UNIVERSITY OF NOTTINGHAM

SCHOOL OF COMPUTER SCIENCE

Novel Strategies to Accelerate Search Algorithms in Data Reduction

Author:

Hoang Lam Le

Student ID: 14299422

Supervisors

Dr. Isaac Triguero

Dr. Dario Landa-Silva

Dr. Ferrante Neri

Doctoral dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

University of Nottingham

UK | CHINA | MALAYSIA

Computational Optimisation & Learning Lab

October 13, 2022

Declaration

I confirm that this thesis dissertation presented for the degree of Doctor of Philosophy at the School of Computer Science, University of Nottingham,

- been composed entirely by myself,
- been solely the result of my own work except stated otherwise,
- not been submitted in whole or in part for any other degree or qualification in any other academic institution.

Hoang Lam Le

October 13, 2022

Abstract

In our current hyper-connected digital world where data is growing enormously, instance reduction is an essential pre-processing phase to obtain cleaner and smaller datasets that are free from noise, redundant or irrelevant samples (the so-called, Smart Data). The data after pre-processing may become more reliable, accurate and useful for subsequent data mining tasks. Instance reduction consists of two types: instance selection and instance generation; each can be formulated as a combinatorial/continuous optimisation problem depending on whether its decision variable is discrete or continuous, respectively. It is an emerging challenge characterised by multimodality and a large number of decision variables. Given such difficulties, derivative-free methods are likely promising approaches to address the problem. They are powerful search algorithms that seek the nearest local optimum and do not necessarily take into account the gradient computation of the objective function like derivative methods. Solutions for instance reduction fall into the intersection of machine learning, data mining and optimisation at which the process of a domain can take part in the execution of another. Thus, the synergy between domains is important to solve the problem more effectively, and this has attracted a significant interest from researchers.

Among many different derivative-free search approaches, the family of direct search methods has introduced various strategies to tackle numerous modern numerical optimisation problems, where population-based meta-heuristics and pattern search can be considered two of the most prevalent in the literature. Population-based meta-heuristics are an iterative search framework composing several subordinate low-level heuristics to control exploration and exploitation for a pool of solution candidates. This set of methods searches for high-quality solutions from multi-points, and thus is usually associated with high computational expense. Pattern search methods seek an improved solution from candidates that are generated from different directions. They examine trial solutions sequentially by comparing each trial solution with the 'best' solution found up to the present time. In this dissertation, we will investigate these derivative-free search strategies to address instance reduction, a critical optimisation problem in the field of data science.

Although many derivative-free methods have been proved effective in addressing instance reduction, they are usually time-consuming, especially when handling relatively large datasets. This impediment limits their practicality in many data mining systems and thus necessitates a solution to accelerate the search process. The need for a fast and effective search framework for instance reduction has motivated us to develop novel search strategies in the family of direct search approaches, aiming to still obtain high quality solutions achieved by state-of-the-art techniques in the domain, but significantly reduce the runtime of the search process. Three major work packages presented in this thesis will cover two direct search approaches for two types of instance reduction, arranged in a progressive order at which findings at an earlier stage will contribute to the understanding of the later outcomes. Firstly, a novel evolutionary search framework for instance selection is proposed to balance the number of samples between classes to address a case study of imbalanced classification. Secondly, we develop another search framework for instance generation based on single-point search and memetic computing, namely Single-Point Memetic Structure. An accelerated mechanism for computing the objective function is embedded into the proposed search design, thus reducing significantly the runtime. Finally, a novel search framework for simultaneous instance selection and generation is designed to handle the instance reduction problem in both combinatorial and continuous search spaces.

In summary, the research conducted here introduces a set of novel search strategies towards derivative-free methods to tackle instance reduction problems. They are different search frameworks which aim to produce a high quality reduced set from a relatively large original source within a reasonable amount of time. This is accomplished by either taking advantage of machine learning integration or the Single-Point Memetic Structure with an accelerated mechanism. The use of machine learning in a meta-heuristic search framework greatly speeds up the computation of the objective function while the Single-Point Memetic Search allows us to reuse virtually all prior calculations for computing the fitness value of newly evolved individuals. Hence, these novel search strategies can save vast computational cost. Finally, we leverage the insights previously found to propose another novel search framework that handles both instance selection and instance generation simultaneously, and operates in both combinatorial and continuous search spaces. These novel search strategies are examined with a large number of datasets in different hyper-parameter settings. The obtained numerical results are comprehensively analysed and verified by different statistical tests to prove the robustness of the proposed search strategies with respect to other state-of-the-art techniques in the domain.

Acknowledgements

Looking back on my path of learning at school and university, I realise that I have been blessed by receiving, unconditionally, a lot of support from a variety of people. They have appeared in my life as guiding spirits to lift me over numerous obstacles and put me back on the education journey. In this brief text, I will try to express my gratitude to as many of those people as possible. Thank you all for accompanying me during the journey of my life, particularly in school and university times.

I guess my voyage in the field of data science can be traced back to school times when I was still a child loving playing riddles with friends in a small village in Vietnam. Back then, I was always fascinated by the knowledge taught by teachers or wisdom conveyed by hermits/priests. I was wondering how they could come to such wise conclusions and even anticipate the future based on their diverse observations from the past. These incidences subconsciously activated and ignited my curiosity regarding knowledge discovery and have transformed me into a person mining knowledge from digital data. Of course, in our wildest thoughts, neither my family nor I would ever dare to believe that one day I could get this far, writing this manuscript about scientific understanding to earn a doctoral degree. Thus, I would like to thank people in my hometown in Xuan Loc, Dong Nai province including teachers, priests and friends, who kindled the fire of study in me. Especially, I must deeply express my thanks to my parents Le Xuan and Hoang Thi Lai who put such effort into keeping that flame alive and even burning bright. I feel very blessed for always having their encouragement on my long journey of learning in different schools and universities.

The time at the University of Natural Science in Vietnam and Myongjy in South Korea gave me a solid foundation in computer science, from which I have grown to the person I am now. How could I forget the very first day of practising C-programming, when I forgot to end a line of code in a semicolon, and consequently spent hours to investigate why my *complicated* 'Hello World!' program could not be executed. Since that first day, I have received generous support and guidance from many mentors who helped me define my career path. To name a few from the long list, I would especially thank Prof. Dong-Min Woo, who led me into the field of data science, and Dr. Nhat-Phuong Tran and Dr. Thanh-Binh Le, who have provided timely advice for my decision making in various problems, regardless of our geographical distance and their busy schedule.

One very important person that I would like to thank is of course my first supervisor, Dr Isaac Triguero, who by a strong belief approved my initial plan of designing

a new creature derived from the two big *monsters* 'Machine Learning' and 'Optimisation'. Frankly, the design was something so vague or overgeneral that I have to admit that we began the course relying much on faith and curiosity. Deriving scientific knowledge from such an initial situation was problematic at the beginning, but perseverance kept us going, though I have to confess that the progression was sometimes as slow as a hundredth of the turtle's speed. Certainly, the journey has consisted of many inevitable difficult periods which more than once shook my will of bringing this PhD to completion. Definitely, I would not have overcome those challenging times without the companionship and encouragement of Isaac and my other co-supervisors Dr. Dario Landa Silva and Dr. Ferrante Neri, to whom I would like to express my deep gratitude for their consistent guidance and patience. Thanks for giving me ideas to explore different work packages, for spending massive time to improve reports and papers, and also simply for hanging out to enjoy a beer or coffee in various circumstances. All these experiences have been great not only in shaping myself to be more professional and independent in research, but also in generating a happy studying time in my PhD life.

I would not have enjoyed my life in Nottingham as much as I did without many other friends and societies at the University of Nottingham, who have constructed a second family of mine here. I'm extremely grateful for having all of you in my life abroad on many occasions such as Bonfire Night, Christmas and Easter breaks, Lunar New Year, etc... All have left a succession of happy memories which I think I will not be able to stop myself chatting about with new friends if I have to live in another place. In particular, the Catholic Society and flatmates at Newman House are significant in keeping me in the faith of the Roman Catholic Church and have been middlemen to introduce me to acquaintances from all over the world. Thanks also to labmates and other colleagues at the Computational Optimisation and Learning Lab for the time having fun together at summer barbecues, cake and pub events, as well as sharing the research ideas in various group meetings.

I was also thankful to be awarded the prestigious 'Vice-chancellor scholarship for postgraduate research' at the University of Nottingham, which allows me to pursue my Doctoral degree in Computer Science. Besides, the High Performance Computing system has made my costly computational experiments possible as they may take months to complete the examination of one parameter configuration if running on an individual desktop. Without these supports, the results presented in this thesis would not have been feasibly conducted.

In the end, though I feel that I have myself satisfactorily answered quite a few questions prompted by the curiosity I have had, my interior desire still continuously reminds me to carry on with more filling up the bottomless curiosity. I could not reason where this strong desire comes from or why it naturally grows in me, but

wherever it is derived from, I have no doubt that it has been the nutritious food that has nurtured my motivation to accomplish this PhD. For that reason, I hope the content presented in this thesis can be considered as an offering or a thanks in return to the Creator who may have created that desire of learning in the little boy in a small village in Vietnam over thirty years ago.

Hoang Lam Le

October 13, 2022

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, many disciplines such as medicine, business, transportation or energy are collecting data at a very striking rate, likely quintillion bytes per day in the world and this trend shows no sign of slowing down in the near future (Wang et al., 2020; Amalina et al., 2019; Hajjaji et al., 2021). This is because of the hyper-connectivity in the digital world induced by the Internet of Things and various social networks as well as the development of powerful storage (Ramírez-Gallego et al., 2018). Vast amounts of data from different sources have been collected for future analysis, aiming to seek economic profit and competitive advantages for companies and society in general. The term 'Big Data' has been coined, discussing multiple attributes of this flood of data such as Volume, Velocity, Veracity, and Variety (among others V's) (Fernández et al., 2014; Hajjaji et al., 2021).

However, the real benefit of data is not on the data itself or their size but the interior information/knowledge they can potentially carry on (Taleb et al., 2016; Taleb, Serhani, and Dssouli, 2018). Deriving knowledge from data has been qualified as science and technology to explore data, known as 'Data Science' or 'Knowledge Discovery in Databases' (KDD) (Larose and Larose, 2014; García, Luengo, and Herrera, 2015). It is a non-trivial process of extracting implicit, previously unknown and potentially useful patterns and trends stored in datasets. The process involves different methods at the intersection of machine learning (ML) and statistics. A KDD process is divided into a number of stages which may not be similar among different research communities, and each mining scheme has its own advantages and disadvantages (García, Luengo, and Herrera, 2015; Han, Pei, and Kamber, 2011). However, a KDD process typically consists of problem definition, data pre-processing, model construction and analysis (Larose and Larose, 2014; García, Luengo, and Herrera, 2015). In the context of Big Data, a KDD process is more challenging because the time and/or memory consumption is likely exceeding the processing capabilities of a usual ML system. Since traditional ML methods are incapable of handling the new data space requirements, distributed technologies such as the MapReduce programming paradigm (Dean and Ghemawat, 2010) and Big Data frameworks such

as Apache Spark (Zaharia et al., 2012) were introduced to address the new data-intensive scenario.

Data mining and ML tools have been employed in various applications of data analytics (Raja et al., 2022) including descriptive (i.e. reveals what happened), diagnostic (i.e. explains why it happened), predictive (i.e. anticipates what will happen in the future), and prescriptive (i.e. provides and analyses the best decision to make among all the possible ones) (Runkler, 2020). Data mining and ML are also essential in a variety of sectors as the vast majority of decision-making is usually driven by their real-time and historical data. Although data mining and ML are powerful to reveal the knowledge, it is undeniable that they assume the data being analysed are reliable, meaning that the mining algorithms do not generally distinguish between good data (i.e. representative, informative) and bad data (i.e. redundant, noise, outlier) on their own. As a result, more data for learning does not always mean 'good', but could be even 'worse'. In general, more data leads to a higher possibility of holding information and therefore better insights may be discovered, but as long as the data is of a high quality and representative. On the other hand, more data also means potentially more noise and redundant samples. Broadly speaking, mining knowledge from a higher rate of erroneous or incomplete data might result in a lower value found. For example, a model trained with inconsistent labels, at best, produces results that are not actionable or insightful. A bigger concern is that it is likely to generate results that are misleading, providing improper directions to decision-making in an application.

In many businesses, data quantity is bound to grow as the business grows, thus it is important to establish a set of rules and procedures to regulate how data is accumulated in order to maintain/increase the intrinsic values of the data collection. Besides, when having more data, costs on data storage and processing power gradually creep up and eventually become less sustainable for businesses. Therefore, data quality standards should be embedded within every data collection process, as letting the pile freely grow can result in getting counterproductive effects in many different aspects. In addition, there would come to a point where no additional data is needed as the dataset was already broad enough to get the most out of a KDD system. In such a circumstance, the data mining system is already completely saturated with data, it might make more sense to reduce rather than to expand the amount of data as there would be no more insights revealed. However, it has not reached a consensus among the answers on the questions about data quantity: Is more data better? Is it necessary to hoard big amounts of raw data that may contain inaccuracies just for the sake of it (Taleb, Serhani, and Dssouli, 2018; Taleb et al., 2016)?

For years in data mining and ML, researchers have focused on improving the problem's representations, or algorithms to improve the performance over different datasets (Filippone et al., 2008; Gündüz et al., 2019; Alpaydin, 2014). One possible reason behind this is that practitioners can reuse the knowledge implemented in a task to address many other similar ones. On the other hand, it is difficult to create datasets that can become generally recognised standards due to the lack of domain knowledge or the unknowability of the instance space. As a result, the data mining and ML communities believe that model-oriented approaches (i.e. focusing on improving the algorithms' performance by techniques such as parameter tuning, model selection, or different problem's representation) might be more promising. Depending on the nature of each learning algorithm, a model-oriented approach may not necessarily require some pre-preprocessing step like noise removal (e.g. Naive Bayes and Decision Tree) as they are robust against noise data (Atla et al., 2011)). However, it does not mean that their performance is not affected at all levels. More generalisable results might be produced with the high-quality training data feeding into these models, not mentioning faster learning and less storage required.

Unlike Naive Bayes or Decision Tree models, many others ML techniques, especially the Nearest Neighbour rule (NN), may perform more poorly than they would have been if the data were not handled properly (i.e. outlier removal, noise removal, redundant check, imputation). As time goes by, their performance may gradually go down due to the contamination of the data collection when having more and more unverified samples added. This situation eventually ends up with learning from a lower-quality dataset, wasting hours on optimising hyper-parameters and other factors of a learning process. Perhaps, there has nothing to do with the improvement on the model but on the data quality because simply maintaining a high-quality set of data will already give a decent set of returns rather than just mindlessly fiddling model hyper-parameters. In addition, models are usually susceptible to concept drift, and their performance is deeply tied to the data they were trained on (Lu et al., 2018; Polyzotis et al., 2018). Pitfalls for being too fixated on models might need correction, perhaps by a different problem-solving paradigm. In various research communities, there has been recently a transition from focusing on modelling (i.e. model-centric) to the underlying data (i.e. data-centric) (Polyzotis et al., 2018; Liu et al., 2019; Sharma and Liu, 2020; Bartel, 2021). Model-centric approaches keep the data fixed and iterate over the model and its parameters to improve performances, while data-centric counterparts concentrate on data quality and enhance the accuracy and generalisability of a more-or-less fixed model architecture. Thus, in this research direction, the data preparation phase in a KDD process may not be a one-time event, performed at the beginning of a project, but a continuation of tuning when the instance space is changed.

