UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUSTAVO RIBEIRO KREMER

Embedded Weights Multi-Objective Feature Selection: A Novel Hybrid Method for Sparse High-Dimensional Data

Thesis presented in partial fulfillment of the requirements for the degree of Master of Computer Science

Advisor: Prof. Dr. Marcio Dorn

Porto Alegre
August 2025

CIP — CATALOGING-IN-PUBLICATION

Kremer, Gustavo Ribeiro

Embedded Weights Multi-Objective Feature Selection: A Novel Hybrid Method for Sparse High-Dimensional Data / Gustavo Ribeiro Kremer. – Porto Alegre: PPGC da UFRGS, 2025.

75 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2025. Advisor: Marcio Dorn.

1. Feature Selection. 2. Multi-Objective. 3. Sparse Problems. 4. NSGA-II. I. Dorn, Marcio. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Prof[a]. Marcia Barbosa
Vice-Reitor: Prof. Pedro Costa
Pró-Reitora de Pós-Graduação: Prof[a]. Claudia Wasserman
Diretor do Instituto de Informática: Prof. Luciano Paschoal Gaspary
Coordenador do PPGC: Prof. Gabriel Luca Nazar
Bibliotecário-chefe do Instituto de Informática: Alexsander Borges Ribeiro

“É justamente a possibilidade de realizar um sonho que torna a vida interessante”

— O Alquimista, Paulo Coelho

AGRADECIMENTOS

Os agradecimentos são minha parte favorita dos trabalhos acadêmicos, vejo essa seção como um espaço de expressão mais livre. De certa forma, é como se fosse possível abrir uma brecha no texto pra espiar o autor por trás da obra. Forçando um pouco, dá até para tentar deduzir como foi desenvolvimento do trabalho, afinal, os agradecimentos são um reflexo de quem estava lá dando suporte enquanto as coisas aconteciam.

Agradeço ao governo federal pelo investimento em ciência no Brasil e por manter a instituição pública, gratuita e de qualidade na qual cursei graduação e mestrado. Agradeço também as agências de fomento Capes/CNPQ, por fornecerem a mim e aos meus colegas as bolsas de auxílio permanência, que foram responsáveis por me manter financeiramente ao longo dos últimos dois anos.

Agradeço à minha família por estar lá nos momentos em que foi necessária, e por ter me dado suporte ao longo dessa jornada acadêmica, desde o começo, até a graduação e a pós.

Agradeço também aos meus amigos e colegas, muitos os quais conquistei na graduação ou até antes, e que tem me acompanhado desde então. Para ser justo, cometerei a injustiça de não citar nomes dessa vez, afinal a rede é grande e muitas pessoas se envolveram de diferentes formas com o meu processo. Tampouco citarei os nomes dos grupos, porque isso é brega. Agradeço de forma especial àqueles que me fizeram lembrar porquê estou fazendo as coisas, serviram de escuta, ajudaram meu cérebro a funcionar e me motivaram mandando eu calar a boca e escrever o texto.

ABSTRACT

Feature selection approaches are an essential step in most machine learning pipelines, with benefits that go far beyond just improvements in classifier accuracy. Multi-objective approaches are typically chosen to minimize both the number of features and classifier error. These often combine heuristic search methods with machine learning algorithms to create wrapper methods. Wrappers, in turn, can be combined with filters or embedded methods, creating hybrid approaches. However, these methods struggle to scale effectively, particularly in high-dimensional regimes, and are typically limited to datasets with fewer than 2000 features. To address this, we propose the Embedded Weights Multi-Objective Feature Selection (EW-MOFS) algorithm, which consists of a new multi-objective feature selection hybrid method that combines a vector of weights from an embedded technique to guide and optimize the wrapper search space. The proposed method brings new genetic operators specifically designed to deal with highly sparse data, addressing scalability vulnerabilities found in current methods. We evaluate our method on 12 open datasets from Machine Learning repositories and 8 high-dimensional datasets from CuMiDa, which contain up to 54,000 features. EW-MOFS consistently achieves Pareto-fronts with equal or greater hypervolume than competing algorithms in most high-dimensional feature selection tasks.

Keywords: Feature Selection. Multi-Objective. Sparse Problems. NSGA-II.

Embedded Weights Multi-Objective Feature Selection: Um Novo Método Híbrido para Problemas Esparsos de Alta Dimensionalidade

RESUMO

As abordagens de seleção de features são uma etapa essencial na maioria dos pipelines de aprendizado de máquina, com benefícios que vão muito além de apenas melhorias na acurácia do classificador. Geralmente, as abordagens multiobjetivo são escolhidas para minimizar, simultaneamente, o número de features e o erro do classificador. Essas abordagens geralmente combinam métodos de busca heurística com algoritmos de aprendizado de máquina para formar métodos do tipo wrapper. Os wrappers, por sua vez, podem ser combinados com métodos de filtro ou embedded, criando abordagens híbridas. No entanto, esses métodos têm dificuldade para escalar de forma eficaz, principalmente em regimes de alta dimensão, e geralmente são limitados a conjuntos de dados com menos de 2.000 features. Para resolver isso, propomos o algoritmo Embedded Weights Multi-Objective Feature Selection (EW-MOFS), que consiste em um novo método híbrido de seleção de features multiobjetivo que combina um vetor de pesos de uma técnica embedded para orientar e otimizar o espaço de busca do wrapper. O método proposto traz novos operadores genéticos projetados especificamente para lidar com dados de alta esparsidade, abordando as vulnerabilidades de escalabilidade encontradas nos métodos atuais. Avaliamos nosso método em 12 conjuntos de dados de repositórios abertos de Machine Learning e 8 conjuntos de dados de alta dimensão do CuMiDa, que contêm até 54.000 features. O EW-MOFS atinge consistentemente frentes de Pareto com hipervolume igual ou maior do que os algoritmos comparados na maioria dos problemas de seleção de features, principalmente em conjuntos de dados de alta dimensão.

Palavras-chave: Seleção de Features. Multiobjetivo. Problemas Esparsos. NSGA-II.

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
EA	Evolutionary Algorithm
HV	Hypervolume
LLM	Large Language Model
NSGA	Non-dominated Sorting Genetic Algorithm
PCA	Principal Component Analysis
RF	Random Forest
SPEA	Strength Pareto Evolutionary Algorithm
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding

LIST OF FIGURES

Figure 2.1 Examples of K-Fold cross-validation plotted. Source: Adapted from the scikit-learn website. 24
Figure 2.2 Hypervolume visually represented. Source: The author 26
Figure 2.3 Non-dominated and Crowding distance visual explained. Source: (Blank; Deb, 2020)....27

Figure 4.1 Flowchart of the EW-MOFS algorithm. 41
Figure 4.2 F1-score for number of features selected, from an artificially made problem.43

Figure 6.1 Representative Mean Pareto Fronts for EW-MOFS and baselines: two with almost no difference (top), two where EW-MOFS gets outperformed (middle), and two where EW-MOFS outperforms (bottom) cases. 54
Figure 6.2 Convergence values obtained during the execution algorithms for two representative datasets....58
Figure 6.3 Average number of features selected per run for each algorithm and each dataset 61
Figure 6.4 Average execution times (seconds) for each algorithm on each dataset.....63

Figure A.1 Mean Pareto Fronts obtained after executing the six algorithms on the twenty datasets. 71
Figure A.2 Hypervolume values obtained during the execution algorithm on the twenty datasets. 72

LIST OF TABLES

Table 3.1 Comparison between the related work and the EW-MOFS. 38

Table 5.1 Parameters used to evaluate each algorithm. 50
Table 5.2 Dataset used for benchmarking. NS stands for the number of samples, NF stands for the number of features, and NC stands for the number of classes. 50

Table 6.1 Final hypervolumes values obtained after the executions of the six algorithms on the twenty datasets, each cell shows [Hypervolume] [Standard Deviation] [Wilcoxon Test Result]. In bold are the cases where EW-MOFS outperformed the other algorithms. Other algorithms are also in bold if they matched the performance. 57
Table 6.2 Initial solutions' hypervolume values obtained at the start of the executions of the six algorithms on the twenty datasets, each cell shows [Hypervolume] [Standard Deviation] [Wilcoxon Test Result]. In bold are the cases where EW-MOFS outperformed the other algorithms. Other algorithms are also in bold if they matched the performance. 57
Table 6.3 Average number of features selected per run for each dataset with each algorithm. 59
Table 6.4 Pre-processing times (in seconds) of the six algorithms on the twenty datasets. Less is better. 61
Table 6.5 Average execution times (in seconds) of the six algorithms on the twenty datasets. Less is better. 62

CONTENTS

1 INTRODUCTION **12**
1.1 Overview **12**
1.2 Problem Statement **14**
1.3 Hypothesis **15**
1.4 Research Question **15**
1.5 Contribution of This Work **15**
1.6 Organization of the Dissertation **15**
2 OVERVIEW **17**
2.1 Machine Learning **17**
2.2 Feature Selection **18**
2.2.1 Data Characteristics 18
2.2.2 Filter 19
2.2.3 Embedded 19
2.2.4 Wrapper 20
2.2.5 Hybrid 21
2.3 Multi-objective Approach **21**
2.4 Evaluation Metrics in Multi-objective Feature Selection **22**
2.4.1 Classifier Evaluation Metrics 22
2.4.1.1 Multi-class Metrics 23
2.4.1.2 Cross-Validation 24
2.4.2 Multi-objective Evaluation Metrics 25
2.5 Evolutionary Algorithms for Feature Selection **25**
2.5.1 NSGA-II 26
2.5.2 SPEA-II 27
2.6 Random Forest for Feature Selection **28**
3 RELATED WORK **30**
3.1 Literature Review Methodology **30**
3.2 SparseEA **31**
3.3 MOFS-RFGA **32**
3.4 Enhancing classification with hybrid feature selection (MOO-HFS) **33**
3.5 Motivation **34**
3.5.1 Use of Binary Tournament 34
3.5.2 Scalability and Benchmarking 36
3.5.3 Embedded Technique 36
3.5.4 Limit on Number of Features 37
3.6 Comparison **38**
3.6.1 Convergences 38
3.6.2 Divergences and Contributions 39
4 ALGORITHM AND IMPLEMENTATION **41**
4.1 Implementation Framework **42**
4.2 Initialization Strategy **42**
4.3 Genetic Operators **44**
4.4 Limiting the Number of Features **46**
5 EXPERIMENTS **47**
5.1 Experimental Setup **47**
5.2 Datasets **48**
5.3 Metrics **49**
5.4 Reproducibility and Extension **51**

6 RESULTS.....53
6.1 Final Hypervolumes.....53
6.2 Initialization and Convergence.....56
6.3 Solution Diversity.....59
6.4 Scalability and Execution Times.....60
7 CONCLUSIONS.....64
7.1 Main contributions.....64
7.2 Future Works.....65
REFERENCES.....66
APPENDIX A — PARETO FRONTS AND CONVERGENCE GRAPHS.....71
APPENDIX B — RESUMO EXPANDIDO.....73

1 INTRODUCTION

1.1 Overview

We live in the age of data, and our choices are heavily influenced by algorithms from recommendation systems, from what to buy to the content we consume or whom we vote for (O'Neil, 2017). Even the way we move around the city, whether it's the route we choose or the means of transport, is subject to the influence of optimization and machine learning algorithms, carefully calibrated to return an optimized solution based on cost, travel time, profit, influence, retention, or other important metrics (Mittelstadt et al., 2016). These algorithms aim to find the best solution, but they need to consider more than one objective (such as the plane ticket with the shortest connection time and the lowest cost, for example), and naturally, these objectives are conflicting. Thus, there won't be a solution that simultaneously obtains the best results for both (Deb, 2011). All of these optimization and learning algorithms, however, cannot function in a vacuum; they rely on an ever-expanding supply of users generating data as their fuel (Provost; Fawcett, 2013).

Data has become one of the most important resources, as it is used directly to feed these machine learning-based systems (Provost; Fawcett, 2013). This data is produced 24 hours a day, every day, in various ways, whether it's GPS data, exams, or purchases. The trend is for everything to be collected and stored, preferably in a safe place. This huge volume of data is called big data, which not only challenges traditional collection and storage methods, but also processing and analysis methods. As the data scales up, problems like the so-called curse of dimensionality arise when training machine learning algorithms used in these systems (Verleysen; François, 2005).

In machine learning, the curse of dimensionality refers to the challenges that arise when working with high-dimensional data. As the dimensionality of the data increases, several issues emerge that negatively impact the model's performance, such as longer training and inference times and poorer classification accuracy (Debie; Shafi, 2019). As dimensions increase, the data becomes more sparse, making it harder for algorithms to find meaningful patterns and relationships. For some ML algorithms, high-dimensional data can be prohibitive due to the increase in computational costs, which can be exponential (Hira; Gillies, 2015).

One approach to the problem of high-dimensionality is feature reduction, where the premise is simple: reducing the number of features accelerates model training, reduces

inference times, and helps maintain generalization, thereby improving computational performance, mitigating overfitting, and enabling the use of more complex models (Barbieri; Grisci; Dorn, 2024). Feature reduction techniques are an essential step for most pipelines, and they can be defined in two main tasks: feature extraction and feature selection (Guyon et al., 2002).

Feature extraction techniques are widely known in the machine learning field; they transform the original data to a new feature space, like the Principal Component Analysis (PCA) or the t-Distributed Stochastic Neighbor Embedding (t-SNE), which projects the original data into a lower-dimensional representation while trying to maintain as much information as possible (Hira; Gillies, 2015). Feature extraction is great for solving problems where the input data is not understandable or important; however, when interpretability is a factor to consider, feature extraction is not preferred, as it does not retain the physical meanings of the variables when transformations are performed on features (Li et al., 2018). On the other hand, feature selection simply selects a subset of the original features before building the model, offering a distinct advantage in interpretability, as the selection preserves the original meaning by retaining just a more representative subset of features. Thus, it enhances model performance in terms of training and inference times and predictive power, as well as interpretability. (Li et al., 2018).

In Bioinformatics, feature selection is critical in improving the interpretability of machine learning models. This interpretability is particularly important in contexts such as gene expression microarrays and RNA-seq data analysis, because it enables the identification of key genes as potential biomarkers (Cubillos-Chaparro et al., 2024; Grisci; Feltes; Dorn, 2019), directly linking computational results to biological insights. Moreover, feature selection also enhances computational efficiency and predictive accuracy in Bioinformatics, since reducing the number of features typically leads to faster model training and more accurate outcomes (Barbieri; Grisci; Dorn, 2024).

Feature selection can be defined into four categories: filter, wrapper, embedded, and hybrid methods (Ang et al., 2016). Each method has its advantages and disadvantages (which will be further discussed in chapter 2), quickly summarized:

- *Filter methods* assess feature relevance independently of classifiers, typically through univariate or multivariate statistical measures, offering low computational costs (Ang et al., 2016; Li et al., 2018).
- *Embedded methods* integrate feature selection within classifier training, but they have a higher computational cost than filter methods and can identify interactions

between features (Ang et al., 2016; Li et al., 2018).

- *Wrapper methods* employ search algorithms to evaluate feature subsets through classifier performance, resulting in high computational complexity due to exponentially growing search spaces (Ang et al., 2016; Li et al., 2018).
- *Hybrid methods* are a combination of multiple approaches, such as a filter preselection followed by a wrapper refinement, forming a combined pipeline.
- *Ensemble methods* aim to construct groups of feature subsets and then aggregate the result, trusting in the knowledge of the crowds (Ang et al., 2016; Li et al., 2018).

Wrappers excels over filter-based and embedded methods, as they consider the interactions among features and feature subsets' impact on the learning algorithm (Saadatmand; Akbarzadeh-T, 2024). Unfortunately, wrappers have the highest computational cost, and do not scale well as the number of features grows, as the search space grows exponentially, but to mitigate this weakness, it can be combined with filters or embedded methods to form the so-called hybrid methods; thereby, combining the strengths of the approaches (Saeys; Inza; Larrañaga, 2007; Barbieri; Grisci; Dorn, 2024).

1.2 Problem Statement

Critical domains, such as bioinformatics, are often characterized by datasets containing tens of thousands of features, the vast majority of which are typically irrelevant or redundant to the classification task (Barbieri; Grisci; Dorn, 2024). As discussed, feature-selection methods (e.g. filters, wrappers, or embedded methods) either fail to scale to such high dimensionality or ignore crucial interactions among variables, compromising both interpretability and computational efficiency. At the same time, many real-world applications require optimizing more than one objective (e.g., predictive performance vs. model complexity), for which existing multi-objective feature selection algorithms struggle to generate diverse, high-quality Pareto-optimal fronts within a reasonable time on sparse, high-dimensional data. There is therefore a pressing need for a hybrid approach that can guide the search process effectively in these sparse conditions, reducing the exponential growth of the subsets search space while preserving the strengths of wrapper-based evaluation.

1.3 Hypothesis

We hypothesize that incorporating a feature importance weight vector, extracted via an embedded method (e.g., Random Forest), can guide the search operators of a multi-objective wrapper algorithm (NSGA-II). Thus, forming a new hybrid method that yields Pareto fronts with hypervolume metrics better than those produced by state-of-the-art methods, while maintaining a feasible runtime on datasets with tens of thousands of sparse features.

1.4 Research Question

In high-dimensional, sparse feature domains, does the proposed Embedded Weights Multi-Objective Feature Selection (EW-MOFS) algorithm, leveraging an embedded derived weight vector to guide the NSGA-II search, produce solutions that dominate or at least tie with leading wrapper and hybrid methods in terms of hypervolume, without incurring prohibitive computational costs?

1.5 Contribution of This Work

This dissertation introduces the Embedded Weights Multi-Objective Feature Selection (EW-MOFS) algorithm, a novel hybrid approach that extracts a weight vector via an embedded method (Random Forest) and employs it to guide the search operators of NSGA-II. By guiding the wrapper search toward promising regions of the feature-subset space, EW-MOFS aims to produce Pareto fronts with superior or comparable hypervolumes to existing algorithms, while keeping computational effort manageable on high-dimensional, sparse datasets. Comparative experiments against well-known methods demonstrate the performance gains, and we discuss the main challenges encountered when applying wrapper and hybrid methods in such contexts.

1.6 Organization of the Dissertation

Chapter 2 introduces the concepts of feature selection and the literature review. Chapter 3 explains the methodology of the technique development. Chapter 4 introduces

the main components of the proposed technique. Chapter 5 presents the testing results. Chapter 6 discusses the conclusions obtained by the study and the possibilities of future work.

