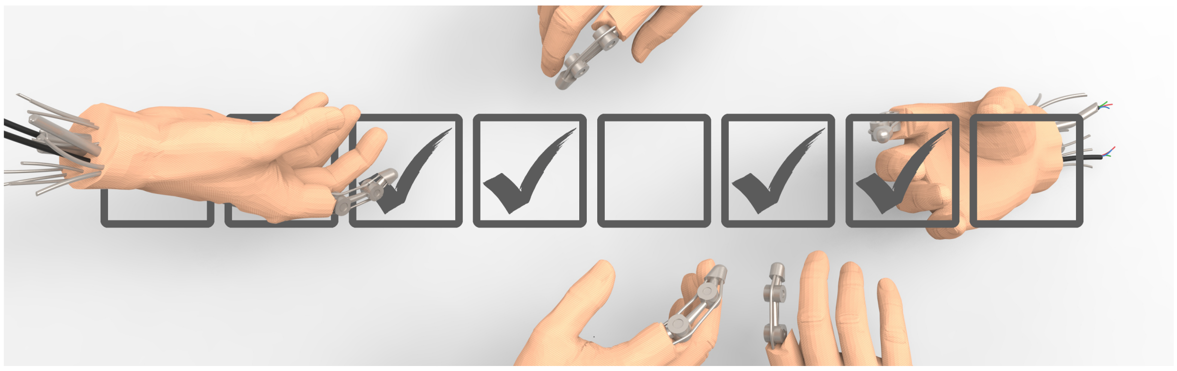


Multi/Many-Objective Optimization in Feature Selection

Duarte Miguel Côrte-Real Ramalho Rolim

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Susana Margarida da Silva Vieira
Prof. João Filipe Pinto Ribau

Examination Committee

November 2017

The struggle itself towards the heights is enough to fill a man's heart. ***One must imagine Sisyphus happy.***

Albert Camus

Acknowledgments

This thesis embodies the conclusion of my long academic life. I'm fully aware that without guidance I would not be in this position, so it seems only fair to honour those who have guided me to this important milestone.

Foremost I feel obliged to show gratitude to my parents, for their never-ending crusade in my education, and their inexhaustible support. This sentiment extends to my siblings, my biggest allies.

To the rest of my family, I'm thankful for their interest, fellowship, and example.

For the friends I've held along these years, I'm sincerely grateful for their motivation, ability to distress me and the good times in these bohemian years.

To Beatriz, my accomplice for the last four years, I am forever thankful for her presence, support, and patience, but most importantly for her persistence.

I also recognize Prof. Susana Vieira and my research area colleagues for their valuable insights to this work.

Lastly, to all anonymous or forgotten people, including professors and classmates, that influenced and guided me directly or indirectly throughout these 23 years, their influence is not forgotten.

Abstract

Feature selection, the removal process of non-essential variables in a dataset, is a crucial step in any machine learning algorithm since it not only simplifies the model but also increases the predictor's performance. However, admitting that the removal of unnecessary features does not improve all performance metrics simultaneously, different applications require distinct classifier's performance metrics. Additionally, it might be advantageous to use diverse metrics for the process of finding good feature subsets.

This work has three major contributions relating to binary classification using a wide-set of wrapper performance metrics and multi-class classification dividing it into several binary sub-problems: firstly, a relationship analysis between wrapper's performance metrics is made, comparing and conjecturing which are made redundant by each other; The second contribution is a first study on the sets of classifier's performance metrics' performance in feature selection, testing if the inclusion of more than 2 objectives is beneficial; Lastly, a feature selection decision interface was built, which aids in the solution selection process.

The first analysis shows that less than a handful of the tested performance metrics for binary classification is not simultaneously improved in the feature selection process. Using state of the art multi-objective algorithms, results suggest a better performance, in terms of convergence and diversity, of feature selection when using a high number of objectives in binary classification, despite some being redundant. In relation to multi-class classification, only diversity is improved when dividing it into several binary sub-problems and using accuracy to each one.

Keywords

Feature Selection; Wrapper Evaluation; Evolutionary Computation; Multi-Objective Optimization; Decision Interface.

Resumo

A seleção de variáveis é um passo crucial em qualquer algoritmo de inteligência artificial, que não só simplifica o modelo, mas também aumenta a capacidade preditiva. Admitindo que a remoção das variáveis desnecessárias não melhora simultaneamente todas as medidas de performance, aplicações diferentes exigem medidas distintas. Adicionalmente pode ser vantajoso utilizar várias métricas no processo de otimização.

Este trabalho lida com classificação binária utilizando várias métricas diferentes, e também com classificação de múltiplas classes, dividindo o problema em sub-problemas binários, e pode ser dividido em três grandes contribuições: a análise da relação entre medidas de performance dos classificadores, identificando redundância; a comparação do desempenho do processo de otimização utilizando diferentes conjuntos de métricas; Prototipagem de uma interface de decisão que permite examinar detalhadamente todas as soluções disponíveis.

Os resultados mostram que, em classificação binária, das 9 métricas analisadas somente entre 4 a 5 são não-redundantes, enquanto que problemas de múltiplas classes divididos em sub-problemas binários têm a totalidade de objetivos não-redundantes. Algoritmos do estado da arte de otimização multi-objectivo foram aplicados ao problema de seleção de variáveis, e os resultados sugerem uma melhoria de performance, em termos de convergência e diversidade, ao utilizar um maior número de objetivos em classificação binária, mesmo se redundantes. Em relação à classificação com múltiplas classes, embora não haja melhorias em termos de convergência, a diversidade de soluções é melhorada quando se divide o problema em vários sub-problemas binários e se o utiliza a exatidão para cada.

Palavras Chave

Seleção de Variáveis; Avaliação *wrapper*; Computação Evolucionária; Otimização Multi-Objetivo; Interface de Decisão.

Contents

1 Introduction to Feature Selection **1**

1.1 Search Procedure . . . 4

1.2 Evaluation Methods . . . 5

1.3 Motivation . . . 6

1.4 Contributions . . . 7

1.5 Thesis Outline . . . 8

2 Classification in Feature Selection **9**

2.1 Feature Selection Datasets . . . 10

2.1.1 Dataset Description . . . 10

2.1.2 General Procedure . . . 11

2.2 Machine Learning Algorithm . . . 12

3 Metrics in Wrapper Feature Selection **15**

3.1 Performance Metrics in Wrapper Classification . . . 16

3.2 Objective relationship . . . 17

4 Search Procedure **25**

4.1 State of the Art . . . 26

4.1.1 Concepts . . . 26

4.1.2 Methodologies . . . 27

4.2 Algorithms . . . 29

4.2.1 NSGA-II . . . 31

4.2.2 NSGA-III . . . 32

4.2.3 MOEA/D . . . 33

4.2.4 PICEA-g . . . 34

4.2.5 HypE . . . 35

4.3 Multi-Objective Optimization Benchmark Problems . . . 36

4.4 Performance Metrics for Multi-Objective Algorithms . . . 39

4.4.1 Convergence . . . 41

4.4.2 Diversity . . . 41

4.4.3 Convergence and diversity . . . 42

4.5 Search Algorithms Validation . . . 42
4.6 Encoding for feature selection . . . 44
4.6.1 Individual Representation . . . 44
4.6.2 Initial Population, Size and Stopping Criteria . . . 45
4.6.3 Genetic Operators . . . 46
4.6.4 Algorithms' parameters . . . 47

5 Results **49**
5.1 Algorithm's Validation for Feature Selection . . . 50
5.2 Metrics for Wrapper based Feature Selection . . . 51
5.2.1 Binary classification . . . 51
5.2.2 Multi-class classification . . . 53
5.3 Visual comparison . . . 56
5.4 Multi/Many-Objective Algorithms in Feature Selection . . . 56
5.4.1 Binary Classification . . . 57
5.4.2 Multi-class Classification . . . 60

6 Decision Interface **63**

7 Conclusions and Future Work **69**
7.1 Summary . . . 70
7.2 Future Work . . . 70

Bibliography **71**

List of Figures

1.1 Feature selection versus feature extraction 2
1.2 Feature selection advantages. 3
1.3 Characteristics of feature selection algorithms. 3
1.4 Wrapper approach using randomized search. 6

2.1 General procedure in subset evaluation. 12
2.2 Processing time comparison between classifiers. 13

3.1 Exemplary multi-class classification decomposition into binary sub-problems. 17
3.2 Relationships between objectives 18
3.3 Number of features versus accuracy for the Spectf dataset. 19
3.4 Selected features for each dataset 19
3.5 Number of different best solutions. 20
3.6 Parallel coordinates plot 21
3.7 Parallel plot of MOP problem. 22
3.8 Kendall correlation to binary datasets. 23
3.9 Kendall correlation averaged. 23
3.10 Kendall correlation to multi-class datasets using accuracy in each class. 24

4.1 Design and objective space and their mapping. 27
4.2 Solutions in a minimization bi-ojective problem. 27
4.3 Optimum concept difference. 28
4.4 Selection mechanisms in MOP. 29
4.5 Ratio of non-dominated solutions depending on the number of objectives. 30
4.6 Proceedings of NSGA II. 31
4.7 Random population in all three problems: DTLZ4, DTLZ7 and MOKP. 39
4.8 NSGA-II evolution in DTLZ4,DTLZ7 and MOKP with M=2 39
4.9 All algorithms final population for each problem using M=2 39
4.10 All algorithms final population for each problem using M=10 40
4.11 Three different example solutions 40
4.12 Two example solutions for a bi-objective problem. 41
4.13 Time behaviour of the algorithms in the benchmark problems. 43

4.14 Hypervolume behaviour of the algorithms. 43
4.15 Spacing behaviour of the algorithms. 44
4.16 Set coverage behaviour of the algorithms. 45
4.17 Two point crossover 46
4.18 Bitwise mutation. 46
4.19 Sensitivity analysis in MOEA/D. 48
4.20 Sensitivity analysis in HypE. 48
4.21 Sensitivity analysis in PICEA-g. 48

5.1 Evolution of solutions in HypE applied to binary feature selection using Spectf dataset. 50
5.2 Evolution of solutions in HypE applied to multi-class feature selection using DNA dataset. 50
5.3 Hypervolume comparison of wrapper's performance metrics for binary classification. . . 52
5.4 Spacing comparison of wrapper's performance metrics for binary classification. 53
5.5 Set coverage comparison of wrapper's performance metrics for binary classification. . . . 54
5.6 Hypervolume comparison of wrapper's performance metrics for multi-class classification. 55
5.7 Spacing comparison of wrapper's performance metrics for multi-class classification. . . . 55
5.8 Set Coverage comparison of wrapper's performance metrics for multi-class classification. 56
5.9 Suggested approach versus traditional approach and mRMR. 57
5.10 Running time performance of the algorithms in binary dataset Spectf. 58
5.11 Hypervolume performance of the algorithms in binary datasets. 58
5.12 Spacing performance of the algorithms in binary datasets. 59
5.13 Set coverage performance of the algorithms in binary datasets. 59
5.14 Running time of the algorithms in multi-class dataset Vehicle. 60
5.15 Hypervolume performance of the algorithms in multi-class datasets 61
5.16 Spacing performance of the algorithms in multi-class datasets. 61
5.17 Set coverage performance of the algorithms in multi-class datasets. 62

6.1 NSGA-II's exemplary final population of Spectf dataset. 64
6.2 Decision Making Interface for Feature Selection. 65
6.3 Draggable lines example. 65
6.4 Clicking examples. 66
6.5 Weighted sum example. 66

List of Tables

1.1 Exhaustive search subset evaluations depending on the number of features. 4

2.1 Binary classification datasets. 10

2.2 Multi-class classification datasets. 11

3.1 Nine binary classifier's performance metrics. 16

3.2 Results to each sub-problem of Figure 3.1 calculating accuracy and recall to each class. 17

3.3 Sets of metrics used in binary classification. 24

4.1 MOP for validation and their characteristics 38

4.2 MOP algorithms performance indicators. 40

4.3 Indices results to the solutions presented in Figure 4.12 42

4.4 Parameters values 48

List of Acronyms

AI Artificial Intelligence

DM Decision Maker

DT Decision Tree

DTLZ Deb, Thiele, Laumanns and Zitzler

EA Evolutionary Algorithm

EC Evolutionary Computation

FP False Positives

FN False Negatives

GIGO Garbage In, Garbage Out

KDD Knowledge Discovery in Datasets

MCDM Multi-Criteria Decision Making

MOKP Multi-Objective Knapsack Problem

MOP Multi-Objective Problems

TP True Positives

TN True Negatives

1 Introduction to Feature Selection

Contents

1.1 Search Procedure 4
1.2 Evaluation Methods 5
1.3 Motivation 6
1.4 Contributions 7
1.5 Thesis Outline 8

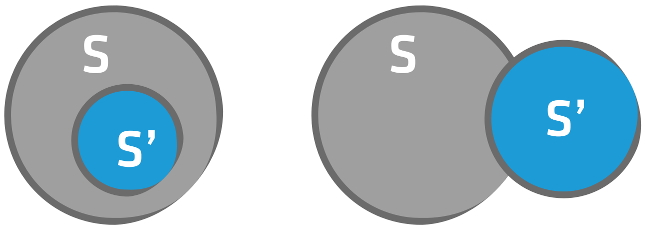
Society rapidly evolves towards a future where computers will be able to act and react without being explicitly programmed to do so. Hence, computers should be able to acquire data and learn with it, becoming able to predict everyday life's situations.

Artificial Intelligence (AI) algorithms are already being applied to innumerable quotidian problems, such as social media, internet security, and market analysis, working in cooperation with a Decision Maker (DM). These algorithms are usually trained by example, i.e., data is acquired to train a model/algorithm.

Nowadays data collection and storage are available effortlessly, allowing researchers and scientists to gather and store enormous datasets, frequently prioritizing the storage of all variables, disregarding their importance. There are also applications where variable relevance is unknown *a priori* and all variables end up being collected. Consequently, most datasets became contaminated with redundant, noisy or simply not relevant variables. The data collection and storage evolution was of such magnitude, that only two decades ago a dataset with more than 20 variables was considered large-scale [1], while nowadays that designation is used when dealing with thousands of variables.

Garbage In, Garbage Out (GIGO) principle states that the quality of the output data is limited by the input's condition. Therefore, to "clean" input data is of crucial relevance in data mining, and is included as one of the five main steps in Fayaad's[2] Knowledge Discovery in Datasets (KDD) process.

The process of removing unnecessary variables from data with the intention of finding an optimal set is called feature selection. Not to be confused with feature extraction, a similar process, frequently combined with feature selection, in which features are combined and modified to create new ones. The difference is illustrated in Figure 1.1, in which S is the original set of features and S' is the final.



(a) Feature selection **(b)** Feature extraction

Figure 1.1: Feature selection versus feature extraction

There are several advantages to this exclusion, some of which presented in Figure 1.2. Less features results in a simpler, more transparent model, quicker to train and test, and easier to understand. Additionally, removing redundant and noisy variables improves performance by increasing the predictive value of data, avoiding the curse of dimensionality [3].

As an example, consider an algorithm designed to classify an individual's gender through a camera. Many variables can be accessed fairly easily by image. The person's height, hair length, and presence of facial hair are most likely useful since women tend to be less tall, have longer hair, and non-existent facial hair. But adding a variable such as height in other unit, or facial hair's length might be redundant since they are both highly correlated to height and presence of facial hair, respectively, making them redundant variables that add no information to the dataset, and will most likely

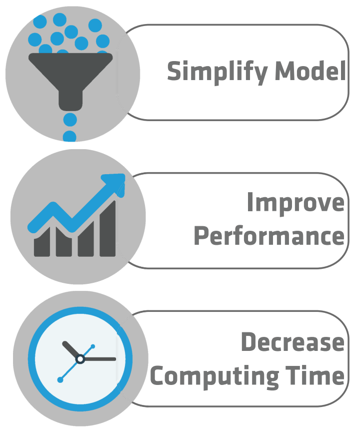


Figure 1.2: Feature selection advantages.

deteriorate the performance of a classifier that isn't able to discover that redundancy. Moreover, if the woman instances of the training set (in supervised learning, where training data is labelled) are mainly composed of a basketball female team, height becomes an inferior feature, despite not appearing so.

Feature selection is a complex duty not only considering the enormous search space for medium-sized datasets, which makes feature selection an NP-hard problem [4], but also due to the intricate interactions between features. These interactions might make features useless by themselves be beneficial when paired with others, and apparently redundant ones can also be advantageous when combined [3].

Any feature selection algorithm can be characterized by two main traits: a search procedure, and an evaluation method. The first is simply the operation of searching for solutions, and the latter is how each solution is evaluated. These characteristics and their main categories are indicated in Figure 1.3, which will be briefly discussed. For further information the reader is referred to [3].

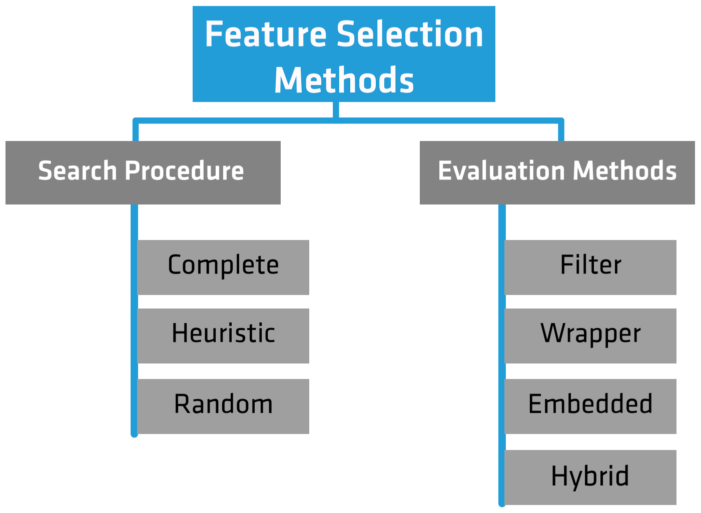


Figure 1.3: Characteristics of feature selection algorithms.

1.1 Search Procedure

As mentioned in Figure 1.3, there are three types of search procedures, i.e., algorithms created to hunt the best feature subset, according to some metric.

Exhaustive

This brute-force method tries each possible feature subset. Depending on the number of features N, and since at least one feature must be chosen, $2^N - 1$ subsets must be evaluated. It might be a good approach to very small problems, however, if the problem has a reasonable number of features the amount of combinations to evaluate is colossal, as illustrated in Table 1.1, making this approach simply impractical.

Table 1.1: Exhaustive search subset evaluations depending on the number of features.

Nr. of Features	Combinations
3	7
5	31
10	1023
15	32767
20	1.05E+06
30	1.07E+09
50	1.13E+15
100	1.27E+30

Heuristics

The simplest response to exhaustive search' weakness is to apply heuristics. In each generation, the features yet to be selected (rejected) are considered for selection (rejection) [5]. These are usually computationally faster needing fewer evaluations than exhaustive search, but tend to be stuck in local optimums. Their simplicity and robustness in more adequate for a quick analysis, making it, undoubtedly, the most used search procedure.

Examples include greedy algorithms such as sequential forward selection (SFS)[6] or sequential backward selection (SBS) [7]. Relief [8] is also a notorious example.

Meta-heuristics

Randomized methods or meta-heuristics are black box algorithms heavily based on a biological or physical process, and traditionally using a population of solutions in each iteration, yielding several solutions. These are famous worldwide for their ability to escape local optimums, in which heuristics often get stuck.

Nevertheless, they're slower and more complex, and considering their stochasticity, different runs will likely not result in the same population.

Evolutionary Computation (EC) applied to feature selection has been around for more than a decade, but only recently with datasets' growth has it became popular [9].

1.2 Evaluation Methods

A search procedure expects a solution evaluator, i.e., a fitness function to assign each subset and be able to compare them. There are four main types: Filter, wrapper, embedded and hybrid.

Filter

This is the simpler kind of performance metric. Essentially each feature or subset is assigned a score, traditionally based on correlation, mutual information, p-statistic or a similar statistical value, and the top n ones are included. Single feature ranking does not consider feature interaction, and consequently adopt redundant ones. As a consequence, subset filter ranking as a whole has become popular, using metrics that take into account the relationship between features, such as mRMR [10].

Despite being simple and extremely fast, its downsides include the difficulty in finding features' interaction, in minimizing redundancy between features and in incorporating features less relevant by themselves but important when joined with another one. As proved in [3], this is a real concern. Additionally, a classifier has no input in the decision, and some features important for some classifiers might be disregarded.

Wrapper

In wrapper methods, a subset's score is the performance of a classifier trained and tested using only the features included in that subset. What type of classifier and what performance to use are problem and user dependent.

These are quite slow, because they involve the training and testing of a machine, but usually show better results than filter methods, despite the possibility of over-fitting, being reactive to the intricate interactions between feature.

Embedded

The third type of methods combines the strengths of both filter and wrapper approaches, but also its weaknesses. A model is trained and tested, while simultaneously being able to understand which features best contribute to the model's performance. Therefore the training and feature ranking part of the method cannot be separated.

One of the most notorious examples of embedded algorithms are regularization methods such as Lasso [11]. They are complex and heavier than filter and wrapper approaches, and also more recent.

Hybrid

Finally, hybrid methods use filter and wrapper approaches independently. Firstly, a combination of filter methods are used to reduce the original set of features, and then wrapper evaluation is implemented to the remaining features to search for the best subset.

1.3 Motivation

Despite slower and more complex, or maybe as a consequence of it, combining **randomized search** procedures, or meta-heuristics, with **wrapper evaluation** is the most promising approach to feature selection in the current state of technology. Randomized search is undeniably the best approach to successfully explore such an immense search space, for its stochastic nature and ability to avoid local optimums, and is known for its high performance in NP-hard problems. Wrapper approach to feature subset evaluation is the most reliable one, despite the computational costs, being the only approach truly reactive to feature interactions. The general procedure is roughly illustrated in Figure 1.4.

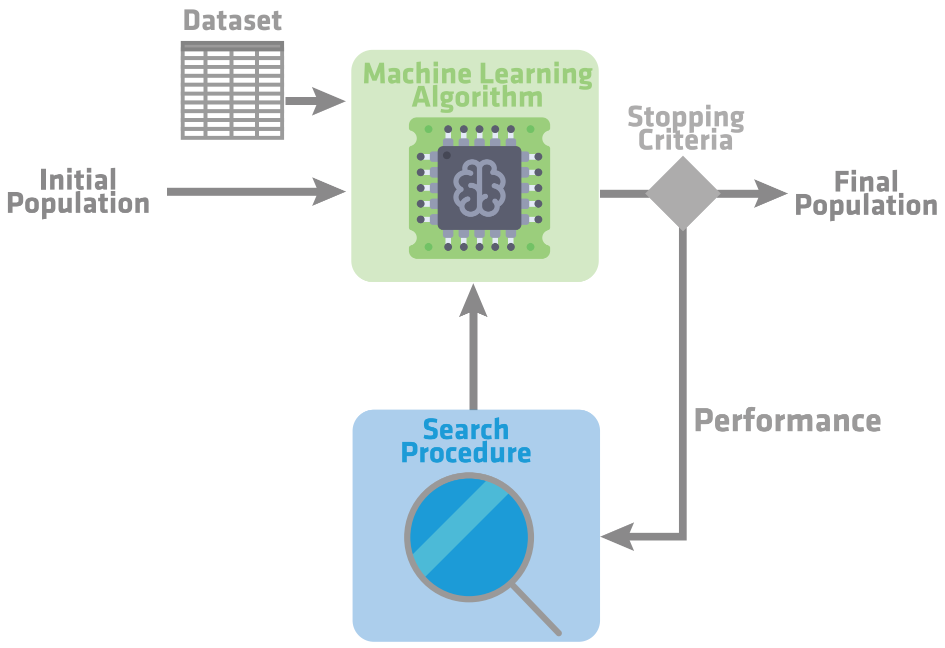


Figure 1.4: Wrapper approach using randomized search.

Aside from classifier's performance, it's evident that when considering two feature subsets with different sizes but same exact performance, the one with less number of features is most valued. This alone changes the problem's nature, adding another goal for the optimization process, but a deeper thought about how performance is evaluated changes it further. Consider the following three examples:

A medical diagnostic application is designed to predict whether a patient has a malignant tumour, based on physiological variables. It's significantly more important to reduce the number of wrongly diagnosed sick patients than to diagnose healthy ones as sick, i.e. reducing the number of False Negatives (FN) is the priority over False Positives (FP). Nevertheless, the DM might want to still control FP.

The second example is an image search in a typical search engine such as Google or Bing. While searching for kittens images, a user might be irritated if a dog image shows up by mistake. Another less sensitive user might allow dogs to pop up if its guaranteed all pictures containing cats are presented. Therefore, in this example, the evaluation of each solution is user dependent, and that user preference is only known *a posteriori*.

The last example, this time concerning multi-class classification, is an algorithm built to help in the decision making of assigning a grade G to a thesis $G \in \{1, 2, 3, ..., 18, 19, 20\}$. Depending on the user, it might only want the predictor's opinion if $G > 17$, or to confirm a negative. In that case, only the classifier's performance in relation to specific classes is interesting.

These examples highlight the importance to give the DM possibility to apply their own preferences while choosing the most suitable feature subset. Moreover, the second and third examples show the importance of doing this only after the optimization process, requiring several solutions to be found and presented.

While it may seem like reducing the number of redundant, noisy and irrelevant features might increase all classifier's performance metrics, there might exist features suitable only for specific metrics. In the case of different performance metrics requiring distinct feature subsets, including several metrics in the optimization process, might improve the process, reaching better and more diverse options to supply the user.

There's a need to answer these questions, opening way to future researchers in this area, and further investigate the application of multi and many-objective algorithms to feature selection studying whether there are advantages in using more than a single performance metrics.

Background

As mentioned above, using EC in feature selection with wrapper evaluation is dated back in the late 80's [12] but has only gained popularity in the last decade with the ever-growing size of datasets. For an extended review of this field, alongside with feature selection introduction, the reader is referred to [9].

The vast majority of research uses single-objective algorithms, in which only a performance measure is optimized, but there are plenty of multi-objective examples. Generally, the multi-objective problem is stated as bi-objective, minimizing the number of features/classifier's complexity and maximizing the classifier's accuracy or other performance measure. Notorious examples in binary classification are [13–16]. A similar analysis was made in [17] for multi-class classification, which uses the number of features and the Hamming loss function.