In the literature, the term **Smart Data** indicating quality data has been widely used (Fernández et al., 2014; Iafrate, 2014; Triguero et al., 2019). It describes the development of transforming raw data into quality data that can be exploited in almost all domains and industries (Lenk et al., 2015). In this process, we optimise the number of elements that will remain in the final set as well as their location quality in the instance space which may help distinguish samples between classes. Thus, this practice can be modelled as an optimisation problem at which the Smart Data set will be obtained through different search strategies such as meta-heuristics (e.g. Genetic Algorithms - Crossover elitism population, Half uniform crossover combination, Cataclysm mutation (CHC)) (Eshelman, 1991), Memetic Algorithms (García, Cano, and Herrera, 2008), Differential Evolution (Triguero, García, and Herrera, 2011)), machine-learning-based techniques (e.g. clustering (Xu and Wunsch, 2005; Yen and Lee, 2009)), pattern search (Neri and Rostami, 2021) and others (Hedjazi et al., 2015; Triguero et al., 2014). The process of extracting insights from the high-quality resulting set (big or not) is called **Smart Data discovery** highlighting twofold: higher quality data mining and reduction of cost (i.e storage and computation) (Iafrate, 2014; Lenk et al., 2015; García-Gil et al., 2019). The term **Smart Data discovery** is used to distinguish itself from data discovery used in the past which was usually a time-consuming and effort-inefficient practice as it did not leverage the aid of automation but relied on human's understanding of the data (Lenk et al., 2015). With the introduction of various data mining techniques and recent augmented analytics capabilities, Smart Data discovery has become the replaced model to convert raw data into actionable insights. It is worth mentioning that though the existing mining techniques are effective to convert data from raw to smart, they demand high-computational cost and are time-consuming (Fernández et al., 2014; Triguero et al., 2019). Thus there is room for either reducing the expense of computation or runtime of execution.

Motivated by the recent transition from model-centric to data-centric, and Smart Data discovery, the content presented in this thesis discusses different strategies to obtain such a high-quality dataset in a fast manner. We mainly focus on solutions for **IR** which are important techniques in the family of pre-processing methods for a KDD process (Fayyad, Piatetsky-Shapiro, and Smyth, 1996b; Larose and Larose, 2014; García, Luengo, and Herrera, 2015). Its goal is to clean and correct input data so that the modelling phase later applied may be faster and with greater accuracy. The remainder of this chapter will present the basic concepts and structure of the conducted research. Concretely, Section 1.1 presents pre-processing techniques in a KDD approach, emphasising the fundamentals of IR. Then, the motivation and objectives are discussed in Section 1.2. The contributions of the thesis are outlined in Section 1.3. Finally, the structure of the whole thesis is summarised in Section 1.4.

1.1 Data Pre-processing

As stated before, data mining techniques may be ineffective when the input data is impure. Therefore, enhancing the data quality becomes one of the most relevant stages to enable data mining methods to transform raw data into actionable insights (Alexandropoulos, Kotsiantis, and Vrahatis, 2019). Data preprocessing contains a variety of methods such as data preparation (i.e. integration, cleaning, normalisation and transformation), data reduction tasks (see Figure 1.1) (i.e. feature selection, feature extraction, IS, discretisation) and others (Liu and Motoda, 2007; García, Luengo, and Herrera, 2015; Zebari et al., 2020). In general, data pre-processing usually aims to generate a less sizable dataset with respect to the original set, improving the efficiency of the data mining algorithms and saving storage. More importantly, the data pre-processing originates high-quality data, so that, more in-depth and accurate knowledge may be revealed.

This thesis focuses on the development of novel **IR** strategies, aiming to provide quick, simple and effective solutions. An **IR** solution seeks for a smaller set of the starting dataset which is as informative as the original source (and potentially freer of noise) to enhance the performance of supervised learning algorithms. Most of the existing research has been focused on classification tasks (Garcia et al., 2012; Saha et al., 2022), but its use for regression has recently been gaining popularity (Arnaiz-González et al., 2016; Kordos, Blachnik, and Scherer, 2022). IR methods are beneficial for a data mining process, and bring more other benefits to Big Data systems such as reducing storage, decreasing computational complexity and system complexity, and cutting down in-network movement of data. Considering a dataset with $\mathbf{l}$ samples, each has $\mathbf{m}$ features as a matrix of $\mathbf{l} \times \mathbf{m}$, the target of an **IR** solution is to reduce the number of rows (i.e. $\mathbf{l}$) to obtain $\mathbf{p}$ so that $\mathbf{p} \ll \mathbf{l}$. Research about **IR** can be categorised into two main directions, that is, instance selection (IS) (García et al., 2012) and instance generation (IG) (Triguero et al., 2012). **IS** searches for representative examples in the available source while generation creates artificial ones, if needed. The former group has frequently been modelled as a binary combinatorial optimisation problem since it deals with the decision whether or not to include a sample in the final subset, whilst the latter counterpart may be modelled as a continuous optimisation problem since the decision variables are operating on the continuous space. **IG** typically considers modifying feature values of the existing samples and this results in new examples non-existing in the source but better representing the training data. A similar reduction approach not focusing on rows but columns is feature selection which shrinks the number of columns (i.e. $\mathbf{m}$) to get $\mathbf{k}$, thus $\mathbf{k} \ll \mathbf{m}$. As a result, methods proposed for **IR** can be technically employed in feature reduction.

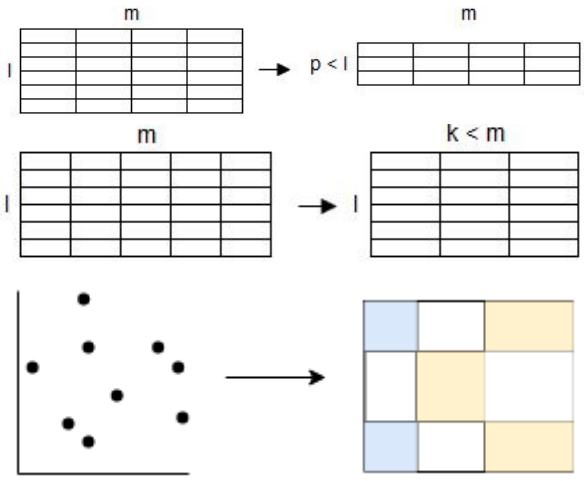


FIGURE 1.1: Instance reduction (top), feature selection (middle) and data discretisation (bottom).

IS methods are usually categorised into three main types of techniques: edition (Wilson, 1972), condensation (Hart, 1968), and hybrid (García et al., 2009). Edition methods aim to remove noisy instances, condensation methods focus on removing superfluous instances that have a little impact on the classification accuracy if removed, and hybrid methods search for a small subset that achieves the elimination of both noisy and superfluous instances. On the other hand, **IG** methods are grouped based on their different properties such as type of reduction, resulting generation set, generation mechanisms, and the evaluation of the search (Triguero et al., 2012).

Since **IR** can be formulated as an optimisation problem in the combinatorial or continuous search spaces, different methods in the optimisation domain can be used to address the problem. Due to its complexity (i.e. multimodal, high dimensionality), a derivative-free method is likely more promising to tackle the problem rather than a gradient-based approach (Brent, 1973; Conn, Scheinberg, and Vicente, 2009; Caponio et al., 2007). Derivative-free methods neglect the gradient of the objective function and seek the local optimal solution from the current position. Derivative free methods can be categorised into model-based and direct search groups. The former builds a surrogate model of the objective function from generated data points, and then computes the gradient of the surrogate model to search for the optimum (Cartis et al., 2019; Cartis and Roberts, 2019), while the latter explores the domain and searches for the optimal solution using the objective function values. Among various techniques proposed in this search family, direct search methods construct different strategies to seek for the locally optimal solution by generating and evaluating new data points (Caraffini, Neri, and Picinali, 2014). Population-based metaheuristics and pattern search are the two most popular macro groups of direct search in the literature (Neri and Rostami, 2021), which will be the two main search strategies adopted in this dissertation. To the best of our knowledge, population-based

evolutionary search solutions have achieved the highest performance among the proposed techniques addressing **IS** and IG, considering both reduction rate and classification accuracy metrics, though they were created in 2008 and 2011, respectively (García, Cano, and Herrera, 2008; Triguero, García, and Herrera, 2011).

In the literature, most of the existing **IR** solutions (Brighton and Mellish, 2002; Sánchez, 2004; Leyva, González, and Pérez, 2015; Ougiaroglou and Evangelidis, 2016) were proposed to enhance the performance of the well-known kNN classifier (i.e. k is the number of nearest neighbours) (Cover and Hart, 1967), which will be also adopted in this thesis. As a lazy algorithm, kNN does not build any model, but works based on the concept of similarity between samples, given a similarity metric. In a classification task, it finds the distances between a query and all examples in the data, and the assigned label is voted by the most frequent labels of the samples holding the closest distances. However, as kNN considers all neighbours as equally important in classification, it is more vulnerable to noise at the class boundaries. Although kNN experiences a series of difficulties such as high-computational cost (i.e. in large datasets), high-storage requirements and sensitivity to noise, it is characterised as a simple yet effective data mining technique in various applications (Triguero et al., 2019). Most of the existing **IR** techniques were proposed to address kNN drawbacks, but the resulting reduced dataset can be used in many other learning algorithms or the proposed **IR** mechanism can be tailored to a new classifier (Luengo, García, and Herrera, 2012; Cano, Herrera, and Lozano, 2003).

One major benefit of **IR** is the increase of effectiveness and accuracy in predictive tasks, demonstrated in numerous studies (He and Garcia, 2008; Marchiori, 2009; Wilson and Martinez, 2000). Performing **IR** is important in most datasets even where all classes are mostly equally distributed, and thus more necessary in skewed distribution datasets (e.g. imbalanced classification) where the data values trail off more sharply on one side than on the others (Fernández et al., 2018a; Thabtah et al., 2020). Resampling (e.g. undersampling) is one of the critical strategies to help balance the samples among classes, thus the classifiers can perform their learning task appropriately. Handling imbalanced classification has been a rising challenge in many domains such as bioinformatics, business management, or network analysis (López et al., 2013; Haixiang et al., 2017; Zhu, Baesens, and Broucke, 2017; Chen et al., 2018). We will discuss more details of **IR** in skewed datasets in Chapter 3.

Another significant advantage of **IR** is that it can increase the feasibility of utilising ML algorithms over high volumes of data. For example, handling a large dataset (i.e. multiple gigabytes) may not be feasible in a normal individual desktop computer, but perhaps a much larger computer with tens of gigabytes of RAM (e.g. Amazon Web Services). In the Big Data context, distributed technologies and Big Data frameworks may be required to perform data mining tasks because the set is

so large and complex. If the volume of data is reduced to a manageable size while keeping the information of the original source, ML can easily perform its tasks without or with a low level of requirement of large memory and high technology. In the literature, 'Green AI' or 'Sustainable AI' has been recently introduced to refer to AI research that yields high performance results while gives consideration to the lower computational cost or resources consumed (Schwartz et al., 2019). We have aimed at these advantages for our **IR** approaches and will present them concretely at each research work in the later chapters.

It is important to highlight that though aiming at a shared objective of handling large datasets for an ML task, we focus on designing smart algorithms and try not to rely on hardware expansion or modern Big Data frameworks (Triguero et al., 2014; Triguero et al., 2015a). As a result, our proposed pre-processing techniques can be implemented in a single desktop computer to handle relatively large datasets (e.g. 20k-30k samples) in a reasonable time using an evolutionary-based search method, while it was impractical or likely impossible to execute an evolutionary search with such relatively large datasets (Triguero et al., 2017; Triguero et al., 2015a). It is worth noting that though **IG** solutions are determined in the instance space where this dissertation takes into consideration, the proposed algorithms are not limited to the instance space, but can be applicable for the feature space (i.e. feature selection and feature generation).

1.2 Motivation and Objectives

In data mining and ML, **IR** solutions have been widely researched for years and have demonstrated their effectiveness in many research fields (Brighton and Mellish, 2002; Triguero et al., 2012; Sánchez, 2004). In addition, **IR** is also considered as an important stage to turn raw data into Smart Data for knowledge discovery. The two main research directions of **IR** have been effectively resolved by evolutionary-based approaches, particularly, Steady State Memetic Algorithm (**SSMA**) for **IS** (García, Cano, and Herrera, 2008) and Differential Evolution-based algorithms for **IG** (Triguero, García, and Herrera, 2011). Despite their effectiveness, evolutionary-based approaches are typically very time-consuming, especially in large datasets, due to the computational cost associated with numerous objective function calls and memory cost due to the exhausted search with a population-based scheme. In the recent literature, several solutions have been provided to address these challenges.

- **Divide-and-Conquer:** The idea of this approach is to parallelise the execution of **IR** by splitting the training data into a number of chunks, typically through Big Data technologies, see (Triguero et al., 2017; Triguero et al., 2015b; Triguero et al., 2015a). Whilst they are necessary when the training set does not fit in

main memory, the main limitation of this approach is that it does not address the computational complexity of the problem, but only processing time, by using additional computational resources. In addition, a trade-off between the number of splits and the accuracy that can be obtained exists, and must be experimentally found for the dataset at hand.

- **Approximation:** Different fitness approximation approaches, typically surrogate models, have been investigated for problems solved by evolutionary techniques (Jin, Olhofer, and Sendhoff, 2000; Salami and Hendtlass, 2003; Jin, 2005; Jin, 2011; Rosales-Pérez et al., 2015; Sun et al., 2019; Brownlee and Wright, 2015; Chugh et al., 2020). In addition, existing methods are usually designed for problems in the continuous search landscape, while those for combinatorial domains have been under-explored (Moraglio and Kattan, 2011; Bartz-Beielstein and Zaefferer, 2017) due to the complexity of the field which requires tailored domain knowledge. In the case of IS, the quality of a reduced set (RS) was approximately estimated by windowing (Bacardit et al., 2004)) and stratification (Cano, Herrera, and Lozano, 2005). The underlying idea of these methods is to consider subsets of training data for fitness evaluation, reducing the evaluation cost on larger datasets. While this approach reduces both runtime and computation, its main limitation is that an approximated objective function may mislead the search of the optimisation algorithm depending on how representative the samples at each window are for the original source.

The above justification motivates us to develop novel strategies to deal with IR, either to speed up Evolutionary Algorithm (EA) search techniques or to propose different simpler search paradigms. The below research questions help us narrow down our focus to the core issues, whose answers will be revealed in the later chapters.

- Fitness approximation methods for **IR** solved by EA approaches is under-explored due to the huge size and the nature of the combinatorial search space (i.e. instance space). Aiming to infer the fitness of other RS(s) when knowing the actual fitness of an RS, the representation of RS(s) has to vary consistently with its associated fitness. To do this, the representation among RS(s) must hold the continuous spatial relation, so that a small variation in RS(s) will cause a smooth change in their fitness space. Since the current representation of the **IR** problem is a binary chromosome, it is necessary to find a way to transform this representation from the combinatorial space into its equivalent form in the continuous space. One question promptly appears in mind that what are the new features that can be used and to what standard that they are defined? In addition, fitness approximation can be conducted when we are able to group similar chromosomes into clusters. To do this, unsupervised

learning methods in the family of ML techniques are important to help divide candidate solutions into different groups. Thus, the entire framework we are investigating falls into the employment of an ML technique in a search method (i.e. meta-heuristic techniques) which is still under-explored (Song, Triguero, and Ozcan, 2019).

