UNIVERSITY OF MACEDONIA

GRADUATE PROGRAM

DEPARTMENT OF APPLIED INFORMATICS

A REDUCED VARIABLE NEIGHBORHOOD SEARCH APPROACH FOR GENE SELECTION IN CANCER CLASSIFICATION

Master Thesis

of

Pentelas Angelos

Thessaloniki, 06/2019

A REDUCED VARIABLE NEIGHBORHOOD SEARCH APPROACH FOR GENE SELECTION IN CANCER CLASSIFICATION

Pentelas Angelos

Bachelor in Mathematics, Aristotle University of Thessaloniki, 2015

Master Thesis

submitted for the partial fulfillment of the demands of the

MASTER OF SCIENCE IN APPLIED INFORMATICS

Supervisor Professor
Sifaleras Angelo

Approved by the three-member committee 25/06/2019

Sifaleras Angelo	Samaras Nikolaos	Hristu-Varsakelis Dimitris
....	....	....

Pentelas Angelos

....

Abstract

In this work we propose a Reduced Variable Neighborhood Search (RVNS) metaheuristic algorithm, to handle the gene selection problem in cancer classification. RVNS is utilized as the primary search method and gene subsets obtained are evaluated by three learning algorithms, namely support vector machine, k-nearest neighbors, and random forest. Experiments are conducted on five publicly available cancer-related datasets, all characterized by a small sample size and high dimensionality. Since RVNS seeks gene subsets that yield accurate predictions for all three aforementioned classifiers, the proposed results can be considered more reliable. To the best of our knowledge, the said methodology is innovative due to the fact that, it combines the Recursive Feature Elimination (RFE) heuristic with an RVNS algorithm. Despite the large size of the problem instances, the proposed FS scheme converges within reasonably short time. Results indicate high performance of RVNS that, is further improved when the RFE method is applied as a pre-processing step. Moreover, our approach seems to outperform similar recent algorithms in both terms of accuracy and run-time.

Keywords: Reduced Variable Neighborhood Search, Feature Selection, Cancer Classification.

THANKS

I would like to thank my supervisor professor, Dr. Angelo Sifaleras, for his unreserved help and support during my work on this thesis. His suggestions and comments were more than insightful. In addition, I am thankful for Dr. Georgia Koloniari's guidance on topics related to her expertise that my thesis bears on. Last, I want to thank my parents who, discretely, always support me.

This work is dedicated to my family.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

Introduction

Machine intelligence is an achievement humans have been seeking to reach for centuries. The advancements in computing performance along with a well-established theoretical background have been proven decisive towards that goal. Thus, Artificial Intelligence (AI) altered from a completely theoretical scientific field to both theoretical and applied. Due to the new circumstances, Artificial (or Machine) Intelligence was, and still is, a very tempting scientific area for a wide range of researchers and industries originating from quite diverse domains. The phenomenal scientific and business focus on AI resulted in partitioning this vast field. Thus, terms like Machine Learning (ML) and Deep Learning emerged.

At the same time, other technological achievements and industry needs resulted in a tremendous rise on data generation. The feeling that data-driven decisions carry less bias motivated people to develop devices and techniques that enable them to store information about, almost, everything. That led to the introduction of a new term called Big Data. Within the spectrum of Big Data lie datasets that can be described by the, so called, four V's. They stand for volume (i.e., data scale), velocity (e.g., streaming data), veracity (i.e., uncertainty of data) and variety (i.e., different forms of data).

In the aftermath of the aforementioned facts, AI and ML applications usually take advantage of Big Data and many experts nowadays consider them intertwined. Such applications can be found in many aspects of modern societies. In retail, where a common task is to uncover potential clients; in finance, where stock market analysis has been reinvented under the recent developments; in transportation, where major steps towards autonomous vehicles have been made; in healthcare, where ML algorithms enhance doctors' judgment on whether a person is sick or not, are only a few indicative examples.

However, applying the available techniques rarely is a piece of cake. Despite all the helpful software solutions that have been developed (e.g., python's machine learning libraries, WEKA, tensorflow, keras etc.) some datasets remain complex enough and hard to manipulate or mine knowledge from. To ease the manipulation of complex datasets, experts like data scientists and machine learning engineers use to apply data pre-processing methods whose purpose, among others, is to produce simpler and more comprehensible datasets.

Bioinformatics is one of the many scientific domains leveraging modern data manipulation techniques and, in fact, many processes within the said field strongly rely on the latter. As Figure **1.0.1** illustrates, there is a wide variety of ML applications within the world of bioinformatics. During the present thesis, we specifically address challenges and offer solutions within the *diagnosis and prognosis* domain.

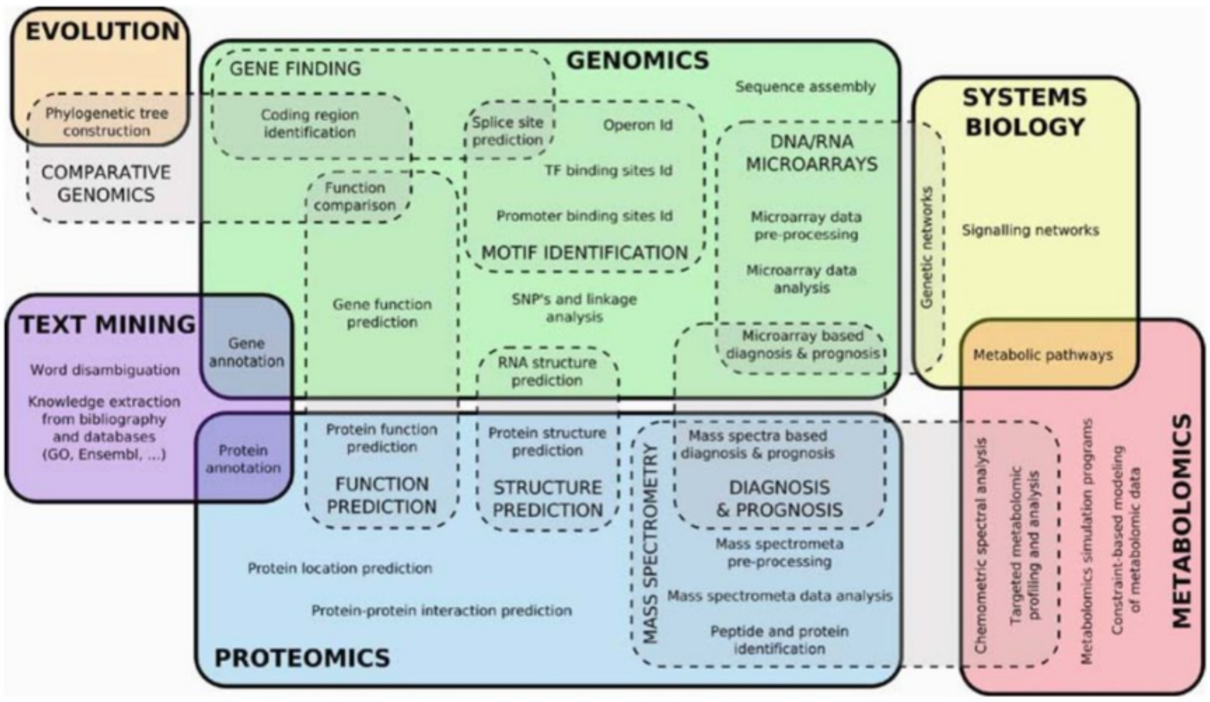

EVOLUTION
Phylogenetic tree construction
COMPARATIVE GENOMICS
GENOMICS
GENE FINDING
Coding region identification
Splice site prediction
Operon Id
Sequence assembly
DNA/RNA MICROARRAYS
Function comparison
TF binding sites Id
Promoter binding sites Id
MOTIF IDENTIFICATION
Microarray data pre-processing
Microarray data analysis
Gene function prediction
SNP's and linkage analysis
Microarray based diagnosis & prognosis
RNA structure prediction
SYSTEMS BIOLOGY
Genetic networks
Signalling networks
Metabolic pathways
TEXT MINING
Word disambiguation
Knowledge extraction from bibliography and databases (GO, Ensembl, ...)
Gene annotation
Protein annotation
Protein function prediction
FUNCTION PREDICTION
Protein structure prediction
STRUCTURE PREDICTION
MASS SPECTROMETRY
Mass spectra based diagnosis & prognosis
DIAGNOSIS & PROGNOSIS
Mass spectrometa pre-processing
Mass spectrometa data analysis
Peptide and protein identification
Protein location prediction
Protein-protein interaction prediction
PROTEOMICS
Chemometric spectral analysis
Targeted metabolomic profiling and analysis
Metabolomics simulation programs
Constraint-based modeling of metabolomic data
METABOLOMICS

Figure **1.0.1**: General scheme of the current applications of machine learning techniques in bioinformatics.

Source: https://twitter.com/gp_pulipaka

Diagnosis and Prognosis, essentially, encompasses the *microarray based diagnosis and prognosis* and the *mass spectra based diagnosis and prognosis.* Microarrays can be used to measure the relative quantities of specific mRNAs in two or more tissue samples for thousands of genes simultaneously (Kerr, Martin, and Churchill 2000), while mass spectrometry is a central analytical technique for protein research and for the study of biomolecules in general (Domon and Aebersold 2006). It is worth to point out that, in this thesis, we mainly focus on the computational aspects of diagnosis and prognosis, therefore, we will not further elaborate on the said terms.

A common characteristic of the two approaches is the fact that they both yield datasets with thousands of features (genes), while, at the same time, a high degree of gene redundancy is believed to be an issue (Yu and Liu 2004). Yu and Liu also mention that, most of the time, such datasets involve measurements from just a few patients, which in statistical terms means they consist of a small sample size. It is the combination of both a vast feature space and the small sample size that diminishes our effort in analyzing this kind of datasets. Luckily, ML utilizes techniques from statistics and combinatorial optimization that help towards the direction of simplifying them.

1.1 Problem Statement - Significance

An indispensable segment of data pre-processing and ML is Feature Selection (FS). FS is the process of selecting a subset of the dataset's attributes (i.e., features) that is considered relevant to the imminent analysis' scope. For instance, let's assume someone wants to analyze the semester's grade of ten students in a class given information about their actual grade (Feature A or class/target variable), how much time they spend studying per week (Feature B), how much time they spend on social media per week (Feature C) and if they own a pet or not (Feature D). A FS method would probably end up with the subset {Feature B, Feature C} in order to extract only relevant information concerning students' grades.

The aforementioned example might seem a simple one. However, in most challenging datasets it is prohibitive to judge a-priory the importance of a feature. Even in this naive example, someone could find a strong correlation between students' grades and their ownership of a pet. Thus, FS is a process that can't be treated in an offhand way since valuable insights might come from any possible subset of the initial feature set. Mathematically speaking, in a dataset with d available features there are $2^d - 1$ possible subsets that need to be evaluated in some way before selecting the final one, meaning that the problem's complexity grows exponentially to the number of initial features. In fact, FS has been proven an NP-hard problem (Narendra and Fukunaga 1977) and traditional commercial solvers fail to produce good results in a reasonable amount of time.

It is also valuable to underline the benefits of FS in a more formal manner. Typically, the significance of FS lies on the following four points:

1. model simplification,
2. shorter computing time,
3. curse of dimensionality avoidance,
4. enhance generalization by avoiding overfit

While the point of (2) is obvious, a short commenting is needed for (1), (3) and (4). The perks of a simple model are mainly related to the model's easier interpretation by experts. It is also our belief that, the simpler a model, the better. Concerning the third allegation, curse of dimensionality is a term that describes the difficulty to identify data points that are distant or close to each other in a high dimensional space (i.e., the concept of distance deteriorates as the dimensions grow). Finally, ideal ML models can be generalized, meaning that they can make right predictions even on a wide range of unknown/future data. Overfitting depicts the state of a model that is only capable of producing correct predictions on data it was trained with (i.e., data known to the model) and is unable to do the same with unknown/future data.

1.2 Purpose - Goal

The scope of this thesis is to develop and examine the behavior of a *Wrapper* and an *Embedded-Wrapper* hybrid FS technique on five well known public cancer-related datasets which consist of thousands of attributes but only a few samples (i.e., high dimensional/small sample size datasets). Most of these datasets have been extensively used from researchers in an attempt to obtain genes that, based upon their values, someone will be able to predict patients with malignant cancer or to classify cancer types. The proposed FS schemes utilize the Variable Neighborhood Search metaheuristic algorithm as a baseline for the selection of the genes.

The methodology that is used aims to overcome some of the greatest drawbacks of wrapper techniques that, as we will later see, are model overfitting and extensive computing time. Hopefully, the proposed algorithms will end up discovering few but informative features from each dataset and the final results will enable studies in cancer research move forward.

1.3 Assumptions

During this work, we take three points for granted. The first is related with the objective of our approach. According to previous work on the field, we go after implementing a methodology that is able to produce as smaller gene subsets as possible, while keeping the learning algorithms' accuracy high enough. The second assumption that we considerate regards the correctness of the datasets that we later use to evaluate our methods. Finally, for the rest of this thesis, the terms feature, gene, attribute and variable have the same meaning within the scope of FS.

1.4 Contribution

Variable Neighborhood Search and its variants have been slightly studied within the FS process and, especially, on cancer-related data. To the contrary, other metaheuristic algorithms like genetic algorithms, simulated annealing or particle swarm optimization have been overused. Thus, this thesis contributes to the scientific community in terms of expanding the set of known FS methods and their performance within the cancer classification domain.

1.5 Thesis Structure

The remaining of this thesis is structured as follows. In Chapter 2 we state the necessary theoretical background on which our algorithmic development relies. More specifically, in this chapter we describe the class of combinatorial optimization problems, their computational complexity and the need of efficient intelligent-searching algorithms. Moreover, we give a high-level overview of ML and some widely used and known algorithms that are later integrated into our method. We then unfold the gains metaheuristics bring into the class of combinatorial optimization problems and, more specifically, we describe the variable neighborhood search algorithm along with its variant, the reduced variable neighborhood search, the core of both of our algorithms. Finally, we proceed in defining the FS problem and thoroughly present the categories FS algorithms are divided into according to the literature.

We move on in detailing some of the most recent research work in the field of cancer classification in Chapter 3. Here, we are concerned about the FS category each method lies in, the search methods used, the learning algorithms that evaluate each gene subset, as well as the final results regarding accuracy and computational time.

Next comes Chapter 4. It is within this chapter that we meticulously mention each step and intuition of the proposed methodology. In addition to algorithmic specifications and parameter settings, the reader is also referenced to the technical stack used during this thesis.

Having discussed about the methodology that we follow, we proceed in presenting the results and the overall performance of our methodologies both separately in each dataset and in comparison with other well-performing recent approaches. All of the above can be found in Chapter 5.

Chapter 6 addresses the fundamental statistical analysis of our algorithms. This analysis aims at giving insights on performance metrics (e.g., accuracy, gene subset size) and search efficiency factors (e.g., computational time, improvements per neighborhood).

Finally, in Chapter 7, we comment both on our approach to tackle the gene selection problem and the gains of recent trends applied in the field. We conclude in the same chapter by presenting some general thoughts and future directions regarding combinatorial optimization.

CHAPTER 2

Theoretical Background

Before diving into the proposed methodology's technical details, it is useful and insightful to set the necessary theoretical background. In this chapter, we introduce the Machine Learning (ML) field and some, widely used, ML algorithms that are involved in our work. We also discuss the concept of metaheuristics and their applications, especially within the Combinatorial Optimization context. Emphasis is given on the background of a specific metaheuristic algorithm called Variable Neighborhood Search (VNS) along with its variant, Reduced VNS, which is moreover integrated in our methodology. Last, the Feature Selection (FS) problem and the categorization of FS techniques are presented. For the reader's convenience, we start the background overview from some fundamental aspects this thesis deals with, which are the computational concerns and issues that, eventually, helped the beautiful scientific area of combinatorial optimization grow.

2.1 The Need for Efficient Search

This thesis would be deficient without introducing the reader to the scientific area of Combinatorial Optimization (CO). Tightly attached to the *computational complexity* domain, CO encompasses a wide variety of problems met in real life, that demand fast and effective solutions, while most of them belonging in the class of NP-hard problems. The latter, essentially indicates that such problems have a vast solution space that grows exponentially with the problem size and, most importantly, there is no known algorithm that gives an exact solution within polynomial time. For more on this topic, we reference the reader to (Papadimitriou 2003).

According to `cs.cmu.edu` glossary, CO is the process of searching for maxima (or minima) of an objective function F whose domain is a discrete but large configuration space. Examples of combinatorial optimization problems are:

1. The Travelling Salesman Problem: given the (x,y) positions of N different cities, find the shortest possible path that visits each city exactly once, starting from a certain one.

2. The Bin-Packing Problem: given a set of N objects, each with a specified size s_i, fit them into as few bins (each of size B) as possible.

3. The Assignment Problem: given N employees, N jobs and c_{ij} the cost of employee i performing job j, find the less costly mapping between employees and jobs.

4. The Job-Shop Scheduling Problem: given N jobs of varying processing times, which need to be scheduled on M machines with varying processing power, minimize the makespan, i.e., the time between the initialization of the first job and the termination of the last one.

5. The p-median Problem: given a set L of candidate location points and a set D of demand nodes, locate p facilities to minimize the demand weighted average distance between demand nodes and the nearest of the selected facilities.

It can be easily observed that, this kind of problems have a wide applicability within many industries. For instance, consider a typical problem most cloud data-centers have to deal with. They receive a continuous stream of virtual machine instantiation requests that need to be placed into the data-center's servers. Ultimately, this is a Bin-Packing problem, where servers represent the bins and virtual machines the items. Beloglazov et. al., in (Beloglazov, Abawajy, and Buyya 2012) challenge this problem targeting an energy efficient consolidation policy. Another example, this time in the logistics industry, is the the need of an optimal vehicle routing policy for trucks delivering goods in warehouses or stores. This problem is called the vehicle routing problem and consists a generalization of the Travelling Salesman Problem. Further information can be found in (Toth and Vigo 2002).

A substantial difference of CO Problems (COPs) with other optimization ones is that, algorithms developed to *exactly* solve them require a huge amount of computational time, even with modern hardware materials. With the term *exactly*, we refer to the capability of an algorithm to obtain the mathematically best, among trillions, quadrillions or more, of feasible solutions for a problem. Naturally comes the question; *given a problem and an algorithm that exactly solves it, is the problem practically solvable if the algorithm takes years to compute a solution?*.

Today's situation is that, companies striving for solutions to their COPs, are, in fact, willing to sacrifice their optimality with a reasonable computing time. Reasonable computing time is subjective, of course, and significantly varies among problem domains. A couple of hours might do the job for a hospital trying to assign shifts to its doctors, but the same amount of time is extremely large for a network route planning for an internet package to traverse.

An initial approach to the said issue, i.e., of finding good but not necessarily the best solutions, is to run exact algorithms in any COP and terminate them whenever it suits us. However, following this technique someone would likely acquire a poor solution and the reason of it is that exact algorithms are not developed to behave that way, meaning that, even at their penultimate execution step, there is no guarantee that they carry a good enough solution.

That led to the adoption of a special kind of algorithms, capable of finding acceptable solutions within short computing times. Their name is *heuristics*, reminding the Greek word used for *finding*. As their name implies, heuristics are a separate category of algorithms that obtain feasible solutions to problems fast, by finding a solution according to a certain search policy. Their main characteristic is that they are problem dependent, meaning that they are developed to tackle a specific COP. As such, they cannot be generally used to any kind of COP. Heuristics do not guarantee, by any means, a convergence to an optimal solution. Thereby, they pertain to the family of approximate algorithms.

Several heuristic techniques have been implemented to tackle NP-hard COPs. In the literature, one can find the savings algorithm (Paessens 1988), which was developed to produce acceptable solutions to the vehicle routing problem, the recursive feature elimination heuristic (Guyon, Weston, Barnhill, and Vapnik 2002) for the feature selection problem, that is also later used in our work and the best-fit heuristic which performs impressively in some applications of the bin-packing problem. Those are just three among, perhaps, hundreds or thousands of heuristics especially developed to face difficult COPs.

There comes the queries, though, of how good do eventually these heuristics perform and how they behave with respect to the objective function they try to optimize. The answer to the first question is that the goodness of a solution is again subjective and problem dependent. The interested individual has to decide whether they are happy or not with a solution and, in any case, they should stay always motivated to improve their heuristics. In our opinion, heuristics' greatest flaw comes from the second concern. Since they usually follow a certain, pre-determined, policy to find or construct a solution, they get easily trapped into local optima and have no means of escaping from them. An extensive research towards keeping the good and avoiding the bad from heuristics led to a new kind of algorithms, *metaheuristics*.

