Universidad Veracruzana

Artificial Intelligence Research Institute

Sampling instances for computational cost reduction in feature selection using Multi-objective Differential Evolution

Submitted by:

Jesús-Arnulfo Barradas-Palmeros

As the fulfillment of the requirement for the degree of:

Master in Artificial Intelligence

Thesis Advisor:

Dr. Efrén Mezura-Montes

Thesis Co-Advisor:

Dr. Rafael Rivera-López

June 2023

Sampling instances for computational cost reduction in feature selection using Multi-objective Differential Evolution

Universidad Veracruzana

Artificial Intelligence Research Institute

Jesús-Arnulfo Barradas-Palmeros

June 2023

Abstract

Wrapper approaches in the feature selection problem require more computational time than other methods. Given the nature of the process, each individual must create a model with the classification algorithm used to be evaluated with the fitness function. The dimensionality of the data is a crucial thing to be considered since it has a direct impact on time consumption. The research proposal stated in this document is based on using sampling methods in the feature selection process during the feature subset search.

Three base proposals are used: fixed, incremental, and evolving sampling fractions. Each is applied to the permutational-based Differential Evolution procedure for feature selection. Memory is also implemented to avoid repeated evaluations and decrease the resources required for the process. A Differential Evolution adaptation control parameter mechanism is applied to the instance reduction proposals to reduce the method's dependence on parameter tuning. Additionally, the problem is taken to the multi-objective approach using the proposed instance reduction schemes.

Experimentation is conducted to identify the top-performing proposals in the single-objective and multi-objective approaches. The results show that applying the fixed and incremental sampling fraction proposals with memory to avoid repeated evaluations successfully maintains competitive performance and reduces the computational resources required for the single-objective and multi-objective approaches.

Dedication

For my parents, Carmen and Arnulfo.

For my sister Alejandra and my niece Andrea.

For my great friends.

Acknowledgments

First, I want to thank my family for their unconditional support during my studies and life. It is an honor for me to dedicate this work to you.

I am grateful to Dr. Efrén for being an exceptional thesis advisor.

I thank Dr. Rafael for all his support and guidance for this research work and all his help during my research stay at the Instituto Tecnológico de Veracruz.

I want to express my gratitude to all my friends who have made this journey an incredibly wonderful experience.

I would like to thank Dr. Gabriel Acosta, Dr. Aldo Márquez, and Dr. Homero Rios for their reviews and suggestions that helped me to improve this document.

I want to thank the Mexican Council of Humanities, Science, and Technology (CONAHCYT) for the scholarship given to me during my master's degree studies (CVU 1142850).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The machine learning algorithms for classification struggle with the increasing dimensionality in the datasets found nowadays. Larger datasets do not necessarily imply better data. The created model's performance is affected by irrelevant features and noisy instances in the dataset [42]. Given this, data preprocessing is an essential part of the process.

One tool during the data preprocessing is the feature selection process. It consists of selecting the most relevant feature subset for the learning process and model induction. The presence of irrelevant and redundant features in the dataset makes the learning process slower and decreases the model's performance [32]. The method gives the advantage of selecting the most relevant features and reducing data dimensionality, but searching for the near-optimal subset can be computationally expensive.

Most classical feature selection methods suffer from stagnation in local optima. This way, evolutionary computation presents itself as a global search metaheuristic tool with higher efficiency in using resources [42]. Even with this, the associated cost is still elevated, making it more difficult for evolutionary computation to be used in machine learning processes. Therefore, it is relevant to reduce the processes' computational cost [46].

One of the most common feature selection processes is the wrapper approach, in which the search is guided by the performance of a model of a classification algorithm induced by each individual in the population. Since this has to be repeated several

times, the computational cost of the complete process increases. Additionally, with even larger datasets, the process keeps augmenting its demand for computational resources.

Consequently, the proposal for this research project consists of reducing the dimensionality of the data during the search process to reduce the computational cost associated with evaluating an individual. The goal of using sampling methods to reduce the number of instances in a dataset and the power of a metaheuristic such as Differential Evolution is to maintain a competitive method performance while reducing the resources that it uses. The base algorithm considered is the permutational-based Differential Evolution algorithm for feature selection $DE-FS^{PM}$ proposed in [32] and its extension to be a multi-objective procedure as proposed in [28].

1.1 Hypothesis

It is possible to reduce the computational cost associated with a wrapper scheme in a feature selection process using sampling methods for reducing the dataset instances during the search for the subset of features, maintaining a competitive method's performance.

1.2 Objectives

1.2.1 Main objective

The main goal of this research project is to evaluate the computational cost reduction and the impact on the performance of the permutational-based Differential Evolution algorithm for feature selection with the implementation of sampling methods for the dataset instances during the search for the near-optimal subset of features.

1.2.2 Secondary objectives

In the process of the research proposal, the following specific objectives are considered:

- To propose and evaluate schemes of instance reduction using sampling methods.
- To implement the instance reduction schemes during the search process of the feature selection algorithm to reduce the number of instances used while calculating

the fitness value of the population individuals.

- To evaluate the impact on the algorithm's performance regarding computational cost reduction and the selected feature subset size and classification performance metrics.
- To use an adaptive control parameter algorithm in the feature selection problem using the proposed instance reduction schemes.
- To assess the impact of using the instance reduction schemes under a multi-objective approach.

1.3 Methodology

The research proposal is stated as reducing the computational cost of the permutational-based Differential Evolution algorithm for feature selection proposed in [32]. In this way, sampling methods are implemented during the search process to reduce the number of instances taken into account in the individual's fitness evaluation. Minimizing the cardinality of the selected feature subset and maximizing the classification accuracy of the classification algorithm modeled with the reduced dataset are the objective functions in the multiobjective approach of this work.

The development of the research proposal in this document considers the following steps.

1. To review the literature in state-of-the-art works on feature selection, Differential Evolution, and computational cost reduction of wrapper approaches in the feature selection problem.
2. To implement and modify the $DE - FS^{PM}$ with the following proposals:
   (a) Given a fixed sampling fraction, apply sampling methods to reduce the number of instances from the dataset during the search process. In this case, the entire search is conducted with a fraction of the dataset. Ultimately, the best individual found is evaluated with the selected features and the complete dataset instances.
   (b) Similarly to the previous point, the process will start with a reduced number

of instances given a sampling fraction. After that, the number of instances is augmented as the number of generations increases until the last generations use the complete dataset.

(c) Include a sampling fraction in the individual codification and evolve it during the search process. In the end, it is expected to find the near-optimal feature subset and the ideal sampling fraction for the dataset.

3. Implement additional mechanisms to reduce the computational cost of the procedure. It is proposed to use memory to avoid repeated individual evaluations and incorporate this proposal with points 2a and 2b of the methodology.

4. Using the SHADE [36] Adaptive Differential Evolution scheme to calculate the algorithm's parameters while the sampling proposals previously stated are used to reduce the computational cost.

5. Implement the multi-objective approach based on the $DE - FS^{PM}$ extension for feature selection from [28]. Apply proposals from points 2 and 3 of the methodology in the mentioned procedure.

6. To evaluate the performance of the single-objective and multi-objective approaches with the proposals for computational cost reduction in classification accuracy, dimensionality reduction, and computational time.

1.4 Document structure

This chapter and the rest of the document are organized as follows:

- Chapter 1: This Chapter introduces the problem and the foundations of the research proposal.

- Chapter 2: Some related works from feature selection methods and attempts to diminish the computational cost of the process are presented. This work's contributions are stated in this chapter.

- Chapter 3: The feature selection problem is described, and some solution methods are presented. This Chapter explains the multi-objective modeling of feature selection and the basic concepts of multi-objective optimization.

- Chapter 4: The Differential Algorithm is presented and described. This Chapter includes a description of how the mutation procedure of the algorithm is adapted to the problem of Feature Selection using a search space based on permutations. The adaptive parameter control and the multi-objective version of differential evolution are also presented.

- Chapter 5: This Chapter presents the details of the proposals of the fixed, incremental, and evolving sampling fraction used to reduce the dimensionality of the data in the search procedure. Additionally, it explores how memory can be used to avoid repeated evaluations and save computational costs.

- Chapter 6: The experiments performed are described, and the results are presented.

- Chapter 7: This Chapter presents the conclusions of the proposal and states possible future work areas to continue with the research.

- Appendix A: Preliminary experimentation is presented using the fixed sampling fraction proposal.

- Appendix B: Preliminary experimentation is presented using the incremental sampling fraction proposal.

- Appendix C: In this Appendix, the results of Experiment 4 are included with all the related and calculated information.

- Appendix D: a comparison of the most successful proposed methods with an exhaustive search method is conducted. The selected features of each method and their respective performance are analyzed.

Chapter 2

Related work

With the state-of-the-art exploration, some relevant works presented that result of interest for the research proposal are reviewed and commented on in the following subsections. The first Section contains works that only focus on solving the feature selection problem. The second Section discusses some research on instance selection and computational cost reduction for the feature selection process. Finally, the proposal contributions are stated in the final Section of this chapter.

2.1 Feature selection

In [42], the different evolutionary and swarm approaches for solving the feature selection problem are classified by the number of objectives taken into account, the metaheuristic used to guide the search, and the approach used to evaluate the feature subsets. As reported in that paper, differential evolution was not one of the most popular evolutionary metaheuristics for feature selection. Other methods, such as Genetic algorithms or Particle Swarm Optimization, are presented with more related works.

A comparison of various swarm intelligence search methods for feature selection using filter and wrapper approaches is shown in [44]. The metric used for the filter approach is correlation-based, while the wrapper uses a support vector machine algorithm to guide the search. In general, the filter approaches selected a smaller subset of features, and the wrapper approaches presented higher classification performance. The different search algorithms tested showed slight differences among them.

A many-objective approach for feature selection is presented in [34], where five objectives are considered: classification accuracy, selected feature subset size, feature relevance, feature redundancy, and feature interaction. It is reported that these five objectives allowed the authors to generate a wider variety of solutions in the Pareto front and improve the discrimination capabilities of the model.

The multiobjective model from [43] considers three objectives: the feature subset size, the classification performance, and reliability. The third objective deals with missing data in the dataset after using mean imputation. The algorithm used is the non-dominated sorting genetic algorithm-III (NSGA-III).

The permutational-based differential evolution algorithm for feature selection ($DE-FS^{PM}$) presented in [32] uses a wrapper approach with the k-nearest-neighbors algorithm guiding the search. This method showed good performance when compared with classical methods such as Sequential Forward Selection and Sequential Backward Elimination, as well as with other evolutionary approaches like Particle Swarm Optimization. In this method, the codification of the individuals uses a vector with integer indices representing the features in the dataset.

In [28], an extension of the $DE - FS^{PM}$ algorithm is proposed using a multiobjective approach with the Feature Selection Generalized Differential Evolution 3 (FS-GDE3) algorithm. The permutational mechanisms are maintained, and two objectives are minimized: the classification error and the number of selected features. Some of the proposed future work implies dealing with the high computational cost associated with the procedure.

2.2 Instance selection and cost reduction

As shown in [29], instance selection methods attempt to reduce the training set in creating the model of a classifier algorithm. The main goal is to maintain classification accuracy using fewer instances. Various methods for the instance selection problem are mentioned following wrapper and filter approaches similar to the feature selection problem.

An analysis of evolutionary algorithms in feature and instance selection is seen in [39]. The results show that applying both preprocessing methods slightly drops the

model's performance, but the training-associated computational cost is reduced. The authors use the processes of feature and instance selection independently. It is reported that better results are obtained if feature selection is made before the instance selection process.

In [21], the three-objective search used in the paper got better results than the two-objective approaches in instance selection. The procedure is used in filter and wrapper processes. The three objectives used in the wrapper approach correspond to the number of selected instances, and the other two are associated with the classification error of the algorithm. One objective uses a p-fold cross-validation process in an attempt to prevent overfitting.

A simultaneous feature and instance selection process using a genetic algorithm is shown in [4]. The results show that the algorithm allowed the authors to successfully reduce the number of features and instances, maintaining at least the performance of the classification algorithm and improving it in some cases.

A variant of the feature selection problem called cost-sensitive feature selection is used when a specific cost is associated with a dataset's features. The goal is not to reduce the computational cost of the process but instead to reduce the associated cost of the selected features in the dataset. An example is in [19], where a cost evaluation criteria is proposed using a fuzzy mutual estimator with differential evolution.

An approach for computational cost reduction using surrogate models in the feature selection process is shown in [46]. In this process, auxiliary tasks called minions are assigned with a fraction of the dataset, and each one tries to serve a specific objective that will help guide the process's primary task. The proposed multitask evolutionary scheme can reduce up to 40% of the computational cost associated with the process.

Another approach for reducing the feature selection process's computational cost is using surrogate models, as seen in [20]. The authors' proposal uses Particle Swarm Optimization and a dynamic transfer function that approximates the fitness value of an individual. Results show a considerable time reduction compared to other Particle Swarm Optimization algorithms without the surrogate models.

In [38], a Variable Length Particle Swarm Optimization method is proposed to overcome the obstacle of the computational cost in high-dimensional datasets. The

procedure uses particles of shorter lengths to explore some regions of the search space. A length-changing mechanism allows the PSO to move out of local optima and focus on small and more prosperous areas. The approach can find smaller feature subsets outperforming other fixed-length PSO methods in classification performance.

An interesting and highly related work to the research proposal of this document is shown in [26]. The authors show that using random instance selection in a single-objective feature selection approach can considerably reduce the training data for a classifier algorithm with minimal impact on the accuracy performance. In this approach, the number of selected instances is fixed to 100, 250, 500, 1000, 1500, and 2000. It is proposed as future work the analysis of the impact of using a reduced number of instances in the parameter calibration process of the algorithm.

The computational cost of repeated evaluations in the differential evolution algorithm is analyzed in [22]. Even with a real-value representation, the authors identify a 20% of repeated fitness function evaluations. A hash table is proposed to avoid repeated evaluations, and a restart mechanism is implemented considering the repeated individuals in the population.

2.3 Research proposal contributions

Different approaches have been proposed to reduce the computational cost of the feature selection procedure, such as surrogate models, variable length procedures, and instance selection methods. Nonetheless, there are stated future work and open issues that are considered in this proposal with the following contributions:

- The proposal of using random sampling selection from [26] is applied in this document's three different instance reduction schemes. The fixed, incremental, and evolving sampling fraction represents novel mechanisms for implementing random instance selection during the feature subset search.
- An adaptive parameter control mechanism is implemented in the differential evolution procedure to deal with the parameter tuning when conducting the feature subset search with the instance-reduced data.
- As suggested in [22], a memory mechanism is used to avoid repeated evaluations.

- The processes of instance sampling are applied using the $DE - FS^{PM}$ algorithm [32] as a basis with the permutational-based differential evolution procedure in the single-objective approach and extended to the multi-objective approach proposed in [28].

Chapter 3

Feature selection

Access to a larger storage capacity may lead to storing more and more features for a dataset instance. The additional information kept is only sometimes helpful. It increases the model generation computational cost and could decrease the algorithm's performance [6]. An irrelevant feature value for a dataset instance consists of not being related to the instance's class. With many irrelevant features, classifying an instance can be misleading [23]. Given this, preprocessing the data gains relevancy in Machine Learning (ML), where feature selection tries to select the attributes considered the most important ones in the dataset.

Feature selection is a popular method that discards unrelated and redundant features in a dataset. Given this, the dimensionality of the data is reduced. Furthermore, the ML model uses attributes identified as the most relevant features. Therefore, it is expected to increase its classifying capabilities. This procedure is beneficial when dealing with large-scale data [1].

3.1 The feature selection problem

As seen in [4], feature selection provides advantages such as data dimensionality reduction, fastening the learning process, simplifying the generated model, and increasing the algorithm's overall performance by removing irrelevant features. There are two main approaches for feature preprocessing: feature selection and feature construction. The first one tries to find a near-optimal subset of features from the original data that

includes the most relevant and informative features to be extracted from the data and discard the ones the procedure finds irrelevant or redundant. On the other hand, feature construction builds new features after the original ones. This document's research proposal focuses on finding a near-optimal subset of features from the original data using feature selection.

The complexity of the feature selection procedure is given by the ample search space generated by the different combinations of the features from the dataset. This complexity grows exponentially as the dimensions of the dataset increase. The features are not considered individually. In a dataset, there is a complex series of interactions between features. In this way, one feature that could be discarded separately becomes relevant when interacting with other features. The number of possible solutions in the search space is given by 2^n where n is the number of features [42]. Figure 3.1 shows a basic scheme of the Feature Selection process. Given a dataset, a subset of features is extracted. Then, the subset is evaluated. If the terminal condition is not fulfilled, another subset is extracted from the data, and its goodness is assessed. Otherwise, the final subset of features found is validated.

One way to classify the feature selection problem is according to the cardinality of the feature subset as presented in [32]. The weak feature selection process needs to define a subset size initially, and all the explored subsets only contain that number of features. In this process, selecting the correct number of features for a dataset is

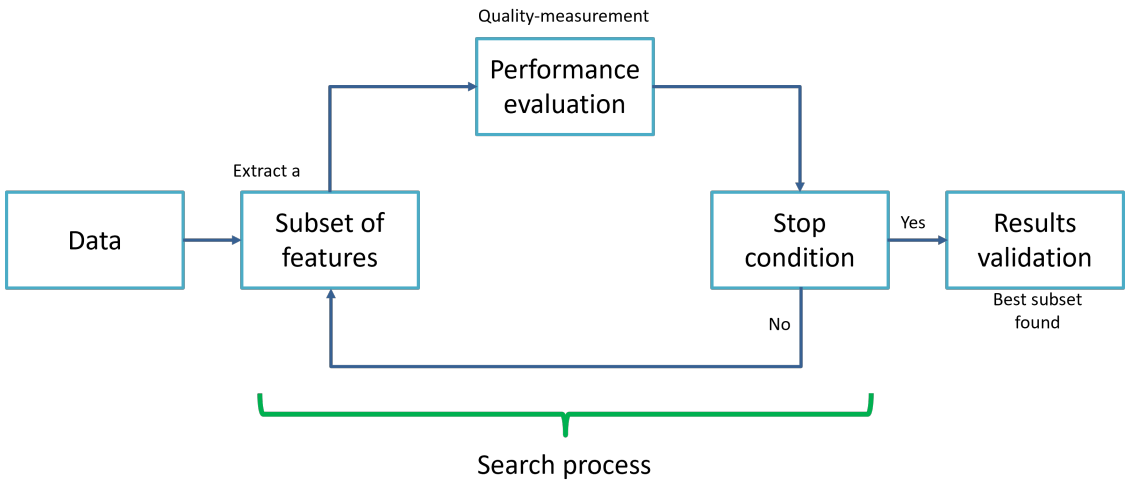

Figure 3.1: Simplified Scheme of the Feature Selection process

challenging and could lead to poor performance. In opposition, strong feature subset selection considers all possible subset lengths for the subset, giving the procedure more adaptation capabilities. Still, it increases the complexity of the search space and the search strategy used. The process followed by the proposal in this document corresponds to the strong feature selection problem.

The approaches to solving the feature selection problem can be classified according to how a subset of features' goodness is evaluated. There are three main categories of feature selection methods as shown in [14]:

- **Filter:** a metric, such as Divergence, Information Gain, Dependency, or Consistency, is calculated to assess the quality of a subset. This search process is independent of using an ML algorithm until the best possible subset is found and evaluated with a classifier.
- **Wrapper:** This approach uses an ML algorithm to calculate the performance of the subset of features. In this case, a model is trained and evaluated for each subset generated in the search process. The metrics used for this approach are accuracy-based. In the end, the near-optimal subset of features found is evaluated to determine the final performance.
- **Embeded:** In this approach, the feature selection process is performed while the classifier algorithm is modeled in its training phase. As a result, the trained model considers the most relevant features of the dataset. An example of this type of approach is shown in [25], where a decision tree method is used with its capabilities of performing feature selection while creating the model.

Some works like [42] categorized the embedded methods as part of the wrapper approach, given the interaction of the procedure with the ML algorithm. In [1], an additional classification is included as a Hybrid approach where a filter method is used to generate a solution or set of solutions given to a wrapper method to improve it. Filter methods are usually less computationally expensive than wrappers, but wrappers often result in better accuracy performance given their interaction with the ML algorithm in the search process. Embedded approaches require search methods capable of dealing with the nature of the approach. For example, in evolutionary computation, only genetic programming (GP) and learning classifier systems (LCSs) are adequate to perform embedded feature selection approaches [42]. Another difference stated in

[14] is that filter approaches are robust against overfitting to the data, while wrapper approaches are more likely to overfit. The research proposal in this document follows a feature selection process under the wrapper approach.

One of the most popular classifier algorithms for wrapper approaches corresponds to the k-nearest-neighbors (KNN) algorithm[1], whose low computational cost is highlighted. Other classifier algorithms used are Logistic Regression, Random Forest, Naive Bayes classifier, and Artificial Neural Networks [15].

