function and handling class imbalance are crucial in MOO DFS as they respectively ensure effective binary feature selection and maintain balanced classification performance across diverse classes, particularly in dynamic, real-time environments.

The variable-length representations, as seen in methods like PSO with cooperative coevolutionary strategies [55], [23], offer flexibility in adjusting to different feature set sizes, which is advantageous in dynamic environments. This approach reduces the search space and computational overhead, but many methods still rely on fixed-length representations, limiting their ability to adapt in real-time scenarios.

In balancing exploration and exploitation is a central challenge in MOO. Many methods in the literature are seek to address this balance. However, while these methods excel in static environments, they may not be fully optimized for dynamic, real-time applications where the ability to quickly adapt to changing data is crucial.

Solution selection of non-dominated solutions and effective solution selection are crucial for ensuring that the selected features remain relevant and diverse. Majority of methods in the table that employ strategies like crowding distance [178] [173] [177] [174] or ranking [100] are particularly effective. Nonetheless, voting technique important to ensure the diversity in selection and ignoring favoring a specific search space and ignoring the rest.

Handling concept drift and data imbalance two critical challenge mentioned in the problem statement. Although the method [100] explicitly addresses the challenge of concept drift, most other methods in table do not while data imbalance tackled by [54] [176], revealing a gap in the ability to maintain accuracy and relevance in the face of changing data distributions.

The method proposed in this study jointly considers all these challenges started from the multi objective nature of FS, high-dimensional data handling, DFS for handling the feature drift, variable-length representations to reduce the complexity of FS, balancing exploration and exploitation, and handling concept drift and data imbalance, to improve the DFS process for online decision-making. By integrating these aspects, our method not only addresses the gaps identified in the existing literature but also enhances the adaptability and accuracy of feature selection in real-time, dynamic environments. This comprehensive approach represents a significant advancement in the field of MOO-based FS, making it well-suited for applications that require continuous, real-time decision-making.

Table 2.6: Literature surveys existing methods of MOO-based for FS

Ref	High Dimension Handling	Dynamic Feature Selection	Fixed VS Variable Length	MOO Method	Trade-off Exploration and Exploitation	Non-Dominated Solution	Solution Selection	Binary Representation	Transfer Function	Concept Drift Handling	Imbalance Handling	Improvement
[178]	×	×	F	Bee Colony Algorithm (ABC)	√	√	Crowding Distance/Archive	√	×	×	×	Balancing the trade-offs between classification performance and feature cost
[173]	×	×	F	Differential Evolution (DE.)	√	√	Crowding Distance/Archive	√	×	×	×	Improve algorithm's ability to in balance between global exploration and local exploitation
[48]	×	×	F	Particle Swarm Optimization (PSO)	√	√	Crowding /Tolerance Coefficient/Archive	√	×	×	×	Enhancing its effectiveness in real-world applications where costs of features are not precisely known
[55]	√	×	V	Particle Swarm Optimization (PSO) + cooperative coevolutionary	√	×	Crossover/Correlation	√	×	×	×	Reduces computational cost
[177]	×	×	F	Harris Hawks Optimization (HHO)/Fruitfly Optimization Algorithm (FOA)	√	√	Distance	√	S-Shape	×	×	Balance between diversity and convergence
[50]	√	×	F	Genetic Algorithm (GA)	×	√	LR/Crowding/Feature Rank/Crossover/ Mutation	√	S-Shape	×	×	Improved classification accuracy and reduced computation time
[179]	×	×	F	Grey Wolf Optimizer (GWO)	√	√	Crowding / Density Estimation/Archive	√	S-Shape	×	×	Balance the reduction of feature subsets and the minimization of classification error rates
[180]	×	×	F	Particle Swarm Optimization (PSO)	√	√	Crowding/Frequencies/Archive/Mutation	√	×	×	×	Reducing the number of features without compromising classification accuracy

Table 2.6: Continued

[23]	√	×	v	Particle Swarm Optimization (PSO)	√	×	Wrapper/distance measure	×	×	×	×	Smaller feature subsets with higher classification performance in a shorter time
[56]	√	×	F	Particle Swarm Optimization (PSO)	√	×	Ranking/Mutation	√	S-Shape	×	×	Enhances the relevance and reduces redundancy in the selected feature subsets
[176]	×	×	F	Grey Wolf Optimizer (GWO)	√	×	Wrapper/Crossover/Mutation	√	S-Shape	×	√	Improve classification accuracy, feature subset reduction, and computational efficiency
[174]	×	×	F	Grey Wolf Optimization (GWO)/Particle Swarm Optimization (PSO)	√	×	Wrapper/Distance	√	S-shape	×	×	Improves the classification accuracy while reducing the number of selected features
[181]	×	×	F	Butterfly Optimization Algorithm (BOA)	√	×	Wrapper /Distance	√	S+V Shape	×	×	Maximize classification accuracy while minimizing the number of selected features
[52]	√	×	F	Butterfly Optimization Algorithm (BOA)	√	×	Features numbers /Accuracy /Mean	√	S-shape	×	×	High classification accuracy and reduced feature redundancy
[51]	√	×	F	Butterfly Optimization Algorithm (BOA)	√	×	Highest Fitness/Mutation	√	×	×	×	Classification accuracy, reduced computational complexity
[100]	√	√	F	Dynamic Filter-Based Feature Selection	√	√	Relevance Ranking /Crowding Distance	√	×	√	×	Improves classification accuracy and the ability to quickly adapt to changes in the data stream
[175]	×	×	F	Salp Swarm Algorithm (SSA)	√	×	Fitness Function	×	×	×	×	Improve classification accuracy, fewer selected features, and faster convergence

Table 2.6: Continued

[172]	×	×	F	Grey Wolf Optimization (GWO)	×	×	Fitness Function	√	Mod Equation	×	×	Minimizing the number of features and maximizing detection performance, which results in a more efficient and effective IDS capable of identifying different types of attacks
[54]	√	×	F	Particle Swarm Optimization (PSO)	√	×	Distance Measure/Weighting Coefficient	×	×	×	√	Reduces the computational cost and improves the overall efficiency, particularly in high-dimensional
[53]	√	×	F	Particle Swarm Optimization (PSO)	√	×	Fitness Function	√	×	×	×	Improved classification accuracy and reduced feature sets
Proposed method	√	√	V	Particle Swarm optimization (PSO)	√	√	Voting	√	S+V Shape	√	√	Balancing between accuracy and memory reduce dimensions

2.13 Summary

This chapter presents a comprehensive literature review in the field of IDS, focusing on the challenges and methodological shortages associated with non-stationary data streams. It explores the evolution and principles of IDS, highlighting their significance in the dynamic network security landscape. The chapter examines various challenges, including high-dimensional data, concept and feature drift phenomena, and imbalanced datasets. It then discusses feature selection methods and the distinction between static and DFS techniques. The integration of MOO with PSO in feature selection is also discussed, emphasizing their role in enhancing IDS performance in terms of accuracy and memory consumption. Thus, the chapter identifies a gap in IDS for evolving feature handling while simultaneously balancing accuracy and memory. MOPSO is utilized for its simplicity and effectiveness in managing multiple objectives, making it ideal for DFS and ensemble learning with a GP combiner to address concept drift. This approach highlights MOPSO's suitability for improving IDS performance in dynamic environments. The chapter lays the foundation for understanding IDS challenges within modern network data streams, exploring potential strategies to address these challenges. It paves the way for subsequent chapters, delving into advanced solutions for enhancing IDS efficacy in dynamic environments.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In the field of Intrusion Detection Systems (IDS), the significance of feature drift emerges as a pivotal challenge, closely entwined with the triad of additional challenges: high dimensionality [182], concept drift [8], and class imbalance[183]. Feature drift a phenomenon marked by the varying relevance of feature subsets over time, necessitates the constant adaptation of learning algorithms to determine currently significant features amidst a dynamic data stream [11]. This challenge is noticeable in IDS due to the evolving nature of network traffic and the tactics of intruders. The complexity is further augmented by the high dimensionality of data traffic, which potentially complicates the selection of relevant features among a thousand of features. Concurrently, concept drift alters the underlying data distribution, demanding ongoing model updates or replacement. Additionally, class imbalance further worsens this scenario, potentially skewing classifier performance toward the majority samples and obscuring the detection of infrequent yet critical security threats. An integrative approach that simultaneously addresses these challenges is essential for sustaining the relevance, accuracy, and performance of IDS in such dynamically evolving data streams.

Therefore, the research objectives are focused on developing a comprehensive methodology for stream data-based intrusion detection, adept at managing feature drift. This encompasses the formulation of an innovative Dynamic Feature Selection

(DFS) strategy and the creation of a Multi-Objective Optimization (MOO) algorithm, specifically designed for complex optimization scenarios and tailored for DFS that simultaneously optimize both accuracy and memory. This approach ensures the ongoing selection of the most relevant features as data streams evolve. The research culminates with a thorough evaluation of the proposed algorithms against contemporary methodologies, employing standardized metrics to evaluate their effectiveness in real-world scenarios characterized by shifting class distributions and feature relevance.

3.2 Methodology Background

This section has covers essential algorithms pivotal to the methodology. The focus has been placed on the Genetic Programming (GP) Combiner, designed for managing non-stationary data streams in intrusion detection, highlighting its adaptability to concept drift and class imbalance. The role of Multi-objective Particle Swarm Optimization (MOPSO) in MOO has been discussed, alongside an explanation of Tabu search. Furthermore, the Drift Detection Method (DDM) for identifying concept drift in data streams and the Synthetic Minority Over-sampling Technique (SMOTE) for addressing class imbalance have been explored.

3.2.1 Genetic Programming (GP) Combiner

The GP combiner-based ensemble classification framework, is designed to address the challenges posed by the non-stationary nature of modern network data flow, especially in the context of intrusion detection scenario [32]. The framework leverages a GP combiner function to dynamically fuse predictions from an ensemble of base classifiers, making it well-suited for non-stationary intrusion detection data. Unlike

traditional fixed methods like averaging, this GP-based approach adapts to changes in data, ensuring the model remains effective. The ensemble, comprising base classifiers and the GP combiner, adjusts to concept drift, maintaining robust performance in dynamic environments. For a detailed explanation, refer to Algorithm 1.

Algorithm 1: Pseudocode of Genetic Programming Ensemble Classifier

Input:
(x, y) is a current data tuple, where y may be a label or special symbol $\perp$ for unlabeled
$E = (B_E, F_E)$ ensemble model, where F_E is the combiner function and B_E a list of base classifiers
Buffer is a list of tuples for temporary storage
DDM: drift detection model
maxC is a maximum count of base classifiers in the ensemble
n a number of tuples in window size
learners a list of base classifier induction algorithms
l maximal number of base classifiers to select at each learning session, where $l < |\text{learners}|$
Output:
Predicted class for x
Updated E, Buffer, and DDM
Drift flag indicating concept drift
Method:
Stage 1, Classify x using the current ensemble E
Build the decision-profile matrix H^{B_E} of B_E for x
Compute a global support degree $\mu_j^{B_E,F_E}(x)$ for each class ω_j by applying combiner F_E to the $j-$ th column of H^L
Compute the predicted class label ẏ of x as follows $\hat{y} = \arg\max_j \{\mu(x)[j]\}$;
If $(y \neq \perp)$ Then // A labelled tuple is arrived
Insert (x, y) into Buffer;
drift := drift **or** DM. update&check(x, y, yj); // updates the state of DM and returns a drift detection flag
if (|Buffer| = n) **then** // A novel chunk of training data has become available
if (drift) **then**
drift := false;
Stage 2: Extend the list of base classifiers in the ensemble model E
Partition Buffer into a training set Train and a validation set Valid;
Build a set NewModels of classifiers by running each of the methods in learners on Train;
Insert into B_E the l classifiers in NewModels that achieve the highest l accuracy scores on Train;
If $(|B_E| > \text{maxC})$
Select and remove $\text{maxC} - |B_E|$ classifiers in Bc; // According to some chosen replacement criterion
end if
Stage 3: Update the combiner of the ensemble model E
Construct the decision profile $H^{B_E}(\dot{x})$ of all tuples $(\dot{x}, \dot{y}) \in$ Valid, and store them all into a list DP
$F_E :=$ computeGPcombiner (DP, Valid); // compute a novel GP-based combiner by using the CAGE tool
end if
Empty Buffer; // a new data chunk will start being formed at next run of the algorithmẋ ẏ
end if
end if
return $(\hat{y}, E, \text{Buffer}, \text{DM}, \text{drift})$;

3.2.2 Multi-Objective Particle Swarm Optimization (MOPSO) and Repository

MOPSO optimizes multi-objective problems by using an external repository to store nondominated solutions and a geographic method to maintain diversity, ensuring effective exploration and avoiding premature convergence. For more details, refer to Algorithm 2.

Algorithm 2: Pseudocode of Multi-Objective Particle Swarm Optimization

Input:
Multi-objective mathematical function = {FON, KUR, ZDT1, ZDT2, ZDT3, ZDT6}
Output:
Pareto front
Start:
Initialize_Population(Pop, Max)
  For i from 0 to Max
    $Pop_{(i)} \leftarrow$ Initialize_Particle()
Initialize_Speed(Pop, Max)
  For i From 0 to Max
    $Vel_{(i)} \leftarrow 0$
Evaluate_Particles(Pop)
Store_Nondominated_Positions(Pop, Rep)
Generate_Hypercubes(Pop)
Initialize_Memory(Pop, Max)
  For i from 0 to Max
    $P_{Best(i)} \leftarrow Pop_{(i)}$
While not Max_Cycles_Reached()
  For i from 0 to Max
    $Vel_{(i)} \leftarrow$ Compute_Speed $(Vel_{(i)} = W * Vel_{(i)} + R_1 * (P_{Best(i)} - Pop_{(i)}) + R_2 * (Rep_{(h)} - Pop_{(i)})$
    $Pop_{(i)} \leftarrow$ Update_Position$(Pop_{(i)} = Pop_{(i)} + Vel_{(i)})$
    Ensure_Bounds$(Pop_{(i)})$
    Evaluate_Particles(Pop)
    Update_Repository(Rep, the geographical representation in hypercubes Pop)
    If current particle position better than the position in the memory Update_Memory($P_{Best(i)}$, $Pop_{(i)}$)
    Else
      Retain_Current_Memory($P_{Best(i)}$)
  Increment_Cycle_Counter()
End while

3.2.3 Tabu Search

Tabu Search is an algorithm that employs local search methods for solving optimization problems. It enhances the performance of a local search by using a Tabu list to avoid local optima by prohibiting or penalizing moves that would reverse recently made changes [184]. For more clarification refer to Algorithm 3.

Algorithm 3: Pseudocode Tabu Search Algorithm

Input
Initial Solution S
Tabu List T (initially empty)
Maximum Iterations max_iter
Iteration Counter iter $=$ 0
Initialize
Set $S_{best} = S$
Algorithm
while iter $<$ max_iter:
    Find the neighborhood of S, N(S)
    For each candidate solution S' in N(S):
      if S' is not in T and is better than S_{best}:
        Update $S_{best} = S'$
    Move to the best candidate S' in N(S) that is not in T
    Update the Tabu list T
    Update $S = S'$
    iter $++$
Output
Return S_{best} as the solution
End

3.2.4 Drift Detection Method (DDM)

The DDM is a dynamic approach employed in data stream mining for detecting concept drift, which refers to changes in the statistical properties of the target variable over time. DDM monitors the error rate of a learning model, identifying significant changes that signify a drift in the data stream. This method is particularly effective in environments where data is evolving, necessitating continuous adaptation of the model

to new patterns and distributions [185]. DDM works by continuously monitoring the error rate of the classifier. As clarified in Algorithm 4.

Algorithm 4: Pseudocode for drift detection in data stream

Input:
time_series: A sequence of data points representing the time series.
Output:
A string indicating whether drift is detected ("Drift Detected") or not ("No Drift Detected").
Start:
Initialize Variables:
$n = 0$ (Number of examples processed)
$p_{min} = 1$ (Minimum error rate observed)
$s_{min} = 1$ (Minimum standard deviation observed)
warning_triggered = False
drift_triggered = False
stored_examples = []
Process Time Series:
Function ProcessTimeSeries(time_series):
  drift_detected = False
  for each example in time_series:
    drift_detected = UpdateDDM(example)
    if drift_detected:
      return "Drift Detected"
  return "No Drift Detected"
Update Drift Detection Method (DDM):
Function UpdateDDM(example):
  $n = n + 1$
  error = DetermineError(example)
  $p_i = CalculateErrorRate(n, error)$
  $s_i = sqrt(p_i * (1 - p_i) / n)$
  if $(p_i + s_i) < (p_{min} + s_{min})$:
    $p_{min} = p_i$
    $s_{min} = s_i$
  if $(p_i + s_i) >= (p_{min} + 2 \times s_{min})$ and not warning_triggered:
    warning_triggered = True
    StoreExample(example)
  if $(p_i + s_i) >= (p_{min} + 3 \times s_{min})$:
    ResetModel()
    TrainNewModel(stored_examples)
    $p_{min} = p_i$
    $S_{min} = s_i$
    warning_triggered = False
    stored_examples = []
    return True
  return False
Helper Functions:
ResetModel: Resets the learning model.
TrainNewModel: Trains a new model with stored examples.
StoreExample: Stores an example for future use.
CalculateErrorRate: Calculates the error rate.
DetermineError: Determines if an example is an error.
End:
The process ends when all examples in the time series have been processed, and the drift status is returned.

3.2.5 Synthetic Minority Over-sampling Technique (SMOTE)

The SMOTE method is a statistical technique to balance the imbalanced dataset by creating synthetic samples from the minority class instead of creating copies. This is achieved by taking samples of the feature space for each target class and its nearest neighbors. The method involves interpolating these samples to create new, synthetic samples of the minority class. For further clarification about SMOTE, refer to Algorithm 5.

Algorithm 5: Pseudocode of class imbalance handling (SMOTE)

Input:
D_t: Current data chunk with potential class imbalance
N_{min}: Number of minority class instances in D_t
k: Number of nearest neighbors to consider, $k = 5$
Output:
$D_{balanced}$: Balanced dataset with original and synthetic minority class instances
Procedure:
Initialize $D_{balanced}$ as an empty dataset
if Imbalance detected in D_t then
for each minority class instance x_i in D_t do
Add x_i to $D_{balanced}$
Neighbors[] = Find k nearest minority class neighbors of x_i in D_t
for $j = 1$ to k do
z_i = Neighbors[j]
for s = 1 to (N_{target} / N_{min}) do
x_{new} = GenerateSyntheticSample(x_i, z_i)
Add x_{new} to $D_{balanced}$
end for
end for
end for
end if
return $D_{balanced}$
Supporting Functions:
Find k nearest minority class neighbors of x_i :
- Use a suitable distance metric to find the k closest minority class instances to x_i.
GenerateSyntheticSample(x_i, z_i):
- Randomly choose a point along the line segment connecting x_i and z_i to create a synthetic sample.
- This can be done by interpolating between the feature values of x_i and z_i with a random factor between 0 and 1

Selecting $K = 5$ in the k-NN algorithm for SMOTE is a well-justified choice due to its effectiveness in balancing bias and variance. If K is set too low, the model risks overfitting to noise, leading to high variance. Conversely, a higher K could result in an overly generalized model with high bias, smoothing out the decision boundary excessively. Empirical evidence from various studies supports $K = 5$ as a practical compromise that generally provides robust performance across different datasets. This value allows SMOTE to generate synthetic samples that accurately reflect the local structure of the minority class, without being overly influenced by the peculiarities of individual data points [186]. Moreover, the use of $K = 5$ is consistent in the literature, making it a standard choice that facilitates comparison across different research efforts [187] [188].

3.3 Methodology Design

This section describes the methodology design of the thesis, which serves as the architectural framework, as represented in Figure 3.1. The design is methodically structured to address the demands of stream-based IDS, with particular emphasis on the phenomena of feature drift phenomenon, and to jointly handle concept drift, high-dimensionality, and imbalanced data. the methodology unfolds across three strategic objectives represented by three stages, each contributes to a comprehensive research approach, ensuring that the resulting IDS framework is theoretically sound, empirically robust, and practically applicable in real-world applications. The methodology commits to a precise evaluation protocol. The proposed algorithms are accurately benchmarked against current state-of-the-art methodologies within the field, using well established and standardized evaluation metrics. This evaluation is

crucial to validate the performance of the proposed solutions and to assert their significance and innovative contribution to the domain of IDS.

The methodology in the figure begins with stage 1, where a framework is developed specifically for stream-based IDS named Dynamic Feature Aware Genetic Programming Ensemble (DFA-GPE). This framework is developed to effectively detect and adapt to both feature and concept drift, and class imbalance, ensuring the IDS remains dynamic and reactive as the data evolves. This achievement is realized by initiating the development of a standalone MOO algorithm as shown in stage 2 called Multi-Exemplar Particle Swarm Optimization with Local Awareness (MEPSOLA), tailored to solve complex optimization problems. This framework stands out for its incorporation of smart initialization, multi-exemplar strategy, and conditional local search based on tabu search for exemplar selection enhancement techniques, enhancing the algorithm's precision in navigating the solution space. The framework is evaluated based on critical metrics in MOO evaluation, and the results are also compared with state-of-the-art benchmarks in the literature to assess the MOO effectiveness. Moving forward, in Stage 3, the MEPSOLA algorithm undergoes further refinement and adjustment to adeptly handle the complexity of a high-dimensional, discrete search space real world dataset. This evolution results in the Variable Length Multi-Objective Particle Swarm Optimization (VLMO-PSO), tailored for online decision-making in feature selection, facilitating DFS in real-time within the IDS framework. Rigorous evaluations of the IDS framework focus on critical classification and memory reduction metrics, comparing results with leading feature selection benchmarks. This approach ensures the IDS's adaptability to evolving data distributions, maintaining effectiveness amidst changing data patterns.

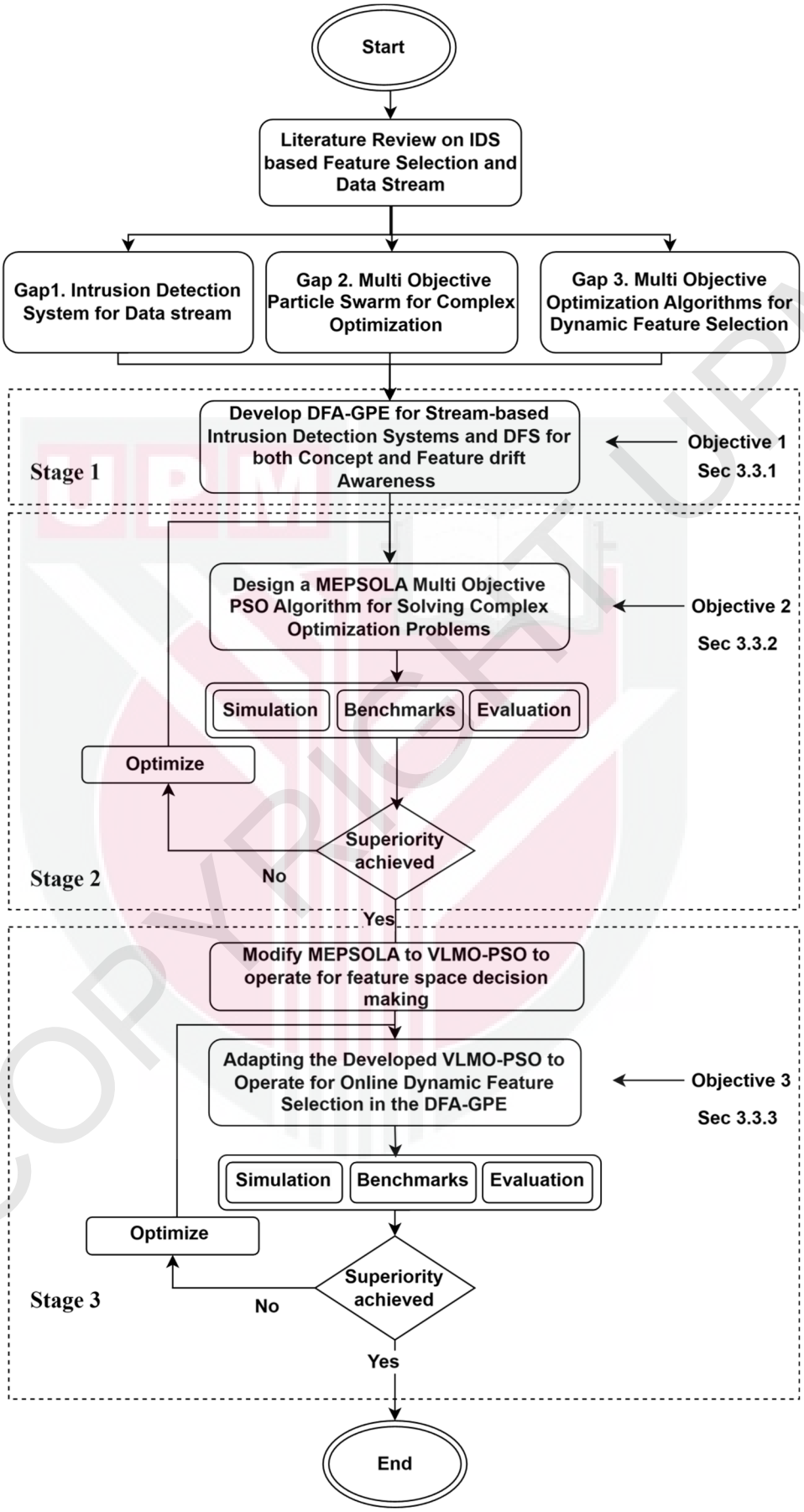


Figure 3.1: The general flowchart of our methodology

3.3.1 Objective 1: Dynamic Feature Aware Genetic Programming Ensemble (DFA-GPE) Framework

This section outlines the conceptual design of the IDS framework aimed at tackling dynamic data stream challenges in classification scenarios. It focuses on creating a framework responsive to the evolving relevance of features in stream-based IDS, known as feature drift, which is crucial for accurately classifying data streams where feature content and significance may vary. The framework is also designed to address the concept drift challenge. The goal is to develop an IDS classifier that adapts to these challenges and utilizes a DFS-based Developed MOPSO, balancing between maximizing accuracy and minimizing memory usage simultaneously.

The proposed framework operates through an incremental learning procedure and aims to seamlessly integrate various components, including advanced DFS techniques, drift detection mechanisms, and class imbalance handling, into a cohesive genetic programming ensemble. This integration facilitates a system that not only responds to immediate data distribution changes but also evolves with them, ensuring sustained performance and relevance. The ensemble's adaptability is key to addressing the dual challenges of concept and feature drift, a necessity for creating effective and resilient IDS solutions in the ever-changing landscape of network security.

3.3.1.1 Problem Formulation

In the dynamic environment of data stream classification such as in IDS, the relevance of feature subsets continually shifts, this shifting probably in both length and content. Such a variations necessitate a system that can adeptly handle both the challenges of

feature drift and concept drift. In this context, the symbol y_t represents the class label of the instance, where y_t belongs to the set of $\{1,2,\ldots,m\}$, and m denotes the number of classes to which indicating its category. It is suggested that y_t is known only at subsequent times t, or the knowledge of the ground truth of x_t is presented to the classifier partially, with a percentage of R, at the moment $t_{\mathrm{GT}} = t + T_L$, where T_L indicates the delay time in receiving label information for the data chunks. Additionally, it is understood that at any given moment t, only a subset of the features is relevant, and the length and the indices of these features vary with time. The challenge lies in constructing a prediction for the class of the sample, considering the provided label information under these constraints, with the goals of maximizing accuracy and minimizing memory usage.

The GP combiner framework has been considered to tackle the challenge of concept drift effectively [32]. However, the aim is to extend this capability to address feature drift simultaneously within a comprehensive system, alongside with critical challenges such as high dimension and imbalanced data. The effort is to develop an advance framework named DFA-GPE, which not only responds to concept drift but also to the variability in feature relevance over time. The integration of various components, including the GP-combiner, DFS mechanisms, drift detection technique and class imbalance handling into a unified framework. This integration poses a significant challenge. It requires accurate coordination between these elements to ensure the alignment of their adaptive capabilities. This coordination is pivotal for the DFA-GPE framework to achieve its final aim, to maintain high classification accuracy and optimize memory usage while navigating the twin challenges of concept and feature drift in data streams.

3.3.1.2 Dynamic Feature Aware Genetic Programming Ensemble (DFA-GPE)

The DFA-GPE framework is distinguished by its Sequential Adaptive Refinement (SAR) procedure, which is vital for managing the dynamic nature of the data stream. This process unfolds in two distinct yet interconnected phases as shown in Figure 3.2.

The first phase of SAR involves the intelligent distillation of incoming data to its most relevant features. This is achieved through an improved MOPSO algorithm named VLMO-PSO, which is designed to select and prioritize features that are indicative of the current state of the data stream. By focusing on relevant features, the algorithm also effectively reduces the dimensionality of the data flow, enhancing the accuracy, reduces computational load, and adapts to changing data. This selective process goes beyond mere data dimensionality reduction; it represents a strategic approach that guarantees subsequent predictive models is both efficient and relevant to the task at timing.

The subsequent phase is the GP combiner involves with the DFS to further improve the predictive models. This adaptive phase is where the models are not only generated but also continuously optimized to maintain high accuracy in the face of changing data stream. The improvement here is ongoing; as the data stream evolves, exhibiting concept and feature drift, the models are adjusted accordingly, ensuring that the system's predictive accuracy does not decrease over time.

The SAR process offers several notable advantages, key among them being the enhancement of predictive performance. By consecutively focusing on a relevant set

of features, the GP combiner is capable of producing models that are finely adapted to the fluctuation in the patterns of data, leading to improved predictive accuracy. Additionally, this process contributes to memory efficiency. By filtering out irrelevant features early on, the framework preserves computational resources, a benefit that is particularly valuable in stream-based settings where data volumes can be significant. Moreover, the interaction between feature selection and model optimization within the DFA-GPE framework exemplifies its sophisticated architectural design. This interaction is carefully coordinated, ensuring that each stage of the SAR process both informs and enhances the other. Consequently, the system can quickly and precisely adapt to new data, modifying its internal mechanisms to stay in sync with evolving data nature.

The DFA-GPE framework, reinforced by its SAR process, marks a considerable progression in IDS. By cohesively addressing concept drift and feature drift, the framework establishes a new benchmark for developing IDS solutions that are not only reactive but also proactive in data analysis. Such adaptability is crucial for designing IDS systems that can efficiently function in today's increasingly complex and dynamic cyber environments.

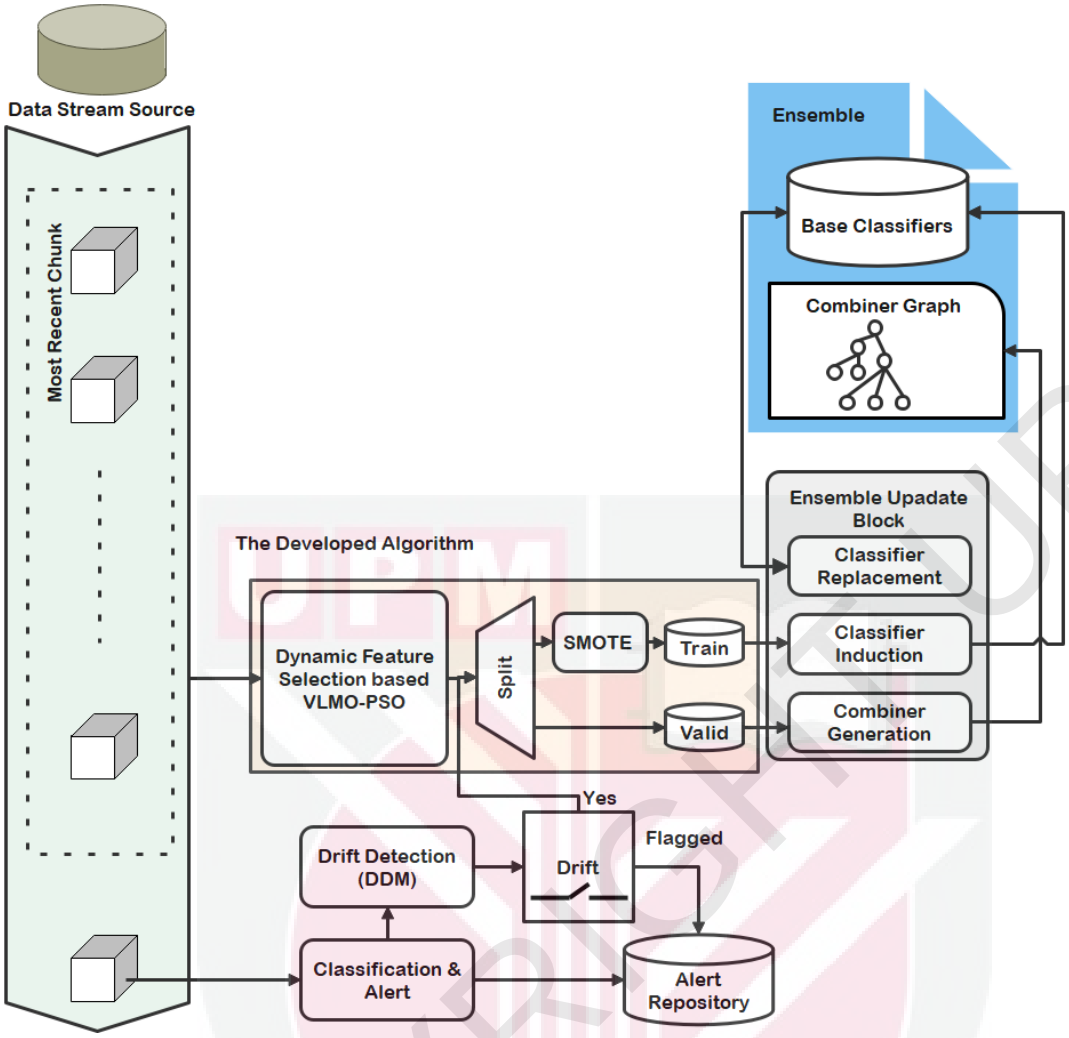


Figure 3.2: The proposed Dynamic Feature Aware Genetic Programming Ensemble (DFA-GPE) for concept drift detection and handling feature drift

3.3.1.3 Incremental Learning Procedure

In the proposed DFA-GPE framework, the continuous and potentially unbounded nature of data streams is a fundamental consideration [189]. Let's assume D_t represents the data chunk received at a given time t, and h_t represents a hypothesis formulated based on D_t. Within an IDS that employs incremental learning, each hypothesis h_t corresponds to the probability that a particular network event is an intrusion [190], as inferred from the patterns discerned from D_t.

Upon the arrival of a new data chunk, the DFA-GPE framework adapts by incrementally updating its knowledge base. Rather than only discarding the previous hypothesis h_t, the framework refines itself, utilizing the newly acquired data chunk to adjust the existing hypothesis. This refinement process negates the need for storing previous data, as the cumulative knowledge is encapsulated within a series of progressively updated hypotheses. This method enables the system to continuously evolve its understanding of the data distribution, ensuring the detection model remains updated and effective. The integration of an incremental learning algorithm within the DFA-GPE framework and aligns with the stages depicted in the Table 3.1. The table explains how the hypothesis h_t is projected and evolves through each stage.

Table 3.1: The incremental learning procedure alignment with each stage of DFA-GPE

The component	The process	Incremental learning procedure
Data Stream Source and Most Recent Chunk	The reception of a new data chunk D_t at each time step t aligns with the framework's input stage	The hypothesis generation from D_t corresponds to the initial data observation and characteristic notation in the flowchart
Dynamic Feature Selection (DFS) Based on Improved PSO	The selection of relevant features from D_t using an improved PSO algorithm is consistent with the DFS component of the framework	The enhancement of the hypothesis h_t with insights on feature relevance reflects the process of feature optimization and selection in the flowchart
Ensemble of Base Classifiers and Combiner Graph	The use of selected features to train base classifiers and their combined outputs is a core operation of the DFA-GPE	This operation and the resulting hypothesis h_t are in harmony with the ensemble learning and hypothesis amalgamation represented in the flowchart
Ensemble Update Block	Refining the ensemble with new data and feedback is depicted in the flowchart as the Ensemble Update Block	The continual refinement of h_t is accurately projected as part of the incremental learning procedure
Drift Detection Mechanism (DDM)	Monitoring for concept drift and updating the model aligns with the Drift Detection component	The adjustment of h_t in response to concept drift resonates with the adaptability aspect of the DFA-GPE framework
Model Training, Error Calculation, and Validation	Training the model with D_t, calculating error rates, and model validation align with the system's continuous learning and validation phases	The evolution of h_t based on performance mirrors the flowchart's depiction of iterative learning and validation
Model Retention or Adjustment	The decision to retain or adjust the model based on performance encapsulates the flowchart's feedback loop for optimization and refinement	The determination of h_t at time t is a critical endpoint in the incremental learning process

Therefore, the adopted incremental learning procedure as illustrated in Algorithm 6, Upon receiving new data, the framework learns from it by updating its understanding and forming a new hypothesis, without needing to keep old data. This process of continuous learning and adaptation allows the framework to stay updated with changes in data over time.

Algorithm 6: The pseudocode of incremental learning for intrusion detection

Input:
Data Stream
Validation Data Stream
Output:
Trained Model M
Start:
Initialize Model M
Initialize empty list of hypotheses H
Initialize error rate e
For each time step t:
Receive new data chunk D_t from Data Stream
If D_t is empty:
Output: Final Model M
End Algorithm
If $t > 1$:
Use $h_{(t-1)}$ to establish new hypothesis H_t based on D_t
Else:
Establish initial hypothesis h_1 based on D_1
Add h_t to H
Train Model M with D_t using h_t
Calculate error e_t on D_t using Model M
Print e_t // Optional: Can be used to keep track of error during training
Evaluate Model M on validation data
Calculate validation error e_{val}
Print e_{val} // Optional: Can be used to keep track of validation error
If Model M performance improves (e.g., if e_{val} decreases):
Save Model M
Update e to e_{val}
Else:
Adjust learning parameters
End

3.3.1.4 Imbalance Data Handling

In DFA-GPE framework, the challenge of data imbalance is addressed through the implementation of the SMOTE, an effective method for managing the variance in class

distribution. In scenarios where significant imbalances exist between the quantities of data points across different classes, the integration of SMOTE into the preprocessing phase of the DFA-GPE framework becomes crucial.

The procedure for implementing SMOTE shown in Figure 3.3, the framework begins with the identification of data imbalance in the received data chunk, D_t. Once an imbalance is detected, SMOTE is employed to oversample the minority class data points. This is achieved by creating synthetic examples rather than merely duplicating existing instances, a method preferred over random oversampling. The rationale behind this preference is rooted in the tendency of random oversampling to lead to overfitting, particularly in cases involving high-dimensional datasets [191]. SMOTE's approach to generating new data points serves to mitigate this risk effectively.

During the application of SMOTE, a minority class data point, denoted as x_i, is selected randomly. From this point, k nearest neighbors within the same class are identified, with k typically set to 5. The distances between x_i and these neighbours are calculated, facilitating the selection of one closest neighbour, denoted as z_i. New synthetic samples, x_{new}, are then generated along the line segment connecting x_i and z_i. This process of synthetic sample generation continues iteratively until a balance in class distribution is achieved. The resulting dataset, a blend of original minority class data and synthetically created samples, forms the foundation for model training within the DFA-GPE framework.

It is crucial to note that the SMOTE algorithm is applied exclusively to the training set of the data. The segregation of data into training and testing sets is conducted in such

a manner that the testing set remains unaffected by any preprocessing, including SMOTE. This untouched testing set, maintaining its original distribution, is critical for ensuring an unbiased evaluation of the model's performance. By applying SMOTE exclusively to the training set and keeping the testing set untouched is crucial for maintaining an unbiased evaluation of the model's performance. The testing set must represent the real-world distribution of data to provide a realistic measure of how well the model generalizes to unseen data. Altering the testing set with SMOTE would distort this distribution, leading to a biased assessment and potentially unrealistic performance results. By preserving the original imbalance in the testing set, the model's true ability to handle imbalanced data in real-world scenarios can be accurately evaluated, ensuring that the results are both valid and reliable.

Post-SMOTE application, the balanced training data undergoes the process of feature selection as part of the DFA-GPE framework. This process is facilitated by an improved PSO algorithm, which selects relevant features from the now-balanced dataset. The ensemble of classifiers within the DFA-GPE framework is subsequently trained on this dataset. This training is crucial as it ensures that the developed models are not predisposed towards the majority class, a common pitfall in scenarios of class imbalance. Finally, the performance of these models is evaluated using the untouched testing set. This evaluation is instrumental in providing a realistic assessment of the model's effectiveness in scenarios reflective of actual operational environments, where class imbalance is a typical challenge.

Given the various approaches to handling class imbalance, SMOTE has chosen due to its ability to generate synthetic samples, thus increasing the representation of the

minority class without the risk of information loss associated with under sampling. Unlike random oversampling, which may lead to overfitting by duplicating existing samples, SMOTE creates new instances that contribute to a more robust model. Although SMOTE can sometimes introduce noise by generating synthetic samples based on irrelevant or redundant features, this risk is mitigated by implementing feature selection beforehand. Feature selection retains only the most relevant and informative features, reducing the likelihood of noise and enhancing the quality of the synthetic samples. While cost-sensitive learning offers another alternative by adjusting misclassification penalties, it involves complex implementation and calibration, which can complicate the modeling process. Therefore, SMOTE was selected as it strikes a balance between effectiveness and simplicity, improving model performance while addressing the common issues associated with other methods [192].

3.3.1.5 Genetic Programming (GP) Combiner

The framework of the ensemble classifier based on the GP-combiner, consists of several components as presented above in Figure 3.2. First, the stream data is fed into the framework and is assumed to be partitioned into a sequence of chunks. Each chunk is to be partially labelled, which means feeding its labelled part to the training sub-block and another sub-block that is intended for classification and detecting the concept drift. Afterwards, there is the core of the GP-combiner which is the ensemble update process which consists of three sub-components, namely, base classifier replacement, classifier induction, and combiner generation and graphs for prediction. The ensemble update process feeds the graph to the ensemble that performs the prediction and keeps the basic classifier pool for the next round. Additionally, in this

approach, utilize the DDM for concept drift detection. DDM is a window-based technique that detects the significant increases in the classification error rate and updates the classifier to adapt to new concepts. This mechanism ensures timely adaptation to concept drift and accurately maintains the framework's performance.

The main reason for choosing Genetic Algorithms (GA) for the combining function in the ensemble is primarily due to their flexibility and effectiveness in evolving optimal solutions over time, which is particularly beneficial in dynamic and non-stationary environments such as intrusion detection systems. GA offers a robust mechanism to discover complex combiner functions that can adapt to the changing characteristics of data streams. This adaptability is crucial for maintaining the accuracy and reliability of the ensemble model as it encounters varying patterns and concept drifts. By using GA, the framework can automatically evolve an expressive combiner function, which allows for more accurate integration of base classifiers' predictions, leading to improved overall performance in detecting intrusions [32].

3.3.1.6 Concept drift and feature drift handling

To handle concept drift within the DFA-GPE framework, the algorithm integrates a DDM that continuously monitors the data stream for signs of drift. The detection process involves tracking the model's performance over time and identifying significant deviations from expected behavior, which indicate concept drift. When a drift is detected, the algorithm initiates a series of updates. Initially, the incoming data tuple is classified using the current ensemble of base classifiers, and if the tuple is labeled, it is stored in a buffer for potential model updates. The DDM then evaluates

whether the current data distribution has changed compared to previously observed data. If a drift is detected, the model flags this and proceeds to update the ensemble. The algorithm partitions the buffered data into training and validation sets, and new classifiers are trained on the most recent data, ensuring that the ensemble adapts to the new concept. The best-performing classifiers, based on accuracy, are selected and integrated into the ensemble, replacing older classifiers as necessary. Additionally, the GP-based combiner function, which integrates predictions from multiple classifiers, is updated to reflect the new data distribution. This ensures that the ensemble continues to make accurate predictions even as the underlying data evolves, maintaining high performance in environments where data distribution is constantly changing.

The DFA-GPE framework addresses feature drift by dividing the incoming data into chunks, performing feature selection dynamically on each group of instances as they arrive. This approach contrasts with static methods, which typically apply feature selection once before classification, potentially overlooking the evolving significance of features over time. By employing an improved variant of VLMO-PSO, DFA-GPE manages feature drift dynamically and efficiently.

The VLMO-PSO algorithm is designed to support variable-length solutions, reducing the complexity of the feature selection process and allowing it to adapt to changes in feature content and length. This adaptability is further enhanced by a smart population initialization strategy based on Bernoulli distribution and symmetric uncertainty, ensuring that the initial solutions are well-distributed across the search space. The algorithm also utilizes a set of transfer functions that map the mobility equation outcomes to the decision space, improving its ability to explore and exploit the feature space effectively.

Additionally, the method decomposes the feature selection objective over certain iterations, preventing the overemphasis of one objective over another. It also employs an ensemble approach, leveraging the collective wisdom of multiple solutions, which leads to a more robust and stable feature selection process compared to relying on a single solution. The entire process is performed dynamically on each data chunk, ensuring that feature relevancy is evaluated as enough instances of the data stream arrive, thereby maintaining the accuracy and adaptability of the model over time.

3.3.2 Objective 2: Multi-Exemplar Particle Swarm Optimization with Local Awareness (MEPSOLA) Algorithm Development

This section focuses on the second objective, the development of a MOO algorithm, an essential component in tackling complex problems with multiple conflicted objectives. It begins with a detailed problem formulation, the focus is on optimizing a series of objective functions, each of which maps potential solutions to real-number values within a defined feasible solution space constrained by specific parameters. Central to this optimization process is the principle of non-domination, which is influential in defining the Pareto front. The section then introduces the Multi-Exemplar Particle Swarm Optimization with Local Awareness (MEPSOLA) algorithm, highlighting its systematic approach to optimization. This includes smart initialization for comprehensive exploration of the solution space, and an iterative process involving interaction validation, multi exemplar interaction, particle updates, and mutation are explored in depth. These elements ensure the algorithm's efficiency in navigating the MOO problem's complex optimization. Additionally, the section discusses the integration of Tabu search for exemplar local search and details the main algorithm procedure, emphasizing the algorithm’s sophisticated approach to MOO.

3.3.2.1 Problem Formulation

Let assume that $f = (f_1, f_2, \dots f_n)$, be a vector of n objective functions that are to be optimized. Each objective function $f_i: X \to \mathbb{R}$ maps a solution $x \in X$ to a real number where X is the set of all feasible solutions defined by a set of constrains $g_j(X)$. $X = \{x \in \mathbb{R} \mid g_j(X) \leq 0, h_k = 0\}, j = 1, \dots m,\ k = 1,2 \dots p$ where m denotes the number of inequality constraint $g_j(X) \leq 0$, p denotes the number of equality constraints $h_k = 0$.

A solution x_1 is said to dominate another solution x_2 or $x_1 \prec x_2$ if and only if: $f(x_1) \leq f(x_2)$ for all $i = 1,2 \dots n$ and $f(x_1) < f(x_2)$ for at least one $1 \leq i \leq n$.

The multi-objective optimization problem based on the concept of non-domination can be formally stated as: Find $x^* \in X$ such that $\nexists x \in X$ for which $x \prec x^*$. All the solution x^* that meet this condition are given in Pareto front $X^* = \{x \in X | \nexists x' \in X, x' \prec x\}$.