Using more than two objectives for binary classification feature selection was proven to be of great interest when [18] shown that using three objectives (number of features, specificity, and recall) reaches better solutions in comparison to only using two (number of features and accuracy). However, that investigation was not further explored.

Despite works such as [19] for binary classification, and [20] for multi-class classification, have used many objectives, the first using recall, precision, accuracy, F1-score and Cohen's kappa, and the second diving the problem into binary sub-problems and using accuracy in each, their analysis is somewhat limited since no comparison proving its benefits was made.

1.4 Contributions

This work has three major contributions, relating to binary and multi-class classification: firstly, a relationship analysis between wrapper's performance metrics is made using multi and many objective optimization algorithms, comparing and conjecturing which are made redundant by each other; The second contribution is a first study on which classifier's performace metrics combination improve the searching procedure in feature selection; Lastly, a feature selection decision interface is iterated, which allows the decision maker to easily study all solutions population and aids in the selection process.

1.5 Thesis Outline

The reader can expect a complete analysis of wrapper feature selection in this work. After this small introduction to the feature selection conundrum, chapter 2 a classifier is selected along with datasets to represent binary and multi-class classification problems. In chapter 3 a deep study about performance metrics in wrapper classification unfolds, analysing redundancy, and the most promising metrics combinations are selected.

Thereafter, the field of multi-objective optimization is introduced in chapter 4. Five state of the art algorithms are selected for analysis, along with performance metrics to compare their solutions. Following a short validation of these algorithms in benchmark problems, each algorithm's parameters and general ones are discussed and selected.

The crucial chapter, chapter 5 finally compares the use of the different classifiers' performance metrics sets, discussed in chapter 3, in multi-objective feature selection, concluding about which set is more advantageous both for binary and multi-class classification. A comparison between algorithms is done to select the most advantageous for the two kinds of problems.

Finally, in chapter 6, a first version of a decision-making interface for feature selection is presented, which supports the DM into selecting the ideal feature subset according to his desired criteria.

2
Classification in Feature Selection

Contents

2.1 Feature Selection Datasets 10
2.2 Machine Learning Algorithm 12

The introductory chapter advocated the use of wrapper evaluation to rank each solution in supervised learning, which implies using a classifier and choosing datasets. Thus, some datasets to represent binary and multi-class classifications problems in the remainder of this work are chosen, followed by a look and selection at traditionally used classifiers in feature selection. Pre-processing of those datasets and the general subset evaluation procedure are also discussed.

2.1 Feature Selection Datasets

Any feature selection algorithm must be tested in data and for this purpose several were chosen, limiting the choice to binary and multi-class classification datasets. The main restrictions made in the dataset selection were the number of citations, prioritizing highly used datasets, and the number of features, choosing a minimum of 20 features, but having a wide range of sizes. Additionally, special care was taken to the number of instances, ensuring datasets have enough instances to successfully train a classifier.

2.1.1 Dataset Description

The set of binary and multi-class classification datasets are listed in Table 2.1 and Table 2.2, respectively, including general characteristics. Most of them are public and have been subject to studies for decades. Those were obtained via UCI's website [21].

Table 2.1: Binary classification datasets.

Name	**Features**	**Instances**	**% Positives**
AldoA	74	1434	89.1
Mushroom	112	8124	48.2
Musk	166	476	43.5
Phishing	68	11055	55.7
Sonar	60	208	46.6
Spectf	44	267	79.4

AldoA is a medical dataset, provided by an author's colleague. It's quite unbalanced and no knowledge about its features is known. This is the only dataset not publicly available and the name is derived from the colleague's name.

Mushroom dataset is composed of 22 attributes, about different characteristics of a mushroom such as cap shape and colour, and the idea is to predict if a mushroom is edible. Features are predominantly categorical, and the remaining are binary. Of these, variable #12 has missing data and therefore is not used. The categorical variables were then expanded into several dummy binary variables, totalling 112 binary features.

Musk dataset describes the shape of a molecule and some of its bonds characteristics. The goal is to predict if a new molecule is a musk, a type of aromatic substance, based on molecular properties. It's the biggest dataset considered, with 166 continuous features representing distances and displacements.

Any attempt to illegally access sensitive information online such as usernames, passwords, and credit card details is called phishing. Many websites are infected with this malware, to try to gather data form a visitor. Phishing dataset consists of 30 categorical variables about websites such as the presence of certain characters, url abnormality, popups, etc. These were expanded into 68 binary variables.

Sonar dataset's goal is to predict whether a sonar signal is bounced off a metal cylinder or a roughly cylindrical rock. It contains 60 continuous features describing the signal's energy within a particular frequency band.

The Spectf dataset is a collection of cardiac Single Proton Emission Computed Tomography, or SPECT, images, classified in normal or abnormal. The features consist of integer values from 0 to 100, representing regions of interest in the image. The 267 instances of the dataset were originally divided in 80 for training and 187 for testing, and this configuration was kept.

Table 2.2: Multi-class classification datasets.

Name	**Classes**	**Features**	**Instances**	**Unbalaced**
DNA	3	180	3186	Yes
Pendigits	10	16	10992	No
Satimage	6	36	6435	Yes
Vehicle	4	18	846	No
Vowel	11	10	990	No

DNA dataset, originally called Splice-junction Gene Sequences, is composed of three different classes, exons, ions or neither exons nor ions. It's composed of categorical variables that characterize if a DNA sequence, and have been expanded into 180 binary variables.

Pendigits, short for Pen-Based Recognition of Handwritten Digits, is a dataset that uses 16 integer features collected from 44 writers in order to predict the hand-written digit. Therefore, it has 10 classes, the digits from 0 to 9.

Satimage dataset consists of instances that represent a 3x3 square neighbourhood of pixels taken from a Landsat satellite image. each feature is an integer value between 0 and 255, although most features do not reach this range. The idea is to classify the soil in 6 different groups, using only that pixel information. All attributes are numerical.

The Vehicle dataset contains information about the 2D silhouettes characteristics such as compactness, maximum length, or circularity of a double-decker bus, Chevrolet van, Saab 9000 and an Opel Manta 400.

Finally, the vowel dataset is an attempt to predict what vowel a speaker is saying. There's no publicly available info about what the features represent.

2.1.2 General Procedure

The pre-processing of all datasets consisted of removing instances with missing values, expanding categorical data into several binary variables and normalizing features between $[-1;1]$, using Equation 2.1.

$$x'_i = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \tag{2.1}$$

Furthermore, datasets not originally divided intro training and testing, were divided using 75% - 25% ratio of the least prominent class, keeping the training data balanced, and the testing set with the same balance as the original dataset. Figure 2.1 represents this procedure.

This data division into training and test was chosen in detriment of cross validation for its simplicity. Cross-validation is more convoluted, and makes the process exponentially slower since it evolves training and testing a classifier several times with carefully chosen data. It is justifiable when one wants to maximize the accuracy of a particular classifier, which isn't the purpose of feature selection in itself, which aims at finding the best feature subset. Nevertheless, some risk of overfitting is unavoidable, diminished by the high number of existent instances in each dataset.

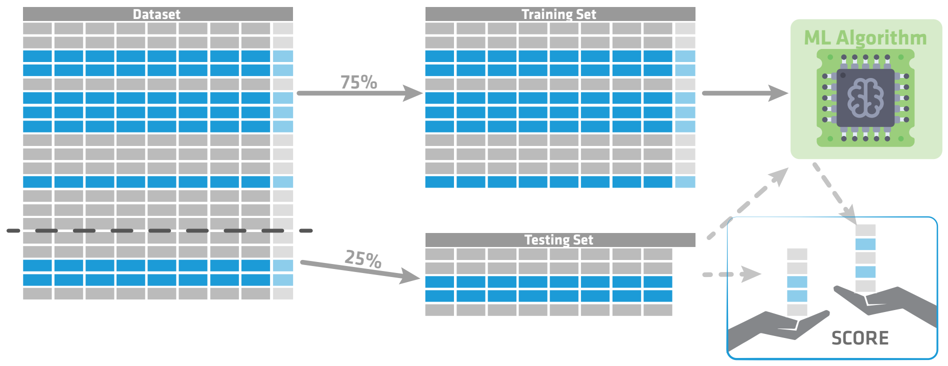


Figure 2.1: General procedure in subset evaluation.

2.2 Machine Learning Algorithm

A classifier, or a machine learning algorithm, is a method capable of learning by experience. Famous ones include random forests, neural networks and fuzzy logic. A good survey on classification methods can be found in [22].

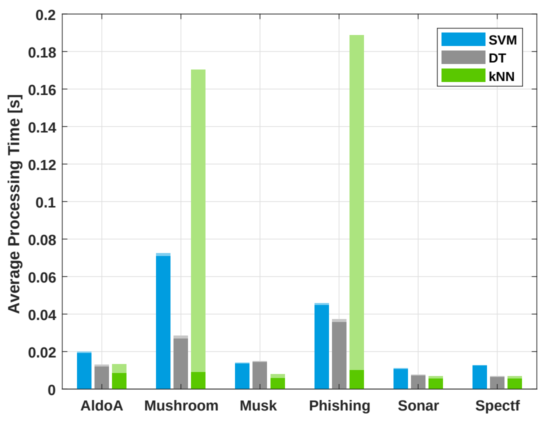
Considering that both the search procedure, later discussed, and subsets evaluation are naturally complex in feature selection, a concern to decrease performance evaluation processing time is essential. This is why algorithms such as artificial neural networks and fuzzy logic aren't considered in this work. Additionally, it can be argued that these kind of methods have intrinsic processes that have a similar effect as feature selection.

A comparison could be made in terms of classifier's performance, but the goal of feature selection isn't to achieve maximum performance during the process itself, but to be able to identify and react to feature's inherent interactions.

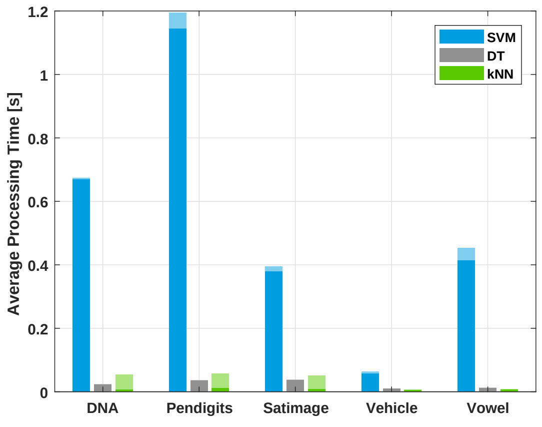
Traditionally, feature selection researchers use support vector machines (SVM), Decision Tree (DT) or k-Nearest Neighbours (kNN) [3, 23]. Naive Bayes is also an honourable mention.

Figure 2.2 compares the three in respect to time, tested in the binary and multi-class datasets presented above. The results are an average of 150 randomly generated feature subsets. The opaque fraction of each bar represents training time, and the rest is testing time. For binary classification, kNN predictor presents excessive testing time in mushroom and phishing datasets and therefore is not considered. Looking at the results in multi-class classification, SVM proves to be consistently slower, being impractical for the number of evaluations a meta-heuristic needs.

Since one of this work's goal is to use a global approach to the problem, DT classifiers are selected, for their consistency both on binary and multi-class classification.



(a) Binary datasets



(b) Multi-class datasets

Figure 2.2: Average processing time of each classifier. The opaque fraction of each bar represents training time alone, while the rest is testing time.

Decision trees are good classifiers considering two aspects: very accurate and relatively fast [24]. The type of decision tree used is CART [25], as a native function in MatLab.

3 Metrics in Wrapper Feature Selection

Contents

3.1 Performance Metrics in Wrapper Classification 16
3.2 Objective relationship 17

As mentioned in section 1.3, it's advantageous to present the DM with multiple metrics for the same feature subset *a posteriori*, allowing him to select and judge solutions based on his preferred metrics. In this chapter, typical metrics for both binary and multi-class classification are presented, followed by a discussion about subset structure of the best solutions for each metric to try understand if there are features only good for specific metrics, or if feature selection yields better performance according to all metrics.

The relationship between metrics is then discussed, in a effort to identify the redundant ones, that might be of no interest in multi-objective feature selection, along with a selection of different sets of metrics to further study.

3.1 Performance Metrics in Wrapper Classification

Foremost, and following the example of the vast majority of research mentioned in section 1.3, at least one solution evaluation metric should be based on the cardinality of the solution, i.e., how many features are used. This ensures that between solutions with the same performance, the one using fewer features is preferred. For that purpose, M_1 as defined in Equation 3.1 will be used. Note that this ensures $F_1 \in [0;1]$. Another option would be to measure the classifiers complexity, similarly to [14] which used DT's size.

$$\mathbf{M_1} = \frac{1 - sum(\mathbf{x})}{1 - dim(\mathbf{x})} \tag{3.1}$$

A binary classifier will output a binary vector. In supervised learning, this vector is compared to the target in order to test classifier's performance. Each prediction instance can be one of four results: True Positives (TP),True Negatives (TN),FP and FN, with meaning self explanatory. Many metrics using these four results can be used, some of the most notorious ones listed in Table 3.1.

Table 3.1: Nine binary classifier's performance metrics.

Name	Formula
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
Specificity	$\frac{TN}{TN+FP}$
F1-Score	$\frac{2TP}{2TP+FP+FN}$
Kappa	$1 - \frac{1-p_o}{1-p_e}$
NPV	$\frac{TN}{TN+FN}$
Matthews	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
Markdness	$Precision + NPV - 1$

In Cohen's kappa, p_o is similar to the accuracy metric, and p_e is a measure of random agreement [26].

Regarding multi-class classification, where models output integer vectors, F1-score and all other metrics listed in Table 3.1 can be adapted, either by creating one of each metric individually for each class or by averaging that result. There are also some well-known performance metrics specifically conceived for the comparison of integer vectors. A good assessment of these are discussed in [27], and include the Jaccard index and the Haming loss, see Equation 3.2 where D and L are the numbers of samples and of labels, respectively, and x_P is the predicted labels vector and x_T is the target vector. Using the number of features and the Hamming function will be considered the first approach.

$$\text{HammingLoss}(x_P, x_T) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{xor(x_{P_i}, x_{T_t})}{|L|}, \tag{3.2}$$

Traditionally multi-class multi-objective algorithms divide the problem in a binary sub-problem for each class and use accuracy individually in each sub-problem as objectives [20]. This division process is illustrated in Figure 3.1. This way an accuracy is calculated based on the obtained values of TP, FP, FN and TN to each class. Using accuracy assigns similar importance to true positives and true negatives, which seems to be inadequate considering the smaller number of positives than negatives in those sub-problems. Therefore, recall for each class is selected as a third approach to multi-class classification. The results for each class accuracy and recall in the problem of Figure 3.1 is listed in Table 3.2.

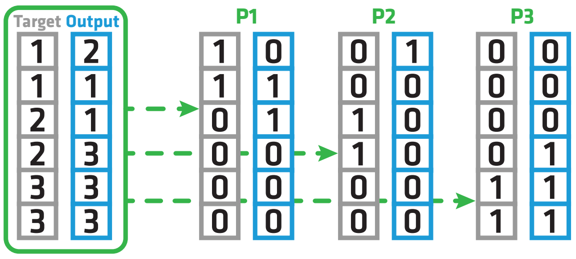


Figure 3.1: Exemplary multi-class classification decomposition into binary sub-problems.

Table 3.2: Results to each sub-problem of Figure 3.1 calculating accuracy and recall to each class.

	P1	**P2**	**P3**
Acc	$\frac{1+3}{6} \approx 0.67$	$\frac{3}{6} = 0.50$	$\frac{2+3}{6} \approx 0.83$
Rec	$\frac{1}{1+1} = 0.5$	$\frac{0}{0+2} = 0$	$\frac{2}{2+0} = 1$

As mentioned before, this work focuses on the relationship between these performance metrics and their applicability to feature selection. The first step in discussing which metrics to use is to analyse their relationship.

3.2 Objective relationship

Objectives relation is pairwise, meaning that it exists in pairs, and can be of three types [28], as shown in Figure 3.2.

Independent objectives are autonomous in their optimization, i.e., optimizing one does not affect the other one. If they do affect one another, then are considered dependent, which bisects in harmonious and conflictual. **Harmonious** objectives are improved or deteriorated simultaneously. The last relationship, **conflictual**, happens when while enhancing one objective the other is imperatively damaged as a side-effect.

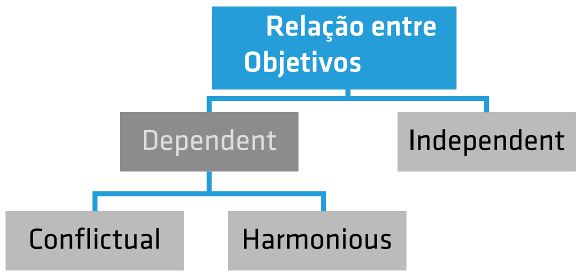


Figure 3.2: Relationships between objectives. Adapted from [28]

In the case of being independent, the objectives can be optimized separably by parallel algorithms, or even by two single-objective optimization algorithms. Harmonious objectives imply that one of them is redundant and can be removed. Nevertheless, inclusion does not necessarily harm the search [29], although algorithms can become biased towards those objectives. Another consequence of including unnecessary objective is the natural complexity added to the optimization procedure. Finally, conflictual objectives are the interesting ones, producing a trade-off surface filled with diverse solutions.

It is also worth mentioning that these relations are dynamic through the search space, p.e., two objectives might be conflictual in some part of the search space and harmonious in other.

Applying this to feature selection conundrum, the elimination of some features might improve or at least not degrade the classifier's performance, but further elimination will likely decrease it. Taking Sonar dataset as an example, 30000 randomly generated solutions with all spectrum of used features, Figure 3.3, shows advantages when slightly reducing the number of features, to between 10 and 20, but further reduction severely deteriorates performance. The curve obtained matches the theoretical ones stated in [30]. It's also interesting to verify in Figure 3.3 that the highest performance subset is composed of merely four features.

Therefore the relation between number of features and overall performance is initially harmonious but then shifts to conflictual.

But what specific metrics should be used in the performance evaluation of a classifier? Accuracy in unbalanced datasets or problems where priority is given to, p.e., TP, is not enough. Table 3.1 lists some of the binary performance indicators. At first sight, when a classifier is trained with the best features, all its performance metrics are improved. But this may not be the case. Nevertheless, it's evident that the relationship between all performance metrics will be either conflictual or harmonious, never independent.

To understand how structurally different the solutions preferred by each metric are, Figure 3.4 was created. It's a binary image, where each solution is a line and columns represent features. A blue

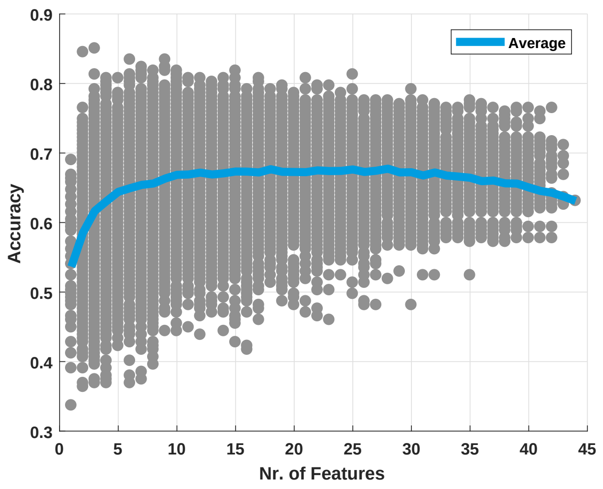
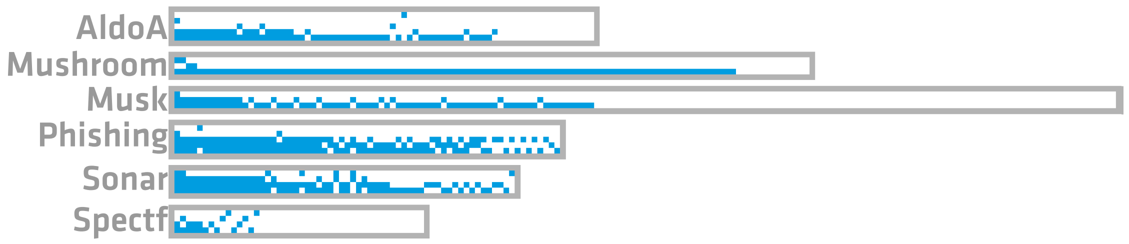


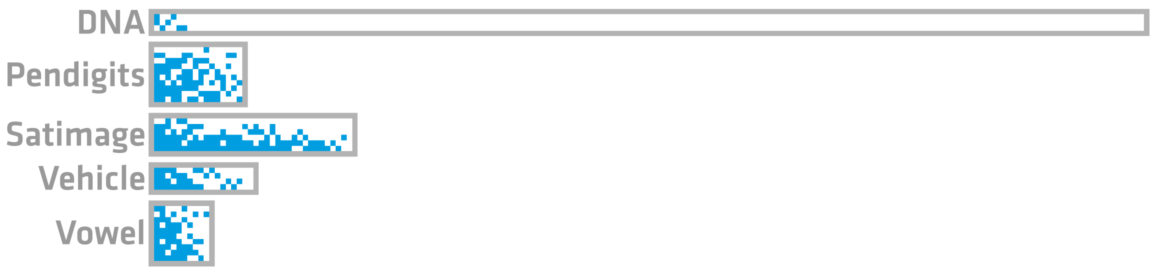
Figure 3.3: Number of features versus accuracy for the Spectf dataset. Made using 30000 randomly generated unique solutions.

square indicates that feature is selected, and a white square means it isn't. Features' position was rearranged for presentation purposes.

For each dataset, 300000 solutions were randomly generated (except for the Vowel which only has 1023 different possible solutions), and the best solution in respect to each one of the 9 metrics present in Table 3.1 for binary classification, and for the accuracy in each class for multi-class classification, was found. Duplicate solutions, resulting from different metrics having the same best solution, were eliminated, and only unique solutions were considered. Keep in mind 30000 solutions are, for most datasets, way less than 0.0000001% of all possible subsets.



(a) Binary datasets



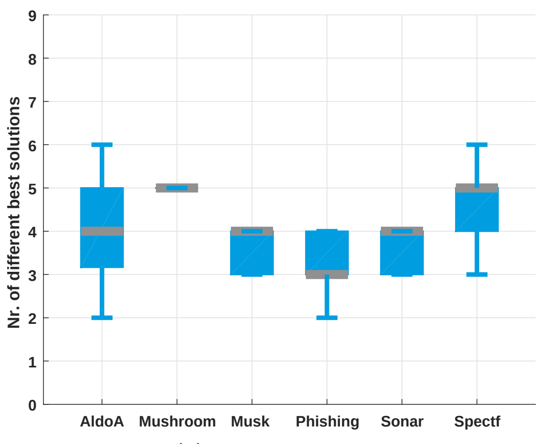
(b) Multi class datasers

Figure 3.4: Selected features for each dataset, considering 30000 randomly generated solutions.

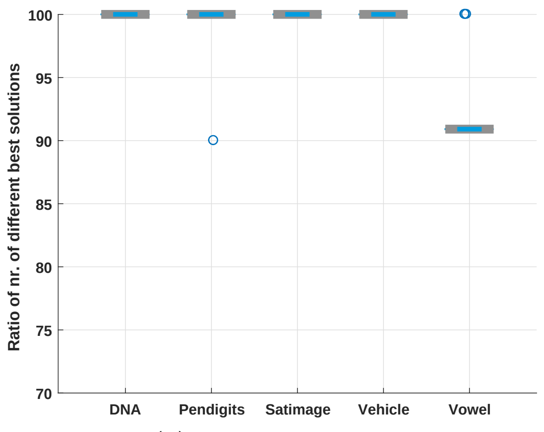
The first striking conclusion is that most features are not needed, proving the importance and justifying feature selection. Looking carefully at the results, it's visible that only Musk, Sonar and Vehicle datasets use at least one feature in all solutions, suggesting there might be "must have"features in those datasets. Nevertheless, the fact that in all other datasets that doesn't happen, suggests that metrics have a different preference of features. Furthermore, it's possible to see an intrinsic contrasting distribution of features. For example in Musk, with only three distinct solutions, the second and third have a common core of features and then use completely diverse one.

These results suggest that feature importance depends on the metric being considered, opposite to the assumption that good features improve the classifier in all metrics. Moreover, these results seem to prove the existence of features only suitable for specific metrics. Nevertheless, it's noticeable the low number of different solutions for binary datasets, which implies some of the metrics considered, presented in Table 3.1, have the same best solution, suggesting redundancy.

To further examine this redundancy issue, 20 random populations of 5000 individuals were created, and the number of different best solutions was calculated. Figure 3.5 is the box plot [31] obtained for both binary and multi-class datasets, the first using Table 3.1, and the latter using accuracy for each class. Harmony between some metrics is evident in binary classification, averaging at 4 or 5 the number of different solutions, while multi-class classification shows their conflictual nature, having almost 100% of conflictual objectives.



(a) Binary datasets



(b) Multi-class datasets

Figure 3.5: Number of different best solutions to each dataset. Made using 20 runs of a randomly generated population of size 5000. Multi-class dataset is normalized per number of objectives.

This analysis, although efficient in discovering redundancy, lacks identification of the redundant metrics. A quantitative analysis should, therefore, be made. Foremost, the concept of coordinates plot is introduced. Parallel coordinates plot are the most intuitive tool to visualize M-dimensional data. Several axis are displayed horizontally, representing each objective. A solution is composed of a value (point) in each axis, which are connected by lines. Figure 3.6 is an example of parallel coordinates plot.

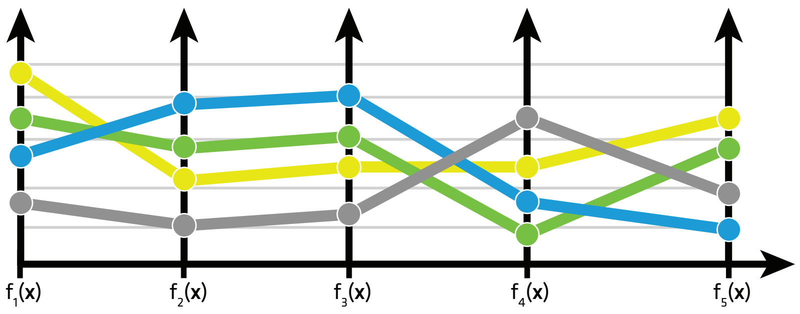


Figure 3.6: Parallel coordinates plot.