- Many studies approximate the objective function by using a subset of data, assuming that it can represent the original source. Will it be possible to approximate the objective function more accurately by using the entire training data? Other ideas are to leverage the hardware expansion to tackle large datasets, but it depends heavily on the number of available computing nodes and the supporting implemented framework. Will it be possible to design a simplified algorithm achieving competitive results as an evolutionary-based approach, so that it will be more independent from external computing resources?

- In the literature, hybrid approaches of combining **IS** and **IG** using population-based evolutionary search methods have achieved the highest accuracy performance at **IR** solutions, considering kNN as the base classifier. **IS** and **IG** performed their tasks in separated and subsequent stages. More specifically, **IS** reduces the size of the dataset while **IG** refines the results on the reduced data. This is because **IS** is usually employed to decide the best distribution of instances per class, providing a good starting point to let **IG** optimise the positions of the instances. Will it be possible to merge these two stages of a hybrid approach into one and let them synergise with each other?

Based on the motivations and research questions, we summarise the main aim of this thesis as: To develop novel strategies to accelerate different search algorithms in data reduction, which are capable of mitigating the computational cost of the objective function or simplifying the algorithmic design while maintaining state-of-the-art performance achieved. As opposed to parallelisation techniques that merely focus on reducing processing time, these strategies aim at reducing the computational complexity; thus they can save substantial runtime and be employed in Big Data frameworks at each single computing node. From the motivations and research questions referred above, we define three objectives that cover IS, **IG** and a hybrid of **IS** and IG. These three objectives will lead to the subsequent research phases deployed in the following chapters, concretely:

(i) Objective 1: Propose a novel evolutionary-based search strategy for **IS** with the integration of a clustering technique to reduce the computational cost of fitness evaluation, therefore decrease the excessive runtime consumed in an evolutionary search.

(ii) Objective 2: Propose a simplified pattern search for **IG** to compete for the complex design of an EA approach. The proposed pattern search is embedded in a memetic search, an advanced search model in the family of evolutionary-based algorithms.

(iii) Objective 3: Devise a novel single search approach which enables hybridising **IS** and IG. This search framework thus can save significantly the computational cost if **IS** and **IG** are conducted separately.

1.3 Contributions

The work presented in this dissertation focuses on different strategies to target the three mentioned objectives. The key aspect of these contributions is the capability of speeding up different search methods, as well as saving computation and storage. It is worth mentioning that all proposed techniques are applicable in addressing many real-world problems such as imbalanced classification in network intrusion, spam detection, identification of rare diseases, fraudulent transactions, etc. The contributions of this thesis are completed following a progressive order as follows:

(i) Contribution 1: In the context of binary imbalanced classification, we propose a novel strategy to integrate a clustering-based technique into an evolutionary search approach to accelerate the fitness computation. The main contribution lies at not only substantial runtime saving but also the insights about devising a surrogate model for a binary combinatorial optimisation problem (i.e. **IS** for undersampling), which is under-explored in the literature.

(ii) Contribution 2: In the continuous search space, we investigate a simple and yet effective domain-specific pattern search approach for IG. The proposed algorithm is composed of a novel domain-specific implementation of local search hybridised with a global evolutionary operator, which characterises this search framework memetic computing (MC). The search algorithm is associated with an acceleration mechanism to drastically reduce the cost of the objective function when using the NN algorithm as the base classifier. Note that the implementation of the global operator can help prevent the search from getting stuck in local optima.

(iii) Contribution 3: Stemming from the algorithmic design in contribution 2, we propose a novel solution to tackle **IR** which can work in both combinatorial and continuous search spaces. Unlike previous studies which can only address **IS** and **IG** separately in each single search domain, the newly designed algorithm can perform **IS** and **IG** within a single framework. The algorithmic design is also associated with an acceleration mechanism for the objective

function, which maintains the main theme of substantial runtime saving in this thesis.

The aforementioned contributions are part of or included in the following list of works completed during my PhD study:

1. Le, H. L., Landa-Silva D., Mikel G., Salvador G., Triguero I. "EUSC: A Clustering-based Surrogate Model to Accelerate Evolutionary Undersampling in Imbalanced Classification." Applied Soft Computing 101 (2021):107033.

   The content of this paper is covered in Chapter 3.

2. Le, H. L., Landa-Silva, D., Mikel G., Salvador G., Triguero I. "A Hybrid Surrogate Model for Evolutionary Undersampling in Imbalanced Classification." In 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1-8. IEEE, 2020.

   The content of this paper is covered in Chapter 3.

3. Le, H. L., Neri F., Triguero I. "SPMS-ALS: A Single-Point Memetic Structure with Accelerated Local Search for Instance Reduction." Swarm and Evolutionary Computation 69 (2022): 100991.

   The content of this paper is covered in Chapter 4

4. Le, H. L., Neri F., Landa-Silva, D., Triguero I. "Accelerated Pattern Search with Variable Solution Size for Simultaneous Instance Selection and Generation." The Genetic and Evolutionary Computation Conference (GECCO 2022).

   The content of this paper is covered in Chapter 5

1.4 Structure of the Thesis

The structure of this thesis is outlined as follows:

- Chapter 1 presents an introduction to the research topic of this thesis. Next, it discusses the research questions, motivations and objectives, followed by a summary of the contributions achieved through multiple work packages and an outline of the contents.
- Chapter 2 introduces the necessary background for the appropriate comprehension of multiple work packages conducted at different stages of the thesis. This includes an overview to ML, optimisation, mutual interactions of ML and optimisation, and a comprehensive literature review of accelerating the objective function evaluation. In addition, we also present data mining and different data pre-processing techniques, followed by the background of derivative-free search strategies that will be employed in the research.

- Chapter 3 describes the novel methodology of constructing a clustering-based surrogate model to accelerate an evolutionary search for IR. Considering imbalanced classification as a study case, this novel surrogate design is thoroughly examined with a wide range of skewed datasets to validate its effectiveness. This chapter also includes an extension of fitness approximation when another objective function approximation method (i.e. windowing) is combined with the proposed clustering-based surrogate model.
- Chapter 4 delves into pattern search and MC to propose a simple and effective single-point memetic structure for tackling IR. The search framework contains an accelerated mechanism to enable saving fitness computation. This novel algorithmic design is thoroughly verified over a large number of datasets and competed with state-of-the-art algorithms in comprehensive experimentation.
- Chapter 5 presents a new methodology for IR, bridging the two search domains of combinatorial and continuous in an algorithmic design. This chapter adopts the work conducted Chapter 4 and insights gained in Chapter 3 to construct a novel searching paradigm which can simultaneously handle **IS** and **IG** in a search iteration.
- Chapter 6 concludes the main body of research works with a reflection on the research gaps defined and the contributions achieved. It also analyses the limitations of the dissertation, followed by a summary of several potential lines of future work.

Chapter 2

Background and Related Work

Data mining, ML and optimisation are several noticeably growing fields of AI with an enormous number of computer science applications. The intersection of techniques in these areas is the extraction of the insights from digital data using as minimum as possible the cost of computation and resources. To make an AI system more robust, techniques in these areas are frequently hybridised, so that, one can take advantage of the others for its own process. This thesis mainly delves into enhancing the speed of different optimisation strategies, releasing the burden of expensive computation for the overall mining process.

In this chapter, we present the background and main related works which are essential to develop the content in the later chapters of this dissertation. Firstly, Section 2.1 introduces the basic concepts of ML, optimisation and an overview of the dual interactions between ML and optimisation. Focusing on the literature of the interaction of ML for optimisation, we dig further into the acceleration of objective function evaluation presented in Section 2.2. The literature review as well as the understanding discussed in this section is essential to develop a novel search framework in Chapter 3. Next, Section 2.3 presents the essentials of a typical data mining process, followed by different pre-processing techniques where there are approaches we aim to improve. After that, Section 2.4 places several fundamentals of multiple search strategies that will be used for investigation in this dissertation, including population-based meta-heuristics, and Pattern Search with MC. Finally, the contents of the chapter are summarised in Section 2.5, where they are linked to the following chapters that deploy the main body of research work.

2.1 Machine Learning and Optimisation

This section presents an overview of ML and optimisation, and their dual interaction which are the necessary background for constructing different research work packages in this dissertation. Firstly, Subsection 2.1.1 introduces ML and its categories of techniques. Secondly, Subsection 2.1.2 presents optimisation, giving the

overall picture of methods that will be used or are related to the search strategies used in this thesis. Finally, Subsection 2.1.3 discusses the interaction between the two domains.

2.1.1 Machine Learning

ML is a subset and one of the core components of artificial intelligence, which is the science of getting computers to learn and enhance this learning ability progressively using observed data (Alpaydin, 2014). A definition of ML widely used in the literature (Mitchell, 1997) by Tom Mitchell 'a machine is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E'. ML focuses on the capability of observing provided data to look for values or knowledge and improve the performance with the knowledge.

The main branches of ML, comprising supervised learning, unsupervised learning, and reinforcement learning, have been widely researched, providing numerous real-world applications. These applications can be grouped based on the desired outputs of ML tasks such as classification or regression (Erhan et al., 2014), clustering (Xu and Tian, 2015), reinforcement learning (Večerík et al., 2017), and others.

Supervised Learning: The first type is supervised learning which means learning with labelled data. It aims at inferring output to unseen data by a mapping function using prior knowledge (labelled data). Labelled data is the 'right answers' playing the role of a teacher to guide learning tasks. The term 'supervised' means that the algorithm during the learning process is observed and corrected by the right answers. Supervised learning makes use of labelled data for future prediction because of its generalisation ability which can apply previous experience to new data. Supervised learning problems can be grouped into two types, classification or regression. While classification returns a discrete label to a given instance (i.e. {yes, no}, {red, green, blue}); regression returns the outcome in the form of real values (i.e. the weight of a person or the temperature in a specific time) (Fernández-Delgado et al., 2014). Let X denote input, Y be the output and the mapping function from X to Y is $g(.)$. Given the mapping function $Y = g(X|\theta)$, the approach in supervised learning is tuning the set of parameters θ. Y can be a real number in regression or class label in the case of classification. In this dissertation, although different accelerated search strategies contribute more understanding to the growing research area of **IR** solutions, the ultimate output is devoted to kNN, a widely used algorithm of supervised learning, as well as other learning models.

Unsupervised Learning: Another type of learning where there is no corresponding output for input data, called unsupervised learning, aims to find the structure or distribution of the data. The term 'unsupervised' is opposite to 'supervised' as the algorithm during the learning process is not observed and corrected by the right answers. Unsupervised learning problems can be categorised into two types of clustering and association. The first category (Xu and Tian, 2015) divides data into similar-feature groups or 'like-minded' individuals such as colour similarity. The second type of unsupervised learning, association, aims at discovering relations between attributes represented in a majority of the dataset. Among many other unsupervised learning approaches, k-means (Lloyd, 1982; MacQueen et al., 1967) will be employed for a search methodology in Chapter 3. However, any other unsupervised learning algorithms can be examined as an alternative to k-mean.

Reinforcement Learning: Reinforcement learning (RL) is a paradigm of learning to control, which can determine which action to do within a specific context to maximise the numerical reward (Watkins and Dayan, 1992; Alpaydin, 2014). Unlike learning with a teacher - supervised mode - who tells a learner what to do, this approach learns with a critic that can respond how well it has performed so far. RL is also not unsupervised since feedback is returned. The setting of RL is a decision maker, called the agent, and environment, where the agent makes interaction. When an action is executed by the agent, it triggers a new state in the environment. In RL, as there is an absence of existing training data, the agent just conducts trial-and-error actions and finds the policy from returning feedback, which later becomes its experience. Actions may not affect the immediate reward but may affect the subsequence. A state might yield a low immediate reward but is perhaps followed by other states that yield high rewards. Therefore, the objective is to opt for an action that can maximise the total reward in the long run, not the immediate one.

2.1.2 Optimisation

Optimisation focuses on finding the optimal solution to a given problem with respect to a number of constraints (Hillier, 2012). To clarify the notation employed, we will refer to the general form of a minimisation problem (Silver, 2004):

$$\begin{aligned} &minimise\ f(x) \\ &subject\ \ to\ g_i(x) = 0,\ i = 1, ..., m \end{aligned} \tag{2.1}$$

where x represents the decision variables (i.e. either discrete, continuous or both) and $f()$ is the objective function of the examined optimisation problem. An optimisation algorithm with different search strategies seeks for the value of x that

minimises the objective function $f()$, subject to a set of m constraints denoted as g_i.

Optimisation problems can be grouped into combinatorial and continuous categories, depending on whether the type of the decision variables is discrete or continuous (Boyd and Vandenberghe, 2004; Hillier, 2012). To address these problems, various techniques have been proposed for each domain. For continuous domain, common methods include simplex algorithm, gradient-based methods (i.e. gradient descent method and its variants, Newton's method and its variants, conjugate gradient method, interior-point methods) and gradient-free methods (i.e. Bayesian optimisation, heuristics and meta-heuristics) (Andréasson, Patriksson, and Evgrafov, 2020). For combinatorial domain, common methods consist of approximation algorithms, mathematical programming, dynamic programming, gradient-free methods and hyper-heuristics (Andréasson, Patriksson, and Evgrafov, 2020).

Amongst these many strategies, some of the most prevalent methods in the literature can be briefed as follows. Gradient-based methods rely on the gradient of a differentiable objective function to search in the solution space (Ruder, 2016). Bayesian optimisation is a sequential design strategy for the global optimisation of black-box functions (Mockus, 2012). Heuristic methods are commonly problem-specific, and therefore cannot be easily used to solve other problems. It may quickly lead to a near-optimal solution, but can also get stuck there. A more advanced framework is meta-heuristic which is problem independent and can provide a set of guidelines for multiple heuristics (Sörensen and Glover, 2013). It consists of individual-based search methods (i.e. tabu search (Glover, 1989), iterated local search (Lourenço, Martin, and Stützle, 2003)) and population-based search methods (i.e. EAs (Back, Emmerich, and Shir, 2008), ant colony optimisation algorithm (Dorigo and Gambardella, 1997), particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995)). A more flexible design for selecting and generating multiple heuristics is called hyper-heuristics. This family of techniques is 'An automated methodology for selecting or generating heuristics to solve hard computational search problems' (Burke et al., 2010; Burke et al., 2013). Typically, hyper-heuristics have two separate layers: The first layer operates at the heuristic level to either select or generate a number of low-level heuristics, while the second layer is a domain-specific heuristic working at the solution space.