The difference between single and MOO is indeed profound. In single-objective optimization, the goal is straightforward: to find a single solution that minimizes or maximizes the objective function. However, in MOO, the condition is more complex due to the presence of multiple conflicting objectives. The flowchart in Figure 3.3, depicts The MEPSOLA algorithm, follows a systematic optimization sequence. Starting with smart initialization, it strategically sets up the population and repository. The algorithm then iterates through cycles of exemplar validation, updates of particle velocity and position, a cost calculation and conditional mutation. A key feature is the incorporation of a conditional Tabu search for exemplar localized refinement. This

iterative process continues until convergence criteria or maximum iterations are reached, leading to the formation of a Pareto front, indicating the optimal set of solutions.

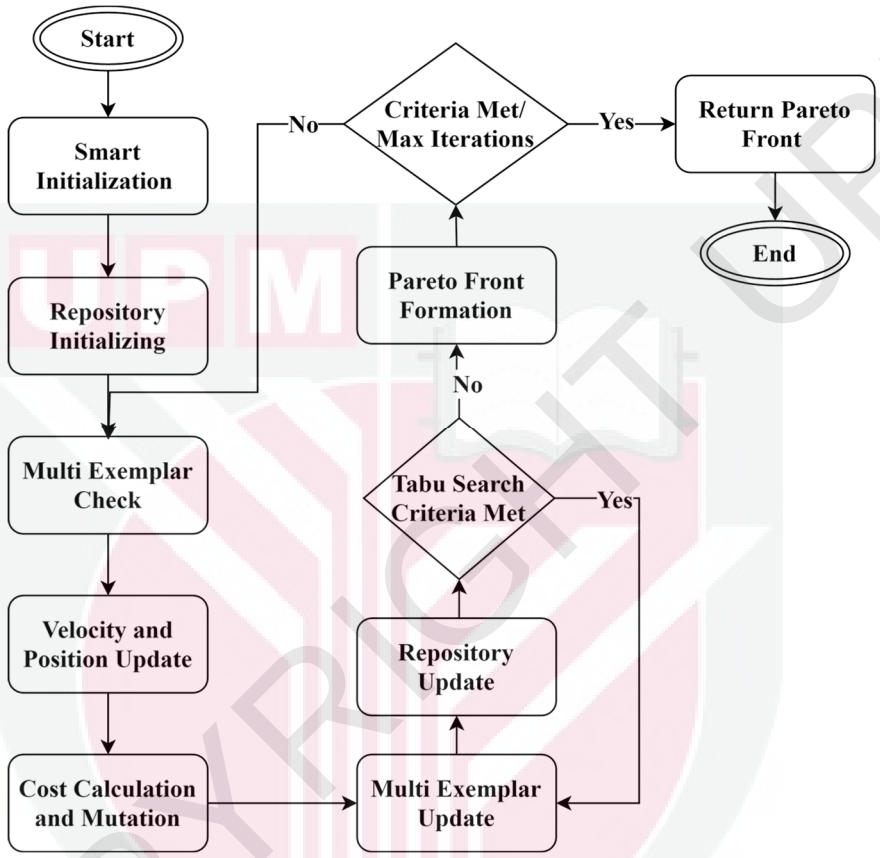


Figure 3.3: Flowchart of the general process for MEPSOLA

3.3.2.2 Solutions Initialization

The initialization phase is crucial, involving equal sampling within the range of each decision variable. This step ensures the initial values are representative of their full range, crucial for the algorithm's performance. By assigning $Range\_values$ to each variable, this equitable distribution guarantees comprehensive exploration of the solution space from the start. Without such balanced initialization, significant areas might be overlooked, potentially leading to suboptimal outcomes.

The Cartesian product applied to these $Range\_values$ creates a set of permutations, encompassing all possible combinations of decision variable values within their full ranges. This comprehensive generation of permutations underscores the depth of exploration undertaken in the solution space. The population is formed by randomly selecting a subset from these permutations, with its size dictated by pop_size. Each particle in this initial population is then given a position in the search space, followed by the initialization of its velocity and the cost. This approach ensures the initial population is unbiased towards any particular solution space region. Notably, the remainder of these permutations is reserved for the conditional mutation phase, rather than relying on random mutation, ensuring an effective mutation.

The Repository's formation involves selecting non-dominated particles from the initial population, using the $get\_particles\_non\_dominated(population)$ function. This step is crucial for preserving a diverse solution set, free from the influence of any dominating traits. Particles are then assigned exemplars through the $exemplar\_selection()$ function, a key step in improving the algorithm's search process, as exemplars act as navigational encouragements for the particles. The algorithm concludes by returning the populated Repository, laying a solid foundation for the computational journey ahead. The detailed process of the initialization phase is depicted in Algorithm 7, highlighting its importance in setting the stage for successful optimization.

Algorithm 7: Pseudocode of generate of first population

```
Input:
 Pop_size: The size of the population.
Output:
 First_pop: The first generated population.
 Repository: A repository of non-dominated particles.
Procedure:
Initialization:
For each decision variable
   Range_values(decision variable) =do equal sampling with the range of decision variable
End
Permutations=apply Cartesian product on Range_values
Positions = random. sample(permutations, pop_size)
Population Setup:
     For each particle in particles:
     Assign a position from Positions to Particle. position.
     Initialize Particle. velocity.
     Calculate Particle. cost.
Repository Formation:
    Form Repository by identifying non-dominated particles from the population using
get_particles_non_dominated(population).
Exemplar Assignment:
    For each particle in particles:
    Assign an exemplar to Particle. exemplar by calling the exemplar_selection() function.
Return:
    Return population and Repository.
End
```

3.3.2.3 Solution Mobility

There are two types of operations for solutions mobility. The first one is solutions interaction with exemplar and the second one is solution mutation. The former involves selecting a multi exemplar and moving the solution toward it and the latter involves changing the solution from predefined range without interaction with other solution.

3.3.2.4 Exemplar and Solutions Interaction

In heuristic optimization algorithms, particularly those akin to PSO, the nuanced interaction between a solution and its exemplar is defined by a specific mathematical

formulas stated in Equation (3.1) [23]. This formula involves multiple components that collectively determine the movement of a solution within the search space. At its core, this interaction involves the update of a solution's velocity, a process influenced by the solution's current velocity, represented by v_i^t. This element of velocity is crucial as it dictates the rate and direction of the solution's movement, fundamentally shaping its direction through the search space. Next, the influence of inertia weight, denoted as w, plays a pivotal role in this process. It determines how much of the solution's previous momentum is carried forward into its new velocity, thus balancing the forces of exploration and exploitation within the algorithm. A larger inertia weight encourages a broader, more explorative approach, while a smaller weight focuses on localized, detailed searches.

$$v_i^{t+1} = w \times v_i^t + c \times r_i \times \left(p_{exmplr(i)}^t - x_i^t\right) \tag{3.1}$$

The interaction is further characterized by a randomized attraction towards the exemplar, represented by $p_{exmplr(i)}^t$. This exemplar, often the best solution found by either the individual or the entire swarm, serves as a point of attraction. The level of this attraction is influenced by a random factor, r_i, a number generated between 0 and 1, introducing unpredictability and variation in the movement of the solution. This randomness is critical in preventing the solution from becoming trapped in local optima. An acceleration coefficient c, modifies this attraction, steering the solution more decisively towards promising regions within the solution space. The influence of this coefficient is crucial in dictating the aggressiveness of the solution's movement towards the exemplar. Finally, the interaction takes into account the solution's current position, x_i^t, by subtracting it from the exemplar's position. This difference is integral

as it quantifies the relative position of the solution to its guiding beacon, thus directing the update of velocity.

Through this well-orchestrated interaction, each solution in the population is methodically guided towards optimal or near-optimal regions. The exemplar acts as a beacon, directing the solution's journey, while the inertia and random elements ensure a diverse and comprehensive exploration of the solution space. This balance of individual movement influenced by personal and collective experiences encapsulates the essence of population-based optimization techniques, leading to effective and efficient discovery of solutions.

Where v_i^t denotes the current velocity of the solution, and r_i denotes a random number between 0 and 1, and $p_{exmplr(i)}^t$ denotes the position of the exemplar of dimension d for solution i.

3.3.2.5 Solution Mutation

The mutation is a concept borrowed from evolutionary algorithms. It serves as a mechanism to introduce diversity into the population. While not a standard component of PSO, mutation can be incorporated in PSO to prevent premature convergence. In the proposed algorithm, adopt picking randomly one of the decision variables and replacing its value with another from its originally generated range values.

3.3.2.6 Exemplar Selection

The pseudocode in Algorithm 8, illustrates the procedure for selecting an exemplar for a given particle in a computational algorithm. This process is critical for guiding the search behavior of particles in an optimization algorithm. The role of conditional calling of local search, the temporary usage of an exemplar, and the usage of multi-exemplar are key elements that significantly influence the algorithm's effectiveness and efficiency.

The algorithm begins with an initialization step, where a $Threshold\_repetition$ value is set. This threshold is crucial as it defines the maximum frequency with which an exemplar can be used before triggering alternative search mechanisms. Additionally, $Used\_exemplars$ is initialized as an empty dictionary. This dictionary will track the usage frequency of each exemplar, thereby playing a vital role in managing the diversity of the search process.

The primary step involves choosing a random Exemplar from the Repository. The randomness here is vital for ensuring an unbiased selection process. The selection process then checks whether the chosen Exemplar is already in $Used\_exemplars$. If it is not, the exemplar is added to $Used\_exemplars$, with its count set to 1. This step ensures that each Exemplar gets a chance to influence the search process. If the Exemplar is already used but its count is less than $Threshold\_repetition$, its count is incremented. This mechanism allows for the repeated use of effective exemplars but within a controlled limit.

The role of conditional calling of local search is highlighted when the count of an exemplar reaches the threshold. In this case, instead of continuing to use the same exemplar, the algorithm calls Algorithm 9, which is responsible on Tabu search. This shift to a local search method, triggered by the threshold, ensures that the search does not decay due to over-reliance on a particular Exemplar. It introduces a new dimension to the search process, allowing for exploration in different regions of the solution space. The temporary usage of an exemplar, as governed by the $Threshold\_repetition$ and the use of $Used\_exemplars$, ensures that no single exemplar dominates the search process. This temporary usage allows for a dynamic and adaptive search strategy, where exemplars are rotated based on their efficacy and frequency of use. It prevents premature convergence and maintains the diversity of the search process. The utilization of multiple exemplars, as facilitated by the dynamic selection process, plays a pivotal role in exploring various regions of the solution space. By allowing particles to be influenced by different exemplars over time, the algorithm benefits from a multi-faceted search approach. This diversity in guidance helps in effectively navigating the solution landscape, aiding in the avoidance of local optima and enhancing the probability of finding global optima. Finally, the algorithm returns the selected exemplar, which is then used to guide the movement of the particle in the optimization process.

Algorithm 8: Pseudocode of exemplar selection

```
Input:
- particle: The particle for which the exemplar is to be selected.
Output:
- Exemplar: The selected exemplar for the input particle.
Procedure:
Initialization:
  - Set a Threshold_repetition value.
  - Initialize Used_exemplars as an empty dictionary.
Selection of Exemplar:
   Choose a random Exemplar from the Repository.
    If Exemplar is not in Used_exemplars:
      Add Exemplar to Used_exemplars and set its count to 1.
    Else, if Used_exemplars[Exemplar] is less than Threshold_repetition:
      Increment the count of Exemplar in Used_exemplars.
    Otherwise:
     Select a new Exemplar by applying tabu_search(particle. position) Algorithm 9.
Return:
     Return the selected Exemplar.
End
```

3.3.2.7 Tabu Search Local Search

Tabu search operates through a sequence of steps to navigate through the solution space of an optimization problem as illustrated in Algorithm 9. It starts by setting the $Initial\_solution$ as both the $Current\_solution$ and the $Best\_solution$. This dual assignment shows the beginning of the search process from a defined point, with the $Best\_solution$ being a placeholder for the optimal solution encountered during the search. A $Tabu\_list$ is also announced, initially as an empty list. The function of this list is to record the solutions that have been recently explored. The maintenance of this list is critical as it helps in avoiding repetitive exploration of the same solutions, thereby preventing the algorithm from being trapped in cyclic patterns.

Another key component is the $Tabu\_list\_Max\_size$, a predefined value that limits the number of entries in the Tabu list. This constraint ensures that the list only retains a certain number of recent solutions, balancing memory efficiency with the effectiveness

of avoiding recent solutions. The core of the algorithm is a loop that continues until a specified stopping criterion is met. Within this loop, the algorithm generates a $Neighborhood$ of solutions surrounding the $Current\_solution$. This neighborhood is a set of solutions that are close to the current solution, typically representing slight modifications or variations of it. From this $Neighborhood$, the algorithm selects the $Best\_neighbor$, which is the most promising solution that is not in the $Tabu\ list$. This step is crucial as it guides the search towards areas of the solution space that are yet unexplored or hold potential for better solutions. Upon identifying the $Best\_neighbor$, the algorithm updates the $Current\_solution$ to this new solution. Concurrently, it checks if this best_neighbor is an improvement over the $Best\_solution$ found so far. If it is, the $Best\_solution$ is updated to reflect this new, better solution.

Finally, the $tabu\_list$ is updated with the $Best\_neighbor$. This update involves adding the new solution to the list and ensuring that the list size does not exceed the predefined $Tabu\_list\_Max\_size$. This management of the Tabu list is essential for the algorithm's ability to efficiently navigate through the solution space without revisiting recently considered solutions. The algorithm concludes by returning the $Best\_solution$ once the stopping criteria are met. This solution is the optimal solution found by the algorithm in its exploration of the solution space, guided by the strategic avoidance of recently explored solutions through the Tabu list.

Algorithm 9: Pseudocode Tabu Search (local search)

```
Input:
Initial_solution
Ouput:
Best_solution
Start:
Current_solution = Initial_solution
Best_solution = Initial_solution
Tabu_list = Empty list
Tabu_list_Max_size = some predefined value
while stopping_criteria_not_met:
      Neighborhood = generate_neighbors(current_solution)
       Best_neighbor = find_best_candidate (neighborhood, tabu_list)
       Current_solution = Best_neighbor
       if better_than(Best_neighbor, Best_solution):
            Best_solution = Best_neighbor
       Update_tabu_list(Tabu_list, Best_neighbor, Tabu_list_Max_size)
   return Best_solution
End
```

3.3.2.8 MEPSOLA

The general process of MEPSOLA, outlined in Algorithm 10, begins with establishing the initial population, $First\_population$, and a repository for non-dominated particles, based on Algorithm 7, intelligent and thorough initialization. This ensures a diverse and representative start for the optimization.

In the main loop of the algorithm, which continues until predefined stopping criteria are met, each particle in the population undergoes a series of steps. An Exemplar Check is first conducted for each particle. If a particle's position is identical to its exemplar's, the exemplar is adjusted using the get_close_number function. This step is crucial as it maintains diversity and avoids stagnation in the search process. The particle's velocity and position are updated next, reflecting the ongoing dynamics of the algorithm. These updates are crucial for navigating the solution space and are influenced by the exemplar's guidance. Following this, each particle's cost is

calculated, and a mutation is potentially applied, introducing variability and aiding in exploring new areas of the solution space.

If a particle's cost is better than that of its exemplar, the exemplar is updated using the $exemplar\_selection$ method from Algorithm 8. This method selects exemplars in a manner that balances exploration and exploitation, using a Tabu list to avoid repetitive exploration and enhance the search's diversity. After processing all particles, the Repository is updated with non-dominated particles from the population. This update is a crucial step, as it maintains a set of optimal solutions found so far, influenced by the diverse and dynamic exemplars. The algorithm concludes by forming the $Pareto\_front$ from the non-dominated solutions in both the repository and the population, as determined by their respective costs and positions. This front represents the set of optimal solutions, balancing various objectives. The final step is returning the pareto front, marking the end of the optimization process.

Algorithm 10: Pseudocode Multi-Exemplar Multi-Objective Particle Swarm Optimization (MEPSOLA)

```
Input:
First_population: Initial set of particles.
Repository: Storage of non-dominated particles.
Output:
Pareto_front The final set of optimal solutions.
Procedure:
Initialization using Algorithm (7)
    Set population to First_population.
    Initialize repository as Repository.
    Initialization of exemplar
Main Loop:
    While the stopping criteria are not met:
      For each particle in population:
        Exemplar Check:
          If particle. position is equal to particle. exemplar:
            Adjust particle.exemplar using get_close_number(particle. exemplar).
        Velocity and Position Update:
          Update particle. velocity using update_velocity(particle).
          Update particle.position using update_position(particle).
        Cost Calculation and Mutation:
```

Calculate particle.cost.
Apply mutation to the particle with apply_mutation(particle).
Exemplar Update:
If particle. cost is better than particle. exemplar. cost:
Update particle. exemplar using exemplar_selection() Algorithm (8)
Repository Update:
Update repository with non-dominated particles from population using get_particles_non_dominated(population).
Pareto Front Formation:
Form Pareto_front from the non-dominated solutions in repository and population using get_solutions_non_dominated(repository, population).
Return:
Return Pareto_front.
End

3.3.2.9 Exploration Mechanisms

The exploration behavior of the proposed algorithm is resulted from several mechanisms that can be stated as follows.

1. The incorporation of smart or fair initialization to enable fair sampling of initial population in the objective space.
2. The incorporation of Tabu search to be conducted on each exemplar every T period.
3. The incorporation of predefined range mutation which is an evolutionary mechanism.
4. The incorporation of grid aware solutions selection mechanism for selecting solutions as a part of the repository.

3.3.3 Objective 3: Adaptation of Variable Length Multi Objective Particle Swarm Optimization (VLMO-PSO) for Dynamic Feature Selection

This section outlines the research's third objective: the vital adaptation of the developed MOPSO for DFS in IDS operating on data streams. It focuses on addressing feature drift and balancing two conflicting objectives: minimizing memory usage and maximizing the accuracy of threat detection. The Variable Length Multi Objective Particle Swarm Optimization (VLMO-PSO) is designed to select the feature sets

dynamically, ensuring an optimal balance between the IDS's resource efficiency and its ability to accurately detect intrusions. The algorithm is thoroughly described, from the initialization of the solution space key to the convergence and quality of results to the complex interactions within the population of solutions. Integral to the algorithm is the selection of exemplars and the decomposition of objectives, alongside the introduction of mixed transfer functions that facilitate the movement of solutions within the search space. The chapter also explores the mutation operator, which improve diversity and prevent premature convergence, and concludes with a strategic voting mechanism for selecting the most suitable features. This comprehensive approach underscores the algorithm's flexibility and robustness in the face of the dynamic challenges posed by cybersecurity threats.

3.3.3.1 Problem Formulation

In the field of IDS, a conflict arises between two primary objectives: reducing memory usage while maintaining high accuracy. This thesis proposes an MOO algorithm with DFS that is specifically designed to achieve an optimal balance between these objectives. By employing VLMO-PSO principles, the algorithm allows the solution vector to have varying lengths (feature sets), thereby reducing the complexity of the PSO.

The target of this approach centers on its ability to dynamically adapt to changing data streams and requirements, ensuring that the IDS remains effective and efficient over time. The problem formulation for the proposed research can be articulated in two distinct parts. The first part involves the development of an online decision-making

MOO algorithm based on VLMO-PSO, and the second part adapts this algorithm to support DFS in IDS, with a focus on addressing the inherent conflict between memory reduction and accuracy.

The objective is to adapt the developed MOO algorithm to support DSF in IDSs, balancing the conflict between memory reduction and accuracy. The problem formulation can be described in the following steps:

1. Dynamic Feature Selection: the gap in IDS, there is a critical need for selecting relevant features within specific time intervals, defined as $DFS(t_1, t_2)(D) = \Phi$ where $\Phi \subseteq X$ and Φ provides the most relevant features in X throughout t_1, t_2. The $t_2 > t_1$ and $t_2 - t_1 = \Delta$ represent the minimum period without feature drift in the data stream.
2. Integration with MOO: The challenge lies in adapting the MOO algorithm to effectively handle the dynamic nature of feature selection in IDS. The objectives include minimizing memory usage and error rate, addressing the trade-off between these conflicting goals.
3. Constraints for Feature Selection: the emphasis is on the limitations or restrictions placed on the feature selection process. These constraints are designed to ensure that feature selection occurs within the available feature space, considering the specific limitations imposed by memory and accuracy requirements.

3.3.3.2 Solution Space

The solution space contains the encoding of the decision of FS. Assuming that we have a total of N candidate features, any solution should select a subset of the total features. Hence, the selection is a whole number that contains the index of the selected feature.

For example, let's assume $x = (1,5,40)$ means that the selection of only three features with the index values of 1, 5, and 40. The size of the solution space is equal to $2^N - 1$.

It is observed that when N is large, the number of candidate solutions increases exponentially. In other words, the order of selected features is $O(2^N)$. This is necessitating the incorporation of some constraints to enhance optimization efficiency. This can be accomplished by calling a process that maps the individual features to subsets of features where each sub-set is regarded for selection or non-selection instead of giving the decision for the original features. In other words, assuming that the original features set is X. X is then rewritten as shown in the equation below [55]:

$$X = \bigcup_{k=1}^{K} X_k \tag{3.2}$$

For any two indices, i, j where $i, j \in \{1, K\}$ and $i \neq j$.

$$X_j \bigcap X_i = \emptyset \tag{3.3}$$

Where K denotes the number of partitions.

Then the size of the solution space is reduced from $O(2^N)$ to $O(2^K)$, $K \ll N$. Identifying a straightforward strategy for assigning individual features to specific subsets presents a challenge, and essential for streamlining the feature selection process.

3.3.3.3 VLMO-PSO General Flowchart

This section, describes the operational flow of the proposed algorithm. The flowchart in Figure 3.4 outlines the algorithm's procedure. The process starts with the

initialization of solutions using a Bernoulli distribution, chosen for its ability to have an equal probability of feature selection in the initial population. Solutions are then ranked by symmetric uncertainty and assigned to the PSO division. A series of iterations follows, during which solutions are propelled towards their exemplar using a mixed transfer function, interspersed with mutations occurring at an epsilon probability. The algorithm periodically assesses whether the criteria to switch the objective for the exemplar have been met. These steps are iterated until the process concludes, either by reaching the maximum number of iterations or by attaining convergence. The final stage involves making the feature selection decision through a voting process.

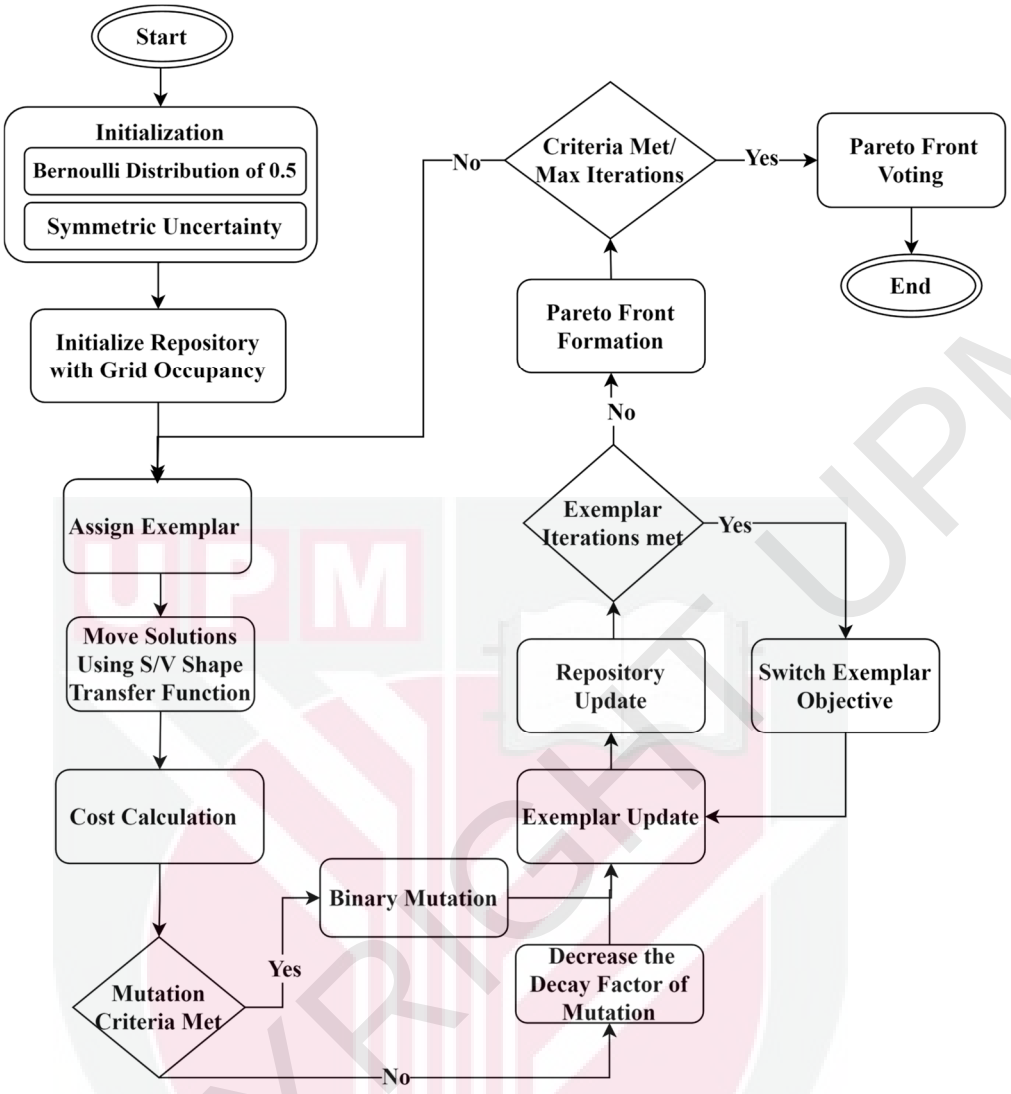


Figure 3.4: A general flowchart of the multi-objective variable length particle swarm optimization

3.3.3.4 Generate First Population (Solutions Initialization)

The first phase in DFS is solutions initialization. It aims to initialize the first population. The initialization plays a critical role in the convergence of the algorithm and the quality of the obtained solutions. Therefore, a heuristic approach is employed for initialization. Considering that the feature space consists of the features $\{f_1, f_2, \dots, f_N\}$, random solutions are generated that contain total decisions for selecting every candidate feature. To accomplish this, it is required that every solution contains

at least one predefined feature while the remaining are just generated using the bernoulli distribution with a probability of 0.5. In the case of high dimensional solution space, the features are ranked based on the symmetric uncertainty and they are allocated into divisions where the search is enabled in divisions of highest symmetric uncertainty. Next, solutions are then evaluated and ordered by descending symmetric uncertainty and order them in descending order. The solutions that have higher symmetric uncertainty are fed into the first population, and the remaining solutions are excluded. The pseudocode of the initialization is given in Algorithm 11.

Algorithm 11: Pseudocode to Generate the First Population

Input:
Code of features $\{f_1, f_2, \ldots, f_N\}$
Initial chunk
popSize
Output:
FirstPop
Start:
Pop = []
For i=1 until 2× popSize
    $X = []$;
    Generate feature code $l = \mathrm{mod}(i, N)$
    Add l to the solution X
    For each from $\{1, .. N\}$ where j≠ l
        Generate random number ε between 0 and 1
        if (ε >0.5)
            add j to the X
        end
    End
    Add X to Pop
End
Sort Pop according to higher symmetric uncertainty using the initial chunk
Keep the highest Pop solutions and delete the remaining
End

3.3.3.5 Population Interaction

The role of the population interaction is to use current generation solutions for generating off-springs or future generations. This is achieved through two primary

operations. The first one is the mobility equation that moves the solutions toward a leader or exemplar. The second one is the mutation that is responsible for changing existing solutions in order to enable more exploration while searching.

3.3.3.6 Exemplar Selection and Objective Decomposition

In MOPSO, usually the exemplar selected based on an algorithm that operates as follows: First, the particle initiates a random number and compared with learning probability named PC, which calculated using Equation (3.4). Second, it sorts the solutions according to their single objective costs ascendingly and assigns a rank to each solution according to the sorting. The selection of the exemplar is guided by the pseudocode presented in Algorithm 12. The calculation for each particle i is represented by the following equation [23]:

$$\mathrm{PC}(i) = 0.05 + 0.45\frac{exp^{10\left(\frac{\mathrm{rank}(i)-1}{S-1}\right)}}{exp^{10}-1} \tag{3.4}$$

Where:

$PC(i)$ is the learning probability for the i^{th}particle ranges from 0.05 to 0.5.

i is the index of the particle.

exp denotes the exponential function.

S denotes the size of the population.

Algorithm 12: Pseudocode of exemplar selection method for single objective

```
Input:
Population without exemplar
Output:
Population with exemplar
Start:
For each particle in the population:
   For d in particle.length:
      If rand() > particle.PC :
         particle.exemplar(d) = particle(d)
      Else:
         Select 2 random particles from the pop and apply 2 tournaments on them
         particle.exemplar(d) = winner_particle(d)
End
```

In the MOO setting, the exemplar selection method is operate as described in Algorithm 13. As it is provided, the algorithm maintains two aspects of performance, exploitation by using objective decomposition which changes the objective of exemplar every 10 iterations and exploration by using the grid criteria in the repository and prioritizing solutions with less crowded grid to be selected as exemplar.

Algorithm 13: Pseudocode of exemplar selection method for multi-objective

```
Input:
Population without exemplar
Rep
Output:
Population with exemplar
Start:
Choose random objective for exemplar
If (mode(iteration, 10) == 0)
      Change objective of exemplar
End
Sort the solutions in rep ascendingly by their grid occupancy Assign a rank to each solution
Update the PC for each solution in Rep using formula (3.4)
Normalize PCs for all rep solutions
For d in Particle.length:
  Particle.exemplar(d) = rep[rouletteWheel(Pcs)].Position(d)
End
End
```

3.3.3.7 Mixed transfer functions of Binary Conversion

The solutions are represented based on their indices (variable representation). However, in order to move the solutions, we need a fixed dimension. Hence, we convert them temporarily to fixed representation. This is done by allocating an element for each feature and using the values of 1 and 0 to indicate activating or deactivating the feature, respectively. Before any solution is moved, its representation is changed to binary. Each solution i is assumed to have an existing velocity v_i^t, the solution velocity changes calculated using Equation (3.5) [23].

$$v_i^{t+1} = w \times v_i^t + c \times r_i\left(p_{\text{exmplr}(i)}^t - x_i^t\right) \tag{3.5}$$

Where:

v_i^t denotes the current velocity of the solution.

r_i denotes a random number between 0 and 1.

$p_{\text{exmplr}(i)}^t$ denotes the position of the exemplar of dimension d for solution i.

After calculating the velocity, the transfer function used to map the velocity to a new solution using one of the equations given in Table 3.2, and the mapping is represented in Equation (3.6) [27].

$$x_i(t+1) = \begin{cases} 0 & \text{If rand}(0,1) < \mathrm{T}\left(x_i(t+1)\right) \\ 1 & \text{If rand}(0,1) \geq \mathrm{T}\left(x_i(t+1)\right) \end{cases} \tag{3.6}$$

Where: $T = S_i \ , \ i = \lfloor \text{uniform}(1,8) \rfloor$

Table 3.2: The mixed S-shaped and V-shaped transfer functions families [57]

S-shape	V-shape
$T(x) = \frac{1}{1+e^{-2x}}$	$T(x) = \left\|\text{erf}\left(\frac{\sqrt{\pi}}{2}x\right)\right\| = \left\|\frac{\sqrt{2}}{\pi}\int_0^{(\sqrt{\pi}/2)x} e^{-t^2}dt\right\|$
$T(x) = \frac{1}{1+e^{-x}}$	$T(x) = \|\tanh(x)\|$
$T(x) = \frac{1}{1+e^{(-x/2)}}$	$T(x) = \left\|(x)/\sqrt{1+x^2}\right\|$
$T(x) = \frac{1}{1+e^{(-x/3)}}$	$T(x) = \left\|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right\|$

3.3.3.8 Mutation Operator

In addition to length changing the mutation has the role of changing some elements in the solution to avoid local minima. This is done by generating a random number between 0 and 1, then comparing it with a small value ε, then changing the corresponding element from zero to one or vice versa when the random number is lower than ε. The value of ε is initiated as $\varepsilon_0 = 0.2$ and it decreases with a decay factor of 0.9. The decay factor decreases using the following Equation 3.7:

$$\varepsilon_n = \varepsilon_0 + n\left({}^{\varepsilon_{final} - \varepsilon_0}/_{G-1}\right) \tag{3.7}$$

Where:

Initial mutation rate ε_0=0.2

Final mutation rate ε_{final}=0.9

G refer to the total number of PSO generations

n represent the iteration number started from $0 \leq n < G$.

3.3.3.9 Voting for Solutions Selections

The objective of the solution selection process is to choose a single solution from the Pareto front to serve as the FS solution. Since our optimization algorithm focuses on two objectives - accuracy and reduction ratio of features - the obtained solutions display varying degrees of quality concerning these objectives. Consequently, choosing one solution will necessitate a trade-off between the two objectives, leading to the prioritization of one over the other. To address this issue, an algorithm is proposed that utilizes the entire set of solutions for FS, employing a voting-based approach. The pseudocode is outlined in Algorithm 14.

Algorithm 14: The Pseudocode of Voting-Based Feature Selection Using Pareto Front

```
Input:
Pareto front
Output:
Selected Features
Begin:
Initialize a Histogram with the size corresponding to the number of features
For each solution in the Pareto front:
  For each selected feature within the solution:
  Increment the Histogram entry at the index of the selected feature
  End inner loop
End outer loop
Determine the maximum frequency maxFreq and minimum frequency minFreq in the Histogram
Calculate the threshold by averaging maxFreq and minFreq: Threshold = (maxFreq + minFreq) / 2
Identify the indices of Histogram entries greater than the Threshold to obtain the SelectedFeatures
Return SelectedFeatures
End
```

The voting-based FS algorithm using the Pareto front presents numerous benefits. Firstly, it ensures a balanced trade-off between two objectives the accuracy and reduction ratio of features - by taking into account the entire set of solutions, thus preventing the over-prioritization of one objective over the other. Additionally, the

algorithm employs an ensemble approach by leveraging the collective wisdom of multiple solutions, leading to a more robust and stable FS compared to relying on a single solution. Furthermore, the adaptive threshold, calculated based on the maximum and minimum frequencies in the histogram, allows for a flexible selection criterion, which is particularly useful when dealing with diverse datasets or optimization problems. This adaptability, combined with the algorithm's applicability to various optimization problems and datasets, results in a versatile and scalable approach. One of the key advantages of considering multiple solutions from the Pareto front is the potential to reduce overfitting, as the algorithm is less likely to select a solution that performs well only on a specific training dataset. Consequently, this can contribute to improved generalization performance, enhancing the overall effectiveness of the approach to unseen data. Moreover, the algorithm generates a histogram representing the frequency of selected features across all solutions. This provides an easily interpretable insight into the importance of each feature, making it simpler to understand the context and relevance of the selected features in relation to the problem at hand.

3.3.3.10 Overall Framework

The general flowchart of the proposed IDS, as referred to in Figure 3.5, starts with receiving the most recent data chunk from the data stream. This chunk is processed by the DFS algorithm, which is based on the Variable Length Multi-Objective Particle Swarm Optimization (VLMO-PSO). The selected features are then split, and SMOTE is applied to balance the data. The processed data is used to train and validate the ensemble of base classifiers, which are combined using a combiner graph. DDM

continuously monitors the data for concept drift. If drift is detected, the system flags it, and the ensemble is updated through classifier replacement, induction, and combiner generation. The final output is the classification and alert, which is stored in the alert repository. This process repeats for each incoming data chunk or terminate by user, ensuring that the IDS adapts to changing conditions in real-time.

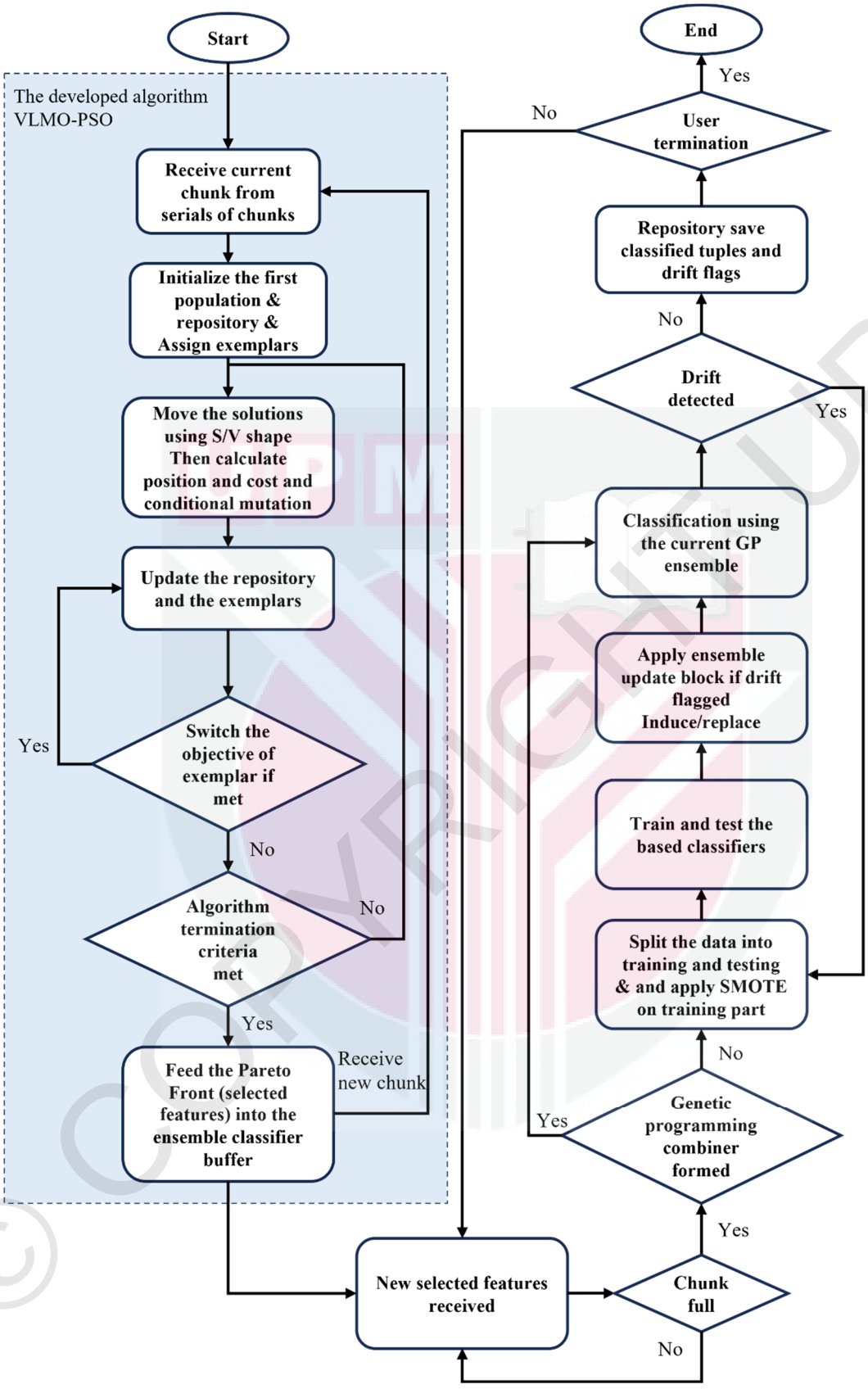


Figure 3.5: General Flowchart of the Proposed IDS Framework for Objectives 1 and 3

3.4 Summary

This chapter presents the methodology designed to address the complex challenges in IDS, with a particular focus on feature drift, concept drift, class imbalance, and high dimensionality. The background of methodology is established by exploring various foundational concepts and techniques integral to the approach. These include the GP-Combiner as classifier aggregation function, the MOPSO algorithm, Tabu Search for local search, the DDM for identifying changes in data streams, and the SMOTE for addressing class imbalance issues.

The methodology's design is outlined, structured around three primary objectives. The first objective involves developing a framework for stream-based IDS, emphasizing dynamic feature-awareness based on DFS and DFA-GPE, and incremental learning procedures. The chapter also discusses methods for handling imbalanced data based on SMOTE, and the integration of GP-Combiner for enhanced model performance in terms of the concept drift challenge.

The second objective focuses on developing a MOO algorithm, detailing the processes of solution initialization, mobility, interaction, mutation, and exemplar selection. The incorporation of Tabu Search for local search into this framework is highlighted, along with an exploration of various mechanisms in the algorithm.

The third objective is the adaptation of this algorithm for DFS in IDS, proposing a VLMO-PSO approach tailored to handle the complex problem optimization and evolving nature of feature selection in IDS.

This chapter presents a comprehensive methodology to enhance IDS by addressing challenges like feature drift, concept drift, class imbalance, and high dimensionality. The methodology integrates advanced techniques such as DFS, MOPSO, and Tabu Search to create an adaptive framework. This framework enables efficient feature selection, incremental learning, and improved performance in evolving data environments, ultimately providing a more robust and adaptive IDS solution.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

The field of IDS is defined by the ever-evolving nature of network threats and the dynamic characteristics of data streams. Traditional IDS struggle with the inherent variability and unpredictability of such data, encountering challenges like feature drift, concept drift, high dimensionality, and class imbalance. Therefore, continuous adaptation is crucial for IDS to maintain accuracy and efficiency in real-time threat detection, necessitating a robust system capable of selecting important features within a non-stationary data stream scenario. This chapter is organized into two crucial parts as follows, with the results presented based on the series of the development process.

Firstly, the experimental design and results of MEPSOLA are introduced; please refer to Figure 3.1, Stage 2. This step aims to address and optimize complex and conflicting objectives efficiently. MEPSOLA employs a smart initialization phase, designed to thoroughly explore the research space. Following this, a multi-exemplar technique, enhanced by an exemplar selection mechanism through conditional interaction with local search, is utilized. This foundational method lays the groundwork for the development of online decision-making processes for DFS. The performance of the MEPSOLA algorithm is assessed using a spectrum of multi-objective mathematical functions as FON, KUR, ZDT1, ZDT2, ZDT3, and ZDT6 to evaluate its optimization efficacy. The analysis extends to the efficacy of the MOO approach, employing

metrics such as set coverage, hyper-volume, the number of non-dominated solutions, generational distance, and the delta metric. MEPSOLA is compared with well-established benchmarks like NSGA-II, NSGA-III, and MOEA, positioning these frameworks within the current state-of-the-art approaches and substantiating their effectiveness.

Proceeding from MEPSOLA, the MOPSO is refined and adapted to serve as the Variable Length Multi Objective Particle Swarm Optimization (VLMO-PSO). Through this adaptation, DFS is enabled to dynamically select feature sets, striking an optimal balance between resource efficiency and detection accuracy within IDS frameworks. This phase of optimization is illustrated in Figure 3.1, Stage 3, marking a pivotal advancement towards overcoming the presented challenges.

Secondly, after MOPSO adapted the experimental design and results of the DFA-GPE are presented in this chapter. This incremental learning IDS framework is integrated with several key components: the GP combiner, the developed MOO for DFS, drift detection mechanisms, and class imbalance handling. The integration of these elements into a unified system ensures that the algorithm remains effective amidst the complex optimization challenges posed by cybersecurity threats, as illustrated in Figure 3.1, Stage 1. Two datasets, namely the TON_IoT Telemetry Dataset 2020 and HIKARI 2021, are utilized to precisely assess the classification efficacy of the DFA-GPE IDS framework. The evaluation incorporates several metrics, including accuracy, precision, recall, F1 score, memory reduction, and feature counts. DFA-GPE is compared with Random Forest (RF), Logistic Regression (LR), Recursive Feature

Elimination (RFE), and NSGA-II for feature selection methods. This comparative analysis positions DFA-GPE in the current landscape of state-of-the-art approaches.

4.2 Evaluation

This section presents various datasets to evaluate the classification performance of the IDS framework, followed by a presentation of mathematical problems to assess the MOPSO. Then introduce the classification measures utilized for binary and multi-class classification performance of IDS, and then present the memory consumption and feature count measures. Finally, the section concludes with an assessment of the MOO approach using five key metrics such as, set coverage, hyper-volume, number of non-dominated solutions, generational distance, and delta metric.

4.2.1 Real Datasets

This subsection presents the two utilized recent datasets in two different fields to evaluate the performance of the IDS, the HIKARI 2021 dataset and TON_IoT Telemetry 2020 dataset.

4.2.1.1 HIKARI Dataset 2021

The HIKARI-2021 dataset, introduced by [193] and released in August 2021. The dataset stands as a response to the critical need for modern and diverse datasets in the field of artificial intelligence (AI), particularly for improving IDS. The dataset was built from real data collected in a laboratory, specifically designed to develop network

traffic and classification models relevant to network intrusion detection. A key attribute of the HIKARI-2021 dataset is its applicability to various machine learning methods. Its feature set originally comprised 83 features.

The creation of the HIKARI-2021 dataset involved a unique approach to simulate real-world conditions. A specialized lab was set up where network traffic was recorded without any filtering or firewall, thereby capturing a mix of benign and potentially malicious traffic. This setup aimed to mimic human behaviour by running scripts using Selenium to simulate browser activity. Additionally, attack data was generated on separate machines, with attacks categorized into types such as Brute Force, Brute Force-XML, and probing as shown in Figure 4.1. The dataset was labelled using Zeek software, which produced two labels: (traffic category) to denote the type of traffic and "label" to indicate whether the traffic was benign 0 or an attack 1.

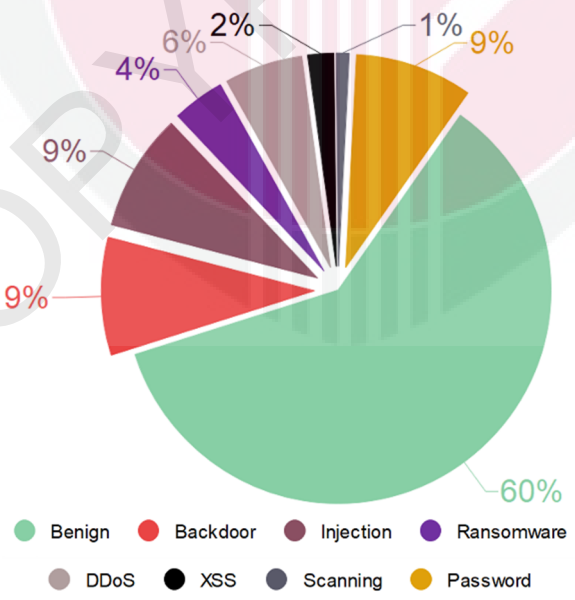


Figure 4.1: The data distribution in HIKARI dataset

In the context of feature drift, the authors also introduced a statistical analysis of the HIKARI-2021 dataset using a correlation matrix, which employs a colour gradient to represent correlation measures. This analysis is crucial for understanding the relationships between features. The correlation matrix revealed a significant disparity between features' correlations, ranging from high to low to negative. This disparity among features indicates that the significance of individual features can change over time. The dynamic nature of feature relevance underscores the necessity for an IDS to adapt continuously to maintain its effectiveness. The HIKARI-2021 dataset, with its diverse and changing feature correlations, is therefore well-suited for studying and addressing feature drift. This characteristic ensures that the IDS can learn from a broad spectrum of unique attributes, enhancing its robustness against the evolving nature of network traffic and intrusions in real-world environments.

4.2.1.2 TON_IoT Telemetry Dataset 2020

The TON_IoT dataset, introduced by [80], represents a new data-driven dataset for the IoT and Industrial Internet of Things (IIoT). It is specifically designed to reinforce the effectiveness of IDS tailored for IoT applications. This dataset addresses the critical need for updated and representative IoT datasets, which are vital for training and evaluating IDS-enabled IoT systems. The significance of TON_IoT lies in its unique contribution to a domain that previously lacked benchmark IoT and IIoT datasets. Notably, TON_IoT incorporates a label feature distinguishing between normal and attack classes, and a type feature specifying sub-classes of attacks targeting IoT/IIoT applications, making it apt for multi-classification problems.