2 OVERVIEW

2.1 Machine Learning

Machine learning algorithms refer to computational techniques that can find connections between a set of inputs and a set of outputs by learning relevant patterns in the data (Santosh; Das; Ghosh, 2022). Tom Mitchell defined it as: "A program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell, 1997).

According to (Mahesh et al., 2020), there are three types of machine learning: Supervised, Unsupervised, and Reinforcement Learning. To summarize:

- Supervised Learning: A machine learning task that maps inputs to an output based on pairs of examples of input and output. The goal of the algorithm is to train a classifier that can correctly determine the class of new examples that have not yet been labelled.
- Unsupervised Learning: Unlike supervised learning algorithms, the objective is not to estimate a target class; instead, the algorithms try to find and represent interesting structures in the data on their own. Although there is no correct class, evaluating the quality of the patterns found is still possible.
- Reinforcement Learning: The task consists of training software agents to take actions in an environment in order to maximize some objective. Instead of data being labelled or unlabelled, the agent interacts with the environment, and the training is based on the penalties/rewards the agent gets.

Most feature selection techniques focus on supervised learning, although unsupervised learning can also benefit from feature selection techniques (Ang et al., 2016). Interestingly, unsupervised and reinforcement learning techniques are also sometimes used as feature selection strategies (Ang et al., 2016; Liu et al., 2023). This work focuses exclusively on feature selection for supervised learning.

2.2 Feature Selection

Feature selection is a vital part of most supervised learning pipelines; according to (Saeys; Inza; Larrañaga, 2007), it presents three major advantages. (1) It helps to improve model performance, reduces the risk of overfitting, and enables models to better generalize, thus obtaining better classification metrics. (2) It provides models with faster training and inference times, improving the computational performance, as fewer features lead to simpler models. (3) It allows a better understanding of the data; selecting or assigning importance values to the features can help specialists understand the reasons behind the model decisions.

According to (Yu; Liu, 2004), features can be divided in four categories: (a) Noisy and irrelevant; (b) Redundant and Weakly relevant; (c) Weakly relevant and Non-redundant; (d) Strongly relevant. The classification performance is boosted when the noisy, redundant, and irrelevant features are removed (Venkatesh; Anuradha, 2019). The main goal of feature selection is to reduce the features from groups a and b, maintaining only the subset that contains the features from groups c and d, which are the key factors for the correct classifier predictions.

2.2.1 Data Characteristics

It is important to consider the characteristics of the data when performing feature selection. For example, some features may be irrelevant individually, but become relevant when combined with other features, like in the classic example of the XOR problem (Wang; Jiang; Jiang, 2021). In the XOR problem, knowing the target class without considering all the inputs involved in the operation is impossible. This is a well-known example where univariate feature selection methods are not well-suited.

Another important data characteristic to consider is the dimensionality. In supervised learning algorithms, as the dimensionality increases, the computational cost also increases, usually exponentially (Hira; Gillies, 2015). Feature selection can help address this problem by selecting a smaller subset of features, but feature selection algorithms are also prone to computational complexity. Some methods do not scale well with higher dimensionality, some are suited, and some are designed specially for this type of problem (Hira; Gillies, 2015).

(Saeys; Inza; Larrañaga, 2007) Separate the feature selection algorithms into four

categories: Filter, embedded, wrapper, and hybrid.

2.2.2 Filter

Filters work by calculating the relevance of each feature with statistical tests against the target variable; normally, the features are ranked with some relevance score (Barbieri; Grisci; Dorn, 2024). Unlike the wrapper and embedded methods, filter methods are independent of the classifier (Saeys; Inza; Larrañaga, 2007). Filters can be considered the method with the lower computational complexity; they can be computed relatively quickly, which means that they are well-suited for large datasets. Their simplicity and lack of interaction with the classifier mean that the risk of overfitting is low, and they can be used with any classifier (Warkiani; Moattar, 2025).

The main disadvantages when using filters come with the inherent simplicity of the technique: (1) filters evaluate and rank features based on relevance, without interacting with the classifier, they are not capable of identifying redundant features, making the feature subset larger than necessary, and (2) ignore feature interactions, which can be crucial depending of the nature of the data (Pudjihartono et al., 2022).

Filters can be divided into two categories: Univariate methods and Multivariate methods (Saeys; Inza; Larrañaga, 2007). Univariate methods are simpler, faster and scale better with large data but are vulnerable to the feature interactions problem. Multivariate techniques can better map the feature dependencies, normally evaluating them pairwise, with some newer techniques being capable of detecting even higher orders. However, the computational complexity of multivariate filters limit their utilization with high-dimensional data while still suffering from the Filter's limitations, like the lack of classifier interaction (Pudjihartono et al., 2022).

2.2.3 Embedded

In Embedded methods, feature selection is carried out during model construction; all the features are passed to the training phase, the internal weights are adjusted, and at the end, in addition to the constructed model, it is also possible to extract the importance of the features that the model has defined (Saeys; Inza; Larrañaga, 2007). This method is more complex than filters, usually taking more computational resources to execute; however, it

can be considered more powerful, as it considers the interaction with the machine learning model (Barbieri; Grisci; Dorn, 2024). Similar to filters, embedded methods rank returned features in order of relevance.

Regularization techniques can be used to extract the feature importances (e.g., SVM and logistic regression), obtaining them from the support vector (Molnar, 2020). Tree-based methods (e.g., Decision Trees, Random Forest, Gradient Boosting, etc.) are widely used as embedded techniques as well (Molnar, 2020). They perform well in most cases and can capture the interaction between features. Unfortunately, there are some limitations, as the effectiveness of tree-based methods to detect feature interactions declines as the number of features increases (Pudjihartono et al., 2022).

2.2.4 Wrapper

Wrapper methods work by directly testing subsets of features using a model; the features that achieve the best performance after evaluating the model are selected (Saeys; Inza; Larrañaga, 2007). As the number of features grows, the subset search space grows exponentially, and to address this, heuristic search methods are used to guide the search for an optimal subset (Saeys; Inza; Larrañaga, 2007). There are two main search strategies for Wrappers: deterministic and randomized (Hira; Gillies, 2015).

Deterministic wrappers use a greedy algorithm to decide which features to select. The Sequential Feature Selector, for example, implemented by (Pedregosa et al., 2011), chooses the best feature to add or remove (the process can happen forward or backward) based on the cross-validation score of the model. Recursive Feature Elimination (RFE) selects features by recursively eliminating them based on the weights obtained by the SVM model (Guyon et al., 2002). The main disadvantage of deterministic wrappers is that they are more prone to getting stuck in a local optimum (Saeys; Inza; Larrañaga, 2007).

Randomized wrappers, on the other hand, use some heuristic method to navigate the search space. For example, Genetic Algorithms and Simulated Annealing are widely used (Saeys; Inza; Larrañaga, 2007). Besides being less prone to getting stuck in local optima, they have a higher risk of overfitting than deterministic search strategies.

Wrappers have a much higher computational cost than filter and embedded methods, as each step the wrapper performs involves training and testing a model. Also, the selection performed depends on the classifier used with the wrapper; therefore, satisfac-

tory results may not be obtained if the selection is used with different classifiers. Wrappers in general are also prone to overfitting (Barbieri; Grisci; Dorn, 2024)

2.2.5 Hybrid

Hybrid methods combine different methods to take advantage of the strengths of each technique (Saeys; Inza; Larrañaga, 2007). There is always a trade-off between computational complexity and performance in feature selection (Pudjihartono et al., 2022). For example, a filter method might first eliminate a subset of irrelevant features, followed by a wrapper method performing a more refined optimization of the remaining features, thereby taking advantage of the generalization and fast computation of the filter and the more costly and sophisticated wrapper search.

2.3 Multi-objective Approach

Reducing the number of features normally leads to better-performing models, both from the computational and statistical point of view (Barbieri; Grisci; Dorn, 2024). However, at some point, when only the features from the "groups c and d" (Weakly relevant and Non-redundant, and Strongly relevant) are removed, the model begins to lose information, as removing features will remove useful information altogether. Unfortunately, the ideal number of features is not known. Still, the main objective is to minimize the number of features and the model's error, which can be conflicting. Therefore, the problem fits in a multi-objective optimization model (Jiao et al., 2024). In multi-objective optimization, we seek to obtain a specific set of solutions representing the tradeoffs between them, known as Pareto-optimal solutions (Deb, 2011).

A Pareto-optimal solution represents a decision vector where no objective can be improved without worsening at least one other objective (Deb, 2011). This concept appears in scenarios with conflicting goals, where a solution is said to dominate another if it is better for both objectives or at least better in one and equal in the other(s); the set of solutions that are not dominated by any other solution is called a non-dominated set (Deb, 2011). The Pareto front is a set formed only by the non-dominated solutions. The Pareto front captures the trade-offs between both objectives, reflecting non-dominated solutions without requiring prior preference weighting (Veldhuizen; Lamont et al., 1998).

This multi-objective formulation for the feature selection problem involves two objectives: the first one being the number of features, and the other the classification error, which normally is a metric that evaluates the model's inference performance (e.g., accuracy, precision, sensitivity, specificity) (Jiao et al., 2024). Formally, given a set $X = \{x_1, x_2, ..., x_n\}$ representing all features in the dataset, a subset $S \subseteq X$, and a function $g(S) : R^n \rightarrow R$ that evaluates the error of a classifier trained using only the selected features in S, the multi-objective problem can be defined as:

$$\min f_1(S) = e(S) \tag{2.1}$$

$$\min f_2(S) = |S| \tag{2.2}$$

2.4 Evaluation Metrics in Multi-objective Feature Selection

There are two main aspects to consider when evaluating the performance of the multi-objective feature selection method. (1) How well a solution performs is how well that subset of selected features performs when trained/tested with the model; for that, normally, confusion matrix-based metrics are used. (2) How well the set of solutions performed, that is, how close the Pareto-front formed by the solutions found by the method is to the optimal Pareto-optimal set; in the multi-objective field, there are metrics specifically designed to evaluate that (Jiao; Xue; Zhang, 2024).

2.4.1 Classifier Evaluation Metrics

The most straightforward evaluation metrics for the classifier are the accuracy, sensitivity, specificity, and precision. They are all confusion matrix-based metrics (Rainio; Teuho; Klén, 2024).

- The accuracy is the ratio of correct predictions over the total number of predictions (Rainio; Teuho; Klén, 2024);
- The sensitivity, or recall, is the ratio of True Positives over the total Positive class predicted; it measures how well a model can identify positive instances or true positives (Rainio; Teuho; Klén, 2024);
- The specificity measures how well a model can identify negative data points, it is

the ratio of True Negatives over the total Negative class predictions that were made (Rainio; Teuho; Klén, 2024);

- The precision measures how often a model correctly predicts positive instances, it is the ratio between correctly predicted Positive instances and the total predictions made for the positive class (Rainio; Teuho; Klén, 2024).

Accuracy, sensitivity, specificity, and precision can be formally defined in the Equations 2.3 2.4 2.5 and 2.6, respectively, where the TP represents True Positives, TN True Negatives, FP False Positives, and FN False Negatives.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{2.4}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2.5}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.6}$$

2.4.1.1 Multi-class Metrics

Confusion-matrix-based metrics are designed for binary classification (when the prediction consists of only two classes). These metrics can be generalized using macro-averaging or micro-averaging for problems with $k \geq 3$ classes. To do that, it is necessary first to construct a $k \times k$ Confusion Matrix where each row represents true classes, and columns represent predicted classes.

The element n_{ij} at row i, column j is the number of instances from class i predicted as class j. Then generate Class-Specific 2×2 Confusion Matrices for each class i $(i = 1, ..., k)$ where i class i treated as positive and all the other negative, the entries for

the i-th binary matrix are defined in Equation 2.7. 2.7

$$\text{TP}_i = n_{ii}, \quad \text{TN}_i = \sum_{j \neq i} \sum_{h \neq i} n_{jh}, \quad \text{FN}_i = \sum_{j \neq i} n_{ij}, \quad \text{FP}_i = \sum_{j \neq i} n_{ji}, \quad i = 1, \ldots, k. \tag{2.7}$$

Now the traditional metrics (e.g., Sensitivity, Specificity) for each class i can be calculated using TP_i, TN_i, FP_i, FN_i. The Macro-Average calculation for any metric across all k classes is shown in Equation 2.8. To calculating the micro-average is necessary to aggregate all TP_i, TN_i, FP_i, FN_i across classes, then compute the metric once using the same equations from the binary class (Equations 2.3 2.4 2.5 and 2.6). The macro-average computes metrics for each class individually and averages the results across all classes, while the micro-average computes metrics using total instances. (Rainio; Teuho; Klén, 2024)

$$\text{Macro-Average} = \frac{1}{k} \sum_{i=1}^{k} \text{Metric}_i. \tag{2.8}$$

2.4.1.2 Cross-Validation

Researchers frequently utilize k-fold cross-validation for evaluating feature subsets. In this method, the dataset is divided into k equally sized folds, over k iterations; each fold acts as the test set once, with the remaining $k-1$ folds used for training (Rainio; Teuho; Klén, 2024). Stratified cross-validation preserves the ratio of samples for each class in each fold, making it a good alternative when the dataset is unbalanced (Pedregosa et al., 2011). A comparison between normal, stratified, and shuffled stratified K-Folding is shown in Figure 2.1.

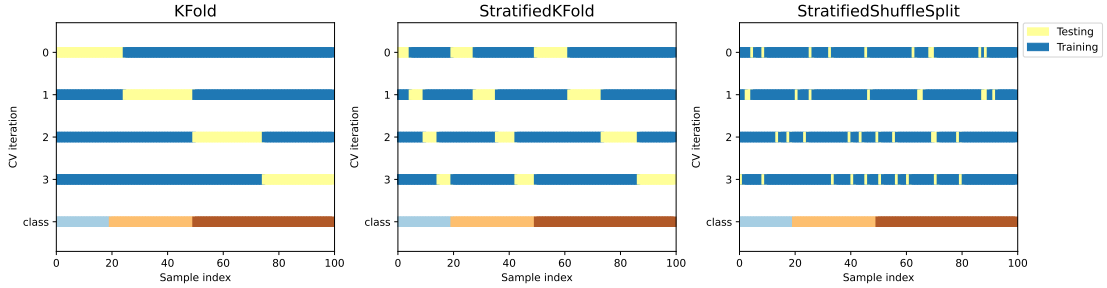


Figure 2.1 – Examples of K-Fold cross-validation plotted. Source: Adapted from the scikit-learn website.

Shuffling can be used to repeat the cross-validation; it improves reliability by executing multiple runs with randomized splits, averaging results to mitigate variability. However, it is important to note that resampling methods inherently produce correlated evaluation scores due to data overlap across folds, which underestimates metric variance and risks biased conclusions (Rainio; Teuho; Klén, 2024).

2.4.2 Multi-objective Evaluation Metrics

There are three main aspects to evaluate the set of solutions obtained from multi-objective optimization: (1) the convergence, that is, how well the obtained Pareto front approximates the true optimal Pareto set; (2) the diversity, which refers to the distribution and the spread of the solutions across the Pareto front; (3) the cardinality which refers to the number of solutions obtained (Riquelme; Lücken; Baran, 2015). However, (Deb, 2011) only considers the first two aspects (convergence and diversity), thus combining the cardinality and diversity characteristics. This work focuses mainly on the convergence aspect with an emphasis on identifying high-quality feature subsets while minimizing their size. However, diversity-preserving mechanisms are an important part of the algorithms, as diversity can lead to better convergence solutions.

One of the most used metrics for evaluating both convergence and diversity is the Hypervolume (HV) (Riquelme; Lücken; Baran, 2015). The HV measures the size of the objective space covered by an approximation set. The HV can consider both aspects because if a front dominates another, the front farther from the reference point will obtain the greater HV; if the fronts are substantially close to each other, the one with higher diversity tends to obtain the best HV (Riquelme; Lücken; Baran, 2015).

HV has the advantage of not requiring a reference Pareto front for calculating the approximation, only a reference point, and this is useful when the true optimal Pareto set is not known (Riquelme; Lücken; Baran, 2015). An example of HV for a set of solutions is shown in Figure 2.2

2.5 Evolutionary Algorithms for Feature Selection

Evolutionary algorithms (EA) are among the most researched algorithms to solve multi-objective problems (Jiao et al., 2024); they work by approximating the optimal

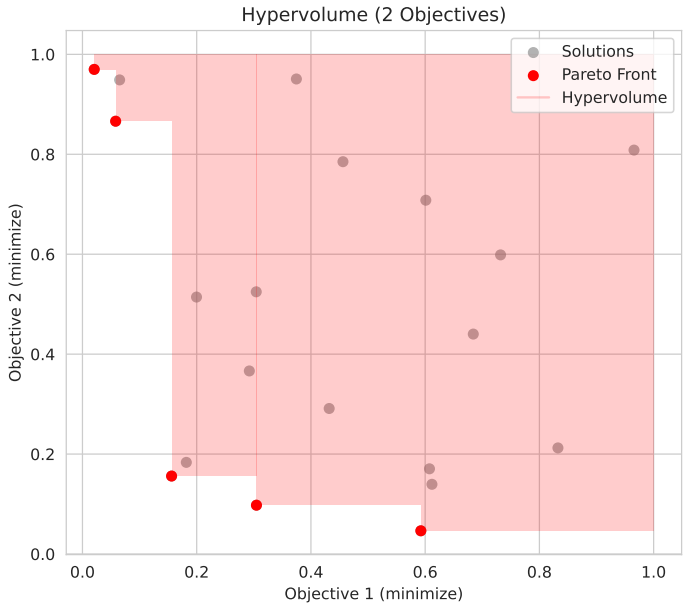


Figure 2.2 – Hypervolume visually represented. Source: The author

Pareto set with some population-based approach, which is iterated over the generations. The main advantages of using EAs are: (1) They are relatively simple to implement and flexible for adaptation; (2) They are robust, performing well on a wide variety of problems; (3) They enable the use of parallel processing power for evaluating the population across the different processes and (4) They naturally identify multiple Pareto-optimal solutions, ideal for handling trade-offs in multi-objective problems (Guliashki; Toshev; Korsemov, 2009)

Most MOEAs, both for theoretical research and for practical applications, share more or less the same framework, similar to NSGA-II (Zhou et al., 2011). There are different types of evolutionary algorithms that use different frameworks, like the MOEA/D, which is based on decomposition (Zhang; Li, 2008). However, the scope of this work remains focused on genetic algorithms, similar to NSGA-II.