A variant of the Feature Selection Problem is described in [1] as Online Feature Selection (OFS). In the traditional version problem, the complete set of data features with all the instances are known at the beginning of the process, and the method selects the most important among them. In contrast, OFS considers cases where instances and features change during the process and more information is available for the data. For this document, unless stated, the reference to the feature selection problem is for the traditional one.

Some challenges are still open for feature selection techniques. In [1], it is stated that dimensionality, class imbalance, and scalability, among others, are still open areas for improvement. The work in [14] enumerates a series of challenges for feature selection in specific research areas in Machine Learning. One challenge for wrapper approaches with metaheuristics methods is its associated high computational cost [42]. This way, more efficient evaluation processes are expected.

3.2 Feature Selection methods

A classic method proposed in [40] corresponds to the Sequential Forward Selection (SFS). Given a number d of features, it consists of selecting the one in the dataset with the highest score after evaluation. Then, add a second feature that, together with the first, has the highest score. This is repeated for adding more features until the subset of features is of size d. This process requires evaluating every possible combination of one feature and previously selected ones. Once a feature is included in the selected subset, it is not changed.

[1] In the k-nearest-neighbors algorithm, a test instance is classified accordingly to the class to which its neighbors in the training set belong. k is a parameter that determines how many neighbors are considered, and these neighbors are the closest points to the instance given a distance measure [23].

Another classical approach is the Sequential Backwards Selection (SBS) [27]. In this method, all the dataset features are first considered and evaluated together. Then, all the subsets of features that correspond to the elimination of one of the features in the dataset are evaluated. The feature that has the less significant effect on the performance of the subset is removed. Then new subsets of features with the elimination of another feature are evaluated.

Both methods, SFS and SBS, follow a greedy search where a selected feature cannot be discarded, and a discarded feature cannot be selected [42]. An alternative is shown in [31], where the methods are complemented with a procedure that can discard a previously selected feature during an SFS and select a previously discarded feature in SBS. The methods are called Sequential Forward Floating Selection (SFFS) and Sequential Backwards Floating Selection (SBFS), respectively.

As seen in [15], there has been a superiority of the metaheuristic-based approaches in the feature selection problem in recent years. The most popular ones correspond to Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Genetic Programming (GP), Ant Colony Optimization (ACO), and Differential Evolution (DE). Most of them belong to the Evolutionary Computation (EC) area. Moreover, a variety of new metaheuristics have been more recently proposed. Examples of these metaheuristics are Cuckoo Search (CS), Grey Wolf Optimization (GWO), Whale Optimization Algorithm (WOA), and Harris' Hawk Optimization (HHO), among others.

As population-based metaheuristics, the EC methods possess the advantage of evaluating a series of possible solutions (different subsets of features in the feature selection problem) in a single iteration of the process. This allows the search procedure to explore different parts of the search space at the same time. Nonetheless, that advantage has a high computational cost requirement as a downside [42].

3.3 Multi-objective feature selection

Feature selection consists of two main objectives: minimizing the cardinality of the subset of selected features and maximizing the model's performance when its accuracy is evaluated [42]. These objectives often conflict, giving space for multiobjective optimization techniques such as Evolutionary Multi-objective Optimization [4]. This

section gives a brief introduction to the basic concepts of Multi-Objective Optimization. After that, it is explored how it can be applied to the Feature Selection Problem.

3.3.1 Basics of Multi-objective Optimization

Optimizing real-world problems often deals with objectives in conflict. This is, when one is improved, the other one worsens. In this type of problem, there is not only a solution. The Pareto-optimal solutions are a set of solutions that show the trade-offs between the objectives taken into account. Since more than one objective is considered, a multidimensional space, called the objective space, is formed by the objective functions. Each solution is mapped to the objective space evaluating it with the considered objective functions [11].

Mathematically, a *multi-objective problem* is modeled as shown in [10]: with a solution x that is part of a universe Ω ($x \in \Omega$), the problem consists of minimizing the vector $F(x) = (f_1(x), ..., f_k(x))$ that contains the k objective functions values for x. x is a vector with n components $x = (x_1, ..., x_n)$. Additionally, if the problem presents constraints, they can be considered as the optimization is *subject to* $g_i(x) \leq 0$, $i = 1, ..., m$ and $h_j(x) = 0$, $j = 1, ..., p$.

A concept used in this type of problem that helps to compare different solutions is the Pareto *dominance* as explained in [11]. A solution x is said to dominate another solution x' ($x \preceq x'$) if it satisfies the following two conditions: x is equal or better in all objectives than x' and x is strictly better than x' for at least one objective. If $x \preceq x'$, it is said that x' is dominated by x ($x' \succeq x$).

A *Pareto Optimal* consists of a solution x that is not dominated by any other solution x' in Ω. More than one Pareto optimal is expected to be found and reported as a solution to the problem. The *Pareto Optimal Set* is the set formed by the Pareto Optimals found. When this set is plotted in the objective space, the non-dominated vectors are known as the *Pareto Front* [10].

As mentioned in [9], using metaheuristics for solving multi-objective problems has gained popularity since they are flexible and easy to use. Among those metaheuristics, evolutionary algorithms have become their most used strategy, creating the specific area of Evolutionary Multiobjective Optimization. Different metrics are applied to assess the performance of a multi-objective algorithm. Two Pareto-based measurements that

can deal with problems where the ideal Pareto front is unknown are the spacing [33] and the hypervolume [47] metrics.

Multiobjective problems are not limited to using only two objectives; they can include more. Problems with more than three objective functions are known as many-objective optimization problems. Handling restrictions for the different objectives is necessary to deal with practical applications. Evolutionary Computation techniques allow us to apply additional penalties to the fitness value proportionally to how much the constraint is violated [18]. Nonetheless, the feature selection problem does not have to deal with restrictions and penalties.

3.3.2 Formulation of Feature Selection as a Multi-objective problem

As mentioned previously, the problem of feature selection consists of two main objectives: to increase the performance of a classifier algorithm and to reduce the size of the subset of selected features. As stated in [42], most evolutionary algorithms are designed for continuous problems. However, feature selection has a discrete space given by the dataset features. In the case of wrapper approaches, to see the problem as a minimization one, it could be quickly transformed from an accuracy-based metric to the error of classification.

As seen previously, in multi-objective problems, more than one solution is returned, and each solution corresponds to a specific trade-off between the objectives. In the case of feature selection, a multi-objective approach is expected to return different configurations of selected features. This way, of the reported solutions, some will represent a larger subset with better performance on accuracy. In comparison, others will give a smaller subset but diminish the classifier's performance.

A common way to model the feature selection problem under a multi-objective wrapper approach is shown in [3] following Equation 3.1. Two objective functions are considered: the error in classification accuracy that the model obtains ($f_1(x)$) and the number of selected features ($f_2(x)$).

$$minimize\ F(x) = [f_1(x), f_2(x)] \tag{3.1}$$

The classification error of the model ($f_1(x)$) is calculated with values obtained from the confusion matrix where TP is True Positives, FP is False Positives, TN is True Negatives, and FN is False Negatives. Equations 3.2 and 3.3 are two ways of calculating the error rate. In Equation 3.2, the accuracy value of the model is subtracted from 1, obtaining the measure of the error. In Equation 3.3, the error rate is directly calculated from the values obtained from the confusion matrix. An ideal error measure value is zero, meaning all instances were classified correctly. In contrast, the worst value for error rate is one, meaning that all instances were given the wrong label.

$$Error = 1 - Accuracy = 1 - \frac{TP + TN}{TP + FP + TN + FN} \tag{3.2}$$

$$Error = \frac{FP + FN}{TP + FP + TN + FN} \tag{3.3}$$

In the case of the number of features selected ($f_2(x)$), the size of the selected subset of features is divided by the total number of features available in the dataset as shown in Equation 3.4. This metric will result in a value of one if all the features in the dataset are selected and zero for an empty subset.

$$f_2(x) = \frac{\#Features}{Total\#features} \tag{3.4}$$

For other feature selection approaches different than wrappers, the error metric has to be adjusted concerning the approach, given that a classifier algorithm is not used. Some open challenges of multi-objective feature selection are the scalability of data, the computational cost, and the search techniques used, among others. As mentioned earlier, most multi-objective techniques are designed for continuous search spaces, and feature selection is different than that. This way, search techniques better adapted to the problem are expected [3].

Chapter 4

Differential evolution

The Differential Evolution (DE) algorithm was proposed by [35] as a population-based global search strategy design for continuous spaces. The user-defined parameters that guide the search are the crossover rate (CR), the scale factor F, the population size (NP), and the maximum number of generations that will run the algorithm. The process of evolution consists of generating a *trial* vector u_i for every *target* vector x_i in the population and keeping the most suitable one for the next generation.

In the classic version of Differential Evolution called DE/rand/1/bin, to calculate the *trial* vector u_i, we first need to compute the *noise* vector v_i accordingly to Equation 4.1 [30]. This part of the procedure is also known as mutation. The r_0, r_1, and r_2 vectors are individuals selected randomly from the population and different from each other and x_i. F is the scale factor previously mentioned as a parameter that scales the difference of the vectors r_1 and r_2 before adding it to the r_0 vector.

$$v_i = r_0 + F(r_1 - r_2) \tag{4.1}$$

Once v_i is calculated, the *trial* vector u_i is computed using Equation 4.2. This procedure is the uniform crossover, and the *bin* name comes from the binomial distribution of the components of the resulting vector [17]. $rand_j$ is a random number between 0 and 1 generated for each vector component. J_{rand} is a position j randomly selected to make sure that the *trial* vector takes at least one component from the *noise* vector v_i.

This process is conducted component by component in the vector, and the parameter CR controls it.

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } (rand_j \leq CR) \text{ or } (j = J_{rand}); j = 1, ..., |x_i| \\ x_{i,j} & \text{otherwise} \end{cases} \tag{4.2}$$

To determine which vector will be included in the population for the next generation, a binary tournament face x_i and u_i, the one with the highest fitness value, is selected. Given this, the DE selection mechanism presents elitism, and the best solution is never lost.

DE is highlighted for its performance in different optimization problems and uses only a few parameters (F and CR) to guide the search. Still, DE has some open challenges, such as the codification and decoding strategy that can directly impact search performance and the configuration of the parameters to maintain balance in the exploration and exploitation capabilities of the algorithm [2]. Additionally, the No Free Lunch Theorem [41] states that the algorithm's performance could be really good for some types of problems and not so good for others.

Other variants of the DE algorithm are shown in [2] with alternatives to the noise vector v_i calculation. These variants include using the individual with the highest fitness value x_{best} or using more than three individuals of the population, including more than one scaled difference. Down below, a list with some variants of the mutation strategy of DE is presented.

- DE/rand/2

$$v_i = r_0 + F(r_1 - r_2) + F(r_3 - r_4) \tag{4.3}$$

- DE/best/1

$$v_i = x_{best} + F(r_0 - r_1) \tag{4.4}$$

- DE/best/2

$$v_i = x_{best} + F(r_0 - r_1) + F(r_2 - r_3) \tag{4.5}$$

- DE/current-to-best/1

$$v_i = x_i + F(x_{best} - x_i) + F(r_0 - r_1) \tag{4.6}$$

4.1 Permutational-based Differential Evolution

The Differential Evolution algorithm was initially formulated for continuous spaces, but it has also been adapted to discrete and permutational spaces [30]. In [32] and [5], a variation of the DE algorithm called Permutational-based Differential Evolution for Feature Selection ($PM - DE^{FS}$) is applied to the feature selection problem. The representation of solutions consists of integer-value vectors. Each number in the vector represents the feature in that position in the dataset. The individuals are valid permutations of a vector containing all the dataset's features plus the number zero. The function of the number zero is to serve as a division between the selected and no selected features since the values to the left of the zero are the indexes of the selected features. Figure 4.1 shows how an individual is decoded, selecting a subset of the dataset features.

In this application, the operators used in differential evolution must be adapted for the permutation search space. In [32], the mutation operator is the permutation matrix representing the difference between two permutations as indicated in Equation 4.7. This approximation is made to substitute the difference between r_1 and r_2 in the classical procedure.

$$r_1 \leftarrow \mathbf{P} r_2 \tag{4.7}$$

The scale permutation matrix uses the scale factor F to vary the number of permutations. Using the permutation matrix $\mathbf{P}$, it is scaled accordingly to the Equation 4.8 used in [32]. The algorithm for $\mathbf{P}_F$ calculation is defined in [30] as: for every row i in the matrix, if $\mathbf{P}[i,i] = 0$ and $r_i > F$, find the row of the matrix where $\mathbf{P}[j,i] \neq 0$ and swap rows i and j. r_i is a random number from a uniform distribution with boundaries 0 and 1.

$$v_i \leftarrow \mathbf{P}_F r_0 \tag{4.8}$$

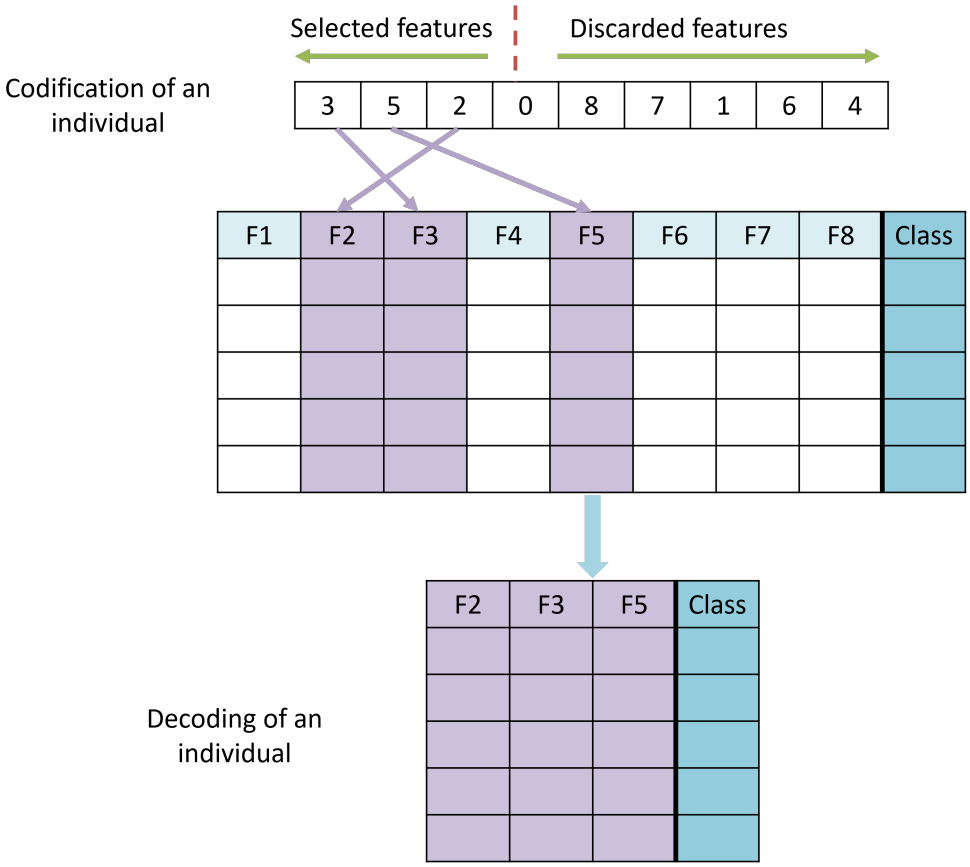

Figure 4.1: Codification and decoding of individuals in the $DE - FS^{PM}$ algorithm

The parameter F substantially affects the $\mathbf{P}_F$ calculation. For a value of F = 1, all the rows in the matrix will be changed, resulting in the diagonal matrix. The diagonal matrix will have no effect in the r_0, which would be the v_i vector, losing all the information of the r_1 and r_2 vectors. If F = 0, $\mathbf{P}$ will have no changes. In this manner, other values between 0 and 1 will perform only a fraction of the permutations represented by $\mathbf{P}$ when v_i is calculated [30].

The crossover procedure is the same as explained before for the original version of DE using Equation 4.2. Nonetheless, given that some elements are selected from v_i, and others from x_i, the resulting vector u_i could no longer be a valid permutation having repeated values. To deal with this problem, a repair mechanism is implemented in [32]: repeated values on the vector u_i are eliminated. The needed values to complete u_i as a valid permutation are taken from the v_i vector and inserted on the end of the vector u_i. The pseudo-code of the $DE - FS^{PM}$ algorithm is presented in Algorithm 1.

Algorithm 1 The $DE - FS^{PM}$ algorithm

Require: $DE - FS^{PM}$ $(CR,\ F,\ NP,\ NG)$

Input: The crossover rate (CR), the scale factor (F), the population size (NP), and the number of generations (NG).

Output: The best individual in the current population $(\mathbf{x}^{best})$.

$\mathbb{X}_0 \leftarrow \emptyset$

for each $i \in \{1, ..., NP\}$ **do**

$\mathbf{x}^i \leftarrow$ A permutation chosen at random from the solution space.

$\mathbb{X}_0 \leftarrow \mathbb{X}_0 \cup \{\mathbf{x^i}\}$

end for

for each $g \in \{1, ..., NG\}$ **do**

$\mathbb{X}_g \leftarrow \emptyset$

for each $i \in \{1, ..., NP\}$ **do**

$\mathbf{x}^i \leftarrow$ Target vector from $\mathbb{X}_{g-1}$

$\mathbf{v}^i \leftarrow$ Mutated vector using Eqs. 4.7 and 4.8.

$\mathbf{u}^i \leftarrow$ Trial vector constructed using Eq. 4.2 and the repair procedure.

$$\mathbb{X}_g \leftarrow \mathbb{X}_g \cup \begin{cases} \{\mathbf{u}^i\} & \text{if } f(\mathbf{u}^i) \text{ is better than } f(\mathbf{x}^i) \\ \{\mathbf{u}^i\} & \text{otherwise} \end{cases}$$

end for

end for

$\mathbf{X}^{best} \leftarrow$ The best individual in $\mathbf{X}_g$

return $\mathbf{X}^{best}$

4.2 Adaptive parameter control Differential Evolution

Parameter selection directly impacts the DE algorithm's performance but is highly problem dependent. As seen in [16], there are two main techniques to set the parameter values: parameter tuning and parameter control. In parameter tuning, well-performing parameter values are determined before executing an evolutionary algorithm. In contrast, in parameter control, the initial values of the parameters are changed while the search process is conducted. Three categories of parameter control are specified:

- **Deterministic Parameter Control:** A deterministic rule is applied without receiving feedback from the search. An example is a rule that changes the parameter values accordingly to the number of generations already completed.
- **Adaptive Parameter Control:** The search process provides feedback, determining parameter values.
- **Self-Adaptive Parameter Control:** The parameter values are coded inside an individual and evolve through mutation and recombination.

Different approaches for Adaptive Parameter Control have been proposed in the literature. This research proposal considers using the Success History Parameter-Adaptation for Differential Evolution (SHADE) proposed in [36] in the parameter control of the $DE-FS^{PM}$ algorithm. SHADE is based on the JADE parameter adaptation method for DE proposed in [45]. In this approach, a new mutation strategy for DE is proposed called DE/current-to-$p$best/1 (see Equation 4.9). x_{pbest} is an individual selected randomly from the top $(100 * p)\%$ of the population and $p \in (0, 1]$.

$$v_i = x_i + F_i(x_{pbest} - x_i) + F_i(r_0 - r_1) \tag{4.9}$$

The JADE procedure [45] includes an optional Archive $\mathbf{A}$ of inferior solutions. If $\mathbf{A}$ is active, the r_1 vector is chosen from the union of the individuals in the population and the ones in $\mathbf{A}$. Otherwise, r_1 is selected randomly from the population. $\mathbf{A}$ starts being empty, then *target* vectors in the population that lost the binary tournament with its *trial* vector are added to $\mathbf{A}$. When $\mathbf{A}$ exceeds the maximum size (proposed as NP), randomly chosen solutions in $\mathbf{A}$ are eliminated to maintain the desired size.

The F_i and CR_i parameters are calculated at the beginning of each generation accordingly to the values of successful sets of parameters from the previous generation. The parameters F_i and CR_i are successful when a *trial* is generated with them and beats its associated *target*. In the SHADE method [36], the previously mentioned concepts and mechanisms are kept. However, there are some changes concerning JADE [45]. The main change is the usage of a pair of memories of size H called M_{CR} and M_F that stores the mean values of the successful parameters of previous generations. At the beginning of the process, all values in M_{CR} and M_F are set to 0.5.