The development of the TON_IoT dataset included creating a scaled-down representative testbed for each IoT device, involving seven different IoT and IIoT sensors. These sensors played a crucial role in capturing telemetry data, which constitutes the backbone of the dataset. The design of the testbed incorporated network elements and IoT/IIoT systems across the Edge, Fog, and Cloud layers, effectively simulating a realistic IoT/IIoT network environment. This setup ensures that the dataset authentically represents the characteristics of IoT services and network configurations. It includes telemetry data, operating systems' data, and network data from the IoT/IIoT network within the testbed.

Covering a broad spectrum of cyber-attacks, the TON_IoT dataset includes nine types such as Scanning, DoS, DDoS, Ransomware, Backdoor, Data Injection, Cross-site Scripting (XSS), Password Cracking, and Man-in-The-Middle (MITM) attacks. These were executed against various IoT and IIoT sensors within the network. The dataset, containing 43 features with a total of 22,339,021 records, and its imbalanced dataset , as depicted in Figure 4.2.

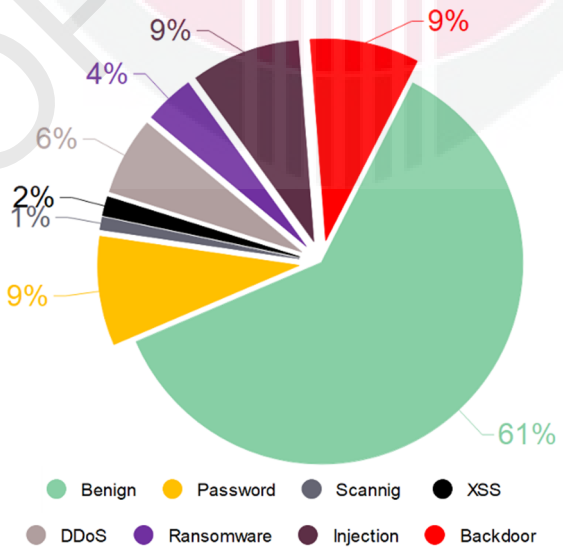


Figure 4.2: The data distribution in TON_IoT Dataset

In addition to the real-world features mentioned above, the TON_IoT Telemetry dataset is an excellent source for studying feature drift due to several key factors. Firstly, the dataset captures a wide range of telemetry data, operating system logs, and network traffic, simulating realistic IoT and IIoT environments. This diverse data collection encompasses various types of cyber-attacks and normal operations, providing a dynamic and evolving dataset that closely mirrors real-world conditions.

Moreover, the dataset includes a significant disparity between feature values, as indicated by the need for normalization to prevent features with larger values from dominating those with smaller values. This disparity suggests that feature correlations could vary significantly between feature over time, making the dataset well-suited for analyzing how the relevance of individual features shifts in response to changing attack patterns and normal behavior.

4.2.2 Mathematical Functions

The MOO algorithms evaluation involved the adoption of six mathematical benchmarking problems. These problems exhibit various shapes and levels of difficulty. Some well-known test problems, commonly utilized for assessing these algorithms, include Fonseca function(FON), Kursawe function(KUR), ZDT-1, ZDT-2, ZDT-3, and ZDT-6 as depicted in Table 4.1 [194].

1. **FON:** The FON benchmark is two objectives with $n - dimensional$ problem, usually defined in a domain of $x_i \in [-4,4], i = 1, \dots n$, It is known for its non-convex Pareto front. The objectives are typically conflicting, and the algorithm's aim is to find a set of solutions that lie on this front.
2. **KUR:** The KUR benchmark is a two objectives optimization problem with $n - dimensions$, generally $x_i \in [-5,5], and\ i = 1, \dots n$. The problem is

particularly useful for testing an algorithm's ability to identify solutions along non-convex and disconnected fronts, which are more challenging to approximate.

3. **ZDT-1:** ZDT-1 benchmark is a convex Pareto-optimal front, a well-known set of problems for evaluating MOO algorithms. ZDT1 is defined in $n-dimensions$, and $x_i \in [0,1], i = 1,2,\dots,n$.

4. **ZDT-2:** ZDT-2 benchmark is also an $n-dimensional$ problem with the same domain as ZDT1. However, it presents a more challenging scenario with a non-convex Pareto-optimal front. Where $x_i \in [0,1]$ and $i = 1,2,\dots,n$, where n equal to 30.

5. **ZDT-3:** ZDT-3 benchmark introduces a convex and discontinuous Pareto front. The discontinuity in the Pareto front makes it particularly challenging for algorithms that are not well-suited for handling such complexities. Where $x_i \in [0,1]$ and $i = 1,2,\dots,n$, where n equal to 30.

6. **ZDT-6:** ZDT-6 benchmark presents another $n-dimensional$ problem with a non-uniform and non-convex Pareto-optimal front. The objectives are conflicting and the Pareto front is narrow, making it difficult for algorithms to maintain diversity while approximating the front accurately. Where $x_i \in [0,1]$ and $i = 1,2,\dots,n$, where n equal to 10.

Table 4.1: The benchmarking mathematical functions

Problem	n	Variable bounds	Objective functions	Optimal solutions	Comments
FON	3	$[-4,4]$	$f_1(x) = 1 - \exp\left(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2\right)$ $f_2(x) = 1 - \exp\left(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2\right)$	$x_1 = x_2 = x_3$	Non-Convex
KUR	3	$[-5,5]$	$f_1(x) = \sum_{i=1}^{n-1}(-10exp\,(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$ $f_2(x) = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5\sin(x_i^3))$	$x_i \in [-5,5]$ $i = 1,2,3\dots,n$	Non-convex disconnected
ZDT1	30	[0,1]	$f_1 = (x) = x_1$ $f_2(x) = g(x)\left[1 - \sqrt{\frac{x_1}{g(x)}}\right]$ Where $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0,1]$ $x_i = 0$ $i = 1,2,3,\dots,n$	Convex
ZDT2	30	[0,1]	$f_1 = (x) = x_1$ $f_2(x) = g(x)\left[1 - (x_1/g(x))^2\right]$ Where $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0,1]$ $x_i = 0$ $i = 1,2,3,,n$	Non- convex

Table 4.1: Continued

ZDT3	30	[0,1]	$f_1 = (x) = x_1$ $f_2(x) = g(x)\left[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}\sin(10\pi x_1)\right]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)\Big/(n-1)$	$x_1 \in [0,1]$ $x_i = 0$ $i = 1,2,3,\ldots,n$	Convex, disconnected
ZDT6	10	[0,1]	$f_1(x) = 1 - \exp(-4x_1)\, sin^6(6\pi x_1)$ $f_2(x) = g(x)\left[1 - (\frac{f_{1(x)}}{g(x)})^2\right]$ $g(x) = 1 + 9\left[(\sum_{i=2}^{n} x_i)/n - 1\right]^{0.25}$	$x_1 \in [0,1]$ $x_i = 0$ $i = 1,2,3,\ldots,n$	non-uniform, non-convex

These mathematical benchmark functions are essential for evaluating the performance of MOO algorithms due to their varying levels of complexity and difficulty. The challenges posed by these functions such as non-convexity, disconnected Pareto fronts, and the need to maintain solution diversity are analogous to the real-world challenges faced when optimizing IDS. In the context of IDS, the key objectives are maximizing classification accuracy while minimizing the number of features used. The non-convex and complex nature of these benchmarks simulates the trade-offs between accuracy as a first objective and feature reduction as a second objective in IDS. For instance, functions like FON and KUR test the algorithm's ability to handle non-convex and disconnected fronts, similar to managing the dynamic and evolving nature of network traffic in IDS. Likewise, the ZDT series, with its varying front shapes and difficulties, helps assess the algorithm's capability to balance and optimize multiple conflicting objectives, which is critical when applying MOO to real datasets like TON-IoT or HIKARI. The successful in navigating these benchmark problems can provide valuable insights into the algorithm's effectiveness in handling the dual objectives of

accuracy and feature reduction within the IDS framework. Such as in the work [195] the author introduce and MOO algorithm to reduce the number of feature while maintain the error rate where the final pareto could forming a shapes close to KUR function or even close to ZDT-1.

These mathematical benchmark problems are essential to apply in the evaluation process to test the MOO algorithms' ability to balance conflicting objectives, such as maximizing accuracy and minimizing feature count, which are critical in IDS. By using these benchmarks, we can assess how well the algorithms perform in optimizing these objectives under various levels of complexity, ensuring that they are robust and effective when applied to real-world IDS datasets like TON-IoT and HIKARI.

4.2.3 Classification Evaluation Metrics

This section introduces the evaluation metrics from a classification standpoint, highlighting the confusion matrix as an essential tool for analysis, as depicted in Figure 4.3. This matrix effectively defines the classification results into two dimensions: the predicted class and actual class. The confusion matrix encapsulates the essence of classification performance by categorizing each instance into one of four groups: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). TP denote instances where the model correctly classified actual attacks as attacks, while TN represent cases where benign instances are accurately classified as benign. Conversely, FN occur when actual attacks are mistakenly classified as benign, and FP are instances where benign occurrences are incorrectly classified as attacks.

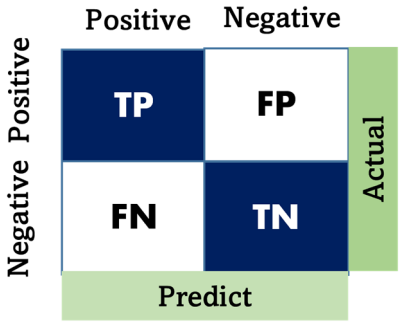


Figure 4.3: Confusion matrix

The classification performance was evaluated using four metrics: Accuracy, precision, recall and F1-score, as illustrated in Equations (4.1), (4.2), (4.3) and (4.5) [102].

1. Accuracy (Acc): Accuracy is defined as the ratio of true classified instances to the total number of classified instances. This can be measured by the Equation (4.1).

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.1}$$

2. Precision (Pr): Precision used to measure the positive instances that are correctly classified to the total classified instances in a positive class (TP, FP). This can be calculated using the Equation (4.2).

$$\text{Pr} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

3. Recall (Rc): recall also called the sensitivity measurement, can be calculated by dividing the number of true positive instances by the sum of true positive instances and false negative instances. This can be calculated using the Equation (4.3).

$$\text{Rc} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.3}$$

4. F1 Score: F1 score or the harmonic mean of precision and Recall. This can be calculated using the Equation (3.11).

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.5}$$

In the context of IDS, evaluating classification performance is crucial to determining how effectively the system differentiates between normal and malicious activities. The key metrics used for this evaluation are Accuracy, Precision, Recall, and F1-Score, all derived from the confusion matrix as illustrated in Figure 4.3. Accuracy measures the overall correctness of the IDS by calculating the proportion of true results (TP and TN) relative to the total number of cases examined. While high accuracy suggests good overall performance, it can be misleading in cases of imbalanced datasets where normal events vastly outnumber attacks. In such scenarios, accuracy might remain high despite poor detection of attacks, emphasizing the need to consider additional metrics [196].

Precision is vital in an IDS context as it assesses the correctness of positive predictions, when the IDS flags an event as an attack, how often is it accurate. High precision is crucial for minimizing false alarms or FP, which can devastate administrators and lead to alert exhaustion, especially in environments where the cost of false positives is significant. Recall, or sensitivity, evaluates the IDS's ability to detect actual attacks by measuring the proportion of true positives among all actual attacks TP and FN. High recall is essential for identifying most attacks but must be balanced with precision to avoid an increase in false positives. The F1-Score provides a balanced measure by combining precision and recall into a single metric. It is particularly useful in scenarios where both false positives and false negatives are costly, offering a comprehensive view of the IDS's performance. The F1-Score is calculated as the harmonic mean of precision and recall, reflecting the trade-off between these two metrics [196].

A confusion matrix for multi-class classification extends the concept of the traditional binary confusion matrix to scenarios where there are more than two classes. This type of confusion matrix is particularly useful in problems where an instance can be classified into one of several categories, as shown in Figure 4.4. In dimensions the matrix is a square matrix, with the number of rows and columns equal to the number of classes. For instance, in a scenario with three classes (A, B, C), the matrix will be 3x3. In rows and columns, each row of the matrix represents the instances in an actual class (True Labels), while each column represents the instances in a predicted class (Predicted Labels). Here, Class A shown in the figure, was correctly classified 50 times, misclassified as Class B 5 times, and as Class C 1 times, and so on for the other classes.

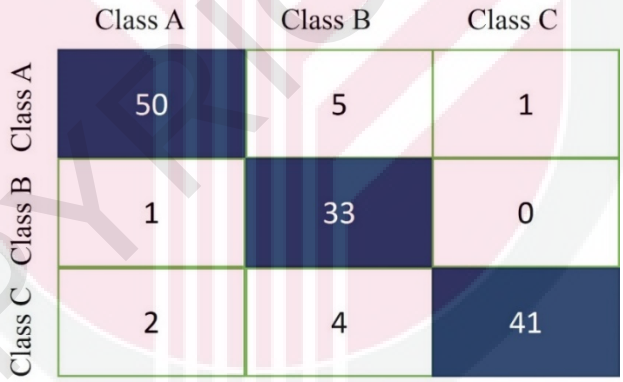


Figure 4.4: Confusion Matrix for Multi Classification Task

4.2.4 Memory Reduction and Feature Counts Measures

Memory reduction refers to the ability of an IDS FS to minimize the size of memory required for processing the data, which is particularly important given the vast volume of data that modern IDS must handle. Efficient memory utilization not only speeds up

the detection process but also reduces the computational overhead, making the IDS more effective in real-time threat detection scenarios. This metric becomes increasingly significant in the context of high-speed networks and high dimensionality data, where memory resources can be a limiting factor. Implementing algorithms and techniques that optimize memory usage without compromising detection accuracy is a key area of research in IDS development.

Calculating memory reduction in IDS typically involves assessing the amount of memory used before and after the implementation of memory optimization techniques. Memory reduction can be calculated using the Equation (4.6):

$$\text{Memory Reduction} = \frac{Memory\ usage\ after\ feature\ selection - IDS}{Memory\ usage\ without\ feature\ selection - IDS} \times 100\% \quad (4.6)$$

This calculation provides a measure of the effectiveness of memory reduction strategies in IDS. It provides valuable insights into the efficiency of FS in optimizing memory, highlighting its significance in the operational effectiveness of IDS.

Feature counts metrics refer to the proportion of features utilized by the IDS- FS that significantly contribute to its detection capabilities. This metric is particularly relevant in scenarios where dimensionality reduction or feature selection techniques are employed to enhance the IDS's performance and efficiency. Feature count can be calculated using the Equation (4.7):

$$\text{Feature Count} = \frac{Number\ of\ Selected\ Features}{Total\ Number\ of\ features} \times 100\% \quad (4.7)$$

The feature account percentage helps in optimizing the IDS by focusing on the most relevant features, thus reducing computational complexity and potentially improving detection speed and accuracy. However, care must be taken to ensure that the removal of features does not omit critical information necessary for accurately identifying threats.

4.2.5 Optimization Evaluation Measures

This subsection, describes the metrics utilized to evaluate the efficacy of the developed MOO method. The evaluation framework is centred on five critical metrics: Set Coverage (SC), Hyper-Volume (HV), the Number of Non-Dominated Solutions (NDS), generational distance (GD), and the Delta Measure (Δ) [197]. Each metric provides a distinct lens through which the effectiveness and robustness of the optimization process can be discerned.

4.2.5.1 Set Coverage Measure

The SC Measure, also known as the C-metric, evaluates the extent to which solutions in one Pareto set are dominated by solutions in another set. It facilitates the comparison of two Pareto sets, $Ps1$ and $Ps2$, using the following Equation (4.8):

$$C(Ps1;\ Ps2) = \frac{|\{\, y \in Ps2 | \exists x \in Ps1 : x > y\}|}{|Ps2|} \tag{4.8}$$

Here, C denotes the proportion of non-dominated solutions in $Ps2$ that are dominated by those in $Ps1$, relative to the total number of solutions in $Ps2$. In the context of

evaluating a Pareto set Ps, the aim is to minimize the C-metric. Ideally, a value approaching 0 suggests that the MOO has yielded a favourable performance.

4.2.5.2 Hyper-Volume Measure

The HV metric is widely utilized in evolutionary MOO to evaluate the performance of search algorithms. It calculates the volume encompassed by the dominated portion of the objective space, taking a predefined worst-case solution as the reference point. The calculated region represents the collective hypercubes formed by the diagonals spanning from the reference point to each solution x within the Pareto set PS. Elevated HV values are indicative of more favourable solutions, suggesting a greater spread and reach of the Pareto front. The HV is mathematically articulated in Equation (4.9).

$$HV = volume\left(\bigcup_{x \in Ps} HyperCube(x)\right) \tag{4.9}$$

4.2.5.3 Delta Measure

The Delta measure, also known as the diversity metric, assesses the distribution and spread of non-dominated solutions within the context of multi-objective optimization. The fundamental aim of the Delta measure is to ascertain that the solutions procured are not merely Pareto-optimal (non-dominated) but are also equitably distributed along the Pareto front. This equitable distribution is vital as it presents a broad spectrum of viable alternatives to decision-makers. The computation of the Delta measure involves analysing the set of non-dominated solutions and yielding a value as defined by Equation (4.10).

$$\Delta = \frac{df + d_l + \sum_{i=1}^{N-1}\left|d_i - \bar{d}\right|}{df + d_l + (N-1)\bar{d}} \tag{4.10}$$

Here, N represents the number of solutions, denoted as d_i. The terms df and d_l refer to the Euclidean distances between the extreme solutions and the boundary solutions of the Pareto front, respectively. Meanwhile, $\bar{d}$ is the average of all consecutive distances d_i, calculated for $i = 1, 2, \ldots, N - 1$. A smaller value of Δ indicates a better and more uniform distribution of the solutions along the Pareto front. A larger value of Δ suggests that the solutions are clustered in certain regions of the Pareto front and not uniformly distributed.

4.2.5.4 Generational Distance Measure

This metric quantifies the distance between the Pareto front obtained through the proposed method and the true or reference Pareto front. A lower GD value is desirable, as it signifies closer proximity to the true Pareto front, and by extension, better performance of the optimization method as referenced in. The calculation of GD follows the formula outlined in Equation (4.11).

$$GD = \frac{1}{N}\sqrt{\sum_{j=1}^{N} d_j^2} \tag{4.11}$$

Where N is the number of solutions founded by the proposed algorithm, and d_i is the Euclidean distance each solution and its nearest point in true Pareto front.

4.2.5.5 Number of non-Dominated Solutions Measure

The NDS serves as a measure to express the effectiveness of the optimization algorithm. It is calculated by determining the cardinality of the Pareto set P_s, which represents the total number of non-dominated solutions obtained. This metric is crucial for evaluating the algorithm's ability to generate a diverse set of optimal solutions. The measure can be calculated using the Equation (4.12).

$$NDS\ (N) = |P_s| \tag{4.12}$$

A higher NDS value suggests that the algorithm has found a larger set of optimal trade-offs between conflicting objectives, which is often the goal in MOO problems.

4.3 Experimental Design

This section outlines the experimental design for both the MEPSOLA and DFA-GPE frameworks. It explains on the experimental parameters and configurations of MEPSOLA, introducing the set values for various mathematical functions and outlining the parameters used for benchmark comparisons methods. The section then transitions to a detailed exploration of the DFA-GPE framework, highlighting the ensemble classifier and feature selection parameters. Additionally, it provides an in-depth description of the parameters used in other benchmarking methods.

4.3.1 Experimental Design for MEPSOLA

In the experimental design of MEPSOLA for precise evaluation, a set of six mathematical functions with different shapes, namely FON, KUR, ZDT1, ZDT2,

ZDT3 and ZDT6, were adopted to assess the algorithms. These mathematical functions, each with a unique shape and characteristics, were selected to strictly test and assess the performance of the algorithms. The configurations for MEPSOLA, the integrated Tabu search, and the varied acceleration parameter (C) for the mathematical functions are all specified in Table 4.2.

The proposed method was compared with well-established benchmarking MOO algorithms, namely NSGA-II [130], NSGA-III [131], and MOEA [133], with the parameters for these algorithms illustrated in Table 4.4. These benchmark algorithms are recognized for their robust performance across a wide range of MOO problems and have been extensively validated in the literature. To ensure parity in the comparative analysis, the parameter settings for benchmarking were directly aligned with those recommended in their original publications. This approach enables a fair comparison by maintaining consistency with the conditions of their established performance benchmarks.

Table 4.2: The experimental design parameters for MEPSOLA

MEPSOLA parameters	**Value**
Population size	100
Inertia weight (W)	1
Acceleration parameter (C)	0.2
Mutation ratio (pm)	0.2
Number of iterations	100
number of decision variables(n)	3
Tabu search parameters	**Value**
Max neighbors	10
Max iterations	80
Tabu list size	10
Deviation range	0.1
Mathematical function	**The acceleration parameter value**
FON, ZDT1	0.2
ZDT2, ZDT6, ZDT3	0.4
KUR	0.7

The selection of specific parameters for MEPSOLA is crucial for optimizing its performance in solving MOO problems. Each parameter has been carefully chosen based on established optimization practices and empirical evidence.

Population Size (100): A population size of 100 is widely used in MOPSO to balance exploration and exploitation. This choice ensures fair comparison with benchmarks by aligning with referenced population sizes all set at 100 [130] [131] [133].

Mutation Ratio with Damping Mechanism: An initial mutation ratio of 0.2 enhances exploration. A damping mechanism gradually reduces this rate, shifting focus to exploitation as the algorithm converges. This gradual reduction refines solutions in later optimization stages[198].

Inertia Weight and Acceleration Parameter: These were chosen based on a parameter sensitivity analysis conducted on the FON function, as shown in Table 4.3 and Figure 4.5. The analysis, across five scenarios, assessed the impact on Pareto front shape concerning diversity, convergence, and solution distribution. Scenario 1 produced the best Pareto front with optimal solution distribution, even with the smallest population size compared to Scenario 4.

Table 4.3: Parameters Sensitivity for MEPSOLA in FON Function

Scenario No	W	C	Population	Iteration
Scenario 1	1	0.2	100	100
Scenario 2	1	0.9	50	100
Scenario 3	1	0.9	100	100
Scenario 4	0.7	0.9	200	100
Scenario 5	0.7	0.5	50	50

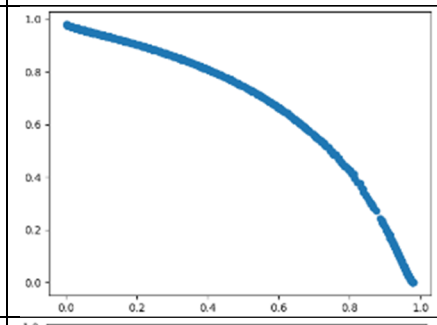
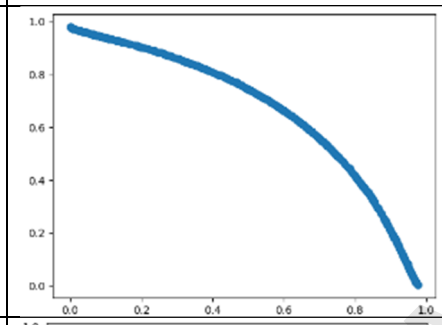
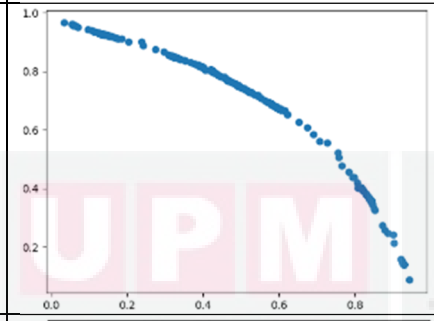
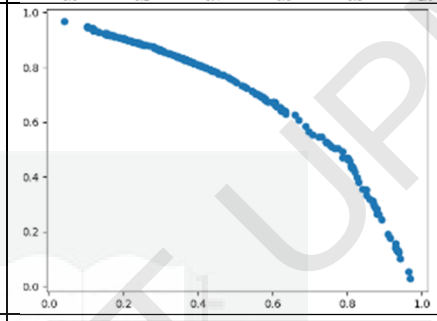
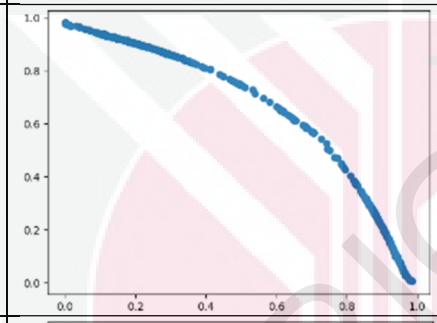
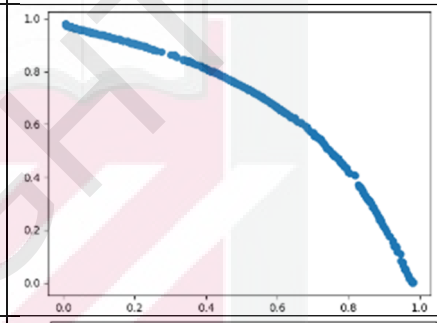
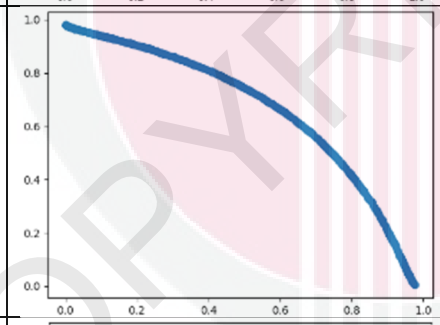
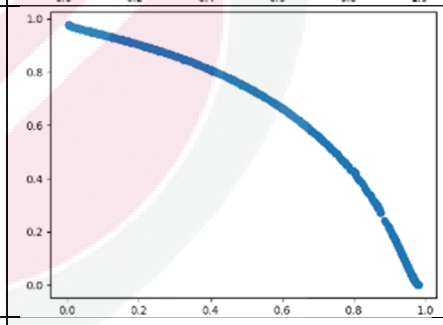
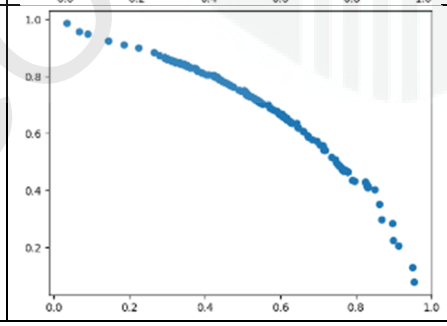
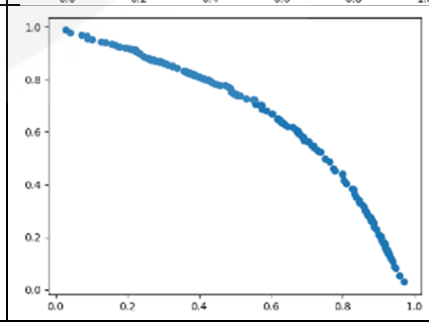
No	Pareto front 1	Pareto front 2
1		
2		
3		
4		
5		

Figure 4.5: Parameters sensitivities for FON function

Table 4.4: The experimental parameters for benchmarks

NSGA-II Parameters [130]	
Parameter	Value
Selection	Binary Tournament
Crossover	SBX (eta=15, prob=0.9)
Mutation	PM (eta=20)
Survival	Rank and Crowding
Number of Generations	200
Tournament Type	Compare by domination and crowding distance
NSGA-III Parameters [131]	
Parameter	Value
Selection	Tournament (custom comparison function)
Crossover	SBX (eta=30, prob=1.0)
Mutation	PM (eta=20)
Survival	Reference
Number of Generations	200
MOEA Parameters [133]	
Parameter	Value
Number of Neighbors	20
Selection	Neighborhood (prob_neighbor_mating=0.9)
Crossover	SBX (prob=1.0, eta=20)
Mutation	PM (eta=20, prob_var=None)
Number of Generations	200
Decomposition Method	Tchebicheff (≤2 objectives), PBI (otherwise)

The experimental simulations were executed using the Python programming language in the PyCharm Community Edition and Google Colab platforms. The hardware setup comprised a 12th generation Intel Core i5 processor with a clock speed of 4.40 GHz, 10 cores, and 16 GB of DDR4 RAM, ensuring that the computational environment was sufficiently capable of handling the intensive calculations required by these optimization algorithms.

4.3.2 Experimental Design for DFA-GPE

In the experimental framework of our DFA-GPE methodology, the feature selection process was precisely initiated by allocating the data into segments, with the original data divided into 0.4 utilized for training and the remaining 0.6 for testing. This approach was particularly crucial in the streaming data context, where the data flow

was segmented into chunks (400 chunks), each tailored to the specificities of the datasets in use. The parameters of the PSO algorithm, vital to the methodology, are detailed in Table 4.5.

To clarify the splitting ratio of the training and testing, in incremental learning based on chunks is particularly valuable because it eliminates the need to build the model from scratch each time new data is received. Instead of retraining on the entire dataset, the model updates incrementally as new chunks of data arrive. This allows the system to adapt to changes in the data stream, such as concept drift, without the high computational costs and time associated with full retraining. This approach not only enhances efficiency but also ensures that the model remains up-to-date and accurate in real-time [25]. Additionally, using a smaller portion for training is crucial in resource-constrained environments like real-time IDS, where processing power and memory are limited. This approach reduces computational load and avoids the need to rebuild the model from scratch. It also helps prevent overfitting, ensuring the model generalizes well to new data rather than just memorizing the training data, which is essential for maintaining accuracy and adaptability over time [199]. Therefore, where learning starts small and gradually increases in complexity is rooted in the concept of incremental learning. By beginning with simpler tasks or smaller portions of data, the learning process allows the model to build a solid foundation of understanding. This method minimizes the risk of the model becoming overwhelmed by too much data at once, which can lead to overfitting a scenario where the model performs well on training data but fails to generalize to new, unseen data [34].

For the parameters alpha (α), beta (β), gamma (γ), and granularity were selected based on their critical role in optimizing the performance of the MOPSO algorithm. Alpha (α) influences the expansion of the search space, allowing the algorithm to explore a broader range of potential solutions. Beta (β) plays a crucial role in the selection of the leader during the optimization process, affecting the convergence rate towards the optimal solution. Gamma (γ) is essential for the selection pressure, which helps in maintaining diversity within the solution set by managing the deletion of particles from the repository. Granularity, on the other hand, determines the density of the search space, directly impacting the precision and diversity of the solutions obtained. These parameters are chosen for their ability to enhance the balance between exploration and exploitation in MOPSO [200].

Regarding mutation damping and inertia damping in MOPSO, these techniques are employed to balance exploration and exploitation throughout the optimization process. Mutation damping gradually reduces the mutation rate, allowing for diverse exploration in the early stages and more focused refinement as the algorithm progresses [162]. Similarly, inertia damping decreases the inertia weight over time, enabling particles to explore the search space widely initially and then concentrate on fine-tuning solutions near optimal regions [177]. For the number of divisions choosing as 5 strikes a balance between search efficiency and computational cost. Dividing the feature space into five subspaces allows for a focused search without creating overly large or small divisions. This setup ensures the cooperative search remains effective by providing sufficient diversity in the subspaces while keeping computational costs manageable [55].

To ensure a comprehensive and rigorous evaluation of the proposed DFA-GPE method, it was crucial to select benchmarking algorithms that are both well-established and effective in handling feature selection, high-dimensional data, and multi-objective optimization. The chosen methods, RF, RFE, LR, and NSGA-II, each bring distinct advantages that make them ideal candidates for comparison.

Table 4.5: Parameterization and values for dynamic feature selection based on MOPSO

Parameter	Value
Population	20
Mutation probability	0.2
Iterations for HIKARI	20
Iterations for TON_IoT	30
Mutation damping	0.9
Number of divisions	5
Beta	5
Chunk length	500
Inertia weight	1
Inertia damping	0.99
Attraction factor C	2
Alpha	7
Grid granularity	0.1
Gamma	2
Number of grids	7
KNN value for HIKARI	5
The objective function for TON_IoT	Ensemble classifier

Random Forest is a famous and widely used algorithm for feature selection, particularly in high-dimensional datasets. It is effective because it operates as an ensemble learning method, combining the outputs of multiple decision trees. This approach enhances stability and reduces the risk of overfitting, which is a common challenge in complex datasets. Moreover, Random Forest not only ranks features based on their importance but also provides a robust mechanism for selecting the most relevant features by accounting for the interactions among them. This makes it an ideal choice for feature selection in scenarios where the dataset has a large number of features, ensuring that only the most informative ones are retained [201].

The greedy algorithm RFE for feature selection by highlighting its ability to systematically remove the least important features and improve the model's performance by focusing only on the most relevant ones. RFE works by recursively fitting a model and eliminating the weakest features, which reduces the dimensionality of the dataset and helps in enhancing the classification accuracy. This method is particularly effective when dealing with high-dimensional data, as it ensures that only the most significant features are retained, thereby improving the overall efficiency and effectiveness of the intrusion detection system [202].

The use of feature selection based on LR by emphasizing its effectiveness in reducing computational cost while maintaining high accuracy in classification tasks. Logistic regression is a linear and cost-effective method that assigns weights to features, allowing for the elimination of irrelevant or less important features. This process ensures that only the most significant features are retained, which simplifies the model and enhances its performance without compromising accuracy, making it a suitable choice for feature [203].

Selecting NSGA-II for comparison with MOPSO lies in the significant similarities between the two algorithms. Both NSGA-II and MOPSO are designed for multi-objective optimization, making them particularly suitable for handling multiple conflicting objectives. As population-based methods, NSGA-II uses a population of potential solutions, while MOPSO utilizes a swarm of particles. Both algorithms explore the search space effectively and converge toward a diverse set of optimal solutions. They also employ Pareto-based selection mechanisms to guide their search processes and incorporate diversity maintenance techniques to avoid premature convergence, ensuring a broad exploration of the solution space [50].

The selection of these specific methods was also influenced by their relevance and established effectiveness in the context of feature selection and multi-objective optimization, particularly within the types of datasets and problems addressed in this thesis. While other advanced machine learning models, such as Artificial Neural Networks (ANN) [204] and Convolutional Neural Networks (CNN) [205], are powerful tools for tasks like deep learning and complex pattern recognition, they were not included in this comparison for several reasons. ANN and CNN typically require large datasets and substantial computational resources, making them less suitable for scenarios where the availability of incoming instances is unpredictable, as is often the case in data streams [182]. Moreover, CNNs are particularly effective in image processing tasks, but their strength in feature selection for non-image data is less pronounced. Additionally, integrating feature selection mechanisms into ANN and CNN models can be more complex and less interpretable compared to the more traditional models selected for this study. These considerations led to the decision to focus on methods that are more directly applicable and interpretable in the context of the datasets and challenges explored in this thesis.

By focusing on RF, RFE, LR for feature selection, and NSGA-II for MOO, the comparison ensures a fair and relevant evaluation of DFA-GPE. This selection provides a comprehensive assessment of the proposed method's effectiveness in various contexts, covering a broad spectrum of feature selection strategies and optimization techniques, while remaining directly applicable to the specific challenges of the datasets under study.

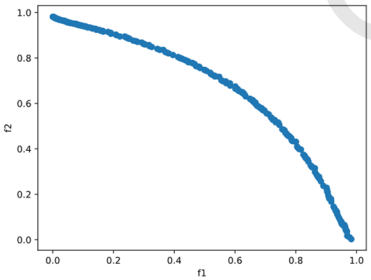
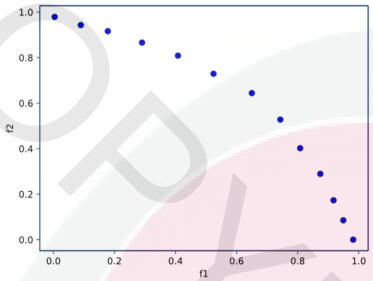
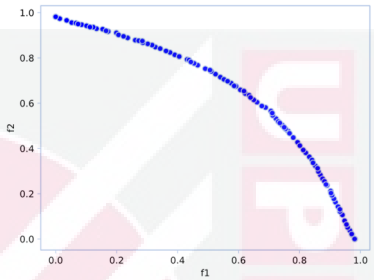
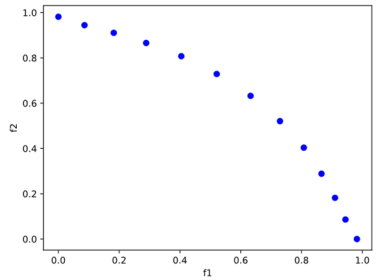
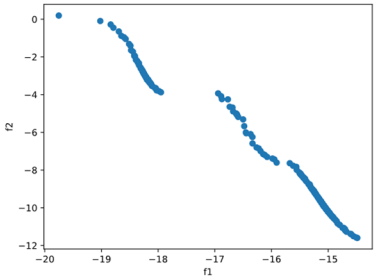
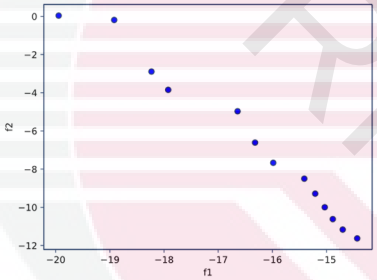
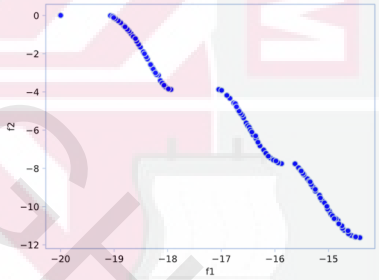
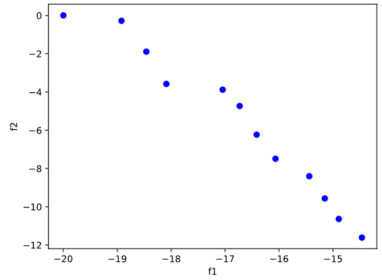
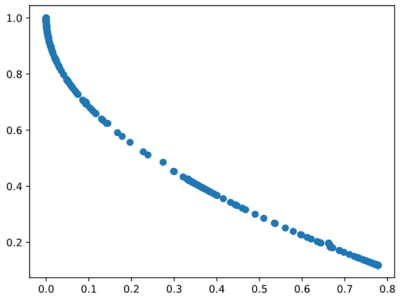
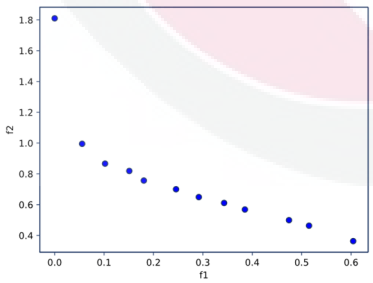
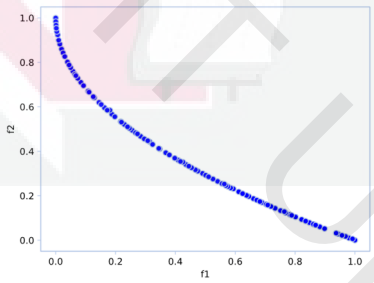
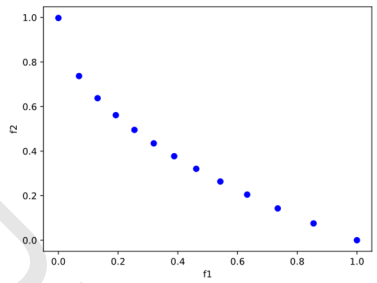
In the classification phase, an ensemble classifier model was utilized. This model integrated ten diverse classifiers with variable parameters, including 3 classifiers of $K\ Neighbors$ with K value of 3, 5, and 7, and 3 classifiers of $Logistic\ Regression$ with iterations of 300, 400, and 500, and 3 of $Decision\ Tree$ with depth of 5, 7, and 8, and a single default parameters $Gaussian\ NB$ classifier, thus creating a robust ensemble classifier pool. Additionally, a base classifier replacement mechanism was adopted, employing an oldness replacement strategy at a 1/3 ratio. For drift detection, the DDM method was employed, reinforcing the reliability of the proposed approach.

To ensure the stability of the results, the experiments were conducted five times. The efficacy of DFA-GPE methodology was validated by benchmarking the results against four leading state-of-the-art methods: RF [206], LR [122], RFE equipped with a Support Vector Classifier (SVC) configured with a linear kernel and a maximum iteration of 500 [123], and NSGA-II [50]. This comparative analysis was essential in establishing DFA-GPE's performance in relation to established benchmarks in the field. The experiments were carried out on two systems. First, 12th generation Intel Core i5 processor with a clock speed of 4.40 GHz, 10 cores, and 16 GB of DDR4 RAM. Second, system running Ubuntu 20.04, equipped with Intel(R) Xeon(R) processor @ 2.30GHz and a memory capacity of 16 GB. The Python programming language was chosen as the tool for these experiments, conducted on both Google Colab and PyCharm Community Edition platforms.

4.4 MEPSOLA Results

In the domain of complex problems optimization, the efficiency of an optimization algorithm is pivotal. In this thesis introduces an improved version of MEPSOLA

approach, specifically designed for solve a complex optimization problem. The experimental analysis presented in this thesis aims to shed light on the algorithm's adeptness in solve the complex of MOO. The results shown in Figure 4.6, showcase the Pareto front obtained from the first run of the ten experimental runs. This visual representation serves as a demonstration to the algorithm's ability compare to the others benchmarking algorithms to generate a diverse and well-distributed set of solutions that closely approximate the true Pareto front. Furthermore, the comprehensive evaluation of MEPSOLA performance for all 10 obtained runs, and the comparison is grounded on several performance metrics that are critical for MOO problems.

	MEPSOLA	MOEA [133]	NSGA-2 [130]	NSGA-3 [131]
FON				
KUR				
ZDT1				

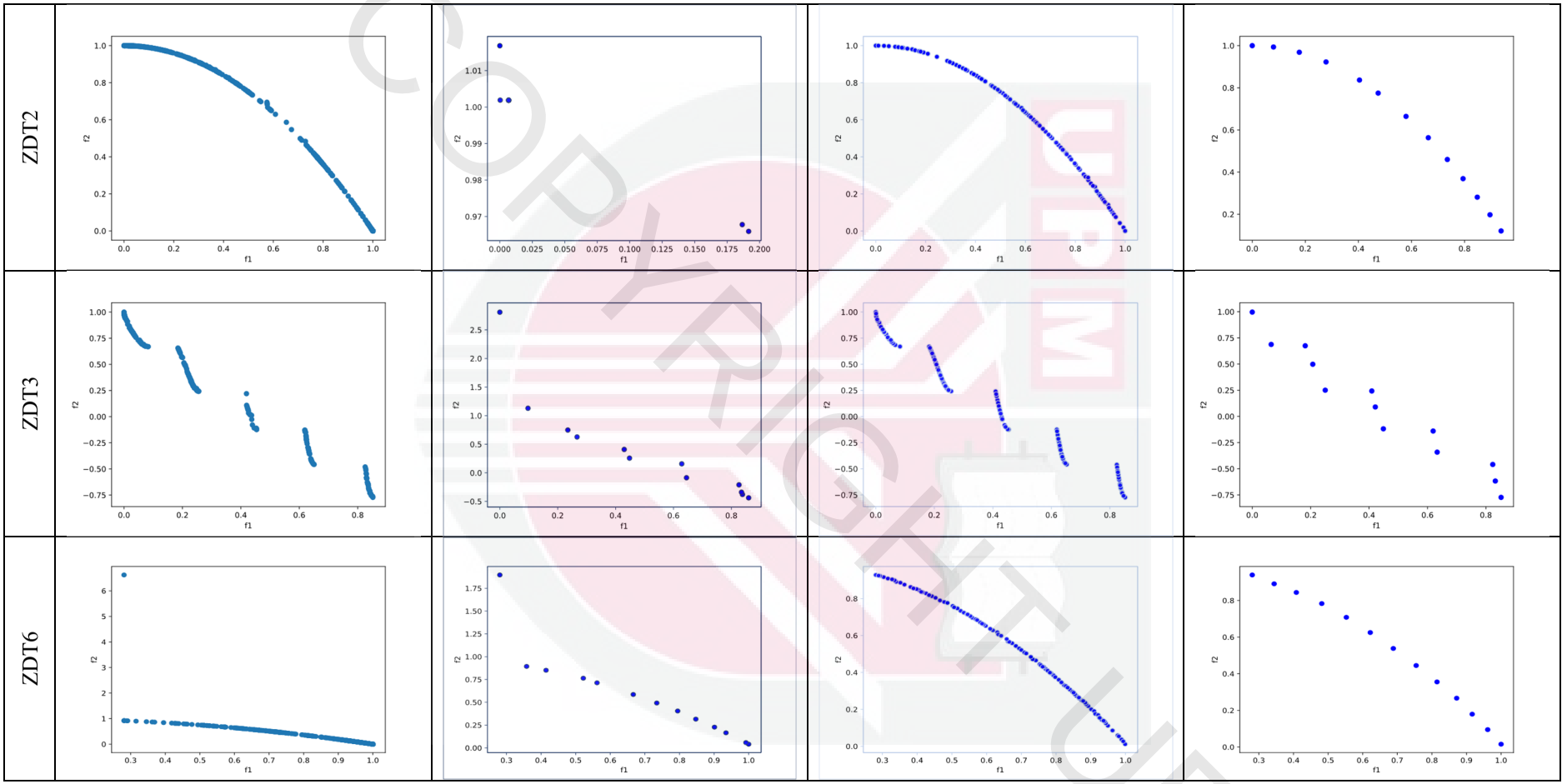


Figure 4.6: First Run Pareto Front Results Obtained from the Proposed Method Compare to MOEA, NSGA-2, And NSGA-3 Based on Mathematical Functions FON, KUR, ZDT1, ZDT2, ZDT3, And ZDT6

4.4.1 Fonseca Function (FON) Results

This section of the results represents a comprehensive comparison of the proposed method compare with three benchmarking MOO algorithms, leveraging the FON benchmark for evaluation purposes. The comparison is anchored on metrics such as Set Coverage (SC), Hypervolume (HV), Delta measure, Number of Non-Dominated Solutions (NDS), and Generational Distance (GD), offering a detailed assessment of each algorithm's performance.

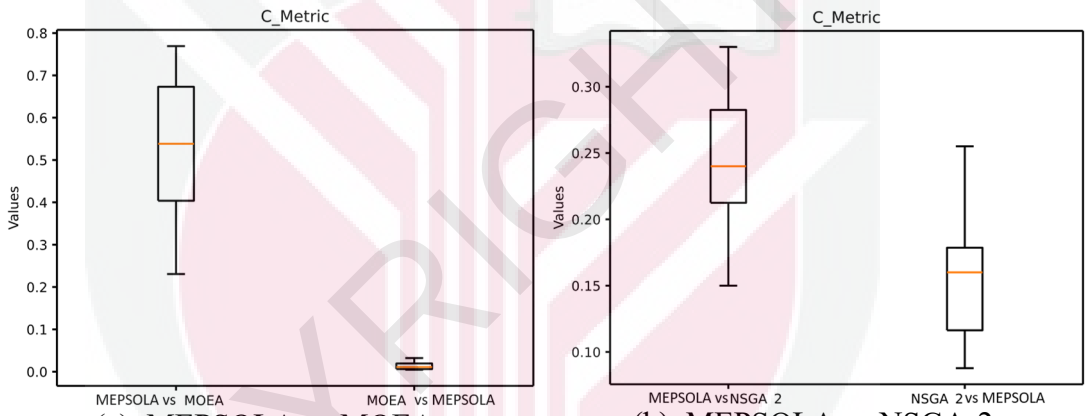
4.4.1.1 Set Coverage Measure

The SC measure results presented in Figure 4.7 for the FON function are used to determine the extent to which one set of solutions dominates another. A higher SC value indicates a greater dominance of one algorithm's solutions over another's. First, MEPSOLA against MOEA in the Figure 4.7 (a), the result shows that MEPSOLA's solutions have a considerable degree of dominance over MOEA's, with a minimum SC value of 0.23 and a maximum of 0.76. The median SC of 0.53 suggests that more than half of MEPSOLA's solutions dominate those of MOEA in most cases. The interquartile range for Q1 is 0.40 and Q3 is 0.67, further indicates a consistent dominance across runs. In contrast for MOEA shows very limited dominance over MEPSOLA's solutions, with a maximum SC value of only 0.03 and a median of 0.01, which is quite low, indicating that MOEA's solutions are rarely dominant over those from MEPSOLA.