2.2 Metaheuristic Methods in Combinatorial Optimization

Metaheuristics are the most recent development in approximate search methods for solving complex optimization problems that arise in business, commerce, engineering, industry and many other areas. A metaheuristic guides a subordinate heuristic using concepts derived from artificial intelligence, biological, mathematical, natural and physical sciences to improve their performance (Osman and Kelly 1996). Given a complex problem, their purpose is to obtain a fairly good solution within a reasonable amount of time. Thus, metaheuristics do not guarantee optimal solutions but they utilize intelligent search techniques to quickly find acceptable solutions within the solution space and, moreover, they encompass mechanisms to escape from local optima.

Depending on the characteristics selected to differentiate among them, several classifications of metaheuristics are possible, each of them being the result of a specific viewpoint (Blum and Roli 2003). From a *number of solutions used at the same time* standpoint, metaheuristics can be divided in population-based and single point search. Algorithms working on single solutions are called trajectory methods and encompass local search-based metaheuristics like Tabu Search, Simulated Annealing and Iterated Local Search. On the other hand, population-based metaheuristics will perform search processes which describe the evolution of a set of points in the search space (Blum and Roli 2003). Genetic Algorithms, Particle Swarm Optimization and Ant Colony Optimization are three well-known algorithms that lie in the population-based category.

For the FS problem, which can be described as a COP, Variable Neighborhood Search (VNS) is the metaheuristic algorithm selected to be studied. In the next three sections, VNS, a VNS variant called Reduced VNS and a number of VNS parallel schemes are described.

2.2.1 Variable Neighborhood Search

In (Mladenović and Hansen 1997), N. Mladenović and P. Hansen proposed a new metaheuristic algorithm called Variable Neighborhood Search (VNS). As authors suggest, the novelty of the new algorithm lies on the fact that, while other local search methods follow a trajectory, VNS explores increasingly distant neighborhoods of the current incumbent solution and jumps from there to a new one if and only if an improvement is made. Following this pattern, favorable characteristics of the incumbent solution will be kept and used to obtain promising neighboring solutions.

Let $N_k, (k = 1, ..., k_{max})$, be a finite set of pre-selected neighborhood structures and $N_k(x)$ the set of solutions in the k^{th} neighborhood of a solution x. The VNS pseudocode is given in Algorithm 1 while Figure **2.2.1** illustrates the said search process of VNS.

VNS is an effective metaheuristic used for solving many kinds of problems. In (Kytöjoki, Nuortio, Bräysy, and Gendreau 2007), authors propose VNS for confronting very large scale vehicle routing problems, while in (Sifaleras, Konstantaras, and Mladenović 2015) it was used to manage the lot sizing problem. Besides considered a relatively new algorithm and further applications are yet to be explored, VNS utilizes only a few parameters and that is a great advantage compared to other metaheuristics, where parameter tuning can be proved quite time consuming.

Researchers that invented VNS posed three questions regarding the search in multiple neighborhoods:

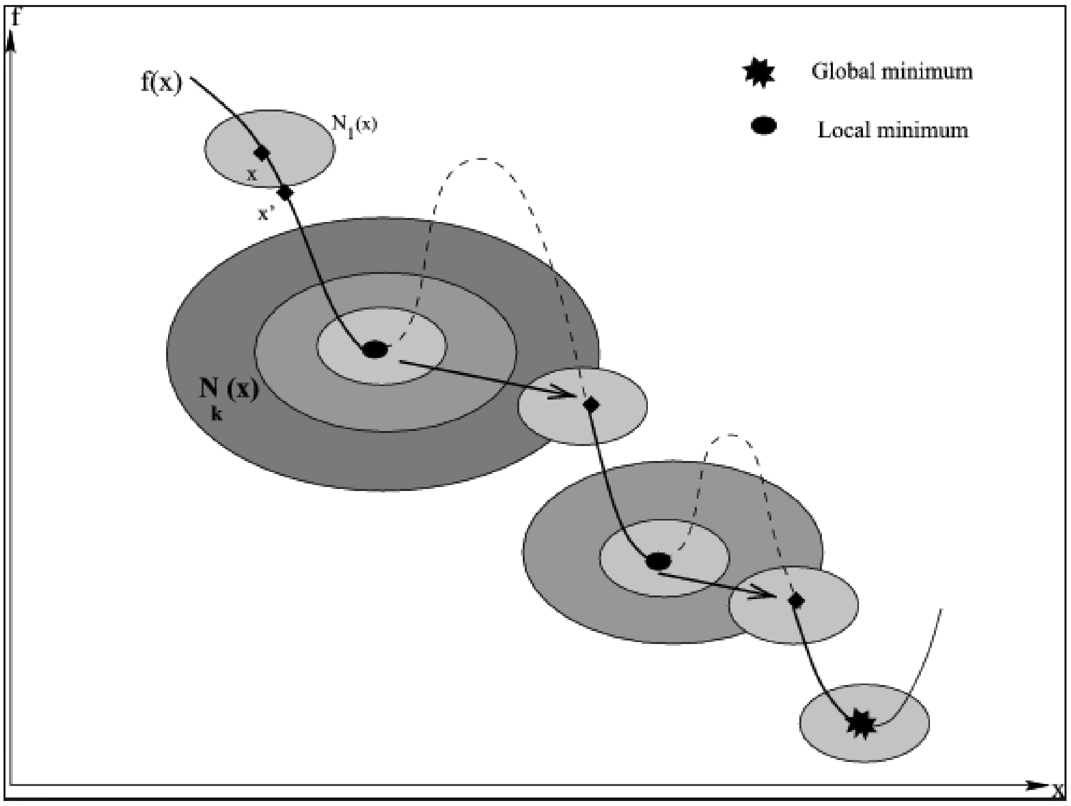


Figure **2.2.1**: Illustration of the VNS procedure.

Source: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-74759-0_694

Algorithm 1: VNS pseudocode for a minimization problem

```
initialize solution $x$;
while stopping criteria are not met do
    $k = 1$;
    while $k < k_{max}$ do
        generate $x'$ a random solution from neighborhood $N_k(x)$;
        $x'' = LocalSearchProcedure(x')$;
        if evalutate($x''$) < evaluate($x'$) then
            $x = x''$;
            $k = 1$;
        else
            $k = k + 1$;
        end
    end
end
return $x$;
```

1. What N_k should be used and how many of them?
2. What should be their order in the search?
3. What search strategy should be used in changing neighborhoods?

Answering the first one, there seems to be no general rule about the number and the structure of neighborhoods. In the literature, authors usually come up with two-four problem-specific neighborhood structures. Concerning their order in the search, again it is something problem-dependent. Worth mentioning that the third question led to many VNS variants like Skewed VNS and VN Descent.

2.2.2 Reduced Variable Neighborhood Search

The Reduced VNS (RVNS) method is obtained if random points are selected from $N_k(x)$ and no local search is applied. Instead, the values of the new points are computed with that of the incumbent and an update takes place in case of improvement. Algorithm 2 describes the said process.

Algorithm 2: RVNS pseudocode for a minimization problem

```
initialize solution $x$;
while stopping criteria are not met do
    $k = 1$;
    while $k < k_{max}$ do
        generate $x'$ a random solution from neighborhood $N_k(x)$;
        if evalutate($x'$) < evaluate($x$) then
            $x = x'$;
            $k = 1$;
        else
            $k = k + 1$;
        end
    end
end
return $x$;
```

RVNS has been proved quite useful in very large instances, for which local search is costly. Although it looks similar to a Monte-Carlo method, it searches in a more systematic manner. In (Xiao, Kaku, Zhao, and Zhang 2011) researchers applied RVNS on several multilevel lot-sizing benchmark problems and obtained, in many cases, new best solutions. RVNS is the main search method used during this work and, as we later show, it performs exceptionally in the FS problem for cancer-related datasets.

2.2.3 Parallel schemes of Variable Neighborhood Search

The application of parallelism to a metaheuristic can and must allow to reduce the computational time (by partitioning the sequential algorithm) or to increase the exploration in the search space (by applying independent search threads) (Hansen, Mladenović, and Pérez 2010). Parallelizations of VNS proposed in the literature can be divided into simple and more complex ones.

Within the category of the simple parallelization strategies someone can find Synchronous Parallel VNS (SPVNS) and Replicated Parallel VNS (RPVNS). The first one attempts to reduce computational time by parallelizing the local search in the sequential VNS while RPVNS tries to search for a better solution by means of the exploration of a wider zone of the solution space, using multistart strategies. Authors in (Hansen, Mladenović, and Pérez 2010) consider the aforementioned techniques simple since they don't utilize sophisticated parallel computing concepts and are pretty easy to implement code-wise.

On the contrary, Replicated-Shaking VNS (RSVNS) and Cooperative Neighborhood VNS (CNVNS) use cooperation (i.e., communication) mechanisms, through a master-worker approach, to improve the performance.

Briefly, in RSVNS, a sequential VNS is being executed in the master processor and the current solution is sent to each worker processor. Workers shake this solution to avoid being trapped in local optima and start a local search from the new neighboring candidate solutions. The final step of RSVNS includes the pass of these improved solutions to the master which selects the best among them and continues the algorithm.

A detailed overview of CNVNS is quoted, since it is a more complex algorithm and the reader might get easily confused:

1. The master process:
   (a) Executes parallel RVNS.
   (b) Sends initial solutions to the individual VNS processes.
   (c) After each communication from an individual VNS process, updates the best overall and communicates it back to the requesting VNS process.
   (d) Verifies the stopping condition.
2. Each VNS process:
   (a) Receives the initial solution, selects randomly a neighborhood and explores it by shaking and local search.
   (b) If the solution is improved, the search proceeds from the first neighborhood with shaking and local search.

(c) If the solution can not be improved, the process:
    i. Communicates its solution to the master.
    ii. Requests the overall best from the master.
    iii. Continues the search from the current solution.

Worth noting that, communication between master and workers happens asynchronously (i.e., not at the same time).

2.3 Machine Learning

As mentioned in Chapter 1, advances in computer technology allow individuals, businesses and organizations to store and process large amounts of data. These stored data become useful only when they are analyzed and turned into information that people can make use of (Alpaydin 2009). Human's intuition that there might be hidden patterns within the observed datasets and that such patterns are impossible to get recognized by the human eye, resulted in the development of ML.

ML is programming computers to optimize a performance criterion using example data or past experience (Alpaydin 2009). The process always involves a mathematical model that is defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. Since ML models are all about learning from training (i.e., existing) data, it becomes obvious that the same models will be applied to future data under the assumption that future will not be much different from the past.

ML applications are divided into two main categories, supervised and unsupervised learning. For supervised learning, classification and regression are the most typical tasks. Classification handles the task of predicting discrete values from a known value set while regression is applied in predicting continuous values. They both use the given values of some attributes in order to predict the value of an unknown one. On the other hand, the major representative of unsupervised learning is clustering. Given a metric, the goal of clustering is to produce groups that are well separated one from another, i.e., maximum inter-distance, and each group should consist of members that are close enough, i.e., minimum intra-distance.

Within this thesis' scope, and since features will be selected for predicting if a patient has cancer or not and the cancer's type, classification is the ML task that will be applied. Next, there is given a short description of the algorithms that are used in the proposed methodology so as the reader gets a full picture of the processes and tools structuring it.

2.3.1 k-Nearest Neighbors

k-Nearest Neighbors algorithm, abbreviated as kNN, is one of the simplest learning algorithms and one of the most frequently used. kNN is a non-parametric (i.e., it does not make any assumptions on the underlying data distribution), lazy learning (i.e., it does not use the training data points to do any generalization) algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point. kNN algorithm is based on feature similarity: How closely out-of-sample features resemble the training set determines how a given data point is classified. In Figure **2.3.2**, the critical importance of the parameter k becomes obvious.

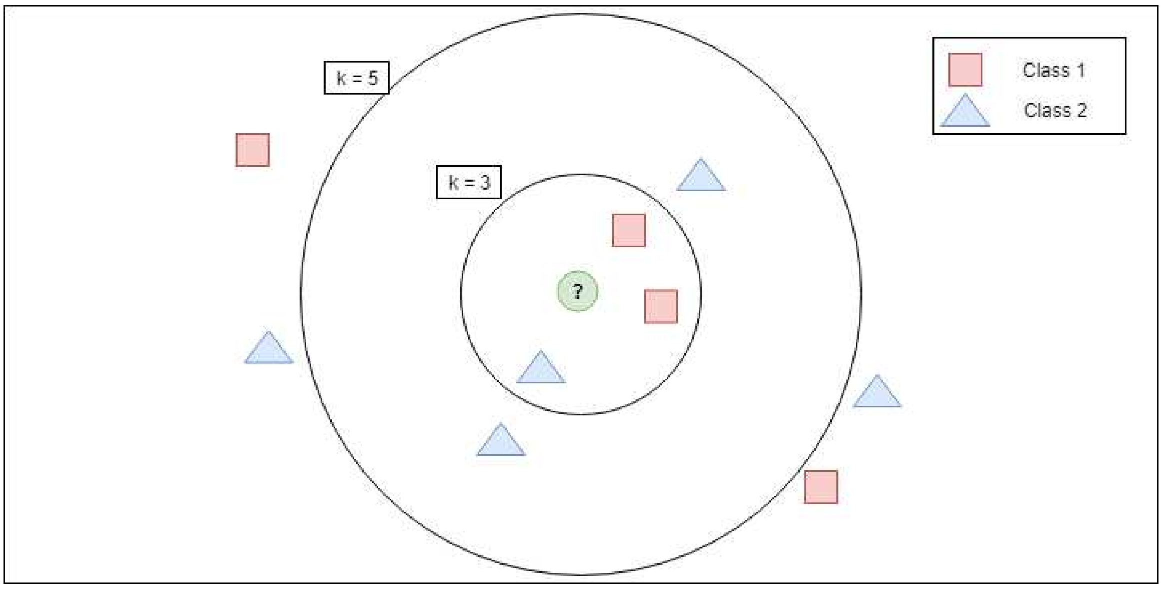


Figure **2.3.2**: The majority of the neighbor classes will determine the new instance's class.

Several studies on the FS problem have utilized kNN as their learning algorithm. In (Chuang, Yang, Wu, and Yang 2011), authors state that some major advantages of the kNN method are its simplicity and ease of implementation, while in (Cover, Hart, et al. 1967) it is mentioned that kNN is not negatively affected by large training data, and is furthermore indifferent to noisy training data.

2.3.2 Support Vector Machine

Support Vector Machine, abbreviated as SVM, can be used for both regression and classification tasks. However, it is mainly used for the second one. The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space(where N denotes the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. The objective is to find a plane that has the maximum margin, i.e, the maximum distance, between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

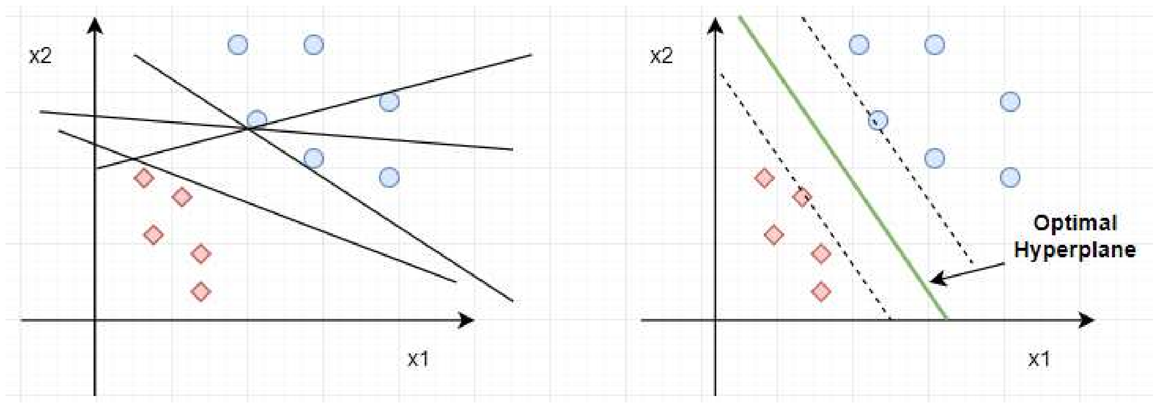


Figure **2.3.3**: SVM finding the optimal hyperplane in a 2-dimensional space

Concerning the discussed subject, many researchers have used SVM for microarray data classification. For instance, in (Alba, Garcia-Nieto, Jourdan, and Talbi 2007) it was combined with a Particle Swarm Optimization variant and with a Genetic Algorithm that were searching the solution space. In fact, SVM has been utilized within the FS problem so extensively, that some of the state-of-the-art methods carry its signature, e.g., SVM-RFE (Guyon, Weston, Barnhill, and Vapnik 2002).

2.3.3 Random Forest

Random Forest is as supervised learning algorithm. As its name implies, it builds multiple decision trees, i.e., forest, and merges them together to get a more accurate and stable prediction. The term *Random* is justified by the fact that, while Random Forest growing the trees, instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally leads to a better model.

Wu et. al. (Wu, Kumar, Quinlan, Ghosh, Yang, Motoda, McLachlan, Ng, Liu, Philip, et al. 2008) underlines that C4.5 algorithm (i.e., a decision tree implementation), is among the top ten most influential data mining algorithms in the research community. This implies that Random Forest is also a popular and safe choice for classification tasks, since it consults with many C4.5 trees to make a decision. In (Saeys, Abeel, and Van de Peer 2008), Random Forest managed to obtain good results within the study's perspective.

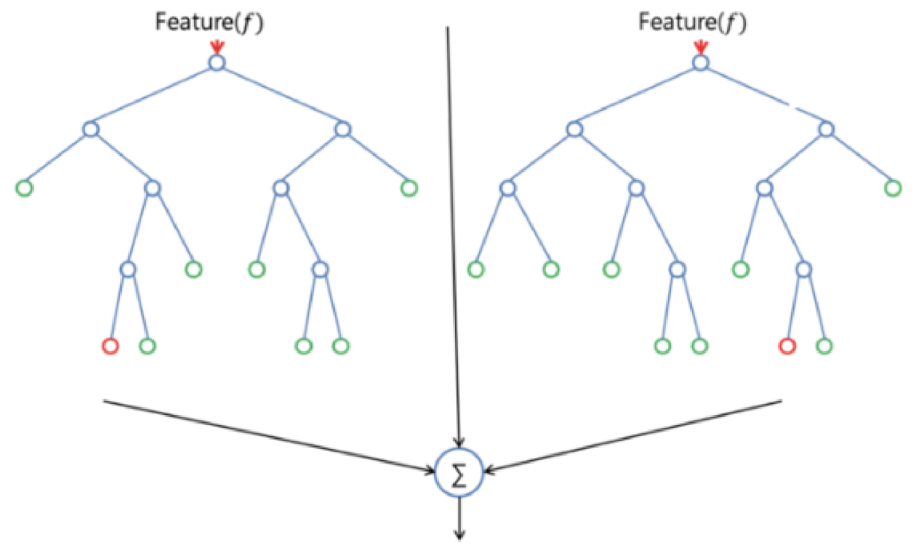


Figure **2.3.4**: Random Forest consults both trees before moving to a decision

Source: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

2.4 Feature Selection

The FS problem is a combinatorial one with $O(2^d)$ computational complexity, where d denotes the number of Features. In (Narendra and Fukunaga 1977) it has been proved that it is an NP-hard problem, meaning that it belongs to the class of problems that are *at least as hard as the hardest problems in NP(not-deterministic polynomial) class*. In such problems, an exhaustive search of the solution space is prohibited, even for small problem instances.

FS methods are constantly emerging and, for this reason, there is a wide suite of methods that deal with microarray data (Bolón-Canedo, Sánchez-Marono, Alonso-Betanzos, Benítez, and Herrera 2014). However, experts in the field have not come to an answer when asked which method is the best. Thus, new FS methods will continue to emerge and compete with each other on certain evaluation criteria such as robustness, time complexity, etc..

In this study, a *Wrapper* FS method is developed, with an effort of overcoming some of wrappers' major drawbacks, e.g., overfitting risk. However, in the following sections, brief descriptions of *Filters*, *Embedded* and *Ensemble* FS methods are also given, so as the reader gets an overview of the up-to-date techniques and concepts in the discussed domain.

2.4.1 Filter Technique

The *Filter* approach incorporates an independent measure for evaluating feature subsets without involving a learning algorithm. Thus, filter methods evaluate the goodness of feature subsets by observing only the intrinsic data characteristics (e.g., statistical measures), in which typically a single feature or a subset of features is evaluated against the class label (Bolón-Canedo, Sánchez-Marono, Alonso-Betanzos, Benítez, and Herrera 2014). Figure **2.4.5** depicts a scheme of a typical Filter FS procedure for a classification task.