We would focus on EAs and their enhanced frameworks as they have been found as the best performing techniques addressing IR. Before digging into the main focus, we would present general understanding of EA-based optimisation techniques.EAs is a population-based meta-heuristic search approach inspired by biological evolution mechanisms proposed by Darwin (Sindhya, n.d.). The evolutionary-based search strategy has been preeminent to solve **IR** problems in comparison with many others such as Ranking-based Instance Selection (RIS) (Cavalcanti and Soares, 2020),

clustering-based (Saha et al., 2022; Czarnowski, 2012), support vector machine (SVM) based algorithms (Srisawat, Phienthrakul, and Kijsirikul, 2006), instance weighting (Atkeson, Moore, and Schaal, 1997; Vallejo, Troyano, and Ortega, 2010) or condensation techniques (Yen, Young, and Nagurka, 2004). This family has various forms of techniques, including genetic algorithm, evolutionary programming, evolutionary strategies, genetic programming and differential evolution (DE). In general, an evolutionary-base search strategy aims to maintain the fittest candidates, which is its principle of survival. An EA works based on the basic rule that a population of candidates are produced and they compete for survival in a changing environment. All individuals in a population are evaluated with respect to the objective value of the problem. The survival individuals are selected based on their fitness values. The search is conducted within an allocated computation budget (e.g. pre-defined number of generations, computational time) to find the individual that is best adapted.

2.1.3 Interaction of Machine Learning and Optimisation

The two fields optimisation and ML interact frequently with each other for the sake of improvement or overcoming their own limitations. In studies relevant to ML and optimisation, many investigations have been conducted where each entity can enrich the performance of the other (Curtis and Scheinberg, 2017; Jensi and Jiji, 2014; Song, Triguero, and Ozcan, 2019). The use of optimisation techniques in ML has been extensively studied for decades, but the opposite view is still under-developed, where optimisation is a target to receive benefits from data mining and ML techniques. In this section, we will present the interaction of the two domains in the literature, and focus more on the EAs as the optimisation technique to receive the advantages of ML.

In a general process of constructing a ML model, several factors having influential impact on its performance can be considered the quality of input instances/features, proper hyper-parameter setting, and loss function optimisation (Fayyad, Piatetsky-Shapiro, and Smyth, 1996a; Alpaydin, 2014). Different optimisation techniques introduced in Section 2.1.2 have played an important role to provide the best condition for a learning process. They have been widely employed to contribute to the growth of ML algorithms (Narendra and Fukunaga, 1977; Yusta, 2009; Ahmad, 2015). Though the view of using ML for optimisation is not rigorously explored, there have been approaches reported, mainly to improve the performance of hyper-heuristics and meta-heuristics. Broadly speaking, the goal of introducing ML into these optimisation frameworks is to accelerate the search speed, to improve the solution's quality or to advance the algorithmic design and thus this benefits for the search to explore and exploit more regions within a limited computational budget.

ML for Hyper-heuristic: Reinforcement learning, a branch of ML, is widely applied for heuristic selection based on the analysis of quality improvement, execution time or overall performance. Reinforcement learning is used to reward good performing heuristics and punish bad performing heuristics (Kumari, Srinivas, and Gupta, 2013; Choong, Wong, and Lim, 2018). The heuristic that has the highest performance score is likely to have a higher probability of being selected in the next iteration (Nareyek, 2003; Algethami, Martínez-Gavara, and Landa-Silva, 2019). For example, the selection mechanism of a crossover operator is associated with a reward function in (Algethami, Martínez-Gavara, and Landa-Silva, 2019). This function updates the score of all operators relying on two factors Fitness and Distance. If there is an improvement in the fitness value, the score also goes up, which indicates a reward for the performing operator. Likewise, if there is a decrease in the fitness value, the score goes down, indicating a pelnaty applied for the performing operator.

ML for Meta-heuristic: Other ML techniques (i.e. clustering, supervised learning) have been attempted to enhance different aspects of the optimisation framework. In this section, EAs and their enhanced frameworks are our main focus. Various EAs with the integration of ML techniques have been reported (Jin, 2011; Zhang et al., 2011; Shi and Rasheed, 2010) including statistical methods, interpolation and regression, clustering analysis, artificial neural networks, support vector machines, and Bayesian network. These techniques have been incorporated into variants of EAs in a number of approaches, and place significant impacts on many aspects of EAs. ML techniques can be investigated based on the processing stages of an EA which are population initialisation, fitness evaluation, population diversification, or algorithm adaptation.

For population initialisation, ML help initialise high quality solution with different approaches (Diaz-Gomez and Hougen, 2007; Poikolainen, Neri, and Caraffini, 2015). For example, orthogonal experimental design (OED), a kind of statistical technique can generate an initial population whose solutions are scattered uniformly over the feasible region (Leung and Wang, 2001). Given a problem constructed with multiple factors and multiple choices per factor, an OED method explores the search space comprehensively using only orthogonal combinations rather than all the combinations of factor choices. Clustering techniques (k-means) is used for the initialised step in a genetic algorithm (GA) to solve a symmetric travelling salesman problem (TSP) (Deng, Liu, and Zhou, 2015). k-means is used to break a large-scale TSP into small problems and then an algorithm computes the optimal path of each partition. GA is carried out to achieve a globally optimal path by randomly rewiring each local optimal solution.

Algorithm adaptation shares a number of similarities with the aforementioned research topic: ML for hyper-heuristics. Regarding the adaptability of EAs, various approaches have been found to produce smarter EAs including parameter and operator adaptation. Algorithm adaptation is about opting for one or multiple parameters/ operators from a pool of candidates adaptively. Parameter control involves changing parameter values with respect to a feedback from the search (Rijn et al., 2015), or attempting to vary parameter values corresponding to the current state of the search (Zhang, Chung, and Lo, 2007). Population size is also the parameter considered to be varied in many studies, in which different values can strongly affect the efficacy of the search (Smith and Smuda, 1993; Algethami and Landa-Silva, 2017). Operator adaptation is a mechanism to better select the right operator at a certain stage of the search. This is because different operators can have different performances and they also perform differently at distinct stages of the search. This research direction is closely similar to ML for hyper-heuristic, see mutation operator adaptive selection (Zhang and Lu, 2008) or both mutation and crossover operators adaptive selection (Magyar, Johnsson, and Nevalainen, 2000). Combined approaches of updating not only parameters but also operators are presented in (Algethami and Landa-Silva, 2017; Algethami, Martínez-Gavara, and Landa-Silva, 2019).

In addition, ML techniques are introduced to maintain the population diversity during the evolutionary development. Clustering has been a popular ML technique as its task is to separate the whole population into different areas based on the individuals' inherent features. The individuals with similar features are stored in the same cluster, and they compete with each other for survival without interference to the counterparts in other regions. As different clusters naturally contain different information about the population, interactions among these clusters can generate a new population that maintains the variation. Several studies have been found in this direction including (Streichert et al., 2003; Aichholzer et al., 2002).

However, an evolutionary population-based search technique always maintains a number of candidates to explore and exploit, the computational cost of evaluation a population through different iterations is the most noticeable. ML for objective function evaluation is thus also important and speed up the search and save computational expense. The next section will present in detail this line of research, which is the key background to develop the work in Chapter 3. Apart from dual-interaction between ML and optimisation, self-interaction has been found in various studies at which the use of other techniques in a domain assists a technique in the same domain. Summary of these types of interactions was comprehensively discussed in a recent review study (Song, Triguero, and Ozcan, 2019).

2.2 Accelerating the Objective Function Evaluation

In many optimisation problems, the evaluation of a solution may have a high computational cost due to the function's complexity or massive calculation. In the literature, numerous studies have been performed to speed up fitness evaluations (Jin, 2005; Buche, Schraudolph, and Koumoutsakos, 2005; Jin, 2011). Broadly speaking, we can find delta evaluation approaches and fitness approximation. Delta evaluation is a way of computing only the different parts between two solutions (Bianchi and Dorigo, 2006). It can make use of previously evaluated similar regions and reuse those parts in the evaluation of a new individual. The strategies of delta evaluation are based on analytical computation to identify which part in the expression needs re-calculating. For example, in a timetabling problem (Ross, Corne, and Fang, 1994), instead of evaluating every timetable as only small changes are made between one timetable and the next, it is possible to merely compute the changes and update the previous cost with the value of that calculation.

Extensive studies have been proposed in the family of fitness approximation from a simple approach like fitness inheritance to advanced techniques like machine learning methods (Jin, 2005; Shi and Rasheed, 2010; Jin, 2011). Fitness inheritance is initially inspired by the idea that an offspring can also inherit a fitness value from its parents, not only its own genes. Thus, its quality can be obtained from where it derives from instead of through a function. Two classical approaches for fitness inheritance (Smith, Dike, and Stegmann, 1995) are the averaged inheritance (adopt the average fitness of its parents) and proportional inheritance (fitness is weighted based on the amount of genetic material taken from each parent). These ideas were later further investigated on several studies using for example fitness sharing in multi-objective optimisation problems (Sastry, Goldberg, and Pelikan, 2001; Bui, Abbass, and Essam, 2005), or using fitness inheritance with Bayesian optimisation (Pelikan and Sastry, 2004).

ML techniques such as clustering and supervised learning can be used in numerous ways to alleviate fitness evaluation. Clustering algorithms including hierarchical clustering, partition clustering, and overlapping clustering, typically aim to decrease the number of original function evaluations (Xu and Tian, 2015). These approaches split the entire population (based on the chromosome representation) into a number of groups by a clustering algorithm, and then the chromosomes closest to the clusters' centres are evaluated by the exact function, while other cluster members are approximated according to their distance to the evaluated solutions (Kim and Cho, 2001; Jin and Sendhoff, 2004; Martínez-Estudillo et al., 2005). However, this approach is mostly applicable for continuous optimisation problems only, and still challenging in combinatorial search space due to the problem of computing

the correlation among solutions to interpolate the fitness value (Bartz-Beielstein and Zaefferer, 2017).

Supervised learning techniques aim to create a surrogate model that can approximate the fitness function by prediction. The model is adjusted based on the known data points accumulated from the evaluation history. Naturally, most surrogate models are assumed spatial models which means the prediction task is about exploiting accepted spatial relations such as a smooth change in a response surface between the fitness values of a query point and known data-points (Li et al., 2008). In other words, a data-driven model is constructed with the assumption that there is continuity among data points, at which a small variation in decision variables will cause a smooth change in the response space. This makes surrogate models naturally suited to continuous optimisation problems, while not easily applicable to combinatorial optimisation problems (Moraglio and Kattan, 2011) because the response in combinatorial space does not necessarily vary smoothly when the discrete variables produce a minor variation. Hence, choosing an appropriate metric to express the correlation between chromosome representation and its quality has been a difficult task, which makes combinatorial landscape analysis significantly challenging. More discussion can be found in (Moraglio and Kattan, 2011; Bartz-Beielstein and Zaefferer, 2017).

2.3 Data Mining and Data Pre-processing

Data mining is considered the process of extracting useful information from a vast amount of data. Nowadays, it is qualified as science and technology for exploring data to discover the existing patterns which may be unknown to users (García, Luengo, and Herrera, 2015). Although a KDD process is widely used as a synonym of data mining, there exist other views considering data mining as the main step of KDD (Han, Pei, and Kamber, 2011; Witten and Frank, 2017). DM is used to discover useful patterns in the data, searching for meaning and relevant information for different businesses.

Data mining tasks can be different depending on different purposes such as description, estimation, classification, prediction, clustering and association (García, Luengo, and Herrera, 2015). Description group finds ways to describe patterns and trends lying within the data. Estimation approximates the numeric value of a target variable given a set of predictor variables. Classification is similar to estimation but the target variable is categorical. Prediction produces the output in either numeric or categorical forms like estimation and classification, respectively, but the results lie

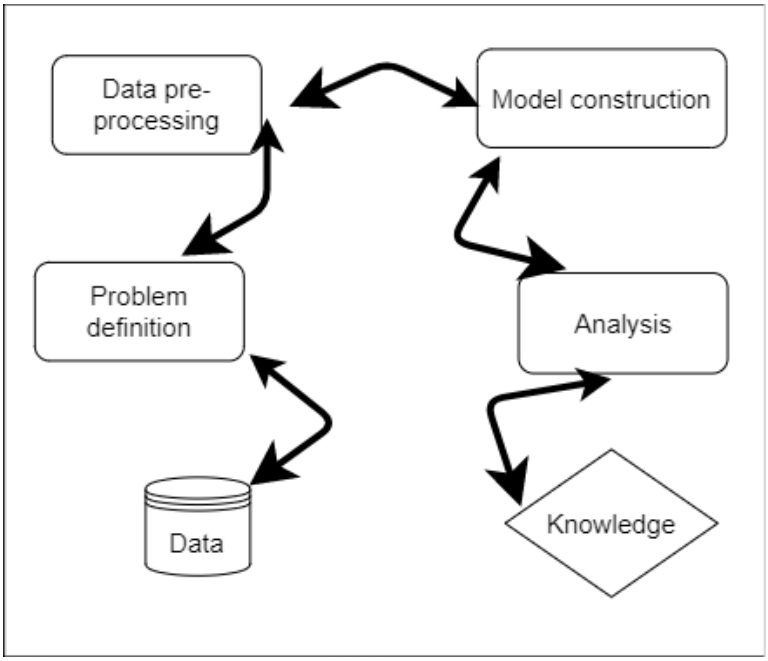


FIGURE 2.1: Multiple stages of a data mining process. The workflow in the chart is inspired by a data mining diagram in (García, Luengo, and Herrera, 2015).

in the future. Clustering refers to categorising observations into multiple groups using a similarity measure. Finally, association seeks for which attributes 'go together' with others.

2.3.1 Data Mining Process

A data mining process is usually divided into multiple stages, displayed in Figure 2.1. Although these stages may be named or split slightly different with some advantages and disadvantages, they reflect the same process and contain the below four stages (Han, Pei, and Kamber, 2011; García, Luengo, and Herrera, 2015). A key aspect that characterises these stages is their interconnection, meaning that a data mining process is actually a self-organised scheme where each phase shapes the other remaining stages and the reverse direction is also enabled.

- Problem definition: The stage requires the design of the application domain, and relevant knowledge from experts in the field. It also includes the comprehension of the selected data associated with the expert's knowledge to pinpoint the problem that is needed to be solved.
- Data pre-processing: This stage operates data cleaning (i.e. noise removal, missing values) and data transformation (i.e. normalisation, data reduction).
- Model construction: This stage involves the technical solution that will be used to mine the data, including the algorithm selection, hyper-parameter tuning

and validation procedures. The model construction may be influenced by the data mining task (i.e. classification, regression, clustering or association).

- Analysis: This phase is involved with human actions to interpret the results obtained into actionable insights.

In our rapidly growing digital world, many new technical terms and phrases have been introduced and those get us overwhelmed or lose track. Several terms have been used interchangeably, unaware that the words mean two different things, for example 'data mining' and 'machine learning'. This is because the shared characteristics between them are vast, leading to a blurred boundary between the two terms. Particularly, under the aegis of data science, both processes are involved in making use of data for solving different complex problems, so consequently, people erroneously employ the two terms interchangeably. ML is sometimes used as a means of conducting useful data mining and data gathered from a stage of data mining can be used for ML to learn. Furthermore, both processes usually employ the same critical algorithms for discovering data patterns, leading to even a more blurred boundary between ML and data mining. Despite the mentioned similarities, there are a considerable number of differences between them.