Similarly, the result of MEPSOLA against NSGA-2 in the Figure 4.7 (b), indicates a notable level of dominance by MEPSOLA, with a minimum SC value of 0.15 and a maximum of 0.33. The median SC of 0.24 shows that MEPSOLA's solutions are

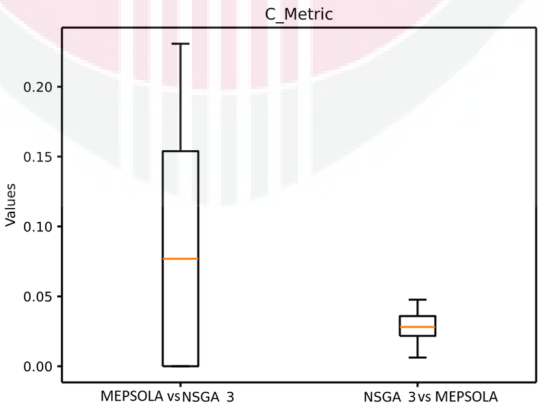
dominant over NSGA-2's solutions in a significant number of cases. However, NSGA-2 shows some degree of dominance over MEPSOLA 's solutions, with a maximum SC value of 0.25, but a lower median of 0.16.

However, the result for MEPSOLA against NSGA-3 in the Figure 4.7 (c), MEPSOLA 's solutions display a median SC value of 0.07, which is lower than against MOEA and NSGA-2, indicating less dominance. While NSGA-3's solutions show a small degree of dominance over MEPSOLA's with a median SC of 0.02, which is higher than MOEA's dominance over MEPSOLA but lower than NSGA-2's.



(a). MEPSOLA vs MOEA

(b). MEPSOLA vs NSGA-2



(c). MEPSOLA vs NSGA-3

Figure 4.7: Set Coverage Measure Results for FON Function

4.4.1.2 Hyper-Volume Measure

The HV measure results, depicted in Figure 4.8 for the FON function, which is represent the volume covered by the Pareto front generated by the algorithm. Higher HV values indicate a front that extends closer to the true Pareto front and covers a more significant portion of the objective space. First, the MEPSOLA achieves a significant HV values, with a minimum value of 525.71 and a maximum of 929.18, illustrating a significant coverage of the objective space. The median HV of 661.98 suggests that MEPSOLA consistently produces Pareto fronts that are expansive and capture a considerable region of the optimal set. The interquartile range for Q1 of 579.4 and for Q3 of 701.09 confirms the stable performance of MEPSOLA across different runs.

In contrast MOEA, shows very low HV values, indicating a limited coverage of the objective space. The maximum HV value of 24.67 is considerably lower than MEPSOLA's minimum, and the median HV of 24.61, along with the narrow interquartile range for Q1 is 24.57 and for Q3 is 24.64, implies that MOEA's Pareto fronts are far from extending across the optimal set. In NSGA-2 presents consistent HV values with a small range between the minimum of 187.91 and the maximum of 188.50. The median HV of 188.21 and the quartile values for Q1of 188.13 and Q3 of 188.27 suggest a moderate coverage of the objective space that is consistent across runs but not as extensive as MEPSOLA's.

The HV measure for NSGA-3, similarly shows consistent but low values, ranging from 24.68 to 24.72. The median HV of 24.7 and the interquartile values indicate a

performance that is similar to MOEA in terms of HV, with a Pareto front that covers a minimal portion of the optimal set. For the FON function, MEPSOLA significantly outperforms MOEA, NSGA-2, and NSGA-3 in terms of HV, demonstrating a superior ability to capture a broad and diverse set of optimal solutions. The low HV values for the benchmarking algorithms suggest that while these algorithms may find optimal solutions, they do so within a much narrower region of the objective space, potentially missing out on the full spectrum of optimal trade-offs.

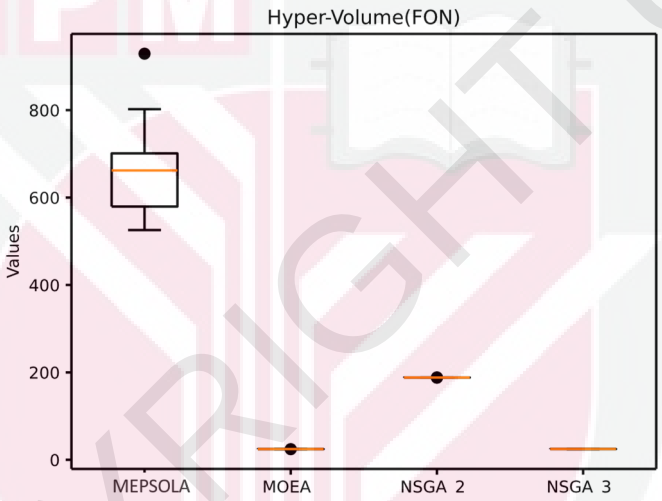


Figure 4.8: Hyper-Volume Measure Result for FON Function

4.4.1.3 Delta Measure

The result of the delta measure for the FON function, as depicted in Figure 4.9, the delta measure acts as a performance indicator to assess the quality of solutions an algorithm generates. It measures the minimum distance between the true Pareto front and the approximated Pareto front generated by the algorithm. In the case of MEPSOLA, the result shows a range of delta measure values from 0.781 to 0.928, with a median of 0.826. Although the maximum value is not ideal, the relatively low

minimum and median values suggest that MEPSOLA maintains a good distribution of solutions, with a tendency towards uniformity but some variability.

In MOEA, the delta values range from 0.673 to 0.685, with a median of 0.679. These values are quite compact, implying that while the solutions are somewhat uniformly spread, they may not cover the Pareto-optimal region as comprehensively as desired for NSGA-2, the delta values range from 0.667 to 0.691, with a median of 0.678. These values indicate a consistent spread, but similar to MOEA, it suggests a moderate uniformity without exceptional coverage. While the NSGA-3 presents delta value median of 0.675, which are close to those of MOEA and NSGA-2. This implies a consistent but potentially narrower spread, lacking in diversity compared to what is ideal [207].

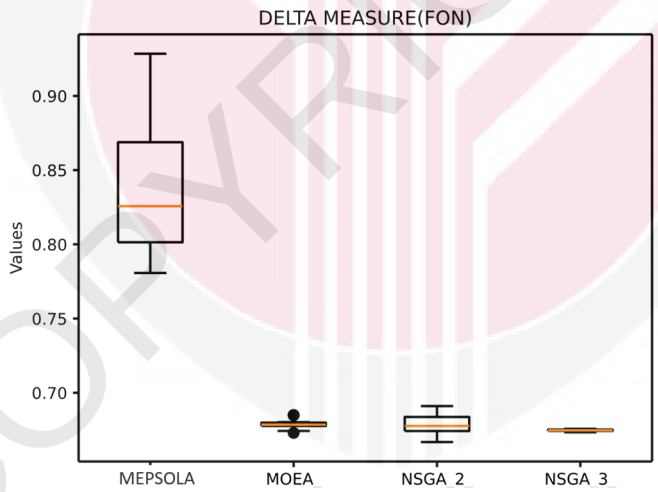


Figure 4.9: Delta Measure Result for FON Function

4.4.1.4 Number of Non-Dominated Solutions Measure

The NDS metric results in Figure 4.10 for the FON function provides insight into the effectiveness of an optimization algorithm at finding a diverse set of Pareto-optimal

solutions. The results of MEPSOLA demonstrate a strong ability to identify a large set of non-dominated solutions, with a minimum count of 277 and a maximum of 490. The median of 346 suggests that MEPSOLA consistently finds a substantial number of optimal solutions across its runs. The interquartile range for Q1 of 306.5 and Q3 of 368.25 highlights the algorithm's stability in capturing optimal solutions. In contrast, MOEA shows significantly lower NDS counts, with all values fixed at 13. This consistency suggests that MOEA may have a much narrower scope in its exploration of the solution space. NSGA-2 results present a constant number of 100 non-dominated solutions across all statistical measures. While this is higher than MOEA's count, it still limits the variety and number of solutions identified by MEPSOLA.

However, NSGA-3 results display similar results to MOEA, with a consistent count of 13 non-dominated solutions. This indicates a similar limitation as MOEA in terms of the diversity and breadth of the solution space exploration. In the FON function MEPSOLA outperforms MOEA, NSGA-2, and NSGA-3 with respect to the NDS measure demonstrating the ability to generate a large and diverse set of Pareto-optimal solutions.

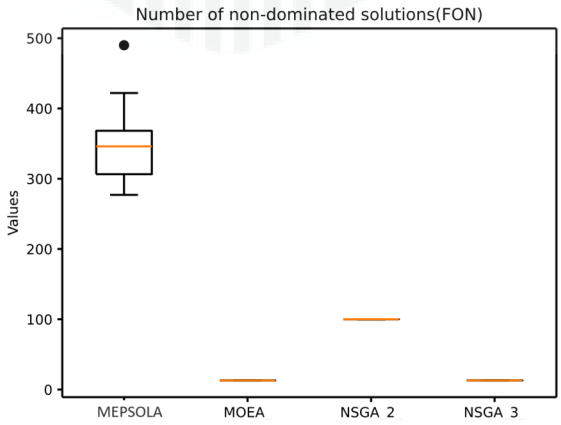


Figure 4.10: Number of Non-Dominated Solutions Measure Results for FON Function

4.4.1.5 Generational distance Measure

The GD measure for the FON function measures the proximity of the generated solutions to the true Pareto front, with lower values indicating that the solutions are closer to the front. In Figure 4.11, the results for MEPSOLA show GD values ranging from 0.0041 to 0.0054, with a median of 0.0043. These values suggest that MEPSOLA consistently produces solutions that are quite close to the true Pareto front. The interquartile range for Q1 of 0.0041 and Q3 of 0.0048 further highlights the consistency of this performance. The MOEA shows a minimum GD value of 0.0014, and the maximum value of 0.0044 and a median of 0.0025 indicate more variability in its performance. The interquartile range for Q1 of 0.002 and Q3 of 0.0025 reflects this variability.

In NSGA-2 presents a range of GD values from 0.0025 to 0.0027, with a median of 0.0026. The narrow range of these values suggests a consistent performance, although the values are generally higher than those of MOEA, indicating a slightly less close approximation to the Pareto front. NSGA-3 has the lowest median GD value of 0.0009, indicating a very close approximation to the Pareto front on average. The range of values is also quite tight for minimum of 0.0008 to maximum of 0.0011, with the interquartile range for Q1 of 0.0009 and Q3 of 0.001 suggesting consistent performance across runs. The NSGA-3 results show the closest convergence to the Pareto front as indicated by the GD measure, followed by MOEA. MEPSOLA and NSGA-2 also perform well, with MEPSOLA having a slightly wider range of values, suggesting a marginally less tight convergence compared to NSGA-3.

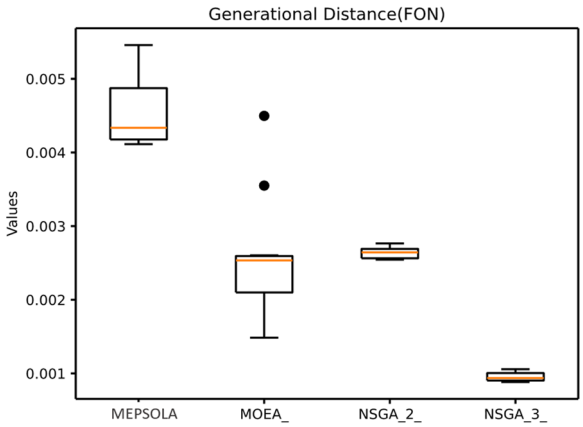


Figure 4.11: Generational Distance Measure Results for FON Function

4.4.2 Kursawe Function (KUR) Results

This subsection evaluates the performance of the proposed method on the KUR benchmark. The aim is to highlight their relative effectiveness and examine the intricacies of their performance in the demanding environment of the KUR function.

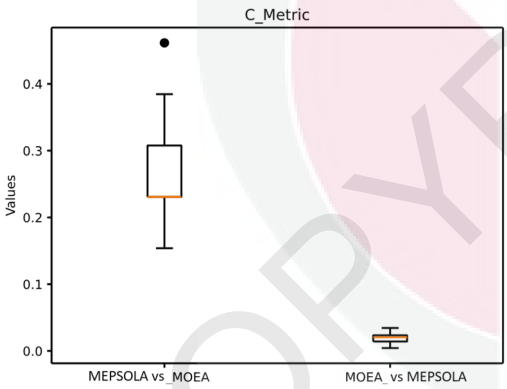
4.4.2.1 Set Coverage Measure

The analysis of the SC results for the KUR function, as illustrated in Figure 4.12 (a), shows that MEPSOLA significantly dominates the solutions provided by MOEA. The SC values range from a minimum of 0.15 to a maximum of 0.46, with a median of 0.23, indicating that a large number of MEPSOLA's solutions outperform MOEA's. Conversely, MOEA exhibits minimal dominance over MEPSOLA, with the highest SC value only reaching 0.03 and a median of 0.02, suggesting rare occurrences where MOEA's solutions surpass those of MEPSOLA.

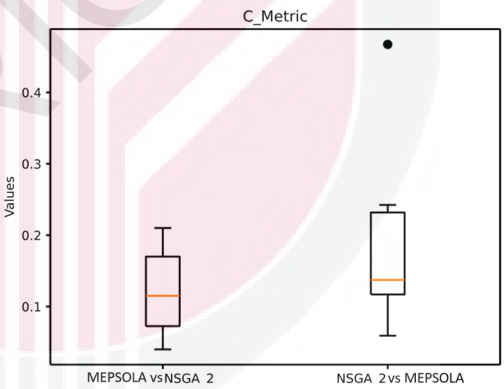
In comparison MEPSOLA against NSGA-2 in the Figure 4.12 (b), MEPSOLA displays some dominance, with SC values extending from 0.04 to 0.21. A median SC

of 0.12 and an interquartile range from Q1 of 0.07 to Q3 of 0.17 indicate that MEPSOLA's solutions frequently cover those from NSGA-2, albeit to a moderate extent. However, NSGA-2 shows a bit more dominance over MEPSOLA, with a peak SC value of 0.47, though with a moderate median of 0.14. Furthermore, the comparison of MEPSOLA against NSGA-3 as shown in the Figure 4.12 (c), MEPSOLA's solutions exhibit a dominance over NSGA-3 median SC value of 0.08, slightly less dominance compared to MOEA and NSGA-2. NSGA-3 demonstrates limited dominance over MEPSOLA with a median SC of 0.03.

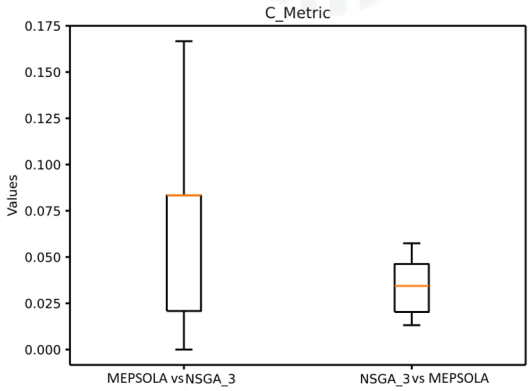
These SC outcomes for the KUR function reveal that MEPSOLA's solutions significantly outperform MOEA and moderately surpass NSGA-3, while NSGA-2 exhibits only minimal dominance over MEPSOLA.



(a). MEPSOLA vs MOEA



(b). MEPSOLA vs NSGA-2



(c). MEPSOLA vs NSGA-3

Figure 4.12: Set Coverage Measure Results for KUR Function

4.4.2.2 Hyper-Volume Measure

The results of the HV metric for the KUR function as shown in Figure 4.13, highlight MEPSOLA exceptional performance, featuring HV values ranging from a minimum of 5391.07 to a maximum of 18171.68. This range underscores MEPSOLA's capability to extensively cover the objective space. The median HV of 9397.94 indicates that MEPSOLA consistently produces solutions encompassing a significant area of the space, confirmed by the interquartile range from Q1 of 8693.74 to Q3 of 12604.38, demonstrating the algorithm's robust performance. In contrast, MOEA exhibits considerably lower HV values, peaking at just 225.33, which signals a very limited exploration of the objective space. The median HV value of 217.76, coupled with a narrow interquartile range extending from a Q1 of 217.43 and Q3 of 218.05, illustrates MOEA's limited capability in covering the space as MEPSOLA's solutions do.

The NSGA-2 result presents moderate HV values, with ranging from 1709.9 to 1743.53. The median HV of 1723.68 and a relatively narrow interquartile range from Q1 of 1718.39 to Q3 of 1729.16 indicate that NSGA-2 covers a greater portion of the objective space than MOEA, yet it does not reach the expansive coverage achieved by MEPSOLA. Similarly, NSGA-3's performance aligns more closely with MOEA's, displaying HV values from 194.57 to 198.73. The median HV of 196.68 and interquartile values from Q1 of 196.32 to Q3 of 197.02 suggest that NSGA-3's Pareto fronts are also confined in scope, covering only a minimal section of the objective space. In the analysis of the KUR function, MEPSOLA markedly surpasses MOEA, NSGA-2, and NSGA-3 in terms of hyper-volume. This demonstrates MEPSOLA's superior ability to generate solutions that not only closely approach the true Pareto front but also offer a comprehensive and diverse representation of optimal solutions.

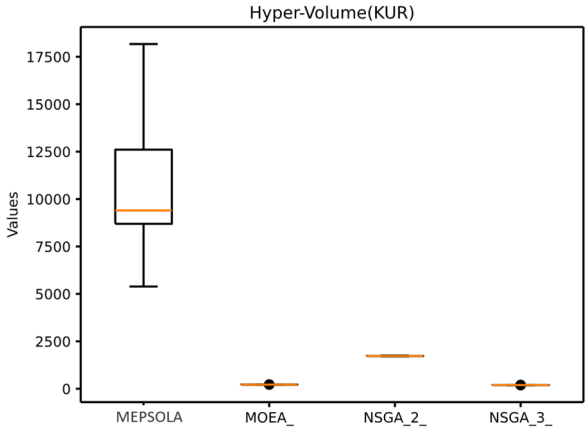


Figure 4.13: Hyper-Volume Measure Results for KUR Function

4.4.2.3 Delta Measure

The Delta Measure outcomes for the KUR function, as presented in Figure 4.14, highlight the distinct performance variations among the algorithms. MEPSOLA exhibits a substantial spread in its solution distribution, albeit with slight non-uniformity, marked by a minimum delta value of 0.936. This indicates a proficient exploration of the search space. Despite a maximum Delta value of 1.161 hinting at some irregularities in the spread, a high median of 1.105 indicates a generally broad and diverse distribution. The interquartile range, with Q1 at 1.053 and Q3 at 1.115, further attests to MEPSOLA's consistent and comprehensive performance. In contrast, MOEA's Delta values present a narrower scope, ranging from a minimum of 0.672 to a maximum of 0.705. In NSGA-2 exhibits a fair distribution, with Delta values spanning from 0.692 to 0.729 and a median of 0.707. The interquartile range from Q1 at 0.697 to Q3 at 0.719 points to reasonable uniformity in solution spread. While in NSGA-3 reveals the most narrowly focused distribution among the compared algorithms, with Delta values ranging narrowly from 0.628 to 0.631 and a median of

0.629. The quartile values, Q1 of 0.628 and Q3 of 0.63, indicate a highly concentrated distribution.

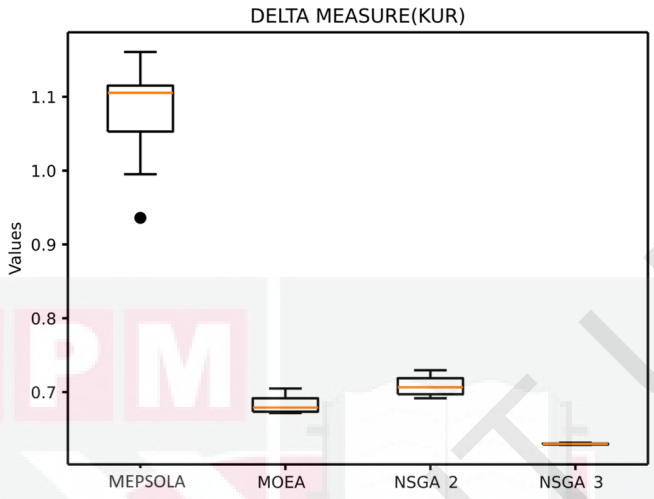


Figure 4.14: Delta Measure Results for KUR Function

4.4.2.4 Number of Non-Dominated Solutions Measure

In the NDS measure for the KUR function, MEPSOLA exhibits a robust ability to discover a wide array of non-dominated solutions, as illustrated in Figure 4.15. The results range from a minimum of 291 to a high of 933, with a median of 525.5, demonstrating MEPSOLA's consistent effectiveness in identifying numerous optimal solutions. The interquartile range, stretching from Q1 at 450 to Q3 at 687.7, reinforces this consistency. In stark contrast, MOEA identifies only a small number of non-dominated solutions, consistently capped at 13. The NSGA-2 maintains a steady count of 100 non-dominated solutions, indicating a moderate exploration capability. Similarly, NSGA-3 mirrors MOEA with a limited count of 12, suggesting a constrained exploration of optimal solutions.

In the KUR function results, MEPSOLA markedly surpasses MOEA, NSGA-2, and NSGA-3 in identifying a broad spectrum of non-dominated solutions. This wide range of NDS values underscores MEPSOLA's superior capacity to explore the solution space comprehensively, capturing an extensive array of Pareto-optimal solutions. This trait is crucial in MOO, ensuring an in-depth exploration of possible trade-offs between objectives.

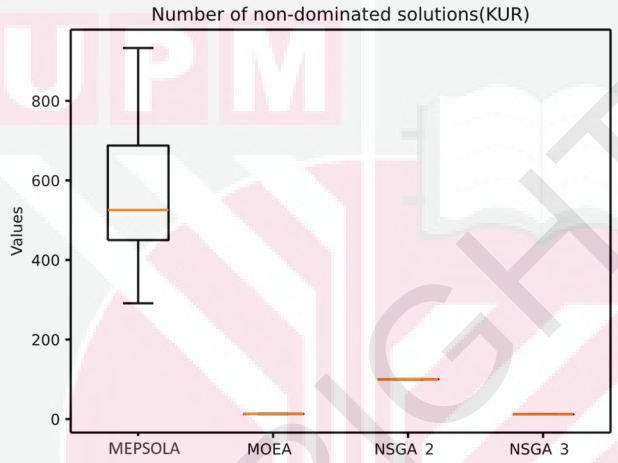


Figure 4.15: Number of Non-Dominated Solutions Results for KUR Function

4.4.2.5 Generational Distance Measure

The GD Measure for the KUR function, showcased in Figure 4.16, provides insightful observations. MEPSOLA exhibits a spectrum of GD values between 0.0096 and 0.0132, centering around a median value of 0. 0111. This range indicates that MEPSOLA's solutions generally lie close to the Pareto front, as further evidenced by the interquartile range stretching from 0.0106 for Q1 to 0.0125 for Q3, highlighting a consistent level of proximity across different runs. In comparison, MOEA exhibits a broader range of GD values, spanning from 0.0107 to 0.0198. Notably, the upper limit of this range is the highest among all the algorithms. The median GD value for MOEA,

at 0.0163, surpasses that of MEPSOLA, suggesting a slightly less consistent approximation to the Pareto front.

The NSGA-2, presents the lowest GD values, indicating a closer approximation to the Pareto front than the others. The values range from a minimum of 0.006 to a maximum of 0.007, reflecting NSGA-2's effectiveness in this regard. In NSGA-3's GD values are generally higher than those of NSGA-2, ranging from 0.0134 to 0.0156. The median value of 0.0146, along with quartile values of 0.0142 for Q1 and 0.0147 for Q3, suggests that NSGA-3's solutions are somewhat further from the Pareto front compared to NSGA-2.

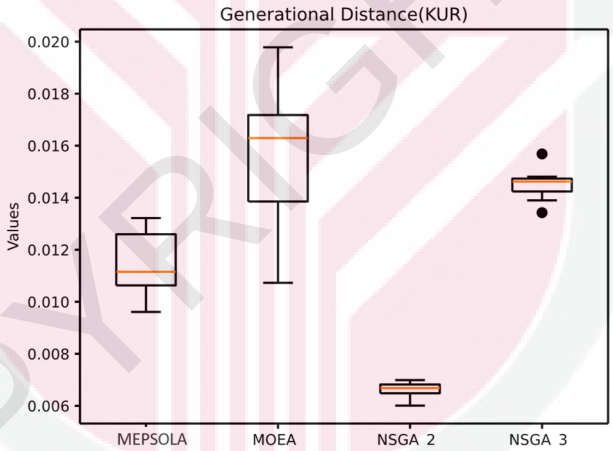


Figure 4.16: Generational Distance Results for KUR Function

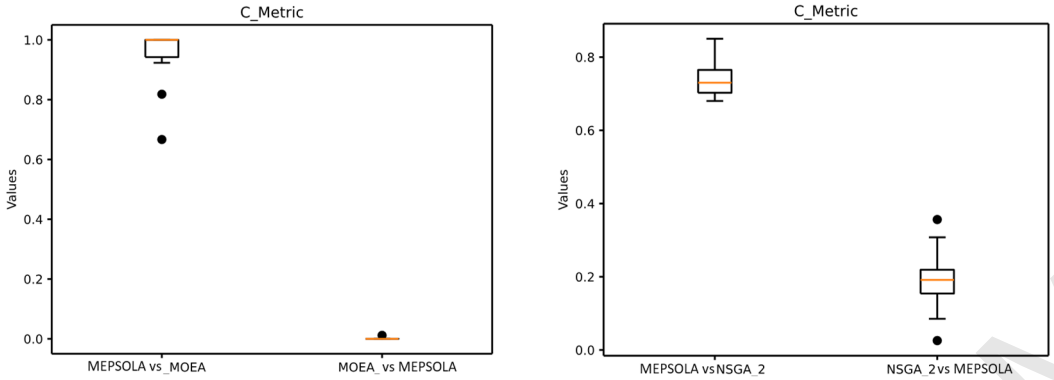
4.4.3 ZDT1 Function Results

In this section, MEPSOLA's performance on the ZDT1 function is examined, with comparisons drawn against three benchmark algorithms. The analysis is concentrated on the ability of each algorithm to generate diverse, optimal solutions and approximate the Pareto front. This highlights the strengths and potential limitations of MEPSOLA in this specific optimization scenario.

4.4.3.1 Set Coverage Measure

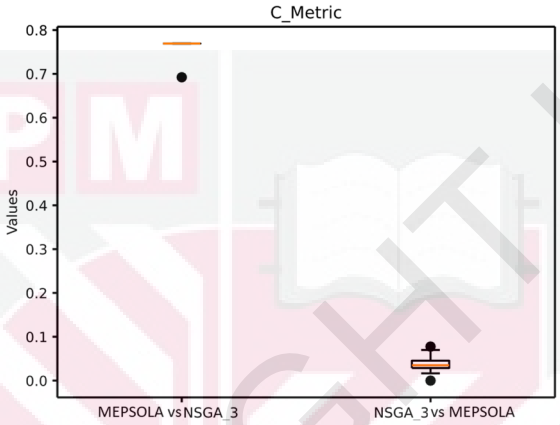
The SC results for the ZDT1 function, MEPSOLA exhibits significant dominance over MOEA as shown in Figure 4.17 (a). The minimum SC value is 0.6666, with the median, maximum, and Q3 values all reaching 1. This indicates that almost all of MEPSOLA's solutions dominate MOEA's. The first quartile value of 0.9423 further reinforces the consistency of this dominance. On the other hand, MOEA shows minimal dominance over MEPSOLA, with most SC values at 0 and a maximum of only 0.0116, reflecting an almost negligible coverage over MEPSOLA's solutions. In the comparison between MEPSOLA and NSGA-2 as shown in the Figure 4.17 (b), MEPSOLA again demonstrates dominance, with a minimum SC value of 0.68 and a maximum of 0.85. The median SC is 0.73, and the interquartile range from Q1 of 0.7025 to Q3 of 0.765 indicates a significant majority of MEPSOLA's solutions are superior to NSGA-2's. However, NSGA-2 does exhibit some dominance over MEPSOLA, although less obvious, with a maximum SC of 0.3563 and a median of 0.1913.

The results MEPSOLA against NSGA-3 in Figure 4.17 (c), MEPSOLA maintains strong dominance, with identical median and quartile SC values at 0.7692, demonstrating a consistent and substantial coverage. NSGA-3, in turn, shows a modest level of dominance over MEPSOLA, with a median SC of 0.0349. The SC results for the ZDT1 function highlight MEPSOLA's impressive capability in consistently outperforming MOEA, NSGA-2, and NSGA-3. MEPSOLA's solutions not only show diversity and proximity to the Pareto front but also exhibit a higher dominance level in direct comparisons with the solutions from these benchmark algorithms.



(a). MEPSOLA vs MOEA

(b). MEPSOLA vs NSGA-2



(c). MEPSOLA vs NSGA-3

Figure 4.17: Set Coverage Measure Results for ZDT1 function

4.4.3.2 Hyper-Volume Measure

The analysis of the ZDT1 function for HV, showcased in Figure 4.18, reveals MEPSOLA's notable performance, exhibiting HV values that span from a minimum of 193.22 to a maximum of 402.87. The median HV of 261.54 indicates that MEPSOLA consistently generates Pareto fronts that cover a substantial portion of the optimal set. The interquartile range, extending from Q1 of 203.97 to Q3 of 367.28, underscores the algorithm's consistent performance across different runs. Conversely, MOEA shows limited HV performance, with its highest HV value only reaching 60.51 and a median HV of 32.98. The quartile values, spanning from Q1 at 31.66 to Q3 at

40.16, suggest that MOEA's Pareto fronts are significantly less effective in covering the objective space compared to MEPSOLA.

In the case of NSGA-2, the HV values are more consistent, though they remain lower than those of MEPSOLA. With a minimum HV of 240.69 and a maximum of 243.88, the median HV of 241.663 indicates a moderate coverage of the objective space. The quartile values further reflect this consistency, with Q1 at 241.24 and Q3 at 242.39. In NSGA-3, on the other hand, exhibits the lowest HV values among the algorithms compared, with a narrow range from a minimum of 31.51 to a maximum of 31.62. The median HV of 31.55, along with quartile values of Q1 at 31.53 and Q3 at 31.56, indicate that NSGA-3's Pareto fronts are the least effective in spanning the objective space. The HV results for the ZDT1 function demonstrate MEPSOLA's superior capability in generating Pareto fronts that not only approach the true Pareto front but also span a broader and more significant portion of the optimal set.

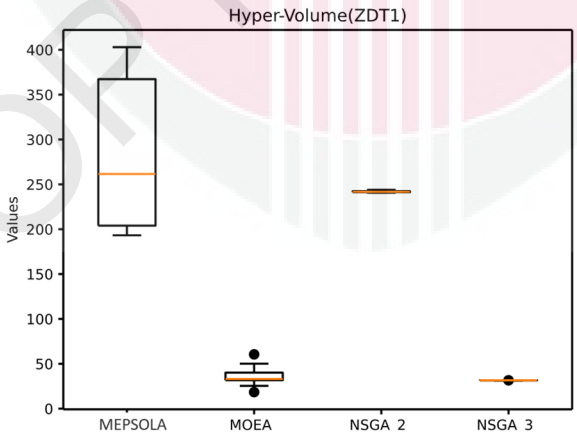


Figure 4.18: Hyper-Volume Measure Results for ZDT1 function

4.4.3.3 Delta Measure

The assessment of the delta measure for the ZDT1 function, presented in Figure 4.19, demonstrates MEPSOLA's commendable performance, highlighting its ability to evenly distribute solutions along the Pareto front. The minimum delta value for MEPSOLA is relatively high at 0.899, and it reaches up to 1.080, signifying a broad but slightly varied distribution along the Pareto front. The median value of 1.025 and quartile values, Q1 at 0.988 and Q3 at 1.049. In contrast, MOEA shows a slightly less delta value than MEPSOLA, with a minimum of 0.6818 and a maximum value of 1.138. The median value for MOEA is 0.8936. The range between Q1 of 0.8558 and Q3 of 0.9835 shows.

The results for NSGA-2 and NSGA-3 both exhibit lower Delta values. NSGA-2's Delta values range from 0.6512 to 0.6939 with a median of 0.6663. Similarly, NSGA-3 has Delta values from 0.6662 to 0.6685 with a median of 0.6671. Despite the slightly higher Delta values, MEPSOLA's overall performance in the ZDT1 function remains robust, as evidenced by its strong outcomes in SC, HV, and the NDS metrics. This consistency across multiple performance indicators highlights MEPSOLA's effectiveness in MOO, balancing between exploration and the generation of diverse, optimal solutions.

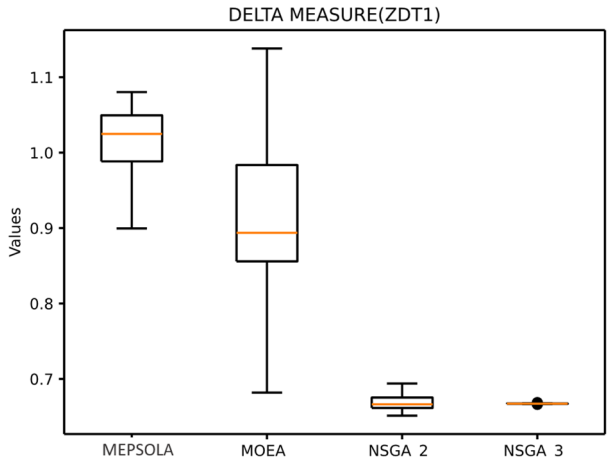


Figure 4.19: Delta Measure Results for ZDT1 Function

4.4.3.4 Number of Non-Dominated Solutions Measure

The NDS metric results for the ZDT1 function illustrated in Figure 4.20. MEPSOLA shows a strong performance, with a minimum of 78 and a maximum of 140 non-dominated solutions. The median of 92.5. The interquartile range, with Q1 at 86.25 and Q3 at 116.5, suggests that MEPSOLA consistently identifies a high number of quality solutions. In contrast, MOEA has much fewer non-dominated solutions, with all statistical values being notably lower than MEPSOLA's, ranging from a minimum of 11 to a maximum of 13. The median and quartile values are consistently around 13, indicating that MOEA struggles to find a diverse set of optimal solutions.

In NSGA-2 presents a consistent number of 100 non-dominated solutions across all statistics, which is higher than MOEA's performance but less than MEPSOLA's maximum value. In NSGA-3 also shows a lower range of NDS, similar to MOEA, with a 13 NDS across all statistics, which indicates a limited exploration of the solution space. The ZDT1 function, MEPSOLA outperforms MOEA, NSGA-2, and NSGA-3 in terms of the identified NDS, demonstrating its effectiveness in exploring the search space and finding a greater number of high-quality solutions.

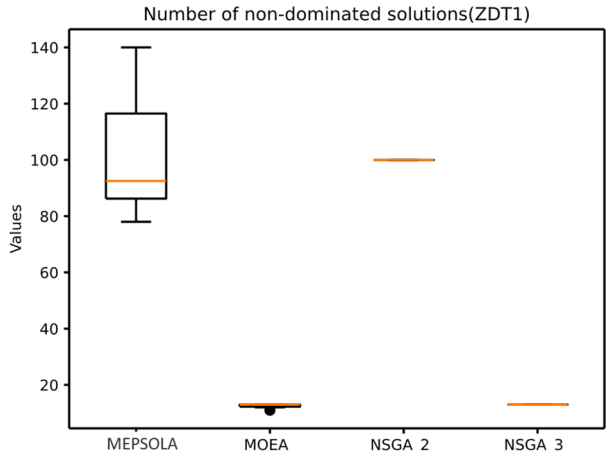


Figure 4.20: Number of Non-Dominated Solutions Results for ZDT1 Function

4.4.3.5 Generational Distance Measure

The GD results for the ZDT1 function, as depicted in Figure 4.21, reveal interesting insights about the performance of MEPSOLA in comparison to other algorithms. MEPSOLA exhibits a range of GD values, from a minimum of 0.0212 to a maximum of 0.041, suggesting a close yet somewhat variable approximation to the Pareto front. The median GD value for MEPSOLA stands at 0.0263, with quartile values at 0.0247 for Q1 and 0.0356 for Q3. On the other hand, MOEA shows a wider variation in GD values, ranging from 0.0198 to a significantly higher 0.1782. This broad range indicates inconsistencies in MOEA's performance, with some runs deviating notably from the Pareto front. The median GD value for MOEA is 0.0711, considerably higher than that of MEPSOLA, and the interquartile range from 0.0409 for Q1 to 0.1018 for Q3 further underscores its inconsistent approximation to the Pareto front.

In contrast, NSGA-2 and NSGA-3 display a low GD value, indicating a more consistent proximity to the Pareto front. NSGA-2's median GD is just 0.0004, and NSGA-3's is 0.0008, both significantly lower than MEPSOLA and MOEA.

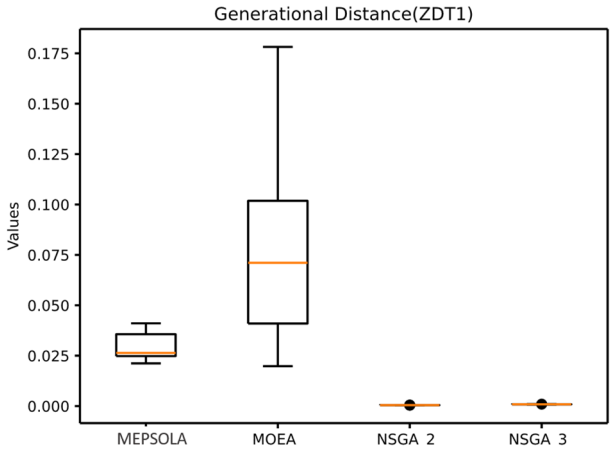


Figure 4.21: Generational Distance Results for ZDT1 Function

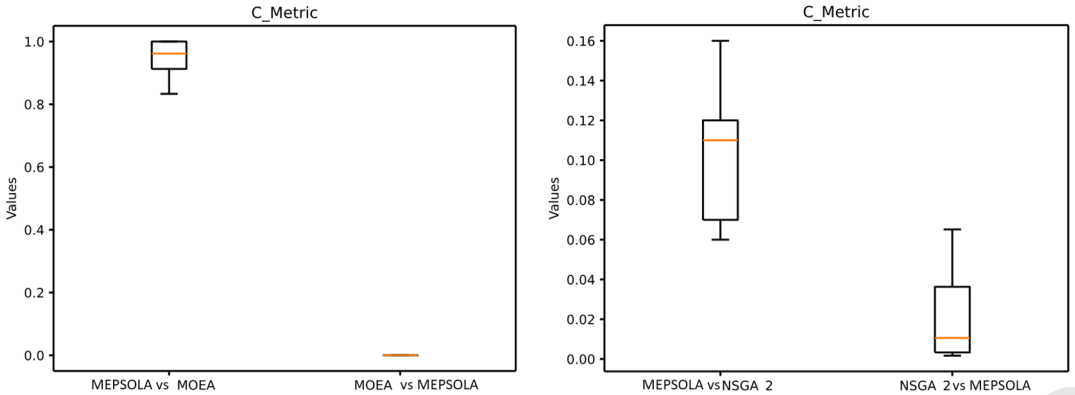
4.4.4 ZDT2 Function Results

This subsection explores into the results for the ZDT2 function, known for its convex Pareto front, which provides a challenging benchmark to evaluate the efficiency of MOO algorithms in generating solutions that closely align with the Pareto front.

4.4.4.1 Set Coverage Measure

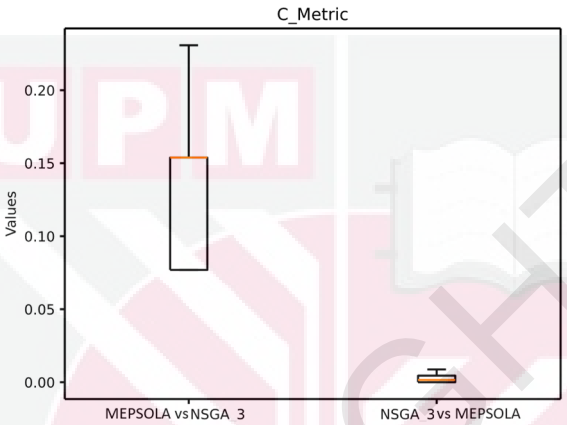
The SC measure for the ZDT2 function, presented in Figure 4.22, reveals notable results. For MEPSOLA against MOEA in the Figure 4.22 (a), MEPSOLA exhibits significant dominance. The SC values range impressively, with a minimum of 0.8333, a median of 0.9615, and a maximum reaching 1. This indicates that most of MEPSOLA's solutions are dominant over those from MOEA. The quartile values, with Q1 at 0.9125 and Q3 at 1, further confirm this dominance. Conversely, MOEA shows no dominance over MEPSOLA's solutions, with all its values at 0.

In the comparison of MEPSOLA against NSGA-2 in the Figure 4.22 (b), MEPSOLA's solutions again show dominance, although to a lesser degree than with MOEA. The minimum SC value here is 0.06, with the median at 0.11 and the maximum at 0.16. The interquartile range, from Q1 at 0.07 to Q3 at 0.12, suggests that MEPSOLA's solutions consistently dominate those from NSGA-2 to a considerable extent. On the other hand, NSGA-2 demonstrates minimal dominance over MEPSOLA, with its highest SC value only at 0.0651. When comparing MEPSOLA against NSGA-3 as in the Figure 4.22 (c), MEPSOLA's solutions display some level of dominance, with a median SC value of 0.1538 and a maximum of 0.2308, indicating substantial dominance in some instances. However, the dominance is less noticeable than against MOEA. In the reverse comparison of NSGA-3 against MEPSOLA, NSGA-3 shows only minimal dominance with a median SC value of 0.0015. The SC results for the ZDT2 function underscore MEPSOLA's effectiveness in generating solution sets with higher dominance compared to MOEA, NSGA-2, and NSGA-3.



(a). MEPSOLA vs MOEA

(b). MEPSOLA vs NSGA-2



(c). MEPSOLA vs NSGA-3

Figure 4.22: Set Coverage Measure Results for ZDT2 Function

4.4.4.2 Hyper-Volume Measure

The HV metric for the ZDT2 function, presented in Figure 4.23, highlights MEPSOLA's impressive HV performance, indicating its comprehensive coverage of the objective space. The algorithm's minimum HV value is 693.59, reaching an impressive maximum of 1365.44. The median HV of 957.47 underscores MEPSOLA's consistent ability to achieve considerable coverage in its runs, as evidenced by the interquartile range stretching from 803.03 to 1042.94. In stark contrast, MOEA displays significantly lower HV values in all aspects, peaking at just 19.67 with a median of 15.19, indicating its Pareto fronts' limited scope compared to MEPSOLA's expansive coverage.

Meanwhile, NSGA-2's results show consistency in HV values, yet exhibit a much narrower range, suggesting a more concentrated but less comprehensive coverage of the objective space. The median HV for NSGA-2, at 192.45, falls notably short of MEPSOLA's. Similarly, NSGA-3 also presents lower HV values with a median of 25.16, albeit slightly higher than MOEA's but substantially lower than MEPSOLA's. These outcomes highlight that NSGA-3's Pareto fronts also cover a lesser portion of the optimal front. The HV results for the ZDT2 function clearly indicate MEPSOLA's superior performance in terms of hyper-volume significantly surpassing MOEA, NSGA-2, and NSGA-3 in this regard.

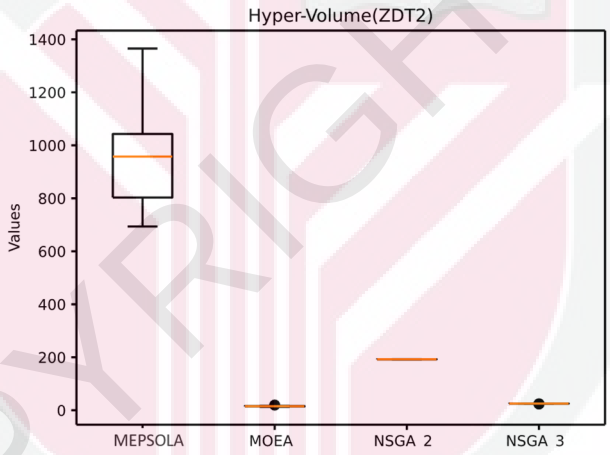


Figure 4.23: Hyper-Volume Measure Results for ZDT2 Function

4.4.4.3 Delta Measure

In the delta measure results analysis for the ZDT2 function, as depicted in Figure 4.24, MEPSOLA's delta values range between 0.981 and 1.189, with a median value of 1.079 and an interquartile range, where Q1 is at 1.043 and Q3 at 1.109. These results may indicate a substantial spread across the Pareto front. While higher delta values might suggest less uniformity, they also highlight MEPSOLA's ability to maintain

solution diversity. In comparison, MOEA's delta range, extending from 0.828 to 1.417, exhibits some variability. Its median value, at 1.081, is higher compared to MEPSOLA's median. However, the broad quartile values, with Q1 at 1.029 and Q3 at 1.236, may suggest less consistency in the spread of its solutions.

The NSGA-2 results shows a more confined range of delta values, from 0.651 to 0.683, leading to a median of 0.663. This tighter range reflects a more uniform distribution, although it might be less diverse. The interquartile values, Q1 at 0.656 and Q3 at 0.673, further support the indication of a more focused distribution. Furthermore, NSGA-3 demonstrates the most narrowly ranged delta values, between 0.646 and 0.648. This minimal variation, along with closely aligned median and quartile values, indicates a highly focused distribution of solutions, potentially with a reduced level of diversity compared to the other algorithms. This conclusion is drawn based on the results of other measures.

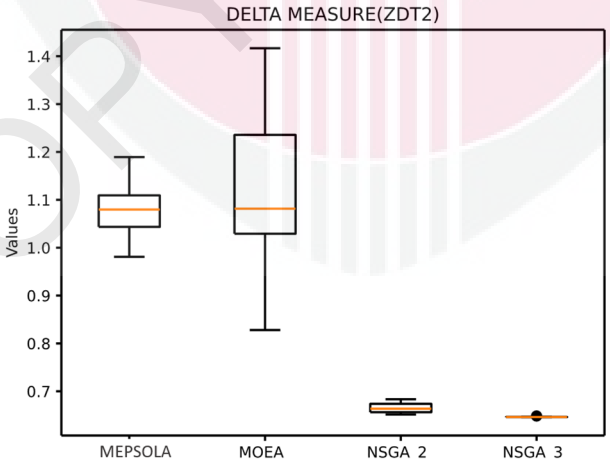


Figure 4.24: Delta Measure Results for ZDT2 Function

4.4.4.4 Number of Non-Dominated Solutions Measure

The results for the NDS metric on the ZDT2 function, presented in Figure 4.25, showcase MEPSOLA's exceptional ability to identify a broad spectrum of non-dominated solutions. It records a minimum NDS of 394 and a maximum of 706, with the median standing at 513. This pattern indicates MEPSOLA's consistent effectiveness in not only finding a significant number of optimal solutions but also in exploring a diverse range of them. The interquartile range, stretching from Q1 of 459.25 to Q3 of 587.75, underscores the reliability of MEPSOLA in extensively covering the solution space. Contrastingly, MOEA demonstrates a significantly lower capacity for identifying non-dominated solutions, with its NDS ranging narrowly from 11 to 13. This uniformity points to MOEA's limited scope in exploring the solution space, particularly when compared to MEPSOLA's performance.