Quantitative methods of relationship identification use the concept of *how many* crossings are there in a parallel plot between objectives. The reasoning is that objectives with few crossings between each other are harmonious and only conflictual objectives have lots of crossings [29]. To this end, [32] developed *MOSS* which basically eliminates objectives with the same order of solutions. We'll use a similar method, but allowing some shuffling between solutions.

Examining Figure 3.6 once again, it is obvious that objectives 2 and 3 have the same ordering of solutions (blue, green, yellow and grey), and therefore must not be conflictual. Further examination shows that objectives 1 and 5 are similar but the ordering of blue and gray solution is swapped. While *MOSS* eliminates only solution 2 or solution 3, it seems justifiable to eliminate also either solution 1 or 5. To this end, a methodology was developed that uses rank correlation to evaluate the relative ordering of each pair of objectives, i.e., how dissimilar is the relative position of each objective solutions.

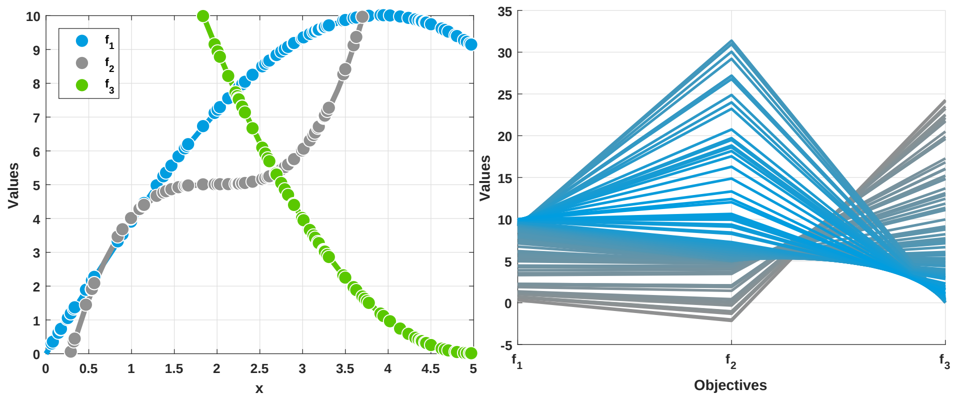
Rank correlation coefficients include Spearman's ρ and Kendall's τ. Of both, Kendall's τ [33] was chosen as suggested in [29]. It assesses the degree of similarity between two solutions' ordering and is defined in Equation 3.3, where n is the number of samples, n_c is the number of concordant samples (as one variable increases/decreases, the other imitates), and n_d the number of discordant ones. Succinctly, Kendall's tau is high when samples have a similar rank, i.e., same ordering of 1^{st}, 2^{nd},.... Some characteristics of Kendall's tau include $\tau(A, B) = \tau(B, A)$ and $\tau \in [-1, 1]$.

$$\tau = \frac{n_c - n_d}{n(n-1)/2} \tag{3.3}$$

Figure 3.7 explains the idea, with three functions to minimize, described in Equation 3.4. A single design variable exists: x, constrained between $[0, 5]$. As x increases, f_3 decreases, and both f_1 and f_2 increase, the latter being characterized by a inflection point in $x = 2$. While f_1 and f_2 "pull" x to the left, f_3 forces it to go to the right. Therefore f_1 and f_2 are harmonious, since they behave similarly. However, with $x > 4$ they become conflictual, making them not perfectly correlated.

$$F(x) = \begin{cases} f_1 = 10sin(\frac{2x}{5}) \\ f_2 = (x-2)^3 + 5 \\ f_3 = (x-5)^2 \end{cases} \tag{3.4}$$

One hundred randomly generated points were obtained in each function and the coordinate plot is presented in Figure 3.7b. It's visible that the two harmonious objectives, f_1 and f_2 have few crossings, mainly in higher values, while f_2 and f_3 have a lot. Indeed $\tau(f_1, f_2) \approx 0.78$, $\tau(f_1, f_3) \approx -0.78$ and $\tau(f_2, f_3) = -1$.

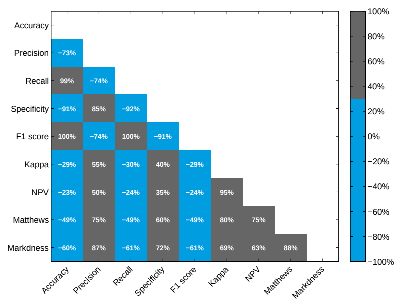


(a) Functions plot **(b)** Parallel Coordinates plot

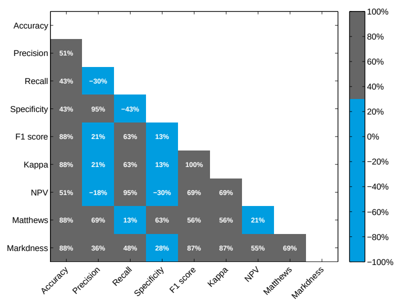
Figure 3.7: Tri-objective problem and parallel coordinates plot.

Kendall's *tau* is a continuous value $\in [-1, 1]$, and therefore to decide whether 2 objectives are conflictual or not, a threshold value K_c must be set. Basically, every pair of objectives is considered conflictual if their Kendall correlation is $< K_c$. Otherwise, harmony is assumed. Independence, as mentioned earlier, does not happen, confirmed by close to zero p-values, calculated using no relationship as the null hypothesis. The threshold value K_c was obtained by getting the maximum Kendall correlation from the Deb, Thiele, Laumanns and Zitzler (DTLZ) benchmarks problem, treated later in section 4.3. A value of $K_c = 0.3$ is found to be acceptable, calculated in a qualitative analysis.

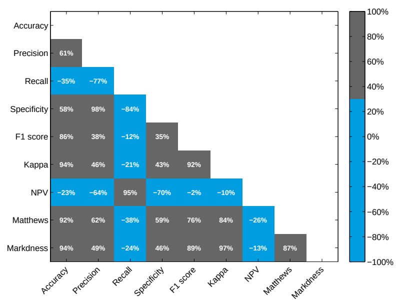
Random populations of size 30000 were obtained for each dataset. Only the best solutions, called non-dominated solutions (see next chapter), were selected, in order to assure only the best part of the search space is considered. Thereupon Kendall's correlation was calculated in a pairwise manner. The results for binary classification for each dataset are presented in heatmap form Figure 3.8, with blue squares meaning that objectives are conflictual.



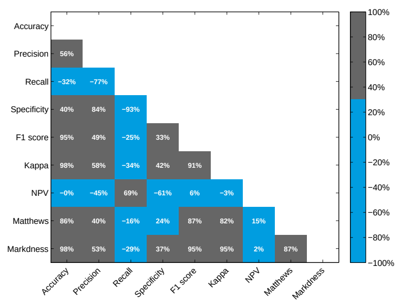
(a) AldoA



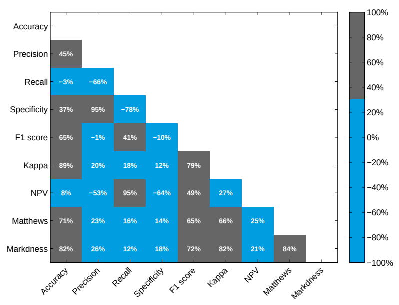
(b) Mushroom



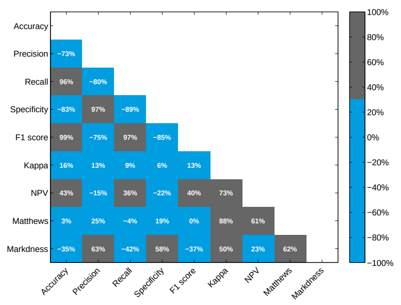
(c) Musk



(d) Phishing



(e) Sonar



(f) Spectf

Figure 3.8: Kendall correlation to binary datasets.

It's clear that all datasets have more than four redundant objectives, but it's interesting to verify that there is no repeated pattern. For example, in the first dataset accuracy is conflictual to most other objectives while in the second one it's harmonious with all of them.

The averaged results of all these datasets are graphed in Figure 3.9. There are several interesting remarks in this results. Firstly, the conflictual nature of precision and specificity, with an averaged $\tau = 92\%$, is not surprising, considering the first is proportional to TP and the second to TN. Another variable proportional to TP is recall, that is highly conflictual with specificity but also with precision. Additionally, F1-score, which is the harmonic mean of precision and recall, is conflictual with the first and harmonious with the second.

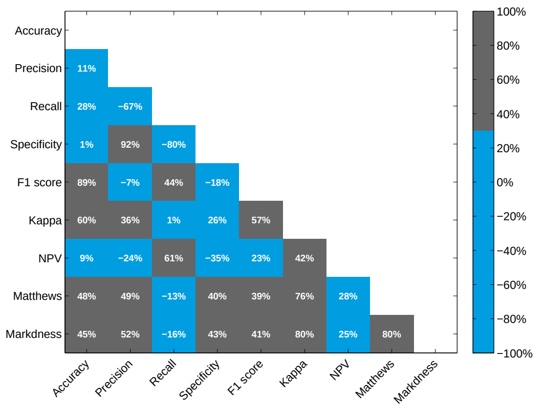


Figure 3.9: Kendall correlation averaged.

Selecting the most conflictual metrics could be, in itself, an optimization problem. Nonetheless, Figure 3.8 and Figure 3.9 can be analysed in terms of which ones are repeatedly redundant, leaving

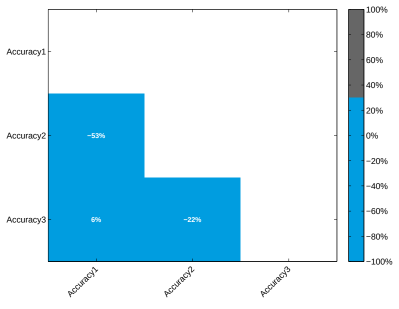
Table 3.3: Sets of metrics used in binary classification.

Nr. of Metrics	Perfomance Metrics
2	Accuracy
3	Recall, Specificity
6	Accuracy , Recall, Specificity, Kappa, NPV
10	All

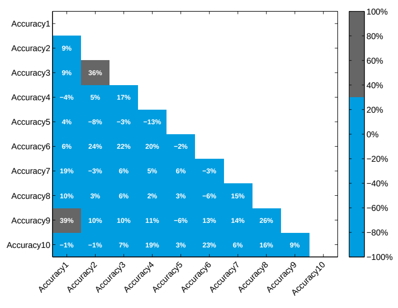
only conflictual ones. In all 6 datasets Matthews and Markedness are highly correlated, therefore, for example, Markedness can be dismissed. Matthews itself is consistently correlated with Cohen's kappa and precision with specificity. Additionally, F1-score is made redundant by accuracy. As a consequence, only 5 metrics remain: accuracy, precision, recall, Cohen's kappa and NPV.

Several sets of metrics should be selected to analyse later in this work, in combination with the cardinality metric described in Equation 3.1. For a 2 objective problem, accuracy is used, although not being quite satisfactory for unbalanced datasets. For 3 objectives problem, it's visible that that specificity and recall are the most conflictual ones in Figure 3.9. It makes sense, considering recall is proportional to TP and specificity to TN. Additionally, a set using the five non-redundant metrics mentioned above and a set with all metrics is selected, totalling four different sets, listed in Table 3.3.

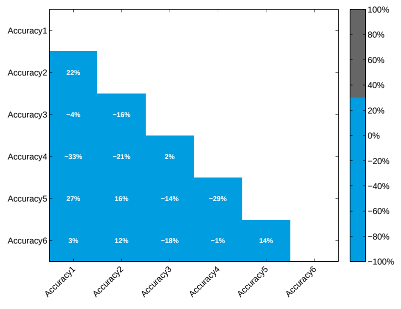
The same Kendall analysis was made to multi-class datasets using accuracy in each class, see Figure 3.10, and the results are quite different from binary classification. It's evident blue's dominance over the heatmap, indicating a lot more conflictual objectives. Using recall to each class yielded similar results.



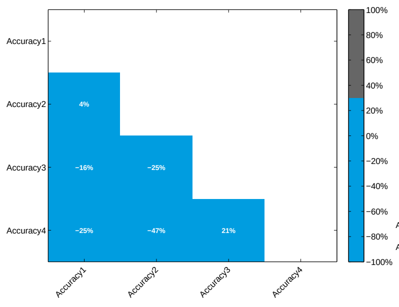
(a) DNA



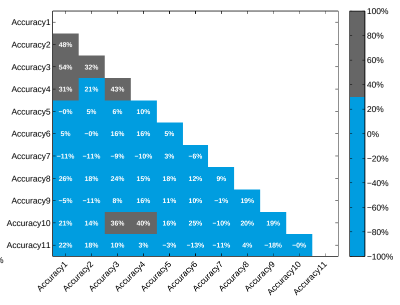
(b) Pendigits



(c) Satimage



(d) Vehicle



(e) Vowel

Figure 3.10: Kendall correlation to multi-class datasets using accuracy in each class.

4 Search Procedure

Contents

4.1 State of the Art 26
4.2 Algorithms 29
4.3 Multi-Objective Optimization Benchmark Problems 36
4.4 Performance Metrics for Multi-Objective Algorithms 39
4.5 Search Algorithms Validation 42
4.6 Encoding for feature selection 44

This chapter examines the second big characteristic of any feature selection algorithm: the search procedure. Randomized algorithms or meta-heuristics were chosen in the first chapter and later it was concluded that not only is there an advantage in using a cardinality performance metric, prioritizing solutions with less features, but also that it might be advantageous to use more performance metrics to explore the search space. This requires the search procedure to use multiple objectives simultaneously.

A brief introduction to multi-objective optimization research area is presented, explaining the fundamental concepts. Then several state of the art algorithms are explained in detail, which will be used throughout this work. Furthermore, these are validated using benchmark problems and performance indices designed specially for multi-objective optimization.

Finally, search procedure parameters such as genetic operators and other parameters are discussed and selected to deal with feature selection.

4.1 State of the Art

4.1.1 Concepts

The reader is referred to [34] for a brief overview of the most recent developments in multi-objective optimization, and to [35] and [36] for a more thorough analysis of this field. Here it will merely be succinctly discussed.

Classical optimization aims at finding the set of decision variables that optimize (either minimize or maximize) a particular goal while being subjected to various constraints. There's a combination of parameters that reach a global optimum, or several, in the case of multi-modal problems. As a natural consequence of research in this field, the problem of scalability was addressed, adding new goals to the problem.

Multi-objective optimization with M objectives can be formally described as in Equation 4.1, for a minimization problem.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & F(x) = \{f_1(x), f_2(x), ..., f_m(x)\} \\ \text{subject to} \quad & g_n(x) \leq 0, n = 1, 2, ..., J, \\ & h_k(x) = 0, k = 1, 2, ..., K \end{aligned} \tag{4.1}$$

The mathematical formulation is not very different from one objective optimization, but the problem's nature is substantially altered. Firstly, two spaces arise: the design space where the variables $[x_1, x_2, ..., x_N]$ are changed, and the objective space which consists of the results $[f_1, f_2, ..., f_M]$. Figure 4.1 represents both these spaces and their mapping.

Additionally, the concept of global optimum disappears, and that is easily understood in Figure 4.2 illustrating five different solutions in a bi-objective problem. There is no global optimum because no solution is minimal in both objectives, so the search for the best solution is infeasible, and is substituted by the search for the worst ones. In Figure 4.2 it's evident that solution 2 is worse in both objectives than solutions 1 and 3. The same happens with solution 4, which is worse than solution 3. Solutions 2 and 4 are said to be dominated by solutions 1 and 3, respectively, and solutions 1,3 and 5 are called non-dominated solutions or Pareto optimums, named after Vilfredo Pareto that first used this

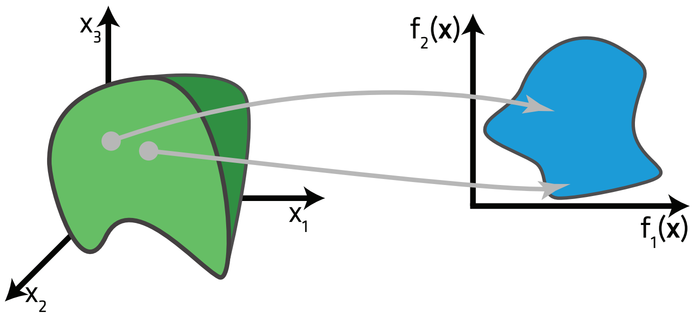


Figure 4.1: Design and objective space and their mapping.

concept in economic efficiency [35]. Together the three of them constitute the non-dominated front. The non-dominated front, ideally, is coincidental to the Pareto front, the multi-objective equivalent to single-objective global optimum.

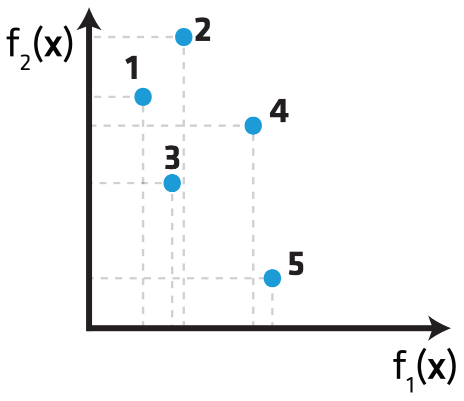


Figure 4.2: Solutions in a minimization bi-objective problem.

A solution A strictly dominates solution B, when there's at least one objective in which A is better, while being no worse in all others[35]. Mathematically $A \preceq B$, if these two conditions are met:

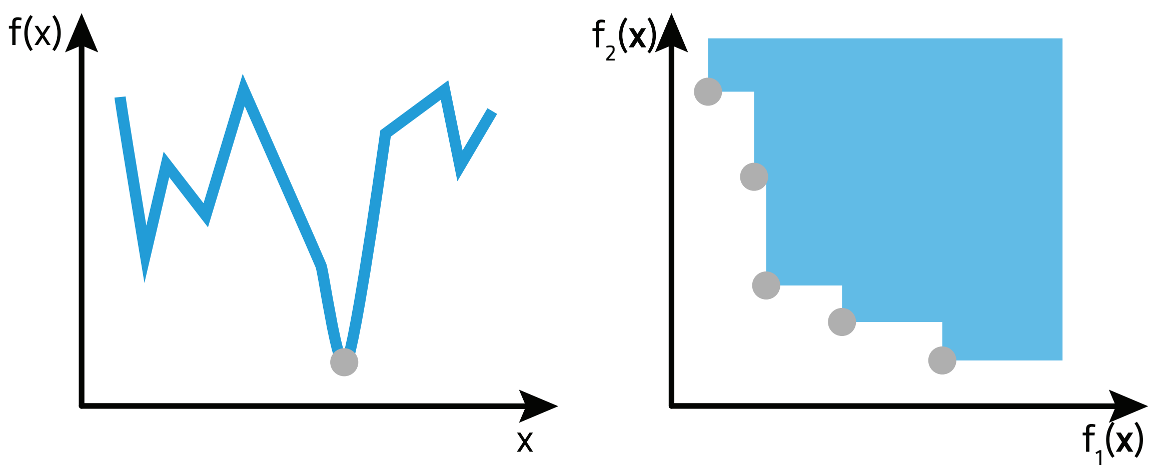
1. $\forall i \in \{1, 2, ..., k\} : f_i(A) \leqslant f_i(B)$
2. $\exists j \in \{1, 2, ..., k\} : f_j(A) < f_j(B)$

This is defined as weak Pareto dominance, while strong Pareto dominance ($A \prec B$) requires all objectives to be better.

Figure 4.3 represents the difference of optimum concept for single-objective and multi-objective optimization, both on minimization problems. In the first, there's an optimal point to search for, while in the second there's a set of optimal points. In continuous problems, they might be infinite.

4.1.2 Methodologies

In section 1.1 randomized methods, or meta-heuristics, were chosen for search procedure since they're the most suitable for NP-hard problems. Additionally, in multi-objective optimization, two more advantages arise: meta-heuristics produce several solutions in one run, and therefore should be able to represent the whole Pareto front. Additionally, meta-heuristics are compatible with all distinct



(a) Single-Objective Optimization **(b)** Multi-Objective Optimization

Figure 4.3: Optimum concept difference.

methods to add objectives. These methods can be classified in numerous distinct ways, but in this work, Cohon and Marks [37] is used. Their classification is based on the timing of the optimization process when the decision maker inevitably introduces information. Their division consists in: *a priori*, interactive and *a posteriori* methods, which will be briefly discussed.

A priori

A priori methods use information provided by the DM before the search begins. Lexicographic ordering, where the objectives are sorted in respect to importance and then each is sequentially optimized, is an example. Another notorious and intuitive example is the weighting method, aggregating all objectives in one only and optimizing that objective. Some of these methods despite being simpler and faster, have the inherent difficulty of reaching non-convex parts of the true Pareto front. *A priori* methods are also criticized for requiring information that most times the decision maker doesn't have, and for needing to be run if the DM changes or simply change its preferences.

Interactive

This type of methods combines the search process and information input in a simultaneous manner. While being very promising, they require both the DM knowledge of the problem and time to participate in the searching procedure. The Normal Boundary Intersection [38] method is among the most popular.

A posteriori

In *a posteriori* methods the decision maker simply chooses one from a set of already available solutions. The goal is, therefore, to get a set of widespread solutions located near the true Pareto front, offering diversity to the DM.

A posteriori methods are, for that reason, preferred. The first step to this latter kind of algorithms was made in 1985 by Shaffer when creating VEGA [39], the first rough genetic algorithm adapted to consider several objectives. Nowadays meta-heuristic multi-objective optimization algorithms differ from classical single-objective ones essentially in the selection method, i.e., the process in which the best solutions are ranked and selected. There are three big groups of selection procedures [40], as listed in Figure 4.4.

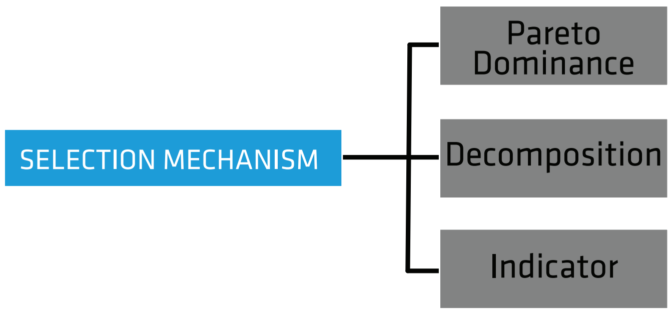


Figure 4.4: Selection mechanisms in MOP.

The first kind is the most common. It uses the concept of Pareto dominance from above to differentiate solutions. Decomposition methods work by dividing the problem into numerous small problems such as weighting function or goal programming. Indicator-based is the least common kind of methods. They use some metric such as the hypervolume or the ϵ-indicator to guide solutions to the Pareto front and are usually computationally slow.

Throughout the years, investigation [41] showed that when introducing more objectives, classic multi-objective optimization methods had their performance decremented. This is true mostly to methods that rely on the concept of Pareto dominance, since the population easily becomes completely non-dominated, which weakens its ability to converge to the Pareto front. Figure 4.5 represents this phenomenon, displaying the box plot of non-dominated population, using different population sizes and number of objectives. Each population consists of uniformly generated random vectors in the interval $[0, 1]^M$, to which non-domination sorting was performed, i.e., which were organized by non-dominated levels $(F_1, F_2, ..., F_N)$. It's visible that increasing population size is an effective way to delay total population non-dominance.

Different counter-measures to this problem are suggested [41, 42] such as the modification of Pareto dominance or density of solutions estimation, but the main alternative nowadays is other selection methods, such as decomposition-based and indicator-based algorithms.

Other issues with objective cardinality increment include computational complexity and difficulty in the solutions' visualization [42]. Therefore optimization problems are customarily divided into single, multi and many-objective problems, and the frontier between multi and many-objective is usually considered to be 4 or 5.

4.2 Algorithms

As discussed in section 3.1, the problems of binary classification and multi-class classification both can use more than 5 objectives, thus entering the category of many-objective problem (even though

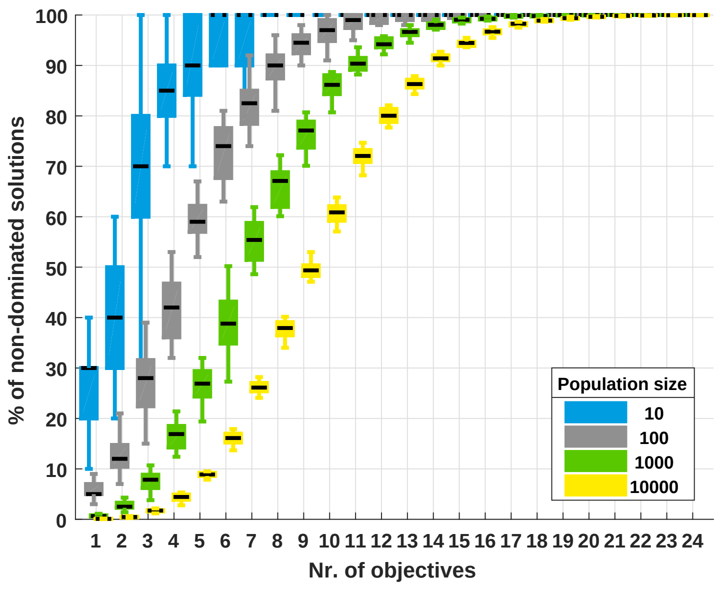


Figure 4.5: Ratio of non dominated solutions depending on the number of objectives and population size, using random generated vectors, in 20 tries.

some objective might be redundant, as shown in section 3.2).

The algorithms to be tested in this work are based on the current state of the multi and many-objective research community, considering some variation in the number of conflicting objectives. The reader is referred to [40, 42, 43] for a more thorough compendium of comparisons and a more detailed list of algorithms. While surveying the field, one should bear in mind the No-Free-Lunch theorem that states “any two optimization algorithms are equivalent when their performance is averaged across all possible problems”[44]. This statement is supported by the work in [45] that compares several multi-objective algorithms in a broad range of problems and isn’t able to find any prominent one.

NSGA-II [46], or the second version of the non-dominated sorting algorithm, is indubitably the best known and most used algorithm for multi-objective optimization. It’s a Pareto based method, and therefore has scalability issues, as discussed in subsection 4.1.2.