In each generation, a parameter CR_i and F_i is calculated for each solution x_i selecting a random position r_i from the memories M_{CR} and M_F and applying Equations 4.10 and 4.11, respectively. $randn_i(\mu, \sigma^2)$ and $randc_i(\mu, \sigma^2)$ are values obtained from a normal and Cauchy distributions using μ as the mean and σ^2 as the variance.

$$CR_i = randn_i(M_{CR,r_i}, 0.1) \tag{4.10}$$

$$F_i = randc_i(M_{F,r_i}, 0.1) \tag{4.11}$$

The successful CR_i and F_i parameters in a generation are stored in the variables S_{CR} and S_F. After that, the values in the memory in the position k are updated following Equations 4.12 and 4.13. In both cases, when there are no successful parameters, the memories maintain the values previously in the position k. k has a value of 1 at the beginning of the process. It is augmented by 1 unit every time a new element is updated in the memory. When $k > H$, k is set to 1.

$$M_{CR,k,G+1} = \begin{cases} mean_{WA}(S_{CR}) & \text{if } S_{CR} \neq \emptyset \\ M_{CR,k,G} & \text{otherwise} \end{cases} \tag{4.12}$$

$$M_{F,k,G+1} = \begin{cases} mean_{WL}(S_F) & \text{if } S_F \neq \emptyset \\ M_{F,k,G} & \text{otherwise} \end{cases} \tag{4.13}$$

In Equations 4.12 and 4.13, the update procedures use the weighted mean $mean_{WA}$

calculated with Equation 4.14, and the weighted Lehman mean ($mean_{WL}$) using Equation 4.16. Δf_j in Equation 4.15 correspond to the improvement in the fitness function of the trial individual that beat its target vector calculated as $\Delta f_j = |f(u_{j,G} - f(x_{j,G})|$.

$$mean_{WA}(S_{CR}) = \sum_{j=1}^{|S_{CR}|} w_j \cdot S_{CR,j} \tag{4.14}$$

$$w_j = \frac{\Delta f_j}{\sum_{j=1}^{|S_{CR}|} \Delta f_j} \tag{4.15}$$

$$mean_{WL}(S_F) = \frac{\sum_{j=1}^{|S_F|} w_j \cdot S_{F,j}^2}{\sum_{j=1}^{|S_F|} w_j \cdot S_{F,j}} \tag{4.16}$$

An additional parameter calculated for every individual at the beginning of the population is p_i using Equation 4.17. p_{min} is determined with a random number from $2/NP$ to 0.2. p_i consists of determining the number of individuals considered when x_{pbest} ranges from at least 2 individuals to the 20% of the population. The pseudo-code from [36] of the complete SHADE procedure is shown in Algorithm 2.

$$p_i = rand[p_{min}, 0.2] \tag{4.17}$$

More recent versions of parameter adaptation for DE based on the SHADE procedure are L-SHADE [37], iL-SHADE [7], and jSO [8]. In L-SHADE, the SHADE procedure is complemented with a Linear Population Size Reduction that changes the NP parameter along the search process. In iL-SHADE and jSO, some changes are implemented along with conditionals in updating the parameter values. As can be noticed, the SHADE procedure and its variants need the setting of the H parameter. In SHADE, H = 30 and H = 50 configurations present slightly better results, L-SHADE and iL-SHADE set H = 6, and jSO uses H = 5.

Algorithm 2 SHADE algorithm

$G \leftarrow 0$ ▷ Initialization phase
Initialize population $\mathbb{X}_0 \leftarrow (x_{1,0}, ..., x_{NP,0})$ randomly
Set all values in $M_{CR}, M_F to 0.5$
Archive $\mathbf{A} \leftarrow \emptyset$
Index Counter $k \leftarrow 1$
while Termination criteria are not met **do** ▷ Main Loop
  $M_{CR} = \emptyset, M_F = \emptyset$
  for $i = 1$ to NP **do**
    $r_i \leftarrow$ Select from $[1, H]$ randomly
    $CR_{i,g} \leftarrow randn_i(M_{CR,r_i}, 0.1)$
    $F_{i,g} \leftarrow randc_i(M_{F,r_i}, 0.1)$
    $p_{i,g} \leftarrow rand[p_{min}, 0.2]$
    Generate trial vector $u_{i,G}$ by current-to-$p$best/1/bin
  end for
  for $i = 1$ to NP **do**
    if $f(u_{i,g}) \leq f(x_{i,g}$ **then**
      $x_{i,g+1} \leftarrow u_{i,g}$
    else
      $x_{i,g+1} \leftarrow x_{i,g}$
    end if
    if $f(u_{i,g}) < f(x_{i,g}$ **then**
      $x_{i,g} \rightarrow \mathbf{A}$
      $CR_{i,g} \rightarrow S_{CR}, F_{i,g} \rightarrow S_F$
    end if
  end for
  Whenever the size of the Archive exceeds $|\mathbf{A}|$, randomly selected individuals are deleted so that $|\mathbf{A}| \leq NP$
  if $S_{CR} \neq \emptyset$ and $S_F \neq \emptyset$ **then**
    Update $M_{CR,k}, M_{F,k}$ based on S_{CR}, S_F
    $k \leftarrow k + 1$
    if $k > H$ **then**
      $k \leftarrow 1$
    end if
  end if
end while

4.3 Multi-Objective Differential Evolution

An extension of the DE algorithm that can deal with multi-objective optimization problems is the Generalized Differential Evolution 3 (GDE3) algorithm [24]. GDE3 handles problems with M objective functions and K constraints. In the case of $M = 1$ and $k = 0$, a single objective problem with no constraints, the GDE3 procedure is the same as the basic version of DE.

GDE3 deals with constraints using a variant of the concept of dominance mentioned in Section 3.3 called constraint-dominance ($\preceq_c$). A solution x_1 constraint-dominates a solution x_2 ($x_1 \preceq_c x_2$) if any of the conditions stated in [24] is true:

- x_1 is feasible and x_2 is infeasible.
- x_1 and x_2 are both infeasible and x_1 dominates x_2 in the constraint function space.
- x_1 and x_2 are feasible, and x_1 dominates x_2 in the space given by the objective functions. This case corresponds to using the classical concept of dominance in the objective space.

To deal with more than one objective, GDE3 changes its selection mechanism concerning the basic version of DE. As seen in this section, DE uses a binary tournament between *target* y *trial*, selecting the one with the better fitness function value. GDE3 implements some changes in the tournament defined by the following rules:

- If both trial and target are infeasible, the *target* is selected unless the *trial* dominates the *target* in the constraint violation space. If that happens, the *trial* is chosen.
- The feasible individual is selected when comparing a feasible vector with an infeasible one.
- For two feasible vectors, the *target* is selected if it dominates the *trial*. The *trial* is selected if it dominates the *target*. If none of the vectors dominates the other, then both are kept as part of the population for the next generation.

Given the third case, when both *target* and *trial* are selected, the population size can be increased and exceed NP. To maintain the population of size NP, two mechanisms from the NSGA-II algorithm [12] are used: the fast non-dominated sorting and the

crowding distance.

The fast non-dominated sorting algorithm from [12] is shown in Algorithm 3. This procedure finds the number of the Pareto front where each solution belongs. This means that the first Pareto front contains the non-dominated solutions of all the individuals. The second Pareto front contains the non-dominated individuals of the remaining solutions and so on. The population for the next generation in the GDE3 algorithm will contain the solutions from the first Pareto fronts until including one of the fronts would imply that the NP parameter is exceeded. In that case, an additional selection procedure is used.

When a Pareto front does not fit entirely in the population size, it is needed some way of ranking the individuals in the front. This is why Crowding Distance is used as proposed in [12]. The crowding distance value is calculated accordingly to the pseudocode in algorithm 4. All the distances are initiated as zero. Then the procedure operates for every objective function ordering the population and assigning an infinite distance to the first and the last individual to maintain the boundary values and not shrink the front. The values for the remaining individuals are calculated in an accumulative way accordingly to their neighbors. The individuals with less crowding distance are discarded at the end of the procedure, maintaining only the solutions that complete the desired population size NP.

Algorithm 3 Fast Non-Dominated Sort

```
for each p ∈ P do
    s_p ← ∅
    n_p ← 0
    for each q ∈ P do
        if p ≻ q then                          ▷ Check if p dominates q
            S_p ← S_p ∪ q                      ▷ q is added to the solutions dominated by p
        else if q ≻ p then                     ▷ Check if q dominates p
            n_p ← n_p + 1                      ▷ Increment the domination counter of p
        end if
    end for
    if n_p = 0 then                            ▷ If no solution dominates p
        p_rank ← 1                             ▷ p is part of the first front
        F_1 ← F_1 ∪ p
    end if
end for
i ← 1                                          ▷ Control the front number
while F_i ≠ ∅ do
    Q ← ∅                                      ▷ Is going to contain members of the next front
    for each p ∈ F_i do
        for each q ∈ S_p do
            n_q ← n_q − 1
            if n_q = 0 then                    ▷ q belongs to the next front
                Q_rank ← i + 1
                Q ← Q ∪ q
            end if
        end for
    end for
    i ← i + 1
    F_i ← Q
end while
```

Algorithm 4 Crowding distance

$l \leftarrow |\mathcal{I}|$ ▷ Number of solutions in front $\mathcal{I}$
for each i in $\mathcal{I}$ **do**
  $\mathcal{CD}_i \leftarrow 0$ ▷ Initializing distances
end for
for each objective m **do**
  $\mathcal{I} \leftarrow \text{sort}(\mathcal{I}, m)$ ▷ Sorting according to objective m value
  $CD_1 \leftarrow \infty, CD_l \leftarrow \infty$ ▷ First and last in the order get ∞
  for $i = 2$ to $(l-1)$ **do** ▷ Calculation for other points
    $CD_i \leftarrow CD_i + (\mathcal{I}_{i+1,m} - \mathcal{I}_{i-1,m})/(f_m^{max} - f_m^{min})$
  end for
end for

Chapter 5

Proposal details

Chapter 3 shows that wrapper approaches for feature selection are usually computationally expensive. A population-based meta-heuristic adds to the cost, given the many evaluations required where several models with different subsets of selected features must be trained and evaluated. In this research proposal, three ways to use sampling methods are presented to reduce the number of instances of the dataset used during the search to evaluate an individual. The proposals are called the fixed fraction, the incremental fraction, and the evolving fraction procedures and are implemented in the $DE - FS^{PM}$ algorithm [32].

Additionally, as seen in [22], computational resources are wasted in the DE algorithm when an individual already evaluated appears again and must be re-evaluated. A simple proposal to avoid this is to use a hash table containing previously evaluated individuals and their resulting objective values. This procedure can be implemented along the fixed and incremental fraction proposals, as shown later in this Chapter.

The main goal of this research proposal is the reduced the computational time of the feature selection procedure. Given this, the methods of selecting the dataset instances are non-complex since we want to avoid adding additional cost to the overall process. That is why just random and stratified sampling methods are used.

The rest of this Chapter is divided into five sections. The first four describe the $DE - FS^{PM}$ original procedure, and the changes applied when a sampling method is used. The final section describes the possibility of implementing the hash table as a

memory used to avoid repeated evaluations along with the sampling proposals.

5.1 $DE - FS^{PM}$earch procedure

In the original $DE-FS^{PM}$ algorithm, the search for the near-optimal subset of features occurs as shown in Figure 5.1. First, the dataset is preprocessed. Missing values are imputed for the most frequent value in the case of a categorical value and for the mean in the case of a numerical feature. Nominal values are converted to categorical values. Finally, a min-max normalization is applied for every feature in the data.

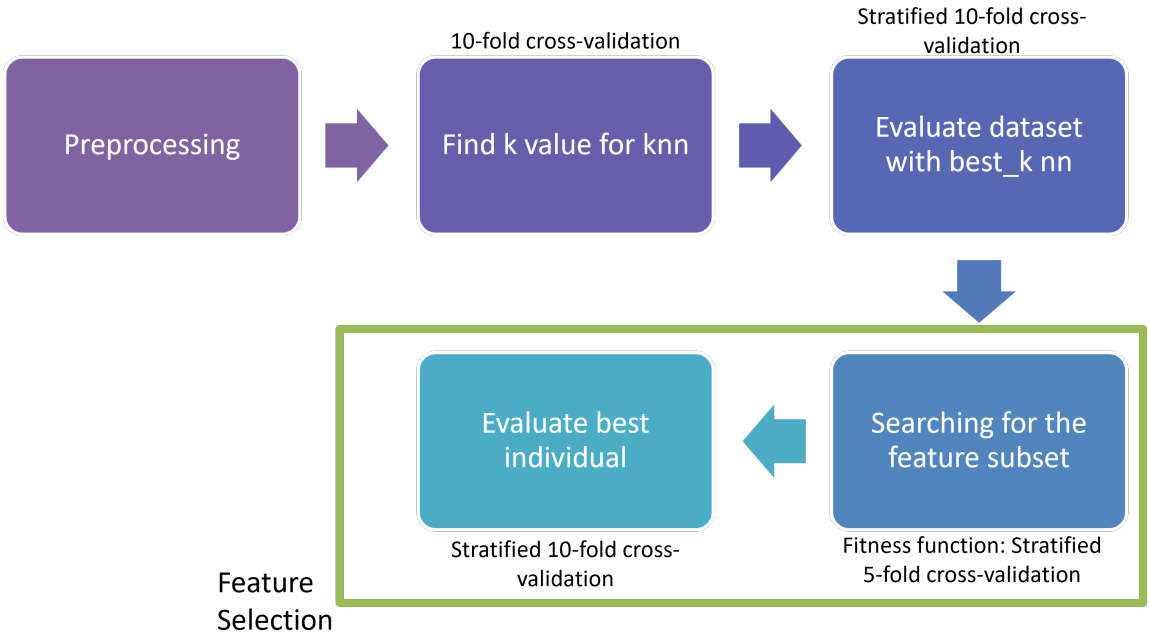

Figure 5.1: Original procedure in the $DE - FS^{PM}$ algorithm

After that, the process starts calculating the best k value for the k-nearest-neighbors classification algorithm. In this way, different values of k from 1 to 20 with a step of two are tried in the algorithm and applied to the data. The k-value corresponds to the highest classification accuracy in a 10-fold cross-validation for the KNN algorithm. Then, the dataset is evaluated with the best k-value in a stratified 10-fold cross-validation process to obtain the model's performance before the feature selection process. The green rectangle in Figure 5.1 encloses the steps of the process where feature selection occurs. As can be noticed, all the dataset instances are used along the search process.

5.2 Fixed sampling fraction

The first proposal for using a sampling method to reduce the number of instances used in the search process is to fix a sampling fraction at the beginning. A reduced dataset is extracted and used in the search process using the sampling fraction, as shown in Figure 5.2. The orange rectangle indicates the parts of the process where the dataset with fewer instances is used.

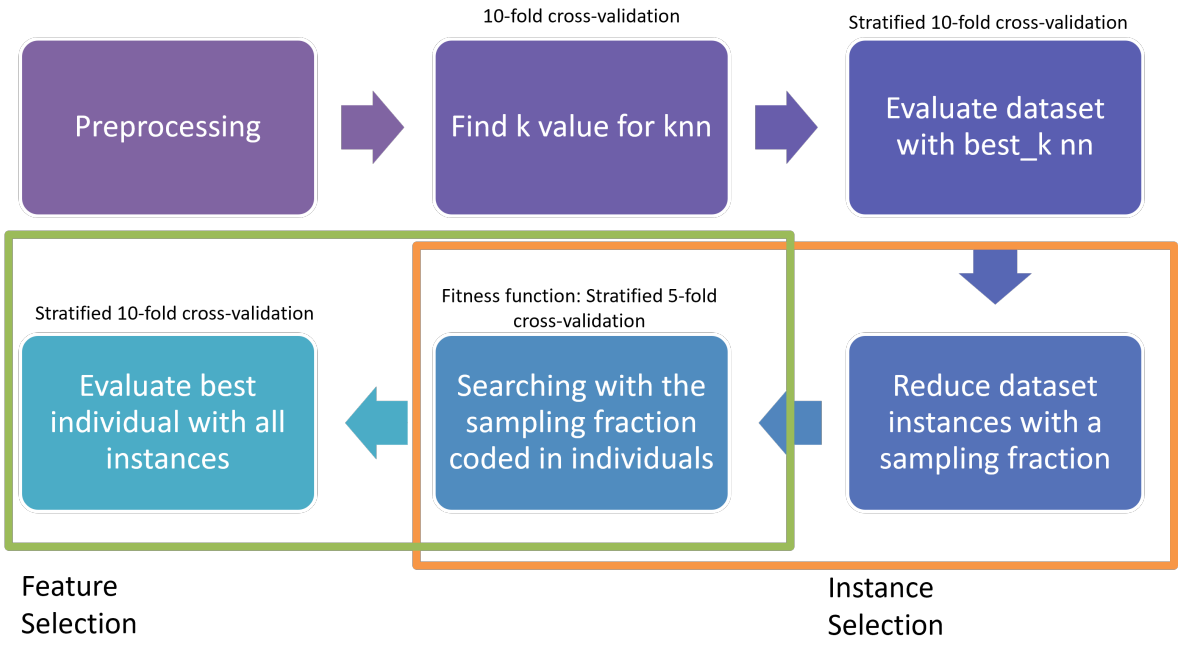

Figure 5.2: Using a fixed sampling fraction in the $DE - FS^{PM}$ algorithm

Throughout the search process, the instances of the dataset do not change. When the best subset of features is reported, a new evaluation takes place using the complete dataset and stratified-10-fold cross-validation, just as the original process. With the result obtained, it is possible to compare the performance of the classifier algorithm with and without feature selection. For this proposal, there is only one extra parameter for the algorithm: the sampling fraction.

5.3 Incremental sampling fraction

In a different approach, the second proposal uses an increasing number of instances as the number of generations of the $DE - FS^{PM}$ algorithm increases. This way, the

search process will initiate with an initial fraction determined by the user, just like the fixed sampling fraction proposal. After some generations have passed, the number of instances is increased by a fraction of the total instances. At the end of the procedure, during the last generations, the dataset used in the search will contain all the instances. Figure 5.3 shows the schematic process of the proposal. The green and orange rectangles indicate the parts of the process where feature selection takes place and the parts of the process that use the reduced number of instances.

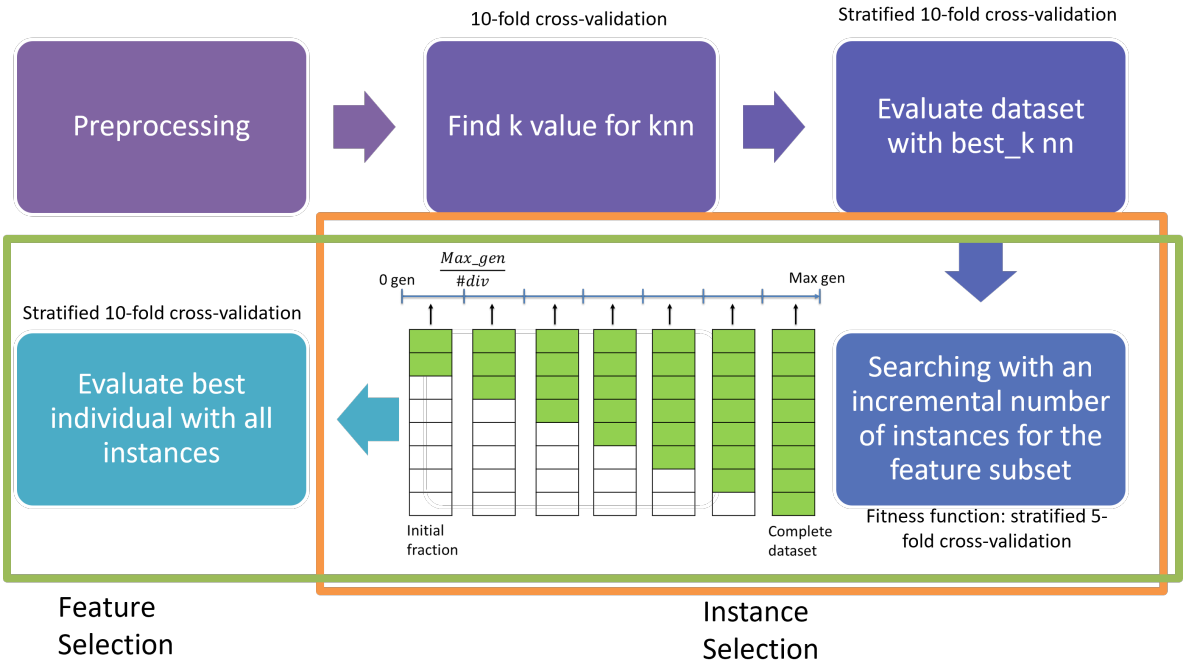

Figure 5.3: Using an incremental sampling fraction in the $DE - FS^{PM}$ algorithm

For this proposal, besides defining the initial sampling fraction in a similar way to the fixed sampling fraction feature selection, it has to be determined the number of steps or divisions of the process when more instances should be included. For example, in a process with 100 generations, if the initial fraction is 0.2 and the generations are divided into five blocks, generations 1 to 20 use 20 % of the dataset instances. In generations 21 to 40, the fraction of the dataset instances used will be 0.4. In generations 41 to 60, 0.6, and so on. Until the block from generation 81 to generation 100 uses all the dataset instances.