The NSGA-2 maintains a steady count of 100 non-dominated solutions across all measures, indicating a consistent but moderate ability to identify a range of optimal solutions, albeit lower than MEPSOLA's. Similarly, NSGA-3's performance mirrors that of MOEA, with all NDS values consistently fixed at 13. This result suggests NSGA-3's focused yet limited exploration of the solution space. MEPSOLA notably outshines MOEA, NSGA-2, and NSGA-3 in terms of identifying non-dominated solutions for the ZDT2 function.

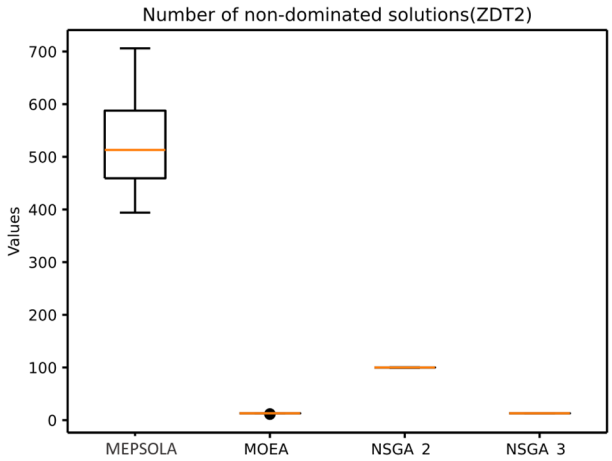


Figure 4.25: Number of Non-Dominated Solutions Results for ZDT2 Function

4.4.4.5 Generational Distance Measure

The GD results analysis for the ZDT2 function, as depicted in Figure 4.26, highlights MEPSOLA's effective approximation of the Pareto front. MEPSOLA's GD values range narrowly from a minimum of 0.0011 to a maximum of 0.002, signaling consistently close proximity to the true Pareto front. The median GD of 0.0017, coupled with the interquartile range for Q1 of 0.0014 and Q3 of 0.00189, underscores MEPSOLA's consistent and accurate alignment with the Pareto front.

In contrast, the MOEA exhibits a broader variation in GD values, with a peak of 0.0145. This significant range suggests that some of MOEA's solutions deviate farther from the Pareto front compared to other algorithms. The median GD of 0.0046 and the quartile range Q1 of 0.0035 and Q3 of 0.0056 point to a variable, though occasionally close, proximity to the Pareto front, not matching MEPSOLA's consistency.

However, NSGA-2 demonstrates the smallest GD values, suggesting a close approximation to the Pareto front. With values ranging from 0.00048 to 0.00053 and

a median of 0.00049, NSGA-2 might outperform MEPSOLA in terms of proximity to the Pareto front, although this does not necessarily reflect the breadth of the solution diversity. Similarly, NSGA-3 also records low GD values, akin to NSGA-2, which implies nearness to the Pareto front. The median GD of 0.00092 and the interquartile range of 0.0009 demonstrate consistent performance, albeit slightly less proximate than NSGA-2.

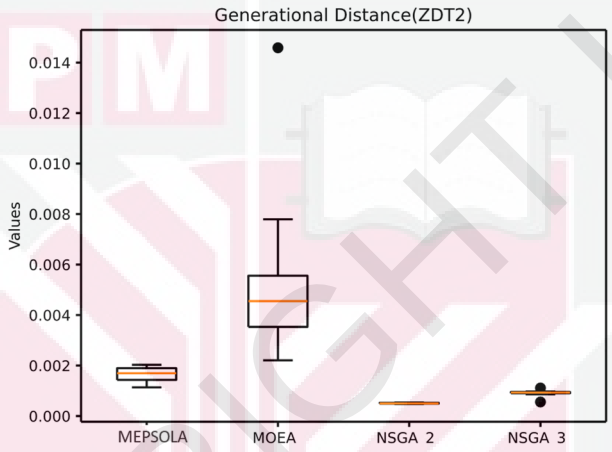


Figure 4.26: Generational Distance Results for ZDT2 Function

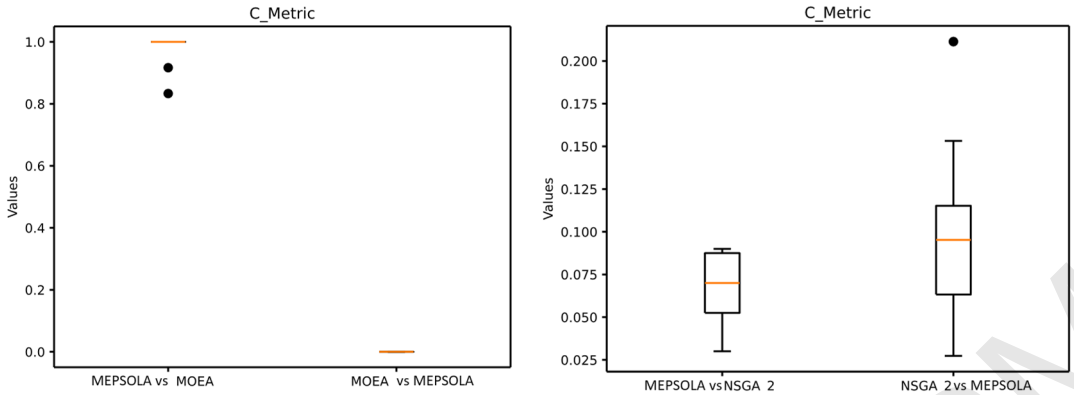
4.4.5 ZDT3 Function Results

This subsection presents the findings for the ZDT3 function, notable for its unique features such as multiple disconnected Pareto fronts. This phase of the evaluation aims to highlight the nuanced capabilities of the proposed algorithm in managing the ZDT3 function's unique challenges.

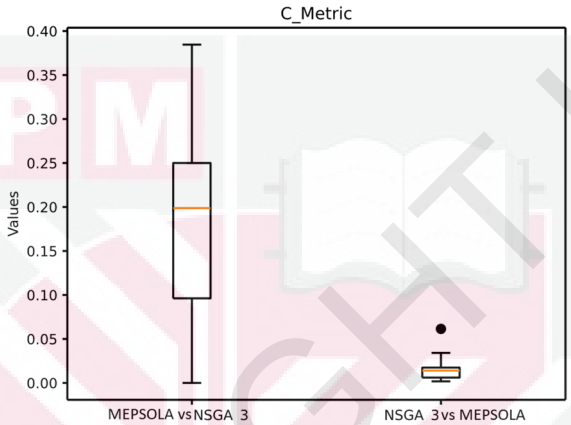
4.4.5.1 Set Coverage Measure

The SC measure results for the ZDT3 function, as presented in Figure 4.27 (a) for the MEPSOLA against MOEA comparison, reveal a pronounced dominance by MEPSOLA. The SC values indicate a minimum of 0.83, with both the median and maximum reaching 1, and quartile values aligning accordingly. This suggests that nearly all of MEPSOLA's solutions dominate those of MOEA. Conversely, the SC values for MOEA against MEPSOLA consistently remain at 0, indicating no dominance of MOEA over MEPSOLA. The MEPSOLA against NSGA-2 comparison in Figure 4.27 (b), MEPSOLA's solutions exhibit some performance with a minimum SC value of 0.03, a maximum of 0.09, and a median of 0.07. The interquartile range, from Q1 of 0.05 to Q3 of 0.087, reflects MEPSOLA's competitive standing. Conversely, NSGA-2's solutions demonstrate some dominance over MEPSOLA's, evident in the max SC value of 0.21; however, the median of 0.09 indicates limited dominance.

In contrast, in the MEPSOLA against NSGA-3 comparison in Figure 4.27 (c), MEPSOLA displays a degree of dominance over NSGA-3, as evidenced by a maximum SC value of 0.38. The median SC of 0.19 and Q3 of 0.25 suggest that while NSGA-3's dominance is present, it is not overwhelming. In the reverse comparison, NSGA-3's solutions show less dominance over MEPSOLA's, with a median SC of 0.014. The SC results for the ZDT3 function reveal that MEPSOLA generally produces a more dominant set of solutions compared to MOEA and NSGA-2. NSGA-3 shows some dominance, but not as consistent across all runs.



(a). MEPSOLA vs MOEA (b). MEPSOLA vs NSGA-2



(c). MEPSOLA vs NSGA-3

Figure 4.27: Set Coverage Measure Results for ZDT3 Function

4.4.5.2 Hyper-Volume Measure

The HV measure results for the ZDT3 function, shown in Figure 4.28, showcase MEPSOLA's outstanding performance, with MEPSOLA achieving significant HV values that highlight its capacity to comprehensively cover the objective space. The HV values range from a minimum of 809.54 to a maximum of 1377.98, with a median at 1113.17. This range, with an interquartile spread from Q1 at 1043.81 to Q3 at 1317.08, confirms MEPSOLA's consistent and expansive coverage, demonstrating its strong convergence and diversity. In contrast, MOEA displays significantly lower HV values, suggesting its Pareto fronts are considerably less space covered. The highest

HV for MOEA is just 63.96, with a median of 33.23. The quartile values, ranging from Q1 at 24.99 to Q3 at 39.97, point towards a limited solution spread and a less effective exploration of the objective space.

In NSGA-2, while exhibiting more uniform HV values, falls short of MEPSOLA's performance. Its HV values span a smaller range, with a minimum of 255.75 and a maximum of 258.56. The median HV of 257.28 and a close interquartile range from Q1 of 256.63 and Q3 of 257.75 indicate a moderate coverage of the objective space. In NSGA-3 registers the lowest HV values among the algorithms examined, with a median of 33.35. This finding implies that NSGA-3's Pareto fronts are not as comprehensive or diverse as those generated by MEPSOLA. The HV results for the ZDT3 function clearly demonstrate MEPSOLA's superior capability in generating wide-ranging and diverse Pareto fronts. MEPSOLA excels in capturing a broad set of optimal solutions and effectively balances the competing objectives of multi-objective optimization, outperforming the other algorithms in terms of both convergence and diversity.

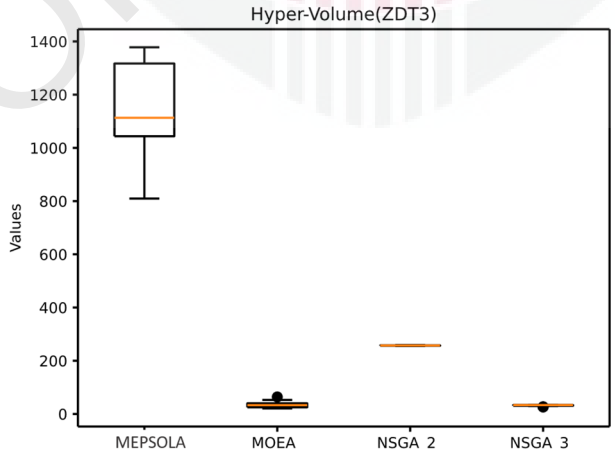


Figure 4.28: Hyper-Volume Measure Results for ZDT3 Function

4.4.5.3 Delta Measure

The delta measure results for ZDT3 function, as shown in Figure 4.29. MEPSOLA demonstrates a relatively broad spread of solutions, with a minimum delta value of 1.067 and a maximum of 1.159. The median value of 1.125 and interquartile range for Q1 of 1.108 and Q3 of 1.138 suggest that MEPSOLA maintains a diverse range of solutions. In contrast, the delta values for MOEA range from 0.784 to 0.915, indicating a tighter distribution compared to MEPSOLA. The median value of 0.84 and interquartile values for Q1 of 0.8 and Q3 of 0.877 suggest a more uniform distribution of solutions but less diversity.

The NSGA-2 displays a more uniform spread of solutions with delta values ranging from 0.737 to 0.783. The median of 0.77 suggests a consistent but narrower range of solutions compared to MEPSOLA. Meanwhile, NSGA-3 shows the most uniform distribution among the compared algorithms, with delta values ranging narrowly from 0.646 to 0.648 and a median of 0.646. This tight clustering indicates a highly uniform but less diverse set of solutions.

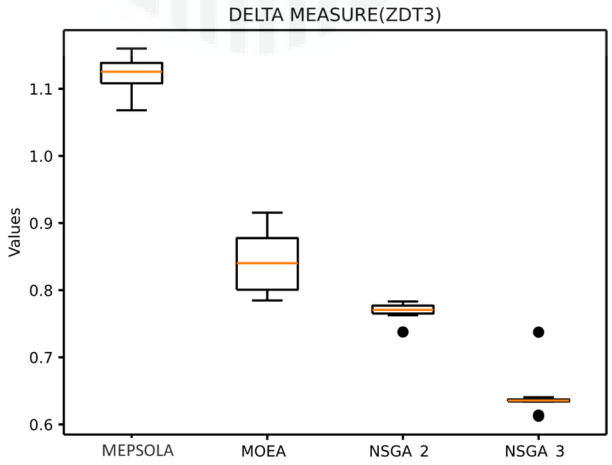


Figure 4.29: Delta Measure Results for ZDT3 Function

4.4.5.4 Number of Non-Dominated Solutions Measure

The NDS measure results for ZDT3 function as shown in Figure 4.30 reveal MEPSOLA remarkable performance in identifying non-dominated solutions. Its NDS values vary from a minimum of 359 to a maximum of 586, indicating a broad exploration of the solution space. The median NDS value at 485.5, coupled with an interquartile range from Q1 at 442.75 to Q3 at 541, underscores MEPSOLA's consistent and comprehensive performance in unveiling a wide array of Pareto-optimal solutions. In contrast, MOEA shows a significantly lower NDS range, oscillating only between 11 and 13. This narrow spread, with a consistent median at 12, suggests MOEA's limited capability in exploring the solution space, resulting in a smaller set of optimal solutions compared to MEPSOLA.

However, NSGA-2 maintains a steady count of 100 non-dominated solutions across all measures, denoting a stable but notably lesser capacity to identify optimal solutions than MEPSOLA. Similarly, NSGA-3 mirrors MOEA in its NDS performance, with a minimal range from 12 to 13 and a fixed median and Q3 value of 13. Like MOEA, NSGA-3's ability to identify diverse non-dominated solutions is considerably constrained compared to MEPSOLA. The NDS analysis for the ZDT3 function distinctly showcases MEPSOLA's superiority over MOEA, NSGA-2, and NSGA-3 in pinpointing a larger array of non-dominated solutions. MEPSOLA's consistent achievement in identifying a higher number of optimal solutions attests to its robustness and efficacy as a MOO algorithm, adeptly navigating the complexities inherent in such problems.

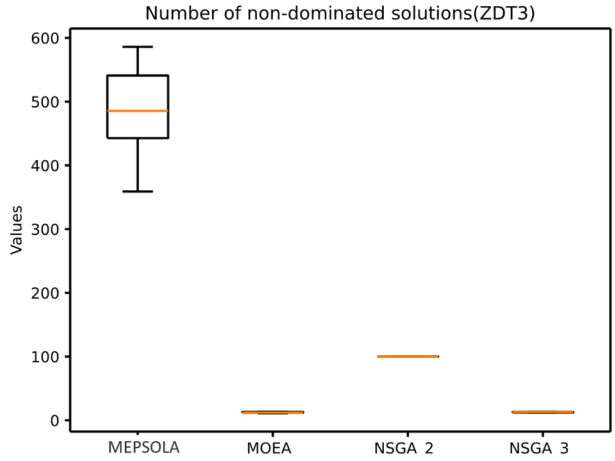


Figure 4.30: Number of Non-Dominated Solutions Results for ZDT3 Function

4.4.5.5 Generational Distance Measure

The GD measure results for the ZDT3 function, as shown in Figure 4.31, shows MEPSOLA's superior convergence towards the Pareto front. This is evident from the very low GD values it achieves. MEPSOLA maintains a minimum GD of 0.001 and a maximum of 0.0019, with a median of 0.0012, demonstrating that MEPSOLA consistently produces solutions that are close to the true Pareto front across its runs. In contrast, MOEA exhibits significantly higher GD values, with a maximum reaching 0.236, indicating that some of its runs are far from the optimal front. The median GD for MOEA is 0.0678, and the wider interquartile range for Q1 of 0.0255 and Q3 of 0.1088 suggests greater inconsistency in convergence compared to MEPSOLA. Meanwhile, NSGA-2 demonstrates a good level of convergence, with GD values ranging from 0.00069 to 0.00076 with median of 0.00073 and narrow quartile range imply consistent performance, although it is not as close to the Pareto front as MEPSOLA. In NSGA-3, higher GD values than MEPSOLA are observed, with a median of 0.0035. While the maximum value was 0.0165, indicating less consistent convergence across runs but higher than MEPSOLA’s.

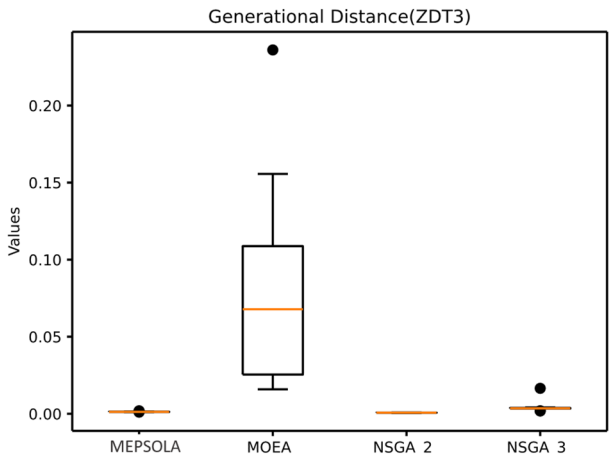


Figure 4.31: Generational Distance Results for ZDT3 Function

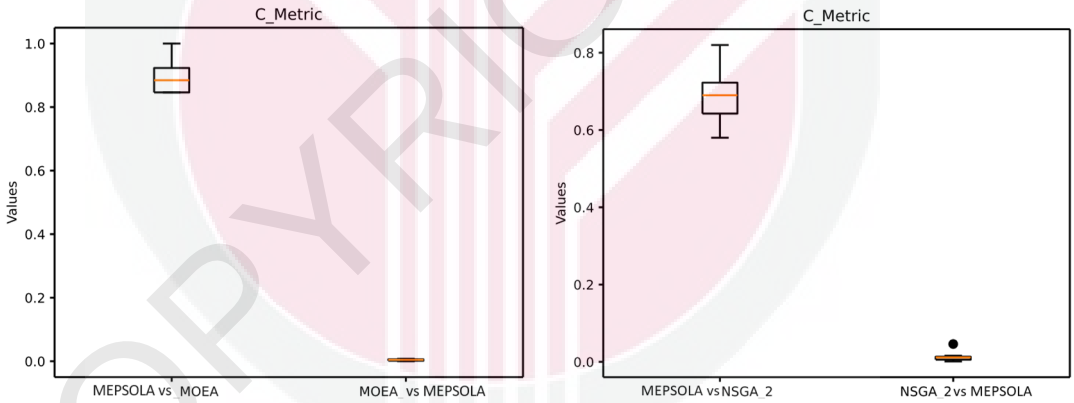
4.4.6 ZDT6 Function Results

The final segment of the results section evaluates the performance of MOO algorithms on the ZDT6 function. This analysis explores each algorithm's ability to handle the complexities of ZDT6, particularly focusing on the challenge of maintaining diversity across the Pareto front.

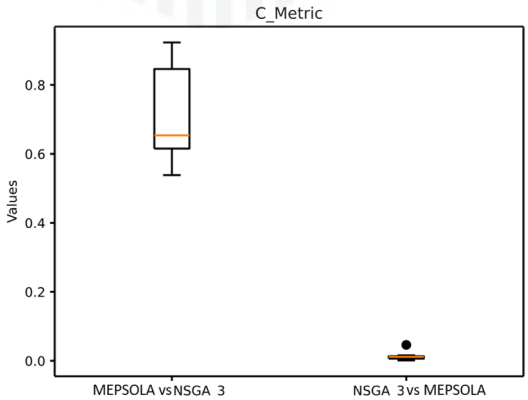
4.4.6.1 Set Coverage Measure

The SC measure outcomes for the ZDT6 function, as shown in Figure 4.32 (a), highlight MEPSOLA's substantial lead in SC measures over MOEA. MEPSOLA records a minimum coverage value of 0.85, achieves a perfect maximum of 1, and holds a median value of 0.88. These statistics consistently emphasize MEPSOLA's superiority over MOEA's solution sets. Conversely, MOEA's comparison to MEPSOLA shows its maximum coverage plateauing at a mere 0.01, starkly underlining MEPSOLA's unequivocal dominance.

Meanwhile, for MEPSOLA compare to NSGA-2 comparison as in the Figure 4.32 (b), MEPSOLA maintains its superior position with a minimum coverage of 0.58, a maximum of 0.82, and a median of 0.69. These values reinforce MEPSOLA's consistent dominance over NSGA-2, where in NSGA-2, the minimum is 0.001, the maximum is 0.046, and the median is 0.011. Similarly, in the MEPSOLA against NSGA-3 comparison as shown in the Figure 4.33 (c), MEPSOLA demonstrates robust performance with a minimum of 0.53, a maximum of 0.92, and a median value of 0.65, asserting its ability to generate more dominant solution sets. In NSGA-3, the minimum is 0.001, the maximum is 0.046, and the median is 0.01. The SC results in the challenging context of the ZDT6 function underscore MEPSOLA's robustness in producing comprehensive and dominant solution sets against all algorithms.



(a). MEPSOLA vs MOEA (b). MEPSOLA vs NSGA-2



(c). MEPSOLA vs NSGA-3

Figure 4.32: Set Coverage Measure Results for ZDT6 Function

4.4.6.2 Hyper-Volume Measure

The results for the HV measure on the ZDT6 function, as depicted in Figure 4.33, underscore MEPSOLA's impressive performance, with a minimum HV of 1519.47 and a maximum reaching an extraordinary 8355.51. The median HV of 2862.66, along with an interquartile range from 2471.91 to 5418.49, illustrates MEPSOLA's robust convergence to the Pareto front and its sustained capability to offer a diverse solution set across runs. Contrarily, MOEA presents substantially lower HV values, peaking at only 119.12 with a median of 52.39. This indicates MOEA's limited success in encompassing a comprehensive and evenly distributed solution set comparable to MEPSOLA's.

The results for NSGA-2 and NSGA-3 reveal even more modest HV values, signifying a constrained exploration of the objective space. With NSGA-2's median HV at 181.9 and NSGA-3's at 23.73, both algorithms exhibit reduced convergence and diversity in their solutions relative to MEPSOLA.

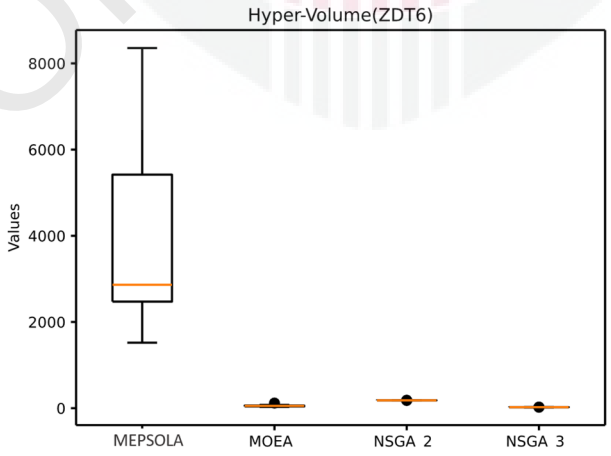


Figure 4.33: Hyper-Volume Measure Results for ZDT6 Function

4.4.6.3 Delta Measure

The results for the delta measure on the ZDT6 function, as shown in Figure 4.34, reveal MEPSOLA displaying a range of delta values from a minimum of 1.227 to a maximum of 1.385, with a median value of 1.301. This range suggests less uniformity in the distribution of its solutions, and could indicating a diverse exploration of the solution space regarding to the number of solutions. The MOEA, with delta values extending up to 1.256 and a median of 1.046, demonstrates a slightly better delta value than MEPSOLA. Quartile values of MOEA, ranging from 1.029 to 1.236, indicate a level of performance fluctuation in terms of solution uniformity.

Conversely, NSGA-2 and NSGA-3 achieve more uniform distributions, as evidenced by their significantly lower median delta values of 0.636 and 0.598, respectively. This indicates a closer alignment with the ideal of even distribution across the Pareto front. However, MEPSOLA's comparatively slightly higher delta values should be viewed in light of its superior performance in other key metrics such as SC, HV, and the NSD.

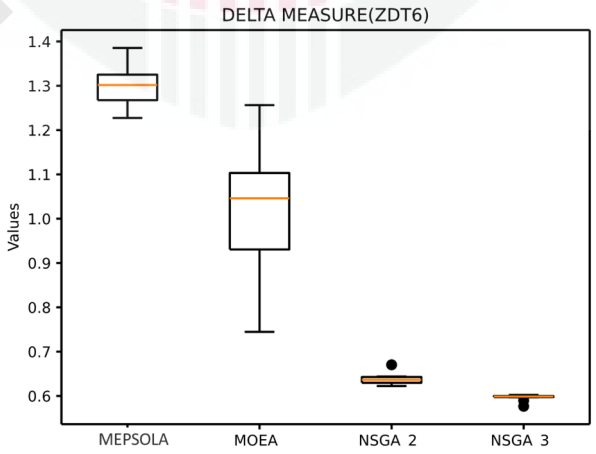


Figure 4.34: Delta Measure Results for ZDT6 Function

4.4.6.4 Number of Non-Dominated Solutions Measure

The results for the NDS metric on the ZDT6 function, presented in Figure 4.35, highlight MEPSOLA's remarkable ability to identify a wide array of non-dominated solutions, ranging from a minimum of 298 to a maximum of 895. The median of 441.5 highlights MEPSOLA's consistent success in finding a substantial number of high-quality solutions, as evidenced by the first and third quartiles at 316 and 682.25, respectively.

Contrastingly, MOEA and NSGA-3 exhibit a markedly lower efficiency, with all values steadfast at 13, indicating a constrained exploration of the solution space and, consequently, a much narrower set of optimal solutions. NSGA-2, while demonstrating a stable performance, identifies a modest 100 non-dominated solutions uniformly across all statistical benchmarks, surpassing MOEA and NSGA-3 but not achieving the breadth or quantity of solutions discerned by MEPSOLA. This analysis clearly illustrates MEPSOLA's superior ability to generate a larger and more varied set of non-dominated solutions for the ZDT6 function, underscoring its dominance over MOEA, NSGA-2, and NSGA-3.

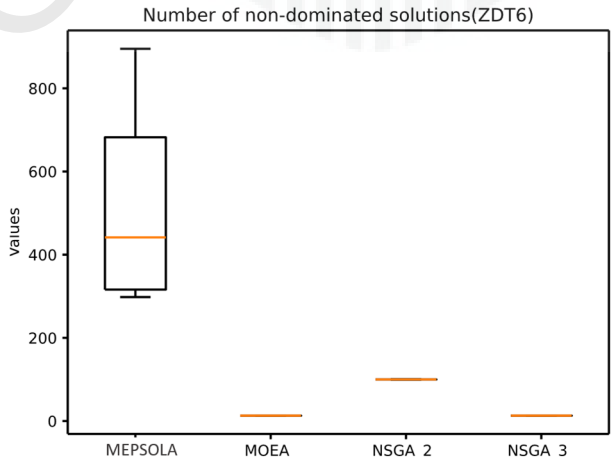


Figure 4.35: Number of Non-Dominated Solutions Results for ZDT6 Function

4.4.6.5 Generational Distance Measure

The results for the Generational Distance (GD) metric on the ZDT6 function, depicted in Figure 4.36, showcase MEPSOLA's notable performance in closely approximating the Pareto front. MEPSOLA's GD values range from a minimum of 0.0346 to a maximum of 0.1123, with a median of 0.0553. The interquartile range, from 0.044 to 0.0637, underscores its consistent proximity to the optimal solutions. Conversely, MOEA exhibits a wider variation in GD values, peaking at 0.4836, indicating variance in its ability to closely approximate the Pareto front. The median GD of 0.14599 for MOEA significantly exceeds that of MEPSOLA, revealing lesser consistency in approximation.

NSGA-2 and NSGA-3 present lower minimum GD values, indicating their potential for close approximation to the Pareto front in optimal scenarios. However, their median and quartile values do not consistently mirror MEPSOLA's closeness. Specifically, NSGA-2's median GD of 0.0006 and NSGA-3's median GD of 0.0028, though superior to MOEA's performance, do not consistently match MEPSOLA's proximity to the Pareto front. This analysis underscores that MEPSOLA's better results than MOEA and slightly higher than NSGA-2 and NSGA-3 values should be viewed in light of its superior performance in other key metrics such as SC, HV, and NSD, as well as its well-balanced result in GD and delta measure.

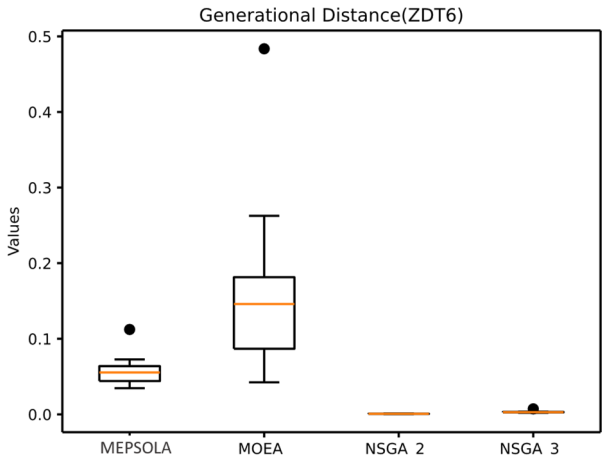


Figure 4.36: Generational Distance Measure Results for ZDT6 Function

4.4.7 MEPSOLA Results Discussion

The comprehensive evaluation of MEPSOLA's performance across diverse mathematical functions confirms its superior standing on various metrics. This underscores its ability in managing the complex MOO problems, as outlined in Table 4.6. A substantial advantage is demonstrated when comparing MEPSOLA to MOEA in a set coverage measure with an average median of 0.7692 for MEPSOLA, highlighting its dominance. Conversely, a lower set coverage measures average median of 0.006 is exhibited by MOEA when compared to MEPSOLA, emphasizing the latter's clear superiority. MEPSOLA's dominance is further established compared to NSGA-2 with a set coverage measure average median of 0.3258, whereas NSGA-2 lags behind at 0.1010. Similarly, when comparing MEPSOLA to NSGA-3, MEPSOLA showcases superior performance with a set coverage measure average median of 0.3226, while NSGA-3 trails with 0.0207.

In the HV metric showcases MEPSOLA's remarkable achievement with an average median of 2542.46, while MOEA, NSGA-2, and NSGA-3 exhibit significantly lower

values of 62.70, 464.20, and 55.86, respectively. This emphasizes MEPSOLA's clear superiority in covering a substantial portion of the Pareto front. Similarly, in NDS metric further underscores MEPSOLA's dominance by recording an average median value of 400.67. In contrast, MOEA, NSGA-2, and NSGA-3 display comparatively lower values of 12.83, 100, and 12.83, respectively. Moreover, in the term of Delta measure and GD metrics, MEPSOLA maintains efficiency with modest values of 1.0771 for Delta measure, and ranked third by obtaining 0.0167 in GD metric. Meanwhile, MOEA, NSGA-2, and NSGA-3 demonstrate lower values across these metrics, with 0.8698, 0.6869, and 0.6420 for DM, and 0.0514, 0.0019, and 0.0039 for GD, respectively.

MEPSOLA's superior performance in MOO is a direct result of its innovative methodology that incorporates a multi-objective-aware criterion. This criterion is specifically designed to address the complexities and trade-offs between conflicting objectives effectively. MEPSOLA well employs a multi-exemplar selection process, which facilitates a concurrent balance between exploration and exploitation, thereby enhancing its convergence capabilities as well the diversity. Furthermore, the incorporation of Tabu search as a conditional local search mechanism within the algorithm for exemplar selection over a predetermined time frame significantly improves search efficiency and solution quality. By periodically updating the search strategy to circumvent previously visited or 'tabooed' solutions, the algorithm is able to escape local optima. This promotes exploration in new and potentially more optimal regions of the solution space. Such strategic application of Tabu search ensures dynamic adaptation to the evolving landscape of the search space, thereby improving the algorithm’s capacity to identify high-quality, diverse solutions across multiple

objectives. Moreover, the employment of a smart initialization strategy enhances the search effectiveness by ensuring that the initial population encompasses a broad and representative range of the solution space. This strategy prevents premature convergence on suboptimal regions, providing a solid foundation for the algorithm to efficiently explore diverse solutions.

Precise experimentation conducted across ten Pareto fronts using various mathematical functions, MEPSOLA's superiority is evidenced by its remarkable performance metrics, notably in SC, HV, and the NDS, along with sensible results in Delta and GD metrics. To explain the results disparity, further investigation of these metrics is required. The HV, quantifies the volume covered by the Pareto front, offering a holistic measure that captures both diversity and convergence [130]. SC complements HV by evaluating how well the Pareto front covers the objective space, ensuring the exploration of diverse solutions [208]. This dual focus on HV and SC provides a nuanced understanding of the algorithm's performance in terms of both spread and representation. NDS, with its focus on diversity, avoidance of clustering, exploration of trade-offs, and robustness provision, emerges as a valuable metric in MOO. It ensures a thorough representation of the Pareto front, offering decision-making flexibility for users and facilitating benchmarking of MOO algorithms [209]. On the other hand, while Delta and GD are valuable measures, but they have limitations in providing a holistic view of the Pareto front. Delta focuses on specific aspects, such as spread relative to a reference set, while GD quantifies the average distance to the true Pareto front, potentially overlooking overall spread [210]. This highlights the need for a more comprehensive evaluation beyond these specific aspects. By considering NDS alongside HV and SC, a more well-rounded assessment of algorithm performance is achieved, encompassing overall quality and diversity.

This balanced approach enhances the understanding of how MOO algorithms navigate various objectives and trade-offs.

While the results obtained from MEPSOLA show the best performance in metrics such as SC, HV, NDS, and close to zero GD the algorithm exhibits a more modest performance in the DM. This modest performance in DM suggests that the solutions might not be uniformly distributed across the Pareto front. High diversity and coverage in the solution set can lead to uneven distributions, where certain regions of the Pareto front become more densely populated than others. This unevenness, while it may indicate comprehensive coverage, results in a lower DM score, reflecting non-uniform solution distribution.

The core challenge lies in balancing these objectives. Enhancing diversity and coverage typically involves mechanisms like mutation or external archive strategies, which can inadvertently compromise the evenness of the solution distribution. Conversely, optimizing for a better Delta Measure may restrict the algorithm's exploratory capabilities, reducing diversity and coverage. This trade-off is a fundamental challenge in MOPSO, requiring finely tuned parameters and adaptive strategies to dynamically balance these competing objectives as the optimization process evolves.

Moreover, the obtained results for MEPSOLA on different mathematical functions, the algorithm demonstrates strong performance on continuous functions such as in (FON, ZDT-1, ZDT-2, and ZDT-6), but modest performance on (KUR and ZDT-3) the disconnected shape functions. Despite these variations in these functions, MEPSOLA achieved the best results in HV, NDS, overall SC, and very close to zero GD.

The cause of these performance differences an attributed to the inherent limitations of MOPSO, particularly its sensitivity to parameter settings [211]. MOPSO requires careful tuning of parameters such as inertia weight, acceleration coefficients to optimize its performance across different types of functions. Without this careful tuning, the algorithm might not fully exploit its potential on more complex or irregular functions, like KUR and ZDT-6's disconnected shape function. Hyperparameters process could improve MEPSOLA's consistency and effectiveness across a broader range of optimization problems [17].

Table 4.6: Overall average of the measures across all functions

Set Coverage Measure		
MEPSOLA vs MOEA	MEPSOLA vs NSGA-2	MEPSOLA vs NSGA-3
0.7692	0.3258	0.3226
MOEA vs MEPSOLA	NSGA-2 vs MEPSOLA	NSGA-3 vs MEPSOLA
0.006	0.101	0.0207

Measure	MEPSOLA	MOEA [133]	NSGA-2 [130]	NSGA-3 [131]
HV	2542.46	62.7	464.2	55.86
NDS	400.67	12.83	100	12.83
Δ	1.0771	0.8698	0.6869	0.6420
GD	0.0167	0.0514	0.0019	0.0039

Further analysis was conducted by comparing MEPSOLA with the method proposed in [161], known as the Simplified Multi-Objective Particle Swarm Optimization (SMPSO) algorithm. This algorithm balances exploration and exploitation while preserving a diverse set of non-dominated solutions. Similar to MEPSOLA, SMPSO implements non-dominated sorting and voting mechanisms to maintain solution diversity and guide the search process. SMPSO aims to simplify the complexity inherent in multi-objective optimization by reducing the number of algorithmic parameters and focusing on core PSO principles to navigate the solution space effectively. These similarities indicate a shared methodological foundation in how the

algorithms structure their optimization process, particularly in their approach to managing and preserving solution diversity.

However, MEPSOLA distinguishes itself by integrating more advanced features, making it superior in tackling high-dimensional and complex optimization problems. Unlike SMPSO, which relies on more traditional and less adaptive methods, MEPSOLA introduces a dynamic multi-exemplar selection process that adjusts based on the optimization landscape. This allows MEPSOLA to explore and exploit the solution space more effectively, ensuring a more thorough search for optimal solutions. Additionally, the incorporation of a conditional Tabu search in MEPSOLA further enhances its ability to escape local optima, providing a significant advantage in scenarios where the search space is complex and fraught with local traps. These advanced mechanisms contribute to MEPSOLA's ability to maintain robust performance across a wide range of benchmark functions.

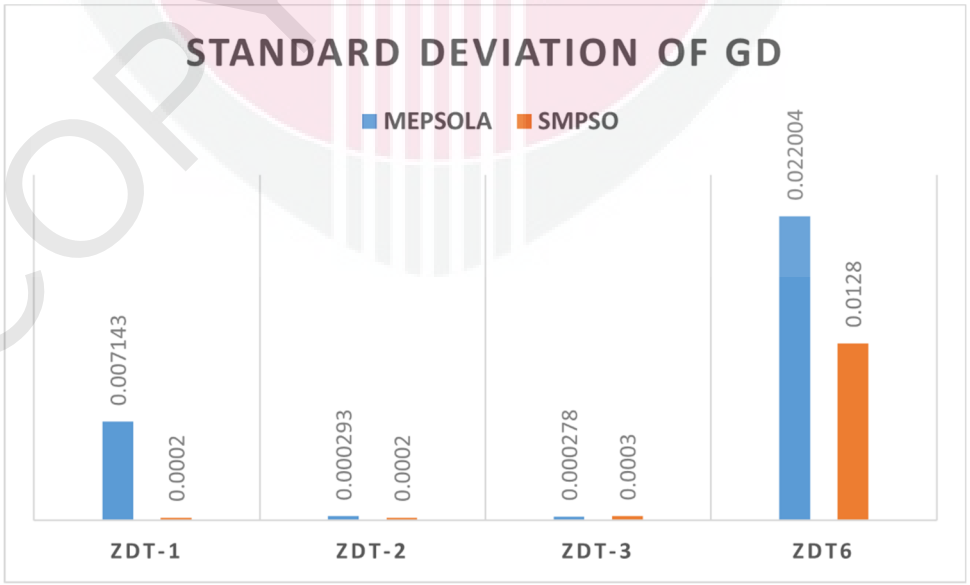


Figure 4.37: Standard Deviation Generational Distance for MEPSOLA against SMPSO [161] based on four Mathematical Functions

In Figure 4.37, a comparison of the standard deviation for GD between MEPSOLA and SMPSO is presented, showing that both algorithms achieved very close SD values across the ZDT-1, ZDT-2, ZDT-3, and ZDT-6 benchmark functions, with slight superiority for SMPSO. This indicates that while both algorithms maintain a consistent solution and very close to zero, MEPSOLA's enhancements make it particularly well-suited for more challenging optimization tasks, offering a slight edge in maintaining diversity and avoiding premature convergence. This superiority is evident in various measures such as SC, HV, and NDS.

MEPSOLA's performance across various mathematical functions demonstrates its robust capability in tackling complex optimization problems, such as those in managing trade-offs between different objectives, such as memory usage and accuracy in feature selection for IDS. The algorithm's ability to outperform traditional approaches like MOEA, NSGA-2, and NSGA-3 in metrics such as Set coverage (SC), Hypervolume (HV), Number of Dominated Solutions (NDS), and achieving near-zero Generational Distance (GD) while maintaining a decent performance in Delta Measure (DM) highlights its effectiveness in producing diverse, high-quality solutions. This underscores MEPSOLA's superior management of the exploration-exploitation balance, making it a valuable tool in the field of multi-objective optimization.

These findings have significant implications for real-world applications, particularly in dynamic and non-stationary environments like IDS. MEPSOLA's demonstrated adaptability and robustness in handling evolving threats and data streams position it as an excellent candidate for real-time decision-making and feature selection in live data streams. With some modifications to further enhance its capabilities for real-time

processing, MEPSOLA could bridge critical gaps in current methodologies, offering a more resilient, responsive, and effective approach to optimization in complex, high-dimensional environments. This makes it not only a powerful theoretical contribution but also a practical solution for pressing challenges in real-world applications.

4.5 DFA-GPE Results

This section presents a comprehensive analysis of the results for the proposed IDS named DFA-GPE and compared with three other feature selection methods: Random Forest (RF) [206], Logistic Regression (LR) [122], Recursive Feature Elimination (RFE) [123], and Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [50]. The evaluation was performed on two of the recent real datasets, HIKARI2021 and TON_IoT 2020, spanning two distinct domains. The evaluation aspects included binary classification, multi-attack classification, memory reduction, feature count, and a concluding discussion on the overall results.

4.5.1 Binary Classification Results

This subsection is dedicated to the exploration of binary classification performance, with a focus on the proposed DFA-GPE method in comparison to other established FS methods. Binary classification, a critical task in many machine learning applications, necessitates accurate differentiation between two classes by the predictive model. The ensuing analysis will investigate the efficacy of each method within this binary classification task, taking into account the complexities and challenges presented by real-world data dynamics.

4.5.1.1 HIKARI 2021 Dataset Result

This subsection represents the performance outcomes of the DFA-GPE method on the HIKARI dataset, benchmarking it against leading feature selection techniques like RF, LR, RFE, and NSGA-II. The evaluation employs crucial classification metrics, including accuracy, recall, precision, and F1-score, to offer a comprehensive assessment of each method's effectiveness.

1. Accuracy

The accuracy results, reflecting the algorithm's capability to accurately classify both positive and negative instances. The results in the Figure 4.38, reveals that the DFA-GPE method consistently outperformed others. It achieved an initial accuracy of 0.99, peaked at an impressive 0.999, and then stabilized at 0.991, demonstrating its exceptional performance throughout. This was followed by NSGA-II, which started with flawless accuracy but then declined to register 0.986, then by the end of the period stabilized to register 0.987. The third method RF, began with also flawless accuracy but exhibited minor variations, where it declined to register 0.984, then stabilized to register 0.987 by the end. Both LR and RFE initially exhibited satisfactory accuracy levels. However, their accuracies slightly decreased over time to register 0.983 and 0.978, respectively, and stabilized at the end to register 0.985 and 0.984, respectively.

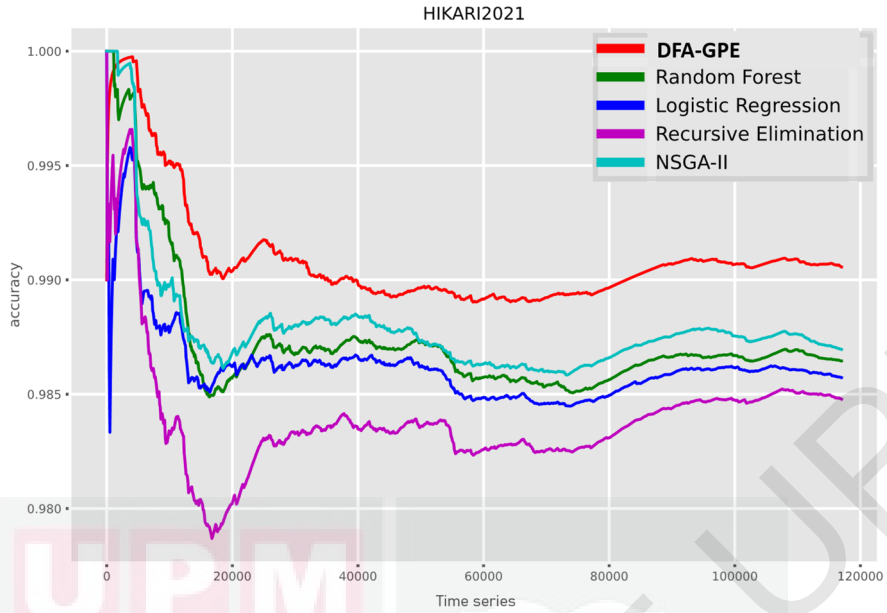


Figure 4.38: Accuracy metric results of method DFA-GPE compare with other methods RF, LR, RFE and NSGA-II on HIKARI 2021 dataset

2. Recall, F1-Score, and Precision

The recall metric results, assessing a framework ability to accurately classify all relevant instances, showcased the DFA-GPE method's exemplary performance. As depicted in Figure 4.39, the DFA-GPE method's recall value started at 0.992 and abruptly rose to record 1.000. It then consistently exhibited superiority over the other methods throughout the entire experiment, stabilizing at the end to record 0.99. The other methods, RF, LR, RFE, and NSGA-II, started with satisfactory recall values but, in a short period of time, exhibited in different time series a decline in their values to register 0.981, 0.978, 0.973, and 0.979, respectively. At the end of the experiment, all methods stabilized at recall values of 0.986, 0.985, 0.984, and 0.987, respectively.