In response to NSGA-II scalability critics, the same authors created the third generation of the non-dominated sorting algorithm, NSGA-III [47]. It’s a decomposition based algorithm, which is fairly recent but has shown great results in many-objective problems. Numerous structural changes were made in relation to its predecessor, including the selection mechanism being replaced by a decomposition-based one.

Nevertheless, it’s worth mentioning that not always does NSGA-III outperform NSGA-II, particularly in knapsack problems [48], which are discrete and therefore of great relevance to feature selection.

While introducing NSGA-III, the comparison was made with NSGA-II and MOEA/D [49]. The last one is also a decomposition-based algorithm that has since become the standard for many-objective optimization. Li [50] showed that frequently MOEA/D outperforms NSGA-II.

Additionally, the work in [51] compared several algorithms (NSGA-III was still unborn) and found that PICEA-g and HypE [52] generally performed better, the first being also decomposition based, in which there’s simultaneous evolution of solutions and a goals’ population, and the second is indicator based.

These are the algorithms to be tested in this work: NSGA-II, NSGA-III, MOEA/D, PICEA-g and HypE. All are population based, meaning a single run is enough to provide distinct solutions to the decision maker, and particularly are applications of genetic algorithms.

A brief introduction will be made to each algorithm, but for a more detailed description the reader is referred to the original papers. PlatEmo [53] was used for the algorithm's implementation in MatLab environment.

4.2.1 NSGA-II

The second **N**on-dominated **S**orting **G**enetic **A**lgorithm is undeniably the most well-known algorithm for multi-objective optimization, due to its performance and apparent simplicity. It was created in 2002 by Deb et. al [46], the selection mechanism is Pareto-rank based, while also preferring isolated solutions by means of a niching score assigned to each solution, which basically consists of the distance from the closest neighbour solutions, to which the authors call crowding distance.

The algorithm is described in Algorithm 1. In the $i-th$ generation the parent population P_t of size N is selected through a crowded tournament, which is similar to binary tournament selection but prioritizes lower ranked solutions, and in case of ties, selects the one with higher crowding distance. Genetic operators are used to produce an offspring Q_t. Both parent and offspring populations are mixed, creating a population $|R_t| = 2N$, ensuring elitism.

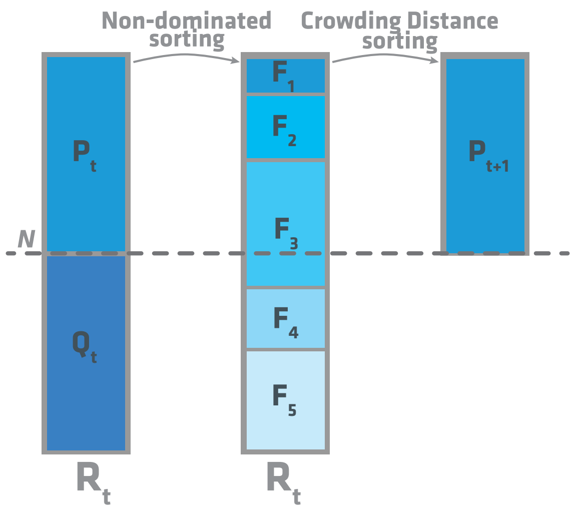


Figure 4.6: Proceedings of NSGA II.

Thereupon, R_t is non-dominated sorted into several fronts of non-dominance $F_1, F_2, ...$ This means that all solutions of F_1 dominate the ones in F_2 and so on. These fronts of solutions are incrementally added to the final population P_{t+1}, until there's a front F_i that cannot be fully inserted into P_{t+1} without exceeding N. Thereon, the solutions in that last front are sorted according to their crowding distance d_t, and only the ones with larger value are added, filling completely the available slots in P_{t+1}. The idea is to prioritize solutions that are in undiscovered areas of the search space.

Algorithm 1 NSGA-II algorithm for the t-th generation

```
    Input: Parent population $P_t$
    Output: Next population $P_{t+1}$
 1: $P_t$=Crowded Tournament($P_t$)
 2: $Q_t$=Crossover+Mutation($P_t$)
 3: $R_t = P_t \cup Q_t$
 4: $d_t$=Crowding Distance($R_t$)
 5: ($F_1$,$F_2$,...)=Non dominated sorting($R_t$)
 6: $P_{t+1} = \emptyset$ and $i = 1$
 7: repeat
 8:     $P_{t+1} = P_{t+1} \cup F_i$
 9:     $i = i + 1$
10: until $|P_{t+1}| + |F_i| \geq N$
11: if $|P_{t+1}| \neq N$ then
12:     $F_i = Sort(F_i, d_t)$
13:     $P_{t+1}$=$P_{t+1} \cup F_i[1 : N - |P_{t+1}|]$
14: end if
```

Besides the regular parameters such as stopping criteria, population size or genetic operators, the algorithm is parameter-less, another reason for its acclaim.

4.2.2 NSGA-III

Non-dominated Sorting Genetic Algorithm III has little similarities with its predecessor. It's relatively new and was conceived to handle many-objective problems. It uses a decomposition based selection mechanism giving priority to points near reference points.

The first step is to spread a set of reference points Z_r in the $M-$dimensional search space, in a hyperplane. The number of reference points should be similar to the population size, to ensure enough diversity while keeping complexity to a minimum, and the placing of these points follows Das and Denni's approach [38].

The general procedure for the $i - th$ generation is described in Algorithm 2. It does not perform tournament selection, instead, it randomly selects the parent population for crossover. R_t is created and the non-dominated sorted, resulting in non-dominated fronts $F_1, F_2, ...$ Similarly to NSGA-II, the final population is incrementally filled with these fronts, until the size N is exceeded. The procedure from thereupon is completely different from its predecessor.

Firstly, each objective is normalized between $[0, 1]$. The normalized population is then associated to each one of the uniformly distributed reference points Z_r and the solutions in F_i associated with the less represented reference points are selected, in a niching function. For a thorough analysis of each sub-function such as normalization, association or niching, the reader is referred to the original paper.

Algorithm 2 NSGA-III algorithm for t-th generation

```
    Inputs: Parent population $P_t$
            $Z$ points in a hyperplane
    Output: Next population $P_{t+1}$
 1: $Q_t$=Crossover+Mutation($P_t$)
 2: $R_t = P_t \cup Q_t$
 3: ($F_1$,$F_2$,...)=Non dominated sorting($R_t$)
 4: $P_{t+1} = \emptyset$ and $i = 1$
 5: repeat
 6:     $P_{t+1} = P_{t+1} \cup F_i$
 7:     $i = i + 1$
 8: until $|P_{t+1}| + |F_i| \geq N$
 9: if $|P_{t+1}| \neq N$ then
10:     ($P_{t+1}$,$F_i$)=Normalize($P_{t+1}$,$F_i$)
11:     ($P_{t+1}$,$F_i$)=Associate($P_{t+1}$,$F_i$,$Z_r$)
12:     $P_{t+1}$=Niching($P_{t+1}$,$F_i$,$Z_r$)
13: end if
```

Similarly to NSGA-II, this algorithm is parameter-less.

4.2.3 MOEA/D

In 2007 Zhang and Li created MOEA/D [49], an algorithm that is based on decomposition and that would later win the Unconstrained Multiobjective Evolutionary Algorithm Competition at the 2009 Congress on Evolutionary Computation.

It takes advantage of the fact that any Pareto optimal solution could be the solution of a scalar function, by decomposing the problem into several scalar sub-problems. Each sub-problem is represented by a weight vector w_i, such that $\sum_{i=1}^{m} w_i = 1$.

The sub-problems can be of any kind. Generally either the weighted sum (Equation 4.2) or Tchebycheff approach (Equation 4.3) are used. In this work, the latter was chosen, considering the first is unable to deal with nonconvex Pareto fronts [49].

$$g^{ws}(\mathbf{x}|\mathbf{w}) = \sum_{k=1}^{m} w_k f_k(x) \tag{4.2}$$

$$g^{tc}(\mathbf{x}|\mathbf{w}, \mathbf{z}) = \max_{1 \leq k \leq m} w_k(f_i(x) - z_i) \tag{4.3}$$

Algorithm 3 lists the steps for a MOEA/D algorithm. It relies on a population of weight vectors, each assigned to one solution. During initialization, aside from a randomly generated population of size N, the algorithm creates those N weight vectors W uniformly spread. With the user-defined number of neighbours T, during initialization, the algorithm finds the T closest neighbours to each weight vector, and stores them in variable $B_i = \{i_1, i_2, ..., i_T\}$. Additionally, the ideal vector z is also initialized.

Each generation consists of going through each weight vector W_i and randomly selecting two neighbours to create a new solution for that weight vector, using genetic operators. Only the first offspring is considered. If any objective of the newly created solution is lower than the previously recorded, the ideal point z is updated. Then the new solution is tested in all neighbours' weight vectors, and in case it has the same or better performance, the new solution is also assigned to that weight vector. Afterwards, the algorithm visits the next weight vector and repeats this process until every weight vector has been studied.

Algorithm 3 MOEA/D algorithm for t-th generation

```
Inputs: Parent population $P_t$
        Uniform spread of weight vectors $W$
        External Population $EP$
        Ideal point $z$
        Neighbours information $B^i$
Output: Next population $P_{t+1}$
1: for $i = 1 : N$ do
2:     $R_i$=Select two random neighbours from $B_t^i$
3:     $P_t^i$=Crossover+Mutation($R_i$)
4:     Update $z$
5:     for $j \in B_t^i$ do
6:         if $g(y|) \leq g(j)$ then
7:             $P_t^j$=$P_t^i$
8:         end if
9:     end for
10:    Update $EP$
11: end for
```

Aside from genetic operators parameters, the number of neighbouring sub-problems must also be defined. The idea of only considering information from a sub-problem's neighbour is that an optimal for a determined sub-problem should be rather close to a similar sub-problem. This is true mainly for continuous problems, but discrete applications of MOEA/D have shown good results [54].

4.2.4 PICEA-g

Preference-based algorithms usually have good performance but limit the algorithm and require *a priori* knowledge. The idea of co-evolving simultaneously both a family of solutions and a set of preference vectors was first discussed in [55], showing promising results against NSGA-II. PICEA-g [51] is a realization of this concept.

The general algorithm is described in Algorithm 4. At generation t, the population P_t spawns an offspring population Q_t using the usual genetic operators. Simultaneously a new goals population QG_t of size N_{Goals} is randomly generated. Parents and offspring of each population are pooled, forming $R_t = P_t \cup Q_t$ and $S_t = G_t \cup QG_t$. Using a concept of goal domination, fitness values are assigned to every solution (F_i^s) and goal (F_j^g), apprising solutions that dominate more goals and goals that are less dominated.

To prioritize non-dominated solutions, which are always preferred, R_t is non-dominated sorted to find those. If the number of non-dominated solutions in R_t is smaller than N, these are assigned maximum fitness ($max(F_i^s)$, and the best N solutions of R_t according to F_i^s are then inserted to the final population P_{t+1}. If, however, the number of non-dominated solutions exceeds N, then these non-dominated solutions are sorted according to F_i^s and the first N become P_{t+1}. The goals population G_{t+1} is obtained simply by sorting S_t according to F_j^g and then truncated.

Algorithm 4 PICEA-g algorithm for t-th generation

```
Inputs: Parent population $P_t$
        Goal Population $G_t$
        Number of Goals $N_{Goals}$
Output: Next population $P_{t+1}$
        Next Goal Population $G_{t+1}$

 1: $Q_t$=Genetic Operators($P_t$)
 2: $QG_t$=Random Generator($N_{Goals}$)
 3: $R_t = P_t \cup Q_t$
 4: $S_t = G_t \cup QG_t$
 5: $F_i^s$ and $F_j^g$ Calculation
 6: $ND$=Non-Dominated($R_t$)
 7: $R_t = R_t \cap ND$
 8: if $|ND| < N$ then
 9:     $F^s(ND) = max(F_i^s)$
10:     $P_{t+1} = sort_{F_i^s}(ND \cup R_t, N)$
11: else
12:     $P_{t+1} = sort_{F_i^s}(ND, N)$
13: end if
14: $G_{t+1} = sort_{F_g^s}(S_t, N_{Goals})$
```

The only parameter to choose in this algorithm is the number of goals.

4.2.5 HypE

Indicator-based algorithms are intuitively the most well suited methods to apply, since they should be the ones who can easier distinguish better solutions. The main drawback of these methods are their computational cost. A notorious example is IBEA [56]. In 2011, Bader and Zitzler [52] created the **Hyp**ervolume **E**stimation Algorithm for Multiobjective Optimization, which they called HypE. The novelty of this algorithm is a Monte Carlo simulation to approximate the hypervolume indicator, vastly reducing the computational cost. Succinctly, for a number of objectives > 3 the algorithm estimates the hypervolume, while for less than three objectives it calculates the exact value.

During initialization, aside from a randomly generated initial population, the number of samples must be chosen, and a set of reference points is created. From then on, the main loop starts. Algorithm 5 lists the steps involved in the $t-th$ generation.

It starts by selecting a parent population P'_t from the initial one P_t, using a binary tournament selection based on the hypervolume value. Genetic operators modify P'_t to create an offspring population Q_t. Then both parent and offspring population are pooled, forming $R_t = P_t \cup Q_t$.

Similarly to what is done in both NSGA-II and NSGA-III, a non-dominated sorting is performed to form front $F_1, F_2, ...$, and one by one they're inserted into P_{t+1}. When the insertion exceeds the population's limit, iteratively solutions that least decrease the hypervolume metric are removed until P_{t+1} has the correct size.

Algorithm 5 HypE algorithm for t-th generation

```
Inputs: Parent population $P_t$
        Goal Population $G_t$
        Number of Goals $N_{Goals}$
Output: Next population $P_{t+1}$
        Next Goal Population $G_{t+1}$

 1: $P_t'$ =Binary Tournament($P_t$)
 2: $Q_t$=Genetic Operators($P_t$)
 3: repeat
 4:     $P_{t+1} = P_{t+1} \cup F_i$
 5:     $i = i + 1$
 6: until $|P_{t+1}| + |F_i| \geq N$
 7: if $|P_{t+1}| \neq N$ then
 8:     $k = N - |P_{t+1}|$
 9:     while $k > 0$ do
10:         $W$=Get Worst Solutions
11:         $F_i = F_i \cap W$
12:         $k = k - 1$
13:     end while
14:     $P_{t+1} = P_{t+1} n F_i$
15: end if
```

The parameters needed are a reference set for the hypervolume calculation (can be automatically generated), and the number of sampling points. Increasing the number of sampling points results in a better estimation of the hypervolume.

4.3 Multi-Objective Optimization Benchmark Problems

In order to validate and make a first comparison between algorithms, tests should be made. Since there are no benchmark problems for multi-objective feature selection, due to its novelty and complexity, classic benchmark problems are used.

It took little time after Deb, Zitzler, and Thiele effort to create a benchmark problem to easily compare algorithms in [57], which originated the famous ZDT problems, to realize the need of scalable test problems, in which the user could easily increase the number of objectives.

In [58], DTLZ were created, consisting of 9 problems, each having their own obstacles, with no external parameters and easily scalable. Furthermore, all problems have known Pareto fronts.

A few years later a new set of problems was created, denominated WFG [59], which expanded DTLZ problems, including a number of properties such as deceptive problems, mixed shape Pareto Fronts, etc. They consist of a series of transformations, each adding a desired characteristic to the problem, such as non-linearity, discrete Pareto front, convexity, etc...

DTZ, DTLZ and WFG, all synthetic problems, are the most widely known sets of problems. WFG is known for being harder, but when the number of objectives increases some of them become extremely related, and therefore do not truly represent the number of objectives used. Therefore DTLZ were chosen to truly represent the number of objectives in this work.

Of the 9 DTLZ problems only a couple are chosen. To challenge both diversity and convergence

DTLZ4 and DTLZ7 were selected, respectively. Both problems have the basic structure depicted in Equation 4.4. Additionally, the Multi-Objective Knapsack Problem (MOKP) was also implemented, for being discrete and therefore related to feature selection.

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & F(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_M(\mathbf{x})\} \\ \text{subject to} \quad & 0 \leq x_i \leq 1, \text{ for } i = 1, 2, ..., D \end{aligned} \tag{4.4}$$

DTLZ4

The first problem, DTLZ4, measures the algorithm capacity to maintain a good distribution of solutions, and has a strong bias. Figure 4.7a represents a randomly population of size 10000 for M=2, along with its known pareto front in black.

Equation 4.4 is modified according to Equation 4.5, where $\alpha = 100$ and $g(x_M) = \sum_{x_i \in x_M}(x_i - 0.5)^2$.

$$\begin{aligned} & f_1(\mathbf{x}) = (1 + g(x_M))cos(x_1^\alpha \pi/2)cos(x_2^\alpha \pi/2)...cos(x_{M-2}^\alpha \pi/2)cos(x_{M-1}^\alpha \pi/2), \\ & f_2(\mathbf{x}) = (1 + g(x_M))cos(x_1^\alpha \pi/2)cos(x_2^\alpha \pi/2)...cos(x_{M-2}^\alpha \pi/2)sin(x_{M-1}^\alpha \pi/2), \\ & f_3(\mathbf{x}) = (1 + g(x_M))cos(x_1^\alpha \pi/2)cos(x_2^\alpha \pi/2)...sin(x_{M-2}^\alpha \pi/2), \\ & \vdots \\ & f_{M-1}(\mathbf{x}) = (1 + g(x_M))cos(x_1^\alpha \pi/2)sin(x_2^\alpha \pi/2), \\ & f_M(\mathbf{x}) = (1 + g(x_M))sin(x_1^\alpha \pi/2), \end{aligned} \tag{4.5}$$

DTLZ7

The second DTLZ problem, DTLZ7, tests the algorithms convergence, being harder than DTLZ4. Additionally, the Pareto front, also known, has $M - 1$ disconnected regions. This is possibly related to feature selection since it's likely disconnected. Figure 4.7b presents the Pareto front and 10000 randomly generated solutions.

Once again, Equation 4.4 is modified according to Equation 4.6.

$$\begin{aligned} & f_1(x_1) = x_1, \\ & f_2(x_2) = x_2, \\ & \vdots \\ & f_{M-1}(x_{M-1}) = x_{M-1} \\ & f_M(\mathbf{x}) = (1 + g(x_M))h(f_1, f_2, ..., f_{M-1}, g), \end{aligned} \tag{4.6}$$

Where

$$g(\mathbf{x_M}) = 1 + \frac{9}{|x_M|} \sum_{x_i \in \mathbf{x_M}} x_i$$

$$h(f_1, f_2, ..., f_{M-1}, g) = M - \sum_{i=1}^{M-1} \left[\frac{f_i}{1+g}(1 + sin(3\pi f_i)) \right]$$

MOKP

Furthermore, since feature selection is a discrete problem, multi objective knapsack problem (MOKP) was also chosen. The single-objective version consists of a set of items, weight and profit associated with each item, and one knapsack with limited capacity. The goal is to find a subset of items to get in the knapsack, while maximizing profit and not exceeding the weight limit [60].

This can be extended to a Multi-Objective Problems (MOP) with increasing number of knapsacks. Formally, using D items and M knapsacks, with

$$\begin{aligned} p_{i,j} &= \text{profit of item } j \text{ according to knapsack } i \\ w_{i,j} &= \text{weight of item } j \text{ according to knapsack } i \\ c_i &= \text{capacity of knapsack } i \end{aligned}$$

Each knapsack becomes dependent of a variable $\mathbf{x} = (x_1, x_2, ...x_m)$, that represents the presence of the item in the knapsack $x_i \in \{0, 1\}$, such that the mathematical formulation of MOKP becomes Equation 4.7.

$$\begin{aligned} & \underset{x}{\text{maximize}} & & F(x) = \sum_{j=1}^{M} p_{i,j} x_j \\ & \text{subject to} & & \sum_{j=1}^{M} w_{i,j} x_j \leq c_i, i = 1, 2, ..., D \end{aligned} \tag{4.7}$$

Although originally it's a maximization problem, it was modified to be a minimization one. Additionally, the randomly generated $p_{i,j}$, $w_{i,j}$ and c_i values were fixed during this work, to allow fair comparison. Refer to [60–62] for more information. Much like the original knapsack problem doesn't have an *a priori* optimal solution, MOKP has no known Pareto front. Figure 4.7c presents 10000 randomly generated solutions for the fixed values of $p_{i,j}$, $w_{i,j}$ and c_i.

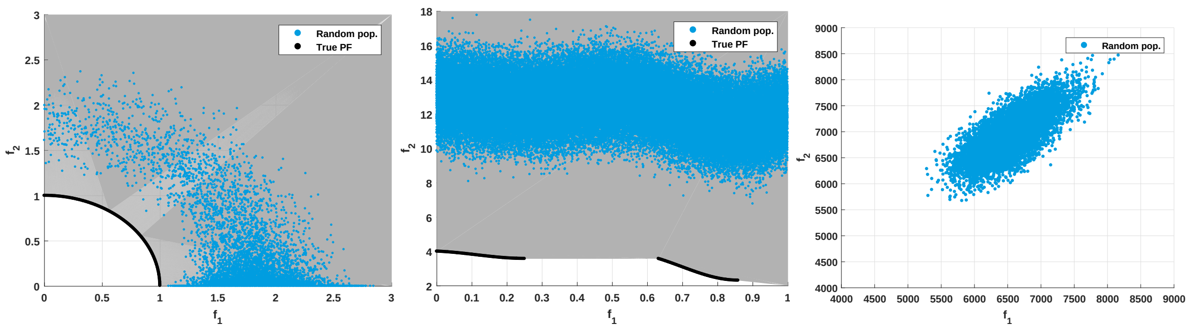
Table 4.1 lists the chosen MOP and their main characteristics.

Table 4.1: MOP for validation and their characteristics

Problem	**Characteristic**
DTLZ4	Diversity
DTLZ7	Convergence & Disconnected
MOKP	Discrete & Binary-Coded

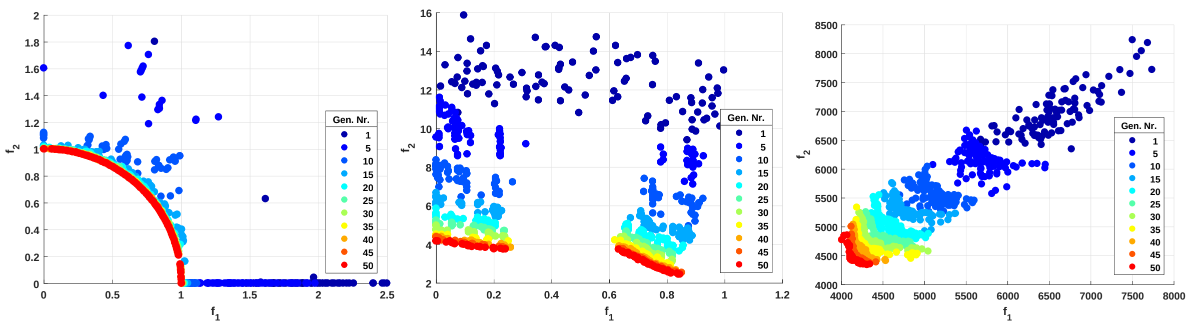
Figure 4.8 illustrates different iterations of NSGA-II applied to each problem, using $M = 2$, where the evolution is visible and the power of meta-heuristics is easily observed. The algorithm is able to reach the Pareto Fronts of both DTLZ4 and DTLZ7 with less than 50 generations.

A qualitative evaluation between final populations for each algorithm is almost impossible. Figure 4.9 shows the resulting populations of each algorithm after 500 generations, using M=2. The results are very similar, but the high number of generations allowed all algorithms to converge. The results for $M = 10$ are even harder to visualize, as can be seen in Figure 4.10.



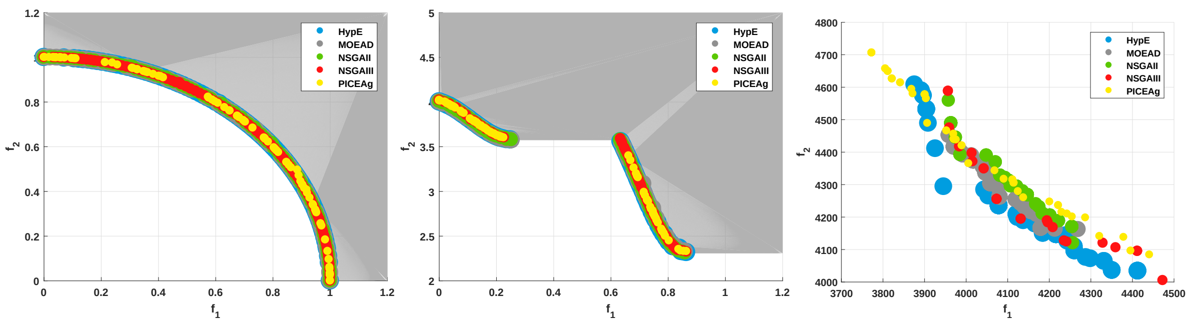
(a) DTLZ4 (b) DTLZ7 (c) MOKP

Figure 4.7: Random population in all three problems: DTLZ4, DTLZ7 and MOKP.



(a) DTLZ4 (b) DTLZ7 (c) MOKP

Figure 4.8: NSGA-II 100 individuals population evolution across 50 generations on each subproblem using M=2.



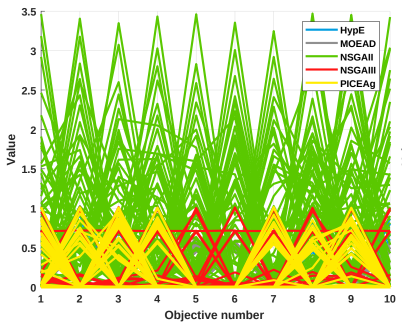
(a) DTLZ4 (b) DTLZ7 (c) MOKP

Figure 4.9: Results for each of the problems with two objectives, for 10000 generations and with population of 100.

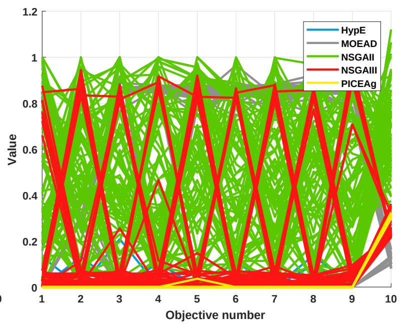
In addition to being hard to compare visually, since these methods are stochastic, these results lack repetition. Therefore an average and variance approach must be made, using suitable performance metrics, instead of eye inspection. These performance metrics are now discussed.