Firstly, they do not share the same purpose. Data mining is a designed method to extract the rules or to determine a particular outcome from large quantities of data, while ML teaches a computer how to learn and comprehend the given parameters. Secondly, the human factor is severely involved in data mining whereas ML only require human for setting up the initial. Data mining relies on human intervention because it is a static procedure and follows pre-set rules, ultimately created for use by people, while ML exists for the reason that it can teach itself through the data and thus not rely on human actions. As a result, data mining cannot learn or adapt, but that is the essence of ML. Lastly, data mining is a process that embodies two components: the database and ML. The former provides data management techniques, while the latter processes data analysis. Hence, while data mining needs ML to accomplish its purpose, and information gathered and processed via data mining can then be used to help a machine learn. However, it is not a necessity in the opposite direction as ML does not necessarily need data mining to perform its task.

2.3.2 Pre-processing Techniques

Once some basic concepts and processes of data mining have been introduced, the next step is to question the data to be used. Different conditions have to be set for the input data such as quantity, structure and format. Unfortunately, multiple negative factors such the presence of noise, missing values, label inconsistence, outliers, and a huge number of samples or features or both strongly impact the quality of

real-world databases, resulting in low-quality data mining performance. For this reason, data-preprocessing is important to enhance the quality of the input data and eventually improve the data mining performance. In this section, we will focus on the general categorisation of data-preprocessing methods in which our dissertation will focus on one of them.

To gain the most from the available dataset, data pre-processing has been conducted to minimise the garbage that gets into a mathematical learning model, so that we can maximise the insights that the learnt model can generate (Larose and Larose, 2014). Data pre-processing includes different tasks, grouped into data preparation and data reduction. The former group consists of data cleaning, data transformation, data integration, data normalisation, missing value imputation, noise identification, while the latter comprises data discretisation, **IR** and feature selection (García, Luengo, and Herrera, 2015). This categorisation can highlight the raised importance of data reduction set of techniques which has recently gained a significant interest in the ML community (Larose and Larose, 2014; García, Luengo, and Herrera, 2015). Data preparation is normally a mandatory step to produce correct input data for the downstream mining process, whilst data reduction aims to maintain the essential structure and integrity of the original data with a downsized amount.

As this dissertation digs into solutions for data reduction solutions, we would narrow down the literature review into this research topic. Data reduction consists of simplifying the domain of an attribute (discretisation), reducing the dimensionality (feature reduction), and reducing the number of instance (instance reduction) (Liu and Motoda, 2007; García, Luengo, and Herrera, 2015; Zebari et al., 2020). Discretisation procedure transforms numerical attributes into discrete values, reducing the value spectrum and obtaining a non-overlapping partition of the continuous domain. Feature reduction is a set of techniques that aim to minimise set of attributes used for learning. Different solutions have been proposed in the literature including feature selection, feature generation and feature weighting. Feature selection seeks for a representative subset of features from the original feature space, while feature generation allows the modification of the internal values to generate new features that can fill in gaps in data. Feature weighting schemes assign a weight to each feature so that the computation of distance between samples is adjusted. Similar to solutions for feature reduction but at the instance level, **IR** is devoted to find a reduced set of samples that represents the original training data. It can be divided into **IS** and **IG** depending on how it creates the reduced set. **IS** attempts to choose a subset of the original training data, while **IG** generates new artificial instances to better represent the decision boundaries between classes. In this manner, **IG** solutions can supply new representative examples that has not existed in some regions in the domain of the problem.

2.4 Search Algorithms for IR

IR is a challenging real-world problem in the field of data science due to the large number of variables (i.e. instances to be considered for selection/discard or huge size of features for perturbation). In the literature, it can be modelled as an optimisation problem with discrete or continuous decision variables depending on the search strategy (i.e. selection or generation). Due to its complexity, it does not allow the application of gradient-based algorithms, see (Caponio et al., 2007) and the fitness landscape could be highly multimodal. One key reason that a gradient-based approach may be impractical is that the numerical computation of the gradient is highly expensive, which results in an unacceptable overhead. In this circumstance, a derivative-free method may be a highly accepted option (Brent, 1973; Conn, Scheinberg, and Vicente, 2009) to tackle such a challenge. Derivative-free methods are algorithms that search for the closest local optimum solution but do not necessarily require computing the gradient of the objective function. Among many strategies, the following four groups can be considered the most prevalent in the literature (Neri and Rostami, 2021):

- Population-based meta-heuristics are an iterative master process that controls the sequence of intensification and diversification phases to explore and exploit efficiently a pool of different solution candidates to produce efficiently high-quality solutions. The process may contain several subordinate low-level heuristics (e.g. crossover, mutation) tailored to a specific solving problem.
- Line search methods iteratively identify a unidimensional direction trans-passing the n-dimensional space, and then optimise the objective function along this direction, see (Box, 1969).
- Pattern search methods use different defined rules to sample a set of potential candidate solutions to cover the entire search space. The rules used for generation are called a pattern, see (Kaupe Jr, 1963).
- Simplex methods search for the optimum based on geometry. They make use of the geometric figure to set the exploratory rules, see (Nelder, 1965)

This thesis will delve into the enhancement of search strategies in the two micro-groups of population-based meta-heuristics and pattern search. Subsection 2.4.1 presents population-based meta-heuristics, Subsection 2.4.2 discusses pattern search and MC.

2.4.1 Population-based Meta-heuristics

Two frameworks of population-based meta-heuristics used in our research are EAs and MC. Both optimisation schemes have been proved effective to tackle **IR** problem and others in numerous studies (Lin and Gen, 2018; Wong, 2017; Reina et al., 2016). From the general understanding of EAs search strategy presented in Section 2.1.3, MC approaches can be considered as further development of EAs when they combine an EA and a local search component (Neri and Cotta, 2012). However, MC is not limited to that only design, but can be flexible and versatile optimisation frameworks (Neri and Cotta, 2012). For example, it can be a hybrid algorithm combining having two or more algorithms joined together in a synergistic manner (e.g. a global search operator integrated with one local search) (Ma et al., 2020; Ting et al., 2018; Ma et al., 2019). We will discuss more details the background of MC in Section 2.4.2, where it is used together with Pattern Search. Many studies implementing the ideas of EAs and memetic algorithms have been proved to hold the highest performing records for **IR** considering both accuracy and reduction rate. In particularly, **IR** solutions with evolutionary search include PSO (Zhan et al., 2009; Nanni and Lumini, 2009; Cervantes, Galván, and Isasi, 2009), DE (Triguero, García, and Herrera, 2010; Triguero, García, and Herrera, 2011), Factor Local Search Differential Evolution (SFLSDE) (García, Cano, and Herrera, 2008; Triguero, García, and Herrera, 2011) and **IR** solutions with memetic algorithms are Steady State Memetic Algorithm (SSMA), SSMA-SFLSDE (Triguero, García, and Herrera, 2011).

2.4.2 Pattern Search and Memetic Computing

The content presented in this section is intended to Chapters 4 and 5 at which we will propose two novel search strategies for **IR** to achieve Objectives 2 and 3. This section is devoted to deliver key backgrounds of the two above ideas, including Pattern Search (Part A) and MC (Part B).

Part A: Pattern Search

To clarify the notation used for the pattern search and **IR** solutions, we would begin this section with important mathematical definitions of **IR** and its objective function formulation, which will be used throughout in later Chapters. These concepts are the essential building blocks to construct the Single-Point Search algorithm.

Mathematical Definition of IR Solutions:
Given an instance I having m features and belonging to a class w.

$$I = a_1, a_2, \ldots, a_m \tag{2.2}$$

In a supervised classification problem, the data is usually split into training (**TR**) and test (**TS**) sets. Each instance belongs to a class w, which is known for **TR** and unknown for **TS**. Both datasets can be viewed as a matrix in which instances $\mathbf{I_i}$ are displayed on the rows whilst their features are shown on the column f_j. $\mathbf{a_{ij}}$ indicates the j^{th} feature value of instance $\mathbf{I_i}$. Given a binary classification problem where an instance $\mathbf{I_i}$ belongs to one of the two classes $\{w_0, w_1\}$.

$$\mathbf{TR} = \begin{bmatrix} & \mathbf{f_1} & \mathbf{f_2} & \ldots & \mathbf{f_m} & \mathbf{w} \\ \mathbf{I_1} & a_{11} & a_{12} & \ldots & a_{1m} & w_0 \\ \mathbf{I_2} & a_{21} & a_{22} & \ldots & a_{2m} & w_0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \mathbf{I_l} & a_{l1} & a_{l2} & \ldots & a_{lm} & w_1 \end{bmatrix} \tag{2.3}$$

The main purpose of an **IR** technique is to clean and compress **TR** into a reduced set **RS**, by either selecting or generating new representative instances, so that, it preserves and provides valuable information for a machine learning algorithm to learn useful insights about a classification problem. Thus, the resulting **RS** should satisfy several conditions such as well-representing the distributions of the classes, significantly reducing in size to minimise the required storage, which would be beneficial to the posterior classification phase. An **RS** is a matrix having p rows ($p \ll l$) and m columns. Note that p is a parameter whose value signifies the compression of the data with respect to l (i.e. the number of rows of **TR**). Hence, the reduction rate is defined as $\frac{l}{p}$. In both matrices **TR** and **RS** each row is associated with its class label, that is each instance $\mathbf{I_i}$ is assigned to its class on the basis of its features.

$$\mathbf{RS} = \begin{bmatrix} & \mathbf{f_1} & \mathbf{f_2} & \ldots & \mathbf{f_m} & \mathbf{w} \\ \mathbf{I_1} & b_{11} & b_{12} & \ldots & b_{1m} & w_0 \\ \mathbf{I_2} & b_{21} & b_{22} & \ldots & b_{2m} & w_1 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \mathbf{I_p} & b_{p1} & b_{p2} & \ldots & b_{pm} & w_0 \end{bmatrix} \tag{2.4}$$

Overall, the two general processes of **IS** and **IG** are summarised by the below two schematic designs. Figure 2.2 displays an example of an **IS** process where instances I_1, I_3 & I_9 are selected to remain in the final RS. Note that none of the features of those selected samples is modified. Figure 2.3 shows an example of an **IG** process

$$\mathbf{TR} = \begin{bmatrix} & \mathbf{f_1} & \mathbf{f_2} & ... & \mathbf{f_m} & \mathbf{w} \\ \mathbf{I_1} & a_{11} & a_{12} & ... & a_{1m} & w_0 \\ \mathbf{I_2} & a_{21} & a_{22} & ... & a_{2m} & w_0 \\ ... & ... & ... & ... & ... & ... \\ \mathbf{I_l} & a_{l1} & a_{l2} & ... & a_{lm} & w_1 \end{bmatrix} \xrightarrow[\text{Select } I_1, I_3 \text{ \& } I_9]{\text{An } \mathbf{IS} \text{ process}} \mathbf{RS} = \begin{bmatrix} & \mathbf{f_1} & \mathbf{f_2} & ... & \mathbf{f_m} & \mathbf{w} \\ \mathbf{I_1} & a_{11} & a_{12} & ... & a_{1m} & w_0 \\ \mathbf{I_3} & a_{31} & a_{32} & ... & a_{3m} & w_1 \\ \mathbf{I_9} & a_{91} & a_{92} & ... & a_{9m} & w_0 \end{bmatrix}$$

FIGURE 2.2: Example of an **IS** process where instances I_1, I_3 & I_9 are selected to remain in the final RS. None of the features is modified.

$$\mathbf{TR} = \begin{bmatrix} & \mathbf{f_1} & \mathbf{f_2} & ... & \mathbf{f_m} & \mathbf{w} \\ \mathbf{I_1} & a_{11} & a_{12} & ... & a_{1m} & w_0 \\ \mathbf{I_2} & a_{21} & a_{22} & ... & a_{2m} & w_0 \\ ... & ... & ... & ... & ... & ... \\ \mathbf{I_l} & a_{l1} & a_{l2} & ... & a_{lm} & w_1 \end{bmatrix} \xrightarrow[\text{Select \& Modify}]{\text{An } \mathbf{IG} \text{ process}} \mathbf{RS} = \begin{bmatrix} & \mathbf{f_1} & \mathbf{f_2} & ... & \mathbf{f_m} & \mathbf{w} \\ \mathbf{I_1} & a'_{11} & a'_{12} & ... & b_{1m} & w_0 \\ \mathbf{I_2} & a_{21} & a_{22} & ... & a_{2m} & w_1 \\ \mathbf{I_5} & a_{51} & a'_{52} & ... & a_{5m} & w_1 \end{bmatrix}$$

FIGURE 2.3: Example of an **IG** process where instances I_1, I_2 & I_5 are randomly selected and then features a_{11}, a_{12}, and a_{52} are modified into a'_{11}, a'_{12} and a'_{52} respectively.

where instances I_1, I_2 & I_5 are randomly selected and then feature values of these samples are modified. Specifically, features a_{11}, a_{12}, and a_{52} are modified into a'_{11}, a'_{12} and a'_{52}, respectively.

Objective Function Formulation:

As discussed in Chapter 1, the development of many data pre-processing techniques such as instance reduction was initially motivated by the imprecision and inefficiencies of the well-known nearest neighbour(s) (NN) algorithm (Cover and Hart, 1967). These weaknesses have turned into strengths and made the NN rule a core algorithm to preprocess raw data (Triguero et al., 2019). Thus, most **IR** techniques verify how well a candidate matrix **RS** represents the entire training dataset, **TR**, by using the NN algorithm as base classifier. To do so, we essentially check how well we can classify the large dataset **TR** using the small dataset **RS** as training data. The Euclidean distance between each instance $\mathbf{I_i}$ (row vector) of **RS** and each instance $\mathbf{I_j}$ (row vector) of **TR** is calculated. This process yields $l \times p$ distance computations. Figure 2.4 shows an example of a distance matrix. An entry $D_{i,j}$ of the distance matrix in position i, j indicates the distance of the i^{th} instance in **TR** to the j^{th} instance in the **RS**:

$$D_{i,j} = \sqrt{\left(b_{i,1} - a_{j,1}\right)^2 + \left(b_{i,2} - a_{j,2}\right)^2 + \ldots + \left(b_{i,m} - a_{j,m}\right)^2}. \tag{2.5}$$

		1	2	3	--	p
Distance matrix	1	0.55	0.12	0.85		1.2
	2					
	3					
	--					
	l	0.21	1.02	3.2		0.98

FIGURE 2.4: Distance matrix of l instances in **TR** and p instances in **RS**. The instance at the first row is verified by instance at column 2, while the instance at the last row is checked by the one at column 1. Blue entries represent the shortest distance among the neighbours.

When the distance matrix **D** is calculated, for each row (i.e. each instance of **TR**), the smallest entry is detected, e.g. 0.12 in the first row of Figure 2.4, and that instance is given the class label w of the closest instance in **RS**. When all instances in **TR** have been classified, there are different ways to assign a score (objective function) to the performance of **RS** (Alpaydin, 2014; Witten and Frank, 2017). In balanced datasets, Accuracy Rate Acc (Alpaydin, 2014; Witten and Frank, 2017) is an appropriate measure, while other scenarios such as imbalanced classification would require a different metric.

$$Acc = \frac{\text{number of correct classifications by means of } \mathbf{RS}}{\text{total number of examined samples}} \tag{2.6}$$

Thus, for an input **RS** the objective function value is Acc, that is

$$f(\mathbf{RS}) = Acc. \tag{2.7}$$

Algorithm 1 describes step-by-step the calculation of the objective function based on the distance matrix.

Of course, since higher values of Acc correspond to a better classification, the objective function needs to be maximised. The variable space is now $(p \times m) -$ dimensions where each one can continuously vary in a normalised interval $[0, 1]$. Hence, the search space for the optimal solution occurs in the set $[0, 1]^{p \times m}$.