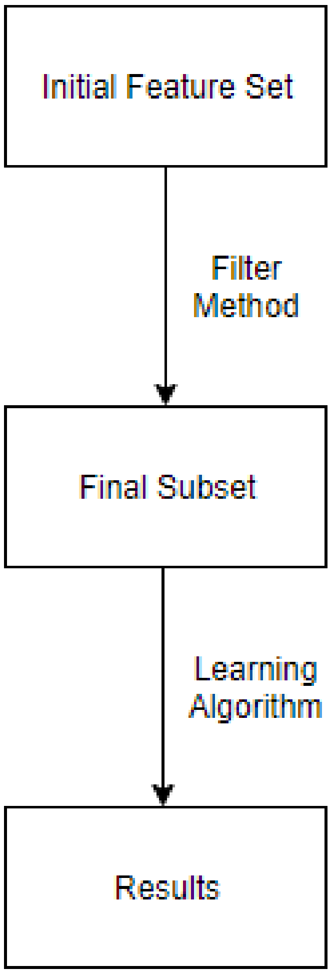


Figure **2.4.5**: A typical *Filter* FS procedure.

In the case of microarray data, Correlation-Based Feature Selection (Hall 1999a), Fast Correlation-Based Filter (Yu and Liu 2003), ReliefR (Kira and Rendell 1992) or the Consistency-Based Filter (Dash and Liu 2003) are some of the classical *Filter* algorithms that have been successfully implemented in the literature.

Most of the time, filters are easy to implement and manage to obtain good results in a short time. However, their major drawback is that while trying to reduce the dimensions (i.e., feature elimination), they might label a feature as redundant when in fact, combined with some other features, it might be proved informative. Since most cancer-related datasets have a relative small sample size (e.g., order of tens) and the sample can not be considered representative of the overall population, the risk of meeting the aforementioned scenario is greater. Moreover, *Filter* methods, as we later present, tend to return somewhat large feature subsets within the gene selection domain.

2.4.2 Wrapper Technique

Opposed to the *Filter*, the *Wrapper* method incorporates a search strategy (e.g., metaheuristics) that constantly selects some features, communicates the feature subset with a learning algorithm, retrieves a metric (e.g., accuracy) that usually is part of a score function and proceeds in obtaining a different subset that might result in a better metric value. The process stops when certain termination criteria are met. It becomes clear that the search process is tightly attached to the learning model. Figure **2.4.6** illustrates how a *Wrapper* method is applied:

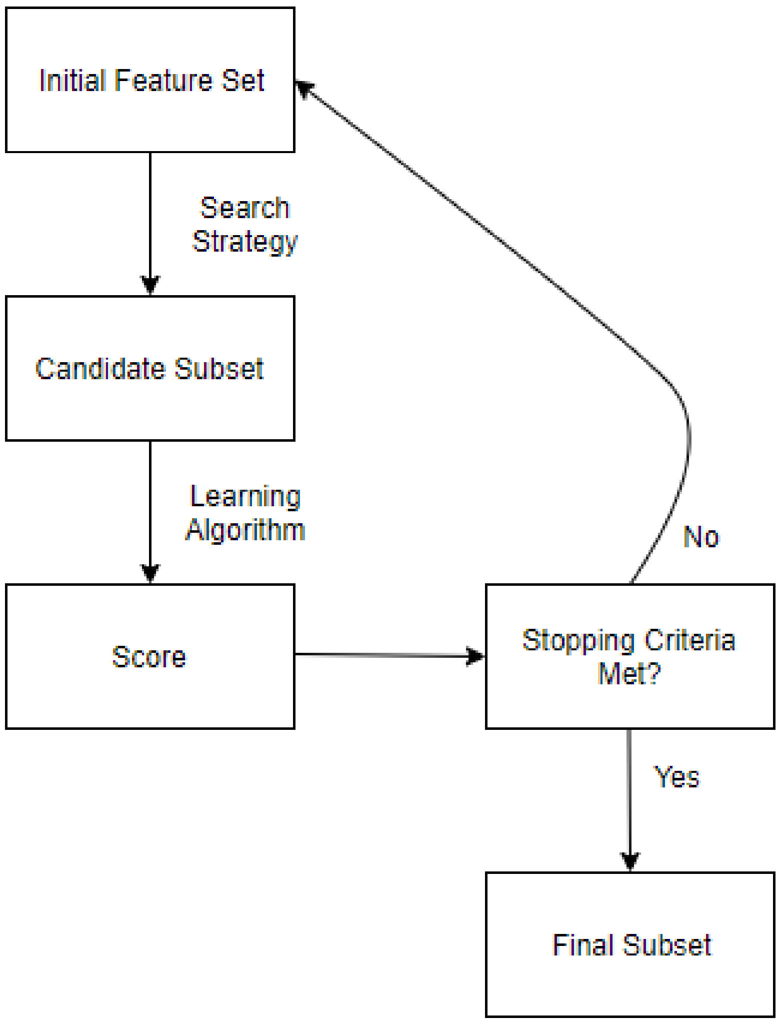


Figure **2.4.6**: The wrapper FS procedure.

The *Wrapper* technique has been given less attention compared to the *Filter* method. That is mainly due to the high computational effort needed and the high risk of overfitting. However, wrappers seem to obtain significantly less genes than popular filters in many cancer-related FS applications, e.g., (Alba, Garcia-Nieto, Jourdan, and Talbi 2007, Alshamlan, Badr, and Alohali 2015, Chuang, Yang, Wu, and Yang 2011). Thus, experts have to study the interaction of a reasonable number of genes, a fact that becomes more tricky, even impossible, from most *Filter* feature selectors' results. Finally, *Wrapper* methods do not label a-priory genes as redundant or relevant. In that way, each feature has a non-zero probability to participate in the final feature subset.

Popular mechanisms utilized to explore the solution space have typically been metaheuristics. Genetic Algorithms, Particle Swarm Optimization, Bee Colony Optimization and their variants have shown notable results within the FS problem. We reference the interested reader to (Alshamlan, Badr, and Alohali 2015, Alba, Garcia-Nieto, Jourdan, and Talbi 2007, Chuang, Yang, Wu, and Yang 2011).

2.4.3 Embedded Technique

Trying to balance the pros and cons of the aforementioned FS classes, the *Embedded* methods emerged. As stated in (Bolón-Canedo, Sánchez-Marono, Alonso-Betanzos, Benítez, and Herrera 2014), such methods use the core of the classifier to establish a criteria to rank features. However, *Embedded* methods are generally driven by heuristic approaches, thus leading to insufficient exploration of the solution space. We proceed in describing a particular method, appertaining to the said FS methods, that we later use within our methodology.

SVM - Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a FS method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are usually ranked by the model's coefficients, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model. In the case where the model is a Support Vector Machine classifier, RFE is typically referenced as SVM-RFE.

RFE requires a specified number of features to keep, however it is often not known in advance how many features are valid. To find the optimal number of features cross-validation is used with RFE to score different feature subsets and select the best scoring collection of features.

Figure **2.4.7** illustrates the procedure that the algorithm follows to reach to a final feature subset. In each iteration, SVM model's coefficients are checked and features with the smallest, by absolute value, ones are eliminated. Subset 1, thus, is a subset of the initial feature set and

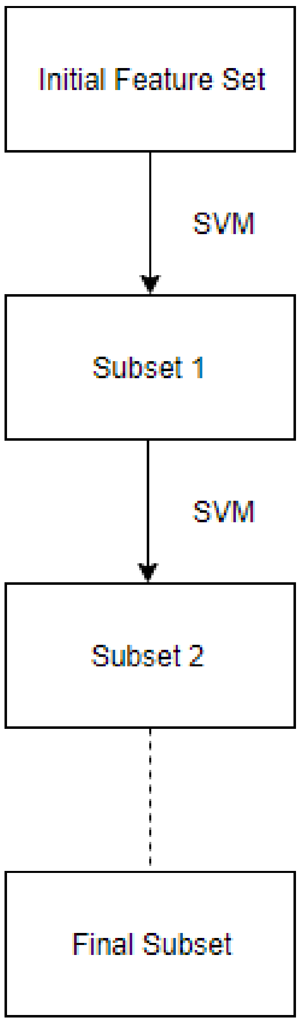


Figure **2.4.7**: The SVM-RFE method.

in general,

$$S_{i+1} \subset S_i, \forall i \in \{0, 1, ..., k-1\} \quad (1)$$

, where S_0 is the initial feaure set and S_k is the final feature subset. The pseudocode of SVM-RFE is given in Algorithm 3.

Algorithm 3: SVM-RFE pseudocode

```
Input: $X_0 = [x_1, x_2, ..., x_k, ..., x_l]^T$ ;                    // training examples
Input: $y = [y_1, y_2, ..., y_k, ..., y_l]^T$ ;                       // class labels
initialize subset of surviving features $s = [1, 2, ..., n]$;
initialize feature ranked list $r = []$;
while $s \neq \emptyset$ do
    $X = X_0(:, s)$;                              // restrict training examples
    $\alpha = \text{SVM-train}(X, y)$;                      // train the classifier
    $w = \sum_k \alpha_k y_k x_k$;                        // compute the weight vector
    $c_i = (w_i)^2, \forall i$;                       // compute the ranking criteria
    $f = argmin(c)$;             // find the feature with the smallest ranking
    $r = [s(f), r]$;                              // update feature ranked list
    $s = s(1 : f-1, f+1 : length(s))$;     // eliminate the feature with smallest
      ranking criterion
end
return $r$;
```

Since SVM-RFE takes the learning algorithm's parameters, e.g., coefficients, into consideration, it is classified as an *Embedded* FS method. In the thesis' context, SVM-RFE will only be used in order to reduce the feature space, i.e., as a pre-processing step, and not as the primary search strategy.

2.4.4 Ensemble Technique

The *Ensemble* method, by definition, involves the usage of a multitude of FS methods. Essentially, this technique tries to take advantage of many expert (FS methods) opinions, by combining their results. For instance, someone could apply two *Filters* and a *Wrapper* on the same dataset, and reach to a final feature subset by unifying the three individual feature subsets each method obtained.

As shown in Figure **2.4.8**, besides the selection of individual FS techniques, the *Ensemble* method needs a module we call *Subset Handling Level* that, actually, determines how to manipulate each outcome. We mentioned earlier that an obvious way would be to yield, as a final

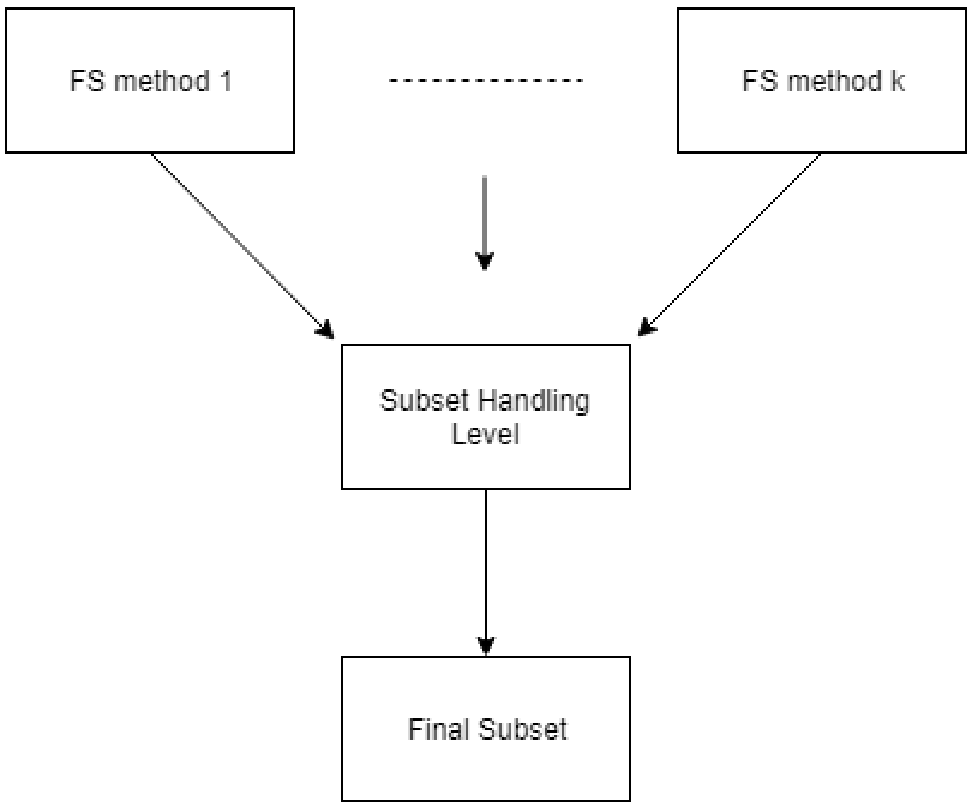


Figure **2.4.8**: The Ensemble FS method.

solution, the union of all subsets. However, another usual approach is to insert a unique *weight* attribute to each FS method. Consequently, the contribution of each method to the final subset build will not be equal. In their paper (Bolón-Canedo, Sánchez-Marono, Alonso-Betanzos, Benítez, and Herrera 2014), Bolano et. al. note that the *Ensemble* FS scheme is inspired from the ensemble of classifiers in the ML domain, e.g., the Random Forest Algorithm we introduced in subsection 2.3.3.

CHAPTER 3

Related Work

Before diving into our methodology's details, it would be quite valuable to examine how the FS problem was tackled by other researchers in the previous years. More specifically, in this chapter we focus on selected and competitive *Wrapper* and *Hybrid* FS techniques developed and published within our century. Among others, some aspects that will concern us are the search strategy used, the choice of learning algorithms that evaluated feature subsets, the final results and a metric that measures the complexity of each method, e.g., number of classifications, time, etc.. The related work that is quoted below follows an ascending chronological order and each section, related to a unique paper study, is named after the algorithms used.

3.1 Genetic Algorithm and Geometrical Particle Swarm Optimization (2007)

In (Alba, Garcia-Nieto, Jourdan, and Talbi 2007), authors implemented two *Wrapper* methods, namely a Genetic Algorithm (GA) and a Geometrical Particle Swarm Optimization (GPSO) to face the Gene Selection problem. These methods were tested on six well-known datasets issued of microarray experiments.

3.1.1 Overview

The basic scheme of how features are selected out from the original microarray dataset using a particle with binary encoding is illustrated in Figure **3.1.1**.

Some basic points we need to keep regarding **3.1.1** are that both GA and GPSO seek for gene subsets in the entire microarray initial dataset and they do not apply any kind of preprocessing steps in order to reduce the vast solution space in contrast with similar work. In addition to that, someone can notice that each gene subset shapes a new dataset with less genes and a Support Vector Machine (SVM) classifier along with a 10-fold cross validation is applied

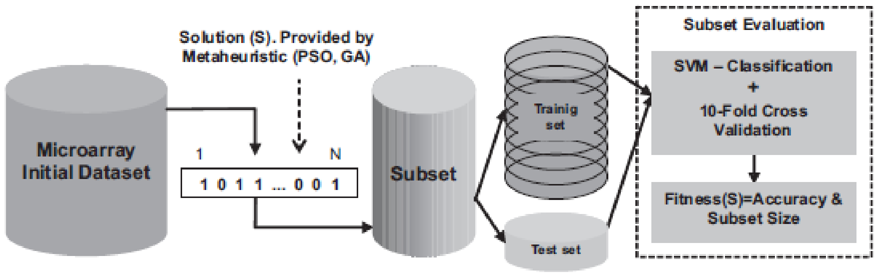

Solution (S). Provided by Metaheuristic (PSO, GA)
Microarray Initial Dataset
1
N
1 0 1 1 ... 0 0 1
Subset
Trainig set
Test set
Subset Evaluation
SVM – Classification + 10-Fold Cross Validation
Fitness(S)=Accuracy & Subset Size

Figure **3.1.1**: The methodology proposed in the discussed work

on it. Finally, worth to mention that the fitness function takes both SVM's accuracy and the subset size into consideration.

3.1.2 Parameters and Results

Researchers that conducted the discussed study have chosen the following parameters for their algorithms:

Table **3.1**: GPSO and GA parameters for gene subset selection and classification.

GPSO		**GA**	
Parameter	**Value**	**Parameter**	**Value**
Swarm size	40	Population size	40
Number of generations	100	Number of generations	100
Neighborhood size	20	Probability of crossover	0.9
Probability of mutation	0.1	Probability of mutation	0.1
(w1,w2,w3)	(0.33,0.33,0.44)	-	-

From a complexity point of view, the two algorithms evaluate $40 * 100 = 4,000$ solutions in order to converge, a number that can be considered large enough compared to our approach that will be later analyzed. Regarding the results obtained, both methods manage to yield remarkable solutions with high accuracy values and small gene subsets.

Datasets marked as bold in Table **3.2** were used on the current thesis as well.

3.2 Recursive Feature Elimination - MRMR (2010)

Studying the work in (Mundra and Rajapakse 2010), the reason of combining FS methods becomes more clear. In this research paper, authors suggest that relevant and redundant features can not be determined independent of the classifier. To this end, they incorporated both outcomes of two feature ranking methods; the Maximum Relevance Minimum Redundancy

Table **3.2**: Mean accuracy and average number of selected genes in parenthesis. Values obtained after 10 independent runs.

Dataset	**GPSOsvm**	**GAsvm**
Leukemia	97.83(3)	97.27(4)
Breast	86.35(4)	95.86(4)
Colon	100(2)	100(3)
Lung	99.00(4)	99.49(4)
Ovarian	99.44(4)	98.83(4)
Prostate	98.66(4)	98.65(4)

(MRMR) *Filter* technique and the Support Vector Machine - Recursive Feature Elimination (SVM-RFE) *Embedded* method.

3.2.1 Overview

As stated above, the novelty of the said research is that genes are ranked by a convex combination of the relevancy given by SVM weights and the MRMR criterion. More specifically, the ranking measure r_i is given by:

$$r_i = \beta * |w_i| + (1 - \beta) * \frac{R_{S,i}}{Q_{S,i}} \tag{1}$$

, where $\beta \in [0, 1]$ determines the trade-off between ranking terms $|w_i|$ and $\frac{R_{S,i}}{Q_{S,i}}$, i.e., expression of SVM and MRMR insights, respectively.

3.2.2 Parameters and Results

Concerning the parameters used, the β value was determined empirically from the set {0.2, 0.4, 0.6, 0.8} and SVM-RFE eliminated 100 genes per iteration if genes where equal to or greater than 10,000, 10 genes per iteration, if genes where less than 10,000 but more than 1,000 and 1 gene per iteration, if genes where less than or equal to 1,000. Next, Table **3.3** with results found follows.

It becomes obvious that the proposed methodology performs well in terms of accuracy. However, the number of genes that takes to yield such accuracy values can be characterized as large in comparison with other methods. Especially from an interpretation point of view, it is extremely difficult to analyze the association of tens of genes with a target variable.

Table **3.3**: Number of selected genes, mean accuracy and standard deviation in parenthesis with the SVM-RFE MRMR method. Values obtained after 100 independent runs.

	#Genes	Accuracy
Colon	78	91.68 (5.13)
Leukemia	37	98.35 (1.93)
Hepato	11	88.15 (4.59)
Prostate	10	98.29 (2.30)

3.3 Correlation Based Feature Selection - Taguchi-Genetic Algorithm (2011)

In (Chuang, Yang, Wu, and Yang 2011), L-Y. Chang et al. developed a hybrid feature selection method for DNA microarray data. Their idea was to apply the Correlation Based Feature Selection (CFS) *Filter* method as a pre-processing step so as to reduce the solution space. The search mechanism that would offer the final solutions was a Genetic Algorithm variant called Taguchi-Genetic Algorithm (TGA). This method was developed by Genichi Taguchi and, in short, as stated in the abstract section of (Tsai, Liu, and Chou 2004), the systematic reasoning ability of the Taguchi method is incorporated in the crossover operations to select the better genes to achieve crossover, and consequently, enhance the Genetic Algorithm.

3.3.1 Overview

Before quoting the basic schema of the method under discussion, it would be insightful to mention the simple, yet effective, idea behind CFS technique which relies on the hypothesis that *good feature subsets contain features highly correlated with the class, yet uncorrelated with each other* (Hall 1999b). Mathematically, this idea is supported by:

$$Merit_S = \frac{k * \gamma_{cf}}{\sqrt{k + k(k-1) * \gamma_{ff}}} \tag{2}$$

, with $Merit_S$ being the merit of a feature subset S containing k features, γ_{cf} being the average feature-class correlation and, finally, γ_{ff} denoting the average feature-feature inter-correlation $(f \in S)$. Figure **3.3.2** depicts the steps of the procedure in more detail.