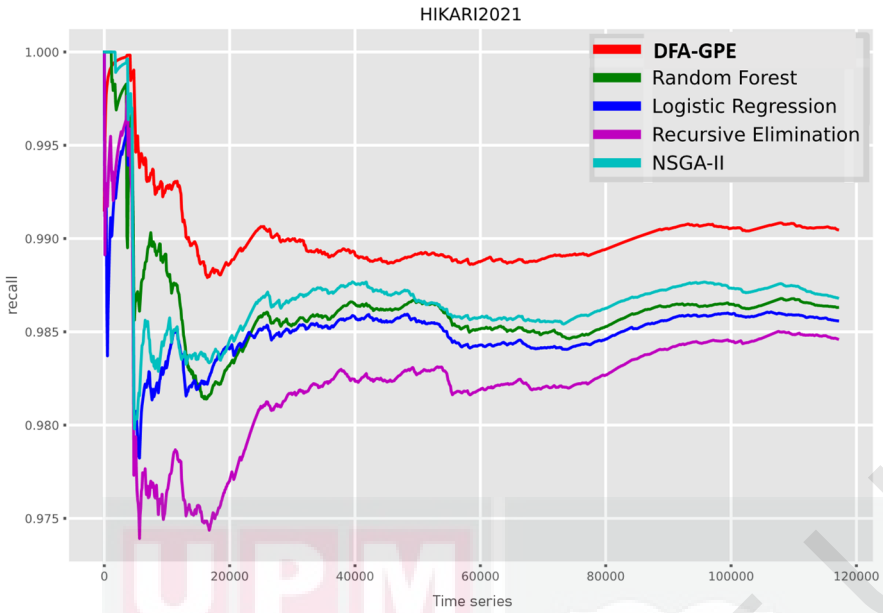


Figure 4.39: Recall metric results of method DFA-GPE compare with other methods RF, LR, RFE and NSGA-II HIKARI 2021 dataset

Similarly, the F1-score metric, which is the harmonic mean of precision and recall, is illustrated in Figure 4.40. It reveals that DFA-GPE initiated with a score of 0.989 and quickly escalated to record 1. Subsequently, the proposed method maintained its dominance over the competing techniques throughout the duration of the time series, eventually stabilizing at a score of 0.991 towards the ending. Conversely, RF, LR, RFE, and NSGA-II began with notable F1-scores but experienced a decrease in their scores shortly recording values of 0.983, 0.983, 0.976, and 0.985, respectively. It's worthy to mention that the methods stabilized at the end of the experiment, recording 0.986, 0.985, 0.984, and 0.987, respectively.

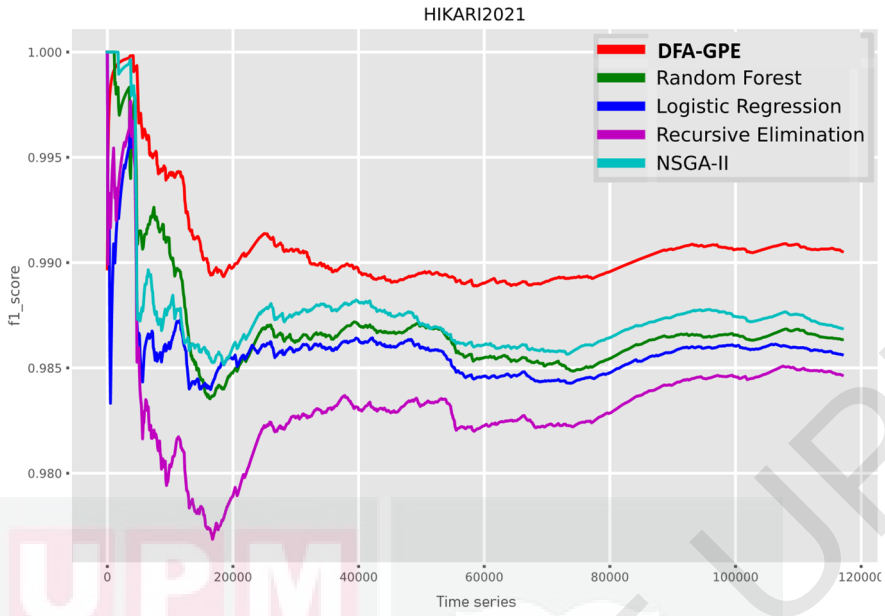


Figure 4.40: F1_score metric results of method DFA-GPE compare with other methods RF, LR, RFE and NSGA-II on HIKARI 2021 dataset

The precision metric, which assesses the accuracy of positive predictions by the model, shows exemplary performance by the DFA-GPE method. As detailed in Figure 4.41, it started with a precision of 0.988, rapidly escalated to 0.999, marking its peak, and then settled at 0.990 by the end of the experiment. This demonstrates a significant dominance over other methods throughout the duration of the experiment. Conversely, when considering the precision, RF, LR, RFE, and NSGA-II began with optimal values. However, their advantage was short-lived due to a decline in their scores, with LR registering 0.983 and RFE 0.980 at the onset of the study. Meanwhile, RF and NSGA-II saw some reduction between time 60000 and 80000 in their scores to record 0.985 and 0.986, respectively. Subsequently, all methods reached a point of stability towards the end of the experiment, recording 0.986 for RF, 0.985 for LR, 0.985 for RFE, and 0.987 for NSGA-II.

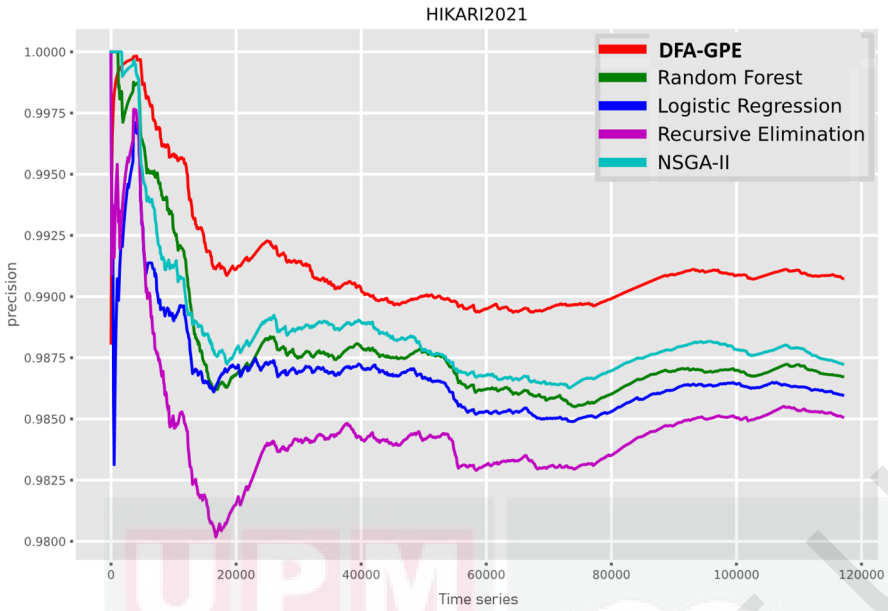


Figure 4.41: Precision metric results of method DFA-GPE compare with other methods RF, LR, RFE and NSGA-II on HIKARI 2021 dataset

4.5.1.2 TON_IoT 2020 Dataset Result

This subsection evaluates the performance of the DFA-GPE method applied to the TON_IoT dataset, aiming to gauge its effectiveness within the IoT domain. The results are compared with other FS benchmark methods, and a comprehensive assessment is conducted using key metrics such as accuracy, recall, precision, and F1-score to offer an in-depth analysis of the performance.

1. Accuracy

The accuracy metric results, as depicted in Figure 4.42, showcases the superiority of the DFA-GPE approach throughout the experiment phase. Initially, DFA-GPE began with a notable accuracy rate of 0.933, consistently outperforming the other approaches recording the highest accuracy scores throughout the entire experiment. The peak of accuracy reached was 0.933, with the minimum dropping to 0.923. Towards the end

of the experiment, the accuracy metric settled at 0.926, highlighting the method's consistent and dependable performance relative to competing methodologies. In comparison to other feature selection techniques the comparative analysis arranged based on the efficacy of each method, the NSGA-II method, ranking second, started with an accuracy of 0.926, ascended to a peak of 0.932, and eventually stabilized at the end with 0.922. Followed by RF, the RF feature selection algorithm initiated at an accuracy of 0.900, peaked to register 0.928, and stabilized at the end with 0.920. Conversely, the LR method demonstrated an initial accuracy of 0.91, observed a slight decrement to 0.897 shortly thereafter, and stabilized at 0.908, with its highest accuracy recorded at 0.931. Lastly, the RFE method began with an accuracy of 0.913, reached a maximum of 0.932, and stabilized at 0.913.

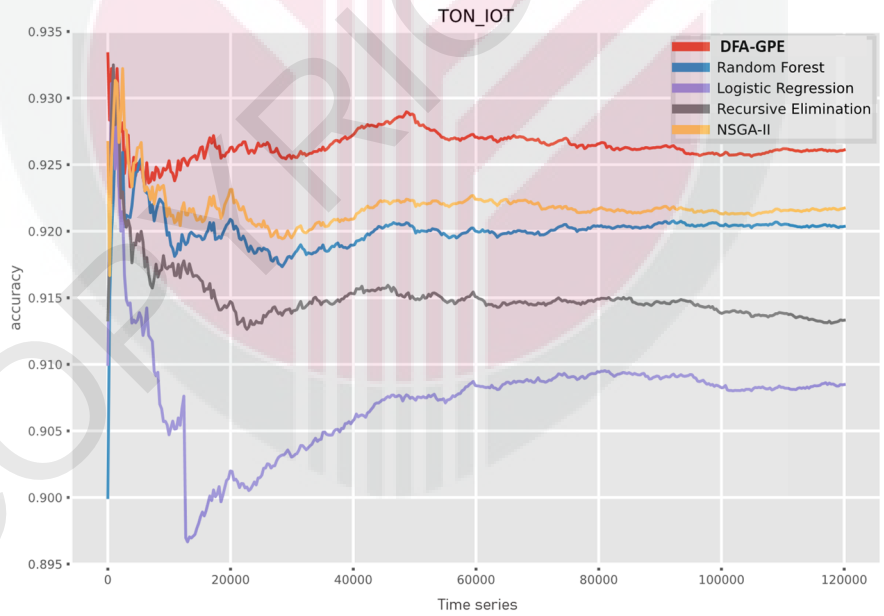


Figure 4.42: Accuracy metric results of method DFA-GPE compare with other methods RF, LR, RFE and NSGA-II on TON_IoT 2020 dataset

2. Recall, F1-Score, and Precision

The recall metric results for the TON_IoT dataset, as illustrated in Figure 4.43, the proposed method outperforms competing methods by achieving the highest recall values throughout the entire duration of the experiment stages. Initially, DFA-GPE method registers a remarkable recall started by 0.934, which is also recorded as the highest value. Subsequently, it maintains a stable performance, with a final recall value of 0.927, highlighting the reliability and consistency of the proposed method. Notably, the lowest recall value recorded by DFA-GPE was 0.925, further demonstrating its steady and steady outcomes. In comparison, the NSGA-II as a second method begins at a recall value of 0.926 and reaches its highest recall of 0.933; the minimum noted value was 0.917, before finalizing at 0.923. The RF method starts at a recall of 0.900, achieves a peak of 0.930, and the minimum noted value was 0.9 then it eventually settles at 0.9213. For RFE started with a recall of 0.910, which is also scored as the minimum value, the RFE method peaks at 0.932 and then finds stability at 0.914. Lastly, LR method starts with 0.91 then underwent to a notable decrease in recall to 0.896 at the experiment's start but recovers to stabilize at the end recording 0.909, it's noteworthy that the peak point for LR was observed at a recall value of 0.931.

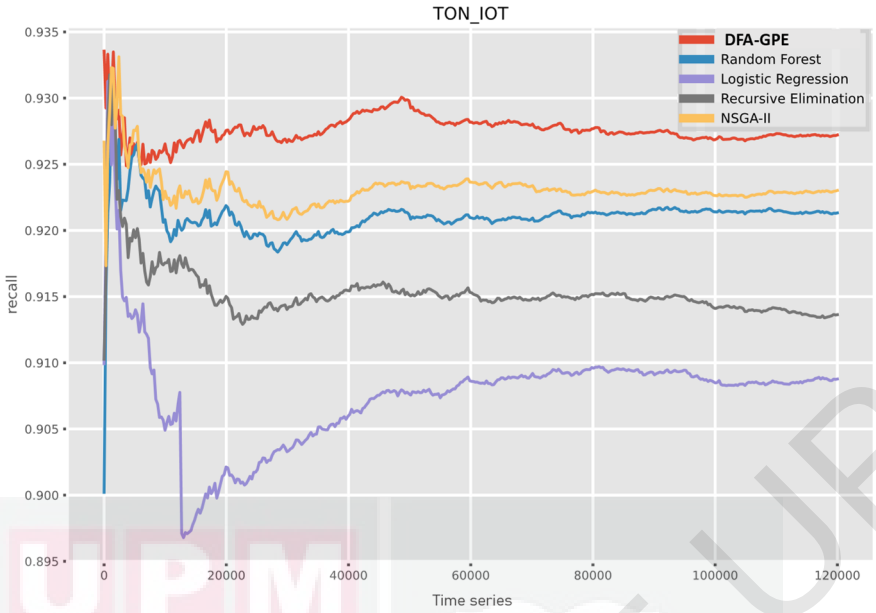


Figure 4.43: Recall metric results of method DFA-GPE compare with other FS methods RF, LR, RFE and NSGA-II on TON_IoT 2020 dataset

In the results of F1-scores for TON_IoT 2020, the superiority of DFA-GPE method was remarkable throughout the entire experiment time series, as depicted in Figure 4.44, the DFA-GPE method initiates with an F1-score of 0.932 and climbs in the beginning to record a peak of 0.933 before settling at 0.928. Notably, the lowest F1-score observed for our method is 0.925. Among the comparative methods, NSGA-II emerges as a significant competitor, initiating at an F1-score of 0.926, peaking at 0.934, recording a minimum of 0.918, and eventually ending at 0.923. The RF method, in close pursuit of NSGA-II, starts at 0.900, escalates to a maximum of 0.930, decreased to a minimum of 0.900, and settles in the end at 0.922. The RFE method, positioned next, begins at 0.912, peaks at 0.933, record a minimum of 0.912, and settles at 0.914. Lastly, the LR method, ranked fifth, start with an F1-score of 0.912, reaches a high of 0.933, drops to a low of 0.898, and finalizes at 0.910.

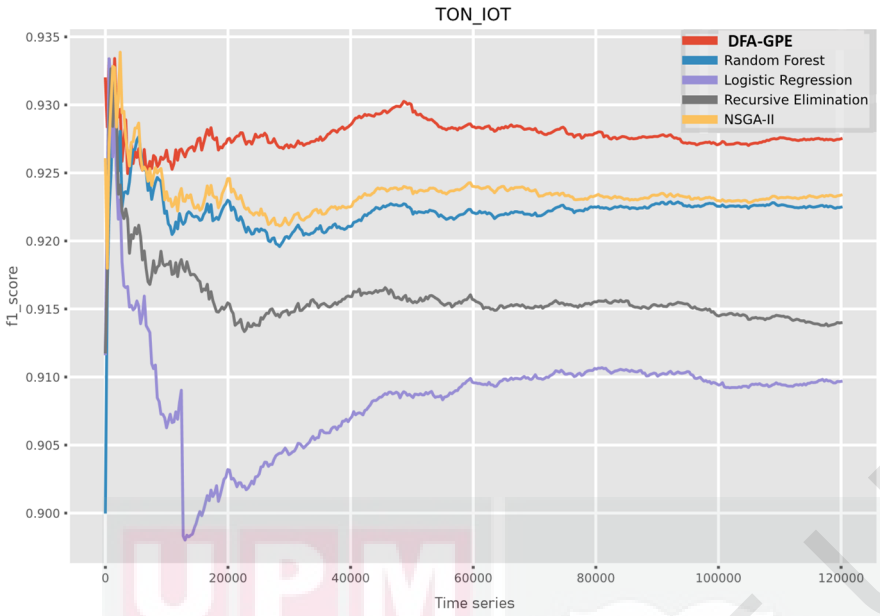


Figure 4.44: F1-score metric results of method DFA-GPE compare with other methods RF, LR, RFE and NSGA-II on TON_IoT 2020 dataset

In the precision metric results, illustrated in Figure 4.45, our method appears as the leading competitor. It consistently demonstrates the highest precision rates at various stages throughout the experiment. This performance is initiated with a precision of 0.935, reaching a high of 0.937, with the lowest value being 0.926, and concluding with a steady precision of 0.928 at the end of experiment. In terms of other methods, RF ranks closely behind as the second place in precision. It starts at a precision of 0.913, peaks at 0.932, dips to a minimum of 0.913, and finally stabilized at 0.925. Next, the third rank in precision is the NSGA-II method, beginning at 0.930 in precision, climbing to a high of 0.936, dropping to a low of 0.922, and concluding with a consistent precision of 0.925. RFE is placed fourth, with an initial precision of 0.916, reaching a high of 0.935, a low of 0.914, and at the end achieving a stable precision of 0.915. Lastly, LR method starts with a precision of 0.917, reaches the highest precision of 0.938, and ends with a consistent precision of 0.911, with the lowest precision recorded being 0.900.

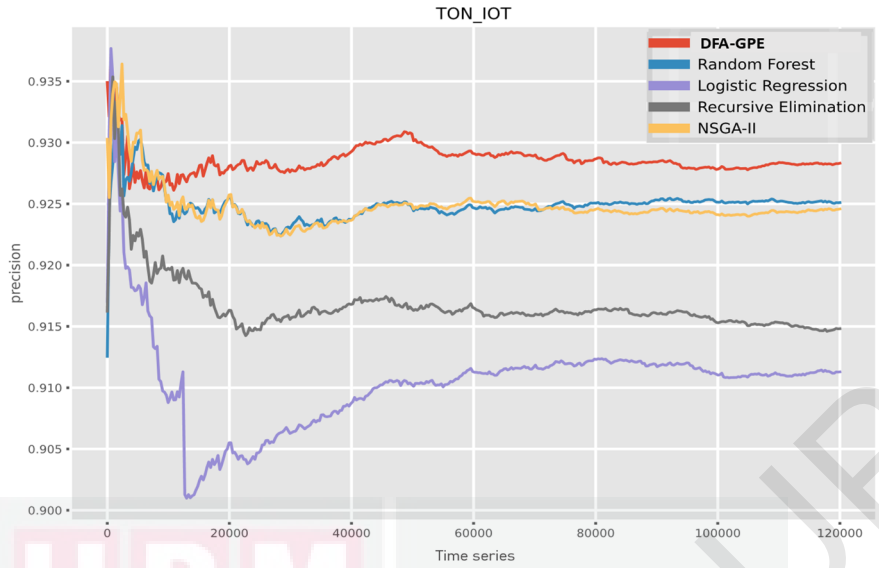


Figure 4.45: Precision metric results of DFA-GPE method compare with other FS methods RF, LR, RFE and NSGA-II on TON_IoT 2020 dataset

The sharp fluctuations observed in the metrics for all methods, including DFA-GPE, especially in Figures 4.38, 4.41, 4.42, and 4.45, particularly at the beginning of the time series, it's an attributed to the initial adaptation phase of the algorithms to the evolving data distribution. At the beginning of a time series, especially in diverse and complex datasets like HIKARI 2021 and TON_IoT 2020, the methods are exposed to a fresh chunk of data, the model may be working with a very limited amount of information from the initial chunks, which can lead to instability in its performance metrics. As the model processes more chunks, it accumulates more knowledge and refines its understanding of the data distribution, leading to more stable and accurate predictions [25]. For DFA-GPE, which is designed to adapt to feature drift and concept drift dynamically, this initial phase is crucial. The algorithm may require several iterations to stabilize its feature selection and ensemble components to align with the underlying data patterns. These fluctuations reflect the model's learning process, as it fine-tunes its knowledge in response to the new data, eventually leading to more stable and accurate predictions as the time series progresses this evident from the performance at the end of the experiment.

Additionally, this phenomenon can be considered a limitation of the incremental learning approach employed by DFA-GPE and other algorithms. While they are designed to adapt to new data without retraining from scratch, this adaptability comes with the trade-off of initial instability, where the method performance stabilizes as it accumulates more chunks and refines its knowledge incrementally. This is particularly noticeable in complex datasets where the data characteristics can change rapidly, requiring the model to undergo significant adjustments [34].

4.5.2 Multi Attacks Classification Results

This subsection presents the results of implementing the DFA-GPE method for detecting multiple attacks across both datasets in a multi-class detection scenario. Starting with the HIKARI dataset, the results underscore the superior capability of the DFA-GPE approach in accurately classifying various attack types within a data streaming context. The confusion matrix depicted in Figure 4.46 illustrates the DFA-GPE method's classification effectiveness across multiple attack types. The impressive accuracy rate, notably 97.33% for the benign class, underscores DFA-GPE method's adeptness at accurately classifying instances of normal network behavior. The matrix also showcases the true positive rates, evident in its diagonal entries, which correspond to the accurately classified instances per class. Specifically, the true positive rates are 99.86% for the bruteforce real value class and 99.95% for the bruteforce-XML class, illustrating DFA-GPE method's precision in classifying distinct attack types, thereby bolstering the overall performance of the DFA-GPE strategy.

Additionally, the method exhibits minimal error rates, with only 0.14% for the bruteforce real value attack (by combining 0.0007 that classified wrongly as benign and 0.0007 classified wrongly as bruteforce-XML) and 0.05% for the bruteforce-XML

attack (by combining 0.0003 that classified wrongly as benign and 0.0002 classified wrongly as bruteforce), alongside a 2.67% error rate for benign traffic (combining 0.0156 that classified wrongly as bruteforce and 0.0111 classified wrongly as bruteforce-XML). These remarkably low error rates testify to the DFA-GPE method's accuracy in distinguishing between different attack vectors, establishing it as a potent instrument for contemporary network security challenges.

The exceptional true positive rates and minimal error margins achieved underscore the method's efficacy and dependability in navigating the intricate landscape of modern cyber threats, showcasing its utility in preemptive intrusion detection and countermeasure implementation.

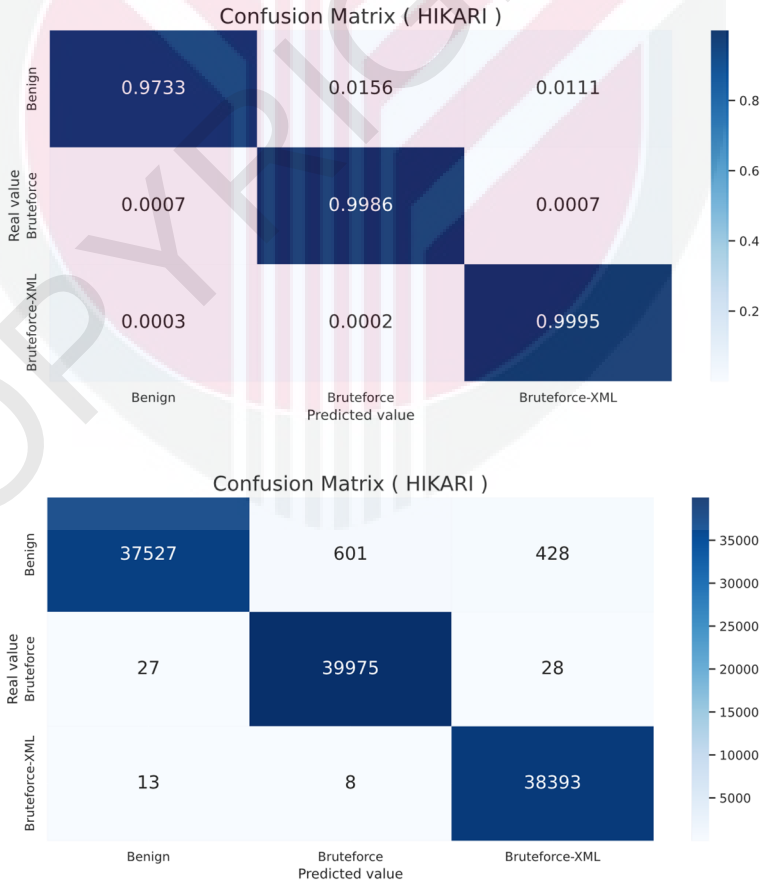


Figure 4.46: Confusion Matrix Analysis for DFA-GPE Method for Multi-Type Attack in HIKARI dataset

Additional analysis of the DFA-GPE method applied to the TON_IoT dataset is shown in Figure 4.47. The confusion matrix spots the light on DFA-GPE method ability to accurately identify various attack types (multi class classification). The matrix's diagonal elements highlight the method's effectiveness in correctly determining each category of attack. Notably, the method recorded creditable accuracy rates: 91% for injection attacks, 90% for DDoS attacks, 94.1% for DoS attacks, and 95.7% for scanning attacks. Furthermore, it adeptly identified benign or normal traffic with an accuracy of 92.8%. The low values off the diagonal suggest few misclassifications among the different attack types and normal traffic, showcasing DFA-GPE's strength in navigating the complexities of various attack patterns and adjusting to the concept and feature drifts encountered within the TON_IoT dataset. These findings underscore DFA-GPE method's proficiency in delivering precise and adaptable DFS, establishing it as an effective solution for real-world IoT scenarios where data streams are dynamic, necessitating efficient and dependable attack detection mechanisms.

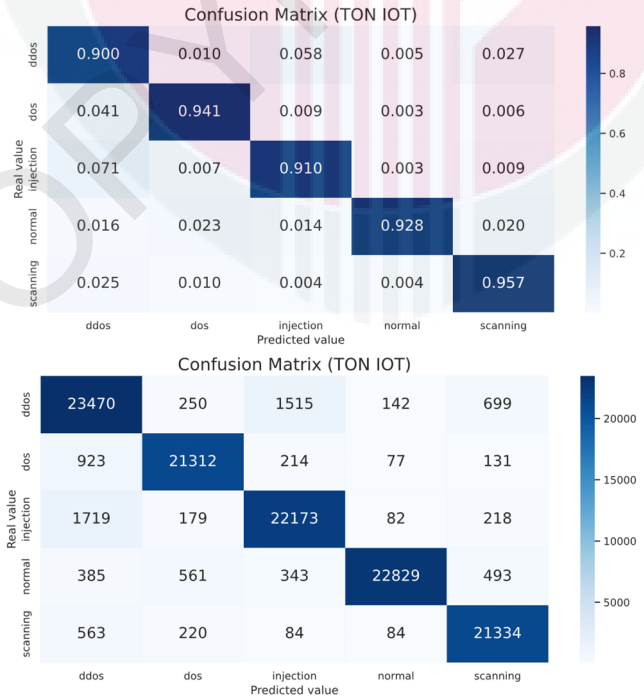


Figure 4.47: Confusion Matrix Analysis for the DFA-GPE Approach for Multi-Type Attack Detection in TON_IoT dataset

4.5.3 Memory Reduction and Feature Counts Optimization

This subsection evaluates the DFA-GPE method's effectiveness in two key aspects: memory reduction and feature count analysis.

4.5.3.1 Memory Reduction Results

The results of memory reduction show the efficiency of the proposed method in minimizing memory usage in data stream processing for both datasets, starting with the HIKARI dataset as illustrated in Figure 4.48. This figure represents the outcomes of implementing DFA-GPE approach on the HIKARI dataset in a streaming context (chunks). We observed variability in the memory reduction values, with memory usage fluctuating between two fixed points: 4,132 bytes and 32,132 bytes, while the average memory utilization settled at 17,713 bytes. In this context, this represents a significant improvement over the baseline memory usage prior to applying the DFS method, which was considerably higher at 272,132 bytes for each of the 400 chunks. The total memory reduction through the entire streaming was 99,382 KB from total 106,301 KB. This comparison highlights the substantial memory savings achieved through DFA-GPE method, underscoring its effectiveness in streamlining data processing in real-time environments.

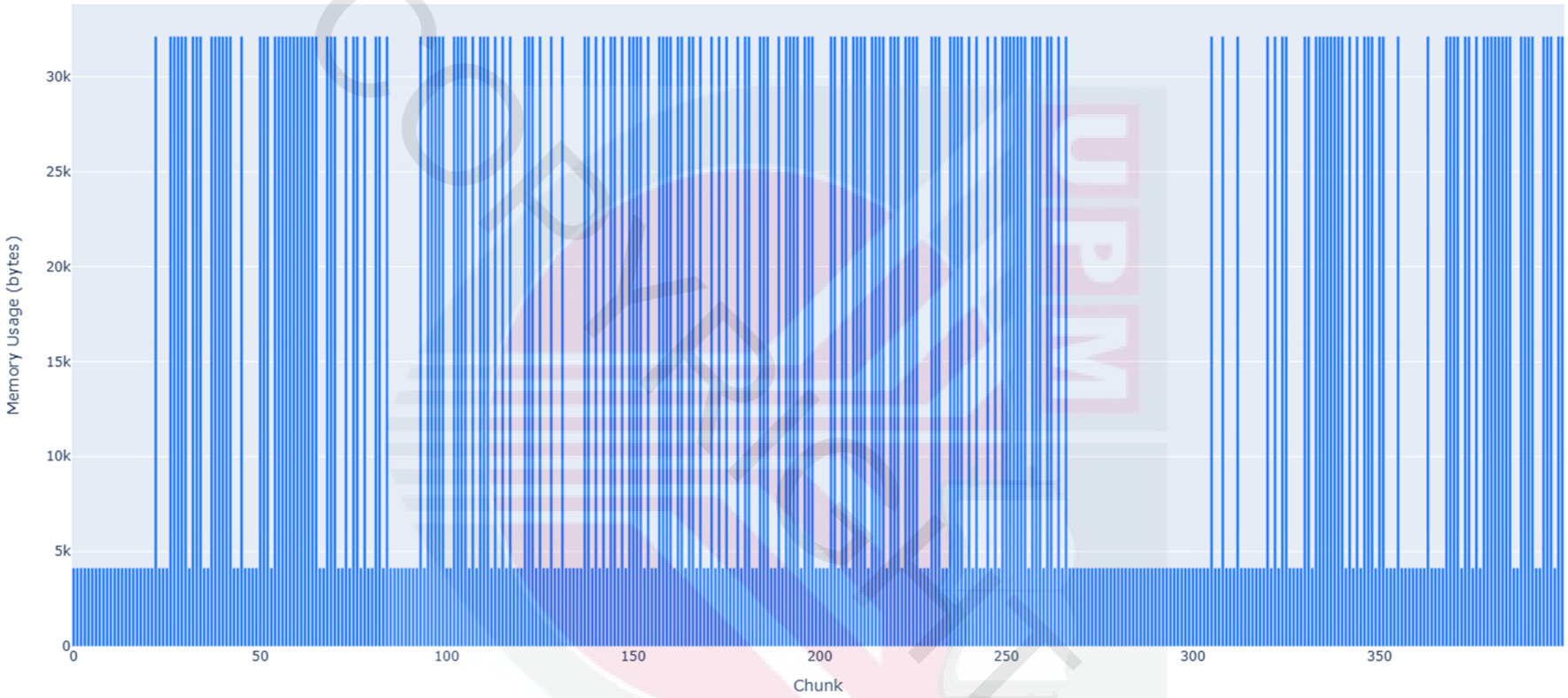


Figure 4.48: Memory Usage for Each Chunk in HIKARI Dataset for the Proposed Feature Selection Method from A Total Usage of 272,132 Bytes per Chunk. Note, the Blue Columns Indicates Memory Usage

Furthermore, the Figure 4.49 provides an evaluation of the memory reduction achieved through the deployment of our method on the TON_IoT dataset. This analysis reveals a broader range of fluctuations in memory usage shape unlike the observations made with the HIKARI dataset. Before implementing of our DFS strategy the initial memory usage for each data chunk was recorded at 168,132 bytes, after implementation of DFA-GPE, the range of memory usage ranged from a minimum of 12,132 bytes to a maximum of 112,132 bytes, with the average memory usage stood at 62,332 bytes. Furthermore, a notable concentration of memory reduction figures, ranging from 55,000 to 67,000 bytes, was observed. The total memory reduction throughout the streaming amounted to 41,328 KB from total 65,676 KB, underscoring the consistent memory efficiency and superior classification accuracy offered by DFA-GPE.

In IDS, the aspect of memory efficiency is essential, especially within environments with limited resources. Employing DFS supported by MOO, which leverages constraints on memory for the sake of accuracy enhancement and incorporates adaptive learning, showcases the potential in minimizing memory consumption within IDS contexts. By continually selecting relevant features for each group of instances, DFS enhances memory usage and improves scalability, allowing the IDS to handle huge quantity of data streams efficiently. The MOO guarantees a balancing between memory saving and improving detection accuracy, leading to enhanced performance. Reduced memory overhead results in resource conservation, making DFA-GPE particularly valuable for scenarios with limited computational resources. DFA-GPE contributes to robust and adaptable IDS with memory efficiency, addressing the critical need for resource optimization in intrusion detection scenarios. Furthermore, comparing our approach to other existing methods, DFA-GPE achieves the highest

accuracy among them. This significant improvement in accuracy further emphasizes the superiority of the proposed method in effectively reducing memory usage while maintaining outstanding classification performance.

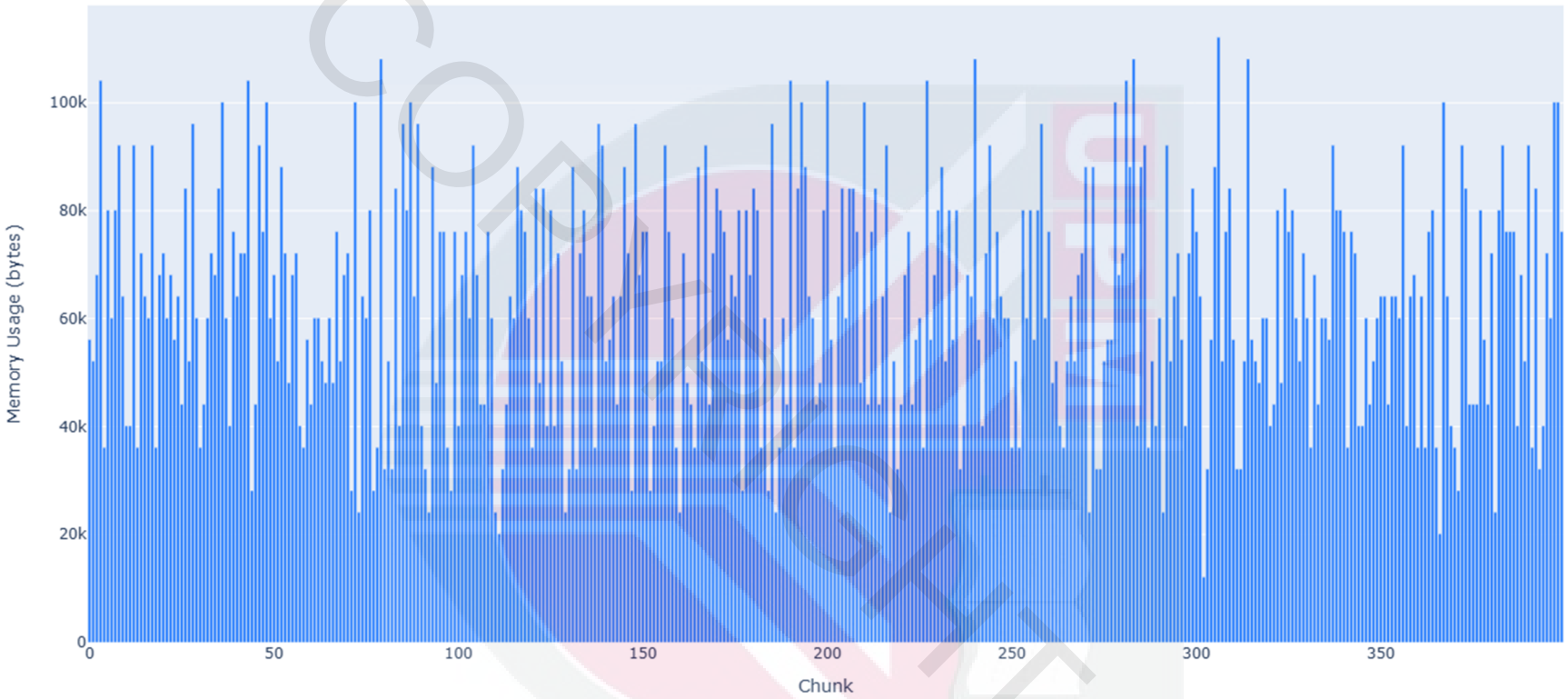


Figure 4.49: Memory Usage for Each Chunk in TON_IoT Dataset of the Proposed Feature Selection Method from A Total Usage of 168,132 Bytes per Chunk. Note, The Blue Columns Indicates Memory Usage

4.5.3.2 Feature Count Optimization Results

This subsection represents the results of the features count utilized in both datasets. First, the HIKARI dataset results, as shown in Figure 4.50 and Table 4.7, the DFA-GPE method is highlighted for its exceptional memory utilization, demanding only 17.298 kilobytes. This positions DFA-GPE as an optimal solution for scenarios where resources are limited, contributing to a perfect reduction in the number of features used from 67 to just 5 features. This reduction translates into a memory saving about 93.49%. The ranking of other methods based on their feature count consumption is as follows, NSGA-II leads with 3 features, followed by DFA-GPE method. RFE is in third place with 8 features, LR comes next with 22 features, and RF has the highest with 25 features. NSGA-II's preference for a minimal feature set is due to its Pareto front's limited diversity, favoring memory efficiency possibly at the expense of classification accuracy. In contrast, DFA-GPE method strikes an optimal balance between minimizing feature count and maintaining high classification performance. Remarkably, it stands out by scoring the highest across essential classification metrics for accuracy, recall, F1-score, and precision, making it a leading choice for effective feature selection.

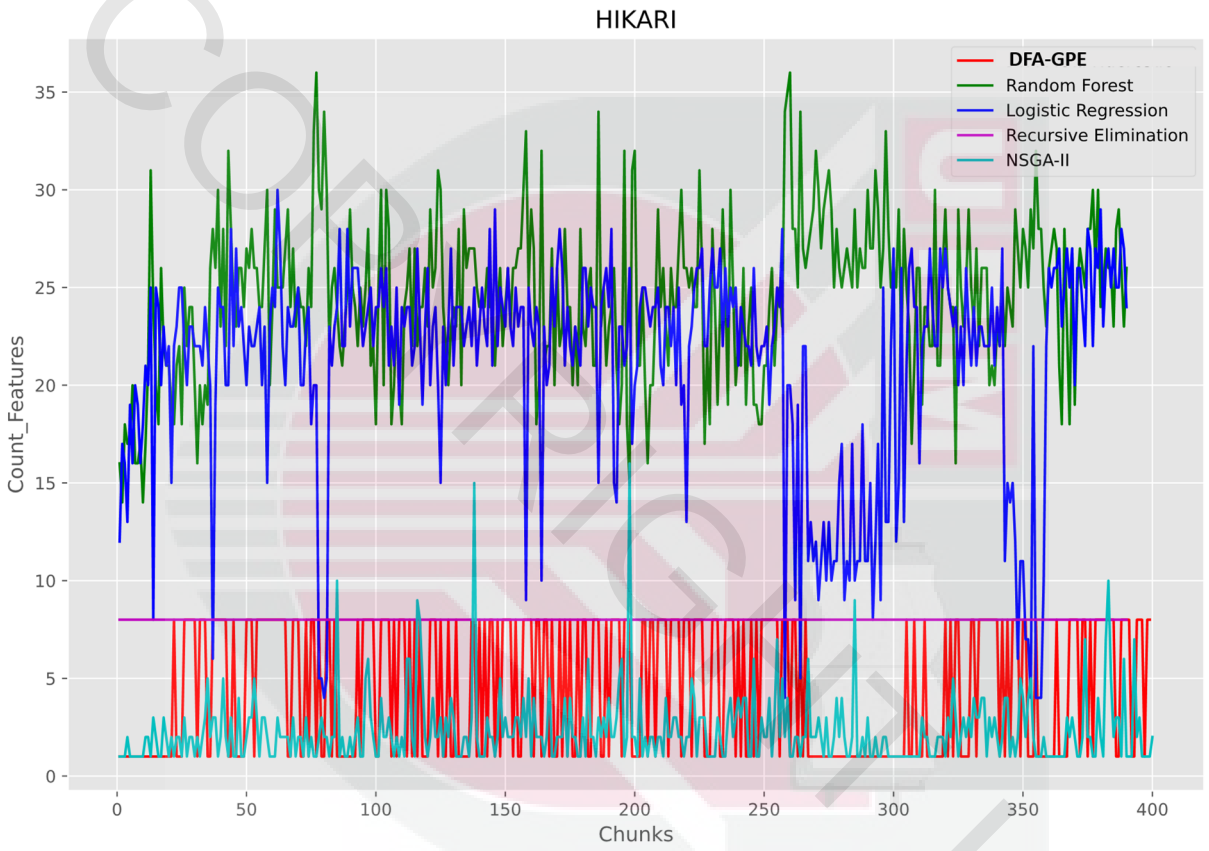


Figure 4.50: Feature Counts for HIKARI Dataset

Moreover, the feature count results for TON_IoT dataset, as shown in Figure 4.51 and Table 4.7, our approach employs only 60.871 kilobytes of memory while reducing the number of features from 42 to 16, achieving a memory saving of 62.93%. Despite this reduction positioning it slightly behind other methods in memory efficiency for the TON_IoT dataset, the slight increase in memory usage is well justified by its superior performance across all evaluation metrics. This is due to the DFS-based MOO's effectiveness in striking a balance between both memory usage and accuracy objectives. This highlights the DFA-GPE method's capability to strike a strategic balance between classifier accuracy and memory usage. The ranking of the other techniques is as follows, NSGA-II leads with the use of 3 features, followed by RFE in second place with 8 features, LR in third with 9 features, and RF in fourth, utilizing 14 features.

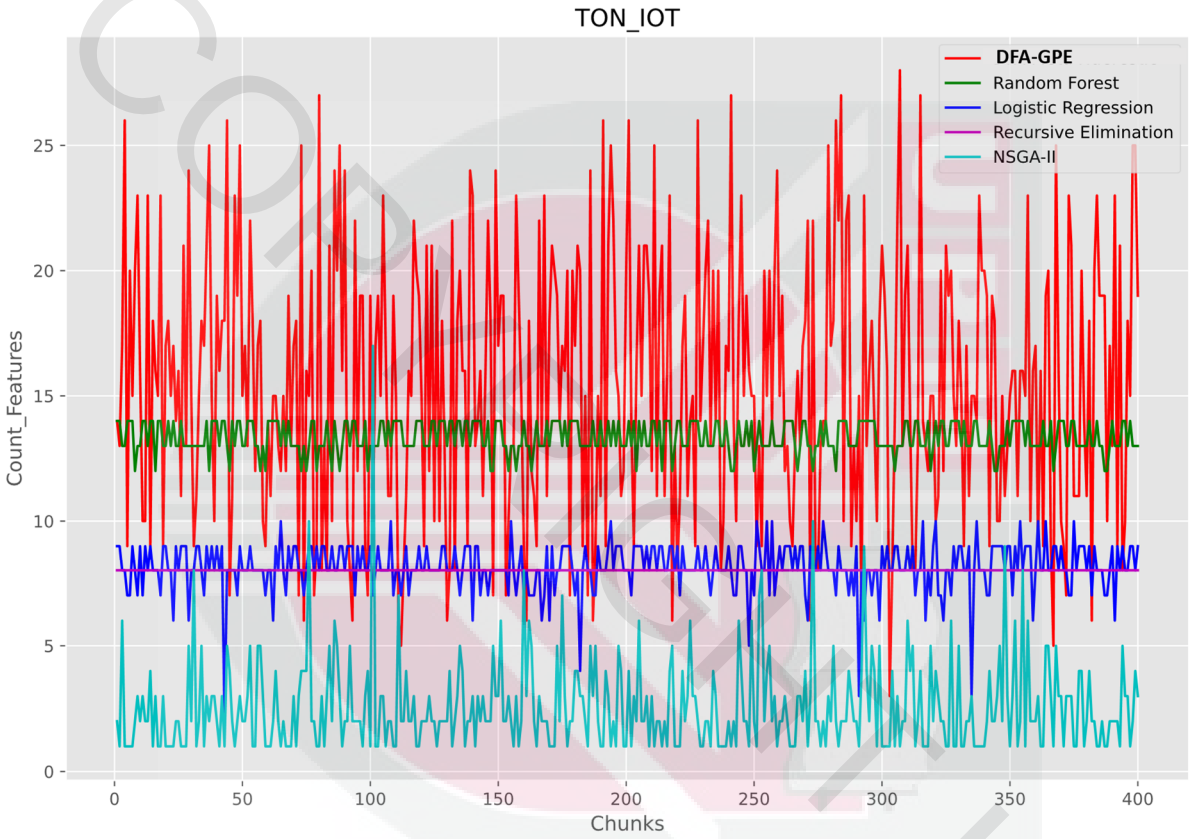


Figure 4.51: Feature Counts for TON_IoT Dataset

Table 4.7: Feature counts results in both datasets

	Feature Count in the HIKARI Dataset					
	Without FS	**DFA-GPE**	**RF [206]**	**LR [122]**	**RFE [123]**	**NSGA-II [50]**
Memory Usage (kilobytes)	265.754	17.298	95.786	82.173	31.378	9.291
Count of Features	67	5	25	22	8	3
	Feature Count in the TON_IoT Dataset					
	Without FS	**DFA-GPE**	**RF**	**LR**	**RFE**	**NSGA-II**
Memory Usage (kilobytes)	164.191	60.871	52.228	31.935	31.378	10.148
Count of Features	42	16	14	9	8	3

The feature reduction merits of DFA-GPE can probably be distilled from the experiments using the HIKARI dataset, as seen in Figure 4.50, where DFA-GPE achieved the second-best performance in feature reduction. The HIKARI dataset, as discussed in the article [212] is designed to reflect realistic network traffic with both benign and malicious data. It has a moderate level of complexity, with the possibility of reducing feature size significantly while maintaining high classification accuracy. On the other hand, the TON_IoT dataset [80], is characterized by its representation of IoT, IIoT, and operating systems environments, incorporating various attack types and telemetry data, which adds layers of complexity. This dataset includes heterogeneous data sources, such as network traffic and sensor telemetry, making it more challenging for feature selection algorithms. The complexity of the TON_IoT dataset stems from its diverse attack scenarios and the intricate interplay between different data types.

This discrepancy in performance between the datasets can be attributed to the inherent complexity and diversity of the TON_IoT dataset compared to HIKARI. The TON_IoT dataset's higher complexity likely requires more features to accurately capture the underlying patterns and interactions within the data. This can explain why DFA-GPE selected a larger number of features for this dataset, as reducing features too aggressively might have compromised the classification performance. The DFA-

GPE method aims to balance feature reduction with maintaining high classification accuracy. While it excelled in reducing features for the HIKARI dataset, the same approach led to a higher feature count for the TON_IoT dataset due to its complexity. This indicates that the algorithm is adapting to the needs of each dataset, prioritizing accuracy over aggressive feature reduction when necessary. Moreover, the voting-based feature selection approach in DFA-GPE ensures a balanced trade-off between feature reduction and accuracy. In complex datasets like TON_IoT, the algorithm may favor retaining more features to avoid overfitting or underfitting, leveraging the diversity and richness of the data to achieve robust classification results. Despite having the highest feature count, DFA-GPE likely achieved superior classification metrics through its ensemble approach, as highlighted in the discussion on the benefits of the voting-based FS algorithm.

4.5.4 DFA-GPE Results Discussion

The overall results discussion of the DFA-GPE method, compare to other state-of-the-art feature selection methods in HIKARI dataset as outlined in Table 4.8. DFA-GPE distinguishes itself by achieving the highest accuracy value with an average of 99.09%, surpassing the average accuracy of 98.79% achieved by NSGA-II. The third position is occupied by RF with an average accuracy of 98.73%, followed by LR with an accuracy of 98.62%, and RFE last recording the lowest accuracy of 98.39%.

In terms of precision, DFA-GPE leads with an average precision of 99.13%, outperforming NSGA-II's second-place with precision of 98.85%, RF's third-place with precision of 98.79%, LR's in fourth-place with precision of 98.68%, and RFE's

last-place with precision of 98.46%. Similarly, in recall metric DFA-GPE again secures the top position with recall value 99.03%, followed by NSGA-II of 98.68%, RF in third position with a recall of 98.62%, LR in fourth with a recall of 98.52%, and RFE in last place with a recall of 98.24%. Moreover, in F1-score metric DFA-GPE continues to lead with value of 99.07%, followed by NSGA-II with value of 98.75%, then RF with a value of 98.69%, LR fourth with an F1-score of 98.58%, finally, RFE with a value of 98.33%.