Regardless of the score used to measure the quality of **RS**, the procedure described above requires the calculation of $l \times p$ Euclidean distances. This operation can be computationally expensive especially when large datasets are under examination. The two main problems that arise when tackling larger datasets are runtime (due to the large number of distance computations required) and memory consumption (e.g. when the size of **TR** does not allow us to store it in main memory). However,

Algorithm 1 Objective Function

1: **INPUT** matrices **TR** $= [a_{i,j}]$ and **RS** $= [b_{i,j}]$
2: Build the matrix of Euclidean distances **D** $= [D_{i,j}]$ (Equation 2.5)
3: correct_classification = 0
4: **for** each row of the matrix **D do**
5: Find the smallest value and save its row and column indices
6: Select, from **TR** and **RS**, the instances corresponding to the calculated indices
7: Check the corresponding labels
8: **if** the labels coincide **then**
9: correct_classification += 1
10: **end if**
11: **end for**
12: Calculate *Acc* (Equation 2.7)
13: **OUTPUT** the objective function value *Acc*

the required runtime to evaluate candidate solutions tends to be the most important factor to enable **IR** of large datasets.

Single-Point Search:
Pattern Search is employed in the continuous domain to tackle **IG** problem at which we aim to to maximise the objective function $f(x)$ defined at Equation 2.6. From an **RS** displayed in Equation 2.4, we flatten the 2-dimensional matrix into a 1-dimensional vector. As a result, the candidate solution x is 1-dimensional and composed of n design variables (i.e. n is the multiplication of p rows with m features), see Equation 2.8. We search for the optimal solution in an n-dimensional space by perturbing each single value of n design variable. The perturbation follows the single-point pattern search at which the new candidate generated from solution x is only different at one single feature value.

$$\begin{aligned} \mathbf{x} &= (b_{11}, b_{12}, \ldots, b_{1m}, b_{21}, b_{22}, \ldots, b_{2m}, \ldots, b_{p1}, b_{p2}, \ldots, b_{pm}) \\ &= (x_1, x_2, \ldots, x_n) \end{aligned} \tag{2.8}$$

where x_i represents the design variable of the feature perturbation process and $n = m \cdot p$.

Part B: Memetic Computing

MC is a broad family of techniques which will be employed in Chapters 4 and 5. Since their earliest definition (Moscato and Norman, 1989; Moscato, 1989), Memetic Algorithms (MAs) were introduced usually to enhance upon the performance of

algorithms, such as Genetic Algorithms and Simulated Annealing. Unlike the majority of other algorithms, MAs are not fixed to a specific structure but are flexible and thus versatile optimisation frameworks, see (Neri and Cotta, 2012). This flexibility is one of the main features of MAs which likely inspired numerous subsequent studies that shaped, over the past three decades, the field of MC.

By following the visionary ideas reported in (Gupta and Ong, 2019) and the classification in (Chen et al., 2011), three groups/generations of MC approaches have been identified:

- **Simple Hybrids**: this group includes hybrid algorithms generated by two or more algorithms joined together in a synergistic manner. Usually, the algorithms of this type combine a global search and at least one local search. Some examples of successful hybridisations are reported in e.g. (Ma et al., 2020; Ting et al., 2018; Ma et al., 2019)
- **Adaptive Hybrids**: this includes hybrid algorithms where multiple local search algorithms are coordinated by an adaptive mechanism that selects the algorithmic elements at runtime. Popular selection criteria are performance-based like in hyper-heuristics (Özcan, Bilgin, and Korkmaz, 2008) and meta-Lamarckian learning (Le et al., 2009; Nogueras and Cotta, 2016), diversity-based (Caponio et al., 2007) or self-adaptive (Nguyen et al., 2009).
- **(Future) Memetic Automation**: this kind reinterprets MAs as a combination of 'agents' without a predefined structure (Acampora, Loia, and Gaeta, 2010; Zhu, Jia, and Ji, 2010) and investigates mechanisms to attain fully self-generated MAs. Although this design approach is still under investigation, some interesting domain-specific frameworks (Feng et al., 2015; López-García et al., 2016) and prototypes (Caraffini, Neri, and Epitropakis, 2019) have been proposed.

The flexibility of the subject facilitating domain-specific algorithmic design is one of the reasons of the success of MAs in real-world applications, see (Amaya et al., 2020; Elola et al., 2017; Zaher et al., 2019). In other words, while robust algorithmic design and testing on multiple abstract mathematical functions is fundamental for the development of novel memetic structures (as well as for any optimisation algorithm) (Fister et al., 2019; Martinez et al., 2019) real-world problems often pose specific challenges which may be addressed by ad-hoc representations and specific operators (Gupta and Ong, 2019). Among the plethora of MC structures the need to design simple algorithm on a limited hardware inspired **Single-Point Memetic Structures** which are the focus of the present study. For example, in (Neri, Iacca, and Mininno, 2011) memetic structures using virtual populations (statistical models of populations) have been implemented directly in the control cards of robots. In (Caraffini et al., 2014), a simplistic single-solution MC approach composed of a

global evolutionary operator and a local search has been proven to be competitive with complex meta-heuristics and has been successfully implemented in the control card of an helicopter robot.

The two search frameworks that will be presented in Chapters 4 and 5 are part a family of MAs designed according to the so-called *Ockham's Razor in MC* principle formulated in (Iacca et al., 2012): simple algorithmic structures designed by combining memes in a bottom-up approach while addressing the knowledge of the problem (prior or available at run-time) often have a high performance despite their simplicity. This idea links to other areas of optimisation research such as the pioneering studies in (Yao, Liu, and Lin, 1999; Chang-Yong Lee and Xin Yao, 2004) and the work on Fitness Landscape Analysis (Malan and Engelbrecht, 2013; Jana, Sil, and Das, 2018).

In Chapters 4 and 5, we will employ MC to design search frameworks tailored to **IR** problem. The details of each algorithmic design will be presented in each chapter, but they generally join the Single-Point Search with one or more other algorithms to synergise the effectiveness of different integrated components. Different from a population-based approach generating a pool of candidates and perturbing multiple variables to search at different points at once, this search strategy maintains only a single solution and seeks improvement upon its performance (objective function value). While the search moves along the axis and perturbs the elements of a candidate solution one by one, it exploits the fact that a sing-point search produces a new candidate that is only 'slightly' different with respect to its previous state (Neri and Rostami, 2021). That is why this search strategy belongs to the continuous domain and is named **Single-Point Memetic Structure**.

2.5 Summary

This chapter has supplied the necessary background for an adequate comprehension of the conducted research throughout the thesis. We have covered different aspects of ML, optimisation, accelerating an objective function, data mining and search algorithms to set up the foundation for understanding different advanced **IR** solutions. The comprehensive literature review in these sectors has reinforced us to understand better the research gap and thus allows us to place our contributions to where necessary. From the literature review, the main issue for current **IR** solutions is largely related to the high cost of evaluating candidate solutions. When tackling bigger datasets, their runtime may become excessive and we can find in the specialised literature parallelisation approaches for **IR** (Triguero et al., 2015b), which allow them to be executed, whilst increasing the need for additional computational resources. Reducing the computational cost of the fitness evaluation is

an under-explored area in IR, and just a few approximation approaches exist (e.g. windowing (Bacardit et al., 2004) or surrogate models (Neri and Triguero, 2020)). Hence, the contributions of this thesis is to provide several novel search strategies to tackle the high cost of objective function evaluation. Different search strategies require different backgrounds, which can be summarised as follows:

Sections 2.1, 2.2 and 2.3 are essential prior to clarifying the development of the methodology to deal with **IS** problem presented in Chapter 3, which is our first contribution as presented in Objective 1. Next, basic concepts about search pattern and MC in Section 2.4 will be used to support the single-point memetic search structure, proposed to tackle **IG** problem in Chapter 4. This chapter presents another contribution to fulfil Objective 2. Finally, the findings from Chapter 4 will take part in a novel algorithmic design for **IS** and IG, presented in Chapter 5.

This chapter has paved the way for the experiments and discussions deployed in the following chapters. The investigation for enhancing different search strategies will be based on the concepts and methods introduced above. As a result, the research gaps, motivation and objectives identified in the introduction now become more conceivable. The dissertation is now followed by our contributions with validated solutions corresponding to the presented research gaps. In the next chapter, we aim to address the first research question, how to improve the evolutionary search strategy with the integration of an ML method.

Chapter 3

A Surrogate Model for Accelerating Evolutionary Undersampling

3.1 Introduction

As we agreed that the main theme of the dissertation is the acceleration of different search strategies. In the first contribution presented in this chapter, we discuss the acceleration for a population-based meta-heuristic search strategy for imbalanced classification problem as a case study, while other search strategies will be discussed as we progress towards later chapters. This is because the **IS** solution employed at the data pre-processing stage, called undersampling, is found the state-of-the-art method for this particular type of problem. This state-of-the-art undersampling approach modelled the selection of samples as an binary optimisation problem and employed a meta-heuristic search framework to tackle the problem. As discussed earlier, a typical problem associated with a meta-heuristic search is fitness computational expense, which demands a solution to reduce the cost.

Furthermore, when it comes to **IR** in general at which samples of all classes will be considered for elimination, **IS** approaches do not hold the state-of-the-art performance but its hybridisation with **IG** techniques. Hence, we investigate imbalanced classification at the first stage of the research. We argue that imbalanced classification is a typical case study found to demonstrate our first contribution established in Objective 1. In addition, it may be less complex to start with because **IS** solutions for imbalanced classification only consider to reduce samples at the majority class. The content below will discuss in details imbalanced classification and is organised as follows. First, we introduce the problem as well as justify why learning with skewed datasets is different from learning with balanced datasets, then analyse different solutions proposed in the literature to tackle this problem. Next, we delve into the bottleneck of the state-of-the-art undersampling solution, and discuss our proposed solution for the problem.

3.1.1 Imbalanced Classification

Learning from skewed data is a challenge arising in multiple domains such as bioinformatics, business management, or network analysis (Dai, 2015; Zhu, Baesens, and Broucke, 2017; Chen et al., 2018). Focusing on two-class datasets, the problem happens when samples from the minority class (usually the class of interest) are highly outnumbered by the counterparts from the majority class (He and Garcia, 2008; López et al., 2013; Haixiang et al., 2017). The majority and minority classes are typically known as the 'negative' and 'positive' classes, respectively.

In skewed datasets, canonical classification algorithms may be biased towards the majority class, being unable to appropriately predict examples from the minority class (Fernández et al., 2018a; Thabtah et al., 2020). Solutions tackling this difficulty can be grouped into data pre-processing (Chawla et al., 2002; Batista, Prati, and Monard, 2004; Triguero et al., 2019) and algorithmic modification (Zadrozny and Elkan, 2001; Oh, 2011). Those operating at the algorithmic level modify the learning algorithms to make them aware of the imbalanced situation at the learning stage, while those at data-level pre-processing intervene in the cardinalities of positive and negative classes to make them less critically unequal. Cost-sensitive learning (Domingos, 1999; Krawczyk, Woźniak, and Schaefer, 2014) and ensemble-based methods (Galar et al., 2011; Fernandes et al., 2019; Sundar and Punniyamoorthy, 2019) have also become very popular. Cost-sensitive techniques can be considered algorithm level modifications that try to learn more characteristics of minority class examples by incorporating a higher cost to their misclassification. Ensemble-based methods usually combine an ensemble learning algorithm (e.g. Bagging, Boosting) (Galar et al., 2011) with one of the mentioned approaches (e.g. data pre-processing (Chawla et al., 2003), cost-sensitive techniques (Li et al., 2014)) to establish a combination of multiple base classifiers.

Data-level pre-processing solutions consist of undersampling (concerned with eliminating redundant examples in the majority class (Yen and Lee, 2009; García and Herrera, 2009)), oversampling (generates new artificial data for the minority class (Chawla et al., 2002; Batista, Prati, and Monard, 2004; Fernández et al., 2018b)), and hybrid methods (a combination of the previous two) (Ramentol et al., 2012). While all approaches are proved effective in many studies, oversampling and hybrid methods tend to generate more data, which may result in a higher computational cost. Undersampling, on the other hand, aims at reducing the data size, which is more advantageous when employed in large datasets or big data scenarios (Triguero et al., 2015a). Among other strategies for undersampling, Evolutionary Undersampling (EUS) (García and Herrera, 2009), an evolutionary **IS** strategy (Garcia et al., 2012)

for imbalanced classification, has been demonstrated to be very effective in multiple studies, especially in combination with Ensemble-based approaches (Galar et al., 2013; Krawczyk et al., 2016; Sun et al., 2018).

EUS is an example of optimisation techniques to improve ML processes (Song, Triguero, and Ozcan, 2019). This does not only help to balance the distribution of classes but, as an **IS** approach, this also allows us to remove noisy instances in the majority class, which is a common issue in real-world applications (Tao et al., 2020; Stojanovic, He, and Zhang, 2020). In particular, the EUS algorithm performs a binary search guided by an Evolutionary Algorithm (EA) to optimise the selection of (training) examples from the majority class that improves the classification performance. The chromosome quality is measured by classifying the entire training dataset based on the pre-processed set represented by the chromosome. Similarly to most previous works (e.g. in the original EUS (García and Herrera, 2009)), in this chapter we adopt the Nearest Neighbour (NN) (Cover and Hart, 1967) rule as the base classifier. The resulting pre-processed dataset, however, should be ready to be used by any classifier.

Despite its effectiveness, the EUS method is typically very time-consuming, especially in large datasets, due to the cost associated to fitness evaluation. Further advancement of the EUS requires two conditions: (1) reduce the processing time and (2) still guarantee a high classification performance. In the recent literature, the processing time of EUS (and other IS/IG based approaches) is being reduced by using distributed approaches in big data platforms (Triguero et al., 2017; Triguero et al., 2015a), increasing the need for a larger number of computing nodes.

In this work, we are interested in reducing the computational cost of EUS by using fitness approximation approaches (Jin, Olhofer, and Sendhoff, 2000; Jin, 2005), such as surrogate models (Jin, 2011; Rosales-Pérez et al., 2015; Sun et al., 2019), which could accelerate the expensive computation of the classification performance of each chromosome. Surrogate-based methods allow us to reduce the computational cost of search algorithms, as opposed to parallelisation techniques that merely focus on reducing processing time. This kind of approach has been widely investigated in various problems employing evolutionary optimisation techniques (Brownlee and Wright, 2015; Bertini et al., 2011; Salami and Hendtlass, 2003; Chugh et al., 2020), following different approaches, such as fitness inheritance and ML methods (Jin, 2005). Existing methods are usually designed for problems in the continuous search landscape. However, methods for combinatorial domains have been under-explored (Moraglio and Kattan, 2011; Bartz-Beielstein and Zaefferer, 2017) due to the complexity of the field which requires domain knowledge to apply fitness approximation.