It is worth to note that authors used the kNN algorithm (with k=1) and the Leave One Out Cross Validation (LOOCV) technique to evaluate the fitness of each candidate solution.

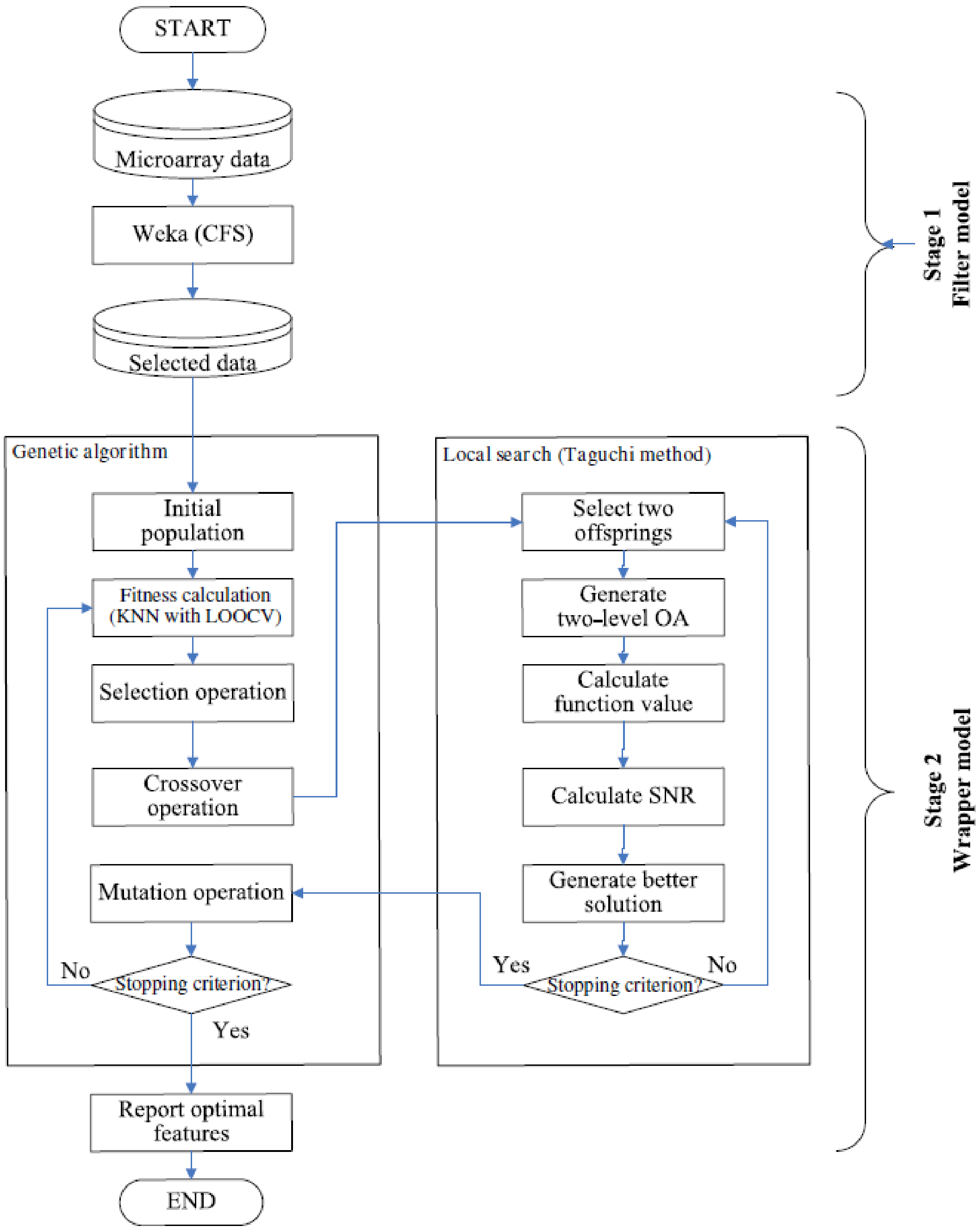


Figure **3.3.2**: The CFS-TGA method.

3.3.2 Parameters and Results

Someone can affirm that the proposed algorithm evaluates the fitness of $30 * 60 = 1,800$ candidate solutions, as shown in table **3.4**.

Table **3.4**: The parameters of the Taguchi-Genetic Algorithm.

Generations	Population size	Probability of crossover/mutation
30	60	1.0/0.01

Table **3.5**: Results obtained after ten independent runs.

Dataset	**Accuracy(std.dev.)**	**#Genes(std.dev.)**
9_tumors	90.50(0.81)	24.6(2.55)
11_tumors	100(0.00)	137.5(4.35)
14_tumors	74.39(0.67)	53.7(4.03)
Brain_Tumor1	99.45(0.59)	44.7(3.68)
Brain_Tumor2	100(0.00)	33.1(2.64)
Leukemia1	100(0.00)	22.4(1.78)
Leukemia2	100(0.00)	35.9(2.92)
Lung_Cancer	98.42(0.31)	195.2(5.41)
SRBCT	100(0.00)	29.0(2.40)
Prostate_Tumor	99.22(0.41)	24.7(1.42)
DLBCL	100(0.00)	17.1(2.33)

Even from a glimpse upon figures in **3.5** it is apparent that CFS-TGA performs really well in almost all datasets, although the number of selected genes is still large enough.

3.4 Genetic Bee Colony Optimization (2015)

A more recent study on the FS field is the one in (Alshamlan, Badr, and Alohali 2015). In a nutshell, H.M. Alshamlan et al. handled the gene selection problem by combining the MRMR *Filter* method with a Genetic Bee Colony (GBC) optimization *Wrapper* technique.

3.4.1 Overview

In this work, authors mention that in high-dimensional microarray datasets, due to the existence of a set of several thousands of genes, it is inefficient to adopt evolutionary algorithms directly to the microarray dataset. Therefore, they add a pre-processing step, i.e., the MRMR algorithm, as shown in **3.4.3**. Afterwards, they apply their search mechanism into a reduced solution space. The reduction is determined by a subset of the initial genes that yielded 100%

accuracy with the MRMR method, with the subset usually comprising of a few hundreds of genes.

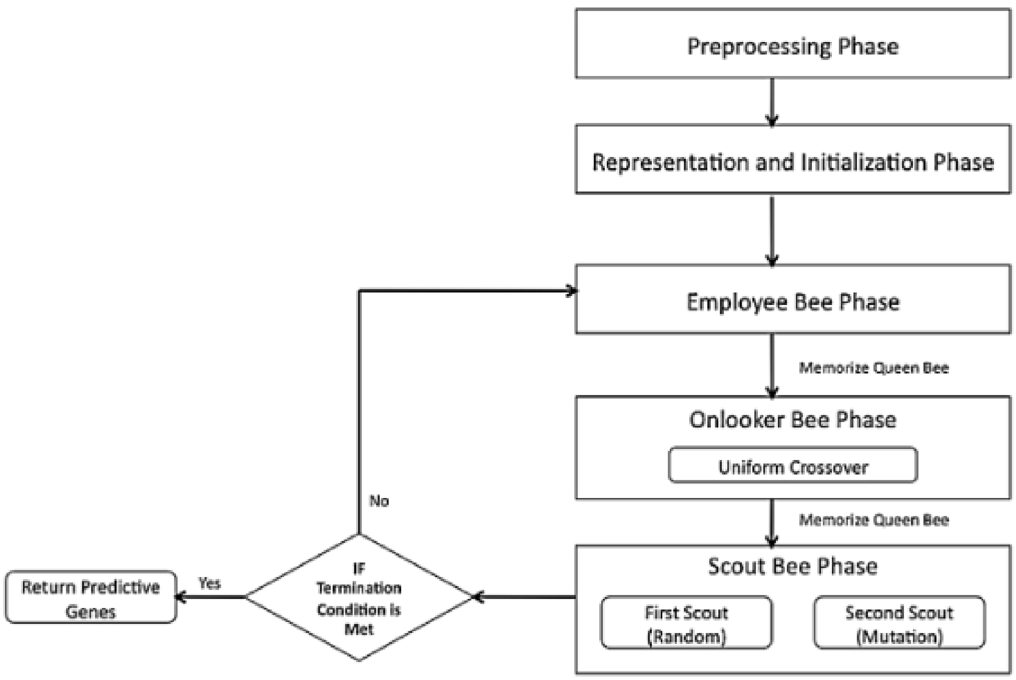


Figure **3.4.3**: The main phases of the GBC algorithm.

3.4.2 Parameters and Results

The fundamental parameters of the GBC algorithm are presented in table **3.7**. The *Colony size*, i.e., population, and the *Max cycle*, i.e., generations, indicate that the algorithm requires at least 8,000 classification trials. Worth noticing that the true number of classifications is way bigger, if someone thoroughly examines **Algorithm 1** in (Alshamlan, Badr, and Alohali 2015).

Table **3.6**: Control parameters for GBC.

Parameter	**Value**
Colony Size	80
Max cycle	100
Number of run	30
Limit	5
Crossover/Mutation propriety level	0.8/0.01

As mentioned in the introduction of this chapter, the choice of the learning algorithm used in each work is something that concern us. In the discussed study, researchers selected the SVM classifier which obtained the accuracy that follows in Table **3.7**.

Table **3.7**: **Best** accuracy obtained by gene subsets found by GBC. Numbers in parenthesis denote the number of genes.

Datasets	**Colon**	**Leukemia1**	Lung	SRBCT	Lymphoma	Leukemia2
Accuracy	98.38(10)	100(4)	100(4)	100(6)	100(4)	100(8)

In general, the GBC algorithm performs well in both terms of accuracy and number of selected genes. Worth noticing that in this paper someone can find a variety of final results, depending mainly on different gene subset sizes. A typical trend that we have noticed is that larger gene subset sizes yielded more accurate results, which was something anticipated in the problem that we discuss.

The caption of table **3.7** witnesses that values from the same table are the **best** ones. In the comparison section of Chapter 5, we mark the mean accuracy of the GBC algorithm with gene subset sizes relatively close to ours. Datasets in bold, which are the Colon and the Leukemia1 datasets, are the ones that we also used in our study and can compare upon.

3.5 Binary Particle Swarm Optimization - Backward Generation (2019)

One of the most recent studies within the field is (Bir-Jmel, Douiri, and Elbernoussi 2019), where authors present two hybrid approaches for the gene selection in DNA microarray data. The two proposed approaches consist of a pre-selection phase carried out using the Fisher and ReliefF filter methods for FBPSO-SVM (Binary Particle Swarm Optimization combined with Fisher algorithm and Support Vector Machine) and RBPSO-1NN (Particle Swarm Optimization combined with ReliefF algorithm and 1-Nearest Neighbors) respectively, and a search phase that determines a good subset of genes for the classification. The latter is based on two meta-heuristics, BPSO and a local search method (Backward Generation). These approaches aim to select an optimal subset of relevant genes from a dataset which contains redundant, noisy or irrelevant data.

3.5.1 Overview

Figure **3.5.4** presents the process that Bir-Jmel et al. follow to conclude to a final gene subset. Step 1, characterized as the filtration module, applies a *Filter* technique to the initial dataset that yields a new gene dataset consisting of 300 genes. In step 2, the BPSO searches for informative gene subsets into the new, smaller, dataset. Finally, a heuristic criterion named Backward Generation is responsible for eliminating genes that do not worsen the learning algorithm's accuracy. This sums up the entire flow that researchers came up with.

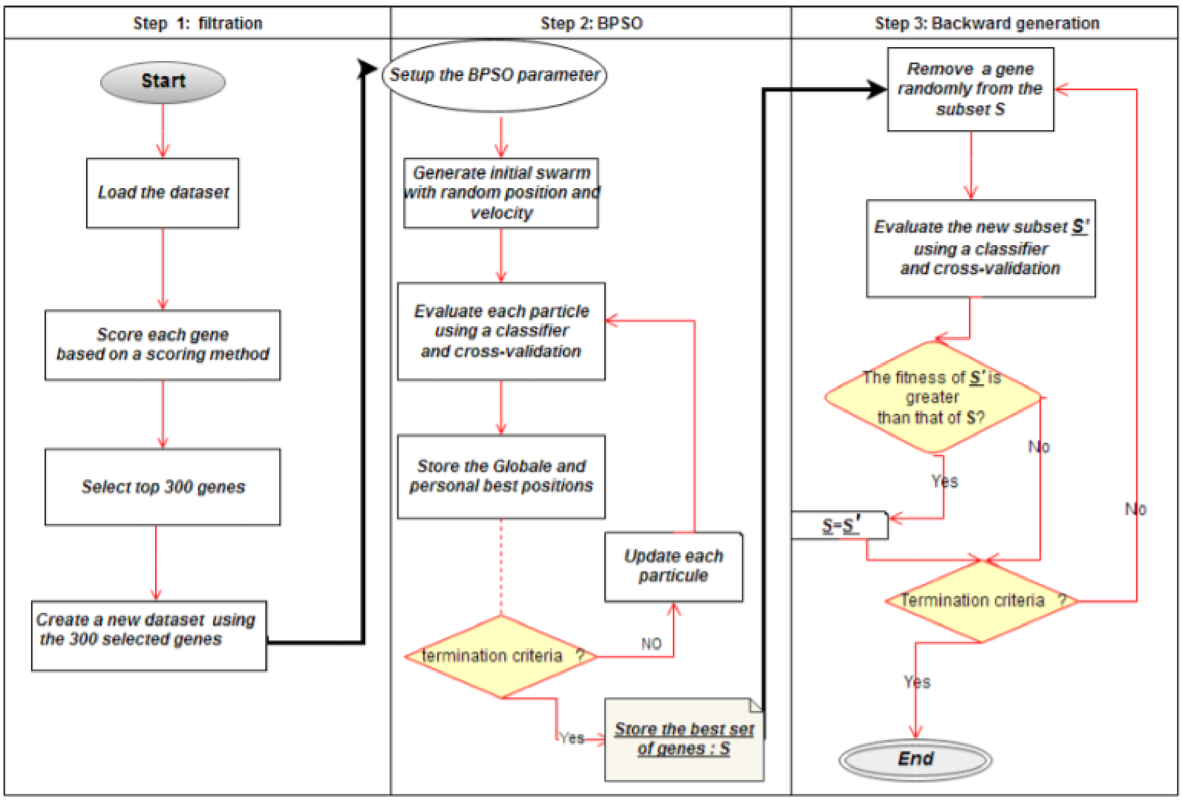

Step 1: filtration
Step 2: BPSO
Step 3: Backward generation
Start
Load the dataset
Score each gene based on a scoring method
Select top 300 genes
Create a new dataset using the 300 selected genes
Setup the BPSO parameter
Generate initial swarm with random position and velocity
Evaluate each particle using a classifier and cross-validation
Store the Globale and personal best positions
Update each particule
termination criteria ?
NO
Yes
Store the best set of genes : S
Remove a gene randomly from the subset S
Evaluate the new subset S' using a classifier and cross-validation
The fitness of S' is greater than that of S?
No
Yes
S=S'
Termination criteria ?
No
Yes
End

Figure **3.5.4**: The BPSO - Backward Generation process.

3.5.2 Parameters and Results

As far as the parameters are concerned, and without getting too detailed, the swarm size was set to 30 and the numbers of generations was set to 100, which leads to 3,000 classification attempts. In addition to that, another 200 iterations will take for Backward Generation to yield the final gene subset.

Table **3.8**: Control parameters for BPSO.

Parameters	**Value**
Swarm size	30
Generations	100
c_1, c_2	2
w	1
v_{max}	4
v_{min}	-4
Backward generation, it_{max}	200

Result-wise, the Table **3.9** witnesses acceptable results in terms of accuracy, although the time needed for the process to complete is considered too long and the gene subset sizes, in some cases, considerably large.

Table **3.9**: Accuracy, number of genes and execution time of BPSO-based methods.

	RBPSO-1NN			**FBPSO-SVM**		
Dataset	**Accuracy(%)**	**Genes**	**Time(h)**	**Accuracy(%)**	**Genes**	**Time(h)**
9_tumors	81.83	29.1	0.69	92.99	45	4.22
Brain_Tumor1	94.00	24.7	1.48	97.22	22.4	4.20
Brain_Tumor2	92.80	24.5	0.52	100	14.3	1.91
Leukemia1	99.72	11.7	0.76	100	8.4	1.60
Leukemia2	100	13.1	0.84	100	8.6	1.99
SRBCT	100	11.7	0.85	100	12.4	3.02
Prostate_Tumor	98.24	11.2	1.01	100	8.3	3.46
BLBCL	100	12.5	1.30	100	6.7	1.49

Unfortunately, we did not test our methods in any of the above datasets, so a direct comparison is impossible at this point.

3.6 Summary and thoughts

A brief overview on some recent papers regarding the FS problem indicates a strong trend; combination of FS techniques, for instance *Filters* and *Wrappers*, is extensively under study, producing good results. *Filters* are usually the first ones applied in order to reduce the vast solution space, and *Wrappers* search later into that smaller space leveraging their mechanisms for not getting trapped into local optima.

Concerning the tools and parameters used, most FS schemes have selected SVM as their classifier, with k-NN also being a choice for some studies. In most cases, the performance of the respective classifier was satisfying, with 100% accuracy being not a surprise. However, in terms of final gene subset sizes and classifications required, the results are not so encouraging. In many research papers, despite authors avoid to explicitly mention the time that their algorithms run, it was easy to extract the thousands of required classifications that would lead to a not so manageable gene subset size.

CHAPTER 4

Research Methodology

The purpose of this work is to propose an efficient search mechanism for gene selection in cancer classification, that limits the drawbacks of wrapper FS techniques, i.e., the risk of model overfitting and the high computational cost, while manages to obtain accurate results. To achieve that, we implement an RVNS algorithm that searches the solution space in a systematic, yet computationally light, manner. Solutions provided by RVNS are shared across SVM, k-NN and RF classifiers for evaluation and their average accuracy, along with the solution's number of selected genes, are taken into consideration from an evaluation function. As a result, the final gene subsets obtained by our algorithm yield accurate predictions for more than one learning algorithms and findings can be further used with more reliability.

In an attempt of enhancing the RVNS's performance, we apply the RFE heuristic approach as a pre-processing step. In that way, a significant number of possibly redundant genes are eliminated and the resulted solution space is given to RVNS to search into. Therefore, a new hybrid search strategy is formed that we will refer to as RFE-RVNS. Figure **4.0.1** depicts the aforementioned process.

4.1 Parameter Settings

4.1.1 Solution Representation

Each candidate solution s is conceived as a binary, 1-dimensional array of length N, with N denoting the number of genes in each dataset. For instance, a candidate solution in a dataset with 5 genes can be:

$$s = [0, 1, 1, 0, 1]$$

, which means that the second, the third and the fifth genes of the dataset are selected while the first and the fourth are not.