2.5.1 NSGA-II

Non-dominated Sorting Genetic Algorithm II, known as NSGA-II, is an evolutionary multi-objective optimization algorithm that efficiently identifies diverse, high-quality solutions approximating the Pareto-optimal set (Deb et al., 2002). While sharing similarities with single-objective genetic algorithms, NSGA-II optimizes multiple objectives through fast non-dominated sorting, which ranks the solutions into successive non-

dominated fronts (F1 containing the best solutions, F2 the second best, and so on) and prioritizes selecting those in the first fronts as shown in Figure 2.3.

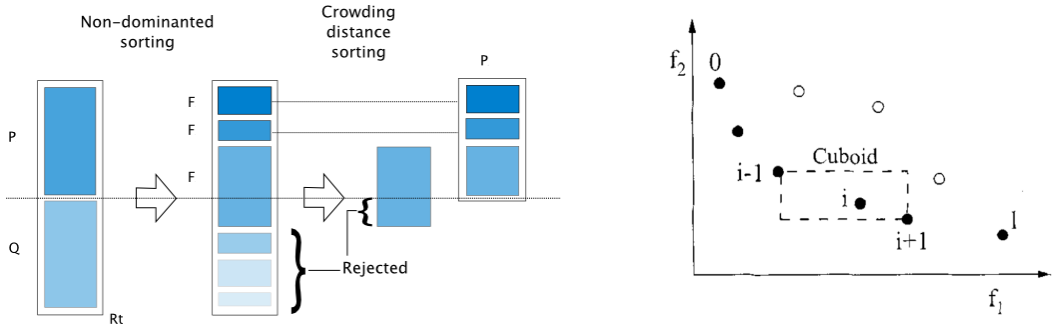


Figure 2.3 – Non-dominated and Crowding distance visual explained. Source: (Blank; Deb, 2020)

The solutions at the last selected front use the crowding distance metric to preserve diversity, thus controlling the density of the solution in the objective space and prioritizing individuals in sparser regions. A more diverse population allows the algorithm to better explore the fitness landscape, leading to a better converged Pareto front (Coello, 2007).

Proposed as a new version of the original NSGA (Non-dominated Sorting Genetic Algorithm) proposed by (Srinivas; Deb, 1994), the main improvements on NSGA-II are the inclusion of the crowding distance mechanism, which improves diversity, and also the elimination of additional parameters, making the algorithm simpler to use. NSGA-II is one of the most representative state-of-the-art MOEAs, inspiring newer versions and implementations across various areas that range from supply chain to bioinformatics (Ma Haiping; Shan, 2023).

2.5.2 SPEA-II

The Strength Pareto Evolutionary Algorithm II, also known as SPEA2, is an evolutionary multi-objective optimization algorithm designed to approximate a well-distributed set of Pareto-optimal solutions through elitism and density-based diversity preservation (Zitzler; Laumanns; Thiele, 2001). Building on genetic algorithm principles, SPEA-II employs a unique fitness assignment strategy that combines strength values by measuring dominance relationships. The algorithm maintains an external archive of non-dominated solutions, which is iteratively updated by merging and truncating the current population and archive. During selection, solutions with the lowest fitness (combining dominance and density) are retained, prioritizing non-dominated individuals while systematically

removing redundant solutions in crowded regions of the objective space. This archive-driven approach ensures a balance between convergence and diversity, enabling SPEA-II to effectively explore complex fitness landscapes and produce a high-quality, evenly spread Pareto front (Coello, 2007).

SPEA-II is proposed as an enhanced version of the original SPEA proposed by (Zitzler; Thiele, 1999). It introduces three key improvements: (1) the refined fitness assignment, which uses a k-nearest neighbor density estimator to rank solutions, (2) an enhanced archive truncation method to preserve boundary solutions, and (3) the elimination of niche parameters required in its predecessor, simplifying its application. SPEA-II is another widely used algorithm; its framework has multiple implementations, including the one by (Blank; Deb, 2020).

2.6 Random Forest for Feature Selection

Random Forest (RF) is a well-known machine learning algorithm (Breiman, 2001). It consists of an ensemble of multiple decision trees, each one trained on a random subset of the data with a random subset of the features. This process, known as bagging, helps reduce overfitting and improve the model's robustness (Breiman, 1996). Each decision tree in the ensemble makes a prediction, and the final decision is determined by majority voting (Pedregosa et al., 2011).

Classified by (Barbieri; Grisci; Dorn, 2024) as an ensemble technique, RF can be used as an embedded technique for feature selection due to its built-in mechanism for evaluating feature importance (Saeys; Inza; Larrañaga, 2007). Feature importance is computed as the mean and standard deviation of the accumulation of the impurity decrease within each tree (Pedregosa et al., 2011).

The main advantages of using RF include the ability to capture interactions between features; unlike most multivariate filters, RF and other tree-based approaches can consider higher-order interactions (with more than two features) (Pudjihartono et al., 2022). However, it has a main limitation that lies in differentiating redundant features, which can significantly degrade model performance and inference time (Pudjihartono et al., 2022). In this work, RF is the embedded model combined with the wrapper to form a hybrid method.

Some key hyperparameters are essential for ensuring robustness when using a random forest for feature selection. Parameters like the number of trees (n_estimators), tree

depth (max_depth), and the number of features tried at each split (max_features) directly affect the variance of the feature importance estimates (Pedregosa et al., 2011). Considering that each split looks to $\sqrt{p}$ (as default in Scikit-Learn for classification problems (Pedregosa et al., 2011)), where p is the total number of features, the probability of a feature not being considered in a split is $1 - \frac{\sqrt{p}}{p}$. The probability of missing a feature, that is, never considering the feature in any split performed by the forest, is stated in equation 2.9, where T is the number of trees, and *splits* is the average number of splits in each tree.

$$probability_not_included = \left(1 - \frac{\sqrt{p}}{p}\right)^{T \cdot \text{splits}} \tag{2.9}$$

For example, in a dataset with 10,000 features, using only 50 trees with an average of 7 splits (maximum number of splits with a maximum depth of 3), using $\sqrt{p}$ features considered in each split means that there is approximately a 3% chance of a feature never being considered in any split. Boosting the forest to a few hundred or more trees makes the probability of missing a feature drop essentially to zero, but at the cost of longer training and inference times.

Ensuring that every feature is considered in at least one split might not be enough to ensure robustness in the relevance scores obtained from the random forest, as, depending on the nature of the data and feature combinations might be required for the algorithm to learn the patterns and consequently require even more trees. There is one general rule of thumb that recommends ten times the number of features as the number of trees (Boehmke; Greenwell, 2019); however, (Lange et al., 2025) show that the stability of random forest increases non-linearly with higher numbers of trees, while the computation time increases linearly, and concludes that the optimal number of features varies according to multiple factors in the data, while proposing a tool that models this relationship between the number of trees and the stability of random forest.

The method proposed (EW-MOFS) relies heavily on the quality of the feature-importance scores from its RF, so the parameters discussed above were considered[1]. The number of trees in each random forest performed by the EW-MOFS was fixed at 10,000, as it was a good balance between computational costs and the number of trees.

[1]It is important to state that the work of (Lange et al., 2025) has not been released yet, while the EW-MOFS was being developed. For this reason, the rule used for determining the number of trees was as much as the computational power allows.

3 RELATED WORK

Recent advances have been achieved in the feature selection field, especially in the research field of hybrid algorithms, which combine the strengths of different types of algorithms to perform better and more stably in different scenarios. Evolutionary algorithms, in particular, are among the most widely researched algorithms for solving multi-objective problems and can be used in hybrid methods in conjunction with other techniques to make a finer selection after a coarser selection process carried out by another method.

There is a type of implementation in the evolutionary algorithm that performs a pre-computation in the initial phase of the algorithm and then utilizes this information to further guide the evolution of the evolutionary process. In this chapter, three multi-objective methods that employ this approach are selected and summarized for comparison with our own proposed technique.

3.1 Literature Review Methodology

To ensure that the present dissertation was grounded in recent and relevant advancements in the field, a systematic literature review was conducted. The methodology for selecting related works was structured to identify hybrid feature selection algorithms based on multi-objective evolutionary approaches, with a specific focus on high-dimensional and sparse data.

The search process was centered on two leading scientific databases: *ScienceDirect* and *Scopus*. The search strategy employed the following query:

```
('feature selection') AND ('sparse' OR 'high dimensional')
AND ('NSGA-II based') AND ('hybrid') AND ('wrapper') AND ('guided')
```

The search was restricted to publications from the last five years, covering the period from 2019 to 2023. This initial query yielded 52 articles from *ScienceDirect* and 58 from *Scopus*.

The article selection followed a two-phase screening process. First, the titles and abstracts of all retrieved articles were read and analyzed to assess whether they had a direct relevance to the dissertation's topic. Articles that were not directly related to multi-objective feature selection, were not applied to high-dimensional problems, or were not based on NSGA-II, were excluded. From this initial screening, the paper "A feature selection approach based on NSGA-II with ReliefF" (Xue; Zhu; Neri, 2023) was identified

as highly aligned with the search criteria and was selected for in-depth analysis.

Additionally, the *snowballing* technique was employed, which involves analyzing the references of selected articles to find other relevant works. Through this method, the paper ”An Evolutionary Algorithm for Large-Scale Sparse Multi-objective Optimization Problems” (Tian et al., 2020) was identified. Although this work was designed for general-purpose sparse optimization rather than exclusively for feature selection, its approach to guiding the evolutionary process in large-scale problems was deemed highly relevant to the hypothesis of this dissertation.

Finally, the work ”Enhancing classification with hybrid feature selection (MOO-HFS)” (Bohrer; Dorn, 2024) was included in the review. This work was developed within the same research laboratory (SBCBLab) where this dissertation was conducted. Its inclusion was strategic to contextualize the present research within the group's investigative lines and to establish a direct comparison with a methodologically similar work.

This structured selection process culminated in the identification of these three key approaches. The following sections provide a detailed overview of each of these works, establishing the foundation upon which our proposed method is built.

3.2 SparseEA

SparseEA, proposed by (Tian et al., 2020), is a multi-objective optimization algorithm designed to tackle high-dimensionality and sparsity problems. It has a similar framework to NSGA-II, but with a pre-computing phase that provides a score vector that will be further used for guiding the initialization and genetic operators. SparseEA is designed to solve both binary and continuous optimization problems; for binary-coded problems (which are the ones used for feature selection) the decision values are encoded in a binary mask, and the real values vector is set to 1.

In the initialization phase, all the variables are individually tested as solutions, non-dominated sorting is performed with the values obtained by each variable individually, and the ranking is stored in a score vector. This score vector is used for initializing the population: the population starts with all genes set to 0 (unselected) and selects a random number of features to be set to 1 (inserted). For each insertion in the individual, two genes are randomly selected and compared, and only the better-performing one will be set to 1, thereby performing a binary tournament using the score vector.

The same binary tournament logic is applied in the genetic operators. For the

crossover, two parents are randomly selected, and then, using the score vector, two genes from one parent are selected, and the best-performing one is set to 1 (added), or the worst-performing one is set to 0 (removed). For the mutation, one gene can be added or removed (50% chance for each); if the operation is an insertion, two genes that are not 1 are selected and compared, and the best score is maintained; if it is a deletion, two are selected, and the worst score one is set to 0.

SparseEA is tested for feature selection in datasets that vary from 100 to 1,000 decision variables; experimental results show the superiority of SparseEA compared with the other seven multi-objective optimization algorithms. For feature selection specifically, four datasets are tested with data ranging from 166 to 617 features; SparseEA obtains better or at least matching HV values in all of them.

3.3 MOFS-RFGA

MOFS-RFGA, proposed by (Xue; Zhu; Neri, 2023), is a hybrid multi-objective feature selection algorithm that integrates the filter method ReliefF (Robnik-Šikonja; Kononenko, 2003) with the wrapper method NSGA-II to address high-dimensional datasets. This approach combines the efficiency of filter methods and the accuracy of wrapper methods, leveraging feature importance scores to guide population initialization and genetic operations, thereby enhancing search directionality and convergence performance.

In MOFS-RFGA, the ReliefF algorithm first assigns weights to features based on their relevance to classification. These weights form a score vector, which is utilized to initialize the population. Individuals are generated by selecting a subset of features through a binary tournament: two features are randomly chosen, and the one with the higher ReliefF score is retained, ensuring that high-quality features are prioritized.

The algorithm introduces novel genetic operators guided by the ReliefF scores. For crossover, a 3-to-1 operator is designed where three parents are combined. Features selected in all three parents are directly inherited by the offspring. For overlapping features selected in two parents, a binary tournament based on ReliefF scores determines retention, favouring higher-scoring features for inclusion or lower-scoring ones for exclusion. Mutation operates similarly: with a 50% probability, either a low-scoring selected feature is removed or a high-scoring unselected feature is added, both determined via binary tournaments on the score vector.

MOFS-RFGA is very similar to SparseEA in the initialization and genetic opera-

tions. The main difference is the use of a filter method instead of manually testing each feature individually. Twenty real-world datasets are used to test MOFS-RFGA against seven other multi-objective algorithms, including NSGA-II, SPEA-II, and SparseEA. Experimental results demonstrate its superiority in most cases, particularly on high-dimensional datasets.

3.4 Enhancing classification with hybrid feature selection (MOO-HFS)

MOO-HFS is a multi-objective genetic algorithm designed for high-dimensional data, proposed by (Bohrer; Dorn, 2024). It is an enhanced version of the NSGA-II to address high-dimensional feature selection classification challenges. Similar to SparseEA, this method integrates prior knowledge into its optimization framework but focuses on combining results from diverse feature selection techniques to guide population initialization and genetic operations.

In the initialization phase, the algorithm constructs a population using a pool of feature sets derived from multiple classical feature selection methods (e.g., ANOVA F-Test, Random Forest, mRMR). These methods generate ranked feature importance lists, which are stored in the pool. For each feature set in the pool, individuals are initialized by probabilistically selecting features based on their importance scores. A randomness factor ensures diversity, allowing even lower-ranked features to be included with controlled probability. The population is further diversified by varying the number of features per individual within predefined bounds.

The genetic operators are customized to leverage the pool's prior knowledge. During mutation, a feature set is randomly selected from the pool, and genes are modified by swapping an active feature (set to 0) with an inactive one (set to 1) based on their rankings in the selected feature set. This ensures mutations prioritize features deemed relevant by existing methods while exploring novel combinations. Although probabilistic selection is employed in the sampling phase, in the genetic operation, the rankings are used just for restricting the feature options within the pool of feature sets.

Experimental results on high-dimensional biomedical datasets demonstrate that the method consistently outperforms individual feature selection techniques, achieving higher F1-scores with significantly fewer features. Key innovations include the integration of feature importance rankings into genetic operations of the NSGA-II's multi-objective framework.

3.5 Motivation

The main objective of this work is to propose a novel hybrid technique, composed of an evolutionary algorithm and an information-guiding approach, with a focus on high-dimensional and high-sparsity data. In this kind of hybrid method, usually the information is obtained in a pre-computing phase, using a filter or embedding technique to sort the relevance of each feature for later guiding the search.

The concept of this information-guided approach is not new; it consists of using the information obtained to interfere in the sampling and genetic operations performed by the evolutionary algorithm. SparseEA and MOFS-RFGA (Tian et al., 2020; Xue; Zhu; Neri, 2023) implement it and manage to outperform other algorithms without that mechanism. MOO-HFS (Bohrer; Dorn, 2024) also uses this approach, but with the difference that it combines multiple filters and embedded methods rather than using just one method for obtaining the information. The aforementioned approaches leave some gaps, which will be explored below.

3.5.1 Use of Binary Tournament

Binary Tournament is a strategy adopted by both SparseEA and MOFS-RFGA to employ the guidance of the evolution process, considering the knowledge obtained in the pre-computing phase. In this mechanism, two candidates are selected, and their precomputed scores (representing the relevance of each feature) are compared, and the candidate with the best score is then selected. This mechanism is implemented in both the sampling and the genetic operations phases.

This approach usually performs great in most cases, as it can "look" for two genes and select just one with greater relevance (if the assigned scores are correct). However, the effectiveness of this tactic progressively diminishes in high-sparsity scenarios.

In a scenario where features can only be classified as relevant or not relevant, normally the chance of a randomly selected feature being relevant is P/N, where P is the number of relevant features and N is the total number of features. Inserting a binary tournament in this selection almost doubles that chance. The Equation 3.1 expresses the chance of any of the selected features being relevant when performing a binary tournament; reposition is considered in this scenario.

$$1 - \left(\frac{N-P}{N}\right)^2 \tag{3.1}$$

As sparsity grows, the chance of at least one relevant feature being selected to participate in the binary tournament gets lower. For example, in a dataset with 100 total features, where 10 features were relevant, assuming the pre-computation algorithm assigned correct relevance values, the chance of at least one of the two selected features is 19%. As sparsity grows, for 1,000 features, for example, the chance gets lowered to 1.99%; for 10,000 features, the chance is only 0.1999%.

Even in an ideal scenario, where the scores were perfectly assigned (with 0 for irrelevant and 1 for relevant, for example), the success of the binary tournament critically depends on at least one relevant feature being selected to participate in the tournament. In highly sparse problems, where N increases significantly while P remains constant, the probability of including a relevant feature in the tournament decreases with the ratio P/N.

An alternative approach is to use the weighted random selection, which is a method of selecting elements based on assigned weights. Unlike the binary tournament, whose effectiveness decreases with increasing sparsity, weighted selection ensures that features with lower relevance have a lower probability of being chosen, as long as the relevance scores are correctly assigned. This property eliminates the statistical dependence of pairwise comparisons, which require the presence of at least one relevant feature in the tournament to perform optimally. The probability of selection is proportional to the scores assigned, allowing (even in extremely sparse environments) the relevant features to be prioritized consistently. If the pre-computing function can perfectly map the feature importance (with 1 for relevant, 0 for not relevant, for example), then every selection performed by the Weighted Random Selection will select relevant features.

For real-world problems, features are not just relevant or not relevant; there are levels of different levels of relevance and redundancy, as described previously in Section 2.2. However, besides depending on the correctness of the feature ranking, Weighted Random Selection is not vulnerable to the expansion of the sparsity, as long as the relevance score is assigned correctly.

3.5.2 Scalability and Benchmarking

Besides being designed for dealing with high-dimensional data, both SparseEA and MOFS-RFGA lack benchmarking in scenarios where the dimensions become even higher. Their reported tests reach a maximum of 1,000 and 2,000 features, respectively. This underestimates the complexity of real scenarios (such as genomic datasets that often exceed tens of thousands of features).