Using the incremental fraction proposal, promising areas of the search space are expected to be found using a less costly evaluation in the first blocks of the process

using the reduced dataset. Nonetheless, since the dataset changes in every block of the process, a process of reevaluation of the individuals in the population takes place when more instances are added to the process. That extra computational cost is expected to be mitigated by resource savings at the beginning of the search process.

Given the processes of reevaluating the individuals in the population, the convergence graph for this proposal is expected to have a particular behavior. The fitness value of the best individual will grow in every block of the process. However, when more instances are added to the process and reevaluation takes place, the fitness value of the best individual could be diminished.

5.4 Evolving sampling fraction

In the evolving sampling fraction proposal, the sampling fraction is coded inside the individual and evolves along the search process. This proposal's overall scheme is similar to the original procedure, as seen in Figure 5.4. The difference occurs during the search process, where the individuals are decoded as a subset of features of the dataset and a fraction of the data instances. The process of decoding an individual is shown in Figure 5.5.

The process of mutating and crossing the individual in the search is not changed for the permutational part of the individual. The procedure of the $DE - FS^{PM}$ algorithm is followed using the permutation matrix from Equation 4.7 and the scale permutation matrix from Equation 4.8. The real value part of the individual representing the sampling fraction is mutated and crossed separately. The classical operators from the DE algorithm shown in Equations 4.1 and 4.2 are used. A repair mechanism is proposed for the sampling fraction in the individual to maintain it between 0.1 and one. If the value exceeds one of these limits, a new value is calculated as twice the exceeded limit value minus the out-of-limits value.

With this proposal, the DE algorithm is expected to find the near-optimal subset of features for the dataset and a well-performing sampling fraction. In this manner, the user does not have to determine the value for the sampling fraction as it occurs in the fixed and incremental fraction proposals. Nonetheless, since there is no external control of the sampling fraction, the procedure could use high sampling fraction values (close to 1) and use almost all the dataset instances in the process diminishing the savings

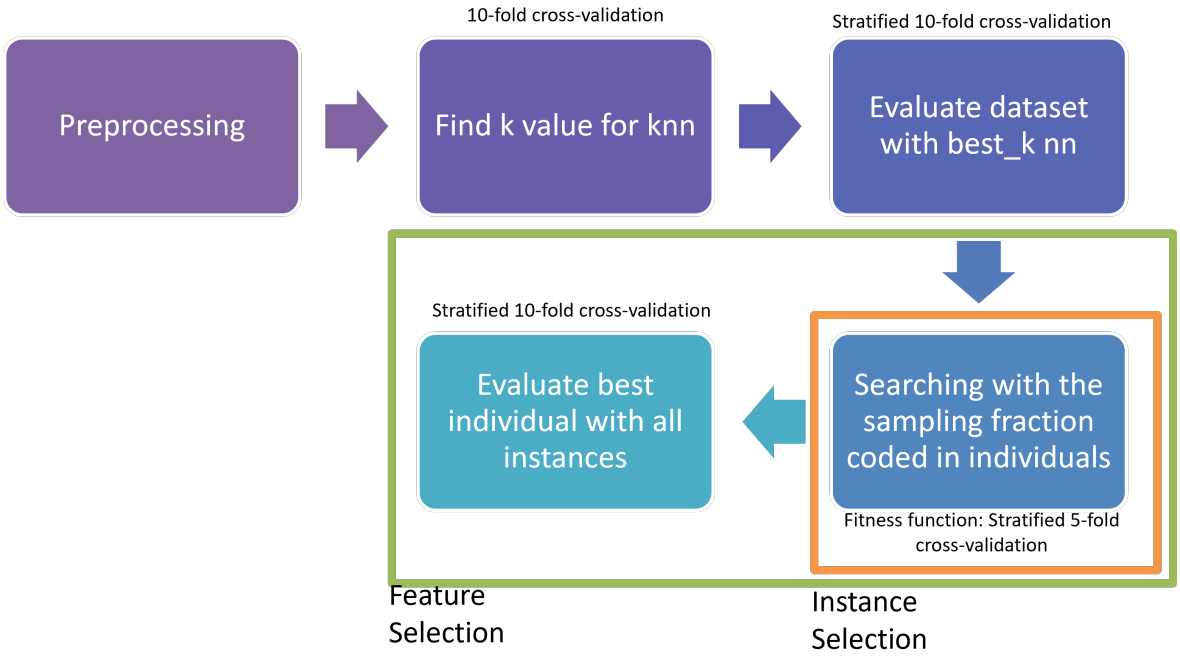

Figure 5.4: Using an evolving sampling fraction in the $DE - FS^{PM}$ algorithm

in computational time. Another possibility is that the procedure finds small sampling fraction values (close to 0) and gets a problem with severely overfitting the subset of selected features to the reduced data.

5.5 Avoiding duplicate evaluations

As mentioned in [22], the appearance of duplicated individuals in the differential evolution process is typical and expected when the algorithm converges. The problem with duplicated individuals is that they have already been evaluated, and they are evaluated every time they appear. To deal with this, simple strategies such as using a memory that contains the individual and its fitness value can avoid the usage of more resources for repeated evaluations.

In this way, when an individual is going to be evaluated, the procedure first looks in the memory if the individual has been previously evaluated. If so, the fitness value stored in the memory is returned, and a new evaluation is avoided. If the individual is not in the memory, it is evaluated and added to the memory as a new entry. In the case

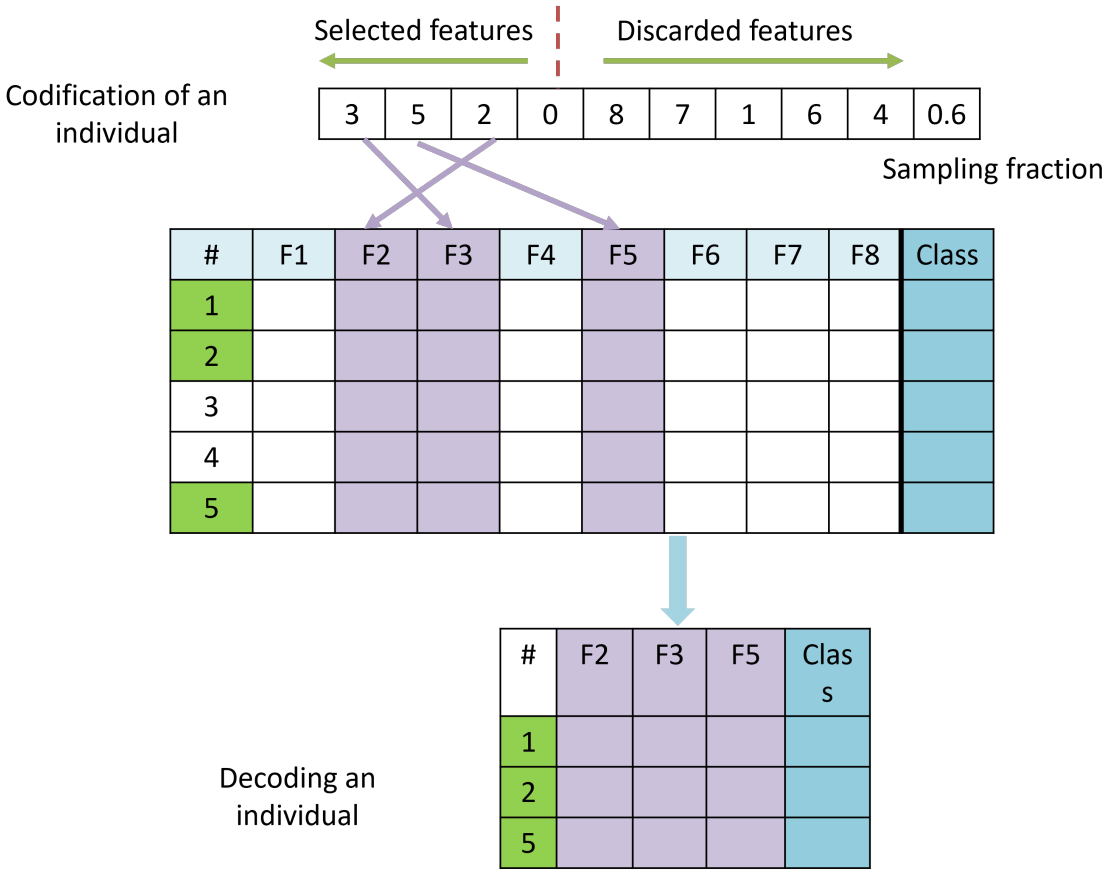

Figure 5.5: Process of decoding an individual in the evolving sampling fraction proposal

of the feature selection problem, different individuals can represent the same subset of the dataset features after being decoded. Consequently, many duplicated individuals are expected to be found during the search process as the algorithm converges.

The proposed use of memory to avoid duplicated evaluations in the $DE - FS^{PM}$ algorithm is shown in Figure 5.6. First, an individual is decoded, obtaining the selected dataset features sorted indices. Given that this is the first time that the subset of features appears, the individual is evaluated using the accuracy performance from the k-nearest-neighbors algorithm. The feature subset and its corresponding fitness value are stored in memory. After that, when the same feature subset appears in another individual decoding, the fitness value is extracted from the memory instead of being evaluated with the classifier algorithm.

The memory strategy described above is proposed to be applied to the fixed and

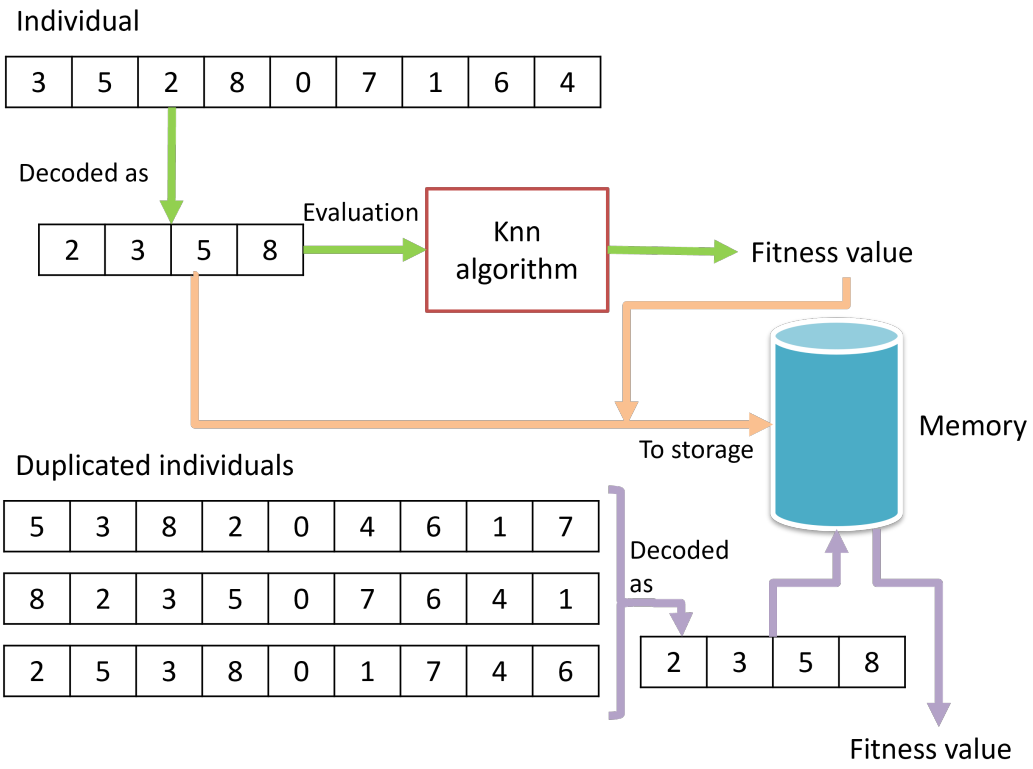

Figure 5.6: Avoiding duplicated individuals evaluation using memory

incremental fraction sampling fraction proposals. The evolving fraction procedure differs because its decoding requires sampling the dataset to a specific sample fraction. In the incremental sampling fraction procedure, the memory is reset every time new instances are added, and the process of reevaluating the population is started. For the use of the fixed sampling fraction proposal with the memory, it is proposed to divide the procedure into blocks, similar to the incremental fraction proposal. In this process, the memory is reset when the process goes into a different block. Resetting the memory is used to limit its growth.

Chapter 6

Experimentation and results

A summary of the proposals for using sampling fraction methods during the search for a near-optimal feature subset is presented in Figure 6.1. These proposals and using memory to avoid duplicated evaluations were initially stated to work with the $DE-FS^{PM}$ algorithm. Still, the proposals are extended for experimentation to be used under an adaptive parameter adaptation scheme as SHADE and the feature selection problem in a multi-objective algorithm as GDE3.

The proposals configuration with $DE - FS^{PM}$ algorithm are the following:

- Fixed sampling fraction proposal (Fixed).
- Incremental sampling fraction proposal (Incremental).
- Evolving sampling fraction proposal (Evolving).
- Fixed sampling fraction proposal with the memory to avoid duplicated evaluations (Fix DA).
- Incremental sampling fraction proposal with the memory to avoid duplicated evaluations (Inc DA).

The proposals tried with the SHADE approach [36] are the same as the ones for the original procedure but using the parameter adaptation procedure:

- SHADE Fixed sampling fraction proposal (SHADE fix).

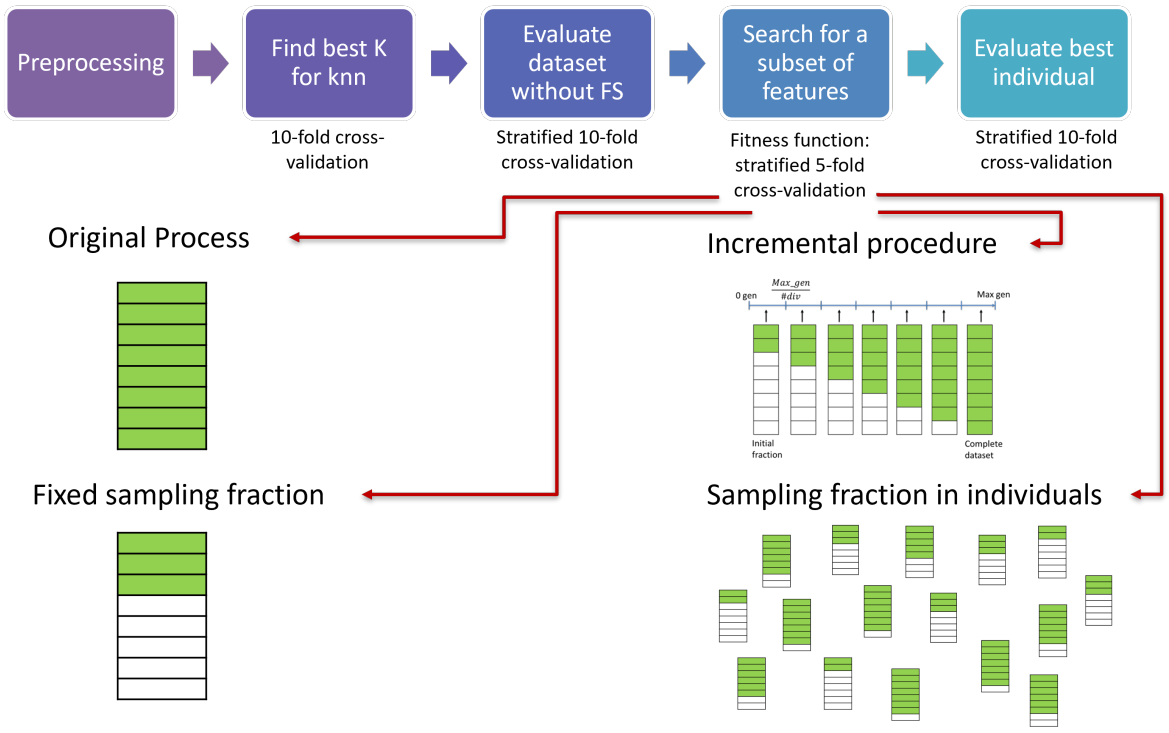

Figure 6.1: Summary of the proposals for using sampling methods during the feature selection search process

- SHADE Incremental sampling fraction proposal (SHADE inc).
- SHADE Evolving sampling fraction proposal (SHADE evo).
- SHADE Fixed sampling fraction proposal with the memory to avoid duplicated evaluations (SHA fix DA).
- SHADE Incremental sampling fraction proposal with the memory to avoid duplicated evaluations (SHA inc DA).

Finally, for the multiobjective approach to feature selection with the GDE3 algorithm [24], the proposals are:

- GDE3 fixed sampling fraction proposal with the memory to avoid duplicated evaluations (GDE3 fixed).
- GDE3 Incremental sampling fraction proposal with the memory to avoid duplicated evaluations (GDE3 incre).

The considered datasets for experimentation are shown in Table 6.1. 18 datasets from the UCI machine learning repository with different numbers of features, instances, and classes are included in the experimentation for the research proposal of this document.

Table 6.1: Description of the selected datasets for experimentation.

Base	**Inst**	**Feat**	**Classes**	**Base**	**Inst**	**Feat**	**Classes**
Arrhythmia	452	279	16	M-libras	360	90	15
Audiology	226	69	24	Musk-1	476	168	2
Australian	690	14	2	Parkinsons	195	22	2
Cylinder-b	540	39	2	Sonar	208	60	2
Crx	690	15	2	Soybean	683	35	19
Dermatology	366	34	6	Spectf	267	44	2
German-c	1000	20	2	Vehicle	846	18	4
H-valley	1212	100	2	Vote	435	16	2
ionosphere	351	34	2	wdbc	569	30	2

Preliminary experimentation includes comparing the fixed fraction proposal's random and stratified sampling methods. The results are shown in Appendix A, and it is seen that there was no apparent difference in the performance of the method when using one or another. Still, random sampling presented a more extensive time reduction than the procedure with stratified sampling. Varying the initial sampling fraction, it is seen that with smaller sampling fractions, the time reduction is more prominent, but the drop in performance is higher. Therefore, it is essential to determine an initial sampling fraction value that handles both things in a middle point.

Appendix B shows the initial experiments with the incremental fraction procedure. Similarly to the fixed sampling fraction proposal results, the initial sampling fraction value directly affects the algorithm's performance and the time reduction achieved. After the preliminary experiments, it is seen that an initial sampling fraction value of 0.6 is a promising option for future experimentation. The number of blocks dividing the procedure was set to ten.

Four main experiments were proposed to compare the different schemes of using the sampling procedures in the feature subset search. First, the proposals with and without parameter adaptation control use a set of representative datasets[1]in the single-objective approach. Given this, the most promising proposals can be selected and further explored. Then, the effects of changing the memory size in the parameter adap-

tation proposals are explored. After that, the multi-objective approach is used along the instances sampling proposals to evaluate its effectiveness in this type of optimization. Finally, an experiment is conducted using the proposals with the complete set of datasets shown in Table 6.1. More details of the experiments are described below.

- **Experiment 1:** Using the representative set of datasets, the five configurations mentioned for the $DE-FS^{PM}$ algorithm and the five of the SHADE approach are tested and compared. 10 runs for each configuration and the original procedure are computed.

- **Experiment 2:** The most interesting SHADE configuration tested is used to compare the memory size H value. Values of 6, 10, and 20 are tested to see if there is improvement in the results. The representative set of datasets is used. Ten runs of each configuration are performed.

- **Experiment 3:** The two highest performance configurations for the $DE-FS^{PM}$ algorithm in experiment 1 are selected and tested when applied to the GDE3 permutational algorithm for feature selection. The representative set of datasets is used to assess the performance changes of using a reduced number of instances during the search. Ten runs of each configuration are performed.

- **Experiment 4:** The final experiments take the highest performance configurations from experiments 1, 2, and 3, and they are applied to the remaining 13 datasets described in Table 6.1. Ten runs of each configuration are performed.

In the experimentation, the early stopping mechanism of the $DE-FS^{PM}$ algorithm applied when the best solution has not changed in 100 generations is omitted. The results of each experiment are shown in the following sections. An additional experiment was conducted to compare the $DE-FS^{PM}$ algorithm using the instance reduction schemes with conducting an exhaustive search. The results of that experiment are shown in Appendix D.

[1]Given the number of proposals to be tested, a representative set of datasets is selected, including ionosphere, sonar, SPECTF, vehicle, and hill valley. The datasets were selected given their characteristics regarding the number of features and instances.

6.1 Experiment 1: comparing single-objective proposals

In this first experiment, ten instance reduction proposals are tested against the original $DE - FS^{PM}$ algorithm as shown in Figure 6.2.