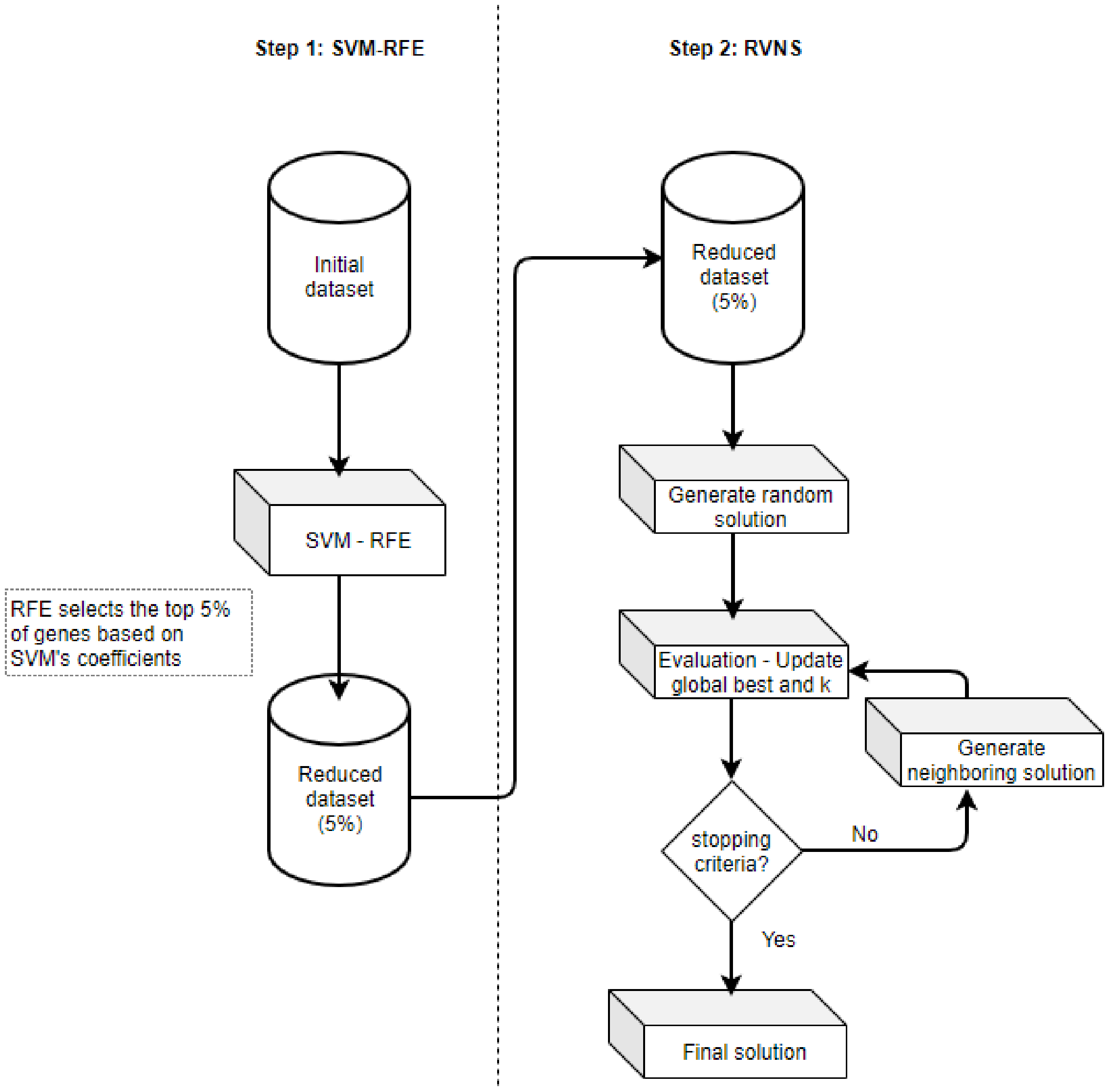


Figure **4.0.1**: The RFE-RVNS flowchart. The value of k indicates the neighborhood structure the algorithm is searching into.

4.1.2 RVNS

A substantial step when implementing a VNS algorithm is the neighborhood structures' definition. In both RFE-RVNS and RVNS, we choose the three following neighborhood structures:

1. *Replace a selected gene of the incumbent solution with an un-selected one.*

2. *Replace two selected genes of the incumbent solution with an un-selected one. If the incumbent solution has only one gene selected, return the incumbent solution.*

3. *Add an un-selected gene to the incumbent solution. If there are no more genes to add, return the incumbent solution.*

The neighborhood order, which is also decisive, is as indicated above. Worth noticing that, in all experiments, the initial solution is generated arbitrarily with two randomly selected genes. Therefore, according to the neighborhood definitions above, no exception-handling is required for the case of zero selected genes.

Example 3.1 Assuming there is a microarray dataset with 5 genes and s the incumbent solution, where $s = [0, 1, 1, 0, 1]$. Let us denote with $N_i(s), i \in \{1, 2, 3\}$, the sets of neighboring solutions of s. Considering the three neighborhood structures as defined above, $s_1 = [0, 1, 1, 1, 0] \in N_1(s)$, $s_2 = [1, 0, 1, 0, 0] \in N_2(s)$ and $s_3 = [1, 1, 1, 0, 1] \in N_3(s)$.

<u>Claim:</u> Given a random solution s and the neighborhood structures $N_i, i \in \{1, 2, 3\}$, both as defined in 4.1.2, then $N_1(s) \cap N_2(s) \cap N_3(s) = \emptyset$, $\forall s \in S$, with S being the solution space.

<u>Proof:</u> Let us consider a dataset with N features and s a random solution. Then, s will have k 1s, $1 \leq k \leq N, k \in \{1, ..., N\}$, and $N - k$ 0s. Moreover, we denote with s_i whichever solution within the neighborhood N_i(s). In the case where $1 < k < N$, s_1 will count k 1s and $N - k$ 0s, s_2 will count $k - 1$ 1s and $N - k + 1$ 0s and s_3 will count $k + 1$ 1s and $N - k - 1$ 0s. Therefore, $s_i, i \in \{1, 2, 3\}$ will be different per two. Consequently, $N_1(s) \cap N_2(s) \cap N_3(s) = \emptyset$. Similar approach proves our claim for the cases where $k = 1$ and $k = N$. ■

The aforementioned claim ensures that search within different neighborhood structures will lead to evaluation of different solutions. Therefore, our algorithms hold the capability of escaping from local optima.

4.1.3 Learning Algorithms

Core parameters of the learning algorithms are empirically selected with two ends in mind; the accuracy performance and the computational efficiency. Seeking for a balance between these two, we tested k-NN with k in {1,3,5,7,9}. Additionally, various RF implementations, with number of DT's in {10,20,30,40,50} and depth of pruning value in {10,15,20}, helped us proceed to our final choice.

The number of neighboring classes that k-NN takes into consideration before classifying an unknown patient is set to 5 and the RF classifier predicts class labels consulting with 20 10-depth pruned decision trees. Moreover, the SVM classifier is implemented with a linear kernel, meaning that it searches for the best linear hyper-plane that is able to discriminate the data. The accuracy obtained from each learning algorithm is averaged after a 10-fold cross-validation.

4.1.4 RFE

The RFE heuristic is applied with an SVM classifier. In each dataset, the SVM-RFE method eliminates 95% of the genes that are considered irrelevant, with a 10% step. Worth mentioning that, in a typical RFE execution, such a step is considered quite large. In an attempt of maintaining the computational complexity low, and since RFE is not the primary search method, we apply it with the aforementioned parameters.

4.2 Evaluation Function

Each candidate solution is evaluated by four metrics; the accuracy of the three classifiers and the cardinality of the incumbent gene subset. Consequently, the evaluation function is a multi-objective one and is described below:

$$F(s) = \alpha * \underbrace{\frac{3}{a_1(s) + a_2(s) + a_3(s)}}_{\text{a(s)}} + (1 - \alpha) * g(s) \tag{1}$$

where $a_1(s)$, $a_2(s)$, and $a_3(s)$ denote the accuracy the SVM, k-NN and RF classifiers yield from solution s, respectively, and $g(s)$ indicates the number of selected genes. Assuming each predictive model performs at least as good as a random classification, $a_i(s) \in [0.5, 1.0], \forall i \in \{1, 2, 3\}$ (binary classification), thus $\frac{3}{\sum_{i=1,3} a_i(s)} \in [1, 2]$, while $g(s)$ is a positive integer restricted by the number of genes in each dataset. Parameter α acts as weight to the average accuracy of the three classifiers, while $1 - \alpha$ acts similarly to the gene subset size. Figure **4.2.2** illustrates the process of evaluating a solution.

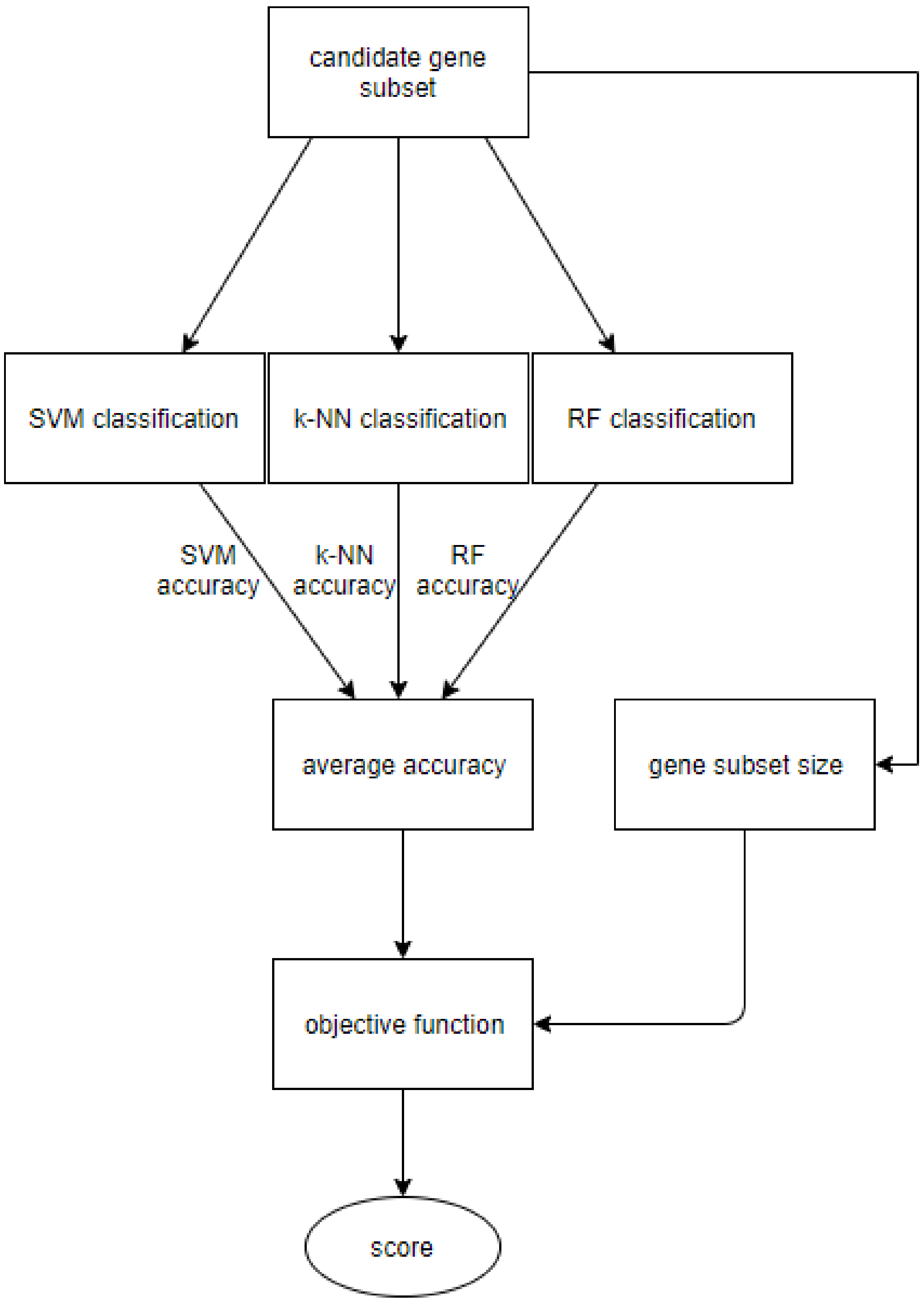


Figure **4.2.2**: The process of evaluating a candidate solution.

The evaluation function in Equation 1 is selected since it can offer a good trade-off between the overall accuracy and the final number of selected genes. Experimentation led us to setting α to 0.99; a value that, is consistent both with our objective of finding informative gene subsets and with the co-domains of $\frac{3}{a(s)}$ and $g(s)$ in Equation 1.

A similar fitness function is used in (Alba, Garcia-Nieto, Jourdan, and Talbi 2007) and manages to balance accurate predictions and small gene subsets, although with different weight values and using only the accuracy of a single learning algorithm.

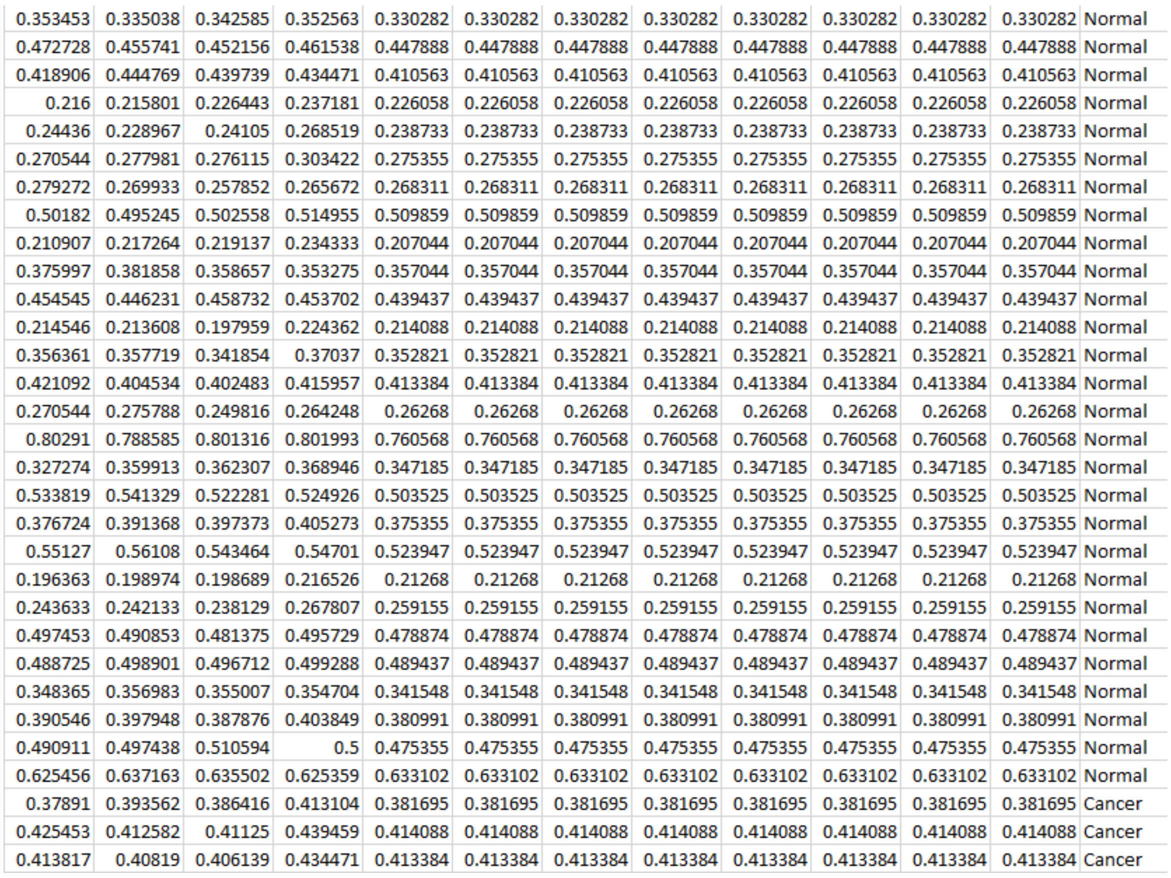
4.3 Data Description and Preprocessing

The proposed methodology is tested on five publicly available cancer-related datasets; the Leukemia (Golub, Slonim, Tamayo, Huard, Gaasenbeek, Mesirov, Coller, Loh, Downing, Caligiuri, et al. 1999), Lung (Gordon, Jensen, Hsiao, Gullans, Blumenstock, Ramaswamy, Richards, Sugarbaker, and Bueno 2002), Ovarian (Petricoin III, Ardekani, Hitt, Levine, Fusaro, Steinberg, Mills, Simone, Fishman, Kohn, et al. 2002), Colon (Alon, Barkai, Notterman, Gish, Ybarra, Mack, and Levine 1999) and Breast (CGAN et al. 2012) cancer datasets. The first four were originally taken from the public Kent Ridge Bio-medical Data Repository and hosted in ELVIRA Biomedical Data Repository with url `http://leo.ugr.es/elvira/DBCRepository/`, while the Breast Cancer Dataset was found under `https://data.mendeley.com/datasets/v3cc2p38hb/1`. Sample size, dimensionality and the number of classes of each dataset are depicted in Table **4.1**.

Table **4.1**: Dataset characteristics

Dataset	Sample size	Number of genes	Number of classes
Leukemia	72	7,129	2
Lung	181	12,533	2
Ovarian	253	15,154	2
Colon	62	2,000	2
Breast	590	17,814	2

Figure **4.3.3** gives an insight of what kind of data values our method deals with, illustrating the tail of the Ovarian dataset. Each row represents a single patient's attribute values, while each column is a different gene. The last column, though, is the class column and, essentially represents the value our learning algorithms want to predict. All data values are normalized and missing ones are replaced by zeros, i.e., their mean. It should be noticed that the only dataset which suffered from a few missing values (less than ten) was the Breast cancer one. Worth to note that, the normalization technique we use does not affect the already normalized datasets like the ones in Figure **4.3.3**.



0.353453	0.335038	0.342585	0.352563	0.330282	0.330282	0.330282	0.330282	0.330282	0.330282	0.330282	0.330282	Normal
0.472728	0.455741	0.452156	0.461538	0.447888	0.447888	0.447888	0.447888	0.447888	0.447888	0.447888	0.447888	Normal
0.418906	0.444769	0.439739	0.434471	0.410563	0.410563	0.410563	0.410563	0.410563	0.410563	0.410563	0.410563	Normal
0.216	0.215801	0.226443	0.237181	0.226058	0.226058	0.226058	0.226058	0.226058	0.226058	0.226058	0.226058	Normal
0.24436	0.228967	0.24105	0.268519	0.238733	0.238733	0.238733	0.238733	0.238733	0.238733	0.238733	0.238733	Normal
0.270544	0.277981	0.276115	0.303422	0.275355	0.275355	0.275355	0.275355	0.275355	0.275355	0.275355	0.275355	Normal
0.279272	0.269933	0.257852	0.265672	0.268311	0.268311	0.268311	0.268311	0.268311	0.268311	0.268311	0.268311	Normal
0.50182	0.495245	0.502558	0.514955	0.509859	0.509859	0.509859	0.509859	0.509859	0.509859	0.509859	0.509859	Normal
0.210907	0.217264	0.219137	0.234333	0.207044	0.207044	0.207044	0.207044	0.207044	0.207044	0.207044	0.207044	Normal
0.375997	0.381858	0.358657	0.353275	0.357044	0.357044	0.357044	0.357044	0.357044	0.357044	0.357044	0.357044	Normal
0.454545	0.446231	0.458732	0.453702	0.439437	0.439437	0.439437	0.439437	0.439437	0.439437	0.439437	0.439437	Normal
0.214546	0.213608	0.197959	0.224362	0.214088	0.214088	0.214088	0.214088	0.214088	0.214088	0.214088	0.214088	Normal
0.356361	0.357719	0.341854	0.37037	0.352821	0.352821	0.352821	0.352821	0.352821	0.352821	0.352821	0.352821	Normal
0.421092	0.404534	0.402483	0.415957	0.413384	0.413384	0.413384	0.413384	0.413384	0.413384	0.413384	0.413384	Normal
0.270544	0.275788	0.249816	0.264248	0.26268	0.26268	0.26268	0.26268	0.26268	0.26268	0.26268	0.26268	Normal
0.80291	0.788585	0.801316	0.801993	0.760568	0.760568	0.760568	0.760568	0.760568	0.760568	0.760568	0.760568	Normal
0.327274	0.359913	0.362307	0.368946	0.347185	0.347185	0.347185	0.347185	0.347185	0.347185	0.347185	0.347185	Normal
0.533819	0.541329	0.522281	0.524926	0.503525	0.503525	0.503525	0.503525	0.503525	0.503525	0.503525	0.503525	Normal
0.376724	0.391368	0.397373	0.405273	0.375355	0.375355	0.375355	0.375355	0.375355	0.375355	0.375355	0.375355	Normal
0.55127	0.56108	0.543464	0.54701	0.523947	0.523947	0.523947	0.523947	0.523947	0.523947	0.523947	0.523947	Normal
0.196363	0.198974	0.198689	0.216526	0.21268	0.21268	0.21268	0.21268	0.21268	0.21268	0.21268	0.21268	Normal
0.243633	0.242133	0.238129	0.267807	0.259155	0.259155	0.259155	0.259155	0.259155	0.259155	0.259155	0.259155	Normal
0.497453	0.490853	0.481375	0.495729	0.478874	0.478874	0.478874	0.478874	0.478874	0.478874	0.478874	0.478874	Normal
0.488725	0.498901	0.496712	0.499288	0.489437	0.489437	0.489437	0.489437	0.489437	0.489437	0.489437	0.489437	Normal
0.348365	0.356983	0.355007	0.354704	0.341548	0.341548	0.341548	0.341548	0.341548	0.341548	0.341548	0.341548	Normal
0.390546	0.397948	0.387876	0.403849	0.380991	0.380991	0.380991	0.380991	0.380991	0.380991	0.380991	0.380991	Normal
0.490911	0.497438	0.510594	0.5	0.475355	0.475355	0.475355	0.475355	0.475355	0.475355	0.475355	0.475355	Normal
0.625456	0.637163	0.635502	0.625359	0.633102	0.633102	0.633102	0.633102	0.633102	0.633102	0.633102	0.633102	Normal
0.37891	0.393562	0.386416	0.413104	0.381695	0.381695	0.381695	0.381695	0.381695	0.381695	0.381695	0.381695	Cancer
0.425453	0.412582	0.41125	0.439459	0.414088	0.414088	0.414088	0.414088	0.414088	0.414088	0.414088	0.414088	Cancer
0.413817	0.40819	0.406139	0.434471	0.413384	0.413384	0.413384	0.413384	0.413384	0.413384	0.413384	0.413384	Cancer

Figure **4.3.3**: A segment of the Ovarian dataset.