The work of (Bohrer; Dorn, 2024) addresses this issue, providing benchmarking with datasets up to 54,000 features. However, it maintains the comparisons only with other filters and embedded methods, not providing any wrapper or hybrid method for the comparison. In some cases, wrappers and hybrid methods can already be considered better performing over filter and embedded methods (Saadatmand; Akbarzadeh-T, 2024); thus, the impact of this work could not be correctly measured from this perspective.

This study focuses on evaluating datasets with extremely high-dimensionality, encompassing datasets ranging from 30 to 54,000 features. To validate the performance improvements, the proposed method is benchmarked against two traditional wrapper approaches and three hybrid methods: SparseEA, MOFS-RFGA, and MOO-HFS, thus addressing gaps related to the scalability and the lack of benchmarks and comparisons against other wrappers and hybrid methods.

3.5.3 Embedded Technique

Embedded techniques are not commonly combined with wrappers to form hybrid methods. SparseEA, for example, uses a univariate evaluation method to perform the relevance scores, so it is not able to detect dependencies between features. MOFS-RFGA uses Relief, which is a more sophisticated filter technique; it is capable of detecting interactions between variables, but does not take into account interaction with the classifier, in addition to having some inherent limitations, such as being vulnerable to the density of neighbours and its losses in performance as the number of irrelevant features becomes large (Urbanowicz et al., 2018). MOO-HFS is the only related work that uses embedded techniques, employing them alongside other filter techniques to form the pool of methods, but it is not possible to verify that the gains in performance are obtained exclusively by the inclusion of embedded methods or the ensemble properties of the approach.

Given the limitations of Relief and other filter methods, we opted using an embed-

ded method for the EW-MOFS. This approach makes it possible to assess the importance of features in an integrated way in the classification model, taking into account interactions between features and the behavior of the classifier.

The embedded method chosen is based on RF (Breiman, 2001). Although RF is effective in identifying relevant features, it has limitations in distinguishing redundant features, a limitation that wrapper methods can overcome (Pudjihartono et al., 2022). By adopting a hybrid approach, it is possible to combine the integrated selection capacity of RF with the sensitivity to redundancies offered by wrapper methods, mitigating the individual weaknesses of each technique and building a more robust model adaptable to complex scenarios.

3.5.4 Limit on Number of Features

One characteristic of heuristic multi-objective search methods in feature selection is that the size of the final feature subset is inherently unknown. It may range from a single feature up to the full set, depending on the problem. In very high-dimensional sparse problems (e.g., 20,000–50,000 features), evaluating large—or even full—subsets can become computationally infeasible for many classifiers. Conversely, in low-dimensional problems, selecting nearly all features can sometimes yield strong performance, and it is important that our method can still explore those regions of the search space.

Limiting the maximum number of features can speed up the execution of the wrapper, as it reduces the complexity of the trained models. At the same time, it reinforces the focus of the search on solutions with a low number of features. Therefore, regarding processing time, this limitation may be necessary in order to compute the execution of the algorithm in time if this is a limiting factor. On the other hand, limiting the number of features too much may restrict the possible solutions that would belong to the Pareto frontier from being found.

From the related work, the only one limiting the number of features is the MOO-HFS. This work introduces a user-defined parameter to allow accommodating both extreme cases: (1) datasets with many features where the optimal solution is small (where limiting features is advantageous), and (2) datasets where larger feature subsets (not limiting is advantageous).

3.6 Comparison

Table 3.1 – Comparison between the related work and the EW-MOFS.

	SparseEA	MOFS-RFGA	(Bohrer; Dorn, 2024)	EW-MOFS (this work)
Uses NSGA-II as baseline	**x**	**x**	**x**	**x**
Specifically Designed for feature selection		**x**	**x**	**x**
Limits Maximum features number			**x**	**x**
Tested with 20.000 features or more			**x**	**x**
Compares with others Wrappers or Hybrids	**x**	**x**		**x**
Use Weighted Random Selection				**x**

To situate the contribution of this work, this section presents a comparative analysis of the convergences and divergences between EW-MOFS and the related works discussed: SparseEA, MOFS-RFGA, and MOO-HFS. The main distinctions are summarized in Table 3.1.

3.6.1 Convergences

A central point of convergence is the adoption of NSGA-II as the foundational multi-objective optimization framework. All four methods share the same hybrid strategy: they employ a pre-computed knowledge mechanism in the form of feature importance scores to guide the sampling and genetic operators of the evolutionary search. This approach, which combines an initial feature relevance assessment with a more refined wrapper-based search, reflects a consolidated trend in the literature for addressing high-dimensional feature selection problems.

3.6.2 Divergences and Contributions

Despite the similar conceptual foundation, EW-MOFS differs in key methodological decisions designed to overcome the limitations of existing approaches, particularly in high-sparsity and extreme-dimensionality scenarios. The main divergences are:

1. **Guided Selection Mechanism:** The most significant methodological divergence lies in how the pre-computed knowledge is used. While SparseEA and MOFS-RFGA employ the binary tournament for their genetic and sampling operations, EW-MOFS introduces weighted random selection. As discussed in Section 3.5.1, the effectiveness of the binary tournament degrades in highly sparse data, as the probability of selecting at least one relevant feature to compete in the tournament diminishes. Weighted selection addresses this vulnerability by consistently prioritizing features with higher assigned relevance, which proves to be more robust in higher-dimensional scenarios.
2. **Feature Importance Technique:** EW-MOFS diverges in the choice of the technique used to generate the importance weights. SparseEA uses a univariate evaluation method, which is unable to detect dependencies between features, while MOFS-RFGA uses the ReliefF filter, which has known limitations as discussed in Section 3.5.3. This work employs an embedded method (Random Forest), which better captures feature interactions within the context of the classification model.
3. **Scalability and Benchmarking Scope:** A fundamental divergence from SparseEA and MOFS-RFGA is the scope of the experimental validation. These works limit their tests to datasets with a maximum of 1,000 and 2,000 features, respectively, which restricts their proven applicability in real-world scenarios like bioinformatics applications. In contrast, EW-MOFS, similarly to MOO-HFS, advances this by validating its performance on datasets with up to 54,000 features, this way addressing the gap and demonstrating its effectiveness in extreme-dimensional problems.
4. **Comparative Framework:** Unlike MOO-HFS, which only compares its results against filter and embedded methods, this dissertation benchmarks EW-MOFS against other state-of-the-art hybrid and wrapper methods, including SparseEA, and MOFS-RFGA as hybrids, and NSGA2 and SPEA2 as wrappers. This ensures a more rigorous evaluation and provides a clearer picture of its performance relative to its direct competitors.

In summary, although EW-MOFS shares a conceptual basis with recent works, its methodological divergences—primarily the adoption of weighted random selection and a focus on validating performance at extreme scales. This constitutes its main scientific contribution, offering a more robust solution for feature selection in sparse, high, and extreme-dimensional data.

4 ALGORITHM AND IMPLEMENTATION

The EW-MOFS algorithm combines an embedded approach based on RF to obtain the importance weights and applies them to a wrapper based on NSGA-II to guide the evolutionary process efficiently. As a first step, the algorithm calculates the vector of importance of each attribute using the RF (this operation is performed only once). These importances are stored in the weights vector.

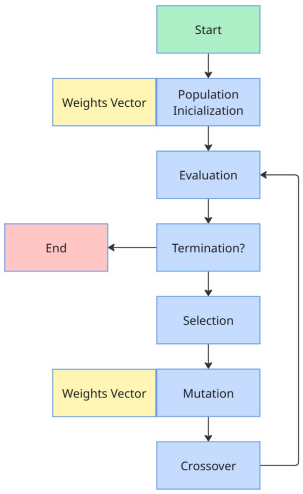


Figure 4.1 – Flowchart of the EW-MOFS algorithm.

Once the vector of weights has been obtained, the sampling phase begins, where the individuals are feature subsets selected according to the weights. After this pre-processing phase, the algorithm follows the same framework as NSGA-II (Deb et al., 2002), with selection via non-dominated sorting and performing the genetic operators. The genetic operators will also make use of the weights vector.

In short, the new strategy modifies the NSGA-II framework with the sampling and genetic operations only, similar to the implementations found in SparseEA, MOFS-RFGA, and MOO-HFS algorithms. The following two sections explain the mechanisms of the initialization strategy and the genetic operations.

In short, the new strategy modifies the NSGA-II framework with the sampling and genetic operations only, similar to the implementations found in SparseEA, MOFS-RFGA, and MOO-HFS algorithms. This entire process is visually summarized in Figure 4.1, where the diagram illustrates the general framework highlighting the two main contributions: the use of the weights vector (obtained via Random Forest) to guide (1) the population initialization and (2) the mutation operators.

4.1 Implementation Framework

The EW-MOFS algorithm was implemented in the Python programming language, leveraging the pymoo library, a robust and well-established framework for multi-objective optimization. The implementation is built upon the existing Non-dominated Sorting Genetic Algorithm II (NSGA-II) provided by the framework, allowing the use of its efficient and battle-tested core components, such as the non-dominated sorting and crowding distance mechanisms.

To integrate the proposed logic, a custom Problem class was defined by extending pymoo's ElementWiseProblem class. This specific parent class is optimized for problems where solutions can be evaluated individually, which fits the feature selection context and simplifies the implementation of the evaluation function. This class encapsulates the feature selection task, where each solution (or individual) in the population is represented as a binary array indicating which features are selected. The two objectives to be minimized, which are the classifier error (1 - F1-score) and the number of selected features, are also computed within this class.

The main contributions of EW-MOFS were integrated by extending pymoo's native classes. The initialization strategy described in Section 4.2 was implemented by creating a new Sampling class that overrides the default random initialization. This new class incorporates the weighted random selection. Similarly, the custom mutation logic, detailed in Section 4.3, was implemented in a new Mutation class. For the crossover operation, the standard UniformCrossover operator from pymoo was employed, with a minor modification to ensure that offspring respect the maximum feature limit defined by the user, as discussed in Section 4.4.

This modular approach of extending pymoo's base classes ensures a robust implementation, preserving the core functionalities of NSGA-II while seamlessly integrating the novel, weight-guided mechanisms of EW-MOFS.

4.2 Initialization Strategy

For the sampling phase, the weights vector is used in a weighted selection for each individual in the population, where the probability of selecting each feature is proportional to its weight. The pseudo-algorithm for the population initialization is shown in Algorithm 1. For that, first an individual is created with no features selected (line

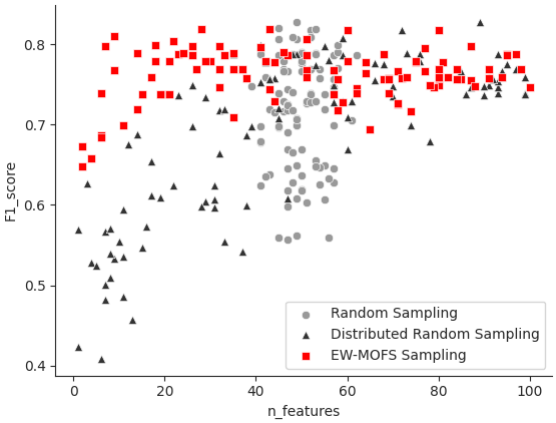


Figure 4.2 – F1-score for number of features selected, from an artificially made problem.

10). Then, a number of features is assigned to this individual (line 11), and each one of these features is randomly chosen with probabilities associated with each feature from the weights vector (lines 12-13). This selection is performed with the Numpy framework. After the individual is assigned with the selected features, it is added to the final population (line 15).

The distribution of the number of selected features in sampling (proposed by (Tian et al., 2020)) is considered when developing the proposed technique, which means that each individual is assigned a number of features randomly chosen from 1 to Max, where Max is the maximum number of features selected by each individual. This feature limitation may be necessary in high-dimensional datasets where fitting the internal model of the wrapper is too costly with a high number of features, or in problems where the maximum number of selected features is known and the search space can be limited. This maximum feature limitation is further discussed in subsection 3.5.4. Instead of attributing a 50% chance of being selected to each feature in the individual, as in the standard Binary Random Sampling implemented by (Blank; Deb, 2020), the ratio of selected features is now chosen randomly for each individual from the interval [0,1].

A quick comparison between the Random Sampling (50% chance for each gene), a naive Distributed Random Sampling, and a Distributed Random Sampling with the EW-MOFS proposed strategy is presented in Figure 4.2. The comparison is made with a synthetic classification dataset, and shows the number of features selected for each individual and the F1-score obtained by each subset. Observing the region with fewer selected features (from 1 to 50), it is possible to conclude that EW-MOFS Sampling obtains individuals with substantially higher F1-scores when compared with the Distributed Random Sampling, while normal Random Sampling simply cannot cover this region entirely.

Algorithm 1 initializePop(N, D, Max, Weights)

```
1:  Input:
2:  N : Size of Population
3:  D : Number of total Features
4:  Max : Maximum number of features selected
5:  Weights: Vector with each feature importance score
6:  Output:
7:  Pop : Initial population
8:  Pop = []
9:  for i = 1 to N do
10:     indv = []
11:     num ← random_integer(1,Max) : Random from 1 to Max
12:     for j = 1 to num do
13:         indv ← weighted_selection(1, D, Weights)
14:     end for
15:     pop ← indv
16: end for
17: return Pop
```

4.3 Genetic Operators

The mutation function for the EW-MOFS is developed to integrate the weights vector, the pseudo-algorithm for the mutation is shown in Algorithm 2. Although it works much like SparseEA and MOFS-RFGA, there are three major points of difference.

- Both other mentioned algorithms pre-define the mutation to occur in exactly one gene per individual. Our mutation sets the probability per gene to $1/N$ where N is the number of total features; consequently, the number of mutations per individual tends to 1, but can vary from 0 to more than 1.
- The mutation on both other techniques occurs by binary tournament, where the same vulnerability to sparsity previously discussed in Section 3.5.1. The proposed technique uses random weighted selection, respecting the weights vector when performing an insertion.
- The embedded weights vector is only used when a gene is being inserted. Each mutation has a 50% chance of being either an insertion or a deletion. In the case of a deletion, the gene is selected randomly. The weights vector is not used during deletions to prevent the algorithm from becoming overly dependent on it.

The mutation is performed in all individuals in the population. For each one, first a number of changes is assigned (line 8), then the True and False indices (selected and uns-

elected features) are identified (lines 9-10), and finally the number of chances is assigned. For each chance, it has a 50% chance of being an insertion or deletion (line 12). If it is an insertion, the weighted random selection is performed (lines 13-15); if it is a deletion, then a random selection is performed (lines 16-18).

Algorithm 2 mutatePop(Pop, Max, Weights)

```
1:  Input:
2:  Pop : Population matrix (individuals × features)
3:  Max : Maximum number of features
4:  Weights : Feature importance vector
5:  Output:
6:  MutatedPopulation : Mutated population
7:  for Individual in Pop do
8:      N_changes ← get_changes(Individual) : Number of mutations
9:      True_indices ← get_trues(Individual) : Genes already selected
10:     False_indices ← get_falses(Individual) : Genes not currently selected
11:     for i = 1 to N_changes do
12:         Insertion? ← random_bool()
13:         if inserted? then
14:             Inserted ← weighted_selection(False_indices, Weights)
15:             Set Individual[Inserted] ← True
16:         else
17:             Deleted ← random_select(True_indices, N_deletions)
18:             Set Individual[Deleted] ← False
19:         end if
20:         MutatedPopulation ← Concat(MutatedPopulation, Individual)
21:     end for
22: end for
23: MutatedPopulation ← verify(MutatedPopulation, max) : Verify final population
24: return MutatedPopulation
```

At the end of the mutation function, the population is checked to enforce the minimum and maximum feature limits (line 23). If an individual ends up with no features, one is randomly selected; if it exceeds the maximum, extra features are randomly removed. The same verification is applied after crossover.

The crossover in the EW-MOFS is the Uniform Crossover, implemented by (Blank; Deb, 2020). Since the algorithm is designed for feature selection in tabular problems, the order in which the features appear in the individual vector does not matter. Therefore, there is no advantage in using single-point or multi-point crossover.

4.4 Limiting the Number of Features

As discussed in Section 3.5.4, it might be necessary to limit the maximum number of features selected by an individual. EW-MOFS uses this user-defined parameter, max, a parameter that caps the number of features any individual can select.

This maximum limit is applied during the initialization and enforced throughout all genetic operators (mutation and crossover), ensuring that every candidate adheres to the same feature-count bound while still allowing the algorithm to explore the full range from 1 to the maximum number of features. In the sampling (Algorithm 1), this limitation occurs in line 11, and in the mutation (Algorithm 2), it occurs in line 23.

5 EXPERIMENTS

EW-MOFS is compared with five other methods: two wrapper methods that use the NSGA-II and SPEA-II algorithms, and three hybrid methods that focus on solving high-dimensional and sparse problems (SparseEA, MOFS-RFGA, and MOO-HFS). Each algorithm runs for 100 generations with 100 individuals and repeats the executions 31 times on each dataset. The execution returns the Pareto optimal sets found for each generation.

The EW-MOFS was implemented in Python using the Pymoo framework. The compared algorithms were implemented in the Pymoo framework, in accordance with the documentation of the original articles and the provided code. To ensure a fair comparison, the limitation on the maximum number of features was incorporated into the sampling functions and genetic operations of all the algorithms.

The parameters used in the six algorithms are the same, the limitation on the number of features is applied on all of them to ensure that possible performance gains are not only caused by a reduction in the search space. For the mutation function, the probability of a mutation occurring in each gene is 1/D (where D is the number of total features in the problem), so the expected number of mutations can vary from 0 to D, however, D mutations occurring is very unlikely to happen. The number should remain close to 1 mutation. In the SparseEA and MOFS-RFGA algorithms, the mutation is set to occur exactly one time per individual, according to the original algorithms of both methods (Tian et al., 2020; Xue; Zhu; Neri, 2023). The parameters can be visualized in Table 5.1.

5.1 Experimental Setup

To ensure reproducibility and the efficient management of the 31 independent runs for each algorithm on each dataset, the experiments were submitted and managed through a batch processing and queue system, ensuring that each run occurred under controlled conditions and without interference. All experiments described in this work were conducted on a high-performance computing node model IBM Lenovo SR650. The hardware configuration of the system used is detailed as follows:

- **Processor:** Intel Xeon Silver 4216 2.1 Ghz
- **Processor Sockets:** 2

- **Processor Cores:** 32
- **Processor Threads:** 64
- **Main memory:** 1 TB
- **Local disks:** 18 TB SAS
- **Accelerators:** RTX A6000 10.752 CUDA cores, 48 GB GDDR6, 768 GB/s Memory Bandwidth, 38.7 TFLOPs (FP32)

The individual-level parallelism is executed by the CPU at the thread level. That is, a thread receives an individual and is responsible for evaluating it. Each thread has its own copy of the dataset being tested for optimizing memory access. It is important to note that all algorithm processing and execution were performed exclusively in the CPU cores, without using the processing power of the GPU.