Subsequently, the TON_IoT dataset results further underscores DFA-GPE's superiority, showcasing its exceptional performance across all evaluation metrics. It achieves an average accuracy of 92.64%, surpassing NSGA-II with an average accuracy of 92.18%, followed by RF with an average accuracy of 92.01%, then RFE with an average accuracy of 91.51%, last LR with an average accuracy of 90.74%. In precision metric, DFA-GPE also ranks first with an average precision of 92.86%, RF in second ranking with an average precision of 92.49%, NSGA-II third with an average precision of 92.48%, RFE fourth with an average precision of 91.67%, LR last with an average precision of 91.05%. Furthermore, in recall metric DFA-GPE preserves the lead with 92.76%, followed by NSGA-II with value of 92.31%, RF third with value of 92.10%, RFE fourth with a value of 91.53%, and LR last with a recall of 90.76%. Last, in F1-score metric, DFA-GPE again ranks highest with a value of 92.78%, followed by NSGA-II with an F1-score of 92.35%, RF third with a value of 92.22%, RFE fourth with a value of 91.57%, LR last with value of 90.86%.

Table 4.8: Overall performance of classification metrics obtained on both datasets

Model	HIKARI Dataset			
	Accuracy	**Precision**	**Recall**	**F1_score**
DFA-GPE	**99.09%**	**99.13%**	**99.03%**	**99.07%**
Random Forest [206]	98.73%	98.79%	98.62%	98.69%
Logistic Regression [122]	98.62%	98.68%	98.52%	98.58%
Recursive Feature Elimination [123]	98.39%	98.46%	98.24%	98.33%
NSGA-II [50]	98.79%	98.85%	98.68%	98.75%
Model	**TON_IoT Dataset**			
	Accuracy	**Precision**	**Recall**	**F1_score**
DFA-GPE	**92.64%**	**92.86%**	**92.76%**	**92.78%**
Random Forest [206]	92.01%	92.49%	92.10%	92.22%
Logistic Regression [122]	90.74%	91.05%	90.76%	90.86%
Recursive Feature Elimination [123]	91.51%	91.67%	91.53%	91.57%
NSGA-II [50]	92.18%	92.48%	92.31%	92.35%

Moreover, Figure 4.52 demonstrates the comparative performance of mean, max, and median values for the various feature selection methods with classification over the two datasets, measured over a data streaming across 400 chunks. The DFA-GPE clearly outperforms its counterparts across both datasets, registering a mean accuracy of 92.64% for TON_IoT and an even more impressive 99.09% for HIKARI. This indicates robust adaptability and superior predictive ability, especially when handling data streams with significant variability. The NSGA-II method consistently holds the second position with mean accuracy of 92.18% for TON_IoT and 98.79% for HIKARI, considered a credible competitor but with noticeably less efficacy than the proposed method. This is followed by RF method with a mean accuracy of 92.01% for TON_IoT and 98.73% for HIKARI, while these results are noteworthy, they reveal a performance differential that the DFA-GPE method efficiently addresses. The RFE method secures the fourth place in TON_IoT by achieving 91.51% and fifth in HIKARI with a mean accuracy of 98.39%, overall, its ranking fourth across both datasets, followed by the LR method, which ranks fourth in HIKARI with a mean accuracy of 98.62% and last in TON_IoT with a mean accuracy of 90.74%.

The DFA-GPE framework's exceptional performance in data stream classification is highlighted by its comprehensive integration of advanced techniques. The innovative application of a GP combiner, combined with concept drift detection method, enables effective adaptation to the changing distributions characteristic of data streams. Central to its success is DFS for handling feature drift and reducing dimensionality, which significantly enhances classification accuracy and provides adaptability in dynamic data scenarios to ensure reliability and maintain classification accuracy over time. Additionally, the inclusion of SMOTE for class imbalance correction produces accurate classifiers. The DFA-GPE method also adopts an incremental learning strategy, allowing for the seamless incorporation of new data without the need to retain old data. This approach conserves memory and ensures efficiency in environments where data distributions can rapidly evolve.

The DFS component, optimized with the advanced MOO algorithm named Variable Length Multi-Objective Particle Swarm Optimization (VLMO-PSO), significantly enhances performance amidst feature drifts and changing streaming conditions. It benefits from strategic population initialization using Bernoulli distributions and symmetric uncertainty metrics to refine the balance between exploration and exploitation. Moreover, the framework's division-based search criterion, which segments the solution space based on objectives' impact, along with variable length representation for features, simplifies the search process, especially valuable in high-dimensional data scenarios. Supported by binary solutions representation and the use of dual transfer functions, DFS finely tunes the balance between exploration and exploitation. The introduction of a novel exemplar selection mechanism with multi-objective awareness ensures a diverse and comprehensive exploration of the solution

space, leading to a well-distributed Pareto front. This MOO FS is further enhanced by statistical feature weighting via the Pareto front, adeptly tackling DFS as a MOO challenge. This strategy is aimed at achieving the highest classification accuracy while minimizing memory usage. As a results DFS's continuously update of the IDS classifier with the most relevant features effectively manages the changing significance of features within data streams.

The justification for achieving top-tier classification metrics while observing modest results in memory reduction is derived from the well balancing MOO method and the voting technique applied in solution selection from the VLMO-PSO Pareto front. This approach ensures a comprehensive exploration of the solution space, focusing on maintaining high classification accuracy without unduly compromising memory efficiency. Where other benchmarks may consider one aspect of feature selection objectives by sacrificing the accuracy aspect, a delicate balance between accuracy and feature set compactness is prioritized, recognizing the inherent trade-offs in MOO. The importance of solution diversity is underscored by leveraging a voting mechanism for feature selection, thus enhancing the generalization ability over minimizing memory usage, which explains the modest memory reduction outcomes.

The innovative design and strategic optimizations of the DFA-GPE IDS method significantly and collectively address existing gaps in handling feature and concept drift, high-dimensional data, DFS, and imbalance management within the IDS domain. This holistic approach not only enhances performance metrics but also ensures a balanced consideration of accuracy and memory efficiency, establishing DFA-GPE as a cornerstone for future advancements in feature selection methodologies for IDS in data stream scenario.

4.5.5 Detection Lag and Model Maintenance

Detection lag analysis and the time delay between the occurrence of an intrusion and its detection by the DFA-GPE IDS, is a crucial factor in real-time security environments. While the thesis primarily emphasizes classification accuracy across various datasets, it is equally important to analyze detection lag, particularly in scenarios requiring immediate threat responses. In our method, detection lag is managed through a chunk-based approach where data is processed in fixed-size chunks of 500 records. Feature selection is performed on each chunk as it arrives, helping to mitigate feature drift. Additionally, a forehead buffer with a capacity of 2000 records is employed to temporarily hold data for further actions. When the DDM identifies a change in data distribution, the current chunk is held for conditional processing. If a drift is detected, the system undertakes two processes: first, new classifiers are trained and the top-performing ones are integrated into the ensemble to handle incoming data; second, older classifiers in the ensemble are replaced at a rate of one-third. Therefore, detection lag is primarily influenced by the time taken to measure data distribution changes with DDM and the time required to train new classifiers for ensemble integration.

For the model maintenance the framework is designed with an incremental learning approach, which continuously updates and refines the model as new data becomes available, eliminating the need for complete retraining. This method allows the model to adapt to new patterns and shifts in the data over time, making it particularly well-suited for real-time applications or environments with evolving data streams. Incremental learning ensures that the model remains relevant and accurate while

efficiently managing computational resources. Additionally, the framework may require:

Performance Monitoring: Regularly monitoring the accuracy and performance of the ensemble model to ensure it meets the expected standards. This involves continuously tracking performance metrics to identify any deterioration or shifts in model effectiveness, allowing for timely interventions when necessary.

Resource Allocation: Ensuring that adequate computational resources are available for real-time data processing, model updating, and re-training. Effective management of memory and processing power is crucial to maintain the system's performance, particularly in high-speed or resource-constrained environments.

These practices are essential for sustaining the long-term effectiveness of the IDS, enabling it to swiftly adapt to new threats and changes in data without the extensive computational burden associated with traditional retraining methods.

Figure 4.52: Mean, Max, Median for Each Metrics Accuracy, Precision, Recall F1-Score in Both Datasets

4.5.6 Synthetic Data Results for DFA-GPE

In the comprehensive evaluation of the performance of the proposed DFA-GPE, it was assessed using synthetic data generated by the MOA simulator. This approach allowed

for the evaluation of performance against specific types of synthesized concept drifts, thereby identifying the strengths and weaknesses of DFA-GPE in relation to different concept drift categories. Additionally, the synthetic data-based evaluation was helpful in determining which types of drifts are more effectively managed by the model and which are less so. This analysis fosters a deeper understanding of the proposed DFA-GPE's capabilities. Four types of multi-dimensional time series data sudden, gradual, recurrent, and blip were generated using MOA. Three numerical features with binary classifications. Drift detected within each time series were identified using ADWIN, a classifier-independent method that facilitates the detection of drifts and the objective comparison of their effects across different types of IDS.

To measure the correlation between features over time for two datasets gradual and sudden, we implemented a rolling window approach for 100 records in both datasets. A window of 10 records was used to compute the correlation matrix for each window. The correlations were averaged to handle any duplicate entries, and a heatmap was generated to visualize the correlation degrees over time. This process implemented on both datasets to analyze feature drifting refer to Figure 4.53 and Figure 4.54.

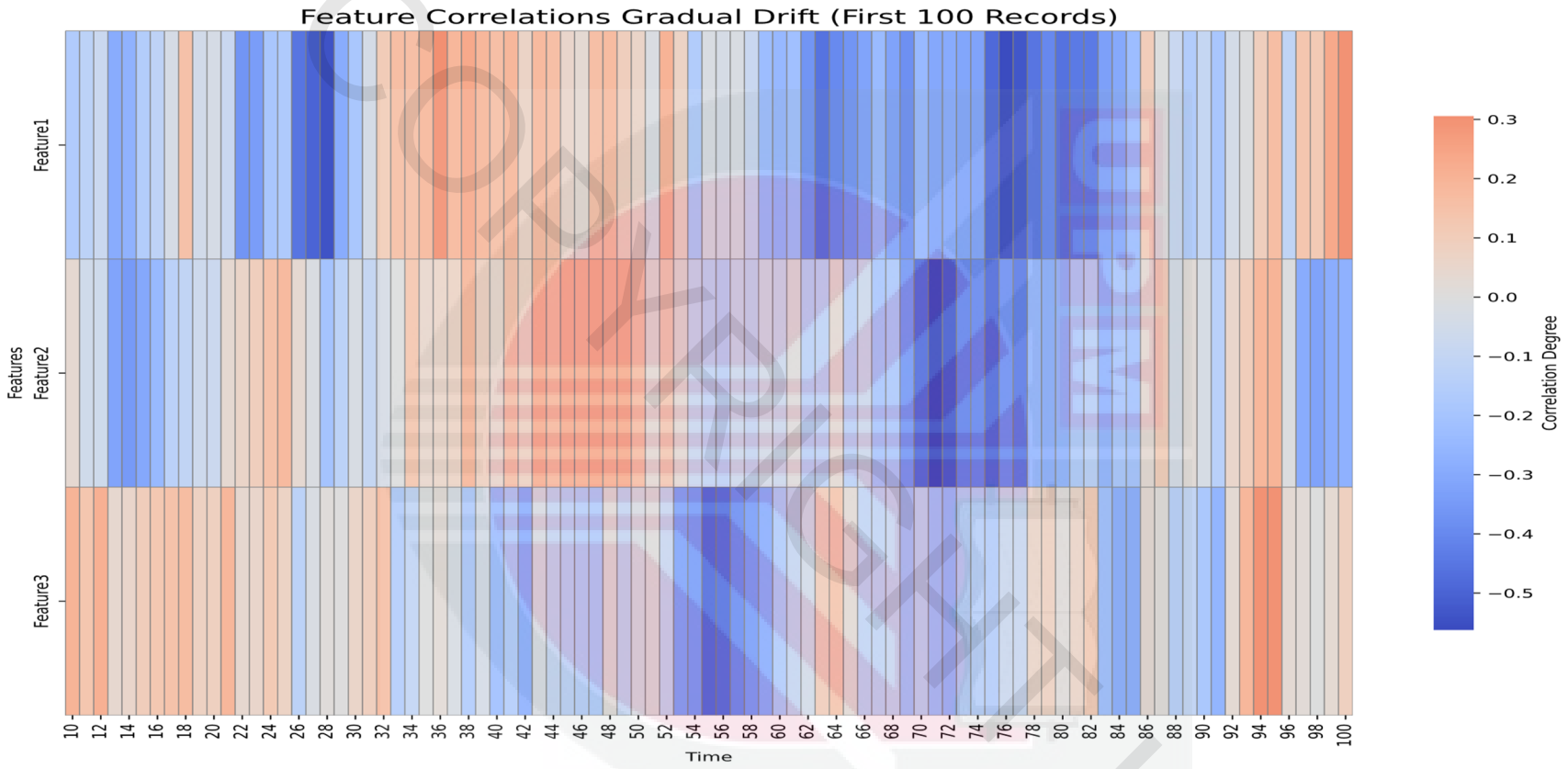


Figure 4.53: Features Correlations Measure in Gradual Dataset for 100 Records

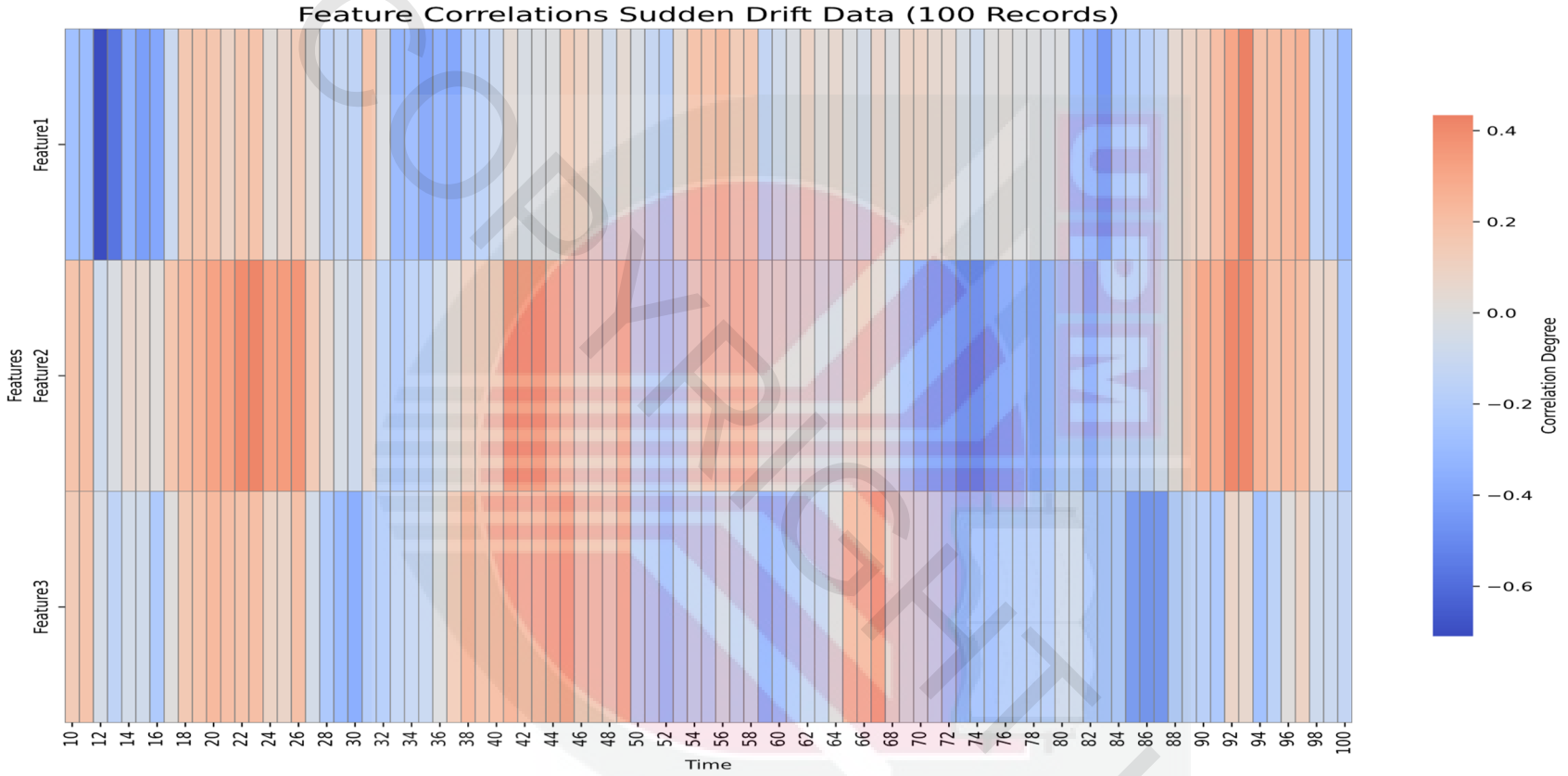


Figure 4.54: Features Correlations Measure in Sudden Dataset for 100 Records

The heatmaps presented show the correlation degrees between features over time for the first 100 records, indicating how correlations vary as a result of feature drifting. The first heatmap illustrates the feature correlations for the gradual drift dataset, while the second heatmap demonstrates the feature correlations for the sudden drift dataset. In both heatmaps, the color variations indicate the degree of correlation, with red representing high positive correlation and blue representing high negative correlation. As observed, the correlation degrees between features (Feature1, Feature2, and Feature3) fluctuate over time. These varying correlations suggest that the importance of features shifts as time passes, indicating feature drifting. Periods where features become more or less correlated reflect changes in their relationships, highlighting the dynamic nature of the datasets and the need for models to adapt to these changes.

4.5.6.1 Sudden Drift

In the experiment for the synthetic dataset as shown in Figure 4.55, a single sudden drift was detected around the 7000-sample period, introducing a critical drift in the time series data. This event served as a test of resilience for the methodologies being evaluated. Post-drift, the DFA-GPE method demonstrated exceptional robustness, leading with a mean accuracy of 0.844, while the RF and LR methods showed commendable performance with identical mean accuracies of 0.843. The FRE, although less resilient, still managed to maintain a respectable mean accuracy of 0.833, indicating a slightly more noticeable susceptibility to sudden drifts. Overall, the DFA-GPE's adaptability to abrupt changes underscored its potential for maintaining predictive accuracy in dynamic environments.

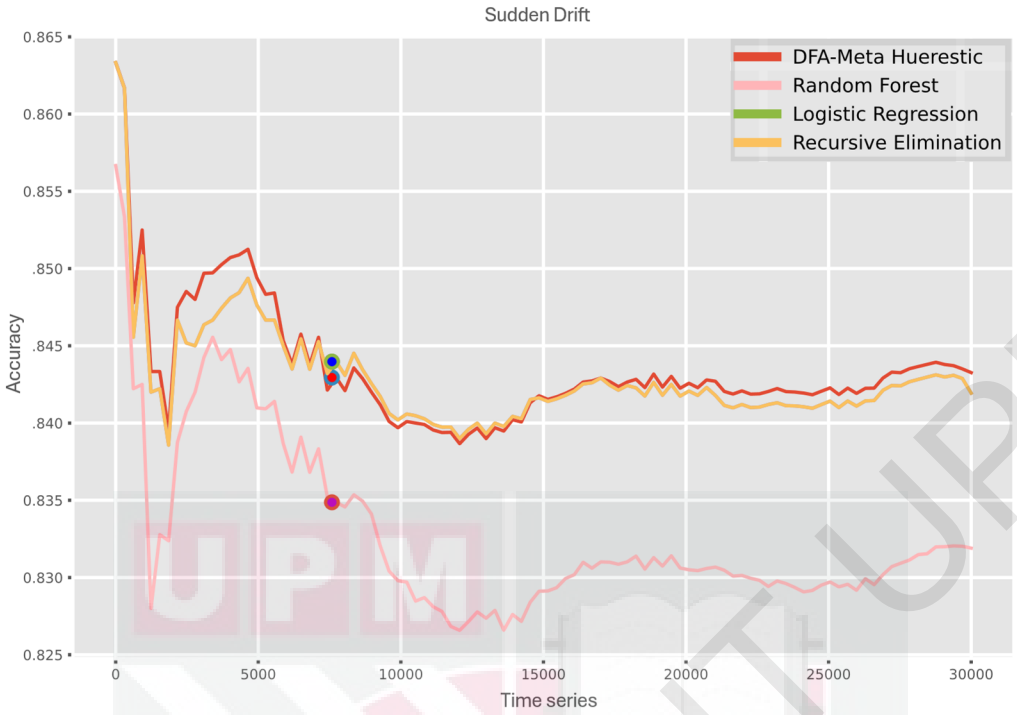


Figure 4.55: Result Comparison of Our Method with Others in Detecting Sudden Drift

4.5.6.2 Gradual Drift

The results of gradual synthetic dataset, the time series analysis as illustrated in the Figure 4.56, five drifts were detected at various sample points, notably around the 5000th, 15000th, and 25000th time series, as indicated by the colored markers on the graph. These moments marked significant shifts in the data distribution, to which the DFA-GPE method demonstrated superior resilience, maintaining a mean accuracy of 0.781. This was contrasted with the other methods, where RF, LR, and RFE showed mean accuracies of 0.771, 0.610, and 0.613, respectively, with more noticeable declines in performance at these critical stages.

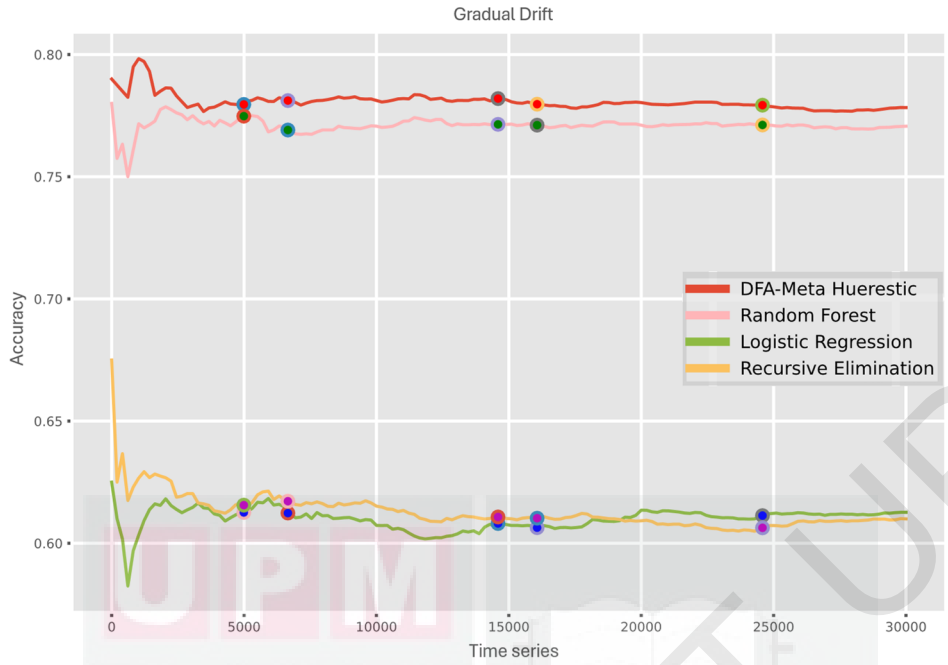


Figure 4.56: Result Comparison of Our Method with Others in Detecting Gradual Drift

4.5.6.3 Recurrent Drift

The time series analysis for recurrent drift data as shown in Figure 4.57, a total of 8 drifts were detected, highlighting the dynamic nature of the dataset. Three of these drifts were identified at the onset of the experiment, indicating immediate shifts in the data stream. Another three drifts were observed in the interval between 5000 and 10000 samples, signifying a period of significant instability. The last two drifts were detected around the 15000-sample mark, marking another phase of change within the data. It was observed that the DFA-GPE method sustained a mean accuracy of 0.773, indicating a robust performance amidst the ebb and flow of data changes. This figure surpasses the mean accuracy of the RF and RFE methods, which held at 0.762 and 0.767, respectively. Interestingly, LR presented a competitive mean accuracy of 0.777, closely rivalling the DFA approach.

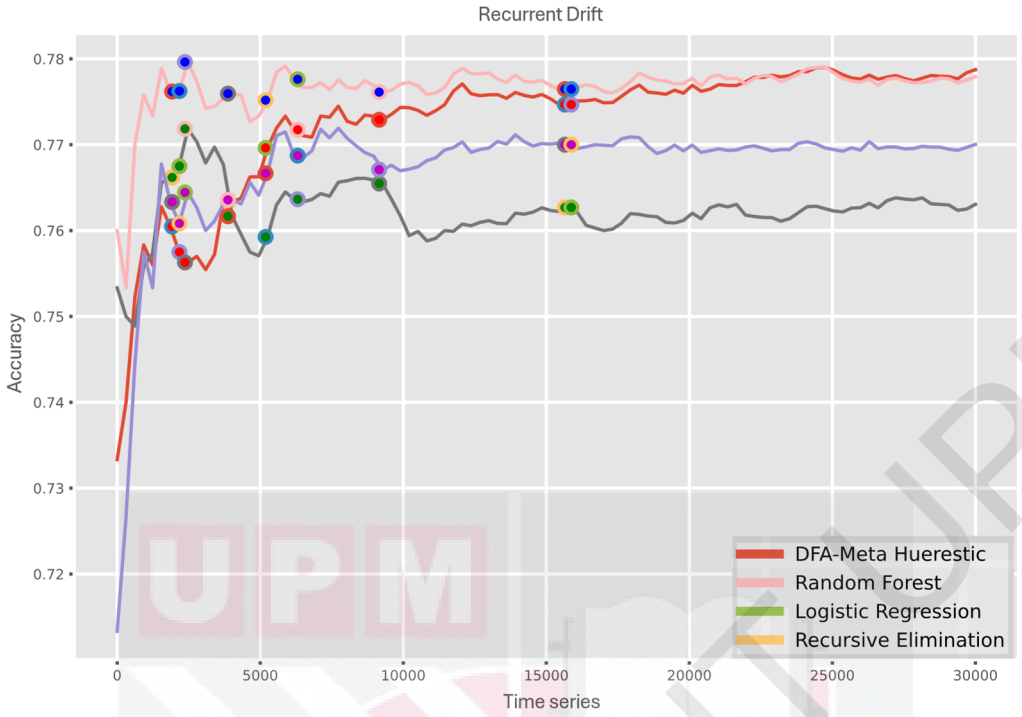


Figure 4.57: Result Comparison of Our Method with Others in Detecting Recurrent Drift

4.5.6.4 Blip Drift

The time series evaluation for blip drift dataset as demonstrated in Figure 4.58, a numerous drifts were detected throughout the entire time series of experiment, indicating a highly dynamic data environment. Such conditions rigorously tested the adaptability of the feature selection methodologies. The DFA-GPE method yielded a mean accuracy of 0.726, reflecting its capability to navigate through the frequent drifts. RF slightly outperformed the DFA-GPE with a mean accuracy of 0.72٨, suggesting a marginally better response to the continuous changes. LR and RFE demonstrated lower mean accuracies of 0.720 and 0.7١٠, respectively, which may indicate a comparative vulnerability to the frequent and sudden shifts characteristic of 'Blip drifts'. This comprehensive analysis underscores the importance of method selection in time series models subject to frequent data distribution changes.

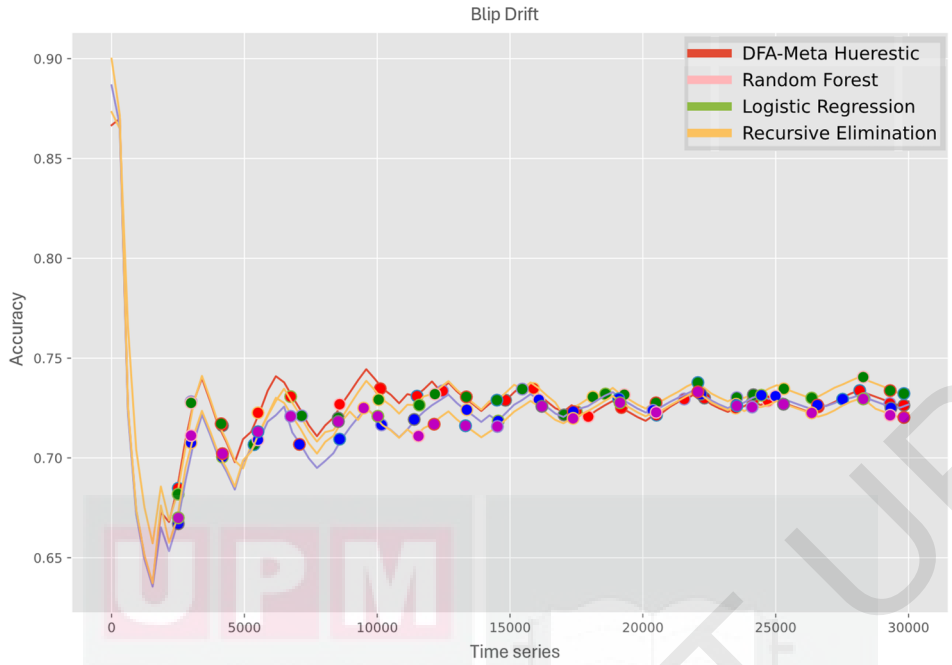


Figure 4.58: Result Comparison of Our Method with Others in Detecting Blip Drift

4.5.6.5 DFA-GPE Synthetic Data Discussion

The evaluating DFA-GPE's performance with synthetic data for generating various types of concept drifts, our method demonstrated robust adaptability to different concept drift types: sudden, gradual, recurrent, and well performance in blip. For sudden drift, a single drift marked, DFA-GPE showed exceptional resilience, leading with a mean accuracy of 0.844, surpassing other methods. In gradual drift, DFA-GPE five drifts are marked while maintained a mean accuracy of 0.781, outperforming RF, LR, and RFE at critical stages. Recurrent drift analysis with 8 detected drifts marked with DFA-GPE's robust mean accuracy of 0.773, closely rivalling LR. Lastly, in blip drift with frequent shifts, DFA-GPE displayed a mean accuracy of 0.726, slightly trailing RF. Overall, these results highlight DFA-GPE as a robust and adaptable solution for handling diverse concept drifts in dynamic scenarios.

DFA-GPE's outstanding adaptability to various concept drifts, sudden, gradual, recurrent, and blip, directly attributable to its GP ensemble design and DFS based on MOO. The GP ensemble

approach provides a diverse set of classifiers, enabling rapid adaptation to new data trends, essential for maintaining accuracy in dynamic DataStream environments. Concurrently, DFS ensures ongoing relevance by selecting the most informative features as data evolves, a critical factor for sustained performance across nonstationary scenarios. This strategic combination allows DFA-GPE to excel, particularly demonstrated by its leading accuracy in sudden drift and effective handling of gradual and recurrent changes. The synergy between GP ensemble diversity and dynamic, optimized feature selection underpins DFA-GPE's robust performance, making it a highly adaptable solution for dynamic intrusion detection challenges.

4.5.7 Theoretical Analysis

The proposed methods are deeply rooted in the principles of Genetic Programming [213] [140] [32] and Particle Swarm Optimization [119] [23] [214], both of which have a well-established history in addressing complex optimization challenges. GP's dynamic evolution of solutions, inspired by natural selection, provides a robust theoretical foundation for managing nonstationary data streams and concept drift. PSO, grounded in swarm intelligence, enables simple agents to collaboratively explore the solution space without the need to exhaustively test all possible feature combinations, making it particularly effective in multi-objective optimization tasks.

Integrating these techniques into the IDS framework is theoretically justified, leveraging GP's adaptability to evolving data distributions and PSO's strengths in balancing exploration and exploitation. The inclusion of an external archive to accumulate the best solutions across iterations further enhances the robustness of the framework [167]. Additionally, the multi-exemplar strategies employed in MEPSOLA significantly increase diversity within the solution

space [26], effectively mitigating the well-known issue of premature convergence in optimization algorithms. The theoretical analysis of PSO's convergence and diversity properties supports the inclusion of adaptive strategies, such as Tabu search, to avoid local optima and ensure resilience in dynamic environments [184].

A comprehensive analysis of the algorithms' performance under varying conditions has been conducted, including tests on different types of mathematical functions, real datasets, and synthetic data. The convergence behavior of MEPSOLA and VLMO-PSO has been theoretically examined, particularly their capacity to navigate complex multi-objective landscapes and escape local optima. This analysis is bolstered by performance bounds that define the expected limits of accuracy and computational efficiency, grounded in the algorithms' design principles. For example, the integration of Tabu search within the PSO framework is theoretically validated as a means to reduce the likelihood of stagnation in local optima, thereby enhancing the algorithm's overall effectiveness in handling complex, high-dimensional spaces.

Furthermore, the theoretical bounds on memory usage and detection accuracy have been carefully evaluated, with particular emphasis on how the framework balances these often-conflicting objectives. The analysis includes a comparison of the expected performance of the DFA-GPE and MEPSOLA algorithms against traditional methods, underscoring their superior ability to maintain high accuracy while minimizing resource consumption. This theoretical foundation ensures that the experimental results are not merely empirical but are also reinforced by a rigorous analysis of the algorithms' anticipated performance in real-world scenarios.

4.6 Summary

This chapter investigates into the development and evaluation of two innovative frameworks: MEPSOLA, designed for solving complex optimization problems, and DFA-GPE, tailored for data stream-based IDS. The experimental design is meticulously outlined, illustrating the application of these frameworks across various mathematical functions for assessing MOO and real datasets to evaluate data stream-based IDS threat detection efficacy and drift handling.

MEPSOLA stands out for its optimization prowess across six mathematical functions, surpassing traditional algorithms like MOEA, NSGA-2, and NSGA-3 across multiple metrics such as in C-metric, HV, NDS and it show a decent performance in the GD and DM. This optimization strength serves as a foundation for enhancing online IDS decision-making processes. Additionally, the efficacy of DFA-GPE is detailed, with results from IoT and network datasets showcasing its strength in binary and multi-classification tasks. It consistently outperforms other methods in accuracy, precision, recall, and F1-score, with significant memory consumption reduction and adaptability to streaming data in both utilized real datasets, exhibiting resilience to feature and concept drift.

These findings affirm the substantial contributions of MEPSOLA and DFA-GPE to the field of MOO and streaming-based network security. Their effectiveness not only validates their application in combating evolving network threats but also establishes a precedent for future advancements in this field.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

The contemporary world, characterized by the widespread proliferation of extensive networks generating complex, high-dimensional data, and presents significant challenges to traditional data analysis techniques. The complexity of these data streams is coupled with the increasing and evolving risk of cyber threats, highlights the vital importance of IDS. However, many existing IDSs, designed to operate in static networks with the assumption of stationary data flow, may struggle to manage the phenomenon of feature drift in non-stationary data streams. In these streams, the significance of specific features may shift, transforming them from relevant to irrelevant or vice versa, potentially rendering existing detection models less effective or even obsolete. The literature highlights the critical need for advanced IDS capable of addressing the complicated challenges presented by non-stationary data streams. This necessity is underscored by the dynamic nature of cyber threats and the evolving landscape of network security, emphasizing the need to address significant gaps in handling the phenomena of feature drift, while also jointly addressing other important challenges such as high dimensionality, concept drift, and imbalanced data.

To address these challenges, this thesis embarked on a comprehensive journey, systematically advancing the field of IDS. It initiated with the development of a framework named DFA-GPE, which combines various techniques to tackle data stream challenges. The process began with DFS periodically assessing feature relevancy through a developed VLMO-PSO algorithm; this

method addresses the problem of feature drift and reduces dimensionality while enhancing adaptability and accuracy. The GP combiner, incorporated for ensemble classification, further refines predictive models to cope with nonstationary data streams. Additionally, the framework employs a DDM to detect and manage concept drift, ensuring sustained model performance in evolving cybersecurity environments. The framework incorporates an incremental learning procedure, essential for streaming IDS, to continually update its knowledge base with new data, enabling real-time adaptation to evolving threats. Furthermore, the SMOTE technique was integrated to ensure a balanced class distribution. The framework was evaluated using two real IDS datasets, namely HIKARI 2021 and TON_IoT 2020, in stream mode with single and multi-class labels, and it was compared with state-of-the-art algorithms in the literature, demonstrating its superiority.

Building on this foundation, the thesis introduced a novel MOPSO framework named MEPSOLA designed to solve complex optimization problems, integrating advanced strategies such as smart initialization, multi exemplar and conditional local search incorporated in exemplar selection enhancement this a multi-objective PSO algorithm specifically designed to address the complexities of high-dimensional solution spaces. Rigorous evaluation of the algorithm has been conducted using standard mathematical benchmarks prevalent in the optimization domain. Furthermore, the algorithm's performance metrics have been evaluated using established multi-objective evaluation metrics, and comparative analyses have been performed against benchmark algorithms in the field of optimization.

The final step involved adjusting and adapting MEPSOLA framework modified to be named a Variable Length Multi-Objective Particle Swarm Optimization (VLMO-PSO) variant, which is tailored for online decision making DFS. This adaptation signifies a critical leap forward in operationalizing the IDS to remain effective against evolving cyber threats, as well addressing

feature drift. The algorithm introduces a division-based search criterion, partitioning the solution space into discrete segments based on their impact on objective values, thereby streamlining the search process. Furthermore, it enhances adaptability to binary solution spaces by integrating mixed transfer functions and a probabilistic selection mechanism, facilitating effective particle mobility. Additionally, a multi-objective-aware criterion for exemplar selection optimizes the balance between exploration and exploitation, significantly improving the algorithm's performance across varied objectives. The voting-based Pareto front solution selection achieves a balanced trade-off between accuracy and feature reduction, avoiding bias towards any single objective.

The results evaluation highlights MEPSOLA emerges as a highly effective approach for MOO challenges, particularly evident through its superior performance in the SC measure for an example. The detailed comparison reveals MEPSOLA achieving a SC measure average median of 0.7692 against MOEA of 0.006, 0.3258 against NSGA-2 of 0.101, and it achieved 0.3226 against NSGA-3 of 0.0207. This dominance in SC measure not only underscores MEPSOLA's ability to navigate and optimize the solution space efficiently but also highlights its capacity to ensure a broad and diverse representation of solutions across the Pareto front results.

In DFA-GPE framework, the focus shifts towards achieving impressive accuracy in the domain of IDS, where it sets new benchmarks against contemporary FS methods. Within the HIKARI dataset, DFA-GPE achieved the highest accuracy of 99.09%, surpassing NSGA-II of 98.79%, RF of 98.73%, LR of 98.62%, and RFE 98.39%. Where the memory utilization, requiring merely 17.298 kilobytes, a stark reduction from the initial memory usage of 265.754 kilobytes. This translates into a memory saving of approximately 93.49% while reducing the number of features from 67 to just 5 features. Similarly, the results for TON_IoT dataset, DFA-GPE leads

with an accuracy of 92.64%, outperforming NSGA-II of 92.18%, RF of 92.01%, RFE of 91.51%, and LR of 90.74%. In the memory reduction, utilizing only 60.871 kilobytes of memory from total utilized 164.191 kilobytes and reducing the feature count from 42 to 16. This achievement represents a memory saving of 62.93%, further highlighting DFA-GPE's strength in executing a strategic balance between classifier accuracy and memory usage.

Based on our knowledge, the thesis significantly advances the field of IDS by introducing an advanced framework adept at handling the challenges of nonstationary data streams, feature and concept drift, high dimensionality, and the constant threat of cyber-attacks. The feature and concept drifting are handled and mitigated by incorporating dynamic feature selection in the Genetic Programming Ensemble DFA-GPE classifier. The framework achieves this by continually assessing the relevancy of features with each arrived group of instances, where the GP ensemble continually monitors the data stream distributions by tracking the classification error rate and adaptively reacting to these changes. The findings demonstrate the effectiveness of incorporating DFS into a unified framework with a GP-combiner ensemble classifier for mitigating the influence of evolving features and reducing dimensionality. This superiority of the proposed method over other benchmarks throughout the experiment with a real dataset and synthetic is proven. Furthermore, the method achieves an optimal balance between accuracy and memory reduction by integrating meta-heuristic techniques based on MOO and PSO, specifically addressing the gap between these two conflicting objectives. By employing an ensemble learning strategy with a GP combiner, it effectively addresses the critical issue of concept drift, ensuring robust and adaptable defense mechanisms against evolving cyber threats. The thesis marks a pivotal shift in network security and data stream processing, asserting that the adaptability of IDS to the dynamic nature of data streams is essential, not a luxury. By pioneering a framework that responds with agility to evolving threats, this work

challenges the status quo and sets a new direction for future defenses. It stands as a testament to the power of innovation in combating cyber threats, ensuring the resilience of digital defenses over time.

5.2 Limitations and Future Works

5.2.1 The limitations of the research are presented below

1. Assessment of Algorithm's Scalability: The scalability of the algorithm on larger and more diverse datasets is yet to be assessed. This limitation is recognized as an area where further investigations are warranted.

2. Optimization of Computational Efficiency: The optimization of computational efficiency remains a future focus. It has been noted that this aspect holds potential for significant improvements, which are yet to be explored.

3. Exploration of Alternative Meta-Heuristic Algorithms: The exploration of alternative meta-heuristic algorithms or variants as enhancements to the current method has been identified as a prospective area for future research. This limitation suggests that the current method's adaptability and efficiency could be further enhanced through such explorations.

4. Application Across Different Domains: The adaptability of the proposed solution, MEPSOLA, to a wider range of real-world problems beyond the initial scope is yet to be fully explored. This limitation points to the potential for broader applicability that requires further research.

5. Enhancement of Convergence Consistency: The enhancement of MEPSOLA's convergence consistency through hyperparameter refinement is recognized as a limitation. It has been suggested that addressing this could potentially improve the algorithm's performance, which is an area for future work.

6. Integration with Other Optimization Techniques: The integration of MEPSOLA with other optimization techniques to provide deeper insights and advancements in the field is identified as a limitation. Future work is seen as necessary to explore these integrations and their potential to advance the field further.

5.2.2 Future works

There are several key areas that future research will aim to explore based on the limitations noted in this study. expanding the assessment of the algorithm's scalability to include larger and more varied datasets will be crucial in understanding its broader applicability. Additionally, efforts to optimize computational efficiency could lead to significant enhancements in the algorithm's performance. Investigating the integration of alternative meta-heuristic algorithms promises to further refine the method's adaptability and efficiency. Exploring MEPSOLA's application across different domains will also be essential in determining its suitability for a wide range of real-world problems. Moreover, focusing on the refinement of hyperparameters to improve convergence consistency presents an opportunity to elevate the algorithm's effectiveness. Lastly, integrating MEPSOLA with other optimization techniques could offer deeper insights and contribute to advancements in the field, marking critical directions for future research endeavors.

REFERENCES

[1] I. Souiden, M. N. Omri, and Z. Brahmi, "A survey of outlier detection in high dimensional data streams," *Comput. Sci. Rev.*, vol. 44, p. 100463, May 2022, doi: 10.1016/j.cosrev.2022.100463.

[2] M. Rong, D. Gong, and X. Gao, "Feature Selection and Its Use in Big Data: Challenges, Methods, and Trends," *IEEE Access*, vol. 7, pp. 19709–19725, 2019, doi: 10.1109/ACCESS.2019.2894366.

[3] I. H. Sarker, A. S. M. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: an overview from machine learning perspective," *J. Big Data*, vol. 7, no. 1, Art. no. 1, Dec. 2020, doi: 10.1186/s40537-020-00318-5.

[4] H. Al-Hamadi, I.-R. Chen, D.-C. Wang, and M. Almashan, "Attack and Defense Strategies for Intrusion Detection in Autonomous Distributed IoT Systems," *IEEE Access*, vol. 8, pp. 168994–169009, 2020, doi: 10.1109/ACCESS.2020.3023616.

[5] L. N. Tidjon, M. Frappier, and A. Mammar, "Intrusion Detection Systems: A Cross-Domain Overview," *IEEE Commun. Surv. Tutor.*, vol. 21, no. 4, pp. 3639–3681, 2019, doi: 10.1109/COMST.2019.2922584.

[6] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," 1980.

[7] A. S. Dina, A. B. Siddique, and D. Manivannan, "A deep learning approach for intrusion detection in Internet of Things using focal loss function," *Internet Things*, vol. 22, p. 100699, Jul. 2023, doi: 10.1016/j.iot.2023.100699.

[8] S. Agrahari and A. K. Singh, "Concept Drift Detection in Data Stream Mining: A literature review," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9523–9540, Nov. 2022, doi: 10.1016/j.jksuci.2021.11.006.

[9] O. Abdel Wahab, "Intrusion Detection in the IoT Under Data and Concept Drifts: Online Deep Learning Approach," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 19706–19716, Oct. 2022, doi: 10.1109/JIOT.2022.3167005.

[10] L. Yang and A. Shami, "A Lightweight Concept Drift Detection and Adaptation Framework for IoT Data Streams," *IEEE Internet Things Mag.*, vol. 4, no. 2, Art. no. 2, Jun. 2021, doi: 10.1109/IOTM.0001.2100012.

[11] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer, "A survey on feature drift adaptation: Definition, benchmark, challenges and future directions," *J. Syst. Softw.*, vol. 127, pp. 278–294, May 2017, doi: 10.1016/j.jss.2016.07.005.

[12] H. Ding, L. Chen, L. Dong, Z. Fu, and X. Cui, "Imbalanced data classification: A KNN and generative adversarial networks-based hybrid approach for intrusion detection," *Future Gener. Comput. Syst.*, vol. 131, pp. 240–254, Jun. 2022, doi: 10.1016/j.future.2022.01.026.

[13] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artif. Intell. Rev.*, Jul. 2021, doi: 10.1007/s10462-021-10037-9.

[14] P. Zhou, S. Zhao, Y. Yan, and X. Wu, "Online Scalable Streaming Feature Selection via Dynamic Decision," *ACM Trans. Knowl. Discov. Data*, vol. 16, no. 5, p. 87:1-87:20, Mar. 2022, doi: 10.1145/3502737.

[15] D. Chou and M. Jiang, "A Survey on Data-driven Network Intrusion Detection," *ACM Comput. Surv.*, vol. 54, no. 9, Art. no. 9, Dec. 2022, doi: 10.1145/3472753.

[16] J. Jesus, A. Canuto, and D. Araújo, "An exploratory analysis of data noisy scenarios in a Pareto-front based dynamic feature selection method," *Appl. Soft Comput.*, vol. 100, p. 106951, Mar. 2021, doi: 10.1016/j.asoc.2020.106951.

[17] W. Elmasry, A. Akbulut, and A. H. Zaim, "Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic," *Comput. Netw.*, vol. 168, p. 107042, Feb. 2020, doi: 10.1016/j.comnet.2019.107042.

[18] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, Art. no. 1, Jan. 2014, doi: 10.1016/j.compeleceng.2013.11.024.

[19] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, and A. Rodríguez, "A better balance in metaheuristic algorithms: Does it exist?," *Swarm Evol. Comput.*, vol. 54, p. 100671, May 2020, doi: 10.1016/j.swevo.2020.100671.

[20] T. Dokeroglu, A. Deniz, and H. E. Kiziloz, "A comprehensive survey on recent metaheuristics for feature selection," *Neurocomputing*, vol. 494, pp. 269–296, Jul. 2022, doi: 10.1016/j.neucom.2022.04.083.

[21] C. A. Coello Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, Honolulu, HI, USA: IEEE, 2002, pp. 1051–1056. doi: 10.1109/CEC.2002.1004388.