4.4 Materials

All learning algorithms mentioned, the RFE heuristic, as well as the data normalization leveraged the scikit-learn library's (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, and Duchesnay 2011), i.e., a Python's library developed for data science purposes, built-in features.

CHAPTER 5

Results and Comparison

5.1 Performance of RFE-RVNS and RVNS

Tables **5.1** through **5.5** depict the performance of the proposed algorithms on five publicly available well-known cancer-related datasets. The metrics measured are the *Best*, *Mean* and *Worst* values of the classifiers' accuracy, along with the respective *#Genes*, i.e., number of genes, values. The *Average accuracy* metric should not be interpreted as a typical learning algorithm's accuracy, rather the ability of our algorithms to obtain informative genes for all classifiers.

Table **5.1**: The performance of RFE - RVNS and RVNS algorithms when applied with the SVM, kNN and RF classifiers on the Leukemia dataset after ten independent runs.

Metrics	RFE-RVNS			RVNS		
	Best	Mean	Worst	Best	Mean	Worst
Average accuracy	99.58	98.88	97.64	97.44	94.26	89.35
SVM accuracy	100	98.75	94.58	100	95.18	86.67
k-NN accuracy	100	99.58	97.08	97.5	93.84	87.56
RF accuracy	100	98.32	97.08	97.5	93.76	87.2
#Genes	3	3.8	5	2	4.7	8

Table **5.2**: The performance of RFE - RVNS and RVNS algorithms when applied with the SVM, kNN and RF classifiers on the Lung cancer dataset after ten independent runs.

Metrics	RFE-RVNS			RVNS		
	Best	Mean	Worst	Best	Mean	Worst
Average accuracy	99.44	98.65	97.22	98.55	95.67	91.88
SVM accuracy	99.44	98.73	97.22	98.36	95.75	91.17
k-NN accuracy	100	98.67	97.22	98.36	95.19	91.14
RF accuracy	100	98.56	97.22	98.92	96.07	93.33
#Genes	2	2.2	3	2	2.5	3

Table **5.3**: The performance of RFE - RVNS and RVNS algorithms when applied with the SVM, kNN and RF classifiers on the Ovarian cancer dataset after ten independent runs.

Metrics	RFE-RVNS			RVNS		
	Best	Mean	Worst	Best	Mean	Worst
Average accuracy	100	99.18	97.87	98.67	96.94	94.77
SVM accuracy	100	99.49	97.62	99.6	98.06	95.63
k-NN accuracy	100	99.21	98	98.4	97.04	94.52
RF accuracy	100	98.85	97.6	95.74	95.74	91.39
#Genes	2	2.4	3	3	4.1	7

Table **5.4**: The performance of RFE - RVNS and RVNS algorithms when applied with the SVM, kNN and RF classifiers on the Colon cancer dataset after ten independent runs.

Metrics	RFE-RVNS			RVNS		
	Best	Mean	Worst	Best	Mean	Worst
Average accuracy	93.73	91.23	88.57	89.68	87.72	85.24
SVM accuracy	96.90	93.36	88.57	93.33	89.50	84.05
k-NN accuracy	96.90	92.69	89.05	90.00	87.79	84.05
RF accuracy	90.24	87.64	84.05	88.57	85.88	80.48
#Genes	4	5.5	8	3	5.5	8

Table **5.5**: The performance of RFE - RVNS and RVNS algorithms when applied with the SVM, kNN and RF classifiers on the Breast cancer dataset after ten independent runs.

Metrics	RFE-RVNS			RVNS		
	Best	Mean	Worst	Best	Mean	Worst
Average accuracy	99.27	98.83	98.37	99.1	98.35	97.06
SVM accuracy	99.32	98.83	98.47	99.32	98.39	96.95
k-NN accuracy	99.32	98.97	98.31	99.32	98.39	97.12
RF accuracy	99.32	98.7	98.14	99.16	98.29	97.12
#Genes	1	1.7	2	2	2.5	3

Commenting upon figures in Tables **5.1**, **5.2**, **5.3** and **5.5**, RFE-RVNS managed to obtain a maximum, i.e., the maximum of *Bests*, of 100% accuracy and a minimum, i.e., the minimum of *Worst*, of 97.22%, while the corresponding figures for RVNS are 99.10% and 89.35%. In the case of the Colon dataset, both RFE-RVNS and RVNS faced some adversities in finding small and informative gene subsets with *Mean* accuracy 91.23% and 87.22%, respectively.

Concerning the gene subset size, the *Mean* number of selected genes is impressively small under both approaches, with 3.8-gene and 4.7-gene subsets being the largest average ones for RFE-RVNS and RVNS, respectively. Again, in the Colon dataset, the behavior differs a little with slightly larger gene subsets.

While both methods perform worthy, not only in terms of yielding informative, to all classifiers, gene subsets, but also small sized ones, the dominance of RFE-RVNS over RVNS cannot be overlooked.

Questioning whether one learning algorithm is favored over the others, or whether their predictive ability significantly varies, Figure **5.1.1** shows that only in the case of the Colon cancer dataset, the RF model performs somewhat worst than the other two.

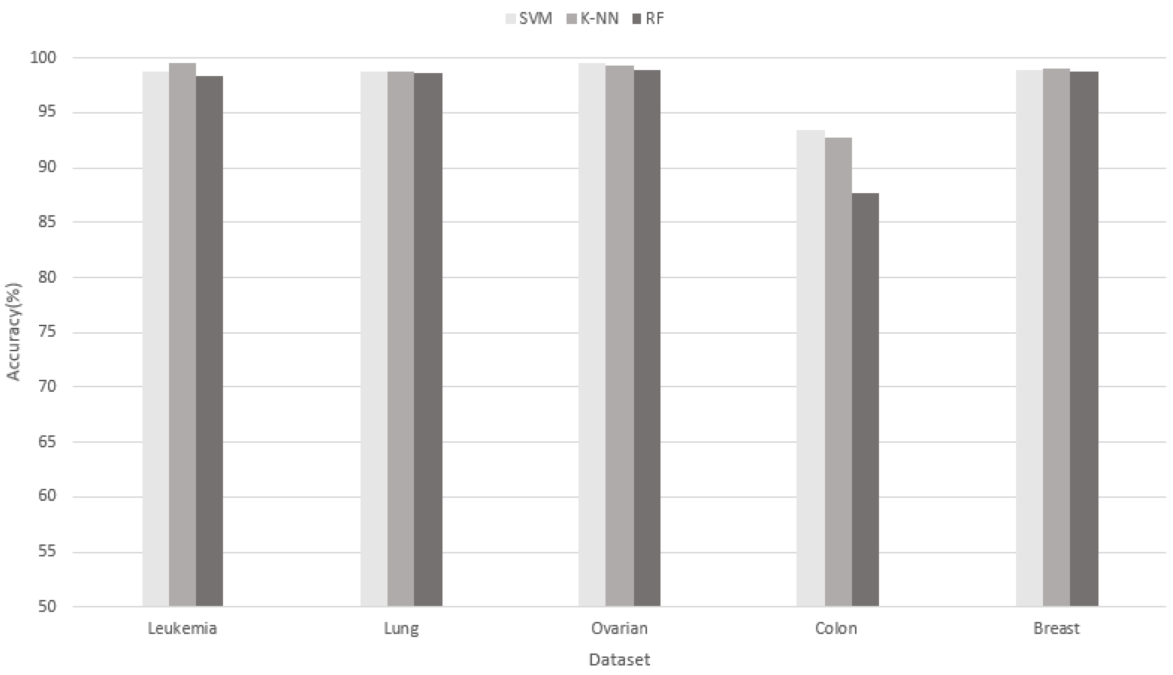


Figure **5.1.1**: SVM, *k*NN and RF mean accuracy in each dataset with RFE-RVNS.

In an attempt of deciphering our method's behavior on the Colon dataset, we illustrated the classifiers' accuracy on current best solutions found every 15 iterations of a typical RVNS execution. Graphs observed in Figure **5.1.2** indicate that gene subsets obtained often improve one learning algorithm's performance but worsen another, e.g., iterations 40 and 145. That phenomenon adds to our intention of implementing a gene selection strategy that returns informative gene subsets for more than one classifiers, in the sense of quality and reliability.

Elaborating on our latter statement, we expect that an expert would prefer to study a gene subset that is considered informative for more than one different learning classifiers, rather a single one. The fact that SVM, kNN and RF do not learn with the same manner can be justified again by Figure **5.1.2**

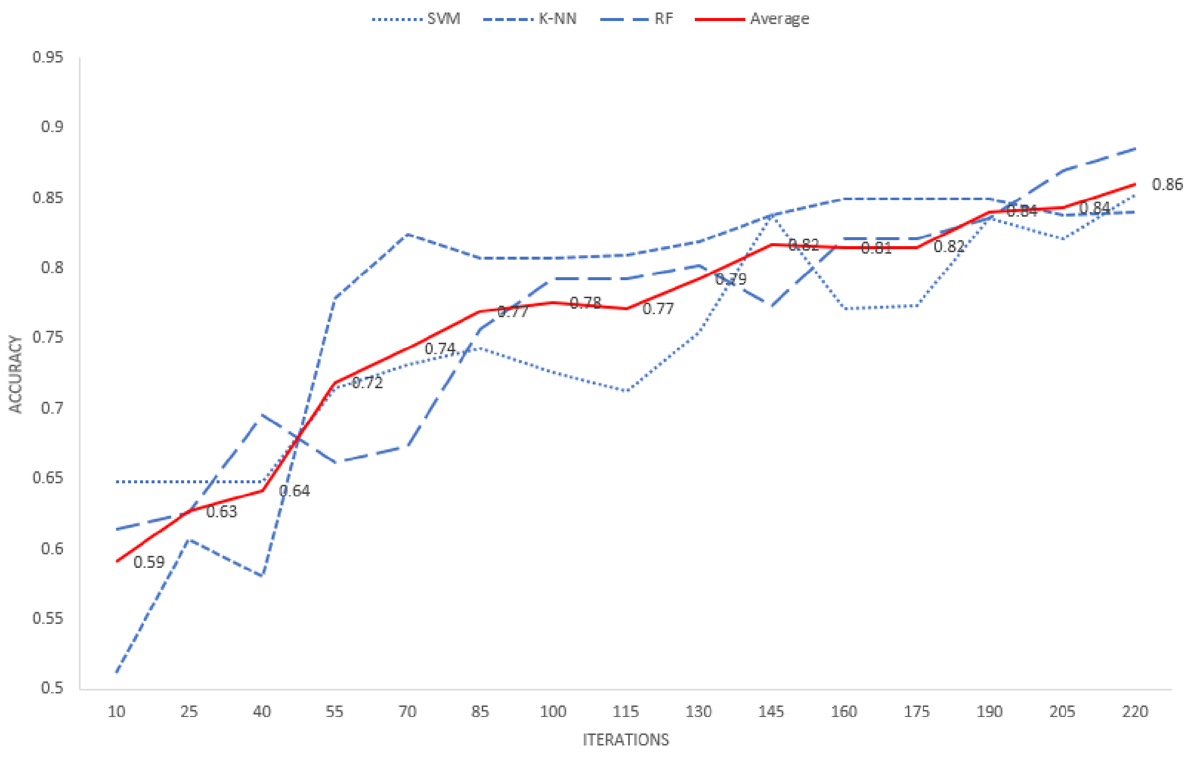


Figure **5.1.2**: SVM, kNN, RF and average accuracy values from a typical execution of RVNS on the Colon dataset.

5.2 Comparison

As stated earlier, the proposed fitness function tries to achieve high performance on three learning algorithms while maintaining a small gene subset size. However, most related work was conducted by targeting one or two ends. Thereby, within the context of a *search strategy* comparison, the objective function of RFE-RVNS and RVNS is modified in order to take only the SVM's accuracy into consideration, meaning that only a third of the classifications originally made will occur. That allows us to intensify the search capability of RFE-RVNS and RVNS by increasing the iterations from 300 to 500 and 1,000 respectively. Comparatively, in most related studies, *Wrapper* methods tend to evaluate a few thousands candidate solutions as mentioned in Chapter 3.

It is also in the same section that we refer to the Genetic Algorithm (GA) (Alba, Garcia-Nieto, Jourdan, and Talbi 2007), the Geometrical Particle Swarm Optimization (GPSO) (Alba, Garcia-Nieto, Jourdan, and Talbi 2007), the Genetic Bee Colony (GBC) (Alshamlan, Badr, and Alohali 2015) and the Maximum Relevance Minimum Redundancy - SVM-RFE (MRMR+SVM) (Mundra and Rajapakse 2010) algorithms. Furthermore, in Table **5.6**, performance of FS schemes like the Multiple Filter Multiple Wrapper (MFMW) (Leung and Hung 2010) and the Ensemble Neural Network (ENN) (Liu, Cui, Jiang, and Ma 2004) is presented. Lastly, it must be pointed out that, the Feature Selection - Random Projection (FS+RP) (Xie, Li, Zhang, and Wang 2016) method does not appertain to typical FS techniques presented in this thesis; instead, it is associated with the feature extraction ones. However, we proceed to a comparison with it since, to the best of our knowledge, no other FS methods tested on the exact Breast cancer dataset can be found in the literature.

Table **5.6**: The performance of RFE-RVNS and RVNS algorithms when applied only with the SVM classifier compared to similar methods.

Reference	Leukemia	Lung	Ovarian	Breast	Colon
RFE-RVNS	99.86[4]	99.51[3]	99.80[3]	99.12[2]	96.69[5]
RVNS	98.84[5]	98.67[3]	98.55[4]	98.66[2]	93.74[6]
GA	95.86[4]	99.49[4]	98.83[4]	-	100[3]
GPSO	97.38[3]	99.00[4]	99.44[4]	-	100[2]
GBC	96.43[5]	-	-	-	91.51[5]
MFMW	-	98.34[6]	-	-	95.16[6]
SVM-RFE	98.35[37]	-	-	-	91.68[78]
ENN	-	-	99.21[75]	-	81.48[-]
FS+RP*	-	-	-	98.97[>100]	-

In Table **5.6**, results indicate that RFE-RVNS outperforms well-known gene selection methods in all datasets except for the Colon one, while RVNS also obtains notable results. Thus, a small, yet informative, gene subset can be successfully obtained under a VNS strategy. Compared to similar methods, our algorithms perform better in terms of computational time since they evaluate significantly less candidate solutions.

CHAPTER 6

Statistical Analysis

In this chapter we present an analysis of RVNS and RFE-RVNS from a statistical point of view. Besides the key performance metrics, which are depicted in tables **5.1** to **5.5**, we also consider other metrics such as the number of iterations needed for the algorithms to converge and how the class imbalance in the datasets affects our methods. Next, we examine the correlation between gene values from the final gene subsets obtained after a typical execution of RFE-RVNS. An interesting insight is to discover the neighborhood structure which yields the most improvements. Finally, we apply a *Mean Equality Test* to check whether the difference between mean performances of the two methodologies, mainly from an accuracy's perspective, are statistically significant.

6.1 Convergence

In Figure **6.1.1** it becomes obvious that 50% of the time, RVNS converges in less than 250 iterations, with the biggest average at around 230 iterations for the Leukemia dataset. Therefore, and in conjunction with the satisfying results we presented in Chapter 5, our intuition of setting the maximum iterations to 300 is a reasonable choice.

Another interesting fact is that, with outliers omitted from the data, the average number of iterations, i.e., the black crosses within the boxes, tend to coincide with the median, i.e., the black horizontal lines within the boxes, and all boxplots indicate that the number of iterations required for the algorithm's convergence resemble the ones from a normal distribution, as shown in Figure **6.1.2**.

Finally, Figure **6.1.3** depicts the number of iterations needed in order for RFE-RVNS to converge. Similarly with the RVNS method, a threshold of 300 iterations consists a justifiable choice.

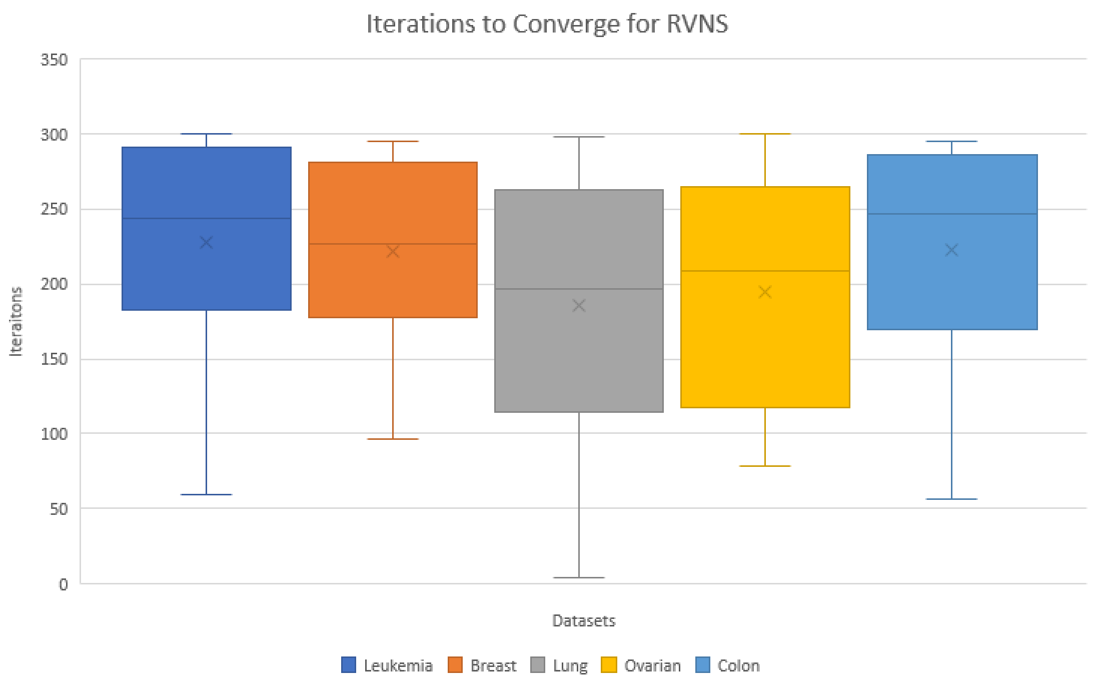


Figure **6.1.1**: Boxplot with the number of iterations needed for RVNS convergence.

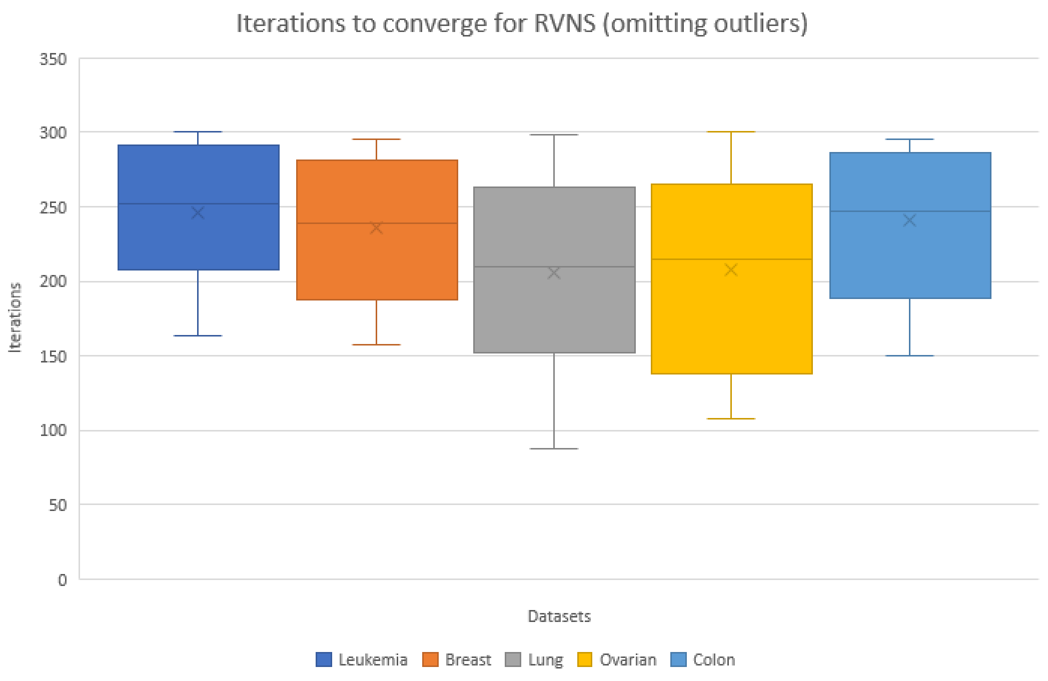


Figure **6.1.2**: Boxplot with the number of iterations needed for RVNS convergence (omitted outliers).