The software environment was based on the Python programming language. The implementation and management of the evolutionary optimization algorithms were carried out with the aid of the Pymoo library (Blank; Deb, 2020), which provided the framework for NSGA-II and SPEA-II, and also facilitated the implementation of the modifications proposed in EW-MOFS and the other compared hybrid methods. Machine learning tasks, such as training the Random Forest models and evaluating performance metrics, were executed using the Source: The author library (Pedregosa et al., 2011).

5.2 Datasets

To evaluate the effectiveness and robustness of the proposed EW-MOFS method, we selected 20 publicly available datasets that reflect a wide range of real-world classification scenarios. Datasets 1 through 11 were obtained from the UCI Machine Learning Repository (Kelly; Longjohn; Nottingham, 2025), 12 is from the OpenML datasets (Vanschoren et al., 2013), and datasets 13 through 20 originate from the CuMiDa database (Feltes et al., 2019), which focuses on gene expression data and typically presents extremely high-dimensional characteristics.

The primary criterion for dataset selection was diversity in dimensionality. We intentionally included datasets with as few as 30 features and as many as over 54,000, aiming to test the algorithm's scalability and performance in both low and high-dimensional spaces. This broad range allows us to assess not only whether the method can reduce dimensionality effectively but also how well it adapts to problems with drastically different

search space sizes.

In addition to dimensionality, the datasets also vary in other critical aspects like class imbalance and cardinality. Several datasets exhibit highly unbalanced class distributions, which can affect the classifier's performance and the stability of selected feature subsets. Both binary (e.g., QSAR androgen, Breast GSE) and multi-class tasks (e.g., Optical Recognition, Brain GSE) were selected, with class counts ranging from 2 up to 26.

5.3 Metrics

The metric chosen to evaluate the wrapper fitness function is the F1-score, since it is well-suited for both multi-class problems and imbalanced class distributions (Pedregosa et al., 2011). To obtain reliable estimates of generalization performance, stratified k-fold cross-validation is used rather than train–test split or leave-one-out CV (LOOCV). A single train–test partition can yield unstable results due to random variations in the split, while LOOCV, although nearly unbiased, can incur prohibitive computational cost on high-dimensional datasets (i.e., large sample size) (Kohavi et al., 1995).

Stratification ensures that each fold preserves the overall class proportions, preventing misleading performance on minority classes (Pedregosa et al., 2011). To further reduce variance in our performance estimates, the 5-fold procedure is repeated five times with different random seeds, and the results are averaged. This is done to avoid the selected features from overfitting the fold splits.

While optimizing both the number of features and the F1-score, it is necessary to evaluate the quality of the overall set of solutions obtained after a run of the wrapper algorithms; for that, the HV metric is used. Before computing the hypervolume, both objectives (number of features and F1-score) are scaled to the [0,1] interval using min–max normalization, ensuring comparable magnitudes.

To statistically validate the performance gains of EW-MOFS, pairwise Wilcoxon tests (Demšar, 2006) are performed on the paired hypervolume (HV) values obtained from 31 independent runs of EW-MOFS and each competing algorithm. The resulting p-values are adjusted using the Holm procedure (Holm, 1979) to control for type I error in multiple comparisons. This test evaluates whether the distribution of differences in HV between the two methods exhibits statistically significant superior or inferior performance.

Table 5.1 – Parameters used to evaluate each algorithm.

Algorithm	Max Features	Mutations	Pop	Gen
NSGA-II	100	1/D * D	100	100
SPEA-II	100	1/D * D	100	100
SparseEA	100	1	100	100
MOFS-RFGA	100	1	100	100
EW-MOFS	100	1/D * D	100	100

Table 5.2 – Dataset used for benchmarking. NS stands for the number of samples, NF stands for the number of features, and NC stands for the number of classes.

ID	Dataset	NS	NF	NC	Source
P01	Wdbc	569	30	2	(Wolberg et al., 1993)
P02	Satellite	6,435	36	6	(Srinivasan, 1993)
P03	OpticalRecognitionHandwritten	5,620	64	10	(Alpaydin; Kaynak, 1998)
P04	Madelon	2,600	500	2	(Guyon, 2004)
P05	UJIIndoorLoc	1,111	520	2	(Torres-Sospedra Joaqun; Avariento, 2014)
P06	Har	10,299	561	6	(Reyes-Ortiz Jorge; Parra, 2013)
P07	Isolet5	1,559	617	26	(Cole; Fanty, 1991)
P08	MultipleFeaturesDigit	2,000	649	10	(Duin, 1998)
P09	CNAE	1,080	856	9	(Ciarelli; Oliveira, 2009)
P10	QSARandrogenreceptor	1,686	1,024	2	(UCI, 2019)
P11	MicroMass	571	1,300	20	(Mah; Veyrieras, 2014)
P12	Colon Cancer	62	2,000	2	(Alon et al., 1999)
P13	Liver GSE 14520	357	22,277	2	(Feltes et al., 2019)
P14	Leukemia GSE 28497	281	22,283	7	(Feltes et al., 2019)
P15	Breast GSE 70947	289	35,981	2	(Feltes et al., 2019)
P16	Colorectal GSE 44076	194	49,386	2	(Feltes et al., 2019)
P17	Renal GSE 53757	143	54,675	2	(Feltes et al., 2019)
P18	Breast GSE 45827	151	54,675	6	(Feltes et al., 2019)
P19	Colorectal GSE 21510	147	54,675	3	(Feltes et al., 2019)
P20	Brain GSE 50161	130	54,675	5	(Feltes et al., 2019)

5.4 Reproducibility and Extension

This dissertation was developed with a commitment to scientific reproducibility, ensuring that the findings can be verified and the work extended by other researchers. To address this, the following measures were adopted:

- **Data Availability:** All 20 datasets used in experiments are publicly available and were obtained from well-known repositories, including the UCI Machine Learning Repository, OpenML (Kelly; Longjohn; Nottingham, 2025), and the CuMiDa database (Feltes et al., 2019). The sources and characteristics of each dataset are detailed in Table 5.2.
- **Implementation Details:** As described in Section **??**, the computational framework (pymoo in Python) and the implementation architecture (e.g., the use of ElementWiseProblem, Sampling, and Mutation classes) are explicitly detailed. This allows the structure of the EW-MOFS algorithm to be precisely recreated and modified.
- **Parameter Transparency:** The parameters used for all compared algorithms, such as population size, number of generations, and mutation specifications, are documented in Table 5.1. The use of consistent parameters across all algorithms allows a fair and replicable basis for comparison.
- **Open-Source Code:** To allow easier reproduction, the source code of this work is fully available. The complete repository, containing the EW-MOFS implementation, the scripts for running the experiments, and generating the results, is publicly accessible on GitHub at: https://github.com/grkremer/ew-mofs.
- **Environment and Dependencies:** To facilitate the setup of a correct execution environment, the GitHub repository includes a requirements file that lists the exact versions of all Python framework dependencies. This enables the precise replication of the software environment.

To ensure full computational reproducibility and control for the stochastic nature inherent in evolutionary algorithms, the implementation includes a global seed parameter. This parameter controls all randomized processes within an experiment, including the population initialization, the genetic operators (mutation and crossover), data splits in the cross-validation, and model fitting. Consequently, executing a given experiment with the same seed value will return identical results, enabling precise verification of the reported

outcomes. The specific seeds used to generate the results presented in this dissertation are preserved within the execution scripts available in the project's public GitHub repository, allowing the replication of all findings.

Together, these measures form a comprehensive base that enables other researchers to verify the results presented here. Extending and adapting the methodology for future investigations is also facilitated, thereby contributing to advances in the field.

6 RESULTS

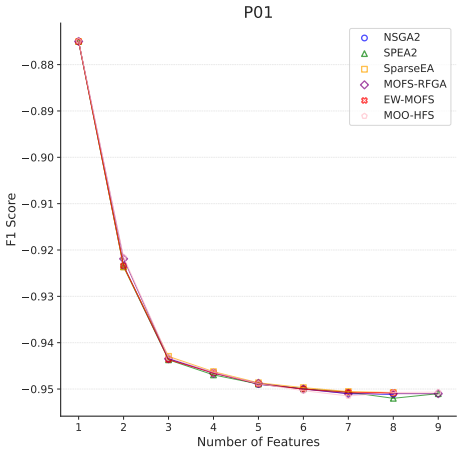
6.1 Final Hypervolumes

The experimental results of the final HV values, including mean HV values, standard deviations, and Wilcoxon test results, are summarized in Table 6.1. In the table, the symbol "+" indicates that EW-MOFS exhibits statistically significant superior performance relative to the compared algorithm (that is, higher HV values with $p < 0.05$), while "-" denotes statistically significant inferior performance (lower HV values with $p < 0.05$). If no statistically significant differences were found between the algorithms ($p < 0.05$), the comparison algorithm receives the symbol "=", which implies inconclusive evidence to favor either method.

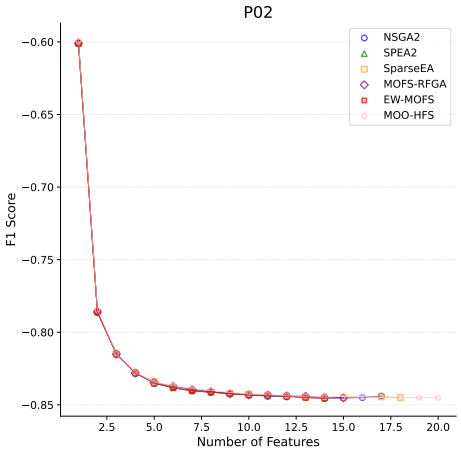
All hypervolume values refer to solution sets computed over objectives, which were first min–max normalized to the [0,1] range. In addition, the p-values from the pairwise Wilcoxon tests have been adjusted using the Holm procedure to ensure proper control of the family-wise error rate in multiple comparisons.

In addition to Table 6.1, two sets of graphs are generated to evaluate the performance of the algorithms visually. The first visualization (Figure A.1) depicts the mean Pareto frontier, calculated from the average F1-score across all solutions for each feature subset size returned by the algorithms. While this frontier serves as an aggregated metric, it does not strictly follow the conventional Pareto optimality properties, as some deviations may occur due to inconsistent solution densities across feature sizes. The second visualization (Figure A.2) presents the convergence curves of the hypervolume metric over generations; these curves allow the comparison of how quickly and consistently each algorithm improves the quality of its Pareto front throughout the evolutionary process.

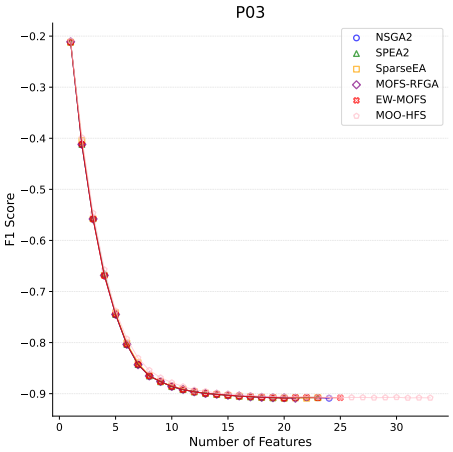
Both complete visualizations are placed in the Appendix A. In this session, only part of the cases will be shown, containing significant examples where EW-MOFS performed worse, the same, or better. For this purpose, 6 graphs have been selected, in the Figure 6.2



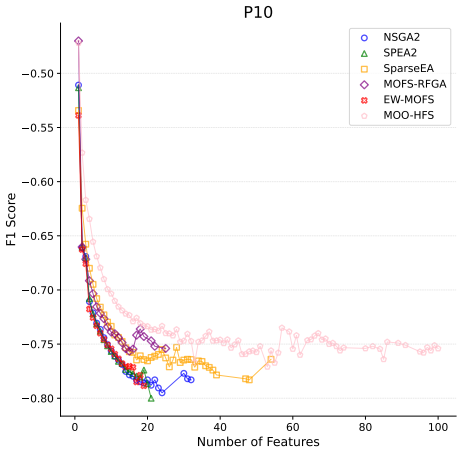
(a) No significant difference.



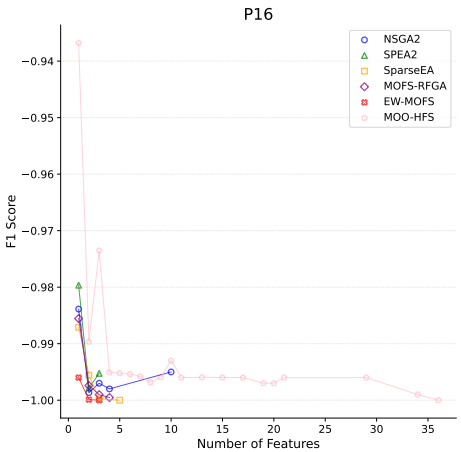
(b) No significant difference.



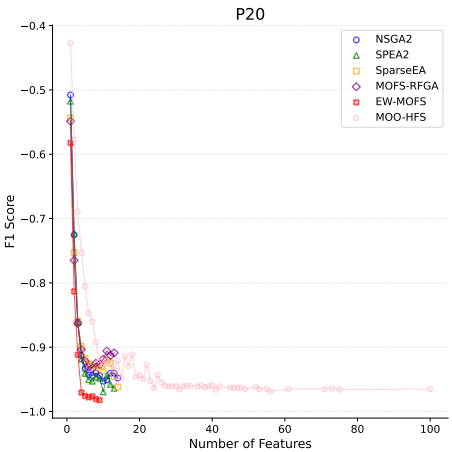
(c) NSGA-II outperform.



(d) SparseEA and MOFS-RFGA outperform.



(e) EW-MOFS outperforms.



(f) EW-MOFS outperforms.

Figure 6.1 – Representative Mean Pareto Fronts for EW-MOFS and baselines: two with almost no difference (top), two where EW-MOFS gets outperformed (middle), and two where EW-MOFS outperforms (bottom) cases.

In the Figure 6.2, for P01 and P02 (marked as (a) and (b), it can be observed that the average Pareto fronts produced by EW-MOFS practically overlap those of the other methods, as in these low-dimensional problems, the new strategy maintains performance comparable to its competitors. In the cases of P03 and P10, marked as (c) and (d), EW-MOFS is outperformed by NSGA-II alone and by the SparseEA and MOFS-RFGA techniques, respectively, suggesting that, under certain data configurations, the custom operators still cannot explore the search space as effectively as traditional schemes; note, however, that in these instances the performance gap is narrow, and these two were the only cases out of twenty in which EW-MOFS was outperformed. Finally, in the graphs for P16 and P20, shown in (e) and (f), EW-MOFS consistently outperforms all other methods across the entire range of subset sizes, reflecting clear gains in convergence and solution diversity in these scenarios.

Besides the visualizations, observing the final HV values in Table 6.1, it is possible to infer that the proposed EW-MOFS obtained the same or better performance in 18 of the 20 datasets. Only in dataset P03 did EW-MOFS obtain a significantly lower score than both SparseEA and MOFS-RFGA, and in P10, it obtained a significantly lower performance than NSGA-II. For datasets P08, P09, and P14 to P20 (9 cases in total), EW-MOFS outperforms all four other algorithms, with no comparable performance observed from any of the alternatives.

It is unreasonable to expect the EW-MOFS algorithm to universally outperform all competing methods across all experimental scenarios, as each algorithm is inherently designed with specialized mechanisms tailored to address specific optimization challenges. However, the empirical evaluation reveals compelling comparative advantages for EW-MOFS:

- When compared to NSGA-II, it outperforms the method in 11 cases, matches in 8 cases, and underperforms in only 1 case. This makes sense because the EW-MOFS is a modification made in the base NSGA-II to explore the search space in high-dimensional problems better, and it performs worse in just one case.
- When compared to SPEA-II, it outperforms the method in 11 cases, matches in 9 cases, and never underperforms. This makes sense, because SPEA-II is another base algorithm with no modifications for better exploring the search space.
- When compared to SparseEA, it outperforms the method in 15 cases, matches in 4, and underperforms in only 1 case. This makes sense, as the features in EW-MOFS were designed to tackle the growth in dimensionality better, as discussed in

section 3.5, and the cases where it got outperformed are the ones with the lowest dimensionality.

- Finally, when compared to MOFS-RFGA, it outperforms the method in 16 cases, matches in 3, and underperforms in only 1 case. This makes sense, as the features in EW-MOFS were designed to tackle the growth in dimensionality better, as discussed in section 3.5.

From the perspective of the optimized problems tested, empirical evidence shows that the EW-MOFS algorithm exhibits robust performance in lower-dimensional datasets (P01 to P08, containing 30 to 649 total features), where it is only marginally outperformed in two instances, as shown in 6.1. However, despite being statistically valid, the difference in the mean Pareto frontiers is low, as can be visualized in Figure A.1.

As dimensionality increases (P09 to P12, ranging from 856 to 2,000 features), the proposed method maintains competitive performance relative to state-of-the-art algorithms. However, the SparseEA and MOFS-RFGA methods begin to exhibit Pareto frontiers that diverge significantly from other techniques, for worse, highlighting their inherent limitations in scaling with dimensionality.

In high-dimensional scenarios (P13 to P20, ranging from 22,277 to 54,675 features), EW-MOFS consistently outperforms all benchmark algorithms, underscoring its efficacy and robustness in handling extreme-dimensional problems. The differences in the Hypervolume values, which were previously in decimal places, are now of a different magnitude. Across these final eight datasets, the mean Pareto frontier generated by EW-MOFS dominates all competing solutions ate every point, with minor exceptions observed only in datasets P13 and P15.

6.2 Initialization and Convergence

The comparative analysis of initial population performance is summarized in Table 6.2, which reports the Hypervolume for Pareto frontiers derived from the first-generation individuals. Statistical significance was assessed using the Wilcoxon signed-rank test, following the same methodology applied to the final hypervolume values in Table 6.1.