[22] M. Nssibi, G. Manita, and O. Korbaa, "Advances in nature-inspired metaheuristic optimization for feature selection problem: A comprehensive survey," *Comput. Sci. Rev.*, vol. 49, p. 100559, Aug. 2023, doi: 10.1016/j.cosrev.2023.100559.

[23] B. Tran, B. Xue, and M. Zhang, "Variable-Length Particle Swarm Optimization for Feature Selection on High-Dimensional Classification," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, Art. no. 3, Jun. 2019, doi: 10.1109/TEVC.2018.2869405.

[24] P. Agrawal, T. Ganesh, D. Oliva, and A. W. Mohamed, "S-shaped and V-shaped gaining-sharing knowledge-based algorithm for feature selection," *Appl. Intell.*, vol. 52, no. 1, pp. 81–112, Jan. 2022, doi: 10.1007/s10489-021-02233-5.

[25] B. Gu, X. Quan, Y. Gu, V. S. Sheng, and G. Zheng, "Chunk incremental learning for cost-sensitive hinge loss support vector machine," *Pattern Recognit.*, vol. 83, pp. 196–208, Nov. 2018, doi: 10.1016/j.patcog.2018.05.023.

[26] W. Song and Z. Hua, "Multi-Exemplar Particle Swarm Optimization," *IEEE Access*, vol. 8, pp. 176363–176374, 2020, doi: 10.1109/ACCESS.2020.3026620.

[27] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization," *Swarm Evol. Comput.*, vol. 9, pp. 1–14, Apr. 2013, doi: 10.1016/j.swevo.2012.09.002.

[28] A. Thakkar and R. Lohiya, "A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenges," *Arch. Comput. Methods Eng.*, Oct. 2020, doi: 10.1007/s11831-020-09496-0.

[29] D. J. Kalita, V. P. Singh, and V. Kumar, "A novel adaptive optimization framework for SVM hyper-parameters tuning in non-stationary environment: A case study on intrusion detection system," *Expert Syst. Appl.*, vol. 213, p. 119189, Mar. 2023, doi: 10.1016/j.eswa.2022.119189.

[30] M. Jain, G. Kaur, and V. Saxena, "A K-Means clustering and SVM based hybrid concept drift detection technique for network anomaly detection," *Expert Syst. Appl.*, vol. 193, p. 116510, May 2022, doi: 10.1016/j.eswa.2022.116510.

[31] B. A. Tama and S. Lim, "Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation," *Comput. Sci. Rev.*, vol. 39, p. 100357, Feb. 2021, doi: 10.1016/j.cosrev.2020.100357.

[32] G. Folino, F. S. Pisani, and L. Pontieri, "A GP-based ensemble classification framework for time-changing streams of intrusion detection data," *Soft Comput.*, vol. 24, no. 23, Art. no. 23, Dec. 2020, doi: 10.1007/s00500-020-05200-3.

[33] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, Sep. 2017, doi: 10.1016/j.inffus.2017.02.004.

[34] Haibo He, Sheng Chen, Kang Li, and Xin Xu, "Incremental Learning From Stream Data," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1901–1914, Dec. 2011, doi: 10.1109/TNN.2011.2171713.

[35] L. L. Dhirani, E. Armstrong, and T. Newe, "Industrial IoT, Cyber Threats, and Standards Landscape: Evaluation and Roadmap," *Sensors*, vol. 21, no. 11, p. 3901, Jun. 2021, doi: 10.3390/s21113901.

[36] "Infographic: Cybercrime Expected To Skyrocket in Coming Years," Statista Infographics. Accessed: Jun. 25, 2023. [Online]. Available: https://www.statista.com/chart/28878/expected-cost-of-cybercrime-until-2027

[37] P. V. N. Rajeswari, M. Shashi, T. K. Rao, M. Rajya Lakshmi, and L. V. Kiran, "Effective intrusion detection system using concept drifting data stream and support vector machine," *Concurr. Comput. Pract. Exp.*, vol. 34, no. 21, p. e7118, 2022, doi: 10.1002/cpe.7118.

[38] A. S. Iwashita and J. P. Papa, "An Overview on Concept Drift Learning," *IEEE Access*, vol. 7, pp. 1532–1547, 2019, doi: 10.1109/ACCESS.2018.2886026.

[39] A. M. Pasikhan, J. A. Clark, and P. Gope, "Incremental hybrid intrusion detection for 6LoWPAN," *Comput. Secur.*, vol. 135, p. 103447, Dec. 2023, doi: 10.1016/j.cose.2023.103447.

[40] A. Kuppa and N.-A. Le-Khac, "Learn to adapt: Robust drift detection in security domain," *Comput. Electr. Eng.*, vol. 102, p. 108239, Sep. 2022, doi: 10.1016/j.compeleceng.2022.108239.

[41] M. Akbanov, V. G. Vassilakis, and M. D. Logothetis, "WannaCry Ransomware: Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms," *J. Telecommun. Inf. Technol.*, vol. 1, no. 2019, pp. 113–124, Apr. 2019, doi: 10.26636/jtit.2019.130218.

[42] S. Venkatesha, K. R. Reddy, and B. R. Chandavarkar, "Social Engineering Attacks During the COVID-19 Pandemic," *Sn Comput. Sci.*, vol. 2, no. 2, p. 78, 2021, doi: 10.1007/s42979-020-00443-1.

[43] H. Du, Y. Zhang, K. Gang, L. Zhang, and Y.-C. Chen, "Online ensemble learning algorithm for imbalanced data stream," *Appl. Soft Comput.*, vol. 107, p. 107378, Aug. 2021, doi: 10.1016/j.asoc.2021.107378.

[44] R. Xu, Y. Cheng, Z. Liu, Y. Xie, and Y. Yang, "Improved Long Short-Term Memory based anomaly detection with concept drift adaptive method for supporting IoT services," *Future Gener. Comput. Syst.*, vol. 112, pp. 228–242, Nov. 2020, doi: 10.1016/j.future.2020.05.035.

[45] O. Abdul Wahab, "Sustaining the Effectiveness of IoT-Driven Intrusion Detection over Time: Defeating Concept and Data Drifts," Feb. 2021, doi: 10.36227/techrxiv.13669199.v1.

[46] F. Pinagé, E. M. dos Santos, and J. Gama, "A drift detection method based on dynamic classifier selection," *Data Min. Knowl. Discov.*, vol. 34, no. 1, Art. no. 1, Jan. 2020, doi: 10.1007/s10618-019-00656-w.

[47] K. Yin, B. Tang, M. Li, and H. Zhao, "A Multi-Objective Optimization Approach Based on an Enhanced Particle Swarm Optimization Algorithm With Evolutionary Game Theory," *IEEE Access*, vol. 11, pp. 77566–77584, 2023, doi: 10.1109/ACCESS.2023.3298058.

[48] Y. Hu, Y. Zhang, and D. Gong, "Multiobjective Particle Swarm Optimization for Feature Selection With Fuzzy Cost," *IEEE Trans. Cybern.*, vol. 51, no. 2, Art. no. 2, Feb. 2021, doi: 10.1109/TCYB.2020.3015756.

[49] H. Han, Y. Liu, Y. Hou, and J. Qiao, "Multi-modal multi-objective particle swarm optimization with self-adjusting strategy," *Inf. Sci.*, vol. 629, pp. 580–598, Jun. 2023, doi: 10.1016/j.ins.2023.02.019.

[50] C. Khammassi and S. Krichen, "A NSGA2-LR wrapper approach for feature selection in network intrusion detection," *Comput. Netw.*, vol. 172, p. 107183, May 2020, doi: 10.1016/j.comnet.2020.107183.

[51] A. Tiwari and A. Chaturvedi, "A hybrid feature selection approach based on information theory and dynamic butterfly optimization algorithm for data classification," *Expert Syst. Appl.*, vol. 196, p. 116621, Jun. 2022, doi: 10.1016/j.eswa.2022.116621.

[52] Z. Sadeghian, E. Akbari, and H. Nematzadeh, "A hybrid feature selection method based on information theory and binary butterfly optimization algorithm," *Eng. Appl. Artif. Intell.*, vol. 97, p. 104079, Jan. 2021, doi: 10.1016/j.engappai.2020.104079.

[53] B. Ahadzadeh, M. Abdar, F. Safara, A. Khosravi, M. B. Menhaj, and P. N. Suganthan, "SFE: A Simple, Fast and Efficient Feature Selection Algorithm for High-Dimensional Data," *IEEE Trans. Evol. Comput.*, pp. 1–1, 2023, doi: 10.1109/TEVC.2023.3238420.

[54] B. Tran, M. Zhang, and B. Xue, "A PSO based hybrid feature selection algorithm for high-dimensional classification," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2016, pp. 3801–3808. doi: 10.1109/CEC.2016.7744271.

[55] X.-F. Song, Y. Zhang, Y.-N. Guo, X.-Y. Sun, and Y.-L. Wang, "Variable-Size Cooperative Coevolutionary Particle Swarm Optimization for Feature Selection on High-Dimensional Data," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, Art. no. 5, Oct. 2020, doi: 10.1109/TEVC.2020.2968743.

[56] M. Rostami, S. Forouzandeh, K. Berahmand, and M. Soltani, "Integration of multi-objective PSO based feature selection and node centrality for medical datasets," *Genomics*, vol. 112, no. 6, Art. no. 6, Nov. 2020, doi: 10.1016/j.ygeno.2020.07.027.

[57] K. K. Ghosh, R. Guha, S. K. Bera, N. Kumar, and R. Sarkar, "S-shaped versus V-shaped transfer functions for binary Manta ray foraging optimization in feature selection problem," *Neural Comput. Appl.*, vol. 33, no. 17, pp. 11027–11041, Sep. 2021, doi: 10.1007/s00521-020-05560-9.

[58] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System," *IEEE Access*, vol. 7, pp. 94497–94507, 2019, doi: 10.1109/ACCESS.2019.2928048.

[59] S. Wang, L. L. Minku, and X. Yao, "Resampling-Based Ensemble Methods for Online Class Imbalance Learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, Art. no. 5, May 2015, doi: 10.1109/TKDE.2014.2345380.

[60] Z. Yang, S. Al-Dahidi, P. Baraldi, E. Zio, and L. Montelatici, "A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, Art. no. 1, Jan. 2020, doi: 10.1109/TNNLS.2019.2900956.

[61] E. A. K. Zaman, A. Mohamed, and A. Ahmad, "Feature selection for online streaming high-dimensional data: A state-of-the-art review," *Appl. Soft Comput.*, p. 109355, Jul. 2022, doi: 10.1016/j.asoc.2022.109355.

[62] S. Subramani and M. Selvi, “Multi-objective PSO based feature selection for intrusion detection in IoT based wireless sensor networks,” *Optik*, vol. 273, p. 170419, Feb. 2023, doi: 10.1016/j.ijleo.2022.170419.

[63] R. A. Kemmerer and G. Vigna, “Intrusion detection: a brief history and overview,” *Computer*, vol. 35, no. 4, pp. supl27–supl30, Apr. 2002, doi: 10.1109/MC.2002.1012428.

[64] D. E. Denning, “An Intrusion-Detection Model,” *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987, doi: 10.1109/TSE.1987.232894.

[65] K. Ilgun, “USTAT: a real-time intrusion detection system for UNIX,” in *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, USA: IEEE Comput. Soc. Press, 1993, pp. 16–28. doi: 10.1109/RISP.1993.287646.

[66] R. Singh, H. Kumar, R. K. Singla, and R. R. Ketti, “Internet attacks and intrusion detection system: A review of the literature,” *Online Inf. Rev.*, vol. 41, no. 2, pp. 171–184, Apr. 2017, doi: 10.1108/OIR-12-2015-0394.

[67] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, “Mining data streams: a review,” *ACM SIGMOD Rec.*, vol. 34, no. 2, pp. 18–26, Jun. 2005, doi: 10.1145/1083784.1083789.

[68] J. Frank, “Artificial Intelligence and Intrusion Detection: Current and Future Directions,” p. 12.

[69] S. M. Kasongo and Y. Sun, “Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset,” *J. Big Data*, vol. 7, no. 1, Art. no. 1, Dec. 2020, doi: 10.1186/s40537-020-00379-6.

[70] A. Tsymbal, “The problem of concept drift: definitions and related work”.

[71] C. Zhang, D. Jia, L. Wang, W. Wang, F. Liu, and A. Yang, “Comparative research on network intrusion detection methods based on machine learning,” *Comput. Secur.*, vol. 121, p. 102861, Oct. 2022, doi: 10.1016/j.cose.2022.102861.

[72] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, “Learning in Nonstationary Environments: A Survey,” *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, Art. no. 4, Nov. 2015, doi: 10.1109/MCI.2015.2471196.

[73] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, “Building an efficient intrusion detection system based on feature selection and ensemble classifier,” *Comput. Netw.*, vol. 174, p. 107247, Jun. 2020, doi: 10.1016/j.comnet.2020.107247.

[74] E. A. Shams, A. Rizaner, and A. H. Ulusoy, “A novel context-aware feature extraction method for convolutional neural network-based intrusion detection systems,” *Neural Comput. Appl.*, Apr. 2021, doi: 10.1007/s00521-021-05994-9.

[75] J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez, and R. A. Abd-Alhameed, "HIDROID: Prototyping a Behavioral Host-Based Intrusion Detection and Prevention System for Android," *IEEE Access*, vol. 8, pp. 23154–23168, 2020, doi: 10.1109/ACCESS.2020.2969626.

[76] S. Kaur and M. Singh, "Hybrid intrusion detection and signature generation using Deep Recurrent Neural Networks," *Neural Comput. Appl.*, vol. 32, no. 12, Art. no. 12, Jun. 2020, doi: 10.1007/s00521-019-04187-9.

[77] Y. Otoum and A. Nayak, "AS-IDS: Anomaly and Signature Based IDS for the Internet of Things," *J. Netw. Syst. Manag.*, vol. 29, no. 3, Art. no. 3, Jul. 2021, doi: 10.1007/s10922-021-09589-6.

[78] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, no. 1, p. 18, Dec. 2021, doi: 10.1186/s42400-021-00077-7.

[79] S. Bhattacharyya, S. Shimoda, and M. Hayashibe, "A Synergetic Brain-Machine Interfacing Paradigm for Multi-DOF Robot Control," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 46, no. 7, pp. 957–968, Jul. 2016, doi: 10.1109/TSMC.2016.2560532.

[80] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems," *IEEE Access*, vol. 8, pp. doi:165130–165150, 2020,.

[81] G. Falco, C. Caldera, and H. Shrobe, "IIoT Cybersecurity Risk Modeling for SCADA Systems," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4486–4495, Dec. 2018, doi: 10.1109/JIOT.2018.2822842.

[82] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, "Intrusion detection systems for IoT-based smart environments: a survey," *J. Cloud Comput.*, vol. 7, no. 1, Art. no. 1, Dec. 2018, doi: 10.1186/s13677-018-0123-6.

[83] P. Kumar, G. P. Gupta, and R. Tripathi, "An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks," *Comput. Commun.*, vol. 166, pp. 110–124, Jan. 2021, doi: 10.1016/j.comcom.2020.12.003.

[84] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017, doi: 10.1109/JIOT.2017.2683200.

[85] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017, doi: 10.1016/j.jnca.2017.02.009.

[86] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015, doi: 10.1016/j.comnet.2014.11.008.

[87] E. Benkhelifa, T. Welsh, and W. Hamouda, "A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 3496–3509, 2018, doi: 10.1109/COMST.2018.2844742.

[88] S. Hajj *et al.*, "A Critical Review on the Implementation of Static Data Sampling Techniques to Detect Network Attacks," *IEEE Access*, vol. 9, pp. 138903–138938, 2021, doi: 10.1109/ACCESS.2021.3118605.

[89] A. Raut, A. Shivhare, V. K. Chaurasiya, and M. Kumar, "AEDS-IoT: Adaptive clustering-based Event Detection Scheme for IoT data streams," *Internet Things*, vol. 22, p. 100704, Jul. 2023, doi: 10.1016/j.iot.2023.100704.

[90] S. Park, S. Seo, C. Jeong, and J. Kim, "Online eigenvector transformation reflecting concept drift for improving network intrusion detection," *Expert Syst.*, vol. 37, no. 5, Art. no. 5, Oct. 2020, doi: 10.1111/exsy.12477.

[91] Z. Wu, P. Gao, L. Cui, and J. Chen, "An Incremental Learning Method Based on Dynamic Ensemble RVM for Intrusion Detection," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 1, pp. 671–685, Mar. 2022, doi: 10.1109/TNSM.2021.3102388.

[92] E. mahmodi, H. S. Yazdi, and A. G. Bafghi, "A drift aware adaptive method based on minimum uncertainty for anomaly detection in social networking," *Expert Syst. Appl.*, vol. 162, p. 113881, Dec. 2020, doi: 10.1016/j.eswa.2020.113881.

[93] A. Liu, J. Lu, and G. Zhang, "Concept Drift Detection via Equal Intensity k-Means Space Partitioning," *IEEE Trans. Cybern.*, vol. 51, no. 6, Art. no. 6, Jun. 2021, doi: 10.1109/TCYB.2020.2983962.

[94] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under Concept Drift: A Review," *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2018, doi: 10.1109/TKDE.2018.2876857.

[95] R. N. Gemaque, A. F. J. Costa, R. Giusti, and E. M. dos Santos, "An overview of unsupervised drift detection methods," *WIREs Data Min. Knowl. Discov.*, vol. 10, no. 6, p. e1381, 2020, doi: 10.1002/widm.1381.

[96] O. A. Mahdi, E. Pardede, N. Ali, and J. Cao, "Diversity measure as a new drift detection method in data streaming," *Knowl.-Based Syst.*, vol. 191, p. 105227, Mar. 2020, doi: 10.1016/j.knosys.2019.105227.

[97] J. F. Kurian and M. Allali, "Detecting drifts in data streams using Kullback-Leibler (KL) divergence measure for data engineering applications," *J. Data Inf. Manag.*, May 2024, doi: 10.1007/s42488-024-00119-y.

[98] S. Yu, Z. Abraham, H. Wang, M. Shah, Y. Wei, and J. C. Príncipe, "Concept drift detection and adaptation with hierarchical hypothesis testing," *J. Frankl. Inst.*, vol. 356, no. 5, pp. 3187–3215, Mar. 2019, doi: 10.1016/j.jfranklin.2019.01.043.

[99] P. M. Gonçalves, S. G. T. De Carvalho Santos, R. S. M. Barros, and D. C. L. Vieira, "A comparative study on concept drift detectors," *Expert Syst. Appl.*, vol. 41, no. 18, pp. 8144–8156, Dec. 2014, doi: 10.1016/j.eswa.2014.07.019.

[100] S. Sahmoud and H. R. Topcuoglu, "A general framework based on dynamic multi-objective evolutionary algorithms for handling feature drifts on data streams," *Future Gener. Comput. Syst.*, vol. 102, pp. 42–52, Jan. 2020, doi: 10.1016/j.future.2019.07.069.

[101] B. Pishgoo, A. A. Azirani, and B. Raahemi, "A dynamic feature selection and intelligent model serving for hybrid batch-stream processing," *Knowl.-Based Syst.*, vol. 256, p. 109749, Nov. 2022, doi: 10.1016/j.knosys.2022.109749.

[102] P. Bedi, N. Gupta, and V. Jindal, "I-SiamIDS: an improved Siam-IDS for handling class imbalance in network-based intrusion detection systems," *Appl. Intell.*, vol. 51, no. 2, Art. no. 2, Feb. 2021, doi: 10.1007/s10489-020-01886-y.

[103] V. H. Alves Ribeiro and G. Reynoso-Meza, "Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets," *Expert Syst. Appl.*, vol. 147, p. 113232, Jun. 2020, doi: 10.1016/j.eswa.2020.113232.

[104] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[105] P. Soltanzadeh and M. Hashemzadeh, "RCSMOTE: Range-Controlled synthetic minority over-sampling technique for handling the class imbalance problem," *Inf. Sci.*, vol. 542, pp. 92–111, Jan. 2021, doi: 10.1016/j.ins.2020.07.014.

[106] S. Thudumu, P. Branch, J. Jin, and J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," *J. Big Data*, vol. 7, no. 1, Art. no. 1, Dec. 2020, doi: 10.1186/s40537-020-00320-x.

[107] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data," *IEEE Access*, vol. 8, pp. 58194–58205, 2020, doi: 10.1109/ACCESS.2020.2982544.

[108] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Inf. Fusion*, vol. 59, pp. 44–58, Jul. 2020, doi: 10.1016/j.inffus.2020.01.005.

[109] J. Tang, S. Alelyani, and H. Liu, "Feature Selection for Classification: A Review," p. 33, 2014 Data classification: Algorithms and applications (2014): 37.

[110] S. Karasu, A. Altan, S. Bekiros, and W. Ahmad, "A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series," *Energy*, vol. 212, p. 118750, Dec. 2020, doi: 10.1016/j.energy.2020.118750.

[111] G. Wei, J. Zhao, Y. Feng, A. He, and J. Yu, "A novel hybrid feature selection method based on dynamic feature importance," *Appl. Soft Comput.*, vol. 93, p. 106337, Aug. 2020, doi: 10.1016/j.asoc.2020.106337.

[112] F. Mokbal, W. Dan, M. Osman, Y. Ping, and S. Alsamhi, "An Efficient Intrusion Detection Framework Based on Embedding Feature Selection and Ensemble Learning Technique," *Int. Arab J. Inf. Technol.*, vol. 19, no. 2, Art. no. 2, 2022, doi: 10.34028/iajit/19/2/11.

[113] F. C. Schuartz, M. Fonseca, and A. Munaretto, “A Distributed Platform for Intrusion Detection System Using Data Stream Mining in a Big Data Environment,” in *2022 6th Cyber Security in Networking Conference (CSNet)*, Rio de Janeiro, Brazil: IEEE, Oct. 2022, pp. 1–7. doi: 10.1109/CSNet56116.2022.9955598.

[114] E. Gyamfi and A. D. Jurcut, “Novel Online Network Intrusion Detection System for Industrial IoT Based on OI-SVDD and AS-ELM,” *IEEE Internet Things J.*, vol. 10, no. 5, pp. 3827–3839, Mar. 2023, doi: 10.1109/JIOT.2022.3172393.

[115] H. Alazzam, A. Sharieh, and K. E. Sabri, “A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer,” *Expert Syst. Appl.*, vol. 148, p. 113249, Jun. 2020, doi: 10.1016/j.eswa.2020.113249.

[116] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, “A Survey on Ensemble Learning for Data Stream Classification,” *ACM Comput. Surv.*, vol. 50, no. 2, Art. no. 2, Jun. 2017, doi: 10.1145/3054925.

[117] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, “Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection,” *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 13, no. 5, Art. no. 5, Sep. 2016, doi: 10.1109/TCBB.2015.2478454.

[118] X. Wang, H. Wang, and D. Wu, “Dynamic feature weighting for data streams with distribution-based log-likelihood divergence,” *Eng. Appl. Artif. Intell.*, vol. 107, p. 104509, Jan. 2022, doi: 10.1016/j.engappai.2021.104509.

[119] N. Kunhare, R. Tiwari, and J. Dhar, “Particle swarm optimization and feature selection for intrusion detection system,” *Sādhanā*, vol. 45, no. 1, p. 109, Dec. 2020, doi: 10.1007/s12046-020-1308-5.

[120] B. Xue, M. Zhang, and W. N. Browne, “Particle Swarm Optimization for Feature Selection in Classification: A Multi-Objective Approach,” *IEEE Trans. Cybern.*, vol. 43, no. 6, Art. no. 6, Dec. 2013, doi: 10.1109/TSMCB.2012.2227469.

[121] R. A. Disha and S. Waheed, “Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique,” *Cybersecurity*, vol. 5, no. 1, p. 1, Dec. 2022, doi: 10.1186/s42400-021-00103-8.

[122] N. Wichitaksorn, Y. Kang, and F. Zhang, “Random feature selection using random subspace logistic regression,” *Expert Syst. Appl.*, vol. 217, p. 119535, May 2023, doi: 10.1016/j.eswa.2023.119535.

[123] H. Jeon and S. Oh, “Hybrid-Recursive Feature Elimination for Efficient Feature Selection,” *Appl. Sci.*, vol. 10, no. 9, Art. no. 9, Jan. 2020, doi: 10.3390/app10093211.

[124] Q. Al-Tashi, S. J. Abdulkadir, H. M. Rais, S. Mirjalili, and H. Alhussian, “Approaches to Multi-Objective Feature Selection: A Systematic Literature Review,” *IEEE Access*, vol. 8, pp. 125076–125096, 2020, doi: 10.1109/ACCESS.2020.3007291.

[125] A. Aboud *et al.*, "DPb-MOPSO: A Dynamic Pareto bi-level Multi-objective Particle Swarm Optimization Algorithm," *Appl. Soft Comput.*, vol. 129, p. 109622, Nov. 2022, doi: 10.1016/j.asoc.2022.109622.

[126] S. Petchrompo, D. W. Coit, A. Brintrup, A. Wannakrairot, and A. K. Parlikad, "A review of Pareto pruning methods for multi-objective optimization," *Comput. Ind. Eng.*, vol. 167, p. 108022, May 2022, doi: 10.1016/j.cie.2022.108022.

[127] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Nov. 1995, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.

[128] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic Algorithms: A Comprehensive Review," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, Elsevier, 2018, pp. 185–231. doi: 10.1016/B978-0-12-813314-9.00010-4.

[129] W. H. Bangyal, A. Hameed, W. Alosaimi, and H. Alyami, "A New Initialization Approach in Particle Swarm Optimization for Global Optimization Problems," *Comput. Intell. Neurosci.*, vol. 2021, pp. 1–17, May 2021, doi: 10.1155/2021/6628889.

[130] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

[131] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014, doi: 10.1109/TEVC.2013.2281535.

[132] S. Verma, M. Pant, and V. Snasel, "A Comprehensive Review on NSGA-II for Multi-Objective Combinatorial Optimization Problems," *IEEE Access*, vol. 9, pp. 57757–57791, 2021, doi: 10.1109/ACCESS.2021.3070634.

[133] Qingfu Zhang and Hui Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007, doi: 10.1109/TEVC.2007.892759.

[134] C. Wu and W. Li, "Enhancing intrusion detection with feature selection and neural network," *Int. J. Intell. Syst.*, vol. 36, no. 7, pp. 3087–3105, Jul. 2021, doi: 10.1002/int.22397.

[135] G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel Chaotic Equilibrium Optimizer Algorithm with S-shaped and V-shaped transfer functions for feature selection," *J. Ambient Intell. Humaniz. Comput.*, vol. 13, no. 6, pp. 3137–3162, Jun. 2022, doi: 10.1007/s12652-021-03151-7.

[136] V. Chang *et al.*, "A Survey on Intrusion Detection Systems for Fog and Cloud Computing," *Future Internet*, vol. 14, no. 3, p. 89, Mar. 2022, doi: 10.3390/fi14030089.

[137] R. E. Schapire, "The strength of weak learnability" Machine learning 5 (1990): 197-227.

[138] J. Gu, L. Wang, H. Wang, and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," *Comput. Secur.*, vol. 86, pp. 53–62, Sep. 2019, doi: 10.1016/j.cose.2019.05.022.

[139] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," *J. Netw. Comput. Appl.*, vol. 66, pp. 1–16, May 2016, doi: 10.1016/j.jnca.2016.03.011.

[140] M. A. Shyaa, Z. Zainol, R. Abdullah, M. Anbar, L. Alzubaidi, and J. Santamaría, "Enhanced Intrusion Detection with Data Stream Classification and Concept Drift Guided by the Incremental Learning Genetic Programming Combiner," *Sensors*, vol. 23, no. 7, p. 3736, Apr. 2023, doi: 10.3390/s23073736.

[141] J. Wilson, S. Chaudhury, and B. Lall, "Homogeneous–Heterogeneous Hybrid Ensemble for concept-drift adaptation," *Neurocomputing*, vol. 557, p. 126741, Nov. 2023, doi: 10.1016/j.neucom.2023.126741.

[142] A. Feitosa Neto and A. M. P. Canuto, "EOCD: An ensemble optimization approach for concept drift applications," *Inf. Sci.*, vol. 561, pp. 81–100, Jun. 2021, doi: 10.1016/j.ins.2021.01.051.

[143] D. Ramos, D. Carneiro, and P. Novais, "Using a Genetic Algorithm to optimize a stacking ensemble in data streaming scenarios," *AI Commun.*, vol. 33, no. 1, pp. 27–40, Aug. 2020, doi: 10.3233/AIC-200648.

[144] G. Folino, F. S. Pisani, and L. Pontieri, "A Cybersecurity Framework for Classifying Non Stationary Data Streams Exploiting Genetic Programming and Ensemble Learning," in *Numerical Computations: Theory and Algorithms*, Y. D. Sergeyev and D. E. Kvasov, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 269–277. doi: 10.1007/978-3-030-39081-5_24.

[145] A. Cano and B. Krawczyk, "Evolving rule-based classifiers with genetic programming on GPUs for drifting data streams," *Pattern Recognit.*, vol. 87, pp. 248–268, Mar. 2019, doi: 10.1016/j.patcog.2018.10.024.

[146] G. Folino, F. S. Pisani, and P. Sabatino, "An Incremental Ensemble Evolved by using Genetic Programming to Efficiently Detect Drifts in Cyber Security Datasets," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, Denver Colorado USA: ACM, Jul. 2016, pp. 1103–1110. doi: 10.1145/2908961.2931682.

[147] S. A. R. Zaidi, "Nearest neighbour methods and their applications in design of 5G & beyond wireless networks," *ICT Express*, vol. 7, no. 4, pp. 414–420, Dec. 2021, doi: 10.1016/j.icte.2021.01.003.

[148] S. Li, K. Zhang, Q. Chen, S. Wang, and S. Zhang, "Feature Selection for High Dimensional Data Using Weighted K-Nearest Neighbors and Genetic Algorithm," *IEEE Access*, vol. 8, pp. 139512–139528, 2020, doi: 10.1109/ACCESS.2020.3012768.

[149] H. Belyadi and A. Haghighat, "Chapter 5 - Supervised learning," in *Machine Learning Guide for Oil and Gas Using Python*, H. Belyadi and A. Haghighat, Eds., Gulf Professional Publishing, 2021, pp. 169–295. doi: 10.1016/B978-0-12-821929-4.00004-4.

[150] M. Bansal, A. Goyal, and A. Choudhary, "A comparative analysis of K-Nearest Neighbor, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning," *Decis. Anal. J.*, vol. 3, p. 100071, Jun. 2022, doi: 10.1016/j.dajour.2022.100071.

[151] O. Peretz, M. Koren, and O. Koren, "Naive Bayes classifier – An ensemble procedure for recall and precision enrichment," *Eng. Appl. Artif. Intell.*, vol. 136, p. 108972, Oct. 2024, doi: 10.1016/j.engappai.2024.108972.

[152] X. Xie, Z. Jin, J. Wang, L. Yang, Y. Lu, and T. Li, "Confidence guided anomaly detection model for anti-concept drift in dynamic logs," *J. Netw. Comput. Appl.*, vol. 162, p. 102659, Jul. 2020, doi: 10.1016/j.jnca.2020.102659.

[153] M. Jain and G. Kaur, "Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data," *Clust. Comput.*, Feb. 2021, doi: 10.1007/s10586-021-03249-9.

[154] A. K. Mananayaka and S. S. Chung, "Network Intrusion Detection with Two-Phased Hybrid Ensemble Learning and Automatic Feature Selection," *IEEE Access*, vol. 11, pp. 45154–45167, 2023, doi: 10.1109/ACCESS.2023.3274474.

[155] A. J. Rabash, M. Z. A. Nazri, A. Shapii, and M. K. Hasan, "Non-Dominated Sorting Genetic Algorithm-Based Dynamic Feature Selection for Intrusion Detection System," *IEEE Access*, vol. 11, pp. 125080–125093, 2023, doi: 10.1109/ACCESS.2023.3328395.

[156] B. Li, Y. Wang, K. Xu, L. Cheng, and Z. Qin, "DFAID: Density-aware and feature-deviated active intrusion detection over network traffic streams," *Comput. Secur.*, vol. 118, p. 102719, Jul. 2022, doi: 10.1016/j.cose.2022.102719.

[157] Y. Li, T. Qin, Y. Huang, J. Lan, Z. Liang, and T. Geng, "HDFEF: A hierarchical and dynamic feature extraction framework for intrusion detection systems," *Comput. Secur.*, vol. 121, p. 102842, Oct. 2022, doi: 10.1016/j.cose.2022.102842.

[158] F. K. Örs and A. Levi, "Data driven intrusion detection for 6LoWPAN based IoT systems," *Ad Hoc Netw.*, vol. 143, p. 103120, Apr. 2023, doi: 10.1016/j.adhoc.2023.103120.

[159] X. Liu, P. Zhang, H. Fang, and Y. Zhou, "Multi-Objective Reactive Power Optimization Based on Improved Particle Swarm Optimization With ε-Greedy Strategy and Pareto Archive Algorithm," *IEEE Access*, vol. 9, pp. 65650–65659, 2021, doi: 10.1109/ACCESS.2021.3075777.

[160] H. R. R. Zaman and F. S. Gharehchopogh, "An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization

problems," *Eng. Comput.*, vol. 38, no. S4, pp. 2797–2831, Oct. 2022, doi: 10.1007/s00366-021-01431-6.

[161] V. Trivedi, P. Varshney, and M. Ramteke, "A simplified multi-objective particle swarm optimization algorithm," *Swarm Intell.*, vol. 14, no. 2, Art. no. 2, Jun. 2020, doi: 10.1007/s11721-019-00170-1.

[162] K. Alkebsi and W. Du, "A Fast Multi-Objective Particle Swarm Optimization Algorithm Based on a New Archive Updating Mechanism," *IEEE Access*, vol. 8, pp. 124734–124754, 2020, doi: 10.1109/ACCESS.2020.3007846.

[163] D. C. Valencia-Rodríguez and C. A. Coello Coello, "Influence of the number of connections between particles in the performance of a multi-objective particle swarm optimizer," *Swarm Evol. Comput.*, vol. 77, p. 101231, Mar. 2023, doi: 10.1016/j.swevo.2023.101231.

[164] H. Han, L. Zhang, A. Yinga, and J. Qiao, "Adaptive multiple selection strategy for multi-objective particle swarm optimization," *Inf. Sci.*, vol. 624, pp. 235–251, May 2023, doi: 10.1016/j.ins.2022.12.077.

[165] J. Yang, J. Zou, S. Yang, Y. Hu, J. Zheng, and Y. Liu, "A particle swarm algorithm based on the dual search strategy for dynamic multi-objective optimization," *Swarm Evol. Comput.*, p. 101385, Aug. 2023, doi: 10.1016/j.swevo.2023.101385.

[166] Y. Lu, B. Li, S. Liu, and A. Zhou, "A Population Cooperation based Particle Swarm Optimization algorithm for large-scale multi-objective optimization," *Swarm Evol. Comput.*, vol. 83, p. 101377, Dec. 2023, doi: 10.1016/j.swevo.2023.101377.

[167] Y. Cui, X. Meng, and J. Qiao, "A multi-objective particle swarm optimization algorithm based on two-archive mechanism," *Appl. Soft Comput.*, vol. 119, p. 108532, Apr. 2022, doi: 10.1016/j.asoc.2022.108532.

[168] Y. Li, Y. Zhang, and W. Hu, "Adaptive multi-objective particle swarm optimization based on virtual Pareto front," *Inf. Sci.*, vol. 625, pp. 206–236, May 2023, doi: 10.1016/j.ins.2022.12.079.

[169] A. Madani, A. Engelbrecht, and B. Ombuki-Berman, "Cooperative coevolutionary multi-guide particle swarm optimization algorithm for large-scale multi-objective optimization problems," *Swarm Evol. Comput.*, vol. 78, p. 101262, Apr. 2023, doi: 10.1016/j.swevo.2023.101262.

[170] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving Deep Convolutional Neural Networks by Variable-Length Particle Swarm Optimization for Image Classification," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2018, pp. 1–8. doi: 10.1109/CEC.2018.8477735.

[171] Y. Xue, Y. Tang, X. Xu, J. Liang, and F. Neri, "Multi-objective Feature Selection with Missing Data in Classification," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 6, no. 2, pp. 355–364, Apr. 2022, doi: 10.1109/TETCI.2021.3074147.

[172] TA Alamiedy, M. Anbar, Z. Alqattan, and Qusay M. Alzubi1, "Anomaly-based intrusion detection system using multi-objective grey wolf optimisation algorithm," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 9, pp. 3735–3756, Sep. 2020, doi: 10.1007/s12652-019-01569-8.

[173] Y. Zhang, D. Gong, X. Gao, T. Tian, and X. Sun, "Binary differential evolution with self-learning for multi-objective feature selection," *Inf. Sci.*, vol. 507, pp. 67–85, Jan. 2020, doi: 10.1016/j.ins.2019.08.040.

[174] Q. Al-Tashi, S. J. Abdul Kadir, H. M. Rais, S. Mirjalili, and H. Alhussian, "Binary Optimization Using Hybrid Grey Wolf Optimization for Feature Selection," *IEEE Access*, vol. 7, pp. 39496–39508, 2019, doi: 10.1109/ACCESS.2019.2906757.

[175] M. Tubishat *et al.*, "Dynamic Salp swarm algorithm for feature selection," *Expert Syst. Appl.*, vol. 164, p. 113873, Feb. 2021, doi: 10.1016/j.eswa.2020.113873.

[176] E.-S. M. El-Kenawy, M. M. Eid, M. Saber, and A. Ibrahim, "MbGWO-SFS: Modified Binary Grey Wolf Optimizer Based on Stochastic Fractal Search for Feature Selection," *IEEE Access*, vol. 8, pp. 107635–107649, 2020, doi: 10.1109/ACCESS.2020.3001151.

[177] B. Abdollahzadeh and F. S. Gharehchopogh, "A multi-objective optimization algorithm for feature selection problems," *Eng. Comput.*, Mar. 2021, doi: 10.1007/s00366-021-01369-9.

[178] Y. Zhang, S. Cheng, Y. Shi, D. Gong, and X. Zhao, "Cost-sensitive feature selection using two-archive multi-objective artificial bee colony algorithm," *Expert Syst. Appl.*, vol. 137, pp. 46–58, Dec. 2019, doi: 10.1016/j.eswa.2019.06.044.

[179] Q. Al-Tashi *et al.*, "Binary Multi-Objective Grey Wolf Optimizer for Feature Selection in Classification," *IEEE Access*, vol. 8, pp. 106247–106263, 2020, doi: 10.1109/ACCESS.2020.3000040.

[180] M. Amoozegar and B. Minaei-Bidgoli, "Optimizing multi-objective PSO based feature selection method using a feature elitism mechanism," *Expert Syst. Appl.*, vol. 113, pp. 499–514, Dec. 2018, doi: 10.1016/j.eswa.2018.07.013.

[181] S. Arora and P. Anand, "Binary butterfly optimization approaches for feature selection," *Expert Syst. Appl.*, vol. 116, pp. 147–160, Feb. 2019, doi: https://doi.org/10.1016/j.eswa.2018.08.051.

[182] I. Souiden, M. N. Omri, and Z. Brahmi, "A survey of outlier detection in high dimensional data streams," *Comput. Sci. Rev.*, vol. 44, p. 100463, May 2022, doi: 10.1016/j.cosrev.2022.100463.

[183] Z. Yang *et al.*, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Comput. Secur.*, vol. 116, p. 102675, May 2022, doi: 10.1016/j.cose.2022.102675.

[184] L. Matijević, M. Đurasević, and D. Jakobović, "A Variable Neighborhood Search Method with a Tabu List and Local Search for Optimizing Routing in Trucks in Maritime Ports," *Mathematics*, vol. 11, no. 17, p. 3740, Aug. 2023, doi: 10.3390/math11173740.

[185] M. Baena-Garcıa, R. Gavalda, and R. Morales-Bueno, “Early Drift Detection Method” Fourth international workshop on knowledge discovery from data streams. Vol. 6. 2006.

[186] H. Guan, Y. Zhang, M. Xian, H. D. Cheng, and X. Tang, “SMOTE-WENN: Solving class imbalance and small sample problems by oversampling and distance scaling,” *Appl. Intell.*, vol. 51, no. 3, pp. 1394–1409, Mar. 2021, doi: 10.1007/s10489-020-01852-8.

[187] R. Blagus and L. Lusa, “SMOTE for high-dimensional class-imbalanced data,” *BMC Bioinformatics*, vol. 14, no. 1, p. 106, Dec. 2013, doi: 10.1186/1471-2105-14-106.

[188] J. H. Joloudari, A. Marefat, M. A. Nematollahi, S. S. Oyelere, and S. Hussain, “Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks,” *Appl. Sci.*, vol. 13, no. 6, p. 4006, Mar. 2023, doi: 10.3390/app13064006.

[189] D. Nallaperuma *et al.*, “Online Incremental Machine Learning Platform for Big Data-Driven Smart Traffic Management,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, Art. no. 12, Dec. 2019, doi: 10.1109/TITS.2019.2924883.

[190] M. A. Sindhu and K. Meinke, “IDS: An Incremental Learning Algorithm for Finite Automata,” 2012.

[191] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, “A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 4, Art. no. 4, Jul. 2012, doi: 10.1109/TSMCC.2011.2161285.

[192] M. Khairy, T. M. Mahmoud, and T. Abd-El-Hafeez, “The effect of rebalancing techniques on the classification performance in cyberbullying datasets,” *Neural Comput. Appl.*, vol. 36, no. 3, pp. 1049–1065, Jan. 2024, doi: 10.1007/s00521-023-09084-w.

[193] A. Ferriyan, A. H. Thamrin, K. Takeda, and J. Murai, “Generating Network Intrusion Detection Dataset Based on Real and Encrypted Synthetic Attack Traffic,” *Appl. Sci.*, vol. 11, no. 17, p. 7868, Aug. 2021, doi: 10.3390/app11177868.

[194] M. Zhang, H. Wang, Z. Cui, and J. Chen, “Hybrid multi-objective cuckoo search with dynamical local search,” *Memetic Comput.*, vol. 10, no. 2, pp. 199–208, Jun. 2018, doi: 10.1007/s12293-017-0237-2.

[195] F. Hosseini, F. S. Gharehchopogh, and M. Masdari, “A Botnet Detection in IoT Using a Hybrid Multi-objective Optimization Algorithm,” *New Gener. Comput.*, vol. 40, no. 3, pp. 809–843, Sep. 2022, doi: 10.1007/s00354-022-00188-w.

[196] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, “A Survey on Machine Learning Techniques for Cyber Security in the Last Decade,” *IEEE Access*, vol. 8, pp. 222310–222354, 2020, doi: 10.1109/ACCESS.2020.3041951.

[197] A. M. Jubair, R. Hassan, A. H. M. Aman, and H. Sallehudin, “Social class particle swarm optimization for variable-length Wireless Sensor Network Deployment,” *Appl. Soft Comput.*, vol. 113, p. 107926, Dec. 2021, doi: 10.1016/j.asoc.2021.107926.

[198] F. Rezaei and H. R. Safavi, "f-MOPSO/Div: an improved extreme-point-based multi-objective PSO algorithm applied to a socio-economic-environmental conjunctive water use problem," *Environ. Monit. Assess.*, vol. 192, no. 12, p. 767, Dec. 2020, doi: 10.1007/s10661-020-08727-y.

[199] J. Hu *et al.*, "An integrated classification model for incremental learning," *Multimed. Tools Appl.*, vol. 80, no. 11, pp. 17275–17290, May 2021, doi: 10.1007/s11042-020-10070-w.

[200] M. Baei and V. Terzic, "Optimal design of dampers in seismic applications utilizing the MOPSO algorithm," *Front. Built Environ.*, vol. 8, p. 1040129, Oct. 2022, doi: 10.3389/fbuil.2022.1040129.

[201] V. Bolón-Canedo and A. Alonso-Betanzos, "Ensembles for feature selection: A review and future trends," *Information fusion 52 (2019): 1-12*.

[202] N. V. Sharma, "An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers," *Microprocessors and Microsystems 85 (2021): 104293*.

[203] Z. Khandezamin, "Detection and classification of breast cancer using logistic regression feature selection and GMDH classifier," *Journal of Biomedical Informatics 111 (2020): 103591*.

[204] M. G. M. Abdolrasol *et al.*, "Artificial Neural Networks Based Optimization Techniques: A Review," Electronics 10, no. 21 (2021): 2689.

[205] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Trans. NEURAL Netw. Learn. Syst.*.

[206] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building Auto-Encoder Intrusion Detection System based on random forest feature selection," *Comput. Secur.*, vol. 95, p. 101851, Aug. 2020, doi: 10.1016/j.cose.2020.101851.

[207] J. A. Nuh, T. W. Koh, S. Baharom, M. H. Osman, and S. N. Kew, "Performance Evaluation Metrics for Multi-Objective Evolutionary Algorithms in Search-Based Software Engineering: Systematic Literature Review," *Appl. Sci.*, vol. 11, no. 7, p. 3117, Mar. 2021, doi: 10.3390/app11073117.

[208] J. D. Knowles and D. W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, Jun. 2000, doi: 10.1162/106365600568167.

[209] J. Bader and E. Zitzler, "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, Mar. 2011, doi: 10.1162/EVCO_a_00009.

[210] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms — A comparative case study," in *Parallel Problem Solving from Nature — PPSN V*, vol. 1498, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds., in Lecture Notes

in Computer Science, vol. 1498. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 292–301. doi: 10.1007/BFb0056872.

[211] Robert Nshimirimana, "A multi-objective particle swarm for constraint and unconstrained problems," 2021.

[212] R. Fernandes and N. Lopes, "Network Intrusion Detection Packet Classification with the HIKARI-2021 Dataset: a study on ML Algorithms," in *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*, Jun. 2022, pp. 1–5. doi: 10.1109/ISDFS55398.2022.9800807.

[213] Q. U. Ain, H. Al-Sahaf, B. Xue, and M. Zhang, "A genetic programming approach to feature construction for ensemble learning in skin cancer detection," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, Cancún Mexico: ACM, Jun. 2020, pp. 1186–1194. doi: 10.1145/3377930.3390228.

[214] Y. Hu, Y. Zhang, and D. Gong, "Multiobjective Particle Swarm Optimization for Feature Selection with Fuzzy Cost," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 874–888, Feb. 2021, doi: 10.1109/TCYB.2020.3015756.

LIST OF PUBLICATIONS

M. S. Noori, R. K. Z. Sahbudin, A. Sali, and F. Hashim, "Multi-Objective Multi-Exemplar Particle Swarm Optimization Algorithm with Local Awareness," IEEE Access, pp. 1–1, 2024, doi: 10.1109/ACCESS.2024.3426104.

M. S. Noori, R. K. Z. Sahbudin, A. Sali, and F. Hashim, "Feature Drift Aware for Intrusion Detection System Using Developed Variable Length Particle Swarm Optimization in Data Stream," IEEE Access, vol. 11, pp. 128596–128617, 2023, doi: 10.1109/ACCESS.2023.3333000.

M. S. Noori, R. K. Z. Sahbudin, M. S. Abood, and M. M. Hamdi, "A Performance Evaluation of Voice over IP Protocols (SIP and H.323) in Wireless Network," p. 11.

M. S. Noori, R. K. Z. Sahbudin, M. S. Abood, M. M. Hamdi, and I. K. Thajeel, "Study the effect of the Stimulated Raman Scattering (SRS) in optical fiber-based on OptiSystem," in 2021 International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE), Nov. 2021, pp. 1–5. doi: 10.1109/ITSS-IoE53029.2021.9615259.