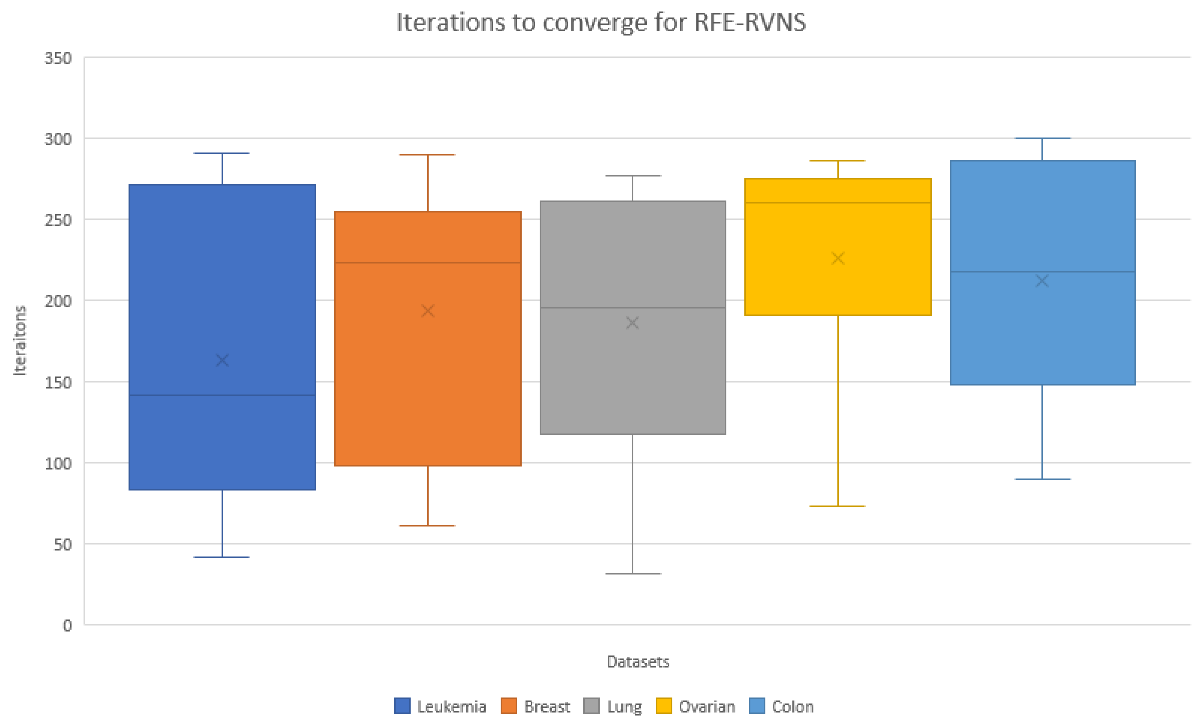


Figure **6.1.3**: Boxplot with the number of iterations needed for RFE-RVNS convergence.

6.2 Class Imbalance Influence

In this section we will elaborate on the effect of the class imbalance in the datasets, if any, on the learning algorithms' performance. More specifically, we will define a metric M as follows:

$$M = \frac{|ClassA|}{|ClassB|}$$

, where $|ClassA|$ and $|ClassB|$ denote the number of samples in class A and class B, respectively. The formulation above suits our case, since all datasets are two-class. An assumption we need to make, though, is that the nominator will always be formed by the most populated class, i.e., $M \geq 1$. Knowing the M value for each dataset, we move on to a *Pearson* correlation check between the latter and each performance metric (mean values) as shown in Tables **6.1** and **??**.

The last line of Table **6.1** indicates no significant correlation between the M-value and accuracy metrics, since correlation values lie within $[-0.5, 0.5]$ and can be considered unimportant. However, the number of genes seems to interact with the M-value and, more specifically, the bigger the M-value, the smaller the number of genes. The same pattern is observed for RVNS in Table **6.2**, with slightly more correlated values. Therefore, the performance of RFE-RVNS is less affected by the class imbalance issue, when compared to RVNS. Of course, we should point out that our sample is restricted to the number of datasets we used in this thesis; thereby, the

Table **6.1**: Correlation of the M-value with the respective metrics for RFE-RVNS.

		RFE - RVNS				
Dataset	**M-value**	**Average**	**SVM**	**k-NN**	**RF**	**#Genes**
Leukemia	1.88	98.88	98.75	99.58	98.32	3.8
Lung	4.84	98.65	98.73	98.76	98.56	2.2
Ovarian	1.78	99.18	99.49	99.21	98.85	2.4
Colon	1.82	91.23	93.36	92.69	87.64	5.5
Breast	8.67	98.83	98.83	98.97	98.7	1.7
Correlation	-	0.34	0.31	0.30	0.37	-0.67

Table **6.2**: Correlation of the M-value with the respective metrics for RVNS.

		RVNS				
Dataset	**M-value**	**Average**	**SVM**	**k-NN**	**RF**	**#Genes**
Leukemia	1.88	94.26	95.18	93.84	93.76	4.7
Lung	4.84	95.67	95.75	95.19	96.07	2.5
Ovarian	1.78	96.94	98.06	97.04	95.74	4.1
Colon	1.82	87.72	89.50	87.79	85.88	5.5
Breast	8.67	98.35	98.39	98.39	98.29	2.5
Correlation	-	0.58	0.50	0.59	0.62	-0.83

said allegations are not accompanied by a statistical significance. However, they can be used as some first insights whose gravity is to be further examined.

6.3 Gene Correlation

An ideal FS scheme should maintain insignificantly correlated features in the rationale that correlated ones carry the same information about the imminent analysis' scope. Thus, it is interesting to see if the final gene subsets, obtained by our methods, consist of genes uncorrelated with each other. To benchmark the results, we compare the correlation of gene values obtained by RVNS with those of random gene subsets of the same size. Experiments are conducted on two datasets, namely the Leukemia and the Colon dataset.

As expected, Figures in Tables **6.3** through **6.6** witness that the correlation of gene expression values obtained by RVNS is insignificant. Random ones, on the other hand, seem to be slightly correlated in some cases.

6.4 Improvements per Neighborhood Structure

Another insightful analysis of our methods is to figure out how were the different neighborhood structures utilized. In particular, we are interested to see if, during a typical search, Table **6.3**: Correlation of gene values per two. Genes are obtained by a typical execution of RVNS on the Leukemia dataset. Mean accuracy with reported genes is 94%.

Gene id	1673	5846	1749	3778	3644	5156
1673	1	0.01	0.16	-0.03	0.21	0.07
5846	-	1	-0.01	0.05	-0.19	0.16
1749	-	-	1	-0.08	0.23	-0.04
3778	-	-	-	1	-0.12	-0.12
3644	-	-	-	-	1	0.10
5156	-	-	-	-	-	1

Table **6.4**: Correlation of gene values per two. Genes are obtained randomly on the Leukemia dataset. Mean accuracy with reported genes is 80%.

Gene id	4697	4649	6312	5920	2667	736
4697	1	-0.10	0.10	0.17	0.09	0.39
4649	-	1	-0.02	-0.30	0.14	-0.21
6312	-	-	1	-0.01	-0.01	0.00
5920	-	-	-	1	-0.25	0.02
2667	-	-	-	-	1	-0.34
736	-	-	-	-	-	1

Table **6.5**: Correlation of gene values per two. Genes are obtained by a typical execution of RVNS on the Colon dataset. Mean accuracy with reported genes is 88%.

Gene id	491	49	1555
491	1	0.04	0.07
49	-	1	0.39
1555	-	-	1

Table **6.6**: Correlation of gene values per two. Genes are obtained randomly on the Colon dataset. Mean accuracy with reported genes is 60%.

Gene id	1976	1474	648
1976	1	0.54	0.58
1474	-	1	0.57
648	-	-	1

there are neighborhoods being overused, i.e., obtaining a better than the incumbent solution is more frequent when searching within them, while others are underused. At this point, it is convenient to remember the three neighborhood structures as defined in Chapter 4:

1. *Replace a selected gene of the incumbent solution with an un-selected one.*
2. *Replace two selected genes of the incumbent solution with an un-selected one. If the incumbent solution has only one gene selected, return the incumbent solution.*
3. *Add an un-selected gene to the incumbent solution. If there are no more genes to add, return the incumbent solution.*

All data are collected after ten independent runs of both algorithms. Results are presented in pie charts and values upon them are the absolute values of improvements per neighborhood and not percentages.

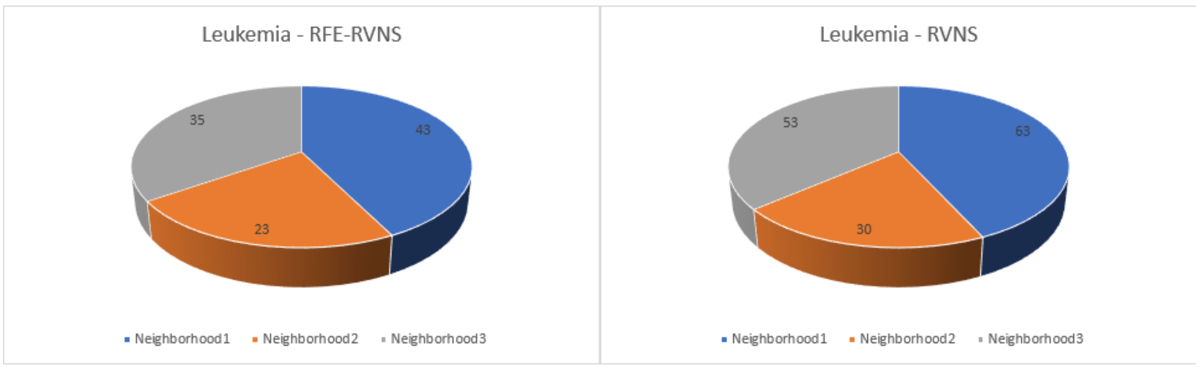


Figure **6.4.4**: Improvements per neighborhood after 10 independent runs on the Leukemia cancer dataset.

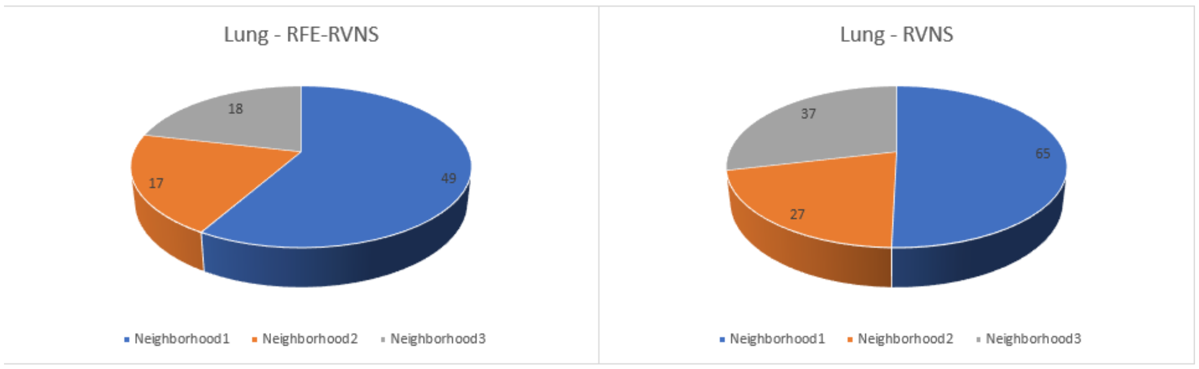


Figure **6.4.5**: Improvements per neighborhood after 10 independent runs on the Lung cancer dataset.

Two are the obvious findings depicted in all pie charts. First, all neighborhood structures can be considered necessary for the search process, since each of them possess a statistically

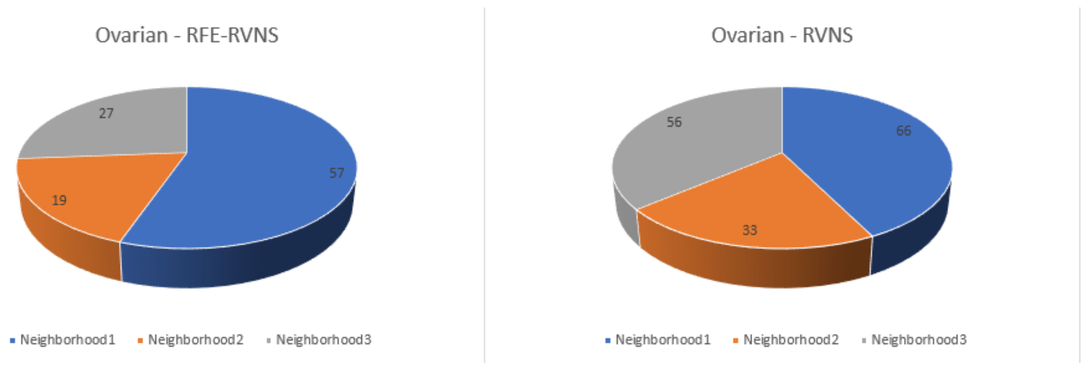


Figure **6.4.6**: Improvements per neighborhood after 10 independent runs on the Ovarian cancer dataset.

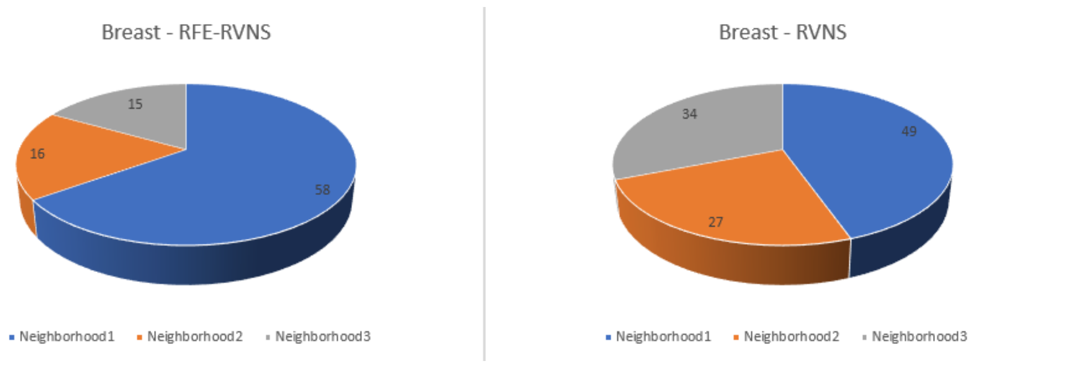


Figure **6.4.7**: Improvements per neighborhood after 10 independent runs on the Breast cancer dataset.

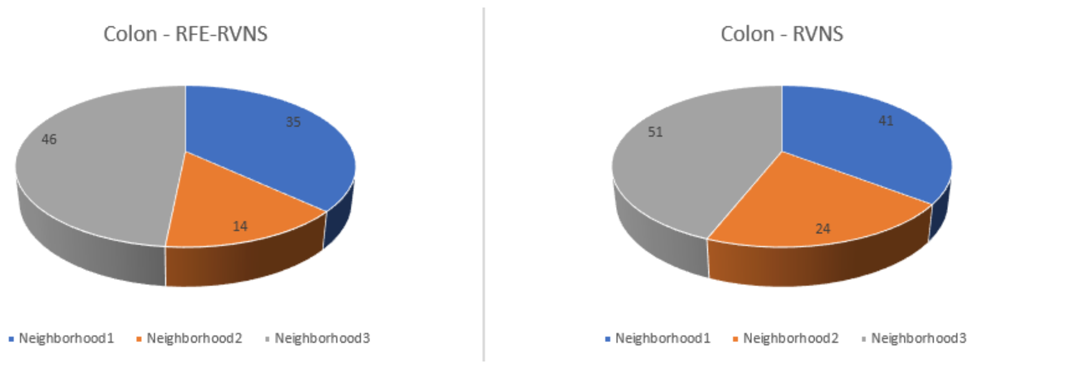


Figure **6.4.8**: Improvements per neighborhood after 10 independent runs on the Colon cancer dataset.

significant share of improvements. Second is the fact that, yet again, our algorithms' behavior does not follow the same trend in the Colon dataset, as compared to the rest. The fundamental difference in our algorithms' behavior is that, in the Colon dataset, neighborhood structure 3 offers more improvements than it does in the other datasets, where neighborhood structure 1 prevails. That can be interpreted as a tendency of algorithms to ask for more information, i.e., genes, in order for the classifiers to obtain more accurate predictions for the Colon dataset.

6.5 Statistical Difference of RVNS and RFE-RVNS

Despite we have already observed some notable differences between RVNS and RFE-RVNS in Section 5.1, a statistical evaluation on major metrics is essential. In order to obtain reliable results, we ran our algorithms 30 times on each dataset. Metrics under examination are SVM, kNN and RF accuracy, as well as their average one. Moreover, the number of genes in the final gene subsets is also a measure we statistically compare.

Table **6.7**: p-values after a t-test of equal means on RVNS and RFE-RVNS performance metrics.

Metric	Leukemia	Lung	Ovarian	Breast	Colon
Average	7.02×10^{-11}	1.26×10^{-9}	1.56×10^{-9}	7.93×10^{-8}	9.41×10^{-11}
SVM	8.24×10^{-11}	7.19×10^{-8}	4.70×10^{-9}	8.91×10^{-8}	2.54×10^{-9}
k-NN	9.78×10^{-11}	1.10×10^{-8}	7.02×10^{-10}	1.22×10^{-7}	1.24×10^{-6}
RF	7.94×10^{-8}	6.63×10^{-8}	7.82×10^{-8}	3.39×10^{-5}	8.62×10^{-4}
#Genes	6.00×10^{-3}	8.51×10^{-7}	5.88×10^{-5}	1.64×10^{-6}	1.34×10^{-1}

Table **6.7** presents the p-values that resulted after a Welch's t-test of equal means. It should be noticed that such a statistical test works under the assumption that the two populations have unequal variances and we use it since there is no indication to assume the two populations' variances coincide. The two hypothesis made by the named test are:

1. H_0: the means of the two populations are equal
2. H_1: H_0 is not true

In case the p-value is less than a certain threshold, i.e., level of significance, then someone should reject hypothesis H_0. In our case, we apply the aforementioned t-test with 0.05% significance level. Figures in **6.7**, thus, indicate different means of all populations studied, except for the number of genes in the Colon dataset. In a nutshell, the said statistical analysis indicates that RVNS and RFE-RVNS do not perform equally in terms of accuracy and gene subset size.

CHAPTER 7

Conclusion

The final chapter is dedicated in summing up the objective, the methodology and the performance of this thesis algorithmic proposal. Moreover, we mention the extent to which this study tackles the FS problem and present references for future work. We conclude by commenting upon the future directions of combinatorial optimization.

7.1 Summary

As already stated, the FS problem pertains to the class of combinatorial optimization problems. More specifically, the solution space grows exponentially with the problem's size and exact integer programming techniques fail to converge to a solution within a reasonable amount of time, even for small problem instances.

The need of solving, to a certain degree, the FS problem, led to the development of specific FS algorithms, as well as the use of search techniques successfully applied in combinatorial optimization problems belonging to other domains. Thereby, nowadays, FS algorithms are divided into *Filters*, *Wrappers*, *Embedded* and *Ensemble*, with the hybridization of two or more from the said categories being a recent trend in the field.

Our initial approach involves the development of a *Wrapper* technique, namely an RVNS algorithm. RVNS, a variant of the basic VNS algorithm, benefits from the systematic changes in the neighborhoods structures it is searching into but avoids local search; thus, it is suitable for large problem instances, like the ones we test it upon. Following the emerging tendency in the FS domain, we have also developed a hybrid RFE-RVNS algorithm, which essentially integrates two-phases, the first one made performed by the RFE and the second one from the RVNS.