NSGA-II and SPEA-II employ uniform binary sampling (each gene has a 50% chance of being selected), which tends to generate dense individuals (around 50% of features active) and slows convergence in sparse domains. This leads to their sampling tech-

	NSGA-II	SPEA-II	SparseEA	MOFS-RFGA	MOO-HFS	EW-MOFS
P01	**0.9403 ± 0.0007 =**	**0.9398 ± 0.0009 =**	**0.9398 ± 0.0007 =**	**0.9401 ± 0.0007 =**	**0.9404 ± 0.0008 =**	**0.94 ± 0.0009**
P02	**0.8324 ± 0.0007 =**	**0.8321 ± 0.0007 =**	**0.8324 ± 0.0007 =**	**0.8321 ± 0.0008 =**	**0.8324 ± 0.0006 =**	**0.8322 ± 0.0007**
P03	0.876 ± 0.0007 =	0.876 ± 0.0007 =	0.8766 ± 0.0005 -	0.8762 ± 0.0005 -	0.8747 ± 0.0011 +	0.8757 ± 0.0007
P04	0.8195 ± 0.001 +	0.819 ± 0.0019 +	**0.8203 ± 0.0012 =**	0.8167 ± 0.0058 +	0.804 ± 0.009 +	**0.8204 ± 0.0012**
P05	**0.9278 ± 0.0013 =**	**0.9277 ± 0.0013 =**	0.9216 ± 0.0032 +	0.9272 ± 0.0016 +	0.9179 ± 0.0065 +	**0.928 ± 0.0008**
P06	**0.9313 ± 0.001 =**	**0.9307 ± 0.0009 =**	0.928 ± 0.0019 +	0.9264 ± 0.0013 +	0.9197 ± 0.0099 +	**0.931 ± 0.0014**
P07	0.7505 ± 0.0046 +	**0.753 ± 0.0043 =**	0.7436 ± 0.0046 +	0.7314 ± 0.0066 +	0.727 ± 0.0059 +	**0.754 ± 0.0037**
P08	0.9397 ± 0.0025 +	0.9398 ± 0.0019 +	0.9377 ± 0.0022 +	0.9357 ± 0.0024 +	0.9174 ± 0.0109 +	**0.9412 ± 0.0011**
P09	0.6987 ± 0.0051 +	0.7003 ± 0.0045 +	0.7015 ± 0.0039 +	0.7002 ± 0.0045 +	0.6813 ± 0.007 +	**0.7053 ± 0.001**
P10	0.7657 ± 0.0111 -	0.7574 ± 0.0106 =	0.7505 ± 0.0084 =	0.7283 ± 0.0129 +	0.7255 ± 0.0112 +	0.7552 ± 0.0087
P11	**0.7411 ± 0.0115 =**	**0.7448 ± 0.0114 =**	0.7265 ± 0.0122 +	0.6753 ± 0.0156 +	0.6908 ± 0.0123 +	**0.7454 ± 0.009**
P12	**0.9653 ± 0.0069 =**	**0.9657 ± 0.0049 =**	0.9532 ± 0.018 +	**0.963 ± 0.0123 =**	0.9246 ± 0.0178 +	**0.9685 ± 0.0034**
P13	**0.967 ± 0.0042 =**	0.967 ± 0.0038 +	0.9658 ± 0.0031 +	0.9653 ± 0.0034 +	0.9576 ± 0.0108 +	**0.9691 ± 0.0023**
P14	0.7989 ± 0.0188 +	0.7991 ± 0.0215 +	0.7742 ± 0.0218 +	0.7634 ± 0.0237 +	0.7797 ± 0.0123 +	**0.8352 ± 0.01**
P15	0.9214 ± 0.0095 +	0.9168 ± 0.0101 +	0.9075 ± 0.0105 +	0.9063 ± 0.0125 +	0.8974 ± 0.008 +	**0.9275 ± 0.0081**
P16	0.9885 ± 0.0018 +	0.9884 ± 0.0018 +	0.9868 ± 0.0021 +	0.9881 ± 0.002 +	0.9785 ± 0.0096 +	**0.9896 ± 0.0006**
P17	0.9384 ± 0.0102 +	0.9376 ± 0.0122 +	0.928 ± 0.0132 +	0.9276 ± 0.0151 +	0.9283 ± 0.0107 +	**0.9542 ± 0.0062**
P18	0.932 ± 0.0215 +	0.9291 ± 0.0189 +	0.9467 ± 0.0151 +	0.9205 ± 0.0286 +	0.9135 ± 0.0148 +	**0.9691 ± 0.0031**
P19	0.977 ± 0.0058 +	0.9796 ± 0.0051 +	0.9779 ± 0.0052 +	0.9752 ± 0.0049 +	0.9611 ± 0.0116 +	**0.9836 ± 0.0038**
P20	0.9328 ± 0.0116 +	0.9376 ± 0.0128 +	0.9191 ± 0.0171 +	0.9206 ± 0.0171 +	0.9286 ± 0.0098 +	**0.9626 ± 0.0035**

Table 6.1 – Final hypervolumes values obtained after the executions of the six algorithms on the twenty datasets, each cell shows [Hypervolume] [Standard Deviation] [Wilcoxon Test Result]. In bold are the cases where EW-MOFS outperformed the other algorithms. Other algorithms are also in bold if they matched the performance.

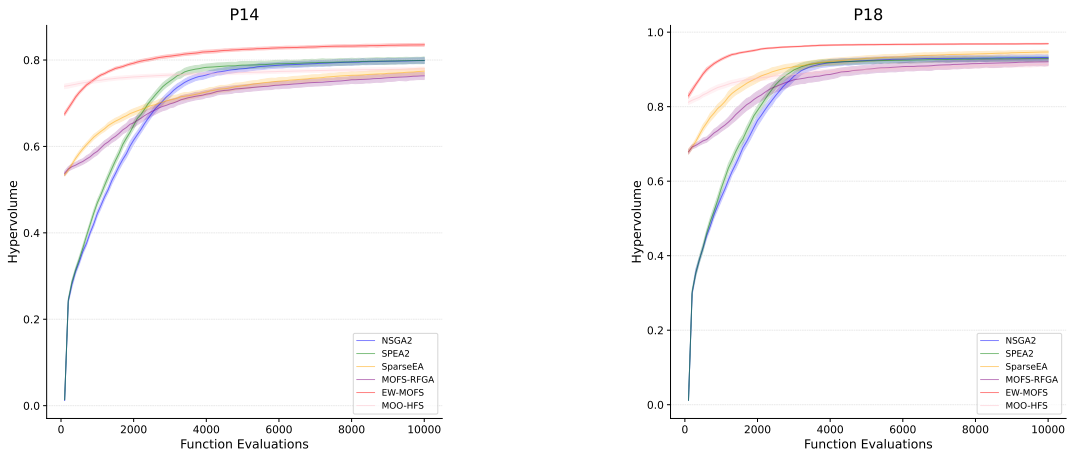
	NSGA-II	SPEA-II	SparseEA	MOFS-RFGA	MOO-HFS	EW-MOFS
P01	0.8605 ± 0.0091 +	0.8629 ± 0.0114 +	**0.9258 ± 0.0029 =**	**0.9267 ± 0.0033 =**	**0.9271 ± 0.004 =**	**0.9271 ± 0.0031**
P02	0.7493 ± 0.0134 +	0.7484 ± 0.0116 +	**0.8226 ± 0.0016 =**	**0.8216 ± 0.0017 +**	**0.8227 ± 0.0027 =**	**0.823 ± 0.001**
P03	0.6918 ± 0.0147 +	0.6889 ± 0.0126 +	0.8449 ± 0.0037 +	0.8464 ± 0.0035 +	0.8471 ± 0.0032 +	**0.8509 ± 0.0028**
P04	0.1848 ± 0.0086 +	0.1859 ± 0.008 +	0.6865 ± 0.0142 +	0.6617 ± 0.0168 +	0.7676 ± 0.0088 -	0.7417 ± 0.0077
P05	0.225 ± 0.0128 +	0.2274 ± 0.0127 +	0.8379 ± 0.0068 +	0.8438 ± 0.007 +	**0.8964 ± 0.009 =**	**0.8956 ± 0.004**
P06	0.2204 ± 0.0122 +	0.2172 ± 0.011 +	0.8857 ± 0.0072 +	0.8797 ± 0.0065 +	0.8999 ± 0.0093 +	**0.9057 ± 0.0034**
P07	0.1563 ± 0.008 +	0.1539 ± 0.0076 +	0.6175 ± 0.0064 +	0.6099 ± 0.0058 +	0.6766 ± 0.0101 -	0.6535 ± 0.0041
P08	0.1964 ± 0.012 +	0.1984 ± 0.0139 +	0.8626 ± 0.0053 +	0.8676 ± 0.0062 +	0.8938 ± 0.0089 +	**0.9005 ± 0.0051**
P09	0.0833 ± 0.0069 +	0.0877 ± 0.0067 +	0.415 ± 0.0157 +	0.4113 ± 0.0169 +	0.6434 ± 0.0083 +	**0.6475 ± 0.0065**
P10	0.1054 ± 0.0077 +	0.1068 ± 0.0095 +	0.6256 ± 0.0077 +	0.636 ± 0.0068 +	0.6788 ± 0.0084 -	0.6586 ± 0.0095
P11	0.0733 ± 0.0037 +	0.0764 ± 0.0062 +	0.4646 ± 0.0069 +	0.4452 ± 0.0116 +	0.5918 ± 0.0123 -	0.5642 ± 0.0065
P12	0.078 ± 0.0075 +	0.0834 ± 0.0084 +	0.7869 ± 0.0198 +	0.7854 ± 0.0253 +	0.881 ± 0.0201 -	0.816 ± 0.0182
P13	0.0201 ± 0.0036 +	0.021 ± 0.0042 +	0.903 ± 0.0087 +	0.9017 ± 0.0082 +	0.9455 ± 0.01 -	0.9287 ± 0.0096
P14	0.0124 ± 0.0022 +	0.0139 ± 0.0029 +	0.5343 ± 0.009 +	0.5378 ± 0.0154 +	0.7393 ± 0.0157 -	0.6751 ± 0.0134
P15	0.0158 ± 0.004 +	0.0166 ± 0.0031 +	0.7859 ± 0.0085 +	0.7892 ± 0.0119 +	0.8726 ± 0.0072 -	0.832 ± 0.0136
P16	0.0169 ± 0.0043 +	0.0181 ± 0.0032 +	0.9629 ± 0.0067 +	0.9543 ± 0.0095 +	0.9748 ± 0.0088 +	**0.9804 ± 0.0071**
P17	0.0136 ± 0.0039 +	0.0143 ± 0.004 +	0.7919 ± 0.0113 +	0.7897 ± 0.0113 +	0.8972 ± 0.0119 -	0.8262 ± 0.0144
P18	0.0121 ± 0.0025 +	0.012 ± 0.0023 +	0.6792 ± 0.0195 +	0.6786 ± 0.0178 +	0.812 ± 0.0169 +	**0.8289 ± 0.0185**
P19	0.0144 ± 0.0039 +	0.0168 ± 0.0021 +	0.8631 ± 0.0147 +	0.8779 ± 0.0176 +	**0.9496 ± 0.0123 =**	**0.9484 ± 0.0097**
P20	0.013 ± 0.0025 +	0.0131 ± 0.003 +	0.7003 ± 0.017 +	0.7157 ± 0.017 +	0.8934 ± 0.0184 -	0.8451 ± 0.014

Table 6.2 – Initial solutions' hypervolume values obtained at the start of the executions of the six algorithms on the twenty datasets, each cell shows [Hypervolume] [Standard Deviation] [Wilcoxon Test Result]. In bold are the cases where EW-MOFS outperformed the other algorithms. Other algorithms are also in bold if they matched the performance.

nique being outperformed across all 20 datasets. As for SparseEA and MOFS-RFGA, EW-MOFS statistically outperforms them on 18 out of the 20 benchmark problems, matching their performance only on the low-dimensional cases P01 and P02, where both pre-computed weight approaches yield similar results.

Against MOO-HFS, MOO-HFS produces higher initial hypervolumes on problems P04, P07, P10–P15, P17, and P20, outperforming EW-MOFS in a total of 10 cases. MOO-HFS integrates multiple initialization strategies (with both filter-based and embedded techniques) to obtain greater diversity in the initial population, which is probably the reason for the better HV values in initialization in these cases.

Despite the static analysis of the initial population, the hypervolume convergence graphs across generations, presented in full in Appendix A (Figure A.2), reveal the search efficiency of each algorithm. Two representative cases illustrate the robust behavior of EW-MOFS. In a different scenario, such as on the P14 dataset, EW-MOFS starts with a lower initial HV than MOO-HFS, but its guided genetic operators promote a faster and more effective convergence, allowing it to overtake and achieve a significantly higher final hypervolume value. On the P18 dataset, EW-MOFS starts the search with the highest HVs and consistently maintains this advantage, with its convergence curve remaining above the others throughout the entire execution. These examples demonstrate that EW-MOFS is capable of maintaining the best HV since the start of the executions, and even when it does not generate the best initial population, the search strategy of EW-MOFS is efficient enough to explore the solution space and converge to higher-quality Pareto fronts.



(a) Starts behind and overcomes.

(b) Head starts and maintains.

Figure 6.2 – Convergence values obtained during the execution algorithms for two representative datasets.

	NSGA-II	SPEA-II	SparseEA	MOFS-RFGA	MOO-HFS	EW-MOFS
P01	12.387	12.290	12.000	13.065	12.258	12.129
P02	18.871	18.000	20.452	18.226	20.548	18.323
P03	24.581	23.871	26.645	25.000	33.645	23.516
P04	12.000	12.097	13.742	11.903	29.871	14.387
P05	12.258	12.935	23.548	16.710	72.742	14.871
P06	27.355	24.903	35.419	28.935	130.161	29.710
P07	47.613	32.484	79.419	47.290	173.355	43.871
P08	24.710	22.581	32.484	24.097	132.710	24.194
P09	15.032	13.194	15.935	14.677	145.194	15.065
P10	26.516	21.677	47.581	28.161	136.613	23.161
P11	52.548	32.323	74.194	40.355	211.839	44.742
P12	11.548	10.774	15.000	10.903	52.839	10.129
P13	13.613	13.677	13.032	12.677	83.677	14.806
P14	20.581	16.452	23.032	18.419	144.548	24.387
P15	19.806	16.032	21.032	16.355	128.290	19.290
P16	5.839	6.194	23.581	12.903	148.677	13.323
P17	15.548	14.419	17.194	12.968	83.935	13.516
P18	16.161	13.806	15.161	13.323	125.419	16.065
P19	9.065	10.516	17.452	10.935	162.839	13.581
P20	15.613	12.194	14.839	14.226	101.323	12.871

Table 6.3 – Average number of features selected per run for each dataset with each algorithm.

6.3 Solution Diversity

This section presents an analysis of solution diversity achieved by the algorithms, measured as the average number of distinct features selected per run for each dataset. Table 6.3 summarizes these values for the six evaluated methods. Figure 6.3 provides a graphical representation of the values in the Table 6.3, for an easier visual comparison.

- For the smaller datasets (P01-P04): all algorithms yield very close average selections with minimal variation (between 11 and 25 features), except for MOO-HFS, which starts to diverge markedly from P04 onward (see Figure 6.3). This consistency across most methods is beneficial, as it indicates similar behavior among the algorithms, enhancing comparability and reliability of the results.
- For the medium-sized datasets problems (P05–P12): most algorithms continue to exhibit similar behavior, with only modest deviations in average feature counts, indicating consistent diversity across methods. Notably, SparseEA begins to vary more significantly, and MOO-HFS diverges even further than in the low-dimensional range, selecting substantially more features (e.g., 130 in P06 and 211 in P11).
- For the larger datasets (P13–P20): MOO-HFS continues exhibiting the poorest per-

formance, selecting features with much more cardinality, with averages reaching above 50 and as high as 162 features in P19. By contrast, EW-MOFS, along with NSGA-II, SPEA-II, SparseEA, and MOFS-RFGA, consistently select relatively few features with closely aligned averages (approximately 10–30 features), preserving compactness and interpretability.

It is important to note that a smaller number of selected features does not always equate to a selection with better quality, as the optimal feature depends on the specific nature of each problem. However, since the overall objective is to minimize the number of features, observing that the average selections remain close across different algorithms indicates stable and expected performance. When feature counts stay around similar means, it suggests that the algorithm behaves consistently and aligns with expectations, reinforcing confidence in its reliability across diverse datasets. Notably, this behavior is evident in tests with the EW-MOFS. Across all dimensionality ranges, EW-MOFS consistently selects a compact set of features while maintaining proximity to the group means, demonstrating both high performance and robustness.

6.4 Scalability and Execution Times

The time used by each algorithm in each dataset can be separated into pre-processing and execution stages. Table 6.5 presents the total execution times for all six algorithms across twenty datasets, and Table 6.4 presents just the time spent in the pre-execution phase. NSGA-II and SPEA-II both register 0 seconds because they do not perform any pre-execution operation.

Despite the additional preprocessing times used by EW-MOFS, its overall runtime (19,231.3s) is lower than that of NSGA-II (26,886.3s) and SPEA-II (24,078.8s), both of which perform no preprocessing at all. This represents a reduction in the total computing time of approximately 28.5% compared to NSGA-II and 20.1% compared to SPEA-II. The results suggest that, despite the time spent in the pre-processing phase representing a huge portion of the total execution time of each algorithm, this time can be saved later in the execution phase, compensated by a more efficient search during the evolutionary process.