4.4 Performance Metrics for Multi-Objective Algorithms

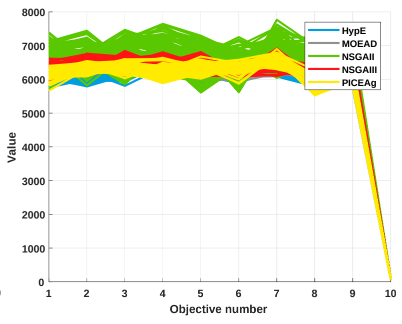
Consider the bi-objective problem with a Pareto front such as the blue lines in Figure 4.11, and the three solutions represented. The first one, 4.11a, is clearly too far away from the Pareto front, and therefore did not converge well. On the contrary, 4.11b actually converges well but does not truthfully



(a) DTLZ4



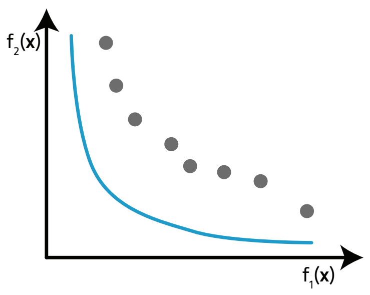
(b) DTLZ7



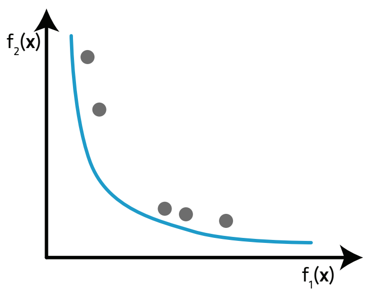
(c) MOKP

Figure 4.10: Results for each of the problems with 10 objectives, for 10000 generations and with population of 100. DTLZ7 last objective was subjected to normalization.

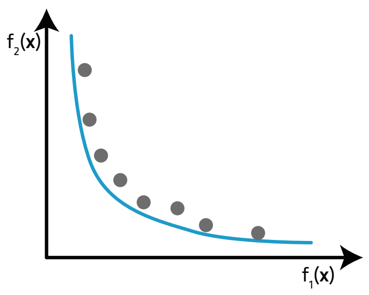
represent the Pareto front because the solutions are too spread out. Finally, 4.11c in addition to being very close to the Pareto front also is homogeneously distributed to represent all front.



(a) Bad convergence.



(b) Bad diversity.



(c) Converged and diverse set of solution.

Figure 4.11: Three different examples of solutions for a bi-objective problem. Pareto front (blue line) and algorithm's solutions (gray circles) relationship.

Consequently, a multi/many-objective optimization algorithm should be evaluated with respect not only to convergence but also to diversity. Table 4.2 is a compendium of a small number of famous indicators, divided by what they measure: there are some that measure convergence and diversity independently, and others that combine them.

Table 4.2: MOP algorithms performance indicators.

Indicator	**Measure**
Error Ration Set Coverage Generational Distance	Convergence
Spacing Spread R2	Diversity
Hypervolume Attainment Distance G-metric	Convergence and Diversity

One performance metric of each kind was chosen for this work.

4.4.1 Convergence

Convergence is measured in a comparative way, using **set coverage**, as presented in [63]. $C(A, B)$ compares two algorithms' non-dominated front A and B calculating the proportion of solutions of B that are weakly dominated by A, as in Equation 4.8. This index is not symmetrical, therefore both $C(A, B)$ and $C(B, A)$ should be calculated.

$$C(A, B) = \frac{|\{b \in B; \exists a \in A : a \succeq b\}|}{|B|} \tag{4.8}$$

In the example of Figure 4.12 solution A dominates only one solution of B, while B dominates 7 of A's. Thus $C(A, B) = \frac{1}{5} = 0.20$ and $C(B, A) = \frac{7}{9} \approx 0.78$, meaning that solution A is closer to the Pareto front, and therefore is better.

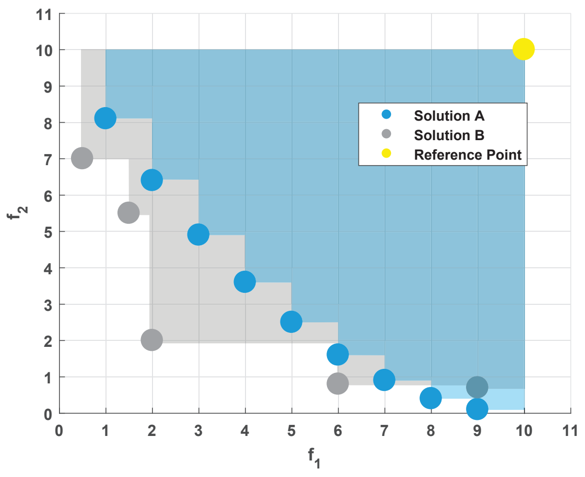


Figure 4.12: Two example solutions for a bi-objective problem.

4.4.2 Diversity

Spacing, S, was used to test for diversity among solutions of a M-dimensional objective space. Introduced by Schott in his master thesis [64] it is a measure of the distance vectors d_i standard deviation. When the solutions are uniformly spaced, this metric is small, since the distance vectors are similar. Therefore, unlike set coverage, the intention is to minimize it.

$$S(A) = \sqrt{\frac{1}{|A|} \sum_{i=1}^{|A|} \left(d_i - \overline{d}\right)^2} \tag{4.9}$$

$$d_i(A) = min_{k \in A \wedge k \neq i} \sum_{m=1}^{M} \left|f_m^i - f_m^k\right| \tag{4.10}$$

$$\overline{d(A)} = \sum_{i=1}^{|A|} \frac{d_i}{|A|} \tag{4.11}$$

Comparing both solutions A and B in Figure 4.12, solution A is uniformly distributed, and therefore $S(A) < S(B)$.

4.4.3 Convergence and diversity

Finally, the well-known hypervolume metric can measure both convergence and diversity. It calculates the area/volume created by the non dominated front in the objective space, given a reference point. This is done by summing the rectangular areas bounded by some reference point [36]. It has been widely used, despite its computation complexity for increasing number of objectives M, where the hypervolume is $(M-1)$-dimensional. High value of hypervolume is usually a synonym of well-spread and converged solutions.

These calculations are simple in a bi-objective problem, but with increasing number of objectives become highly demanding.

Referring to Figure 4.12, the hypervolume (area in this case) of solution B is bigger than solution's A, therefore is better.

Table 4.3 lists the results of each metric to the bi-objective problem illustrated in Figure 4.12, highlighting the best metric values.

Table 4.3: Indices results to the solutions presented in Figure 4.12

Solution	Hypervolume	Spacing	Set Coverage
A	350.50	0.51	$1/5 = 0.20$
B	362.15	0.61	$7/9 \approx 0.78$

4.5 Search Algorithms Validation

Each algorithm was tested with the three benchmark problems already detailed: DTLZ4, DTLZ7, and MOKP. To each one, 3 different objectives were used: 2, 4 and 10, to verify their performance across a wide set of objectives. A population of size 100 and all default parameters were used.

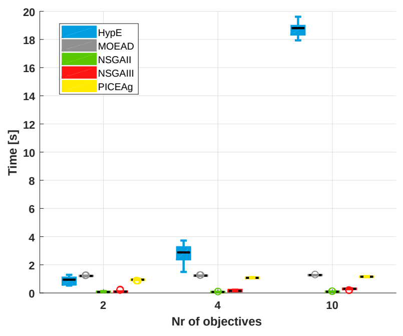
Time Comparison

Starting with a time comparison, and using 2, 4 and 10 objectives to each benchmark problem. The results are graphed in Figure 4.13 and are similar in all benchmark problems, and coincidental to what is expected. As the number of objectives increases HypE becomes increasingly slower, as expected for being an indicator-based algorithm, and even the Monte Carlo approximation for the hypervolume calculation, as mentioned in subsection 4.2.5, is inefficient.

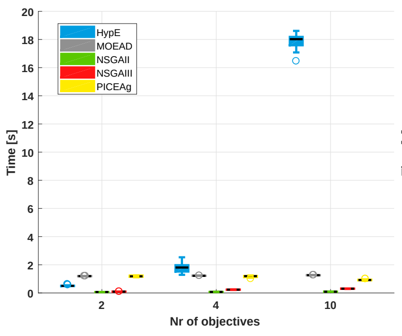
The fastest across all spectrum of objectives is NSGA-II, followed by NSGA-III. MOEA/D and PICEA-g have similar time performance. PICEA-g is the only algorithm that is able to reduce the computation time when dealing with a higher number of objectives.

Hypervolume Comparison

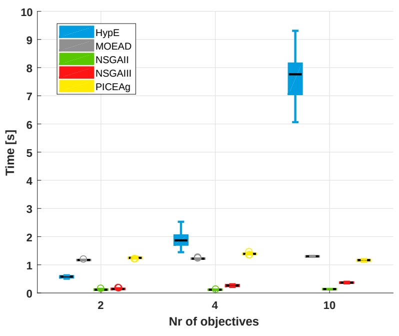
The results of all algorithms relating to hypervolume, the metric that is proportional to both convergence and diversity, and therefore to be maximized, are shown for all benchmark problems and number of objectives in Figure 4.14. The hypervolume values were resized according to Equation 4.12



(a) DTLZ4



(b) DTLZ7



(c) MOKP

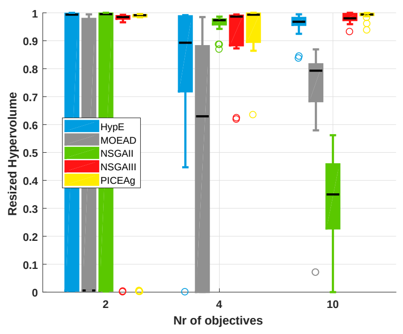
Figure 4.13: Time behaviour of the algorithms in the benchmark problems.

for each objective.

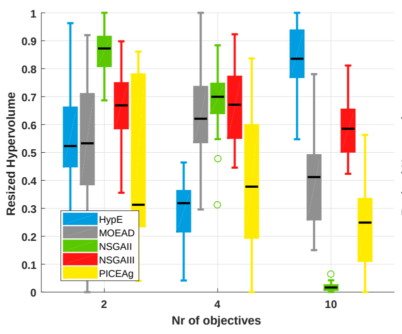
$$x_i' = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \tag{4.12}$$

There's a distinct difference between the continuous DTLZ problems and MOKP. In the first ones, as expected, as the number of objectives increases NSGA-II performance is severely deteriorated, and NSGA-III and HypE stand out. PICEA-g shows poor performance in high-dimensional DTLZ7, similarly to MOEA/D. In all problems, for $M = 4$, the frontier between multi and many-objective problems, NSGA-II is still able to reach high values of hypervolume.

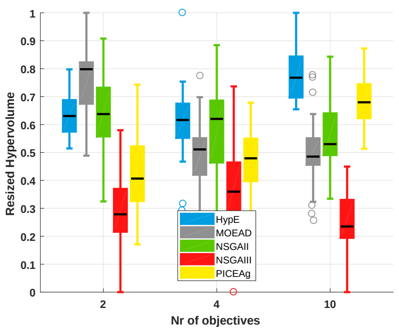
In relation to MOKP, the discrete problem, MOEA/D shows satisfactory performance for the bi-objective problem, but with increasing number of objectives it's surpassed by HypE. NSGA-III is the weakest in this discrete problem, being consistently the worst performer. This is explained by the decomposition mechanism based on normalization, likely not being well-suited for discrete problems.



(a) DTLZ4



(b) DTLZ7

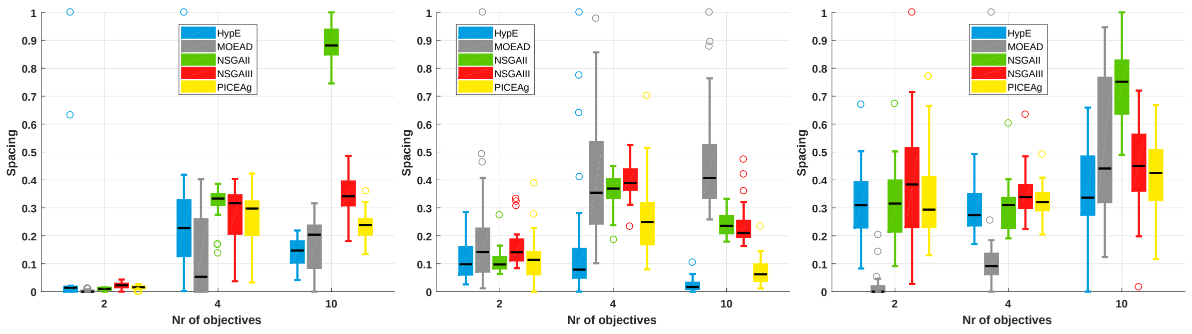


(c) MOKP

Figure 4.14: Hypervolume behaviour of the algorithms.

Spacing Comparison

The spacing analysis also shows expected results. Remembering that a small spacing means the solutions are uniformly spread across the search space, it is visible in Figure 4.15 that there's a pattern in all benchmark problems for NSGA-II to lose its diversity abilities as the number increases. HypE and PICEA-g are clearly the ones more capable of spreading their solutions in continuous problems, while in the discrete problem MOEA/D is better in $M \leqslant 4$, while for $M = 10$ all algorithms' performance is very similar, except for NSGA-II.



(a) DTLZ4 **(b)** DTLZ7 **(c)** MOKP

Figure 4.15: Spacing behaviour of the algorithms.

Set Coverage Comparison

The visualization of set coverage is more difficult, because it involves pairwise comparison. A heatmap would be a good option but wouldn't show the variance analysis of a box plot. Keeping in mind that $C(A, B) =$ %of solutions in B dominated by A. The results for all three problems (DTLZ4, DTLZ7 and MOKP) and for each number of objectives (2,4 and 10) is plotted in Figure 4.16.

The results are very similar to hypervolume's. It's clear that NSGA-II performance is high when dealing with all bi-objective problems since it dominates most others algorithms. HypE also shows great results in a low number of objectives. As the number of objective increases, a domination by NSGA-III, HypE, and PICEA-g is visible, especially in DTLZ4 and DTLZ7. NSGA-II is very poor in those ones, seen by the presence of green bars. Once more, the results in MOKP are somewhat disruptive. NSGA-III's performance is very poor across all number of objectives, and NSGA-II's solutions are also widely dominated. The top performer for a low number of objectives is MOEA/D, but HypE stands out in higher dimensions.

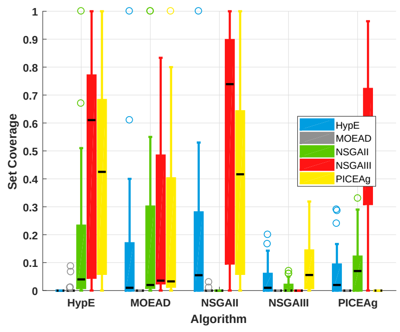
4.6 Encoding for feature selection

In this section, all parameters ranging from general parameters such as population size and stopping criteria, to algorithm-specific parameters are discussed. Some concepts of feature selection are also discussed, in preparation for the results obtained in the next chapter.

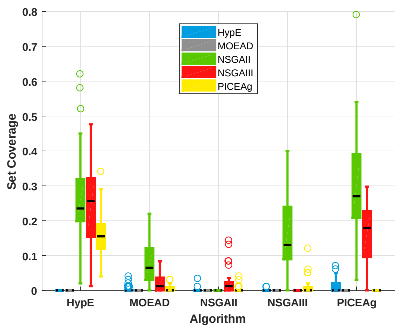
4.6.1 Individual Representation

The first concern is how to represent each solution in the design space. Intuitively this would be made with a vector composed of all features used. In a problem with 10 features that uses the first, third and eighth this would look like $[1, 3, 8]$. However, it's computationally easier for the genetic operators to use a binary vector [5]. In the given example, this would be [1010000100]. The ones obviously mean that the feature is included, and the zeros that it's not.

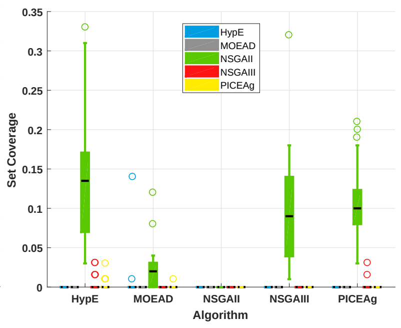
Each individual is assigned a fitness vector, consisting of one cardinality measure, as in Equation 3.1 and multiple performance indexes. All of them are smaller than 1 and are supposed to be maximized. Making it a minimization problem is fairly simple, merely subtracting the performance metrics from



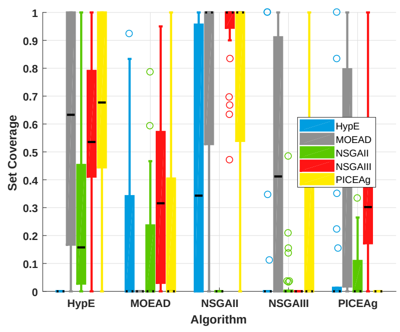
(a) DTLZ4 2 objectives



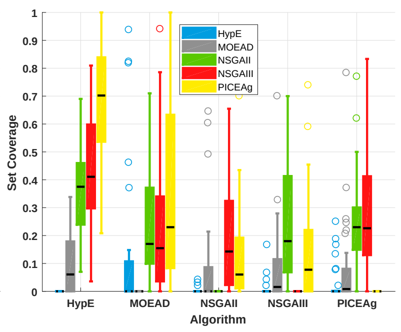
(b) DTLZ4 4 objectives



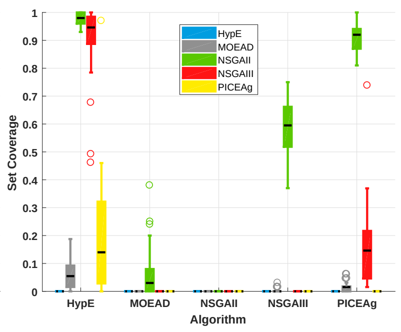
(c) DTLZ4 10 objectives



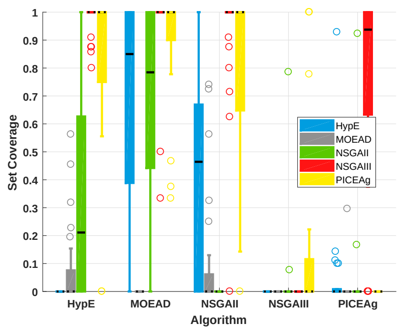
(d) DTLZ7 2 objectives



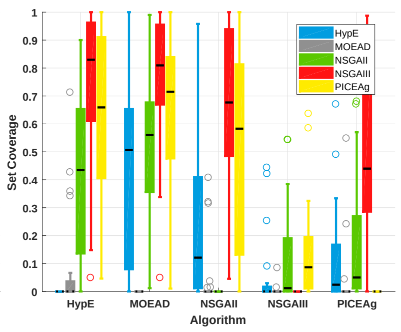
(e) DTLZ7 4 objectives



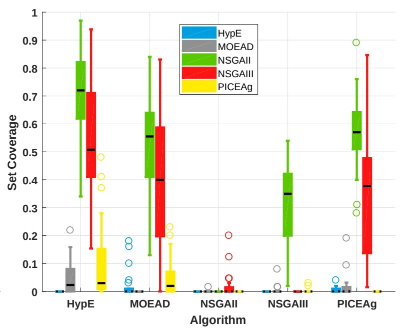
(f) DTLZ7 10 objectives



(g) MOKP 2 objectives



(h) MOKP 4 objectives



(i) MOKP 10 objectives

Figure 4.16: Set coverage behaviour of the algorithms.

1.

4.6.2 Initial Population, Size and Stopping Criteria

The initial population is a major worry in Evolutionary Algorithm (EA), since it will be crucial to the algorithm's genetic operators ability to explore the search space. The approach selected consists of using a uniformly generated random number between 1 and the number of features as the total size of the individual's subset. Afterwards, that amount of features is uniformly sampled at random from 1 to the number of features. And those are the features contained in the subset. This procedure ensures a uniformly spread amount of subset sizes, having approximately the same number of individuals with only one feature and all of them.

Considering that all mentioned algorithms are population-based, the population size must be addressed. Although some algorithms performance might be improved using more individuals per generation, mainly in Pareto-based approaches since it opposes the population getting completely non-dominated, it requires more computing power and also makes the DM task of choosing a final solution harder. Therefore this decision should be made considering only how many different options should the DM have available. It's loosely set to 100 individuals.

The final parameter to set is the stopping criteria. Number of generations will be used since it seems more suitable to fairly compare the results between algorithms.

4.6.3 Genetic Operators

All algorithms chosen are based on EA, and therefore parameters related to genetic operators (selection, crossover, and mutation) should be discussed.

Only NSGA-II and HypE use non-random selection to choose the parents, and both use binary tournament. The first with rank level and crowding distance as second criteria, and HypE with hypervolume-based fitness value of each solution.

Crossover and mutation are a problem of themselves, present in all algorithms. Considering that feature selection is a combinatorial binary vector based problem, crossover usually can be of three types: one-point, two-point or uniform. The most widely used is two-point crossover and that's the approach used. It's quite straightforward: 2 parents are chosen and 2 random points in the binary vector picked. The genome of each parent between those points is switched to create the offspring Figure 4.17.

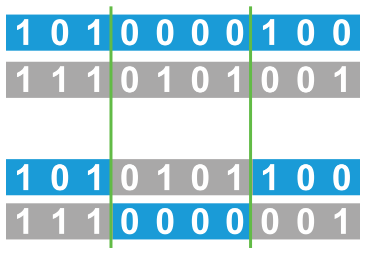


Figure 4.17: Two point crossover

Bitwise mutation is used to add stochasticity to the algorithm. This works by assigning a mutation probability of $p_m = \frac{1}{L}$, where L is the vector's size. Each individual has p_m probability of flipping. This means that, on average, 1 bit will change, but there's also the chance that 2 or more change. The process is illustrated in Figure 4.18.

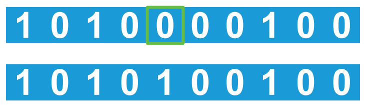


Figure 4.18: Bitwise mutation.

Constraint handling

Constraints are rules that every solution must follow in order to be feasible. All solutions that do not adopt those rules are called infeasible. All EAs are unconstrained by design, this means that they aren't able to deal with these rules by themselves. Coello wrote a deep survey [65] about all the ways to incorporate these rules, the most well known being the penalty functions.

In feature selection there are only two constraints:

1. The solution is a binary vector;
2. At least one feature must be incorporated.

The first one isn't infringed if the initial population is composed of binary vector, as supposed, considering that both crossover and mutations operators are designed to deal with these. However, there's some likelihood that the second constraint is violated and so the algorithm would try to test a solution that uses no features, which is, obviously, infeasible.

Constraint-handling techniques are important when search in infeasible space is necessary in order to reach regions of the feasible region. That's not the case in feature selection since all possible solutions are known and have no major constraint. For that reason a simple penalty function will do, assigning zero performance whenever the algorithm tries to evaluate an empty solution.

4.6.4 Algorithms' parameters

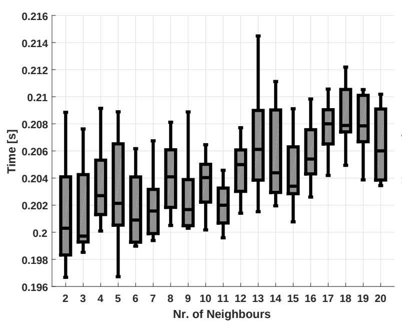
Each algorithm is different and has its own set of parameters. Both NSGA-II and NSGA-III have none aside from genetic operator's parameters, which in fact is one of their strongest assets. NSGA-III only external parameter is the reference set of points, and the authors of the original paper [47] suggest using Das and Denni's method to place H reference points in an (M-1)-dimensional hyperplane, with H being similar to population size.

MOEA/D's only external parameter is the number of neighbours, that is, the number of close subproblems that share information with each other. Despite the claim of the original authors that this parameter does not affect dramatically the algorithm's performance, [51] showed otherwise, regardless of not being able to find an evident conclusion to what value it should take.

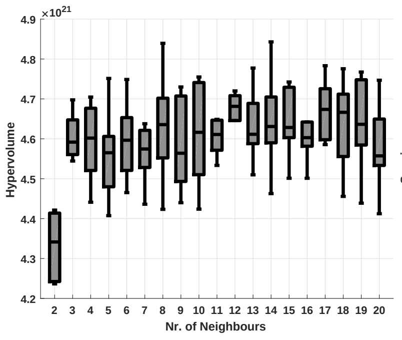
To test this, MOKP benchmark problem was used, since its discrete nature makes it more relatable to feature selection problem. Twenty different simulations were made, varying the number of neighbours from 2 to 20, twenty times each with a population of 30 and 5 objectives. The results are in Figure 4.19. It's evident that with the increase in T, the spacing diminishes, which makes sense since using information from more neighbours allows the solutions to be better spread. Hypervolume sees no evident change, and time only slightly increases.

Similarly to MOEA/D, HypE and PICEA-g only external parameter is a number. Number of sampling points and number of goals, respectively. It has been shown in [51] that both algorithms improve with the increase in these numbers. The decision is, therefore, limited to the appreciation of computational cost. The results obtained are, however, discordant of this analysis. HypE's hypervolume and spacing have only a slight improvement, see Figure 4.20, and the same in PICEA-g.

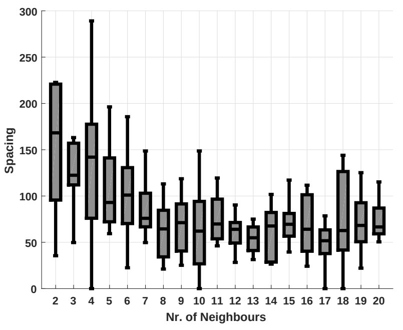
The final chosen parameters are listed in Table 4.4, based on these results and literature such as [51].