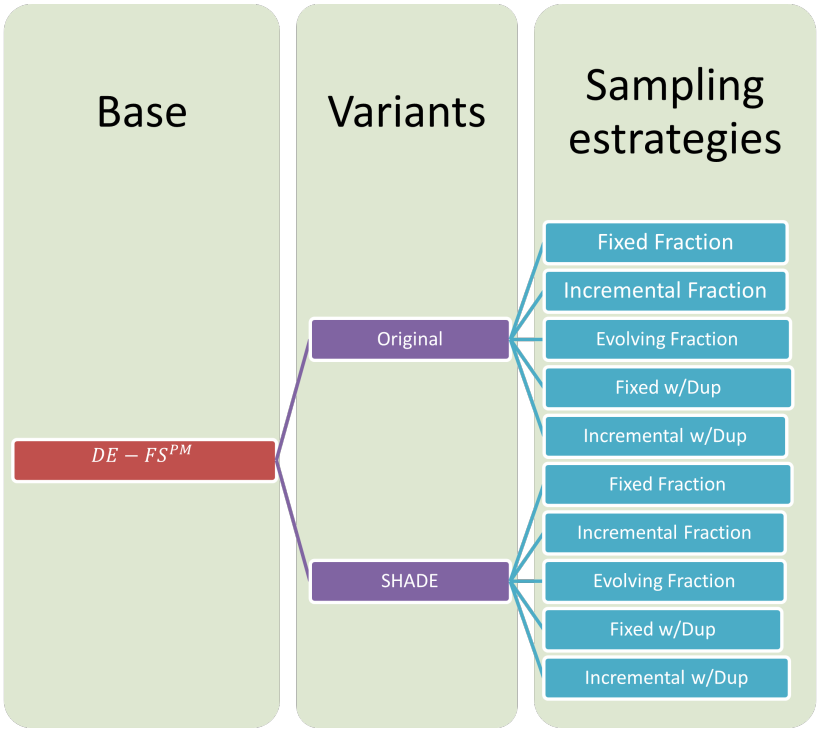

Figure 6.2: Experiment 1 proposed configuration

The parameters used for the sampling methods proposals in the base procedure are the same as the ones proposed in [32]. The parameters of the initial sampling fraction and the number of blocks are used in the fixed and incremental fraction proposals, respectively. The parameter values are the following:

- Population size (NP) $= 5 \times \#features$ bounded to at least 200 and at most 450.
- Scale factor (F) $= 0.1514$
- Crossing Rate (CR) $= 0.8552$
- Maximum number of generations $= 200$
- Initial Sampling fraction $= 0.6$

- Number of blocks = 10

The population size was divided in half for the SHADE-based proposals, and the maximum number of generations was duplicated. This is to maintain a similar number of evaluations while the procedure has more generations to find good parameters for F and CR. The SHADE parameter values are the following:

- Population size (NP) = $round(2.5 \times \#features)$ bounded to at least 100 and at most 225.
- Maximum number of generations = 400
- Initial Sampling fraction = 0.6
- Number of blocks = 10
- Memory size H = 6 as used in [37] and [7].

For the experiment, the accuracy performance of the procedure is analyzed in comparison with the original procedure. Table 6.2 shows the percentage that the proposed procedures drop the accuracy performance of the best feature subset found in each case. It is seen that the proposal with the fixed sample fraction (fixed) has poor performance, but when the memory is used in it (fixed DA), it becomes the highest-performance proposal.

Table 6.2: Experiment 1: impact on the accuracy performance using the instance reduction schemes.

Method/Dataset	Ionosphere	Sonar	SPECTF	Vehicle	Hill valley	Average	Rank
Fixed	3.30%	6.36%	2.55%	2.66%	4.04%	3.78%	7
Incremental	1.50%	1.24%	1.75%	0.87%	0.58%	1.19%	3
Evolving	5.15%	3.53%	7.64%	**0.22%**	**-0.02%**	3.30%	6
SHADE fix	2.80%	5.01%	4.26%	3.04%	4.11%	3.84%	8
SHADE inc	0.17%	2.56%	0.81%	1.10%	3.53%	1.63%	4
SHADE evo	7.23%	8.93%	7.03%	2.70%	1.94%	5.56%	10
Fixed DA	**0.15%**	**0.31%**	**-0.35%**	0.29%	0.05%	**0.09**%	1
Incre DA	1.17%	0.77%	1.20%	1.21%	0.72%	1.01%	2
SHA fix DA	3.73%	5.49%	3.46%	2.74%	4.33%	3.95%	9
SHA inc DA	0.81%	2.30%	1.47%	0.74%	3.71%	1.81%	5

In the original $DE-FS^{PM}$ procedure, all instances are used in all the evaluations. In the proposals, some instances are omitted, as well as some evaluations. The number

of instances used by each proposal is measured. The percentage of instances the procedure avoids using in the evaluation is shown in Table 6.3 using the $DE - FS^{PM}$ as the point of comparison. The results show that the fixed fraction procedure with the memory for duplicated evaluations performs on the top of the proposals.

Table 6.3: Experiment 1: Percentage of instances avoid in evaluation using the instance reduction schemes

Method/Dataset	Ionosphere	Sonar	SPECTF	Vehicle	Hill valley	Average	Rank
Fixed	39.89%	39.90%	40.07%	39.95%	40.02%	39.97%	6
Incremental	16.51%	16.62%	16.46%	16.45%	16.43%	16.49%	10
Evolving	15.01%	9.87%	**86.06%**	10.92%	2.46%	24.87%	7
SHADE fix	40.04%	40.25%	40.49%	40.10%	40.17%	40.21%	4
SHADE inc	18.50%	18.87%	18.81%	18.44%	18.42%	18.61%	9
SHADE evo	52.84%	32.17%	80.96%	25.75%	8.48%	40.04%	5
Fixed DA	79.00%	40.53%	80.51%	75.43%	45.79%	**64.25%**	1
Incre DA	**80.81%**	17.92%	78.60%	79.93%	**46.12%**	60.67%	2
SHA fix DA	50.42%	**50.44%**	57.71%	57.20%	41.33%	51.42%	3
SHA inc DA	23.54%	21.58%	23.10%	36.55%	19.12%	24.78%	8

Time reduction is shown in table 6.4. Measuring time has its associated problems, given that it is hard to guarantee that the computer is under precisely the same conditions in each run of the algorithms. One of the main reasons for this is the running of internal processes and additional use by other programs. The experiments were run in virtual environments in Google Colab and a local environment using a computer with an Intel Core i5-8250U CPU and 12 GB of RAM. A negative value represents that instead of time reduction, the time of the procedure increased. In the case of time measurement, the incremental fraction proposal with the memory for avoiding duplicated evaluations (Incre DA) is the top-performing one.

6.2 Experiment 2: varying memory size H in SHADE

For experiment 2, the most interesting proposal with the SHADE procedure is chosen to vary the memory size H. Two proposals were initially considered, the incremental sampling fraction with and without the memory for duplicated evaluations (SHADE inc and SHA inc DA). Despite having lower performance, the SHADE incremental fraction proposal with the memory for avoiding repeated evaluations (SHA inc DA) was selected because it has a more significant number of avoided instances. The values of H chosen were 10 and 20, besides the value of 6 used in the first experiment. Tables 6.5 and 6.6

Table 6.4: Time reduction in Experiment 1

Method/Dataset	Ionosphere	Sonar	SPECTF	Vehicle	Hill valley	Average	Rank
Fixed	13.44%	7.95%	9.68%	11.94%	15.31%	11.66%	4
Incremental	6.25%	-1.36%	6.54%	2.54%	0.39%	2.87%	7
Evolving	-7.00%	2.63%	4.27%	30.97%	12.87%	8.75%	6
SHADE fix	6.76%	11.24%	0.74%	33.25%	0.75%	10.55%	5
SHADE inc	12.16%	-1.52%	-8.70%	-18.44%	-8.39%	-4.98%	10
SHADE evo	-0.22%	3.63%	4.37%	2.09%	-3.96%	1.18%	8
Fixed DA	58.26%	-1.23%	67.96%	65.29%	-4.34%	37.19%	2
Incre DA	**69.09%**	23.61%	**70.88%**	**78.92%**	**28.51%**	**54.20%**	1
SHA fix DA	8.59%	**29.76%**	31.00%	50.29%	10.20%	25.97%	3
SHA inc DA	-17.66%	8.58%	-1.10%	17.17%	-9.26%	-0.46%	9

show the results obtained in accuracy performance drop and percentage of instances avoided, respectively. The datasets used correspond to the representative set of datasets used in Experiment 1.

Table 6.5: Experiment 2: impact of changing the parameter H in SHADE on the accuracy performance.

Method/Dataset	Ionosphere	Sonar	SPECTF	Vehicle	Hill valley	Average
h = 6	0.81%	**2.30%**	1.47%	**0.74%**	3.71%	1.81%
h = 10	**-0.01%**	3.61%	1.20%	1.27%	3.11%	1.84%
h = 20	0.90%	2.78%	**0.73%**	1.78%	**2.49%**	**1.74%**

Table 6.6: Experiment 2: impact of changing the parameter H on the number of instances used by the procedure.

Method/Dataset	Ionosphere	Sonar	SPECTF	Vehicle	Hill valley	Average
h = 6	23.54%	**21.58%**	23.10%	**36.55%**	**19.12%**	**24.78%**
h = 10	**24.31%**	21.53%	23.02%	35.65%	19.07%	24.72%
h = 20	23.20%	21.37%	**23.14%**	35.49%	19.05%	24.45%

The differences found for the different values of H are minimal in both measurements. Given this, the value of 6 used in literature for the SHADE variants is kept for experiment 4.

6.3 Experiment 3: multi-objective proposals

The GDE3 method for permutational-based feature selection proposed in [28] is used and implemented with the fixed (GDE3 fixed) and incremental (GDE3 incre) fraction

proposals with the memory to avoid duplicated evaluations. The parameters used are the following:

- Population size (NP) = $5 \times \#features$ bounded to at least 200 and at most 450.
- Scale factor (F) = 0.8305 obtained from [28]
- Crossing Rate (CR) = 0.7049 obtained from [28]
- Maximum number of generations = 200
- Initial Sampling fraction = 0.6
- Number of blocks = 10

The accumulated Pareto front is obtained for 10 runs of each proposal, and the best solution in accuracy (bacc), number of features (bfea), and knee solution are reported. Table 6.7 shows the accuracy performance drop of the selected solutions for the proposals GDE3 fixed and incremental GDE3 incre.

Table 6.7: Experiment 3: changes in accuracy performance using the selected solutions in the multi-objective approach.

Dataset	Method	Eval bacc	Eval bfea	Eval knee
Ionosphere	GDE3 fixed	2.16%	1.44%	**1.63%**
	GDE3 incre	**1.85%**	**-3.68%**	4.87%
Sonar	GDE3 fixed	6.65%	**0.42%**	3.41%
	GDE3 incre	**3.91%**	3.42%	**-1.68%**
SPECTF	GDE3 fixed	5.31%	**2.30%**	3.60%
	GDE3 incre	**3.53%**	3.66%	**3.20%**
Vehicle	GDE3 fixed	-3.47%	6.82%	1.25%
	GDE3 incre	**-6.60%**	**4.01%**	**-0.35%**
Hill valley	GDE3 fixed	2.75%	**-16.97%**	1.07%
	GDE3 incre	**2.40%**	3.41%	**0.96%**
Average	GDE3 fixed	2.68%	**-1.20%**	2.19%
	GDE3 Incre	**1.02%**	2.16%	**1.40%**

Table 6.8 shows the percentage of change in the performance of the multi-objective procedure when the instance reduction schemes are used. The metrics in this table are: the number of evaluations, time, avoided instances, hypervolume (HV), and spacing. The selected proposal is the GDE3 incremental procedure with the memory for avoiding repeated evaluations (GDE3 incre).

Table 6.8: Experiment 3: changes in the performance of the multi-objective algorithm with the instance reduction schemes.

Dataset	Method	Evals	Time	Instances	HV	Spacing
Ionosphere	GDE3 fixed	83.33%	82.29%	**89.98%**	**-0.46%**	**80.48%**
	GDE3 incre	**86.33%**	**83.51%**	86.33%	1.89%	77.03%
Sonar	GDE3 fixed	76.69%	74.89%	**85.99%**	**-0.43%**	**98.64%**
	GDE3 incre	**79.90%**	**74.92%**	79.90%	2.88%	96.46%
SPECTF	GDE3 fixed	**88.84%**	**86.93%**	**93.31%**	**-4.01%**	-1284.49%
	GDE3 incre	88.47%	85.07%	88.47%	3.20%	**-404.23%**
Vehicle	GDE3 fixed	85.15%	**88.91%**	**91.09%**	**-0.70%**	88.81%
	GDE3 incre	**86.38%**	88.32%	86.38%	1.72%	**95.62%**
Hill valley	GDE3 fixed	**74.25%**	77.75%	**84.55%**	6.27%	65.71%
	GDE3 incre	74.10%	**82.34%**	74.10%	**1.52%**	**96.10%**
Average	GDE3 fixed	81.65%	82.15%	**88.99%**	**0.13%**	-190.17%
	GDE3 Incre	**83.04%**	**82.83%**	83.04%	2.24%	**-7.81%**

6.4 Experiment 4: Using the best-performing proposals

All 18 selected datasets are used and compared under the best-performing proposals found in the previous experiments. For the single-objective approach, three proposals are considered the fixed sampling fraction (fixed DA), the incremental sampling fraction (Incre DA), and the SHADE incremental sampling fraction (SHA inc DA), all of them with the memory for avoiding repeated evaluations. For the multi-objective approach, the proposal used is the GDE3 incremental proposal with the memory for avoiding duplicated evaluations (GDE3 incre).

The results are shown in the following subsections, first in comparing the single-objective approaches, then for the multi-objective approach, and finally, a comparison between the single-objective and multi-objective approaches. More details of the results can be found in Appendix C.

6.4.1 Single-objective

For the single-objective comparison, Table 6.9 shows the impact on the accuracy performance of the method when the instance reduction scheme is used. In the Table, it is seen that the fixed DA proposal has the lowest drop in performance in comparison with the original procedure. Furthermore, it has the lowest standard deviation of the three methods. The second best performing method is the incre DA.

Table 6.9: Experiment 4: Accuracy performance impact of the instance reducing methods

Dataset/Method	Fixed DA	Incre DA	SHA inc DA
Arrhtythmia	**-0.18%**	0.50%	4.68%
Audiology	1.75%	**1.11%**	2.68%
australian	**-0.10%**	0.13%	0.40%
cylinder-b	**1.20%**	2.15%	1.71%
CRX	**0.46%**	0.68%	0.60%
Dermatology	0.14%	**-0.09%**	0.31%
German-c	**0.75%**	1.43%	2.39%
Hill valley	**0.05%**	0.72%	3.71%
Ionosphere	**0.15%**	1.17%	0.81%
m-libras	**0.03%**	0.09%	1.56%
Musk 1	0.49%	**0.27%**	3.17%
parkinsons	0.43%	0.88%	**0.26%**
Sonar	**0.31%**	0.77%	2.30%
Soybean	-0.12%	**-0.22%**	0.19%
SPECTF	**-0.35%**	1.20%	1.47%
Vehicle	**0.29%**	1.21%	0.74%
vote	0.38%	0.26%	**0.12%**
wdbc	**-0.18%**	0.18%	0.04%
Mean	**0.31%**	0.69%	1.51%
Std dev	**0.52%**	0.61%	1.38%

Statistical tests were conducted to compare the algorithms' accuracy performance as suggested in [13]. First, a Friedman test is computed for the original method with the other three methods where the instance reduction schemes are used. A p-value of 2.8985e-06 indicates significant differences in at least two of the medians in the comparison. Then a Nemenyi posthoc test was performed to identify which groups present significant differences. As shown in Table 6.10, the Fixed DA proposal is the only method that does not imply a significant difference in performance from the original method.

Table 6.10: Experiment 4: Nemenyi test applied to the accuracy performance of the original and the instance reduction methods

	Original	Fixed DA	Incre DA	SHA inc DA
Original	1.000000	**0.335366**	0.001041	0.001000
Fixed DA	**0.335366**	1.000000	0.164615	0.006846
Incre DA	0.001041	0.164615	1.000000	0.632975
SHA inc DA	0.001000	0.006846	0.632975	1.000000

Table 6.11 presents the percentage of instances avoided during the evaluation of individuals in the selected methods. Similarly to the impact in accuracy, the best-performing proposals are the fixed DA and the incre DA. The SHA inc DA proposal falls behind by a considerable percentage but presents a lower standard deviation. It is seen in the Table that the fixed DA represented a more significant reduction for exactly 9 datasets and the incre DA for the remaining 9 datasets.

Table 6.11: Experiment 4: instances avoided by using the instance reduction schemes

Dataset/Method	Fixed DA	Incre DA	SHA inc DA
Arrhtythmia	**44.53%**	39.73%	18.68%
Audiology	**40.65%**	17.19%	23.32%
australian	90.48%	**92.79%**	53.62%
cylinder-b	**84.85%**	84.18%	25.48%
CRX	79.85%	**86.74%**	46.60%
Dermatology	**44.94%**	26.87%	29.90%
German-c	65.56%	**71.28%**	39.45%
Hill valley	45.79%	**46.12%**	19.12%
Ionosphere	79.00%	**80.81%**	23.54%
m-libras	**40.50%**	17.35%	21.34%
Musk 1	**40.00%**	16.61%	19.11%
parkinsons	56.04%	**63.86%**	34.56%
Sonar	**40.53%**	17.92%	21.58%
Soybean	**44.30%**	25.99%	26.76%
SPECTF	**80.51%**	78.60%	23.10%
Vehicle	75.43%	**79.93%**	36.55%
vote	94.04%	**96.05%**	60.32%
wdbc	45.85%	**51.89%**	31.10%
Mean	**60.71%**	55.22%	30.78%
Std dev	20.01%	29.52%	**12.30%**

In Table 6.12, the time reduction of the methods is presented. As it was said during Experiment 1, measuring time is complicated, given the variations in processing that occur in the computer. The results show that in this measure, the top-performing proposal is the incre DA. Interestingly, unlike the accuracy and instance-avoiding performance, the fixed DA proposal is not the best and presents the highest standard deviation. The lowest performing proposal is the SHA inc DA representing an average time saving of only 5.07 %.

Table 6.12: Experiment 4: Time reduction represented by using the instance reduction schemes

Dataset/Method	Fixed DA	Incre DA	SHA inc DA
Arrhtythmia	-17.10%	**4.77%**	-36.90%
Audiology	6.50%	**19.17%**	11.05%
australian	83.39%	**90.68%**	40.31%
cylinder-b	66.23%	**74.53%**	-9.10%
CRX	37.19%	**54.20%**	-0.46%
Dermatology	1.96%	**12.94%**	-7.16%
German-c	40.09%	**68.11%**	22.28%
Hill valley	-4.34%	**28.51%**	-9.26%
Ionosphere	58.26%	**69.09%**	-17.66%
m-libras	37.76%	**39.80%**	5.82%
Musk 1	-3.69%	**3.49%**	-12.88%
parkinsons	**93.90%**	54.21%	14.62%
Sonar	-1.23%	**23.61%**	8.58%
Soybean	5.90%	**19.04%**	12.24%
SPECTF	67.96%	**70.88%**	-1.10%
Vehicle	65.29%	**78.92%**	17.17%
vote	88.92%	**93.77%**	47.59%
wdbc	6.77%	**47.67%**	6.08%
Mean	35.21%	**47.41%**	5.07%
Std dev	36.70%	29.52%	**20.16%**

6.4.2 Multi-objective

In the multi-objective comparison, only the GDE3 incre proposal is used. Table 6.13 compares the GDE3 method without the sampling proposals and the method using the aforementioned instance reduction scheme. It is seen that the accuracy performance of the method is only slightly impacted. Nevertheless, the number of evaluations, time, and instances are reduced considerably. The hypervolume metric does not present a significant change in its value. The spacing metric presents small values given that the objective space in this problem is discrete in one function. That is why the percentages shown in the Table for this metric vary from small to high values.

A Wilcoxon sign-rank test is conducted using the best solutions' accuracy values from the GDE3 method and the GDE3 incre proposals. With a p-value of 0.8650 there is no significant difference in the values from the methods.

Table 6.13: Experiment 4: Impact of the instance reducing scheme in the multi-objective approach.