The issue with the algorithms NSGA-II and SPEA-II is that they tend to evaluate high-cardinality solutions more frequently, increasing execution time due to the compu-

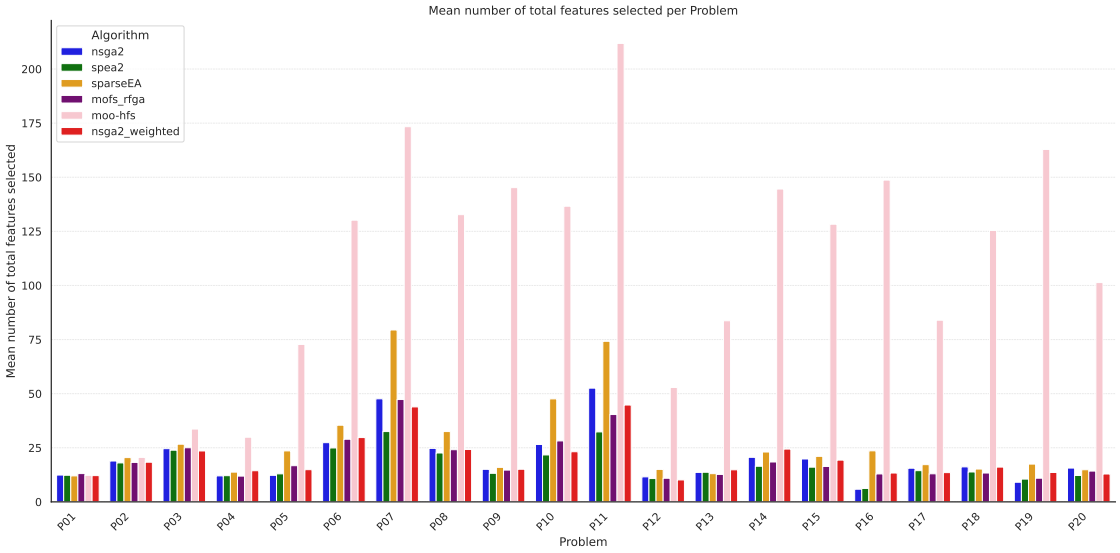


Figure 6.3 – Average number of features selected per run for each algorithm and each dataset

	NSGA-II	SPEA-II	SparseEA	MOFS-RFGA	MOO-HFS	EW-MOFS
P01	0	0	6.7	4.0	2.3	44.6
P02	0	0	10.1	281.7	312.2	54.3
P03	0	0	15.5	284.9	377.0	53.5
P04	0	0	132.0	65.8	273.4	44.1
P05	0	0	85.0	14.0	39.4	48.0
P06	0	0	273.0	544.3	3119.5	108.9
P07	0	0	105.8	22.3	85.7	44.8
P08	0	0	143.2	59.6	488.7	50.2
P09	0	0	145.3	24.9	204.5	49.6
P10	0	0	191.6	17.1	145.4	49.6
P11	0	0	208.2	10.3	100.5	47.8
P12	0	0	318.5	4.4	9.6	40.8
P13	0	0	4219.5	105.3	1096.7	53.2
P14	0	0	5623.6	166.6	1135.9	48.9
P15	0	0	9947.9	135.8	2078.0	57.6
P16	0	0	14319.8	131.1	1316.0	65.6
P17	0	0	20912.0	118.9	1339.2	66.2
P18	0	0	22124.0	188.6	1234.8	72.2
P19	0	0	23238.0	133.6	1009.4	74.4
P20	0	0	19572.0	145.0	1030.0	77.5

Table 6.4 – Pre-processing times (in seconds) of the six algorithms on the twenty datasets. Less is better.

	NSGA-II	SPEA-II	SparseEA	MOFS-RFGA	MOO-HFS	EW-MOFS
P01	123.4	**114.5**	130.0	120.9	270.9	159.2
P02	489.7	**454.4**	525.1	736.4	1607.5	507.3
P03	627.3	602.4	640.3	825.5	2014.0	**597.7**
P04	726.4	661.4	623.1	428.0	1538.3	**413.9**
P05	426.8	398.0	521.6	286.3	850.9	**311.6**
P06	3437.4	2985.2	3270.4	2497.9	10007.5	**1961.8**
P07	908.3	806.4	959.3	**642.6**	2438.2	657.9
P08	741.6	652.3	814.2	**466.7**	2907.0	491.3
P09	576.4	511.6	622.4	345.2	1498.7	**339.3**
P10	659.7	545.8	763.4	**306.5**	1758.2	389.5
P11	553.7	490.5	778.3	**408.3**	1326.5	468.4
P12	280.3	253.1	520.1	**162.5**	422.0	195.4
P13	384.0	350.8	4511.0	332.8	1889.4	**284.7**
P14	477.8	414.3	6059.9	504.6	2251.7	**409.3**
P15	596.0	524.8	10464.7	510.5	3173.1	**441.3**
P16	419.3	408.4	14654.5	452.4	1774.1	**397.2**
P17	528.7	487.5	21356.9	504.0	2096.8	**476.3**
P18	515.2	**467.9**	22560.9	563.7	2193.6	469.6
P19	459.6	442.2	23623.1	496.4	1605.2	**440.1**
P20	511.6	**468.0**	19998.1	545.0	1923.0	779.6
Total	26886.3	24078.8	145202.8	19814.5	71695.1	**19231.3**

Table 6.5 – Average execution times (in seconds) of the six algorithms on the twenty datasets. Less is better.

tation of the learning model (decision tree) over many features. EW-MOFS, in contrast, benefits from its guided initialization and mutation operators that prioritize features with higher relevance, leading to smaller and more effective subsets, and thus lower evaluation costs.

SparseEA shows an exceptionally high average pre-processing time, exceeding 6 hours per dataset from P16 to P19. In total, pre-processing accounts for more than 90% of SparseEA's total runtime. In contrast, EW-MOFS maintains preprocessing times between 40 and 80 seconds, even for large datasets, which represents only about 0.4% of its total runtime.

MOFS-RFGA performs pre-processing based on the relief, with an average cost of approximately 123s per dataset; this is roughly double the 58s required by EW-MOFS. This results in a total runtime of 19,814.5s, about 3% slower than EW-MOFS. When evaluating solution quality, EW-MOFS still outperforms MOFS-RFGA; this demonstrates that EW-MOFS's lighter pre-processing speeds up the overall process compared to MOFS-RFGA.

MOO-HFS exhibits pre-processing times that significantly contribute to its total runtime (71,695.1s), with some datasets requiring over 3,000 seconds before evolution begins. Despite this high cost, the overall execution time of MOO-HFS remains almost 3

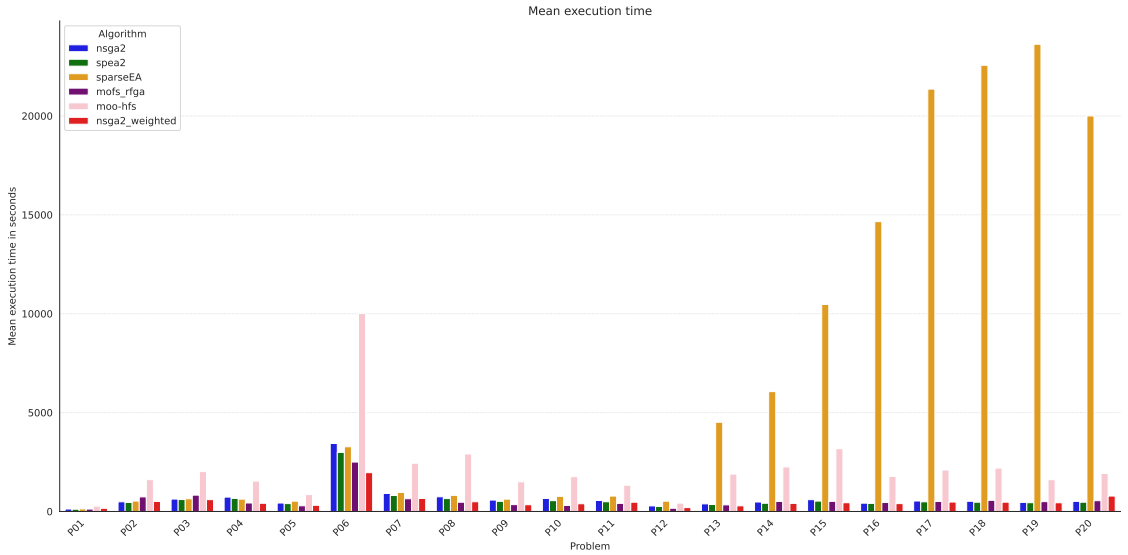


Figure 6.4 – Average execution times (seconds) for each algorithm on each dataset.

times higher than EW-MOFS. MOFS-RFGA also uses preprocessing (based on filter and embedded strategies), but still takes longer than EW-MOFS on average.

Summarizing, EW-MOFS consistently achieves better execution times compared to all other algorithms; a quick comparison can be observed in Figure 6.4. Regarding solution quality, EW-MOFS consistently delivers high-performing Pareto fronts, as shown in Section 6.1. This suggests that the incorporation of feature relevance not only accelerates convergence but also improves the accuracy of the solutions, reinforcing its robustness. Thus, the method demonstrates an optimal trade-off between execution cost and solution quality, validating its applicability in large-scale multi-objective feature selection scenarios.

7 CONCLUSIONS

7.1 Main contributions

This dissertation provides several significant contributions to the field of multi-objective feature selection, particularly for high-dimensional and sparse data. The main contributions are summarized as follows:

- **A Novel Hybrid Algorithm:** The primary contribution is the design and implementation of the Embedded Weights Multi-Objective Feature Selection (EW-MOFS) algorithm. This novel hybrid method effectively integrates an embedded technique (Random Forest) to derive feature importance weights and uses them to guide the search operators of a multi-objective wrapper (NSGA-II). The algorithm introduces new genetic operators specifically designed to handle high data sparsity, directly addressing the scalability vulnerabilities found in existing methods.
- **Demonstrated Empirical Superiority:** The proposed EW-MOFS algorithm was rigorously benchmarked against five state-of-the-art methods across 20 public datasets, with feature counts reaching up to 54,000 features. The experimental results demonstrate that EW-MOFS achieves statistically superior or equivalent performance, measured by the Hypervolume metric, in 18 of the 20 tested scenarios. This superiority is greater in high-dimensional datasets, where EW-MOFS consistently outperforms all competing algorithms
- **Methodological Advancement over Existing Approaches:** This work suggests that using weighted random selection is a more robust strategy for sparse, high-dimensional problems than the binary tournament selection used by other hybrid methods like SparseEA and MOFS-RFGA. By avoiding the statistical pitfalls of binary tournaments in sparse search spaces, EW-MOFS maintains a more effective search trajectory, as evidenced by the superior final Hypervolume values.
- **A Reproducible Contribution:** Beyond the algorithm itself, this dissertation provides a comprehensive and fully reproducible code. The public release of the source code on GitHub, detailed documentation of all experimental parameters, and the provision of execution scripts with fixed seeds allow the replication of all presented results (as detailed in Section 5.4). This facilitates verification, critique, and future extension of the work.

7.2 Future Works

Despite the positive results, this research establishes a solid foundation with possibilities for evolution in future works. The modular framework developed for this dissertation was designed with extensibility in mind, allowing for the implementation of these future directions with targeted adjustments to the existing code. Some key directions include:

- **Weights Generalization:** The algorithm's current dependency on Random Forest for weight computation may restrict flexibility. Future work could explore alternative embedded models, such as gradient-boosted trees or neural networks, to assess the generalization capabilities of the EW-MOFS framework.
- **Adaptive Weight Tuning:** The current implementation uses a fixed weights vector throughout the genetic operations. Developing an adaptive tuning mechanism where weights could be adjusted during the optimization process could allow the algorithm to correct its course and better balance exploration and exploitation.
- **Performance on Ultra-High-Dimensional Data:** While the algorithm scales well to tens of thousands of features, testing its performance on ultra-high-dimensional datasets (e.g., >100,000 features) would better define its practical boundaries and scalability limits.
- **Application to Regression Problems:** Finally, the EW-MOFS framework could be adapted for regression tasks. This would require replacing the F1-score with an appropriate regression metric (e.g., Mean Squared Error) and validating the methodology in new application domains.

REFERENCES

ALON, U. et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. **Proceedings of the National Academy of Sciences**, v. 96, n. 12, p. 6745–6750, 1999.

ALPAYDIN, E.; KAYNAK, C. **Optical Recognition of Handwritten Digits**. 1998. UCI Machine Learning Repository.

ANG, J. C. et al. Supervised, unsupervised, and semi-supervised feature selection: A review on gene selection. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**, v. 13, n. 5, p. 971–989, 2016.

BARBIERI, M. C.; GRISCI, B. I.; DORN, M. Analysis and comparison of feature selection methods towards performance and stability. **Expert Systems with Applications**, v. 249, p. 123667, 2024.

Blank, J.; Deb, K. pymoo: Multi-objective optimization in python. **IEEE Access**, v. 8, p. 89497–89509, 2020.

BOEHMKE, B.; GREENWELL, B. M. **Hands-on machine learning with R**. [S.l.]: Chapman and Hall/CRC, 2019.

BOHRER, J. da S.; DORN, M. Enhancing classification with hybrid feature selection: A multi-objective genetic algorithm for high-dimensional data. **Expert Systems with Applications**, v. 255, p. 124518, 2024. ISSN 0957-4174.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, n. 2, p. 123–140, 1996.

BREIMAN, L. Random forests. **Machine Learning**, v. 45, n. 1, p. 5–32, Oct 2001.

CIARELLI, P.; OLIVEIRA, E. **CNAE-9**. 2009. UCI Machine Learning Repository.

COELLO, C. A. C. **Evolutionary algorithms for solving multi-objective problems**. [S.l.]: Springer, 2007.

COLE, R.; FANTY, M. **ISOLET**. 1991. UCI Machine Learning Repository.

CUBILLOS-CHAPARRO, J. et al. A multiobjective evolutionary algorithm for colon cancer biomarkers identification on gene expression data. In: **2024 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)**. [S.l.: s.n.], 2024. p. 1–8.

DEB, K. Multi-objective optimisation using evolutionary algorithms: an introduction. In: **Multi-objective evolutionary optimisation for product design and manufacturing**. [S.l.]: Springer, 2011. p. 3–34.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, 2002.

DEBIE, E.; SHAFI, K. Implications of the curse of dimensionality for supervised learning classifier systems: theoretical and empirical analyses. **Pattern Analysis and Applications**, v. 22, n. 2, p. 519–536, may 2019.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine learning research**, v. 7, n. Jan, p. 1–30, 2006.

DUIN, R. **Multiple Features**. 1998. UCI Machine Learning Repository.

FELTES, B. et al. Cumida: An extensively curated microarray database for benchmarking and testing of machine learning approaches in cancer research. **Journal of Computational Biology**, v. 26, n. 4, p. 376–386, 2019.

GRISCI, B. I.; FELTES, B. C.; DORN, M. Neuroevolution as a tool for microarray gene expression pattern identification in cancer research. **Journal of Biomedical Informatics**, v. 89, p. 122–133, 2019.

GULIASHKI, V.; TOSHEV, H.; KORSEMOV, C. Survey of evolutionary algorithms used in multiobjective optimization. **Problems of engineering cybernetics and robotics**, v. 60, n. 1, p. 42–54, 2009.

GUYON, I. **Madelon Dataset**. 2004. UCI Machine Learning Repository.

GUYON, I. et al. Gene selection for cancer classification using support vector machines. **Machine Learning**, v. 46, n. 1-3, p. 389 – 422, 2002.

HIRA, Z. M.; GILLIES, D. F. A review of feature selection and feature extraction methods applied on microarray data. **Advances in Bioinformatics**, v. 2015, n. 1, p. 198363, 2015.

HOLM, S. A simple sequentially rejective multiple test procedure. **Scandinavian Journal of Statistics**, [Board of the Foundation of the Scandinavian Journal of Statistics, Wiley], v. 6, n. 2, p. 65–70, 1979. ISSN 03036898, 14679469. Available from Internet: <http://www.jstor.org/stable/4615733>.

JIAO, R. et al. A Survey on Evolutionary Multiobjective Feature Selection in Classification: Approaches, Applications, and Challenges. **IEEE Transactions on Evolutionary Computation**, v. 28, p. 1156–1176, aug. 2024.

JIAO, R.; XUE, B.; ZHANG, M. Solving multiobjective feature selection problems in classification via problem reformulation and duplication handling. **IEEE Transactions on Evolutionary Computation**, v. 28, n. 4, p. 846–860, 2024.

KELLY, M.; LONGJOHN, R.; NOTTINGHAM, K. **The UCI Machine Learning Repository**. [S.l.]: University of California, Irvine, 2025.

KOHAVI, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: MONTREAL, CANADA. **Ijcai**. [S.l.], 1995. v. 14, n. 2, p. 1137–1145.

LANGE, T. M. et al. optrf: Optimising random forest stability by determining the optimal number of trees. **BMC Bioinformatics**, v. 26, n. 1, p. 95, Mar 2025.

LI, J. et al. Feature Selection: A Data Perspective. **ACM Computing Surveys**, v. 50, n. 6, p. 1–45, nov. 2018.

LIU, K. et al. Automated feature selection: A reinforcement learning perspective. **IEEE Transactions on Knowledge and Data Engineering**, v. 35, n. 3, p. 2272–2284, 2023.

MA HAIPING, Z. Y. S. S.-L. T.; SHAN, Y. A comprehensive survey on nsga-ii for multi-objective optimization and applications. **Artificial Intelligence Review**, v. 56, n. 12, p. 15217–15270, Dec 2023.

MAH, P.; VEYRIERAS, J.-B. **MicroMass**. 2014. UCI Machine Learning Repository.

MAHESH, B. et al. Machine learning algorithms-a review. **International Journal of Science and Research (IJSR).[Internet]**, v. 9, n. 1, p. 381–386, 2020.

MITCHELL, T. **Machine Learning**. McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673. Available from Internet: <https://books.google.com.br/books?id=EoYBngEACAAJ>.

MITTELSTADT, B. D. et al. The ethics of algorithms: Mapping the debate. **Big Data & Society**, v. 3, n. 2, p. 2053951716679679, 2016.

MOLNAR, C. **Interpretable machine learning**. [S.l.]: Lulu. com, 2020.

O'NEIL, C. **Weapons of math destruction: how big data increases inequality and threatens democracy**. First paperback edition. New York: B/D/W/Y Broadway Books, 2017.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PROVOST, F.; FAWCETT, T. **Data Science for Business**. [S.l.]: O'Reilly, 2013.

PUDJIHARTONO, N. et al. A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction. **Frontiers in Bioinformatics**, v. 2, p. 927312, jun. 2022.

RAINIO, O.; TEUHO, J.; KLÉN, R. Evaluation metrics and statistical tests for machine learning. **Scientific Reports**, v. 14, n. 1, p. 6086, Mar 2024.

REYES-ORTIZ JORGE, A. D. G.-A. O. L.; PARRA, X. **Human Activity Recognition Using Smartphones**. 2013. UCI Machine Learning Repository.

RIQUELME, N.; LüCKEN, C. V.; BARAN, B. Performance metrics in multi-objective optimization. In: **2015 Latin American Computing Conference (CLEI)**. [S.l.: s.n.], 2015. p. 1–11.

ROBNIK-ŠIKONJA, M.; KONONENKO, I. Theoretical and empirical analysis of relieff and rrelieff. **Machine Learning**, v. 53, n. 1, p. 23–69, Oct 2003.

SAADATMAND, H.; AKBARZADEH-T, M.-R. Many-Objective Jaccard-Based Evolutionary Feature Selection for High-Dimensional Imbalanced Data Classification. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 46, n. 12, p. 8820–8835, dec 2024.