(a) Time

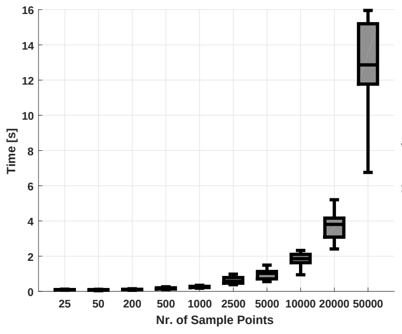


(b) Hypervolume

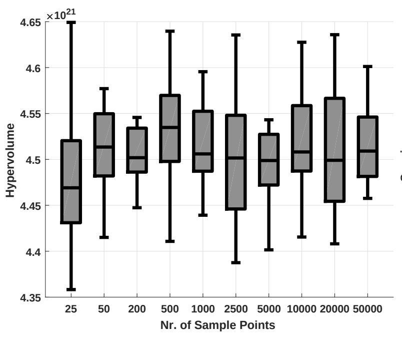


(c) Spacing

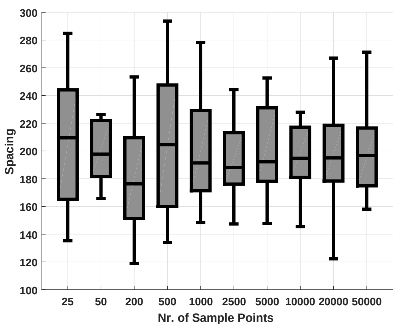
Figure 4.19: Time, hypervolume and spacing dependency on number of neighbours in MOEA/D algorithm. Made using the average of 10 repetitions of MOKP problem with 5 objectives and 500 generations.



(a) Time

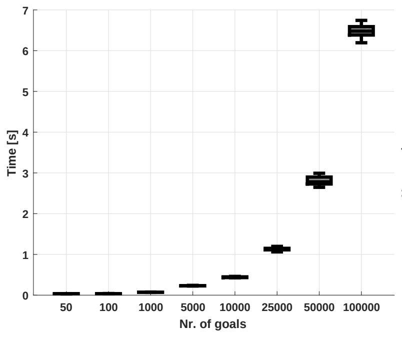


(b) Hypervolume

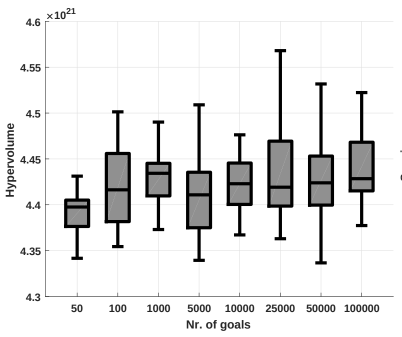


(c) Spacing

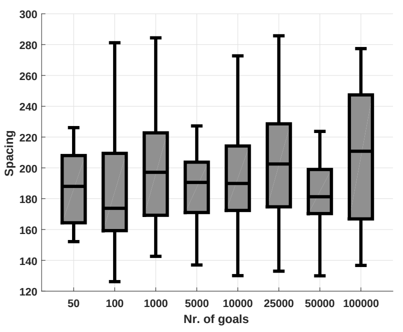
Figure 4.20: Time, hypervolume and spacing dependency on number of samples in HypE algorithm. Made using the average of 10 repetitions of MOKP problem with 5 objectives and 500 generations.



(a) Time



(b) Hypervolume



(c) Spacing

Figure 4.21: Time, hypervolume and spacing dependency on number of goals in PICEA-g algorithm. Made using the average of 10 repetitions of MOKP problem with 5 objectives and 500 generations.

Table 4.4: Parameters values

Algorithm	Parameter Name	Value
HypE	Nr. of Samples	1000
MOEA/D	Nr. of Neighbours	10
NSGA II	-	-
NSGA III	-	-
PICEA-g	Nr. of Goals	5000

5 Results

Contents

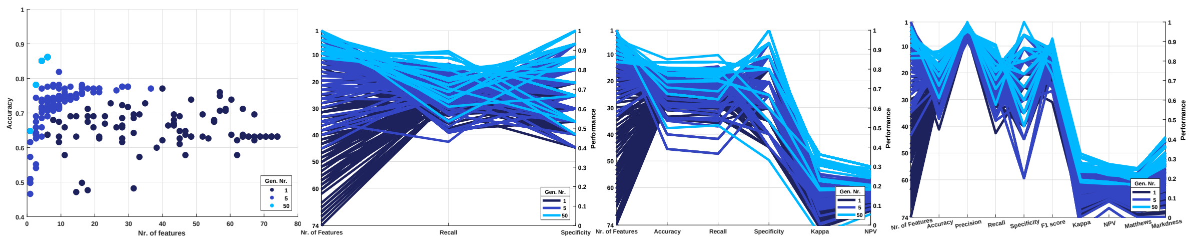
5.1 Algorithm's Validation for Feature Selection 50
5.2 Metrics for Wrapper based Feature Selection 51
5.3 Visual comparison 56
5.4 Multi/Many-Objective Algorithms in Feature Selection 56

All characteristics of a feature selection procedure were discussed up to this point, from the wrapper subset evaluation discussed in chapter 2, the promising sets of wrapper performance metrics in chapter 3, and the multi/many-objective algorithms detailed in chapter 4. Hereupon, all its left is to finally apply all this in feature selection. In the first place, a simple validation is done to verify the correct application of the algorithms to feature selection. After that, a study is made on which set of wrapper's performance metrics are advantageous in the search procedure of both binary and multi-class classification. Thereafter, a comparative analysis between algorithms is made to select the most suitable multi/many-objective algorithm for each kind of classification problem.

5.1 Algorithm's Validation for Feature Selection

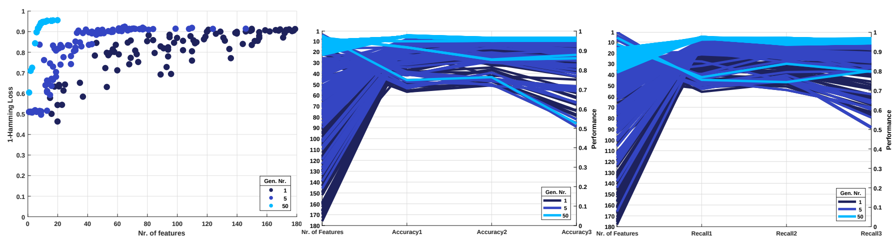
Foremost, a small validation is made to understand if the algorithms presented in the previous chapter are successful in converging to better feature subsets in the immense search space. Figure 5.1 shows the evolution of iteration 1, 5 and 50 of the Sonar dataset using HypE, with two, three, six and ten objectives as discussed in section 3.2. Figure 5.2 shows the same analysis for the DNA dataset, in a first instance by using the Hamming loss and then using accuracy for each class and then recall for each class. For more than two objectives the results are visualized employing the parallel coordinate plot concept, explained in chapter 3.

It's visible in both problems and in all sets of objectives the algorithms' ability to reduce significantly the number of used features while improving or not deteriorating performance in comparison to the initial population. Additionally, final population's wide range of trade-off solutions offered to the DM is visible. Similar results are obtained with other datasets and algorithms.



(a) M=2 **(b)** M=3 **(c)** M=6 **(d)** M=10

Figure 5.1: Evolution of solutions in HypE applied to binary feature selection using Spectf dataset.



(a) M=2 **(b)** M=Acc for each **(c)** M=Recall for each

Figure 5.2: Evolution of solutions in HypE applied to multi-class feature selection using DNA dataset.

5.2 Metrics for Wrapper based Feature Selection

One of this work's main goal is to analyse the potential advantages in using several classifier's performance metrics for a subset evaluation during an feature selection optimization, in comparison to the typical bi-objective optimization using the number of features and a unique performance index. To that end, in section 3.2, sets of metrics were selected as promising, both for binary and multi-class classification.

Succinctly, binary classification uses sets of $M \in \{2, 3, 6, 10\}$ objectives presented in Table 3.3, and multi-class classification uses one of three set of metrics in addition to the number of features: hamming loss, accuracy for each class and recall for each class, as explained in section 3.2.

In order to study this issue, an analysis similar to the one made in the second chapter, using the hypervolume, spacing and set coverage metrics is conducted. The results in section 4.5 suggested that HypE algorithm performed consistently very well in the multi-objective knapsack problem, independently of the number of objectives. Considering the similarities between feature selection and multi-objective knapsack problem, both discrete and binary-coded, HypE was used to conduct this analysis. The analysis consists of comparing the final solutions produced by HypE using the mentioned sets of wrapper's performance metrics.

5.2.1 Binary classification

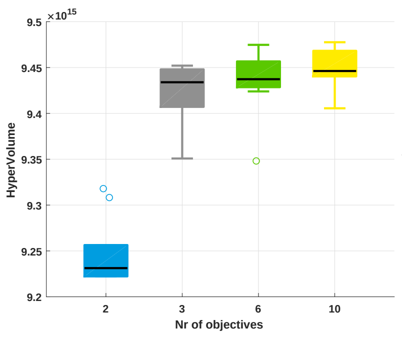
Four sets of metrics were selected as promising in section 3.2, listed in Table 3.3. Each set of metrics was used 20 times for 100 generations composed of 100 individuals, on each dataset using HypE, and the respective final population was stored. The comparison between final populations is made in relation to the biggest set of objectives, which makes sense considering that only converged and diverse solutions are wanted. Therefore the algorithm's final step when using $M \in \{2, 3, 6\}$ was to evaluate once more each solution of the final population using 10 objectives.

Hypervolume Comparison

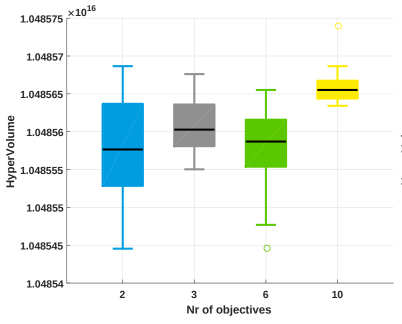
Figure 5.3 presents the boxplot for HypE's hypervolume performance using each set of metrics, for all datasets. It's visible that there's a globally poor performance of the algorithm when using $M = 2$. Despite possibly finding at least one very good solution, using such a low number of objectives yields a very small number of solutions. In this case, HypE's final population when using $M = 2$ contained between 2 and 10 solutions, which are not able to represent trade-off's and offer several options to the DM, which in turn decreases the hypervolume value. This result was already expected because having no other objective to optimize, the algorithm produces a trade-off surface only in those it knows, in this case, number of features and accuracy.

The hypervolume's performance value when using $M = 3$ is also inferior to the rest, except in Musk dataset, exactly for the same reasons: using fewer objectives yields less alternatives and therefore a worse final population in terms of diversity, and perhaps convergence.

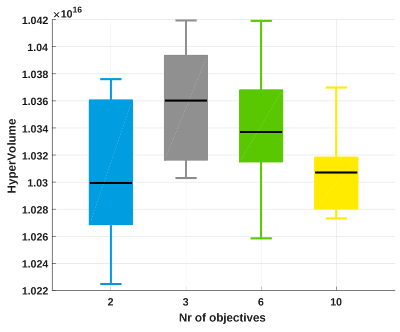
The performance using $M = 10$ or $M = 6$ is very similar, easily explained by the fact that from $M > 6$ no more conflictual wrapper's performance metrics exist, and increasing the number of features does not affect diversity nor convergence, resulting in the same hypervolume value.



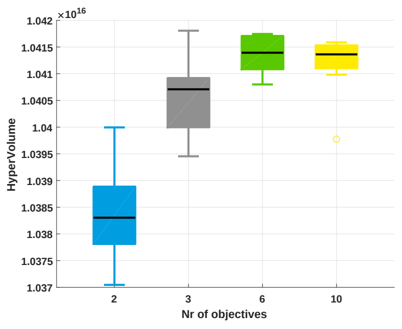
(a) AldoA



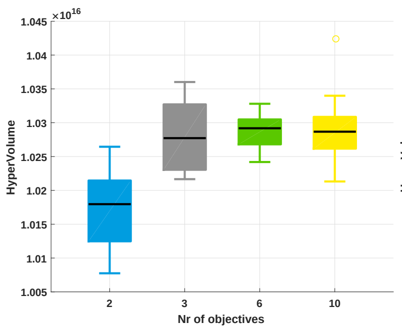
(b) Mushroom



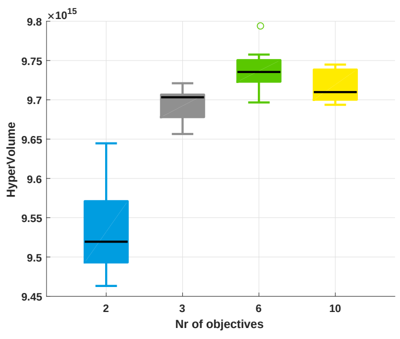
(c) Musk



(d) Phishing



(e) Sonar



(f) Spectf

Figure 5.3: Hypervolume comparison of wrapper's performance metrics for binary classification.

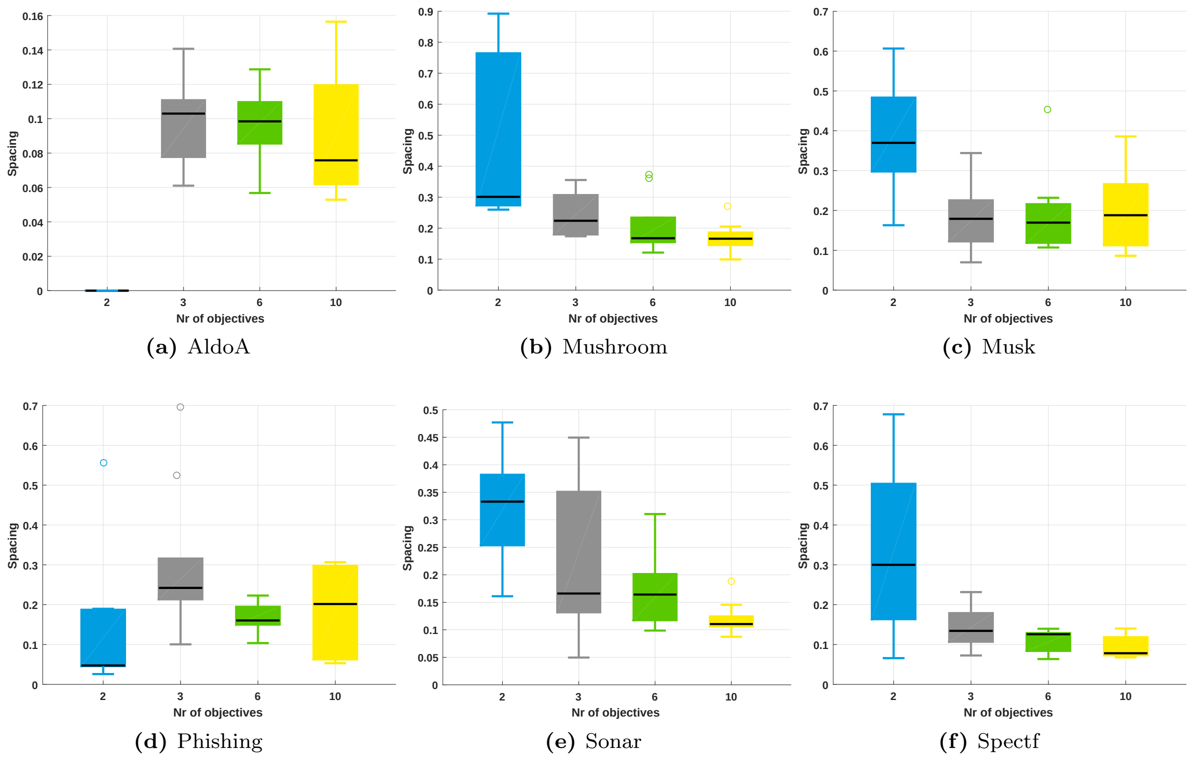
Spacing Comparison

The spacing analysis indicates how well spread the solutions are in the search space and is better when minimized. The results when using HypE are graphed in Figure 5.4. Although the results present some variance, mainly in AldoA, where spacing is zero because it convergences to one unique solution, and Phishing datasets, there's an obvious tendency of improving spacing with higher dimensions, meaning solutions are more uniform and the algorithm is able to faithfully represent the search space.

Set Coverage Comparison

Set coverage metric is the last metric to analyse in binary classification to conclude the analysis of HypE's performance when using sets of increasing number of wrapper's performance metrics in multi/many-objective feature selection. As explained in section 4.4, this metric calculates the percentage of dominated solutions between two populations and is a measure of convergence. The results are shown in Figure 5.5. Although there's some dataset dependency, some general conclusions can be made.

Keeping in mind that each bar represents the percentage of dominated solutions of the respective set of metrics, it's notorious that even though several blue bars are present, related to using $M = 2$, their dominance value is usually lower than the rest. This is easily explained by the small number of well converged final solutions provided when $M = 2$. Despite that, it's visible the high density of bars



(a) AldoA (b) Mushroom (c) Musk (d) Phishing (e) Sonar (f) Spectf

Figure 5.4: Spacing comparison of wrapper's performance metrics for binary classification.

when $M = 10$, dominating all others, and the low valued yellow bars, meaning that it's not usually highly dominated.

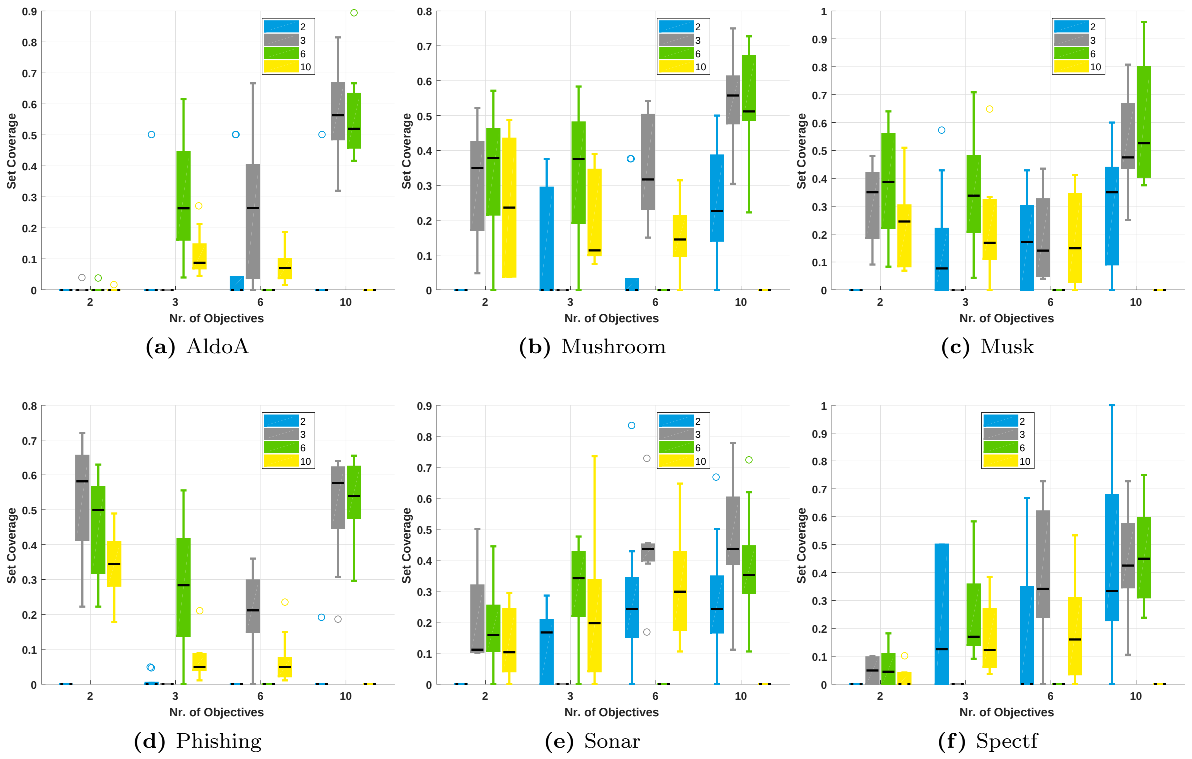
Considering hypervolume, spacing, and set coverage metrics, the conclusion is that using several wrapper's performance metrics in the feature selection process, even if some are redundant, yields better converged and diversified solutions. Not only is the hypervolume and spacing metrics better, meaning that using $M = 10$ yield solutions that represent a wide area of the search space in an uniform form, but additionally the set coverage metric seems to suggest that those solutions are usually better converged.

5.2.2 Multi-class classification

Multi-class classification is subjected to the same analysis, comparing three different sets of metrics: using Hamming score, accuracy for each class and recall for each class. All of them, in similarity to what was done in binary classification, use the number of objectives as a first objective.

In order to compare between the three populations, all having completely different search spaces, two of them should be transformed into the other's search space. Therefore, all solutions resulted of using Hamming and recall for each class were evaluated once more using accuracy for each class. This decision could've been made to transform it to recall for each class, but accuracy was chosen based on the discussion of section 1.3.

Once again HypE is used, with 100 generations each composed of 100 individuals, repeated 20 times.



(a) AldoA **(b)** Mushroom **(c)** Musk **(d)** Phishing **(e)** Sonar **(f)** Spectf

Figure 5.5: Set coverage comparison of wrapper's performance metrics for binary classification.

Hypervolume Comparison

In Figure 5.6, HypE's hypervolume's results using the three distinct set of metrics is compared. Careful examination of the similarity between y-axis values leads to conclude there is no notorious difference between the three sets' values, despite some relative superior performance when using accuracy for each class.

Spacing Comparison

The spacing analysis of the same solutions is graphed in Figure 5.7, in which smaller values indicate better representation of the search space. In all datasets using more metrics, either by using accuracy or recall, proves to result in a more uniform spread set of solutions.

Set Coverage Comparison

Set coverage results for each dataset, presented in Figure 5.8, seems to be inconsequential to this analysis, except for DNA dataset. In all others, the percentage of dominated solutions is insignificant, especially considering that it's normal for a higher number of objectives to have an exponentially greater number of solutions, as explained in subsection 4.1.1, and therefore for a small percentage of those solutions to be dominated is insignificant.

Despite that, in 5.8a, the bi-objective problem clearly dominates all, but only in that dataset. Nevertheless, it seems that using recall yields poorer solutions than using accuracy.

The conclusion of this three-fold analysis using hypervolume, spacing and set coverage, is that

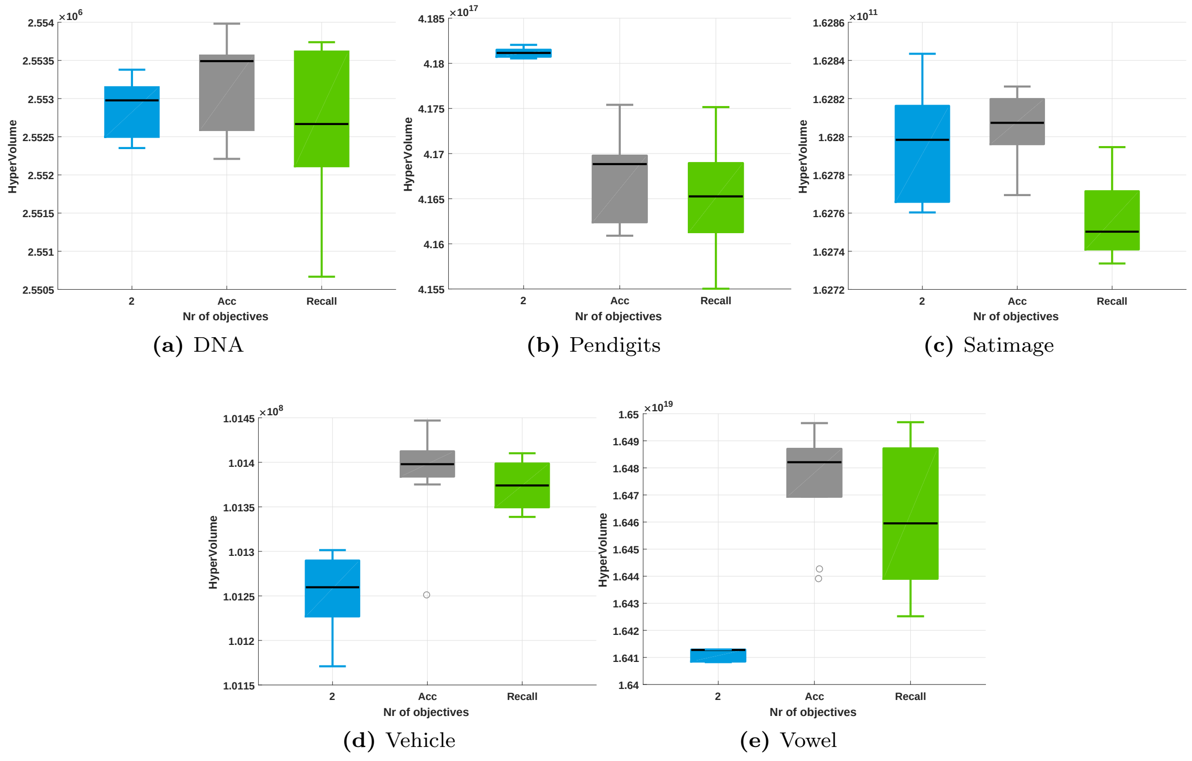


Figure 5.6: Hypervolume comparison of wrapper's performance metrics for multi-class classification.