In the field of evolutionary **IS** (for imbalanced and standard classification), primitive

approaches for fitness approximation have been employed to reduce the computational cost (windowing (Bacardit et al., 2004)) and processing time (stratification (Cano, Herrera, and Lozano, 2005)). The underlying idea of these methods is to consider subsets of training data for fitness evaluation, reducing the cost on larger datasets. The windowing approach for EUS is dependent on the imbalanced ratio ($ImbR$) (defined as the ratio of the number of instances from the negative class and the positive class). For example, in a two-class dataset with 100k instances and an $ImbR < 9$, each evaluation of any chromosome is processed with more than 20k samples of a stratum (10k samples from each class). Thus, the lower $ImbR$, the more computation is required. Whilst these approaches are important in addressing large datasets with EUS, the use of surrogate models for evolutionary **IS** is an underdeveloped area in the literature that can highly reduce the computational cost of this kind of search technique.

3.1.2 Contributions

In this chapter, we propose a two-stage clustering-based surrogate model for EUS (EUSC) that allows us to compute fitness values faster. As opposed to windowing or stratification approaches, EUSC considers the entire training data when computing fitness values. However, it only performs real evaluations for a limited number of chromosomes. First, a preliminary clustering stage of majority examples allows us to transform binary chromosomes into real coding chromosomes that represent the overall location of the instances selected in a solution. Then, in every generation, the entire population is clustered using the new intermediate chromosome representation to approximate the fitness values based on their similarity and imbalanced ratio. Note that in this designed framework, the intermediate phase that transforms the binary into continuous encoding representation is merely for the sake of fitness approximation and thus it does not impact to the type of the solving problem (i.e. combinatorial as IS). The rest of fitness values are approximated using a clustering-based surrogate model that groups binary chromosomes based on their phenotype similarity (which is the location in the space of the selected instances). To the best of our knowledge, this is the first surrogate-based model for EUS, and one of the very few surrogate models that work on a combinatorial problem (Bartz-Beielstein and Zaefferer, 2017).

EUSC operates in a different way in which two elements are stressed: (1) Considering all the data at each evaluation for a few chromosome representatives, (2) Fitness of the representatives is used to infer the quality of other chromosomes based on a clustering-based model. In addition, we also propose a hybrid surrogate model, called a hybrid surrogate model for EUS (EUSHC), that consists of windowing and

EUSC to significantly reduce the computational cost of the objective function without misleading the search. The difference of EUSHC and EUSC is that the hybrid method uses windowing to estimate the fitness value of a reduced number of chromosomes, and the fitness values of the rest of solutions are approximated based on their similarity and imbalanced ratio. These judgements motivate this study whose contributions can be summarised as follows:

- We investigate the challenge of devising surrogate models for a combinatorial/binary optimisation problem (IS for undersampling). We discuss the weaknesses of using the Hamming distance to compute the similarity between binary chromosomes, which would not reflect well how similar two solutions are. The main novelty of this work lies in developing a means to perform that similarity computation based on the phenotype of the chromosome.
- We propose a clustering-based surrogate model for EUS which highly reduces the computational cost of this method, speeding up the algorithm without reducing significantly its classification performance. Thus, the main contribution is in the acceleration of the fitness evaluation of an **IS** method in the context of imbalanced classification.
- We propose a hybrid surrogate model that integrates windowing with EUSC to highly reduce the computational cost of the fitness function. As EUSHC is hybridisation of EUSC and Windowing, the hybrid scheme contains two approximation phases. The first approximation is from the use of windowing to estimate the fitness value of a reduced number of chromosomes, and the second approximation is the fitness inference for the rest of solutions, similar to EUSC. Thus, the entire search is guided by only approximate fitness values.
- To validate the performance of EUSC, we explore different variants of the proposed method considering multiple factors affecting the model, then analysing empirically their effectiveness. We compare the proposed surrogate model against the original EUS algorithm and EUS with windowing evaluation on 44 standard imbalanced datasets with various imbalance ratios. In comparison with the windowing strategy, the results obtained show that EUSC does not only reduces runtime enormously, but it also provides a high-quality solution which does not significantly decrease classification performance in comparison to the original EUS. In addition, we show how the greedy surrogate model EUSHC allows for a massive reduction of computational costs without considerably reducing accuracy

The rest of this chapter is organised as follows. In Section 3.2, we introduce the background and related works, consisting of EUS, recent work on accelerating fitness evaluation in evolutionary search, and existing approaches to speed up EUS.

Next, Sections 3.3 and 3.4 describe our two proposals in detail. Then Section 3.5 introduces the experimental framework used in this study. Before making several concluding remarks in Section 3.8, we analyse the numerical results of EUSC and EUSHC in Sections 3.6 and 3.7, respectively.

3.2 Background

This section presents background information about EUS for imbalanced classification (Section 3.2.1) and the existing approaches to accelerate evolutionary **IS** techniques (Section 3.2.2).

3.2.1 EUS for Imbalanced Classification

In learning with skewed data, balancing the class distributions can alleviate the bias of standard classification algorithms towards the majority class. Among other approaches, undersampling is an interesting alternative for large datasets as they reduce the number of samples in the majority class (contrary to oversampling which generates artificial minority class samples), consequently, enabling standard algorithms to be capable of identifying examples from both classes more accurately. Undersampling techniques can inherit from **IR** methods which were initially designed for other preprocessing purposes in learning methods (**IS** and **IG** (Garcia et al., 2012; Triguero et al., 2012)).

The simplest way to obtain a balanced subset of the original data is to randomly undersample the majority class (Batista, Prati, and Monard, 2004). However, this non-heuristic approach may discard important data in the negative class due to the randomness in this mechanism. EUS (García and Herrera, 2009) on the other hand, is an evolutionary **IS** algorithm that carries out a heuristic search to optimise the subset of samples that are selected, and thus can increase the accuracy of a classifier on both classes. The search is guided by an EA, namely CHC (Eshelman, 1991), which is efficient at maintaining the balance of exploration and exploitation by applying different mechanisms such as incest prevention, reinitialisation of the population when the search does not progress and the competition among parents and offspring for selecting the elitist. The reduced set is evolved from undersampled instances until the highest performance computed by a fitness function is achieved or stopping conditions are met. EUS can achieve two goals which are the balance of samples between classes and high classification accuracy when the selected negative samples are the most representative.

Assuming binary classification, a formal specification of the problem is the following: A two-class dataset has N^- negative class instances and N^+ positive class

instances. Let x_i be an instance in the dataset where $x_i = (x_{i_1}, x_{i_2}, x_{i_3}, ..., x_{i_m}, x_{i_\omega})$, with x_i belonging to a class given by x_{i_ω} and an m-dimensional feature space in which the feature value at the k^{th} position of the i^{th} sample is denoted as x_{i_k}. In the EUS approach, a candidate solution is a binary chromosome in which each gene takes a value of {1, 0} to represent the presence or absence of an instance x_i, respectively. As only majority class instances are examined for elimination, the size of a chromosome is thus equal to N^-. Positive samples are automatically concatenated with the selected negative examples to form a final reduced set **RS** for classification. The representation of a chromosome in the EUS algorithm is expressed as: $chr_j = (v_{x_1}, v_{x_2}, v_{x_3}, \ldots, v_{x_{N^-}})$ where $v_{x_i} \in \{1, 0\}$ indicates whether sample x_i is included or not.

EUS maintains a population of NP chromosomes that are assessed and ranked based on their quality which considers two main factors: classification performance and class imbalance. The fitness function uses **RS** to classify the entire training dataset. However, in imbalanced classification, traditional accuracy measures are no longer valid as they neglect the fact that there is an imbalanced class distribution. Two commonly used alternatives are geometric mean (GM) (Barandela et al., 2003) and the area under the curve (AUC) of Receiver Operating Characteristic (Bradley, 1997). Both measures have been extensively and interchangeably used in many experimental studies of imbalanced classification. In this chapter, we will focus on the GM, defined in Equation 3.1 to report the classification performance. This is not only because it has been used in many experimental studies on imbalanced classification (López et al., 2013; García and Herrera, 2009; Triguero et al., 2015a), but also because it can reflect the balance between the true positive rate (TP_{rate}) and true negative rate (TN_{rate}) at the same time and therefore the contribution on either class does not have a higher impact than that of the other.

$$GM = \sqrt{\mathrm{TP}_{rate} \times \mathrm{TN}_{rate}} \tag{3.1}$$

The complete fitness function for a chromosome chr is as follows:

$$f_{chr} = \begin{cases} GM_{chr} - \left|1 - \frac{N^+}{s^-}\right| \cdot P & \text{if } s^- > 0 \\ GM_{chr} - P & \text{if } s^- = 0, \end{cases} \tag{3.2}$$

where s^- is the number of selected negative instances and P is a penalisation factor that focuses on the balance between both classes. P is typically set to 0.2 as recommended by the authors, since it provides a good trade-off between both objectives.

The time required to evaluate the quality of each chromosome highly depends on

the size of the training set. In this work, we are interested in developing a fast EUS that can quickly estimate the fitness values of chromosomes without misleading the search.

3.2.2 Reducing Processing Time of Evolutionary IS

Several primitive approaches of fitness approximation have been used for evolutionary **IS** strategies (such as EUS), namely stratification (Cano, Herrera, and Lozano, 2005) and windowing (Bacardit et al., 2004).

The first approach distributes the initial data into several disjoint strata where each stratum still preserves original class distributions. Each stratum is then individually processed by an evolutionary-based strategy to produce different reduced sets (RSs). Finally, all RSs are joined into a final global set. In this approach, the fitness function is not directly applied for the original data, but it is used with each stratified small subset. The ultimate goal of stratification is to deal with the memory consumption limitation rather than speed. The computational cost at evaluation is not reduced in total, but the real fitness function is approximated by the way it is used with a smaller scale of data.

Windowing also splits the training data into several strata with equal class distribution, but it computes the performance on one stratum to represent for the entire initial data. A windowing scheme employs all strata using round-robin policy to estimate chromosomes' fitness during multiple iterations of an evolutionary process. As the data quantity is reduced at each evaluation, the demand for computation is therefore reduced. Despite many positive elements, this approach also has several drawbacks which possibly limit it from extending in applications. Although this method can handle larger datasets, each stratum is empirically limited to no more than tens of thousands of instances (Derrac, García, and Herrera, 2010) in evolutionary-based strategies. Thus, the method can alleviate the burden of fitness computation in relatively large scale datasets (Cano, Herrera, and Lozano, 2005) but not in extreme scenarios like big data. As such, this approach does reduce the computational cost of an evolutionary **IS** and was first used for EUS in (Triguero et al., 2015a).

In the big data context, current parallelisation approaches based on MapReduce aim at reducing the processing time by splitting the datasets into several disjoint blocks (similarly to the stratification approach) that are handled in parallel (Triguero et al., 2015b). In (Triguero et al., 2015a), a two-level parallel scheme combining MapReduce and windowing was proposed for EUS. This approach reduces both computational costs and processing time, but relies on windowing (for imbalanced sets) to

reduce the computational cost. Note that windowing could easily be replaced by the proposed EUSC.

3.3 EUS with a Clustering-based Surrogate Model

In this section, we describe the proposed EUSC framework in detail. Section 3.3.1 motivates the proposed approach, detailing the challenges to perform clustering-based fitness approximation. Finally, Section 3.3.2 provides a detailed description of the proposed model.

3.3.1 Challenges to Perform a Cluster-based Fitness Approximation

As mentioned in the chapter Background, Section 2.2, surrogate models usually rely on distance measures between solutions to perform fitness approximations. As EUS encodes solutions as binary chromosomes, the Hamming distance appears as a natural option to measure the similarity among these binary chromosomes. However, if we do so, we would be expecting this distance metric to reflect the change of chromosomes' representation in the fitness landscape, so that, the fitness of chromosomes varies according to the change of the Hamming distance. In preliminary experiments, we observed that this option was not feasible and performed poorly.

In (imbalanced) classification, some instances may be very important when performing classification, due to their location in the classification space (e.g. those instances in the decision boundaries between classes are typically very important). Using the genotype of the chromosome, the Hamming distance considers the presence of all instances with equal merit, ignoring and neglecting the degree of difference that the actual feature values of a particular example may reflect in the fitness computation, misleading the fitness inference. For example, a chromosome with only one gene swapped would be very similar according to this metric, but it may lead to a great difference in classification performance.

This motivates us to propose the EUSC algorithm which can address the challenge of computing differences among different solutions. The main contribution of this work lies in establishing a bridge connecting the chromosomes' representation and their quality in the fitness landscape. The use of estimated fitness values might seem to be linked to a reduction of the performance. However, currently we need to remember two main considerations for any existing **IS** algorithm (Garcia et al., 2012): (1) the use of training data to compute fitness values is in itself an approximate way to measure the quality of a solution and determine how well the resultant **RS** allows us to learn a concept; (2) the search algorithm may overfit the training data. With

these ideas in mind, we aim to develop a surrogate method capable of reducing the computation cost of EUS without misleading the search.

3.3.2 Two-Stage Clustering-based Surrogate Model for EUS

The main advantage in the proposed EUS with clustering is the ability to quickly obtain the fitness value of a chromosome without always computing its classification performance. To do so, we propose a two-step process based on clustering, which is represented in Figure 3.1. Note that the proposed surrogate model has been integrated with EUS, which was based on the CHC algorithm. However, the ideas proposed here could be implemented in any evolutionary-based algorithm for undersampling (or IS). The added computational cost of the proposed method will be introduced twice as we progress into the two-step process based on clustering.

The complete pseudo-code for EUSC is presented in Algorithm 2 which emphasises (in bold-face) the main modifications to the EUS described in Section 3.2.1. Our method follows the same structure of the CHC algorithm (Eshelman, 1991) that maintains an excellent balance between exploration and exploitation. The CHC algorithm has several important characteristics:

- It uses a heterogeneous uniform cross-over (HUX) for the combination of two chromosomes aiming at maximising the differences of the offspring from their parents (Algorithm 2, line 12).
- It applies an incest prevention mechanism in the combination with the HUX operator to impede the cross of two parents if they are too similar in terms of their Hamming distance.
- It has a restart mechanism to reinitialise the entire population when the evolution does not progress (Algorithm 2, lines 19-21).
- When the search is stagnated, a mutation mechanism is used to populates NP candidates from the current best candidate solution (Algorithm 2, line 21).

We also summarise several functions and importnat parameters that are described in Algorithms 2 and 3 as follows:

- PS, NS = Split(TR): Split the entire **TR** into two parts. PS includes only positive samples while NS merely contains negative samples.
- Infer_At_Init is a user-defined hyper-parameter to indicate whether the fitness inference is employ at the initialisation phase or not.
- Real_Evaluation (NP): Input is NP chromosomes that are needed to be evaluated. This function employs Equation 3.2 to assess the chromosome's quality.

- Imbalance_penalisation ($chrR_j$): Compute the penalty of a solution based on the ratio between the number of positive and negative samples. The penalty value is the subtraction part of GM in Equation 3.2.
- EUSC includes two additional parameters, namely k_1 and k_2. k_1 is used at *Apply k_1-means*, line 2, Algorithm 3. k_2 is employed at Fitness Inference lines 6 and 15, Algorithm 3.