Among several domains addressing the need of FS, we proceed in applying our algorithms upon cancer-related datasets, which are essentially comprised from gene values and, thereby, we intertwine the term *feature selection* with *gene selection.* In particular, we are interested to reduce the datasets' variables so that a supervised-learning algorithm is able to correctly classify unknown patients with as little information as possible. In order to have a solid verdict of whether the chosen features carry the required information, we each time train and test three different-learning classifiers with the same gene subset.

Result-wise, both of our algorithms converge reasonably fast and, compared to similar recent methods, significantly faster. Additionally, gene subsets obtained are impressively small and the accuracy levels extremely high. Therefore, we suggest both RVNS and RFE-RVNS as appropriate gene selection algorithms.

From a statistical analysis standpoint, we have shown in Chapter 6 a few quite valuable insights. First, we have presented some boxplots on *the number of iterations* to converge and verified that the selection of 300 unsuccessful iterations is a reasonable choice. Another interesting outcome of the said analysis was the fact that each neighborhood structure contributes, to a certain degree, to the optimization procedure. Last, let us point out once again that the t-test of equal means on the accuracy values of the two methodologies showed that the two proposed methodologies do not perform equally, with RFE-RVNS having a considerate advantage over RVNS.

7.2 Study Limits - Constraints

Within the scope of the present thesis, we develop and test our RVNS and RFE-RVNS algorithms on cancer related datasets and, in particular, on two-class classification problems. Despite the presence of a multitude of datasets necessitating FS, we focus our research on the said problem class, mainly due to the fact that it is extremely difficult, just during a thesis work, to thoroughly examine the behavior of both algorithms in more domains.

Enhancing RVNS with RFE is just one option among many others, where a pre-processing FS phase could be achieved with all of known *Filters*, *Wrappers*, *Embedded* and *Ensemble* methods. However, the range of this work is limited in examining the behavior of only RFE, since RFE-RVNS performs impressively well and we have no reason, for now, to believe that the proposed pre-processing step should be replaced by another FS algorithm.

Finally, it is meaningful to underline the fact that the used datasets are all health-related ones and, as such, they consist of a relatively small sample size. Therefore, a generalization of any algorithmic model tested upon such datasets is considered naive and should be certainly avoided. Although other researchers tend to give prominence to their algorithms performing well on similar datasets, we would like to point out that, at present, there is no clear winner algorithmic-wise.

7.3 Future Work

Extending the present study, we aim at further evaluating our methods on more datasets, in an attempt to get a better view of their behavior and performance. More specifically, we would like to test RVNS and RFE-RVNS in the text classification domain, where the sample size is not an issue, although the initial feature set is large enough. Finally, we are interested to apply more *Filters* as pre-processing FS methods and combine the best performing ones with RVNS.

7.4 Thoughts on Combinatorial Optimization

We have seen, this far, the special kind of problems Combinatorial Optimization (CO) deals with, as well as their applicability in many industry aspects. Moreover, we mentioned that, in many cases, quadrillions or more of possible solutions need to be evaluated in order to mathematically prove a certain solution is the best, with respect to an objective function. The latter led to the rapid development of heuristic and metaheuristic techniques, which encompass strategies that intelligently search a finite solution space, yielding acceptable solutions that improve business processes and operations.

In this final section, we comment on the extend to which CO has infiltrated within industry and its associated operations. More specifically, we attempt to address the observed phenomenon, according to which, the advancements in the field of Operations Research (OR) and CO do not seem to be correlated with their utilization in practice. Therefore, despite OR is derived from real-world problems and its purpose is to tackle them, it is today's research in the field that has lost the track or companies are unaware of how to take advantage of OR practices or both. It is really interesting to observe this last remark in conjunction with the surprisingly huge demand in Artificial Intelligence (AI) applications. Since this demand is accompanied with a strong research focus on the scientific area of AI and ML, today's flows formed are as depicted in Figure **7.4.1**.

Most people involved in the assimilation of technological developments within companies and organizations are able to distinguish between trends that return real value and marketing tricks. Dr. H. Ramalhinho Lourenco, giving a lecture on metaheuristics at the Universitat Pompeu Fabra - Barcelona, mentions that trends and industry needs change and claims that, at some point, OR will be at the forefront of developments again. Thereby, we need to be ready when that time comes. For the whole lecture someone can visit `https://www.youtube.com/watch?v=bOM-M2yXdpc`.

Let us note that, we, by no means, see the advancements in other scientific areas competitively. After all, this thesis remarkably proves how the FS problem nicely combines elements from both OR and AI. However, influenced by the rapid growth and industry's absorption of

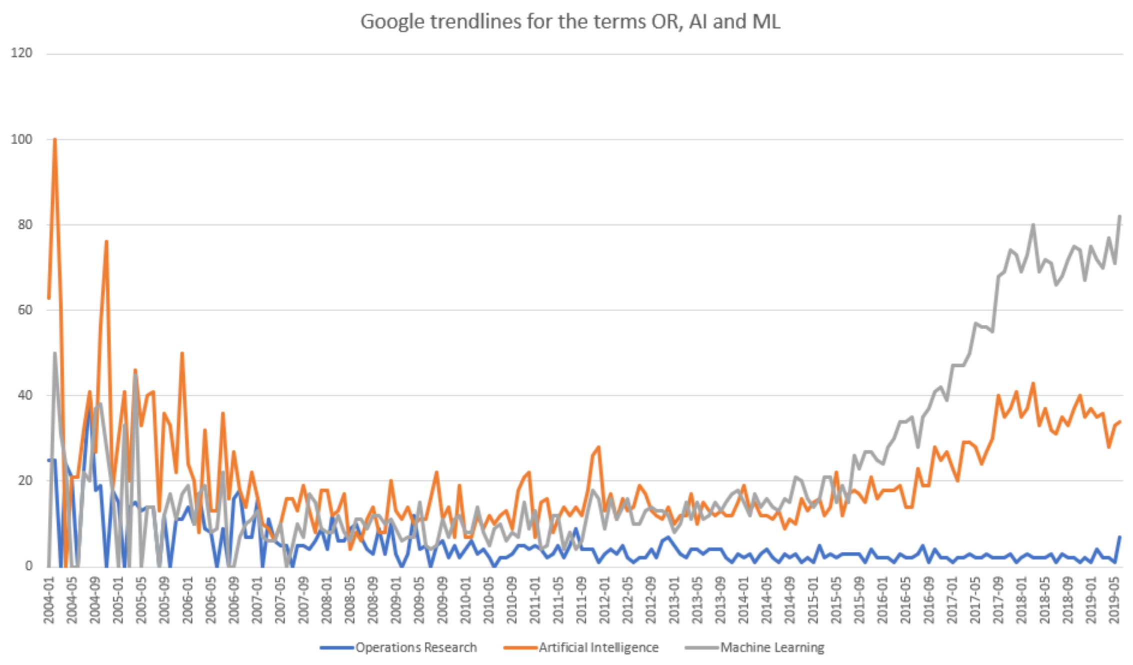


Figure **7.4.1**: Worldwide interest based on google's search engine for the terms OR, AI and ML from 2004-01 until 2019-06.

Source: https://trends.google.com

AI and ML, in this section we also present a short study in recent CO developments that can be proved game-changing.

7.4.1 Combinatorial Optimization Industry Infiltration

It was during World War II where George Dantzig, doing research as a mathematician in the U.S. Pentagon, was asked to solve a COP regarding military operations. As he writes in the Preface of his book with M. N. Thapa (Dantzig and Thapa 2006), the need of addressing the problem he was given was the motive to invent the Simplex method which, in turn, helped mathematical programming and OR shape and flower.

Today, the Simplex algorithm, along with other brilliant exact algorithms, are implemented within commercial solvers like IBM CPLEX, Gurobi and XPRESS, offered as SaaS to companies worldwide. The typical use of such software entails the strict mathematical formulation of the problem, in the form of a linear or a mixed-integer linear program, which sometimes requires the direct communication of the IT department of the client side with experts from the SaaS company's side.

In section 2.1, we established the fact that exact algorithms lack in computational efficiency against heuristic and metaheuristic approximate methods. However, it is worth to notice that the said solvers are capable of handling millions of variables and constraints and, with modern parallelization and cloud computing techniques, they manage to compute optimal solutions within hours (mainly running in batch mode for tasks that a few hours of computing time is worth in favor of an exact solution). Therefore, there are thousands of leading companies that have added this kind of optimization software to their technical stack. For instance, Figure **7.4.2** witnesses only a few of the Gurobi Solver clients.

Figure **7.4.2**: Indicative Gurobi Solver clients.

Source: http://www.gurobi.com/company/example-customers

Turning to the approximate methods, things are not so clear as the exact ones. Essentially, we can only assume that, for problems that are intractable even for the said powerful solvers, companies have to implement heuristic and metaheuristic techniques to tackle them. A certain thing is that, the fact that these methods do not guarantee optimal solutions, constitutes a critical reason for companies not to advertise their usage. Moreover, Dr. H. Ramalhinho Lourenco, in the same lecture we mentioned before, says that perhaps many companies ignore their existence. Of course, if the latter stands to a big extend, anyone involved in the OR field should be worried.

However, there are indications that colossal IT companies leverage the advantages of heuristic and metaheuristic algorithms, and we mainly base this allegation on the research they publish. For example, Microsoft Research Silicon Valley lab published a paper regarding heuristic validation for virtual machines consolidation (Lee, Panigrahy, Prabhakaran, Ramasubramanian, Talwar, Uyeda, and Wieder 2011) that negotiates and compares the efficiency of virtual machine placement policies, using a variety of simulation scenarios. Another publication concerning the same COP is that of (Gao, Guan, Qi, Hou, and Liu 2013), where one of the authors, Liang Liu, was that time working at China's IBM Research department.

The aforementioned cases enhance the reasoning that heuristic and metaheuristic algorithms are, indeed, daily contributors towards optimized business operations in many problems where exact solvers fail. Our final point is that, if someone looks more carefully into the issue we discuss, they might notice that such business operations are information-sensitive. Competency tactics, thus, definitely possess a huge part in keeping intra-business algorithmic developments obfuscated. At the end, companies advertise what they sell and not how they implemented it.

7.4.2 Future Directions of Combinatorial Optimization

Following recent advances within CO, the etching of two courses can be observed. The first one involves the automation of metaheuristic algorithms configuration, while the second concerns with the development of specific hardware that leverages the quantum-inspired concept of a bit's superposition, i.e., the capability of a bit to both be at state 0 and state 1 at the same time. The next paragraphs are dedicated in shortly elaborating on both CO future directions.

Stützle and López, within their recent paper (Stützle and López-Ibáñez 2019), describe the difficulties designers face during the process of developing a metaheuristic algorithm. Using iterated local search (Lourenço, Martin, and Stützle 2003) as an exemplary metaheuristic, they show how a multitude of different configurations can lead to different metaheuristics. Their work concludes by claiming that existing automatic algorithm configuration techniques will lead in the way metaheuristic algorithms will be implemented in the years to come.

The second direction of advances in CO derives from the bottleneck current computers face, e.g., Moore's law. A promising approach is that of Fujitsu and other leading IT companies follow, that is, to intertwine concepts from quantum-mechanics with current hardware materials towards the design of specific machines that are capable of tackling COPs within significantly shorter times than traditional computers do. The said development, as Figure **7.4.3** illustrates, is expected to balance the analogy of data volume and computing performance.

A typical example of this kind of specific hardware consists the Fujitsu Digital Annealer machine. According to its official website, `https://www.fujitsu.com/global/digitalannealer`, Digital Annealer is the world's first Quantum-Inspired technology and, advertises 300x faster computing times than traditional computers, with applicability within a wide range of COPs. In a nutshell, this machine is built on a custom configured FPGA that is able to emulate quantum bits in a digital circuit design.

Despite the aforesaid advances look quite promising, let's not forget that they consist private

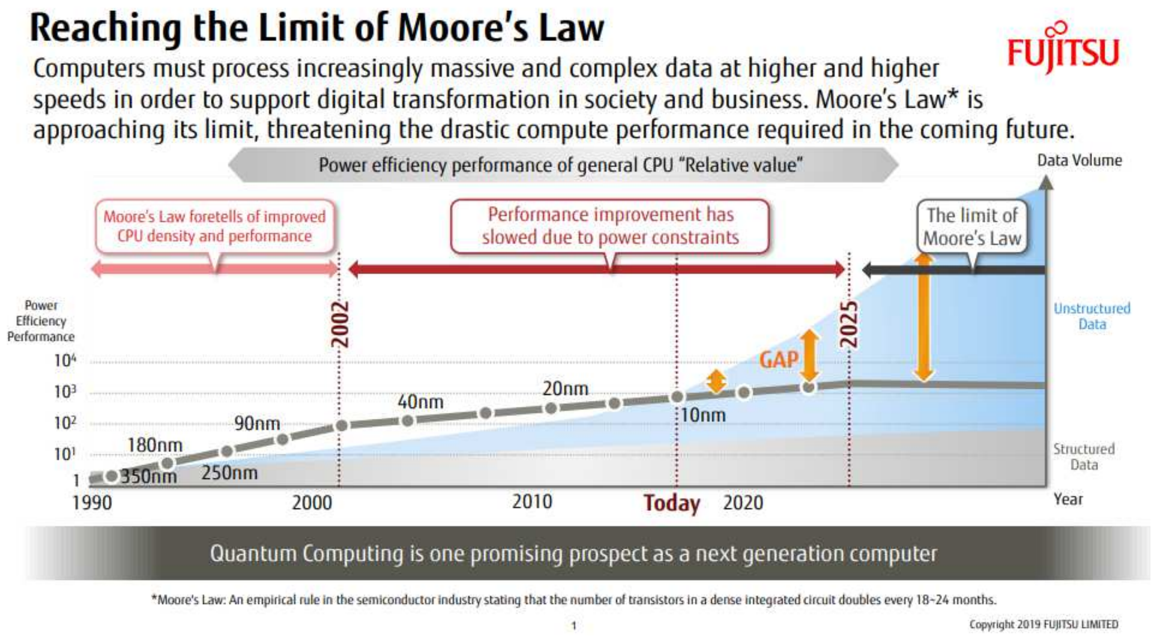


Figure **7.4.3**: The need for more computing power.

Source: https://www.fujitsu.com/global/documents/digitalannealer/services/da-introduction-.pdf

company assets. Therefore, it is more safe to judge them after their performance is extensively tested in real life problems.

We conclude by mentioning the famous, among computer scientists, paraphrased saying that, *good algorithms are better than any hardware.* Our belief is that, in the case of COPs, it is probably developments in both algorithms and computing architectures that will give prominence to the value of OR and CO techniques, making their absorption by industries a necessity.

BIBLIOGRAPHY

Alba, E., J. Garcia-Nieto, L. Jourdan, and E.-G. Talbi (2007). Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 284–290. IEEE.

Alon, U., N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences 96*(12), 6745–6750.

Alpaydin, E. (2009). *Introduction to machine learning*. MIT press.

Alshamlan, H. M., G. H. Badr, and Y. A. Alohali (2015). Genetic bee colony (gbc) algorithm: A new gene selection method for microarray cancer classification. *Computational biology and chemistry 56*, 49–60.

Beloglazov, A., J. Abawajy, and R. Buyya (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems 28*(5), 755–768.

Bir-Jmel, A., S. M. Douiri, and S. Elbernoussi (2019). Gene selection via bpso and backward generation for cancer classification. *RAIRO-Operations Research 53*(1), 269–288.

Blum, C. and A. Roli (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR) 35*(3), 268–308.

Bolón-Canedo, V., N. Sánchez-Marono, A. Alonso-Betanzos, J. M. Benítez, and F. Herrera (2014). A review of microarray datasets and applied feature selection methods. *Information Sciences 282*, 111–135.

CGAN, C. G. A. N. et al. (2012). Comprehensive molecular portraits of human breast tumours. *Nature 490*(7418), 61.

Chuang, L.-Y., C.-H. Yang, K.-C. Wu, and C.-H. Yang (2011). A hybrid feature selection method for dna microarray data. *Computers in biology and medicine 41*(4), 228–237.

Cover, T. M., P. E. Hart, et al. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory 13*(1), 21–27.

Dantzig, G. B. and M. N. Thapa (2006). *Linear programming 1: introduction.* Springer Science & Business Media.

Dash, M. and H. Liu (2003). Consistency-based search in feature selection. *Artificial intelligence 151*(1-2), 155–176.

Domon, B. and R. Aebersold (2006). Mass spectrometry and protein analysis. *science 312*(5771), 212–217.

Gao, Y., H. Guan, Z. Qi, Y. Hou, and L. Liu (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences 79*(8), 1230–1242.

Golub, T. R., D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science 286*(5439), 531–537.

Gordon, G. J., R. V. Jensen, L.-L. Hsiao, S. R. Gullans, J. E. Blumenstock, S. Ramaswamy, W. G. Richards, D. J. Sugarbaker, and R. Bueno (2002). Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer research 62*(17), 4963–4967.

Guyon, I., J. Weston, S. Barnhill, and V. Vapnik (2002). Gene selection for cancer classification using support vector machines. *Machine learning 46*(1-3), 389–422.

Hall, M. A. (1999a). Correlation-based feature selection for machine learning.

Hall, M. A. (1999b). Correlation-based feature selection for machine learning.

Hansen, P., N. Mladenović, and J. A. M. Pérez (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research 175*(1), 367–407.

Kerr, M. K., M. Martin, and G. A. Churchill (2000). Analysis of variance for gene expression microarray data. *Journal of computational biology 7*(6), 819–837.

Kira, K. and L. A. Rendell (1992). The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, Volume 2, pp. 129–134.

Kytöjoki, J., T. Nuortio, O. Bräysy, and M. Gendreau (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & operations research 34*(9), 2743–2757.

Lee, S., R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, L. Uyeda, and U. Wieder (2011). Validating heuristics for virtual machines consolidation. *Microsoft Research, MSR-TR-2011-9*, 1–14.

Leung, Y. and Y. Hung (2010). A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 7*(1), 108–117.

Liu, B., Q. Cui, T. Jiang, and S. Ma (2004). A combinational feature selection and ensemble neural network method for classification of gene expression data. *BMC bioinformatics 5*(1), 136.

Lourenço, H. R., O. C. Martin, and T. Stützle (2003). Iterated local search. In *Handbook of metaheuristics*, pp. 320–353. Springer.

Mladenović, N. and P. Hansen (1997). Variable neighborhood search. *Computers & operations research 24*(11), 1097–1100.

Mundra, P. A. and J. C. Rajapakse (2010). Svm-rfe with mrmr filter for gene selection. *IEEE transactions on nanobioscience 9*(1), 31–37.

Narendra, P. M. and K. Fukunaga (1977). A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers* (9), 917–922.

Osman, I. H. and J. P. Kelly (1996). Meta-heuristics: an overview. In *Meta-heuristics*, pp. 1–21. Springer.

Paessens, H. (1988). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research 34*(3), 336–344.

Papadimitriou, C. H. (2003). *Computational complexity.* John Wiley and Sons Ltd.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

Petricoin III, E. F., A. M. Ardekani, B. A. Hitt, P. J. Levine, V. A. Fusaro, S. M. Steinberg, G. B. Mills, C. Simone, D. A. Fishman, E. C. Kohn, et al. (2002). Use of proteomic patterns in serum to identify ovarian cancer. *The lancet 359*(9306), 572–577.

Saeys, Y., T. Abeel, and Y. Van de Peer (2008). Robust feature selection using ensemble feature selection techniques. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 313–325. Springer.

Sifaleras, A., I. Konstantaras, and N. Mladenović (2015). Variable neighborhood search for the economic lot sizing problem with product returns and recovery. *International Journal of Production Economics 160*, 133–143.

Stützle, T. and M. López-Ibáñez (2019). Automated design of metaheuristic algorithms. In *Handbook of Metaheuristics*, pp. 541–579. Springer.

Toth, P. and D. Vigo (2002). *The vehicle routing problem.* SIAM.

Tsai, J.-T., T.-K. Liu, and J.-H. Chou (2004). Hybrid taguchi-genetic algorithm for global numerical optimization. *IEEE Transactions on evolutionary computation 8*(4), 365–377.

Wu, X., V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems 14*(1), 1–37.

Xiao, Y., I. Kaku, Q. Zhao, and R. Zhang (2011). A reduced variable neighborhood search algorithm for uncapacitated multilevel lot-sizing problems. *European Journal of Operational Research 214*(2), 223–231.

Xie, H., J. Li, Q. Zhang, and Y. Wang (2016). Comparison among dimensionality reduction techniques based on random projection for cancer classification. *Computational biology and chemistry 65*, 165–172.

Yu, L. and H. Liu (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 856–863.

Yu, L. and H. Liu (2004). Redundancy based feature selection for microarray data. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 737–742. ACM.