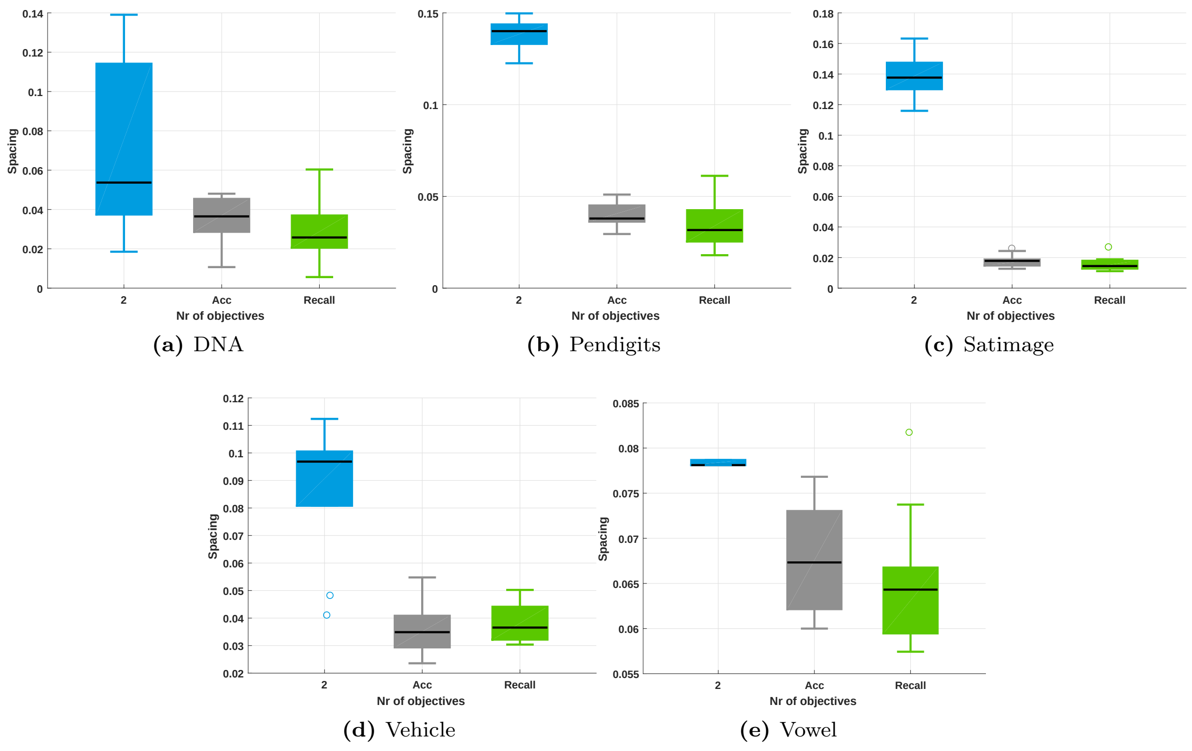
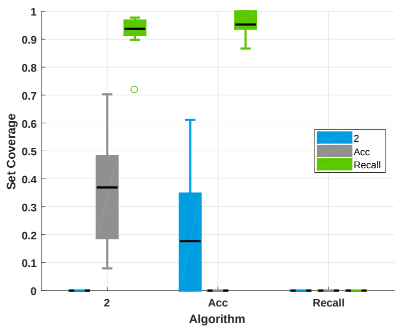
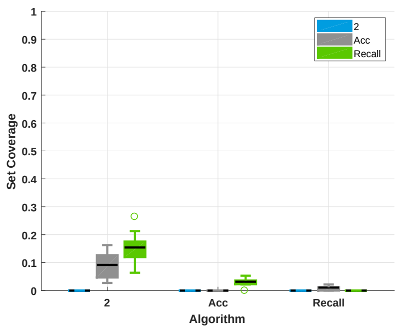


Figure 5.7: Spacing comparison of wrapper's performance metrics for multi-class classification.

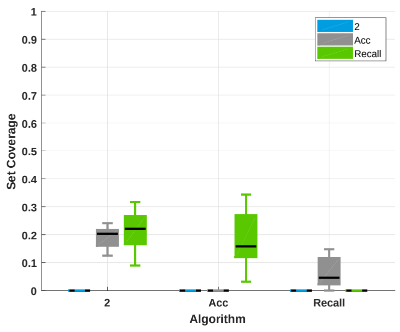
there's no remarkable advantage in dividing a multi-class classification into several binary sub-problems in terms of convergence. However, hypervolume and spacing analysis suggest that this division yields a slightly more diverse and well-spread set of solutions, and therefore is preferred.



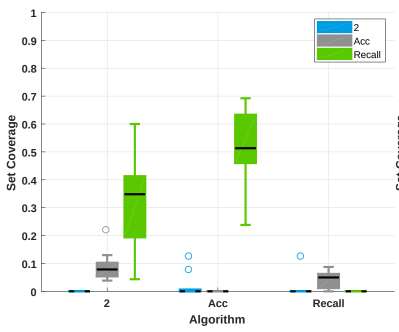
(a) DNA



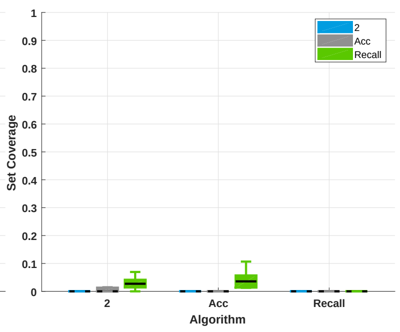
(b) Pendigits



(c) Satimage



(d) Vehicle



(e) Vowel

Figure 5.8: Set Coverage comparison of wrapper's performance metrics for multi-class classification.

5.3 Visual comparison

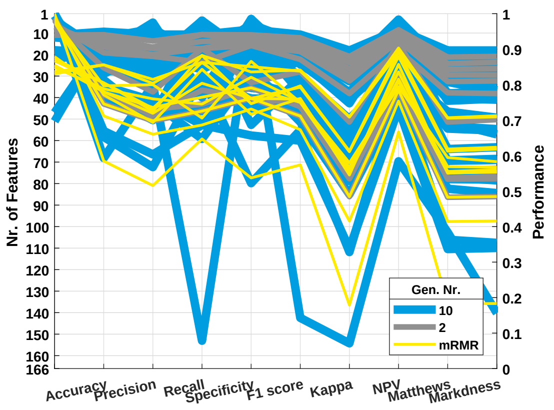
The results obtained were visualized in box plots (or whisker diagrams), which are great for variance analysis but can lack a certain perceptibility. Figure 5.9 compares typical approaches to feature selection with the proposed one, using the objectives suggested in this work. Both compare the results of using many objectives, two objectives, and mRMR [66] feature selection, a well-known filter approach, in binary dataset Musk, and in the multi-class dataset Vehicle.

The increase in performance in the binary dataset when using several objectives, with a higher number of solutions and capable of finding "peak solutions", i.e., solutions that stand out in one objective and have medium/poor performance in the others. Therefore, in binary classification, using several objectives not only provides the DM with a wider range of solutions, but those solutions are better.

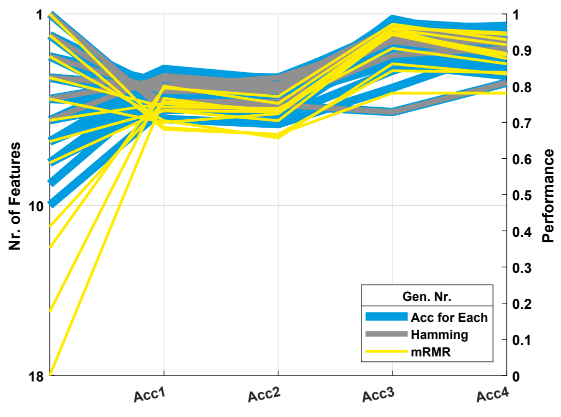
The same does not happen in multi-class classification, where using accuracy to each class yields a more diverse set of solutions, but not necessarily solutions that are significantly better converged.

5.4 Multi/Many-Objective Algorithms in Feature Selection

The last subsection concluded that using several wrapper's performance metrics in binary classification yields better performance of multi-objective algorithms in feature selection, both in terms of convergence and in diversity. As to multi-class classification, despite not resulting in better convergence, it was shown that dividing the problem into binary sub-problems yields a better representation of the search space.



(a) Musk binary dataset



(b) Vehicle multi-class dataset

Figure 5.9: Suggested approach versus traditional approach and mRMR.

Consequently, in both types of classification, the results require the handling of several objectives. This section compares all five algorithms detailed in chapter 4 to discover which one is more suited to use with 10 objectives in binary classification, and with the accuracy to each class in multi-class classification. The same analysis of hypervolume, spacing and set coverage was made, to which a time comparison was added.

5.4.1 Binary Classification

Using the 10 objectives detailed in chapter 3, 30 simulations of each algorithm for each dataset, using 100 generations of 100 individuals, were carried out. Considering the high number of objectives, better performance from algorithms designed for many-objective problems, such as NSGA-III, is expected. Nevertheless, redundancy between objectives was proven in chapter 3, so the actual number of objectives might decrease from 10 to 4 or 5. Moreover, NSGA-III has a proven track of being outperformed by its predecessor in discrete problems such as MOKP [48].

Time Comparison

In Figure 4.13, the slowest algorithm for the multi-objective knapsack problem, which is, similarly to feature selection, discrete and binary-coded, was HypE, followed by PICEA-g. In binary feature selection, the results are quite different. Only Spectf dataset results are shown for simplicity, Figure 5.10, but the remaining datasets have equivalent results. MOEA/D is now the slowest, followed by HypE. NSGAII and NSGAIII are both of very fast computation.

Hypervolume Comparison

Figure 5.11 compares the hypervolume of each algorithm's final population in all six binary datasets. Only in Musk isn't NSGA-II the top performer, meaning that generally, NSGA-II is able to better describe the best part of the search space. The remaining algorithms show also good overall performance, with an honourable mention to NSGA-III.

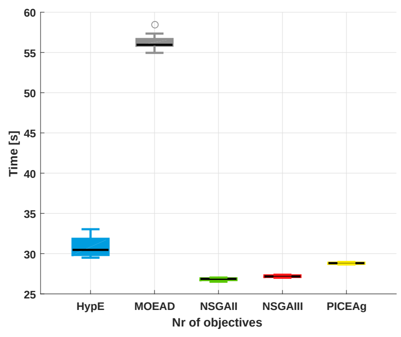


Figure 5.10: Running time performance of the algorithms in binary dataset Spectf.

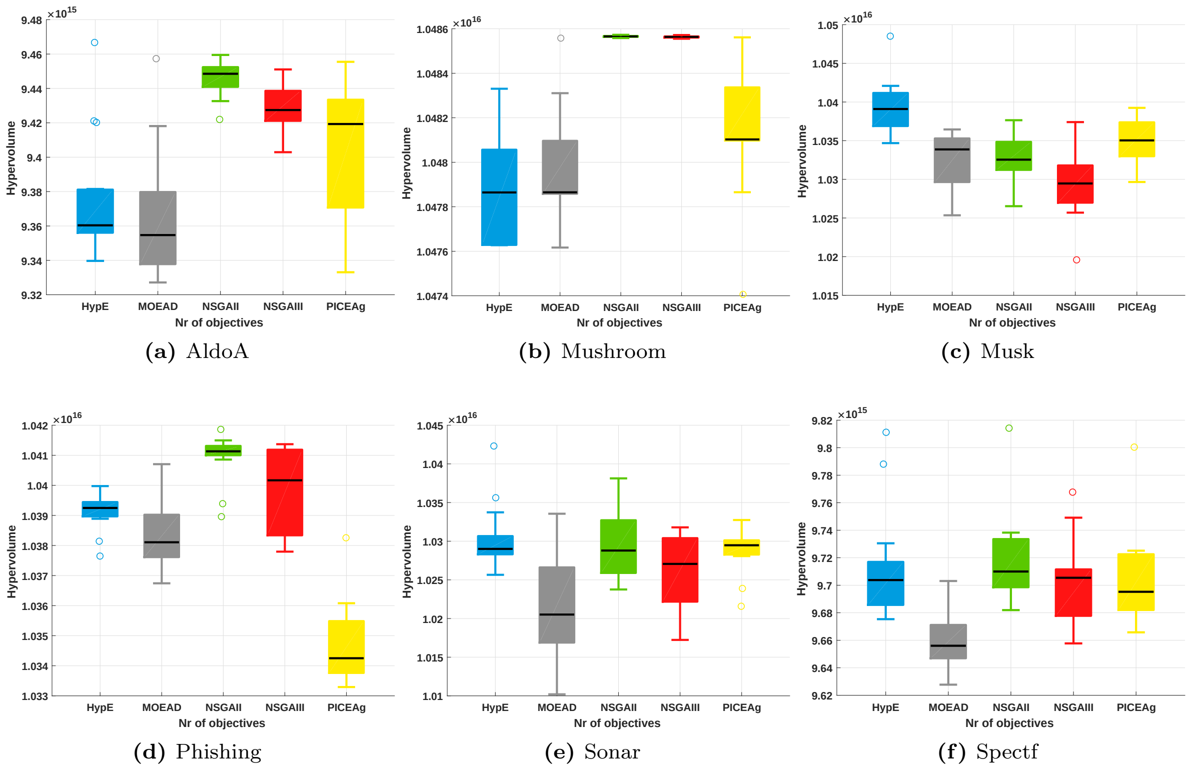


Figure 5.11: Hypervolume performance of the algorithms in binary datasets.

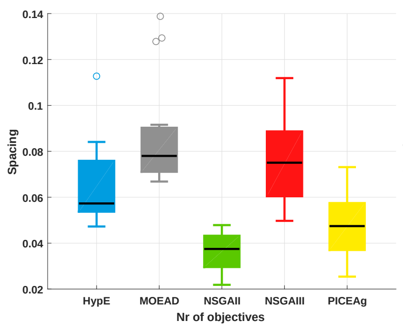
Spacing Comparison

Again in spacing NSGA-II proves to be the top performer, except in Mushroom dataset, Figure 5.12, suggesting it's able to spread its solutions uniformly across the dataset. All other algorithms have reasonable spacing performance, with MOEA/D struggling the most.

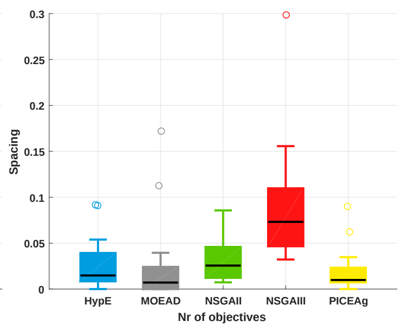
Set Coverage Comparison

Set coverage analysis confirms that NSGA-II is the best algorithm for binary feature selection using 10 objectives. Only in Phishing dataset, the solutions of HypE dominate the ones produced by NSGA-II, while in the remaining datasets these are rarely dominated.

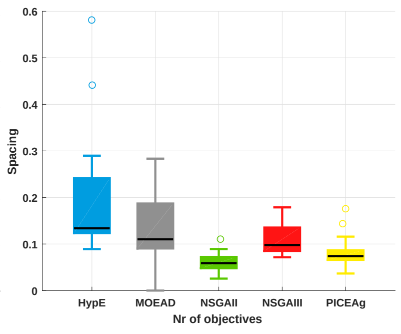
This three-fold analysis indicates that NSGA-II is the top-performer regarding all indicators (hypervolume, spacing and set coverage) when using all 10 objectives. This is most likely a result of objective's redundancy, considering that NSGA-II usually has poor performance in high dimensional



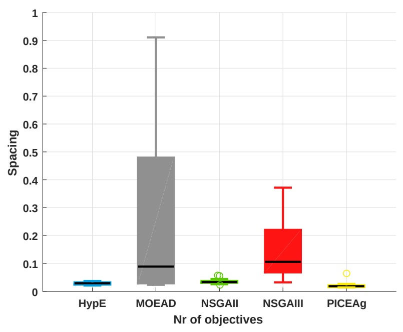
(a) AldoA



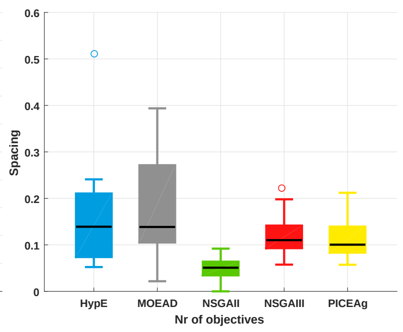
(b) Mushroom



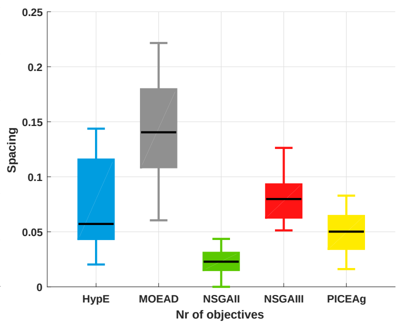
(c) Mushk



(d) Phishing



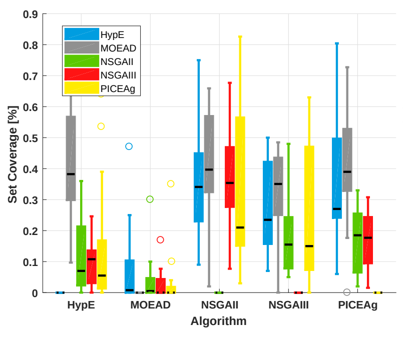
(e) Sonar



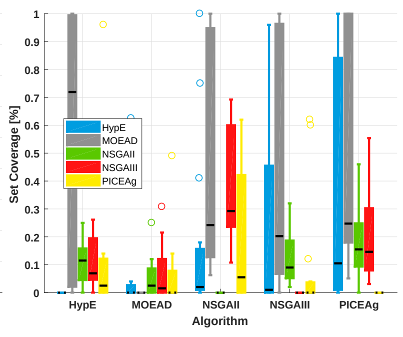
(f) Spectf

Figure 5.12: Spacing performance of the algorithms in binary datasets.

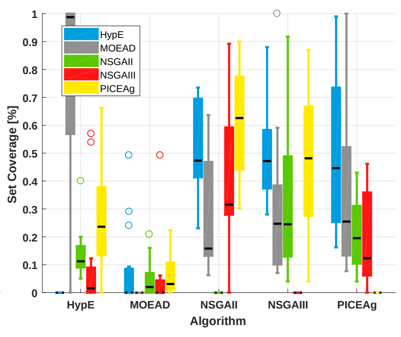
problems.



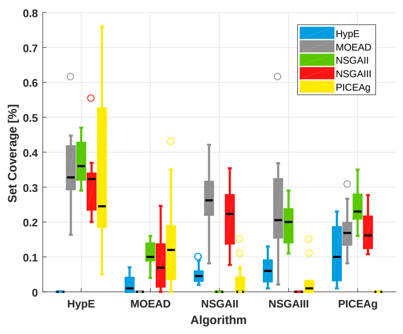
(a) AldoA



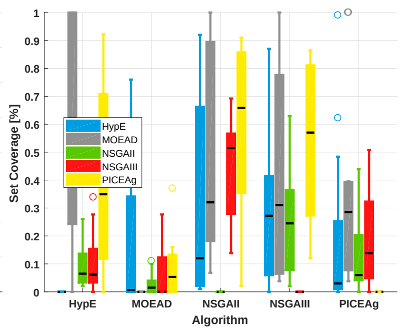
(b) Mushroom



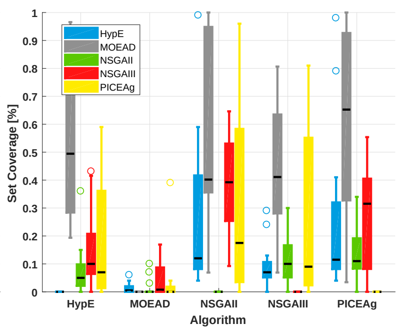
(c) Musk



(d) Phishing



(e) Sonar



(f) Spectf

Figure 5.13: Set coverage performance of the algorithms in binary datasets.

5.4.2 Multi-class Classification

The same analysis is now done to multi-class classification using accuracy to each class, as selected in the previous section. This analysis might be harder considering that datasets have different number of classes and therefore will have different number of objectives, so algorithms performance fluctuation is expected. Once again, 30 simulations of each algorithm for each dataset, using 100 generations of 100 individuals, were carried out.

Time Comparison

The average consumed time in performing 100 generations of the algorithms, Figure 5.14, has similar relative results than binary classification, with MOEA/D being the slowest, followed by HypE, and NSGA-II being the fastest. Similar results were obtained for the remaining datasets.

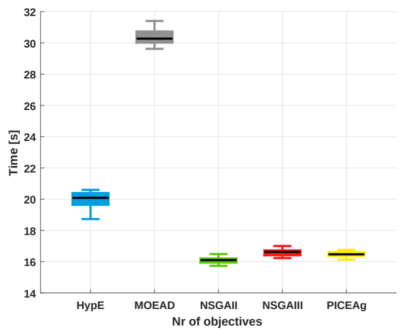


Figure 5.14: Running time of the algorithms in multi-class dataset Vehicle.

Hypervolume Comparison

Hypervolume comparison shows, similarly to binary classification, the general highest performance from NSGA-II, closely followed by HypE and PICEA-g, and independently of the datasets' number of classes, suggesting these can describe the search space in a superior way.

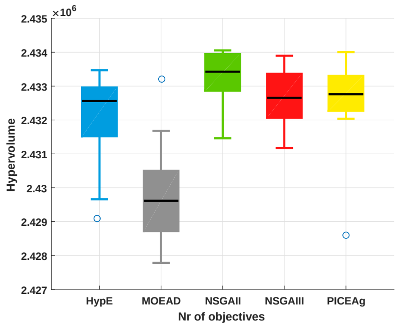
Spacing Comparison

Spacing analysis, Figure 5.16, suggests an similar ability of the five algorithms in dispersing solutions in multi-class feature selection.

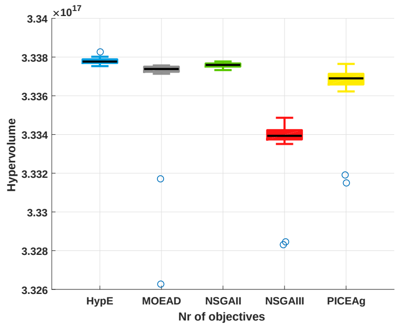
Set Coverage Comparison

Analysing the set coverage results in Figure 5.17, the dominance of a great percentage of NSGA-II solutions by all other algorithms is striking, suggesting that NSGA-II has trouble in finding good solutions even in datasets with a low number of classes such as DNA. HypE and PICEA-g are superior, with a slight advantage to the first.

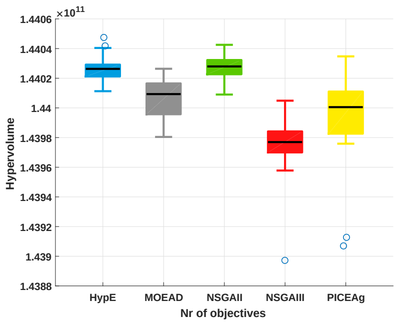
In conclusion, HypE is the most consistent algorithm in multi-objective feature selection in multi-class classification, despite achieving slightly lower results of hypervolume and spacing, but showing, through set coverage analysis, that it's able to reach solutions with higher performance that dominate populations from other algorithms.



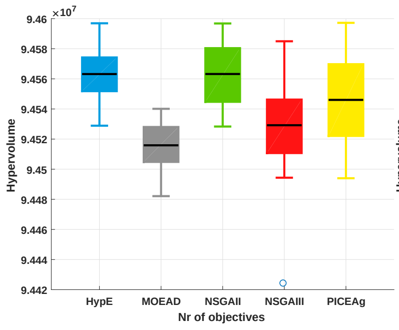
(a) DNA



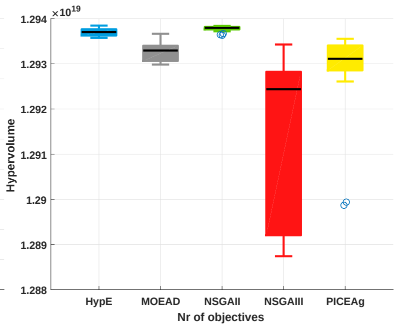
(b) Pendigits



(c) Satimage

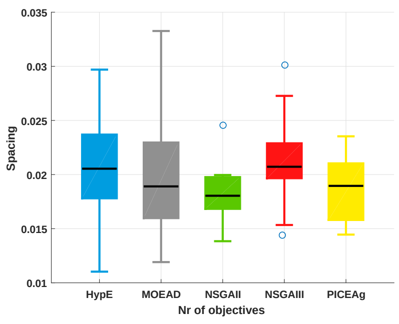


(d) Vehicle

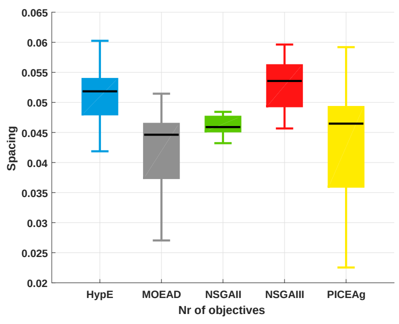


(e) Vowel

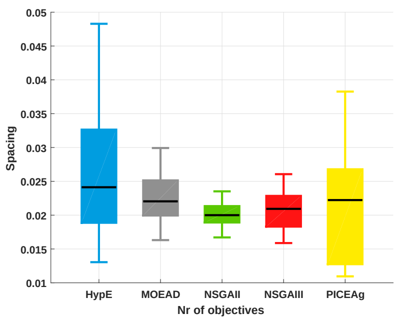
Figure 5.15: Hypervolume performance of the algorithms in multi-class datasets



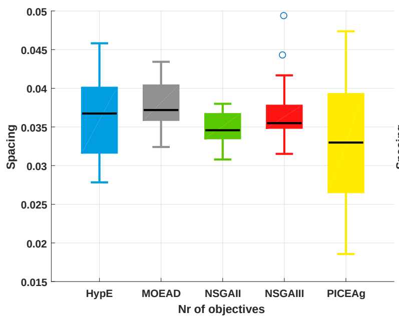
(a) DNA



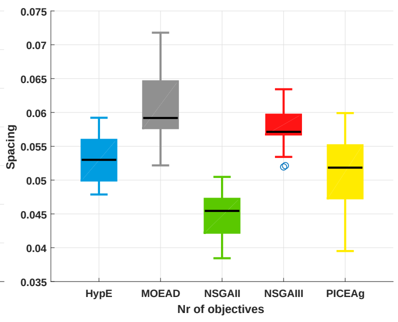
(b) Pendigits



(c) Satimage

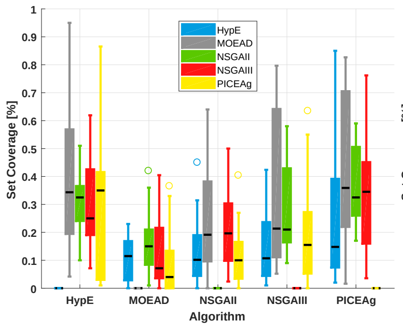


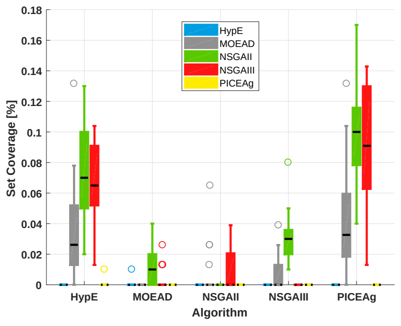
(d) Vehicle

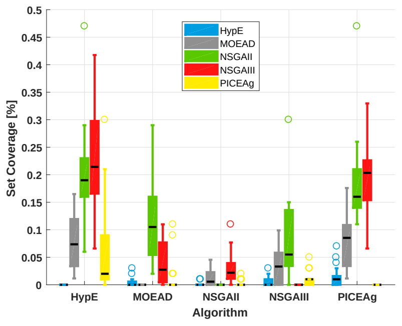


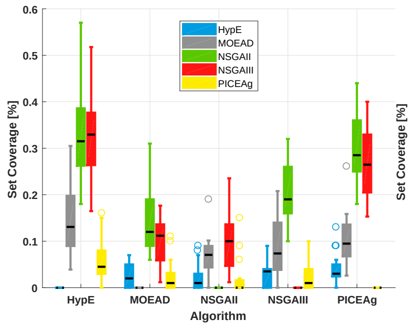
(e) Vowel

Figure 5.16: Spacing performance of the algorithms in multi-class datasets.