Dataset	Eval bacc	Eval bfea	Eval knee	#Evals	Time	Instances	HV	Spacing
Arrhtythmia	-2.20%	0.36%	-3.11%	72.48%	55.06%	72.48%	-1.05%	98.01%
Audiology	1.20%	26.67%	-8.33%	72.68%	71.56%	72.68%	4.01%	74.27%
australian	0.17%	0.00%	0.00%	93.72%	91.62%	93.72%	0.33%	-545.58%
cylinder-b	-0.22%	-1.46%	1.55%	88.02%	85.39%	88.02%	2.15%	0.00%
CRX	-1.52%	0.00%	0.00%	92.65%	90.41%	92.65%	0.66%	-689.12%
Dermatology	-0.88%	-28.28%	-2.76%	67.03%	64.94%	67.03%	0.46%	78.58%
German-c	0.92%	0.42%	-6.10%	87.13%	87.00%	87.13%	1.44%	99.48%
Hill valley	2.40%	3.41%	0.96%	74.10%	82.34%	74.10%	1.52%	96.10%
Ionosphere	1.85%	-3.68%	4.87%	86.33%	83.51%	86.33%	1.89%	77.03%
m-libras	0.63%	-2.04%	-3.58%	59.59%	56.69%	59.59%	1.21%	88.13%
Musk 1	0.00%	-8.69%	-0.76%	46.60%	47.83%	46.60%	2.07%	98.88%
parkinsons	-0.53%	-2.62%	1.73%	84.04%	80.05%	84.04%	1.39%	98.28%
Sonar	3.91%	3.42%	-1.68%	79.90%	74.92%	79.90%	2.88%	96.46%
Soybean	0.00%	-15.90%	2.10%	73.76%	74.36%	73.76%	1.36%	94.10%
SPECTF	3.53%	3.66%	3.20%	88.47%	85.07%	88.47%	3.20%	-404.23%
Vehicle	-6.60%	4.01%	-0.35%	86.38%	88.32%	86.38%	1.72%	95.62%
vote	-1.94%	-0.01%	0.00%	95.06%	92.15%	95.06%	0.76%	100.00%
wdbc	-0.01%	-0.39%	-0.19%	81.60%	79.75%	81.60%	0.38%	41.73%
Mean	0.04%	-1.17%	-0.69%	79.42%	77.28%	79.42%	1.47%	-22.35%
Std dev	2.30%	10.26%	3.13%	12.34%	12.92%	12.34%	1.14%	240.32%

6.4.3 Single-objective vs multi-objective

A final comparison is performed between the single-objective and the multi-objective procedures. The best solutions found in the original $DE - FS^{PM}$ procedure and the fixed DA proposal are included, along with the best solution in the accuracy objective from the accumulated front in the GDE3 algorithm and the GDE3 incremental proposal. As seen in table 6.14, the single-objective approaches present superior results in the accuracy of the best solution. Nonetheless, it is essential to mention that the multiobjective approach finds various solutions instead of just one. The set of solutions found can be given to the user, and then decide which subset of features will be used.

6.5 Discussion

Four main experiments were conducted in this research proposal, with some preliminary and additional experimentation. From the initial ten proposals for the single-objective approach, it is seen that the memory to avoid repeated evaluations has a meaningful impact on the method accuracy performance besides reducing time and usage of instances. The fixed proposal is one of the worst in accuracy, but in conjunction with

Table 6.14: Experiment 4: Comparison between the single-objective and multi-objective approaches.

Dataset/method	Original best	Fixed DA best	Bacc GDE3	Bacc GDE3 inc
Arrhtythmia	76.333	**77.894**	70.560	72.111
Audiology	**87.000**	86.500	83.500	82.500
australian	87.391	**87.536**	**87.536**	87.391
cylinder-b	**85.926**	85.556	84.630	84.815
CRX	**87.826**	87.536	85.942	87.246
Dermatology	**98.911**	**98.911**	97.252	98.108
German-c	78.500	78.500	**76.000**	75.300
Hill valley	72.443	**72.934**	68.729	67.076
Ionosphere	**96.000**	95.167	93.167	91.444
m-libras	89.722	**90.556**	88.333	87.778
Musk 1	**95.780**	95.177	90.137	90.142
parkinsons	99.474	99.474	98.974	**99.500**
Sonar	**96.167**	96.143	87.048	83.643
Soybean	**95.460**	95.458	93.996	93.992
SPECTF	85.413	**86.125**	84.288	81.311
Vehicle	75.174	**75.766**	64.775	69.048
vote	96.786	**97.014**	94.493	96.327
wdbc	**98.067**	97.895	97.713	97.719

the memory, it goes to the top of the performance. The good results are visualized in Figure 6.3, where the plot compares the accuracy of the original method and the fixed DA proposal. It is seen that the impact on the performance is minimal.

The case for the incremental proposal is different. Without memory, the method has a competitive performance in accuracy but not in time and usage of instances. However, its accuracy is improved using memory. Furthermore, it becomes the method with the most significant time reduction. For the fixed and incremental proposals, using memory to avoid repeated evaluations brought more than one benefit.

The evolving sampling fraction has a problem with the procedure when the sampling fraction is evolved to values close to one or zero. Suppose the sampling fraction for most individuals converges to a value close to one. In that case, the number of avoided instances is reduced considerably, and its time reduction benefits are lost. Nonetheless, if the value is close to one, the accuracy performance is expected to be similar to the original process.

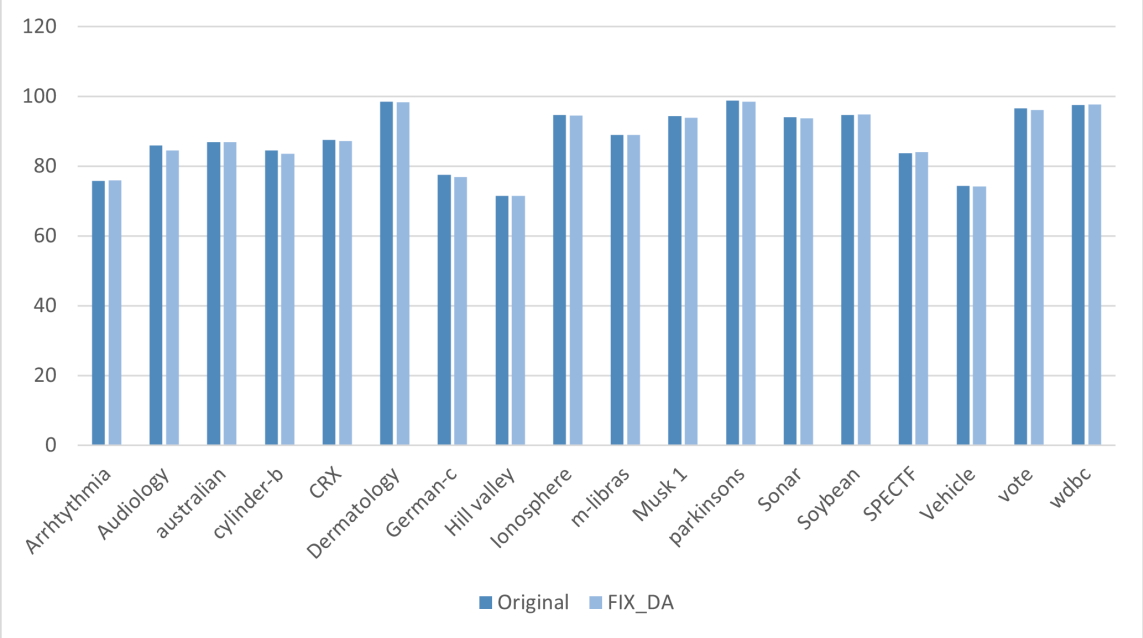

Figure 6.3: Comparison of the results obtained with the $DE - FS^{PM}$ algorithm and the fixed sampling fraction proposal with memory to avoid repeated evaluations.

Conversely, if the sampling fraction converges to a value close to zero, the procedure seems to overfit the subset of selected features to the few instances considered. When the final evaluation using all the instances is conducted, the selected subset of features does not generalize well. Given that the sampling fraction is coded as a real number in the individuals, the evolving proposal is not used along with the memory to avoid repeated evaluations. Finding a repeated individual selecting the same features and having the same sampling fraction is expected to be more challenging. However, it could be tried in future work.

The SHADE proposals were expected to perform better due to their ability to adapt the parameters to guide the search. Nevertheless, it was not the case. Additionally, the extra computation required and the additional diversity the archive gave required additional resources. More recent versions of parameter adaptation control for DE have been published; the proposals may find better results using one of them. Another aspect to be considered is the number of generations defined for the algorithm.

It is an interesting possibility that with a more extensive process, the SHADE procedure could adapt better parameters for the problem. Nonetheless, a more extended search process will require more evaluations, which goes against the objective of reduc-

ing the computational cost of the feature selection method. Further experimentation could also show if there is some type of incompatibility within this mechanism for adaptive control of the parameters and the permutational representation of individuals used in this research proposal.

The multi-objective approach for feature selection does not have the same performance in accuracy as the single-objective procedure. This was somehow expected, given that the multi-objective methods give more importance to an additional objective during the search. It is seen that the number of evaluations is considerably reduced compared with the single-objective approach. This means the procedure finds more repeated individuals, and some mechanism to improve diversity could be helpful.

Chapter 7

Conclusions and future work

7.1 Conclusions

- A variety of proposals were implemented in the single-objective approach for the feature selection problem using a fixed, incremental, and evolving sampling fraction. Additionally, memory is used to avoid repeated evaluations. Each proposal presented some advantages and disadvantages, but using memory represented a variety of benefits for the proposals.
- The proposals that avoid repeated evaluations of the process using memory significantly impact time reduction and the usage of instances. The memory helped the method avoid overfitting in the fixed sampling fraction proposal, increasing its overall performance.
- SHADE-based proposals did not find significantly better results despite their ability to adapt the search parameters. The set of parameters tuned for the original $DE - FS^{PM}$ algorithm tuned in [32] give better results for the single-objective procedure.
- The initial sampling fraction parameter used in some proposals considerably impacts the method's performance. It is seen that when the process attempts to obtain the most considerable time reduction using a small initial sampling fraction, it has the most significant drop in performance.

- In the multi-objective approach, time and instance use reduction was the most significant with the proposals that avoid repeated evaluations. A reason for this to be considered in future experiments is that the search process finds more repeated individuals.

- As seen in the results, computational time reduction for the feature selection process can be achieved using sampling methods to reduce the dimensionality of the dataset during the search process without lowering the method performance. The statistical tests find no significant differences in the results for the fixed and incremental fraction proposals with the memory for avoiding repeated evaluations in the single-objective and multi-objective approaches, respectively. This suggests that this research proposal hypothesis is accepted.

7.2 Future work

- Do further experimentation with the proposals' initial sampling fraction and the number of blocks. Especially the number of blocks that divide the search process and reset the memory.

- Further experiments with datasets that include a more significant amount of instances and features would bring valuable insights into the algorithm behavior.

- Try a different configuration of adaptive parameter control that improves the method search capabilities. An example of this is the use of one of the SHADE variations, such as the ones proposed in [37] and [7]

- Implement and experiment with a parameter adaptation algorithm for the permutational-based multi-objective proposal for the feature selection problem.

- Experiment using the proposed instance-reducing mechanisms in different single-objective and multi-objective feature selection methods. This experimentation would extend the proposals to different individuals' representations and meta-heuristics.

- Use a more elaborated method to reduce the dataset. Random sampling is used in this proposal as an initial point, but the procedures could be extended to more sophisticated methods and serve as a comparison point.

Appendix A

Initial tests: fixed sampling fraction

The $DE-FS^{PM}$ algorithm is applied with the fixed sampling fraction using 15 datasets and 0.5 as the sampling fraction. First, three runs of the original procedure are computed as a reference, and the mean is reported with the label *no sampling*. Then the stratified and random sampling methods are applied following the scheme of the fixed fraction proposal, with three runs each. Results are shown in Table A.1. Table columns contain the accuracy in the dataset without feature selection, the value obtained during the search with the reduced dataset, the final evaluation score of the best individual with all the instances, the time required for the three runs in each case, and the percentage of time reduction.

The previous experimentation used a single fraction for all cases. For this experiment, only three datasets are selected. First, three runs of the original procedure are executed again to take it as a reference. After that, the fixed fraction process for feature selection is conducted using fractions from 0.1 to 0.9 with three runs each for both stratified and random sampling. The selected datasets are ionosphere, SPECTF, and sonar. The results are shown in Figures A.1, A.2, and A.3.

The figures contain two bar plots for each dataset. The one above is for the classification performance obtained, and the second is for the percentage of time reduction using a specific sampling fraction. In the plots, the arrow's color in the figure shows the method that got a higher performance for each sampling fraction tested. In the case of classification performance, the results obtained when using no sampling are included

Table A.1: Comparing the effect of using random sampling and stratified sampling.

Name	wo FS	IS and Searching		with FS	#Features	Time (s)	% reduction
	76.50	No Sampling	87.33	85.50	25.67	9648.75	-
Audiology	76.83	Stratified	85.96	74.67	18.33	7264.68	24.71
	76.83	Random	84.67	**74.83**	27.00	5312.64	**44.94**
	86.81	No Sampling	88.50	86.67	6.67	6196.84	-
Australian	86.23	Stratified	87.93	86.09	6.00	3671.79	**40.75**
	85.80	Random	90.43	**86.52**	7.00	4014.45	35.22
	75.62	No Sampling	87.22	84.51	3.00	4928.43	-
Cylinder-b	76.60	Stratified	84.81	**79.26**	11.67	4222.53	**14.32**
	75.00	Random	83.83	78.77	12.67	4949.11	-0.42
	86.38	No Sampling	89.13	87.63	7.33	5298.96	-
Crx	86.52	Stratified	90.08	**86.52**	6.67	4558.17	13.98
	86.52	Random	90.82	85.70	7.33	4108.53	**22.47**
	97.01	No Sampling	99.27	98.63	20.33	5812.39	-
Dermatology	96.99	Stratified	99.46	**97.09**	16.33	3148.77	45.83
	97.09	Random	99.28	96.99	19.33	3145.93	**45.88**
	76.00	No Sampling	78.37	76.87	11.67	9073.43	-
German-c	75.60	Stratified	80.40	74.13	9.33	6140.53	32.32
	75.67	Random	80.40	**75.17**	8.67	5596.85	**38.32**
	86.52	No Sampling	96.11	94.87	7.00	5229.14	-
ionosphere	87.17	Stratified	97.52	**92.40**	4.67	3877.23	25.85
	87.18	Random	97.18	92.03	5.67	3794.50	**27.44**
	85.83	No Sampling	90.19	88.24	33.00	13381.43	-
M-libras	85.83	Stratified	84.63	**86.94**	23.33	13222.64	1.19
	86.94	Random	84.63	86.20	28.67	13044.09	**2.52**
	96.41	No Sampling	100.00	100.00	7.33	2193.79	-
Parkinsons	96.04	Stratified	100.00	98.30	10.33	2010.01	8.38
	96.57	Random	100.00	**99.50**	7.33	1808.50	**17.56**
	87.37	No Sampling	95.05	92.44	21.67	4481.93	-
Sonar	86.71	Stratified	96.83	84.47	18.00	3221.33	**28.13**
	87.53	Random	97.76	**85.39**	16.33	3697.68	17.50
	91.99	No Sampling	95.99	94.93	19.33	5741.14	-
Soybean	91.71	Stratified	96.01	91.66	15.33	4300.96	25.09
	91.65	Random	96.00	**93.90**	17.33	3752.98	**34.63**
	78.54	No Sampling	88.16	84.29	6.00	4652.75	-
Spectf	78.78	Stratified	89.08	78.92	8.67	3753.22	19.33
	78.64	Random	92.03	**82.40**	9.00	3732.66	**19.78**
	70.76	No Sampling	77.14	74.15	9.00	6554.56	-
Vehicle	70.77	Stratified	77.78	72.06	8.00	5494.93	16.17
	70.88	Random	76.69	**72.30**	8.33	5086.49	**22.40**
	93.65	No Sampling	97.55	96.26	5.67	4908.07	-
Vote	94.18	Stratified	97.41	94.85	6.00	3528.32	**28.11**
	93.78	Random	97.11	**95.64**	3.67	3660.42	25.42
	97.30	No Sampling	98.18	97.66	17.00	5149.78	
wdbc	97.31	Stratified	98.95	**96.72**	13.67	3230.37	**37.27**
	97.25	Random	98.83	96.37	10.00	3302.47	35.87

for reference.

The results in Figure A.1 for the ionosphere dataset show a majority of orange arrows indicating that the random sampling got better results in more cases than the

stratified sampling. One expected tendency is that with the highest sampling fraction, the performance and the measure of the accuracy will be better, but the time reduction will be at its lowest.

In Figure A.2, the results' behavior is slightly different. In the classification performance, the stratified sampling got a higher performance in one more case than the random sampling. However, analyzing the time reduction percentage, the random method was faster in all cases and got a higher time reduction. The distribution of the classes in the tested databases should be considered for future analysis. It would be interesting to see datasets where the classes are highly unbalanced.

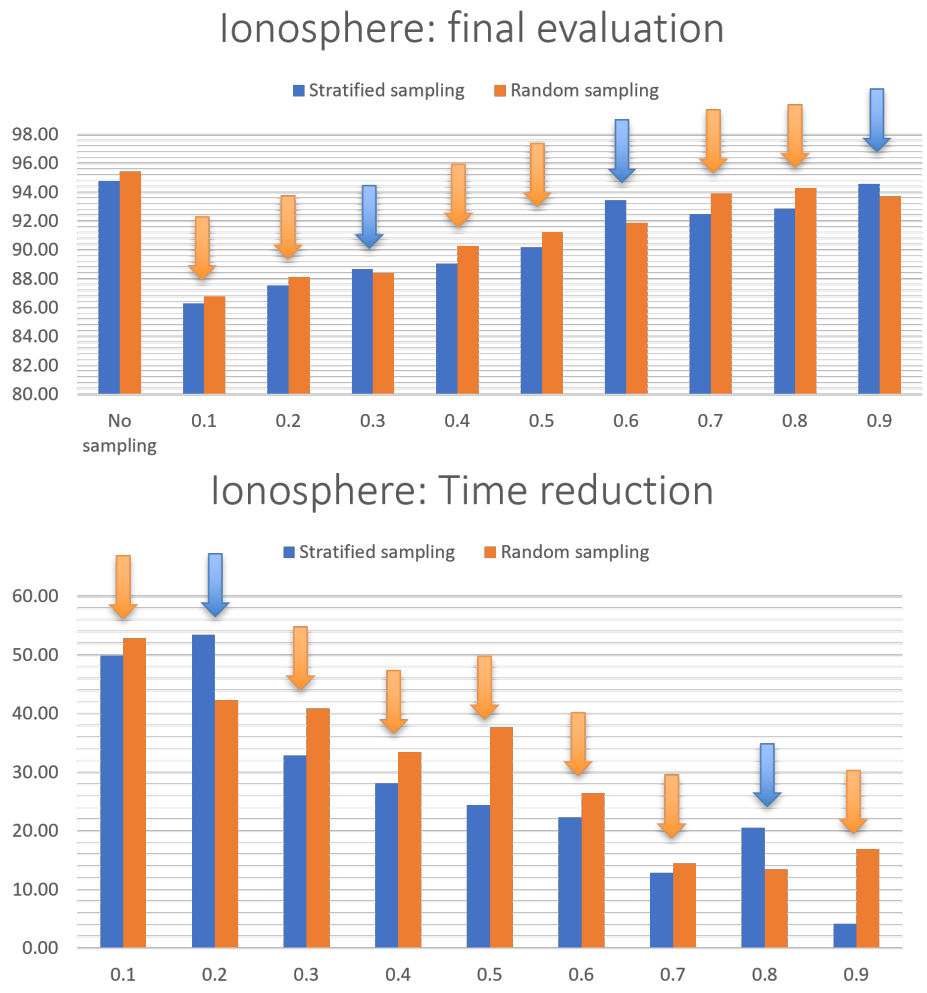

Figure A.1: Results of varying the fixed sampling fraction in the ionosphere dataset

In Figure A.3, where the experimentation results with the sonar dataset are shown, the random sampling got a higher classification performance than the stratified sampling. Nonetheless, the method is slightly better, and the differences are minor. For the time reduction, it is possible to see a different behavior than the other datasets: the smaller sampling fractions did not get the highest time reduction.