SAEYS, Y.; INZA, I.; LARRAñAGA, P. A review of feature selection techniques in bioinformatics. **Bioinformatics**, v. 23, n. 19, p. 2507–2517, 08 2007.

SANTOSH, K.; DAS, N.; GHOSH, S. Chapter 1 - introduction. In: SANTOSH, K.; DAS, N.; GHOSH, S. (Ed.). **Deep Learning Models for Medical Imaging**. [S.l.]: Academic Press, 2022, (Primers in Biomedical Imaging Devices and Systems). p. 1–27.

SRINIVAS, N.; DEB, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. **Evolutionary Computation**, v. 2, n. 3, p. 221–248, 1994.

SRINIVASAN, A. **Statlog (Landsat Satellite)**. 1993. UCI Machine Learning Repository.

TIAN, Y. et al. An evolutionary algorithm for large-scale sparse multiobjective optimization problems. **IEEE Transactions on Evolutionary Computation**, v. 24, n. 2, p. 380–393, 2020.

TORRES-SOSPEDRA JOAQUN, M. R. M.-U. A. A. T.; AVARIENTO, J. **UJIIndoorLoc**. 2014. UCI Machine Learning Repository.

UCI. **QSAR androgen receptor**. 2019. UCI Machine Learning Repository.

URBANOWICZ, R. J. et al. Relief-based feature selection: Introduction and review. **Journal of Biomedical Informatics**, v. 85, p. 189–203, 2018.

VANSCHOREN, J. et al. Openml: networked science in machine learning. **SIGKDD Explorations**, ACM, v. 15, n. 2, p. 49–60, 2013.

VELDHUIZEN, D. A. V.; LAMONT, G. B. et al. Evolutionary computation and convergence to a pareto front. In: CITESEER. **Late breaking papers at the genetic programming 1998 conference**. [S.l.], 1998. p. 221–228.

VENKATESH, B.; ANURADHA, J. A Review of Feature Selection and Its Methods. **Cybernetics and Information Technologies**, v. 19, n. 1, p. 3–26, mar. 2019.

VERLEYSEN, M.; FRANÇOIS, D. The curse of dimensionality in data mining and time series prediction. In: CABESTANY, J.; PRIETO, A.; SANDOVAL, F. (Ed.). **Computational Intelligence and Bioinspired Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 758–770.

WANG, L.; JIANG, S.; JIANG, S. A feature selection method via analysis of relevance, redundancy, and interaction. **Expert Systems with Applications**, v. 183, p. 115365, 2021.

WARKIANI, M. E.; MOATTAR, M. H. A comprehensive survey on recent feature selection methods for mixed data: Challenges, solutions and future directions. **Neurocomputing**, v. 623, p. 129372, mar. 2025.

WOLBERG, W. H. et al. **Breast Cancer Wisconsin (Diagnostic)**. [S.l.]: UCI Machine Learning Repository, 1993.

XUE, Y.; ZHU, H.; NERI, F. A feature selection approach based on nsga-ii with relieff. **Applied Soft Computing**, v. 134, p. 109987, 2023.

YU, L.; LIU, H. Efficient feature selection via analysis of relevance and redundancy. **Journal of machine learning research**, v. 5, n. Oct, p. 1205–1224, 2004.

ZHANG, Q.; LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. **Evolutionary Computation, IEEE Transactions on**, v. 11, p. 712 – 731, 01 2008.

ZHOU, A. et al. Multiobjective evolutionary algorithms: A survey of the state of the art. **Swarm and evolutionary computation**, Elsevier, v. 1, n. 1, p. 32–49, 2011.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. **SPEA2: Improving the strength pareto evolutionary algorithm**. [S.l.], 2001. 21 p. p.

ZITZLER, E.; THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. **IEEE Transactions on Evolutionary Computation**, v. 3, n. 4, p. 257–271, 1999.

APPENDIX A — PARETO FRONTS AND CONVERGENCE GRAPHS

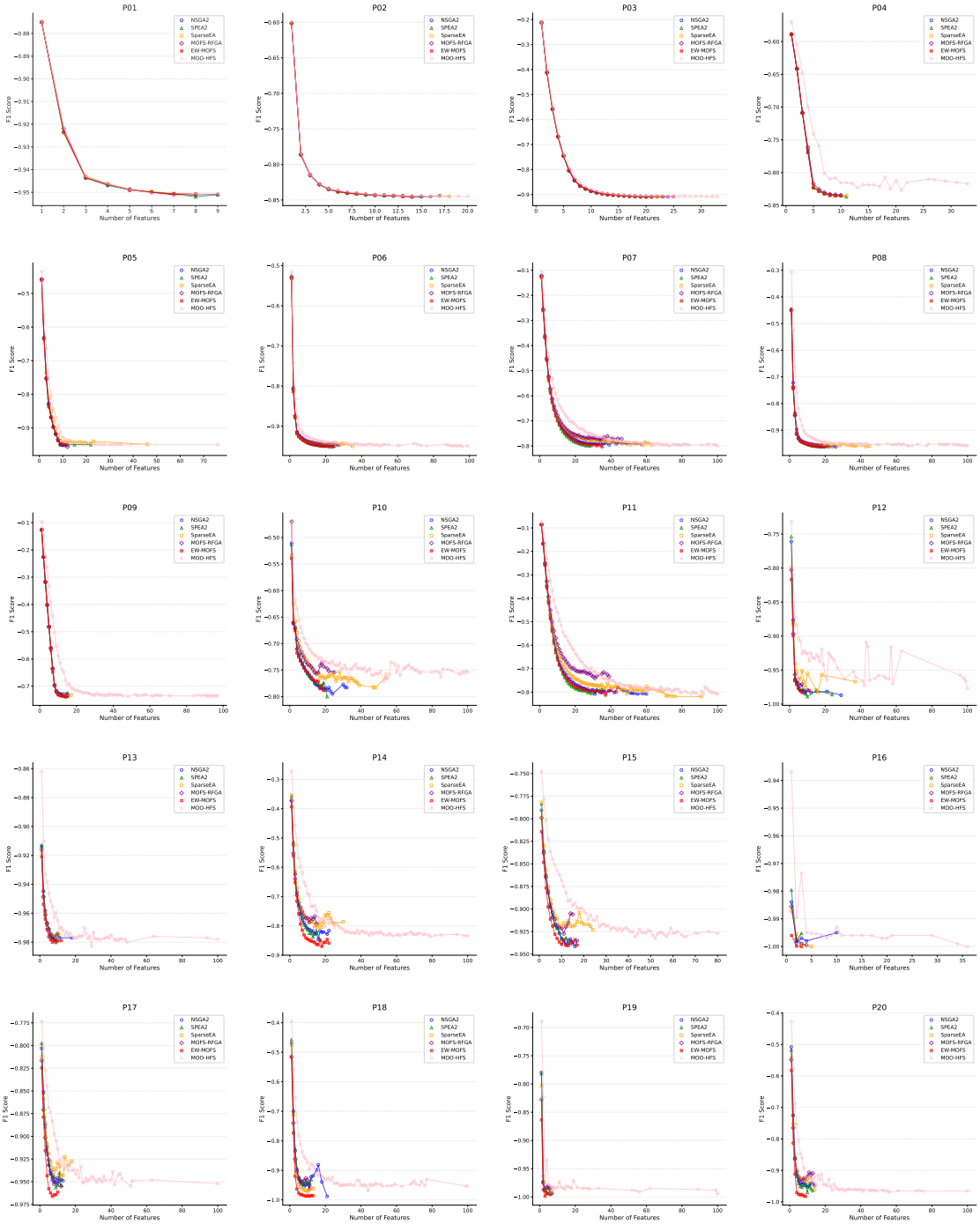


Figure A.1 – Mean Pareto Fronts obtained after executing the six algorithms on the twenty datasets.

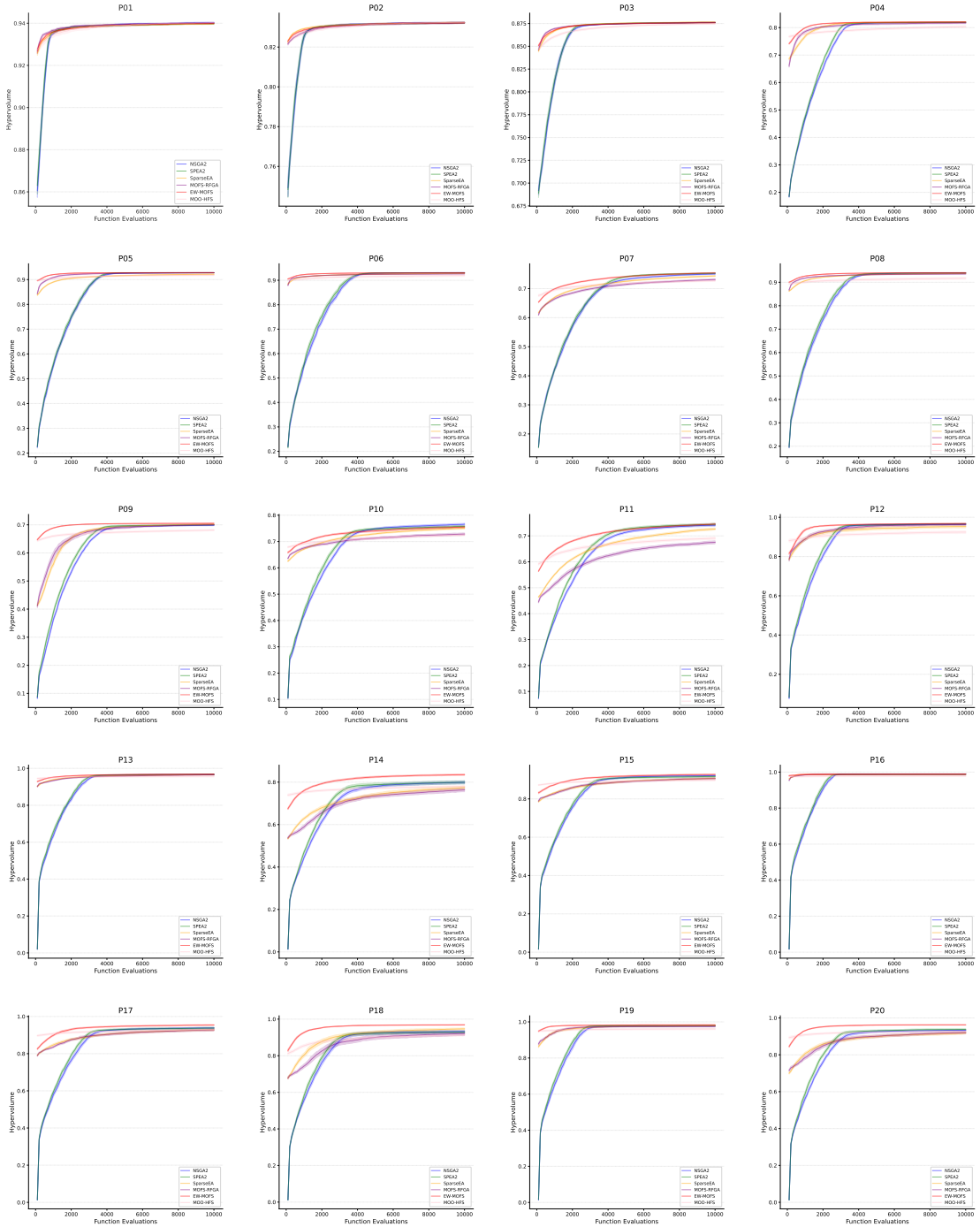


Figure A.2 – Hypervolume values obtained during the execution algorithm on the twenty datasets.

APPENDIX B — RESUMO EXPANDIDO

Na era dos dados, sistemas baseados em aprendizado de máquina influenciam diretamente escolhas individuais (por meio de influência nos sistemas de compras, redes sociais e até sistemas de manipulação de opiniões públicas). Para esses sistemas de ML, que já são parte essencial do sistema econômico e social atual, um dos principais desafios é o crescimento exponencial de dados sendo coletados, chamado de Big Data. Dados frequentemente possuem um alto número de variáveis (chamadas de features), o que acarreta na maldição da dimensionalidade, impactando diretamente o custo computacional, desempenho e interpretabilidade desses sistemas.

Existem técnicas conhecidas de redução de features, que podem ser classificadas em extração e seleção. Enquanto técnicas de extração (PCA e t-SNE, por exemplo) transformam os dados em espaços latentes, perdendo a interpretabilidade das variáveis originais, a seleção de features preserva a semântica original, sendo essencial para promover modelos interpretáveis, além de reduzir custos computacionais e habilitar modelos mais precisos. Atualmente, métodos de seleção de features podem ser separados em quatro categorias:

- *Métodos Filter* avaliam a relevância das features independente de classificadores, normalmente através de medidas estatísticas univariadas ou multivariadas, oferecendo baixos custos computacionais.
- *Métodos Embedded* integram a seleção de features no treino do classificador, mas têm um custo computacional mais elevado do que os métodos de filtro, com a vantagem de poder identificar interações entre features e com o modelo.
- *Métodos Wrapper* utilizam algoritmos de busca para avaliar subconjuntos de features através do desempenho obtido ao interagir com um modelo, o que resulta em custos computacionais elevados devido ao crescimento exponencial da quantidade de combinações.
- *Métodos Híbridos* são uma combinação de várias abordagens, tais como uma pré-seleção de filtros seguida de um refinamento de wrappers, formando um pipeline combinado.

Dentre essas abordagens, destacam-se os métodos híbridos, pois podem combinar as vantagens de dois métodos. Esse trabalho propõe um novo método híbrido que consiste em uma etapa de pré seleção com método embedded, para obter conhecimento geral sobre

o problema, e então utiliza esse conhecimento para guiar uma segunda etapa, onde a busca é feita utilizando um método wrapper.

Abordagens multiobjetivo para seleção de features formulam o problema como a otimização simultânea de dois objetivos conflitantes: minimizar o erro de classificação e o número de features selecionadas. Dessa forma, é encontrado um conjunto de soluções pertencente à fronteira de Pareto, que representa os trade-offs entre menos features selecionadas e maior performance preditiva. Cada solução de Pareto é um subconjunto de features não dominado, isto é, não existe outra solução que apresente menor erro sem aumentar a quantidade de atributos, nem vice-versa.

Para explorar de forma eficiente esse espaço de soluções, geralmente são utilizados algoritmos de busca heurística. Algoritmos evolutivos, em especial o NSGA-II, são amplamente utilizados e, por meio de ordenação não-dominada e métricas de crowding distance, mantêm tanto a convergência quanto a diversidade dos indivíduos na população. Existem trabalhos relacionados que utilizam o NSGA-II como base, são eles o SparseEA, MOFS-RFGA e MOO-HFS.

No contexto de problemas de classificação, enquanto métricas tradicionais como acurácia, sensitividade, especificidade e precisão, são utilizadas para avaliar a qualidade do modelo. Para esse trabalho, a métrica escolhida foi o F1-Score, pois consegue avaliar de forma equilibrada problemas com classes desbalanceadas, e sua implementação também tem suporte a multiclasse.

São necessárias também métricas que avaliem soluções multiobjetivo, para que os conjuntos de features encontrados sejam avaliados, em relação à qualidade da aproximação da fronteira de Pareto. Uma métrica amplamente utilizada é o hipervolume, que mede o volume do espaço objetivo dominado pelas soluções encontradas em relação a um ponto de referência escolhido. O hipervolume foi escolhido nesse trabalho como métrica de avaliação da qualidade das soluções finais de cada método testado.

Esse trabalho propõe o algoritmo EW-MOFS, que consiste em um novo método híbrido de seleção de features multiobjetivo, que funciona combinando um vetor de pesos obtido de uma técnica embedded (Random Forest) e o utiliza para orientar e otimizar o espaço de busca do wrapper. O EW-MOFS se propõe a resolver resolver algumas lacunas deixadas na literatura por trabalhos similares baseados em NSGA-II encontrados, como a utilização de torneio binário na inicialização e operações genéticas, e a falta de benchmarking com problemas de maior dimensionalidade (SparseEA e MOFS-RFGA incluem testes com até 1.000 e 2.000 features, respectivamente, enquanto esse trabalho apresenta

casos de teste com 54.000 features). O algoritmo MOO-HFS também deixa algumas lacunas, sendo a principal entre delas é a falta de testes comparativos com outros métodos de seleção de features do tipo Wrapper ou Híbrido.

O EW-MOFS utiliza o framework do NSGA-II como base, modificando as funções de inicialização e operações genéticas para orientar a busca em direção aos valores de relevância armazenados no vetor de pesos, obtidos com o modelo Random Forest no início da execução. Durante a inicialização, a seleção dos genes adicionados é realizada de forma aleatória ponderada com base no vetor de pesos obtidos. Durante as operações genéticas, os genes adicionados na mutação também passam pelo processo de seleção aleatória ponderada. Utilizando o vetor de pesos, é possível obter populações iniciais de melhor qualidade e guiá-las de forma otimizada para regiões de maior relevância.

Os experimentos utilizados para validar o método são feitos em 12 datasets de repositórios abertos de Machine Learning, e 8 datasets de alta dimensionalidade do CuMiDa. Apesar de o principal objetivo do trabalho, desenvolver um método resistente a dados de alta dimensionalidade, alguns datasets menores também foram incluídos para validar a robustez do EW-MOFS. Dentre os datasets escolhidos, o número de features varia de 30 a 54.000.

EW-MOFS foi comparado com dois métodos wrapper: NSGA-II e SPEA-II, e três métodos híbridos: SparseEA, MOFS-RFGA e MOO-HFS, todos com parâmetros similares, mesma quantidade de indivíduos e gerações. Cada dataset foi testado com cada algoritmo 31 vezes. O EW-MOFS obteve desempenho estatisticamente igual ou superior em 18 dos 20 conjuntos de dados: foi superado apenas em P03 (em relação ao SparseEA e ao MOFS-RFGA) e em P10 (em relação ao NSGA-II). Em particular, nos oito conjuntos de maior dimensionalidade (P13–P20, de 22.277 a 54.675 features), o EW-MOFS conquistou consistentemente os maiores HVs, destacando sua robustez em cenários de alta esparsidade e dimensionalidade extrema.

Para trabalhos futuros, sugere-se explorar outras técnicas embedded (por exemplo, gradient boosting ou redes neurais) para gerar os pesos, avaliar o desempenho em cenários ultra-alta dimensionalidade (>100.000 variáveis), implementar ajuste adaptativo dos pesos durante a otimização e estender o framework para tarefas de regressão.