(a) DNA


(b) Pendigits


(c) Satimage


(d) Vehicle

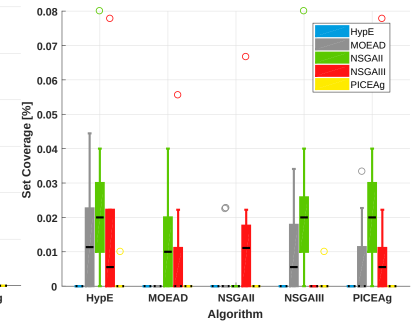

(e) Vowel

Figure 5.17: Set coverage performance of the algorithms in multi-class datasets.

6
Decision Interface

All stages of feature selection were discussed so far, starting by the solution evaluation in chapter 2 and chapter 3, and then the search procedure in chapter 4. The results shown in chapter 5 suggested that using more objectives is beneficial for the search. However, the algorithms chosen provide several dozens of different solutions representing trade-offs between each other, making the selection process rather difficult. In this small chapter, a possible solution to this problem is presented.

Aside from the advantages in searching for the best feature subsets, there's a need to provide the DM several subset's performance metrics, allowing them to set their own preferences and select the subsets based on their priorities, as explained in section 1.3. There's a growing research area aiming at comparing, ranking and selecting several conflictual criteria, called Multi-Criteria Decision Making (MCDM). For a thorough introduction to this field of study the reader is referred to [67].

The populations resulting from the previously discussed algorithms generally contains more than 30 solutions. Figure 6.1 shows NSGA-II's final population of Spectf dataset, using a parallel coordinates plot. With so many solutions and objectives, the selection process is similar to finding "a needle in a haystack".

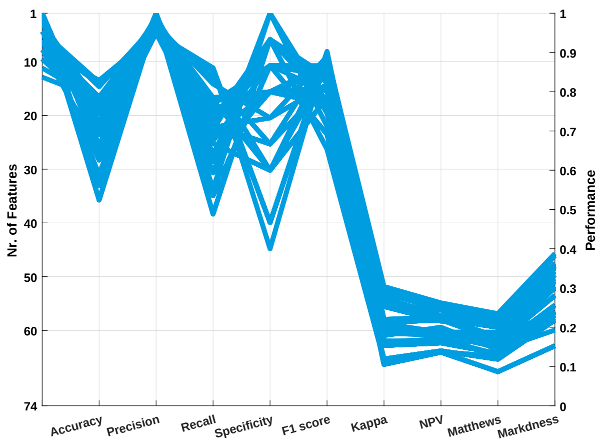


Figure 6.1: NSGA-II's exemplary final population of Spectf dataset.

To help in the decision-making process, an interface was designed and programmed in MatLab. This interface, illustrated in Figure 6.2 consists of two different viewing areas: the largest one uses a parallel coordinate plot to show all solutions, coloured according to the number of features. The second area, under the first, shows the sum of features used by the selected solutions, using a bar plot. Several features of this decision interface are now explained.

Minimum values

Draggable horizontal lines are available in each objective zero that allow the user to set a minimum threshold value for individual objectives, filtering low-performance solutions in respect to those objectives. An example is shown in Figure 6.3, where recall was set to a minimum of $\approx 65\%$, and solution's filtering is visible in comparison to Figure 6.2.

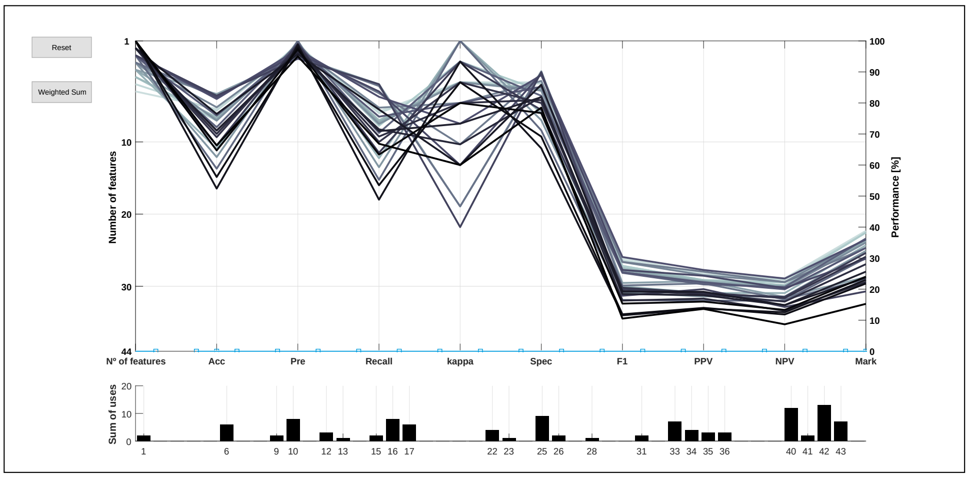


Figure 6.2: Decision Making Interface for Feature Selection.

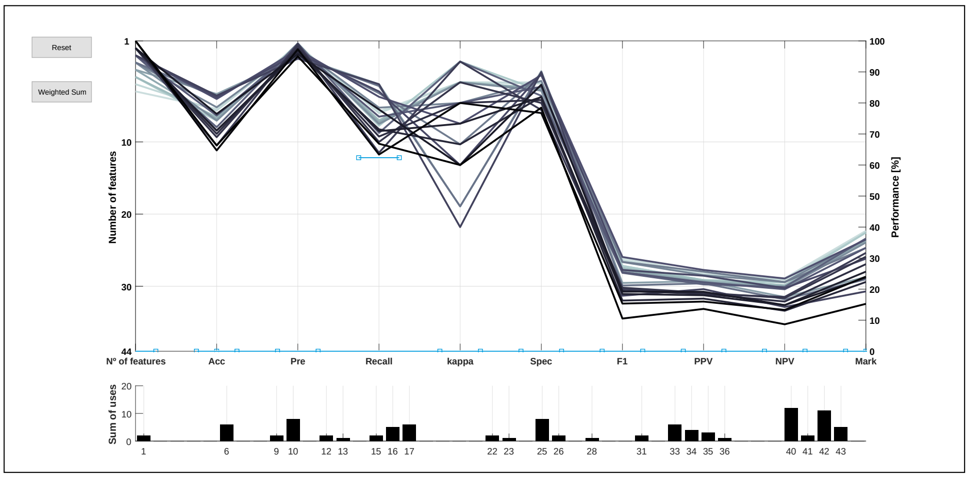


Figure 6.3: Draggable lines example.

Clickable

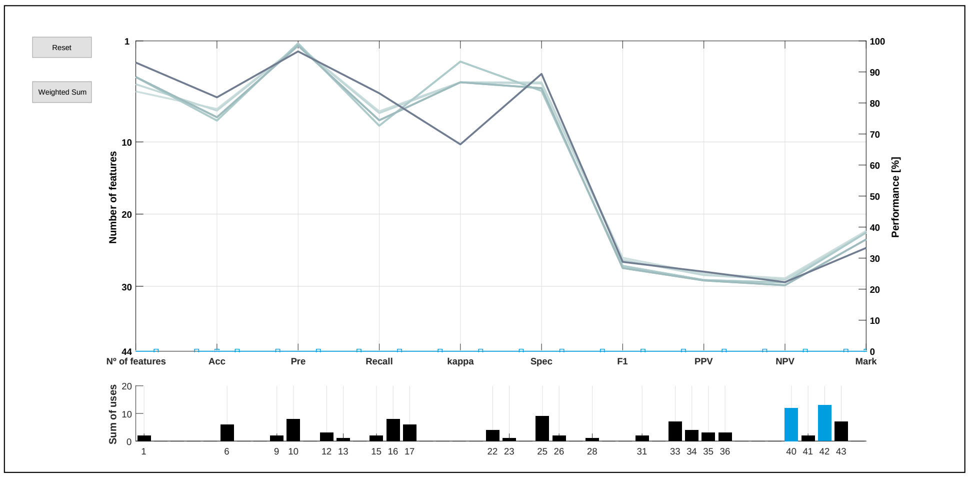
Both the parallel coordinate plot representing solutions' search space and the bar plot below, representing solutions' design space, are clickable. When the DM clicks in a solution in the parallel coordinate plot it's highlighted and the bar plot shows which features are used by that particular solution. Several solutions can be clicked and highlighted.

If in the bar plot a specific bar is clicked, only solutions using that feature are displayed in the parallel coordinates plot.

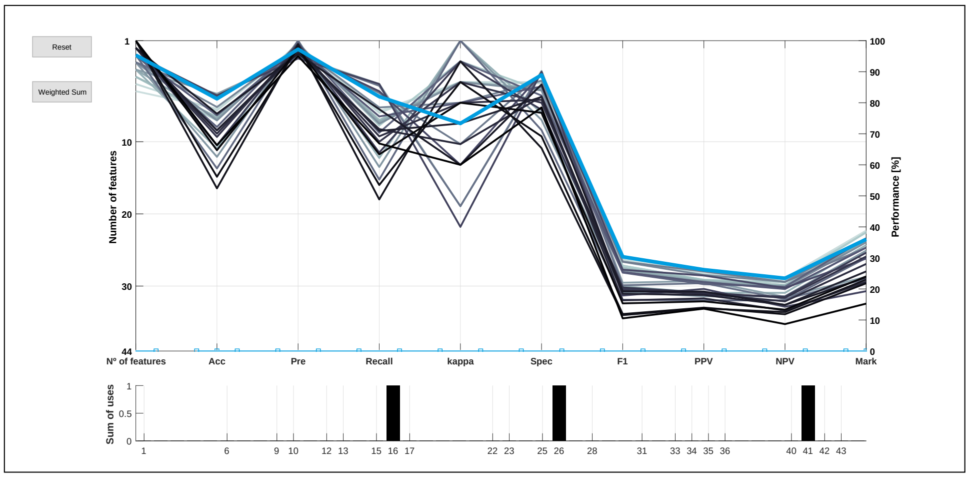
These two characteristics work simultaneously and very smoothly, and are illustrated separately in Figure 6.4.

Weighted Sum

If a user does not have specific threshold values to aim at, the decision is harder. Several methods could be used in the decision-making process, but for validation purposes, only the simplest one was implemented. The weighted sum method ranks each one of the N solutions based on weights w_j assigned to each objective j, as in Equation 6.1. Solutions with greater S_i are selected.



(a) Click on feature



(b) Click on solution

Figure 6.4: Clicking examples.

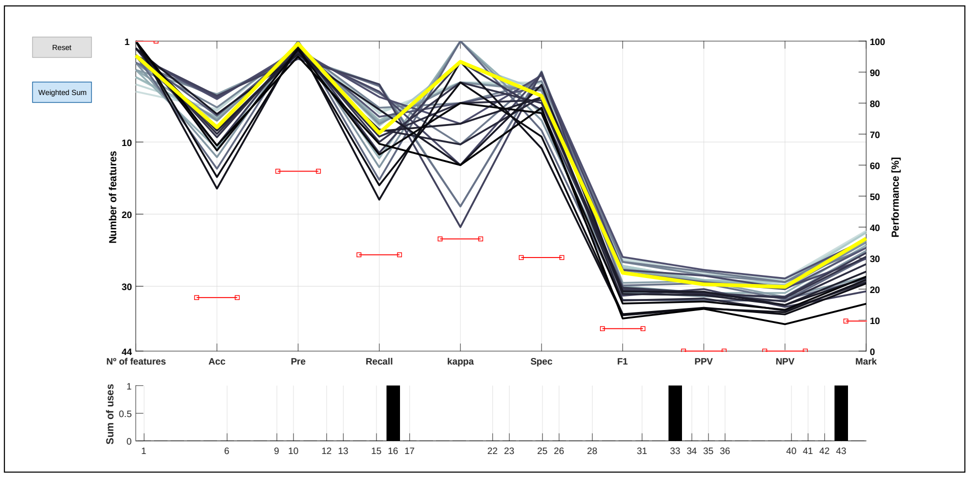


Figure 6.5: Weighted sum example.

$$S_i = \sum_{j=1}^{M} w_j p_{ij}, \text{ for } i = 1, 2, ..., N \tag{6.1}$$

When pressing the button ”Weighted Sum”, the user is presented with red draggable lines, pre-set at

$w = 0.5$. Dragging these lines changes each solution's S_i, and the ones with maximum weighted sum are highlighted in yellow. This process is presented in Figure 6.5.

Other interface features such as objective removal, y-axis control or fuzzy decision making are of great promise but are left for the second version of this prototype.

7 Conclusions and Future Work

Contents

7.1 Summary 70
7.2 Future Work 70

7.1 Summary

A need for giving a DM the freedom to choose solutions based on their preference after the optimization process is over was argued in the first chapter, allowing them to input information only in the final stage of the feature selection process. Moreover, the hypothesis was postulated that using several classifier's performance metrics would improve the search for new feature subsets, yielding subsets with better performance but also more effective in the search space representation, having well spread solutions. Additionally, wrapper subset evaluation, which requires the training and testing of a model, combined with meta-heuristics was selected as the most promising approach for feature selection [5], and for that purpose, DT were chosen as a classifier, in addition to six and five datasets for binary and multi-class classification, respectively.

Thereon, the first analysis studied studying wrapper's performance metrics relationship, in order to foresee which are redundant. Results showed that despite some redundancy between metrics, in binary classification at least a handful of them are conflictual, i.e., are deteriorated when the other one is optimized, thus justifying the need for multi-objective optimization. Likewise, multi-class classification shown strong levels of conflict when divided into N sub-problems of accuracy or recall, with N being the number of classes. Both these results made great prospects to multi-objective optimization.

In that direction, an initial introduction to the world of multi-objective optimization was made, discussing general concepts and selecting five state of the art algorithms to be used, as well as ways of comparing the performance of these algorithms (hypervolume, spacing and set coverage metrics) both in terms of convergence and diversity.

The subsequent chapter finally applied the algorithms to feature selection, and shown there are indeed advantages in using several objectives in the binary datasets in comparison to the traditional bi-objective problem, yielding better converged solutions and also a more diverse set of solutions. In multi-class classification, only diversity was improved. In both cases, the number of different solutions provided was exponentially bigger, giving the DM the opportunity to apply their own preferences. In the same chapter, a comparison between algorithms for binary and multi-class classification was made, indicating that for the first type of problems NSGA-II is the top-performer, while for the latter HypE is the best option.

In addition to the increased performance in finding better feature subsets, raising the number of objectives generates populations exponentially bigger, with several dozens of non-dominated solutions, making the subset selection process. To help in this process, a decision interface was designed and built as the first version a tool to help a DM "find a needle in the haystack", with several interactive features.

7.2 Future Work

One of this work's goal is to further study the multi-objective feature selection field, while inspiring and opening the way for future research in this area. Foremost, a deeper study on the wrapper's performance metrics to use should be made, ideally seeking conclusions applicable to all datasets, and enlarging the set of metrics being considered.

As research in the field of multi/many-objective optimization unfolds, other algorithms can be easily tested in this problem, especially with the increasing supply of algorithms from tools such as jMetal [68] for java, PlatEmo [53] for MatLab, or Shark [69] for C++. These libraries provide several algorithms easily accessible and editable.

In addition to test other algorithms in the feature selection conundrum, finding the set of metrics more advantageous for multi-objective optimization could be made in a optimization process itself.

Finally, the decision interface is of great potential interest to feature selection as a whole. Future steps include improving flexibility to the interface, allowing y-axis control, displaying only selected objectives, and making it more visually attractive. Additionally, other MCDM methods such as fuzzy decision making could be applied, truly making this an indispensable tool for any feature selection process.

Bibliography

[1] W. Siedlecki and J. Sklansky, “A note on genetic algorithms for large-scale feature selection,” *Pattern Recognition Letters*, vol. 10, no. 5, pp. 335–347, 1989.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From Data Mining to Knowledge Discovery in Databases,” *AI Magazine*, vol. 17, no. 3, p. 37, 1996.

[3] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection,” *Journal of Machine Learning Research (JMLR)*, vol. 3, no. 3, pp. 1157–1182, 2003.

[4] E. Amaldi and V. Kann, “On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems,” *Theoretical Computer Science*, vol. 209, no. 1-2, pp. 237–260, 1998.

[5] M. Dash and H. Liu, “Feature selection for classification,” *Intelligent Data Analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[6] A. W. Whitney, “A direct method of nonparametric measurement selection,” *IEEE Trans. Comput.*, vol. 20, no. 9, pp. 1100–1103, Sep. 1971.

[7] T. Marill and D. Green, “On the effectiveness of receptors in recognition systems,” *IEEE Trans. Inf. Theor.*, vol. 9, no. 1, pp. 11–17, Sep. 2006.

[8] K. Kira and L. Rendell, “The feature selection problem: Traditional methods and a new algorithm,” *Aaai*, pp. 129 – 134, 1992.

[9] B. Xue, M. Zhang, W. N. Browne, and X. Yao, “A Survey on Evolutionary Computation Approaches to Feature Selection,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[10] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[11] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[12] W. Siedlecki and J. Sklansky, “A note on genetic algorithms for large-scale feature selection,” *Pattern Recognition Letters*, vol. 10, no. 5, pp. 335 – 347, 1989.

[13] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Feature selection using multi-objective genetic algorithms for handwritten digit recognition," *Proc. Int. Conf. on Pattern Recognition*, vol. 1, pp. 568–571, 2002.

[14] G. Pappa, A. Freitas, and C. A. Kaestner, "A Multiobjective Genetic Algorithm for Attribute Selection," *Proc. of the Fourth International Conference on Recent Advances in Soft Computing*, pp. 116–121, 2002.

[15] S. M. Vieira, M. C. Sousa, and T. A. Runkler, "Multi-Criteria Ant Feature Selection Using Fuzzy Classifiers," *Swarm Intelligence for Multi-objective Problems*, pp. 19–36, 2009.

[16] Y. Zhang, D.-w. Gong, and J. Cheng, "Multi-objective Particle Swarm Optimization Approach for Cost-based Feature Selection in Classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. PP, no. c, pp. 1–13, 2015.

[17] Y. Zhang, D.-w. Gong, X.-y. Sun, and Y.-n. Guo, "OPEN A PSO-based multi-objective multi-label feature selection method in classification," *Scientific Reports*, no. August 2016, pp. 1–12, 2017.

[18] J. Garcia-Nieto, E. Alba, L. Jourdan, and E. Talbi, "Sensitivity and specificity based multiobjective approach for feature selection: Application to cancer diagnosis," *Information Processing Letters*, vol. 109, no. 16, pp. 887–896, 2009.

[19] M. Pal and S. Bandyopadhyay, "Many-objective Feature Selection for Motor Imagery EEG Signals using Differential Evolution and Support Vector Machine," 2016.

[20] A. Khan and A. R. Baig, "Multi-Objective Feature Subset Selection using Non-dominated Sorting Genetic Algorithm," *Journal of Applied Research and Technology*, vol. 13, no. 1, pp. 145–159, 2015.

[21] M. Lichman, "UCI Machine Learning Repository," 2013.

[22] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[23] F. Jimenez, E. Marzano, G. Sanchez, G. Sciavicco, and N. Vitacolonna, "Attribute selection via multi-objective evolutionary computation applied to multi-skill contact center data classification," *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, vol. 07821, pp. 488–495, 2016.

[24] B. Huang, B. Buckley, and T. M. Kechadi, "Multi-objective feature selection by using NSGA-II for customer churn prediction in telecommunications," *Expert Systems with Applications*, vol. 37, no. 5, pp. 3638–3646, 2010.

[25] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees.* Monterey, CA: Wadsworth and Brooks, 1984, new edition [**?**]?

[26] J. Cohen, “A coefficient of agreement for nominal scales,” *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[27] M. Sorower, “A literature survey on algorithms for multi-label learning,” *Oregon State University, Corvallis*, pp. 1–25, 2010.

[28] R. Purshouse and P. Fleming, “On the evolutionary optimisation of many conflicting objectives,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 770–784, 2007.

[29] R. C. Purshouse and P. J. Fleming, “Conflict, Harmony, and Independence: Relationships in EvolutionaryMulti-Criterion Optimisation,” vol. 4403, pp. 388–402, 2007.

[30] G. Hughes, “On the mean accuracy of statistical pattern recognizers,” *IEEE Trans. Inf. Theor.*, vol. 14, no. 1, pp. 55–63, Sep. 2006.

[31] J. Tukey, *Exploratory Data Analysis*, ser. Addison-Wesley series in behavioral science.

[32] D. Brockhoff and E. Zitzler, “On Objective Conflicts and Objective Reduction in Multiple Criteria Optimization,” *Peabody Journal of Education 0161956X*, vol. 81, no. 243, pp. 180–202, 2006.

[33] M. G. Kendall, “A New Measure of Rank Correlation,” *Biometrika*, vol. 30, no. 1, pp. 81–93, 1938.

[34] P. P. Bonissone, *Springer Handbook of Computational Intelligence*, 2015.

[35] K. Deb, “Multi-Objective Optimization Using Evolutionary Algorithms,” p. 497, 2001.

[36] C. A. Coello Coello, G. B. Lamont, and D. a. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2007.

[37] J. L. Cohon and D. H. Marks, “A review and evaluation of multiobjective programming techniques,” *Water Resources Research*, vol. 11, no. 2, pp. 208–220, 1975.

[38] I. Das and J. E. Dennis, “Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems,” *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[39] J. Schaffer, “Multiple objective optimization with vector evaluated genetic algorithms,” *The 1st international Conference on Genetic Algorithms*, no. JANUARY 1985, pp. 93–100, 1985.

[40] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, “Multiobjective evolutionary algorithms: A survey of the state of the art,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[41] H. Ishibuchi, N. N. Tsukamoto, and Y. Nojima, “Evolutionary Many-Objective Optimization: A Short Review,” *Evolutionary Computation*, pp. 2419–2426, 2008.

[42] S. Chand and M. Wagner, “Evolutionary many-objective optimization: A quick-start guide,” *Surveys in Operations Research and Management Science*, vol. 20, no. 2, pp. 35–42, 2015.

[43] C. Von Lücken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," *Computational Optimization and Applications*, vol. 58, no. 3, pp. 707–756, 2014.

[44] D. H. Wolpert and W. G. Macready, "Coevolutionary Free Lunches," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[45] M. Li, S. Yang, and X. Lu, "A Comparative Study on Evolutionary Algorithms for Many-Objective Optimization," vol. 1993, no. March, 2013.

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

[47] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints," *Ieeexplore.Ieee.Org*, vol. 18, no. c, pp. 1–1, 2013.

[48] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "Performance comparison of NSGA-II and NSGA-III on various many-objective test problems," *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, pp. 3045–3052, 2016.

[49] Q. Zhang, S. Member, and H. Li, "MOEA / D : A Multiobjective Evolutionary Algorithm Based on Decomposition," vol. 11, no. 6, pp. 712–731, 2007.

[50] H. Li and Q. Zhang, "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[51] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-Inspired Coevolutionary Algorithms for Many-Objective Optimization," vol. 17, no. 4, pp. 474–494, 2013.

[52] J. Bader and E. Zitzler, "HypE: an algorithm for fast hypervolume-based many-objective optimization." *Evolutionary computation*, vol. 19, no. 1, pp. 45–76, 2011.

[53] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization," pp. 1–20, 2017.

[54] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Evolutionary Many-Objective Optimization by NSGA-II and MOEA / D with Large Populations," *Optimization*, vol. 1, no. October, pp. 1758–1763, 2009.

[55] R. C. Purshouse, C. Jalbă, and P. J. Fleming, "Preference-driven co-evolutionary algorithms show promise for many-objective optimisation," in *Proceedings of the 6th International Conference on Evolutionary Multi-criterion Optimization*, ser. EMO'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 136–150.

[56] E. Zitzler and K. Simon, "Indicator-Based Selection in Multiobjective Search," *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, vol. 3242, no. i, pp. 832–842, 2004.

[57] E. Zitzler, D. Kalyanmoy, and Thie, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[58] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, A. Abraham, L. Jain, and R. Goldberg, "Scalable test problems for evolutionary multiobjective optimization," *Evolutionary Multiobjective*, no. 1990, pp. 1–27, 2001.

[59] S. Huband, L. Barone, L. While, and P. Hingston, "A Scalable Multi-objective Test Problem Toolkit," *Lecture Notes in Computer Science 3410:280-295*, 2005.

[60] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms : A Comparative Case Study and the Strength Pareto Approach," vol. 3, no. 4, pp. 257–271, 1999.

[61] V. Barichard and J.-k. Hao, "Genetic Tabu Search for the Multi-Objective Knapsack Problem," *Tsinghua Science and Technology*, vol. 8, no. 1, pp. 8–13, 2003.

[62] A. Duarte, J. J. Pantrigo, E. G. Pardo, and N. Mladenovic, "Multi-objective variable neighborhood search: an application to combinatorial optimization problems," *Journal of Global Optimization*, vol. 63, no. 3, pp. 515–536, 2015.

[63] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," vol. 3, no. 4, pp. 257–271, 1999.

[64] J. R. J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," 1995.

[65] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

[66] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," vol. 3, no. 2, pp. 523–528, 2003.

[67] M. D. e. Constantin Zopounidis, *Multiple Criteria Decision Making: Applications in Management and Engineering*, 1st ed., ser. Multiple Criteria Decision Making. Springer International Publishing, 2017.

[68] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.

[69] C. Igel, V. Heidrich-Meisner, and T. Glasmachers, "Shark," *Journal of Machine Learning Research*, vol. 9, pp. 993–996, 2008.