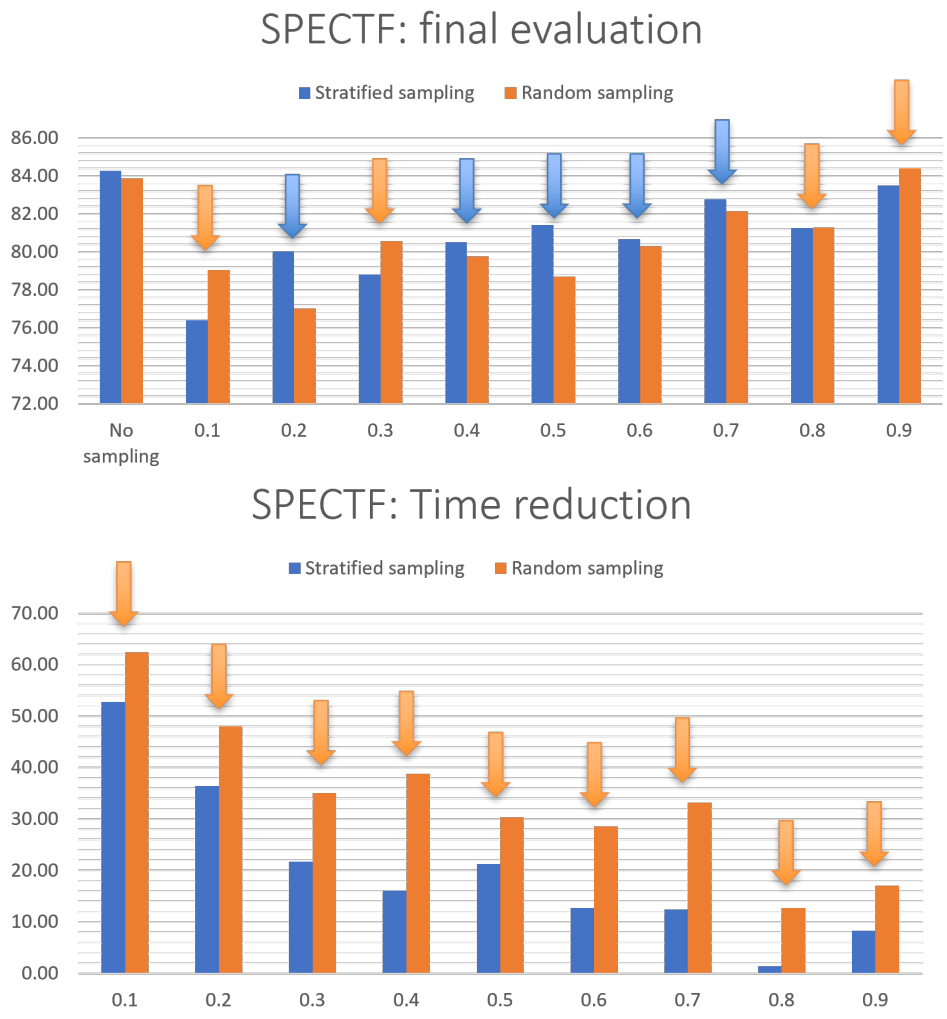

Figure A.2: Results of varying the fixed sampling fraction in the SPECTF dataset

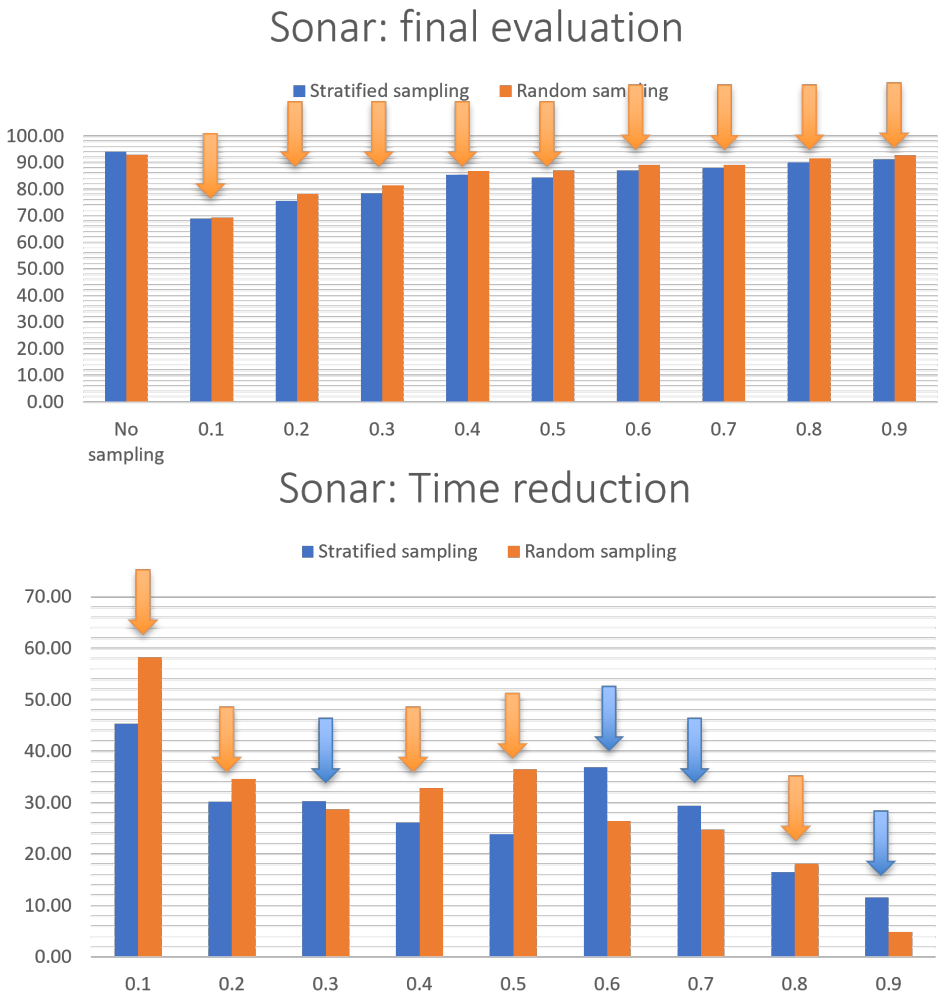

Figure A.3: Results of varying the fixed sampling fraction in the sonar dataset

Appendix B

Initial tests: Incremental Sampling fraction

A difference in this process is that since the database changes when more instances are aggregated, the population should be reevaluated. Reevaluation is an extra step with an associated cost. Given this, the convergence graph of these executions is not expected to have smooth and continuous growth. Instead, it is likely that when instances are added, the fitness value of the best individual and, in general, the mean fitness value of the population drop some percentage. However, then it recovers and even surpasses the previous values. This behavior is shown in Figure B.1.

In a similar way to the previous experimentation, to test the performance of the method of incremental instances for feature selection, three databases are selected: ionosphere, SPECTF, and sonar. The initial fraction parameter varies from 0.1 to 0.9 using ten divisions in the number of generations process. The results are shown in Tables B.1, B.2, and B.3. The columns of the tables are the following:

- In Sam Frac: Initial sampling fraction used.
- wo FS: Evaluation of the dataset without Feature Selection.
- #F selected: Number of selected features.
- wFS search: Evaluation during the search of the best individual in the last generation with Feature Selection.

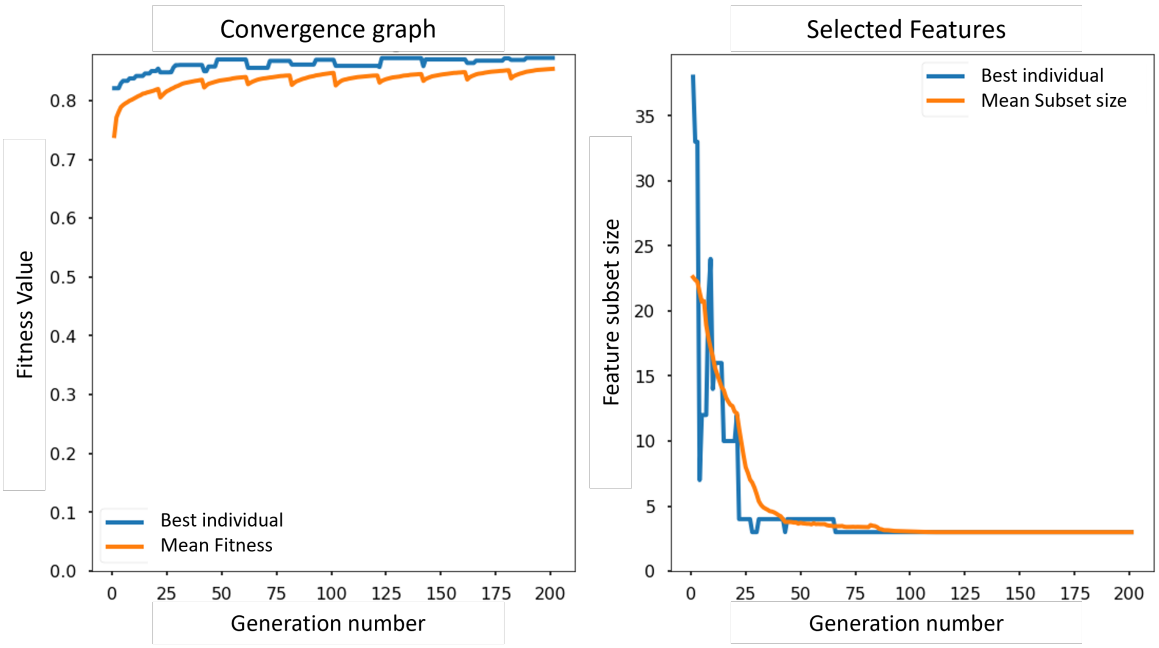

Figure B.1: Example of a convergence graph in the incremental number of instances process for feature selection.

- wFS Eval: Final evaluation of the solution found with Feature Selection and all the instances.
- Time: Time that was taken during the process.
- % reduction: Percentage of time reduced when using the incremental procedure during the search.

As mentioned before, the changes in the composition of the dataset during different periods of the search come with an extra cost. In this case, as seen in the Table B.1, the ionosphere dataset obtains very competitive results when an initial fraction from 0.4 and above is used. Nonetheless, from those experiments, the result came with an associated time reduction in only two cases.

In the case of the SPECTF dataset, some competitive results are obtained with an initial fraction of 0.6 with a time reduction of 8.84%. As shown in Table B.2, the only initial fraction that required more time than the original procedure was the 0.9 fraction, the highest tested.

Finally, as seen in Table B.3, the sonar dataset got some outstanding results, out-

Table B.1: Varying initial fraction in the incremental Feature Selection in the Ionosphere dataset.

In Sam Frac	wo FS	#F selected	wFS search	wFS Eval	Time	% reduction
No sampling	86.79	6.67	96.39	94.49	4159.58	
0.1	86.71	3.00	90.41	87.36	3355.70	19.33
0.2	86.13	7.67	94.11	92.03	3535.35	15.01
0.3	87.37	5.00	94.98	92.87	3638.14	12.54
0.4	86.61	7.00	95.83	93.74	3725.98	10.42
0.5	87.08	6.33	95.73	94.87	3973.82	4.47
0.6	86.99	7.00	95.36	94.21	4223.64	-1.54
0.7	86.99	6.00	95.54	93.82	4258.17	-2.37
0.8	86.04	6.33	95.73	94.20	4549.54	-9.38
0.9	86.72	6.33	96.40	94.87	4628.72	-11.28

Table B.2: Varying initial fraction in the incremental Feature Selection in the SPECTF dataset.

In Sam Frac	wo FS	#F selected	wFS search	wFS Eval	Time	% reduction
No sampling	77.53	7.33	87.91	84.67	4766.57	
0.1	79.54	4.67	86.52	82.67	3721.41	21.93
0.2	77.38	4.67	87.01	83.05	3742.12	21.49
0.3	77.04	6.67	85.90	82.05	4037.99	15.29
0.4	77.26	6.33	86.38	83.62	4301.12	9.76
0.5	77.69	5.33	86.15	82.39	4370.91	8.30
0.6	76.69	6.33	87.39	84.43	4344.98	8.84
0.7	77.62	4.00	85.91	82.17	4242.71	10.99
0.8	76.42	3.00	86.13	83.41	4501.43	5.56
0.9	77.67	3.33	86.89	83.69	5004.77	-5.00

performing the method that does not use sampling in the initial fractions of 0.6, 0.7, and 0.8. Additionally, the fractions 0.1, 0.3, 0.5, and 0.9 got close to the method without sampling. The procedure required less time than the original method in all cases.

Table B.3: Varying initial fraction in the incremental Feature Selection in the Sonar dataset.

In Sam Frac	wo FS	#F selected	wFS search	wFS Eval	Time	% reduction
No sampling	86.40	21.67	95.67	92.66	6660.00	
0.1	86.40	19.67	94.41	92.66	5446.58	18.22
0.2	87.10	17.67	94.08	91.65	5388.53	19.09
0.3	86.20	18.67	94.22	92.02	5499.71	17.42
0.4	86.40	13.00	94.55	90.88	5426.14	18.53
0.5	85.76	16.00	94.08	92.32	5571.17	16.35
0.6	86.97	19.00	94.56	92.93	5678.66	14.73
0.7	87.37	18.00	94.41	92.95	5953.88	10.60
0.8	86.71	23.00	94.89	93.25	6321.15	5.09
0.9	86.75	21.67	95.19	92.31	6546.04	1.71

Appendix C

Results from Experiment 4

This Appendix shows the final results of each method in Experiment 4. First, the methods from the single-objective approach and then the multi-objective approach.

C.1 Single-objective

The single-objective methods considered are:

- The original $DE - FS^{PM}$ procedure (Table C.1).
- Fixed DA: The fixed sampling fraction with the memory to avoid repeated evaluations (Table C.2).
- Incre DA: The incremental sampling fraction with the memory to avoid repeated evaluations (Table C.3).
- SHA inc DA: The SHADE incremental sampling fraction with the memory to avoid repeated evaluations (Table C.4).

The tables contain the average from ten runs of the procedure. The results are reported using the columns:

- Best k nn: best value found of k for the nearest neighbor classification algorithm. A 10-fold cross-validation process is used.
- wo FS: Evaluation of the dataset without Feature Selection. A stratified 10-fold

cross-validation process is used.

- #Fselected: Number of selected features.
- wFS search: Evaluation during the search process for the last generation's best individual with Feature Selection. A stratified 5-fold cross-validation process is used.
- wFS woIS Eval: Final evaluation of the solution found with feature selection and without instance selection. A stratified 10-fold cross-validation process is used.
- Time: Amount of time that was taken during the process.
- # Evals: Number of evaluations required during the search process.
- Instances: Number of instances used during the evaluations in the search process.

Table C.1: Results of the $DE - FS^{PM}$ procedure in Experiment 4.

Dataset	Best k nn	woFS	#Fselected	wFS search	wFS woIS Eval	#Evals	Time	Instances
Arrhtythmia	4.6	58.935	11.9	76.997	75.712	90450.0	6379.35	40883400.0
Audiology	1.0	76.900	26.3	87.400	85.950	71355.0	4695.31	14271000.0
australian	18.2	86.174	6.8	88.304	86.812	40200.0	1448.77	27738000.0
cylinder-b	1.0	74.685	3.6	87.352	84.556	40200.0	1980.15	21708000.0
CRX	10.6	86.087	9.4	88.855	87.522	40200.0	2035.05	27738000.0
Dermatology	5.0	96.970	19.2	99.183	98.443	40200.0	1827.60	14713200.0
German-c	9.4	74.990	10.2	78.960	77.430	40200.0	2902.20	40200000.0
Hill valley	1.0	64.803	9.6	72.508	71.510	90450.0	7062.38	109625400.0
Ionosphere	1.2	86.640	6.3	96.355	94.640	40200.0	1581.06	14110200.0
m-libras	1.0	85.861	24.4	90.472	88.944	90450.0	5762.20	32562000.0
Musk 1	1.0	85.747	34.8	95.293	94.291	90450.0	6708.84	43054200.0
parkinsons	1.0	96.024	10.1	100.000	98.826	40200.0	1567.62	7839000.0
Sonar	1.0	86.757	23.4	95.387	94.002	61305.0	2902.87	12751440.0
Soybean	1.0	91.732	18.1	95.698	94.658	40200.0	2348.92	27456600.0
SPECTF	14.8	78.547	4.2	87.535	83.672	45225.0	1983.83	12075075.0
Vehicle	4.8	70.229	10.1	77.023	74.313	40200.0	4346.58	34009200.0
vote	7.0	93.329	5.3	97.379	96.506	40200.0	1094.46	17487000.0
wdbc	8.8	96.890	16.9	98.298	97.522	40200.0	2445.40	22873800.0

Table C.2: Results of the fixed sampling fraction with memory to avoid duplicated evaluations procedure in Experiment 4.

Dataset	Best k nn	woFS	#Fselected	wFS search	wFS woIS Eval	#Evals	Time	Instances
Arrhtythmia	3.6	58.900	12.8	77.172	75.846	83676.0	7470.37	22676196.0
Audiology	1.0	76.450	23.5	87.500	84.450	70579.9	4390.28	8469588.0
australian	18.2	86.203	6.9	87.957	86.899	6377.8	240.65	2640409.2
cylinder-b	1.8	74.352	6.1	85.019	83.537	10148.6	668.67	3288146.4
CRX	12.2	86.072	10.1	88.391	87.116	13498.6	703.82	5588420.4
Dermatology	3.2	96.989	24.0	99.264	98.303	36826.1	1791.87	8101742.0
German-c	11.8	75.110	11.5	78.100	76.850	23075.1	1738.73	13845060.0
Hill valley	1.0	64.858	9.4	72.312	71.475	81749.9	7368.80	59432177.3
Ionosphere	1.0	86.925	6.9	95.473	94.498	14042.9	659.87	2963051.9
m-libras	1.0	86.278	21.8	90.389	88.917	89701.7	3586.57	19375567.2
Musk 1	1.0	85.097	38.3	95.106	93.828	90323.1	6956.56	25832406.6
parkinsons	1.0	95.913	11.0	100.000	98.405	29450.8	1128.61	3445743.6
Sonar	1.2	86.429	21.3	95.439	93.712	60670.9	2938.53	7583862.5
Soybean	1.0	91.553	15.4	96.106	94.771	37300.1	2210.35	15293041.0
SPECTF	14.6	77.863	6.3	86.514	83.967	14705.3	635.66	2352848.0
Vehicle	7.0	70.294	9.7	76.137	74.099	16447.8	1508.88	8355482.4
vote	8.8	93.293	5.1	96.874	96.140	3990.2	121.26	1041442.2
wdbc	8.8	97.186	17.9	98.315	97.698	36321.2	2279.76	12385529.2

Table C.3: Results of the incremental sampling fraction with memory to avoid duplicated evaluations procedure in Experiment 4.

Dataset	Best k nn	woFS	#Fselected	wFS search	wFS woIS Eval	#Evals	Time	Instances
Arrhtythmia	5.0	58.918	9.9	76.247	75.334	71161.3	6075.07	24640145.1
Audiology	1.0	76.550	21.0	86.750	85.000	73769.5	3795.17	11817836.1
australian	17.0	85.681	5.2	86.652	86.696	4284.5	135.03	1998753.3
cylinder-b	1.8	74.537	3.8	81.981	82.741	9319.9	504.29	3433983.6
CRX	12.2	86.362	8.6	87.362	86.928	7460.4	331.34	3676758.2
Dermatology	4.0	96.911	13.7	98.966	98.529	36987.9	1591.18	10760366.1
German-c	12.2	75.270	7.6	76.900	76.320	15870.2	925.55	11543944.9
Hill valley	1.0	64.597	7.5	71.031	70.998	64457.3	5049.16	59063794.0
Ionosphere	1.2	86.810	5.7	93.309	93.535	11228.4	488.76	2707904.7
m-libras	1.0	85.944	17.8	90.417	88.861	93406.1	3468.85	26912247.2
Musk 1	1.0	85.420	34.4	94.832	94.032	94332.7	6474.71	35901145.9
parkinsons	1.0	96.163	8.6	99.487	97.961	19363.5	717.84	2832749.3
Sonar	1.0	86.836	19.9	94.678	93.281	63029.7	2217.55	10466961.2
Soybean	1.0	91.448	15.8	95.739	94.863	37436.7	1901.78	20320042.7
SPECTF	13.6	77.303	5.6	82.550	82.668	13970.5	577.75	2584166.9
Vehicle	4.8	70.364	9.8	74.385	73.415	11364.0	916.32	6826543.4
vote	6.0	93.454	4.0	96.230	96.251	2472.7	68.24	691102.1
wdbc	10.0	97.100	11.1	97.875	97.347	25412.8	1279.65	11004588.0

Table C.4: Results of the SHADE incremental sampling fraction with memory to avoid duplicated evaluations procedure in Experiment 4.

Dataset	Best k nn	woFS	#Fselected	wFS search	wFS woIS Eval	#Evals	Time	Instances
Arrhtythmia	5.2	59.023	15.8	72.526	72.169	91974.9	8733.64	33246802.1
Audiology	1.0	76.900	34.3	84.800	83.650	68329.3	4176.55	10943560.1
australian	18.0	85.913	8.9	87.681	86.464	23242.2	864.79	12866157.8
cylinder-b	1.6	74.833	7.0	83.944	83.111	37464.0	2160.32	16177548.0
CRX	10.2	86.319	9.7	88.043	87.000	26841.9	1340.04	14811713.1
Dermatology	3.6	96.907	23.2	98.772	98.143	35263.2	1958.49	10314473.5
German-c	11.0	75.430	14.5	77.040	75.580	30575.8	2255.71	24339110.5
Hill valley	1.0	65.068	20.2	69.604	68.855	91442.5	7716.53	88664822.3
Ionosphere	1.0	87.125	6.7	94.645	93.871	38536.3	1860.25	10787964.2
m-libras	1.0	85.861	42.7	88.972	87.556	88970.3	5426.78	25612956.0
Musk 1	1.0	85.226	53.8	91.578	91.298	91516.8	7573.21	34826542.0
parkinsons	1.0	96.092	12.1	99.333	98.574	32901.9	1338.36	5130101.4
Sonar	1.0	86.800	24.6	92.654	91.838	60198.8	2653.75	9999588.3
Soybean	1.0	91.743	17.5	95.241	94.480	36756.7	2061.43	20110117.3
SPECTF	14.8	77.987	10.5	84.952	82.446	43440.6	2005.70	9285872.3
Vehicle	6.4	69.776	10.8	75.155	73.761	31920.7	3600.48	21579628.0
vote	8.0	93.563	4.8	96.736	96.390	20267.7	573.57	6939545.2
wdbc	8.4	97.013	21.5	98.068	97.486	34671.9	2296.64	15761046.9

C.2 Multi-objective

The multi-objective methods considered are:

- GDE3 procedure (Table C.5).
- GDE3 incre: The GDE3 incremental sampling fraction with the memory to avoid repeated evaluations (Table C.6).

The table contains the values obtained from the accumulated Pareto Front after ten procedure runs. The results are reported using the columns:

- #Fsel bacc: Number of selected features from the best solution in the accuracy objective.
- Eval bacc: Evaluation of the best solution in the accuracy objective. A stratified 10-fold cross-validation process is used.
- #Fsel bfea: Number of selected features from the best solution in the number of features objective.
- Eval bacc: Evaluation of the best solution in the number of features objective. A stratified 10-fold cross-validation process is used.
- #Fsel knee: Number of selected features from the solution in the knee point of the Pareto front.
- Eval bacc: Evaluation of the solution in the knee point of the Pareto front. A stratified 10-fold cross-validation process is used.
- # Evals: Number of evaluations required during the search process.
- Time: Amount of time that was taken during the process.
- Instances: Number of instances used during the evaluations in the search process.
- HV: hypervolume metric result.
- Spacing: spacing metric result.

Table C.5: Results of the GDE3 procedure in Experiment 4.

Dataset	#Fsel bacc	Eval bacc	#Fsel bfea	Eval bfea	#Fsel knee	Eval knee	#Evals	Time	Instances	HV	Spacing
Arrhtythmia	6	70.560	1	60.396	6	70.145	904500	55304.60	408834000	0.7200	0.00042
Audiology	38	83.500	1	37.500	10	72.000	713550	39211.37	142710000	0.8020	0.00134
australian	5	87.536	1	85.507	1	85.507	402000	13071.64	277380000	0.8131	0.00001
cylinder-b	2	84.630	1	75.926	2	83.889	402000	20141.28	217080000	0.8426	0.00000
CRX	5	85.942	1	85.507	1	85.507	402000	17919.14	277380000	0.8230	0.00001
Dermatology	23	97.252	1	38.544	4	87.740	402000	16851.95	147132000	0.9237	0.00242
German-c	19	76.000	1	70.900	2	68.900	402000	28819.87	402000000	0.7345	0.01279
Hill valley	7	68.729	1	55.858	7	69.308	904500	76439.35	1096254000	0.7004	0.00044
Ionosphere	8	93.167	1	78.056	2	88.325	402000	18321.41	141102000	0.9147	0.00084
m-libras	19	88.333	1	27.222	7	85.278	904500	45819.50	325620000	0.8725	0.00177
Musk 1	23	90.137	1	58.369	10	86.746	904500	61872.14	430542000	0.9177	0.00057
parkinsons	12	98.974	1	80.447	2	92.842	402000	15356.53	78390000	0.9422	0.00593
Sonar	43	87.048	1	68.190	4	85.095	613050	26213.50	127514400	0.9043	0.00851
Soybean	25	93.996	1	30.467	6	91.068	402000	25596.32	274566000	0.8804	0.00287
SPECTF	3	84.288	1	80.527	3	83.134	452250	19809.66	120750750	0.8406	0.00004
Vehicle	13	64.775	1	50.599	2	65.015	402000	43173.63	340092000	0.6896	0.00247
vote	5	94.493	1	95.634	1	95.634	402000	10313.68	174870000	0.9091	0.00000
wdbc	29	97.713	1	90.320	2	95.611	402000	19625.51	228738000	0.9460	0.00051

Table C.6: Results of the GDE3 incremental sampling fraction with memory to avoid duplicated evaluations procedure in Experiment 4.

Dataset	#Fsel bacc	Eval bacc	#Fsel bfea	Eval bfea	#Fsel knee	Eval knee	#Evals	Time	Instances	HV	Spacing
Arrhtythmia	10	72.111	1	60.179	10	72.329	248922	24854.03	112512744	0.7275	0.00001
Audiology	51	82.500	1	27.500	16	78.000	194971	11150.32	38994200	0.7699	0.00034
australian	8	87.391	1	85.507	1	85.507	25234	1095.26	17411460	0.8105	0.00006
cylinder-b	2	84.815	1	77.037	2	82.593	48159	2942.15	26005860	0.8245	0.00000
CRX	10	87.246	1	85.507	1	85.507	29539	1717.62	20381910	0.8176	0.00009
Dermatology	18	98.108	1	49.444	4	90.158	132554	5907.49	48514764	0.9195	0.00052
German-c	7	75.300	1	70.600	2	73.100	51731	3746.51	51731000	0.7240	0.00007
Hill valley	13	67.076	1	53.955	7	68.641	234298	13501.79	283969176	0.6898	0.00002
Ionosphere	10	91.444	1	80.929	2	84.024	54957	3021.51	19289907	0.8974	0.00019
m-libras	36	87.778	1	27.778	11	88.333	365519	19845.03	131586840	0.8619	0.00021
Musk 1	38	90.142	1	63.440	9	87.402	482995	32275.78	229905620	0.8987	0.00001
parkinsons	14	99.500	1	82.553	2	91.237	64160	3064.17	12511200	0.9291	0.00010
Sonar	47	83.643	1	65.857	5	86.524	123250	6574.17	25636000	0.8783	0.00030
Soybean	22	93.992	1	35.311	6	89.152	105474	6562.73	72038742	0.8684	0.00017
SPECTF	12	81.311	1	77.578	3	80.470	52126	2957.35	13917642	0.8137	0.00020
Vehicle	11	69.048	1	48.571	2	65.245	54751	5042.47	46319346	0.6778	0.00011
vote	3	96.327	1	95.640	1	95.634	19848	809.41	8633880	0.9022	0.00000
wdbc	28	97.719	1	90.670	2	95.789	73955	3973.52	42080395	0.9424	0.00030

Appendix D

Exhaustive search comparison

An exhaustive search was implemented to compare its results with those obtained by the differential evolution procedure used in this document's research proposal. The four datasets with the smallest amount of features were selected. The chosen datasets for experimentation are CRX (15 features), australian (14 features), vote (16 features), and vehicle (18 features).

First, Figure D.1 compares the features selected by the exhaustive search method and the proposals using the $DE-FS^{PM}$ algorithm with the fixed and incremental sampling fraction proposals and the memory to avoid repeated evaluations. The presented heat maps represent the features selected by the exhaustive search and the frequency of selecting the respective feature by the other methods. Since the DE search procedure is stochastic, a different subset of features could be selected in each run.

It is seen that, in some particular cases, the frequency of selection for some features coincides with the exhaustive search selection. These clear examples are feature eight in the CRX dataset, features seven and eight in the australian dataset, and feature three in the vote dataset. The same happens in some of the discarded features as feature fifteen in the CRX dataset, features zero, one, and twelve in the vote dataset, and features fourteen and fifteen in the vehicle dataset.

Nonetheless, the procedure presents more variations in selecting other features, choosing them sometimes, and discarding them in others. Given this, it is necessary to make a numerical comparison of the performance of the methods. Table D.1 shows the

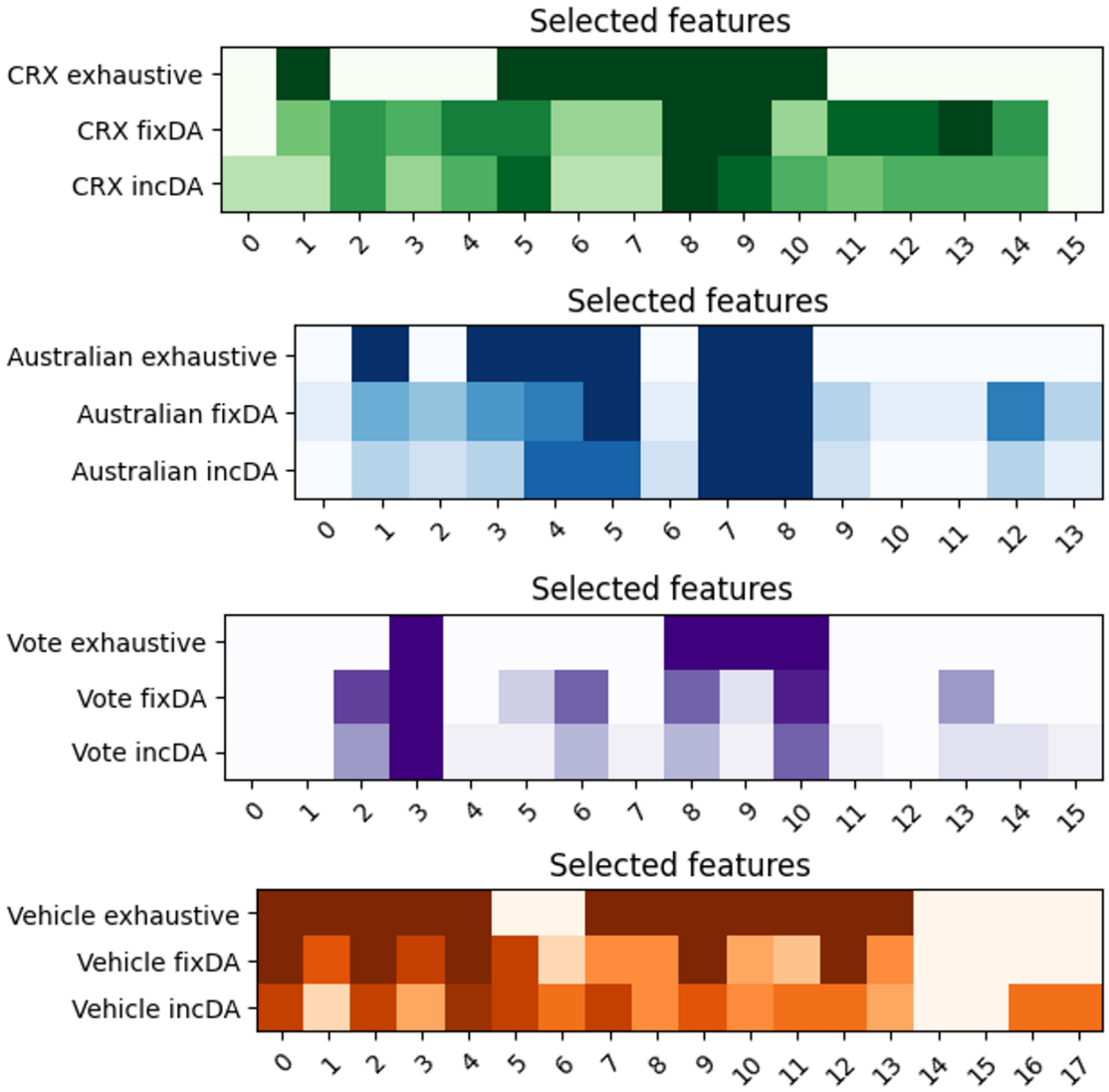

Figure D.1: Heat-maps representing the features selected by exhaustive search and the proposed fixed and incremental sampling fraction methods with memory to avoid repeated evaluations.

results of the methods with the following measures:

- #Fselected: Number of selected features.
- wFS search: Evaluation during the search process for the last generation's best individual with Feature Selection. A stratified 5-fold cross-validation process is used.
- wFS woIS Eval: Final evaluation of the solution found with feature selection and without instance selection. A stratified 10-fold cross-validation process is used.
- # Evals: Number of evaluations required during the search process.
- Time: Amount of time that was taken during the process.
- Instances: Number of instances used during the evaluations in the search process.

Table D.1: Results of using an exhaustive search for feature selection and the fixed and incremental sampling methods with memory to avoid repeated evaluations.

Dataset	Method	#Fselected	wFS search	wFS woIS Eval	#Evals	Time	Instances
CRX	Exhaustive	7.0	88.116	87.681	32768.0	1655.04	22609920.0
	Fixed DA	10.1	88.391	87.116	13498.6	703.82	5588420.4
	Incre DA	8.6	87.362	86.928	7460.4	331.34	3676758.2
Australian	Exhaustive	6.0	87.971	87.391	16384.0	618.96	11304960.0
	Fixed DA	6.9	87.957	86.899	6377.8	240.65	2640409.2
	Incre DA	5.2	86.652	86.696	4284.5	135.03	1998753.3
Vote	Exhaustive	4.0	96.552	95.618	65536.0	1897.24	28508160.0
	Fixed DA	5.1	96.874	96.140	3990.2	121.26	1041442.2
	Incre DA	4.0	96.230	96.251	2472.7	68.24	691102.1
Vehicle	Exhaustive	12.0	76.592	76.129	262144.0	15371.73	221773824.0
	Fixed DA	9.7	76.137	74.099	16447.8	1508.88	8355482.4
	Incre DA	9.8	74.385	73.415	11364.0	916.32	6826543.4

To make an easier comparison, Table D.2 shows the results regarding the percentage drop obtained in the metrics taking the exhaustive search procedure as the basis. It is seen that the number of selected features is primarily augmented; only in the vehicle dataset is reduced. The evaluation values for the selected subset of features are close in three of four datasets. In the vehicle dataset, it is noticed a significant drop in performance.

On the other hand, the number of evaluations, time, and instances is reduced considerably in all cases using the metaheuristic for guiding the search. With this, the main objective of using DE as an optimization algorithm is accomplished, providing

Table D.2: Results of using an exhaustive search for feature selection and the fixed and incremental sampling methods with memory to avoid repeated evaluations.

Dataset	Method	#Fselected	wFS search	wFS woIS Eval	#Evals	Time	Instances
CRX	Fixed DA	-44.29%	-0.31%	0.64%	58.81%	57.47%	75.28%
	Incre DA	-22.86%	0.86%	0.86%	77.23%	79.98%	83.74%
Australian	Fixed DA	-15.00%	0.02%	0.56%	61.07%	61.12%	76.64%
	Incre DA	13.33%	1.50%	0.80%	73.85%	78.18%	82.32%
Vote	Fixed DA	-27.50%	-0.33%	-0.55%	93.91%	93.61%	96.35%
	Incre DA	0.00%	0.33%	-0.66%	96.23%	96.40%	97.58%
Vehicle	Fixed DA	19.17%	0.59%	2.67%	93.73%	90.18%	96.23%
	Incre DA	18.33%	2.88%	3.56%	95.66%	94.04%	96.92%

good results at a fraction of the cost of an exhaustive procedure. While the exhaustive search requires 2^n evaluations where n is the number of features of the dataset, DE uses a considerably smaller number of evaluations. If the difference is evident even with the datasets with the smallest number of features, the savings should be even more significant in larger datasets.

Bibliography

[1] H. M. Abdulwahab, S. Ajitha, and M. A. N. Saif. Feature selection techniques in the context of big data: taxonomy and analysis. *Applied Intelligence*, 52(12):13568–13613, Sep 2022.

[2] M. F. Ahmad, N. A. M. Isa, W. H. Lim, and K. M. Ang. Differential evolution: A recent review based on state-of-the-art works. *Alexandria Engineering Journal*, 61(5):3831–3872, 2022.

[3] Q. Al-Tashi, S. J. Abdulkadir, H. M. Rais, S. Mirjalili, and H. Alhussian. Approaches to multi-objective feature selection: A systematic literature review. *IEEE Access*, 8:125076–125096, 2020.

[4] I. M. R. Albuquerque, B. H. Nguyen, B. Xue, and M. Zhang. A novel genetic algorithm approach to simultaneous feature selection and instance selection. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 616–623, 2020.

[5] J.-A. Barradas-Palmeros, E. Mezura-Montes, H.-G. Acosta-Mesa, and R. Rivera-López. Fitness function comparison for unsupervised feature selection with permutational-based differential evolution. In A. Y. Rodríguez-González, H. Pérez-Espinosa, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and J. A. Olvera-López, editors, *Pattern Recognition*, pages 58–68, Cham, 2023. Springer Nature Switzerland.

[6] M. Bramer. *Data for Data Mining*, pages 9–19. Springer London, London, 2016.

[7] J. Brest, M. S. Maučec, and B. Bošković. il-shade: Improved l-shade algorithm for single objective real-parameter optimization. In *2016 IEEE Congress on Evo-*

lutionary Computation (CEC), pages 1188–1195, 2016.

[8] J. Brest, M. S. Maučec, and B. Bošković. Single objective real-parameter optimization: Algorithm jso. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1311–1318, 2017.

[9] C. A. Coello Coello, S. González Brambila, J. Figueroa Gamboa, M. G. Castillo Tapia, and R. Hernández Gómez. Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. *Complex & Intelligent Systems*, 6(2):221–236, jul 2020.

[10] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Basic Concepts*, pages 1–60. Springer US, Boston, MA, 2007.

[11] K. Deb. Multi-objective optimization. In E. K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 403–449. Springer US, Boston, MA, 2014.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[13] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[14] P. Dhal and C. Azad. A comprehensive survey on feature selection in the various fields of machine learning. *Applied Intelligence*, 52(4):4543–4581, Mar 2022.

[15] T. Dokeroglu, A. Deniz, and H. E. Kiziloz. A comprehensive survey on recent metaheuristics for feature selection. *Neurocomputing*, 494:269–296, 2022.

[16] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[17] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[18] M. T. M. Emmerich and A. H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609, Sep 2018.

[19] E. Hancer, B. Xue, and M. Zhang. Fuzzy filter cost-sensitive feature selection with differential evolution. *Knowledge-Based Systems*, page 108259, 2022.

[20] P. Hu, J.-S. Pan, S.-C. Chu, and C. Sun. Multi-surrogate assisted binary particle swarm optimization algorithm and its application for feature selection. *Applied Soft Computing*, 121:108736, 2022.

[21] F. Jiménez, G. Sánchez, J. Palma, and G. Sciavicco. Three-objective constrained evolutionary instance selection for classification: Wrapper and filter approaches. *Engineering Applications of Artificial Intelligence*, 107:104531, 2022.

[22] T. Kitamura and A. Fukunaga. Duplicate individuals in differential evolution. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2022.

[23] M. Kubat. *Similarities: Nearest-Neighbor Classifiers*, pages 43–64. Springer International Publishing, Cham, 2017.

[24] S. Kukkonen and J. Lampinen. Gde3: the third evolution step of generalized differential evolution. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 443–450 Vol.1, 2005.

[25] H. Liu, M. Zhou, and Q. Liu. An embedded feature selection method for imbalanced data classification. *IEEE/CAA Journal of Automatica Sinica*, 6(3):703–715, 2019.

[26] M. Malekipirbazari, V. Aksakalli, W. Shafqat, and A. Eberhard. Performance comparison of feature selection and extraction methods with random instance selection. *Expert Systems with Applications*, 179:115072, 2021.

[27] T. Marill and D. Green. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9(1):11–17, 1963.

[28] J. A. Mendoza-Mota. *Selección de atributos bajo un enfoque evolutivo multiobjetivo*. Laboratorio Nacional de Informática Avanzada, 2021.

[29] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler. A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143, Aug 2010.

[30] K. Price, R. M. Storn, and J. A. Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[31] P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

[32] R. Rivera-López, E. Mezura-Montes, J. Canul-Reich, and M. A. Cruz-Chávez. A permutational-based differential evolution algorithm for feature subset selection. *Pattern Recognition Letters*, 133:86–93, 2020.

[33] J. R. Schott. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Massachusetts Institute of Technology, 1995.

[34] L. Shu, F. He, X. Hu, and H. Li. A novel feature selection with many-objective optimization and learning mechanism. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 684–689, 2021.

[35] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[36] R. Tanabe and A. Fukunaga. Success-history based parameter adaptation for differential evolution. In *2013 IEEE Congress on Evolutionary Computation*, pages 71–78, 2013.

[37] R. Tanabe and A. S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1658–1665, 2014.

[38] B. Tran, B. Xue, and M. Zhang. Variable-length particle swarm optimization for feature selection on high-dimensional classification. *IEEE Transactions on Evolutionary Computation*, 23(3):473–487, 2018.

[39] C.-F. Tsai, W. Eberle, and C.-Y. Chu. Genetic algorithms in feature and instance

selection. *Knowledge-Based Systems*, 39:240–247, 2013.

[40] A. Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, C-20(9):1100–1103, 1971.

[41] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[42] B. Xue, M. Zhang, W. N. Browne, and X. Yao. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, 2016.

[43] Y. Xue, Y. Tang, X. Xu, J. Liang, and F. Neri. Multi-objective feature selection with missing data in classification. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):355–364, 2021.

[44] Yogender and M. Pal. Effect of search methods on feature selection with hyperspectral data. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 3408–3411, 2021.

[45] J. Zhang and A. C. Sanderson. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.

[46] N. Zhang, A. Gupta, Z. Chen, and Y.-S. Ong. Evolutionary machine learning with minions: A case study in feature selection. *IEEE Transactions on Evolutionary Computation*, 26(1):130–144, 2021.

[47] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Shaker Ithaca, 1999.