Algorithm 2 Evolutionary undersampling clustering-based algorithm - EUSC

Input: Training Data (TR)
Output: Highest fitness chromosome (Best RS)

1: PS, NS = Split(TR)
2: $\{T_1, T_2, ...T_{k_1}\}$ = Apply k_1-*means*(NS)
3: NP = Randomly initialise the population of chromosomes
4: **if** *Infer_At_Init* == *True* **then**
5: **Intermediate_Pop = Transform_binary_rep (NP)** ▷ Figure 3.1 (top)
6: **Fitness Inference (Intermediate_Pop, NP)** ▷ Computed at Alg. 3
7: **else**
8: Real_Evaluation (NP)
9: **end if**
10: **while** eval < MAX_EVALUATIONS **do**
11: **newPop** = Get M chromosomes from NP based on their structure's diversity
12: newPop = crossover_HUX(newPop) ▷ crossover the selected candidates
13: **if** size(newPop) > k_2 **then** ▷ Can do fitness inference
14: **Intermediate_Pop = Transform_binary_rep (newPop)**
15: **Fitness Inference(Intermediate_Pop, newPop)** ▷ Computed at Alg. 3
16: **else**
17: Real_Evaluation (newPop)
18: **end if**
19: NP = Select best candidates from NP and newPop
20: **if** stagnated == True **then** ▷ Restart mechanism
21: NP = Mutation (Best solution in NP)
22: Real_Evaluation (NP)
23: **end if**
24: **end while**
25: return Best solution in NP

Phase 1: Intermediate Form for Chromosomes, summarised in Figure 3.1 (top) In EUS, the original binary representation for chromosomes is helpful to determine which samples are selected/discarded. However, as stated before, one of its major drawbacks is that it does not represent the real phenotype of a chromosome, which is the actual position of the selected instances that interferes significantly in the classification performance.

Our first step to develop a surrogate model for EUS is related to transforming this binary representation into a real-coding one. Note that it is key that this process is

Algorithm 3 Fitness Inference

Input: Intermediate_Pop and Chromosomes that need fitness inference
Output: Chromosomes associated with their fitness

1: $\{C_1, C_2, ...C_{k_2}\}$ = Apply k_2-*means*(Intermediate_Pop)
2: $\{chrR_1, chrR_2, ..., chrR_{k_2}\}$ = Get k_2 Representatives ▷ Representative: Centroid, Random
3: **for** each cluster C_j in $\{C_1, C_2, ...C_{k_2}\}$ **do**
4: GM_{chrR_j} = Classification performance(RS_{chrR_j}) ▷ Computed by Equation 3.1
5: Imb_{chrR_j} = Imbalance penalisation ($chrR_j$)
6: $Fitness_{Rep_{C_j}} = GM_{chrR_j} - Imb_{chrR_j}$
// Infer the fitness of members in C_j ***using computed*** GM_{chrR_j}
7: **for** each chromosome chr_m in cluster C_j $\{chr1_{C_j}, chr2_{C_j}, ..., chrk_{C_j}\}$ **do**
8: Imb_{chr_m} = Imbalance penalisation (chr_m)
9: $Fitness_{chr_m} = GM_{chrR_j} - Imb_{chr_m}$
10: **end for**
11: **end for**
12: return chromosomes with fitness

very quick to really take advantage of a surrogate model (rather than using a real fitness evaluation). The real-coding intermediate form is created in two steps. Note that Step 1 and Step 2 below correspond to the two steps displayed in Figure 3.1 (top):

- Step 1: Before commencing the EUS algorithm, the training set is first split into positive and negative sample sets (i.e. PS and NS, respectively). Then, we group all majority class samples into different regions based on their feature values. This step will later allow us to reflect in which area the selected instances are predominately based. In our experiments, we focus on the well-known k-means algorithm to group data into k_1 clusters $\{T_1, T_2, ...T_{k_1}\}$. This is the considered computational cost added to the EUSC model compared to the EUS. Note that this clustering task may be considerably slow for big datasets, however, it is only conducted once. The information of these clusters is used during the search whenever a chromosome is needed to be transformed into an intermediate form.

- Step 2: Whenever we need to approximate fitness values (initialisation or during the evolutionary cycle), binary chromosomes will be transformed into intermediate forms with the support of $\{T_1, T_2, ...T_{k_1}\}$. Note that each training instance was allocated to a cluster in the previous step. Each intermediate form is a vector of k_1 features which is the number of k_1 clusters. Firstly, from the binary chromosome, we count the number of selected instances (i.e. genes with a 1) that fall into each one of the clusters. Each value of the intermediate chromosome is a real number, obtained from dividing the number of selected

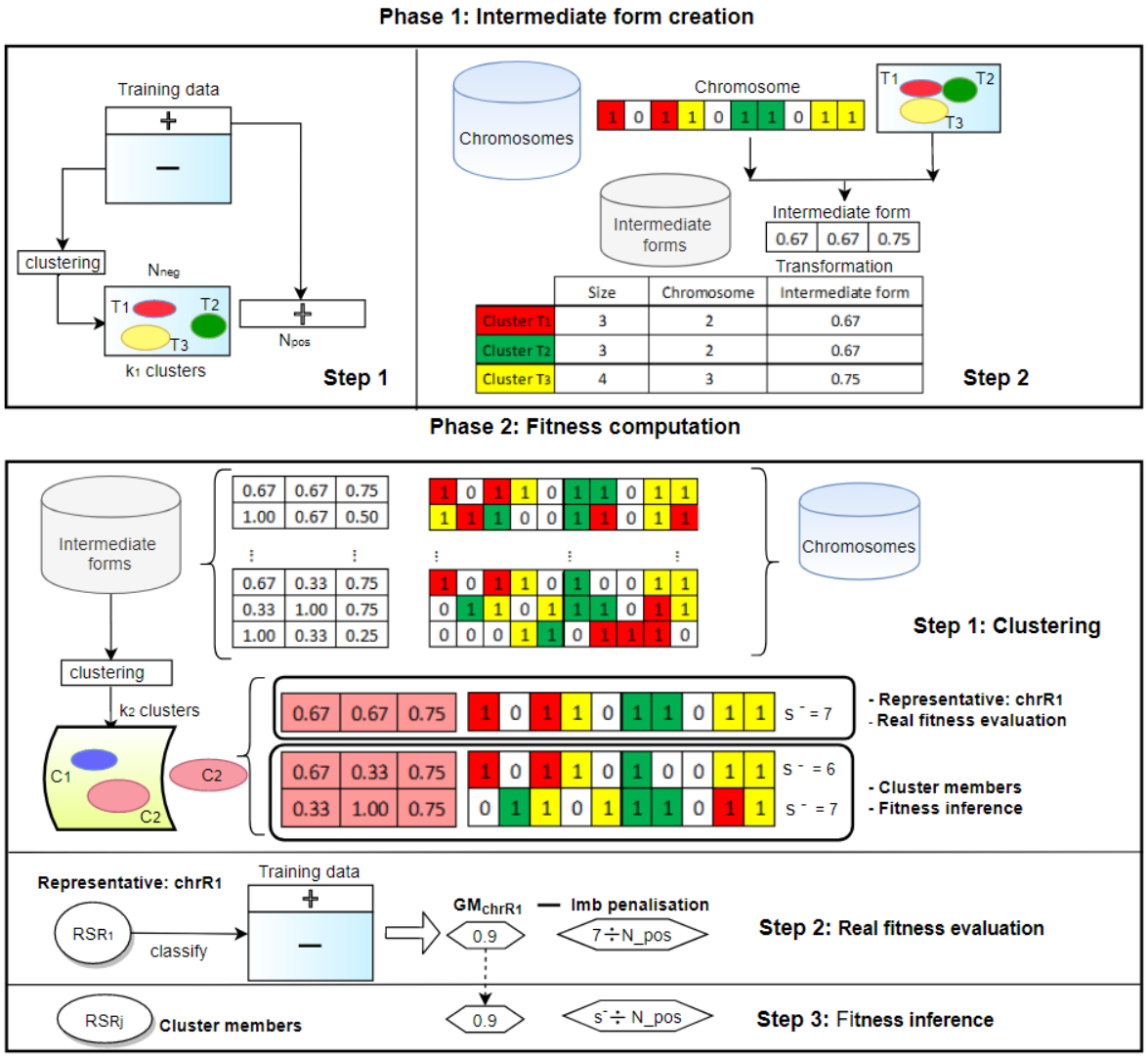


FIGURE 3.1: Two phases of an EUSC process: Chromosome transformation at phase 1 and fitness inference at phase 2.

instances by the cluster's size. These k_1 values tell us the proportion of selected samples at each region, which is approximate information about the location of the selected samples in the instance space. Thus, this is a simple and fast strategy to obtain an approximate phenotype for each binary chromosome.

Phase 2: Fitness Computation, summarised in Figure 3.1 (bottom) After transformation, a chromosome has also a real-coding representation, which allows us to apply fitness inference following similar ideas implemented in the literature for continuous optimisation problems (Kim and Cho, 2001). Algorithm 3 describes in detail the fitness inference mechanism. This stage consists of the following three steps. Note that Step 1, Step 2 and Step 3 below are corresponding to the three steps displayed in Figure 3.1 (bottom)::

- Step 1: To compute the similarity among the current chromosomes considered for evaluation, their intermediate forms are fed into a clustering algorithm

(again we focus on k-means in our experiments) to split the population into k_2 clusters $C_1, C_2, ..., C_{k_2}$. This is an additional computational cost added to EUSC in comparison to EUS. However, the size of the data (i.e. a subset of $NP \times k_1$) we are clustering is so small that the runtime for this process could be considered almost negligible. In this way, the clustering task also indirectly groups the chromosomes into different regions in the binary space. Note that the binary representation is still needed at evaluation time (either real or inferred) to compute the balance between classes (See Equation 3.2).

- Step 2: Compute the fitness value of only k_2 representatives using the real fitness function defined in Equation 3.2. Here we examine different approaches for the selection of representatives. In particular, we analyse the effect of selecting them randomly or using the centroid chromosome from each cluster $C_1, C_2, ..., C_{k_2}$. As a result, we have a set of representative chromosomes $\{chrR_1, chrR_2, ..., chrR_{k_2}\}$ for which we can compute their GM values $\{GM_{chrR_1}, GM_{chrR_2}, ..., GM_{chrR_{k_2}}\}$.
- Step 3: In this final step, we can now infer the fitness values for the remaining chromosomes. As defined in Equation 3.2, the fitness function consists of GM and imbalance penalisation. The GM values of the k_2 representatives will be reused for all the chromosomes belonging to the same cluster. This means that all the members in the same cluster with the representative simply get the representative's GM and thus the cost of classification performance is saved. However, the component of imbalance penalisation can be quickly computed from the binary chromosome for each particular solution. We are aware that transferring the same GM to all members of a cluster may seem to be an oversimplification, and more elaborated solutions will be investigated in the future. However, as we will see in the experimental section, this simple fitness inheritance mechanism allows us to achieve very competitive results.

We have described above the underlying ideas of EUSC but there are a number of factors that should also be taken into consideration. In the evolutionary search, chromosomes are evaluated at initialisation and during the evolutionary loop. When designing EUSC, we realise that the initialisation may be a key step for the entire search, and using approximate values to begin with may not be ideal. Thus, in our experiments we investigate the influence of applying inference at both *Initialisation* and *Evolution* or merely at the *Evolution* phase. Note that whenever the population NP is restarted, we have decided to only perform real fitness evaluations to ensure the search is not misled.

3.4 Hybrid Surrogate Model for EUS

In Chapter 1 and Section 3.1.1, we have introduced Windowing as a fitness approximation method to measure the quality of an **RS** (Bacardit et al., 2004). In this section, we propose a hybrid surrogate model, called a hybrid surrogate model for EUS (EUSHC), that integrates windowing with EUSC presented in Section 3.3 to highly reduce the computational cost of the fitness function without misleading the search for accurate solutions. This section is organised as follows: Initially, Section 3.4.1 discusses the motivation behind this approach, then Section 3.4.2 describes in detail the windowing component and finally Section 3.4.3 details the hybrid approach.

3.4.1 Motivation

Since EUS can determine the best subset of majority class elements to balance the number of samples among classes, it is capable of addressing the imbalanced classification problem (García and Herrera, 2009). The cost associated to its fitness evaluation motivates the use of approximations to assess the quality of a chromosome. The main problem lies in the size of the training data that needs to be classified to measure the classification performance of a given solution.

The use of approximate fitness values might seem linked to a reduction of the performance. However, we postulate that for the problem of **IS** (undersampling) in classification problems, using the training data to compute the fitness value is in itself an approximation of how well the solution (the resulting RS) allows us to learn a concept (which may affect how well we can predict the test set). In addition, the search algorithm may end up overfitting the training data. Hence, these are well-known general weaknesses of any existing **IS** algorithm (Garcia et al., 2012).

The main goal of the proposed EUSHC is to drastically reduce the computational cost of EUS and investigate whether this misleads the search or not. To do so, we integrate two different approaches to approximate fitness values: Windowing and a clustering-based surrogate model. The motivation as to why we add the two different components together is given below.

3.4.2 Windowing for EUS

The idea of windowing was originally proposed in (Bacardit et al., 2004) to accelerate a genetic-based ML algorithm. The key idea is to use partial data instead of the entire dataset for each fitness evaluation. This approach begins with splitting the training set into multiple disjoint strata ($W_1, W_2, ..., W_{nw}$). For each generation of the search, each stratum takes a turn to be used in evaluating candidate solutions.

Due to the reduction of data quantity at each evaluation, the computational cost decreases accordingly.

In the context of EUS for imbalance classification, windowing was first used in (Triguero et al., 2015a). However, dividing the entire training set into several disjoint windows with equal class distribution may produce a significant loss of information of the positive class. Therefore, to apply windowing for EUS, minority class sample will be kept to evaluate a chromosome. However, the set of majority class instances is split into several disjoint strata. The size of each subset of majority examples is set to the number of minority class instances to avoid setting a fixed value for the number of strata. Thus, the number of strata is dependant on the imbalanced ratio.

This simple yet effective approach has been empirically proven to highly reduce the cost of fitness evaluations without significant loss in classification performance. Although the training data is not classified at once to evaluate one chromosome, during the evolutionary process the algorithm utilises all existing training data. The main drawback of this technique lies in the fact that its reduction of computational cost depends on the imbalanced ratio. For this reason, we will use this technique within the proposed hybrid surrogate model to speed up the fitness computation of only some chromosomes of the population.

3.4.3 Construction of the Hybrid Windowing-Clustering Surrogate Model

Fitness approximation based on surrogate models is an under-explored area in binary/combinatorial optimisation. For EUS, the main challenge lies at computing distances between different binary chromosomes, so that, fitness values can be either inherited or approximated based on similarity between chromosomes. EUSC allows us to compute distances between binary chromosomes by transforming them into an real-coding representation. The main advantage of such a model is the ability to very quickly infer the fitness value of a chromosome based on the distance to others without computing any classification. There are two main differences between the windowing and the cluster-based approach:

- Windowing defines a strategy to split **TR** into multiple sets for fitness evaluation, while EUSC approach employs the entire data for fitness evaluation.
- Fitness in Windowing method is approximate in any candidate solutions. For EUSC, fitness is computed by Equation 3.2 for several representative candidates and the approximation is employed for the rest of the solutions.

In this work, we extend that approach by hybridising EUSC with a windowing approach. Figure 3.2 presents the workflow of the entire hybrid model, which consists of the following two phases:

Phase 1: Chromosome Transformation The key point of EUSC is related to transforming the binary representation into a real-coding one. This process should be very quick to really take advantage of a surrogate model (rather than using the real fitness evaluation). The main issue with the binary representation is that does not represent well the real phenotype of the chromosome, which is the actual position of the selected instances of the algorithm.

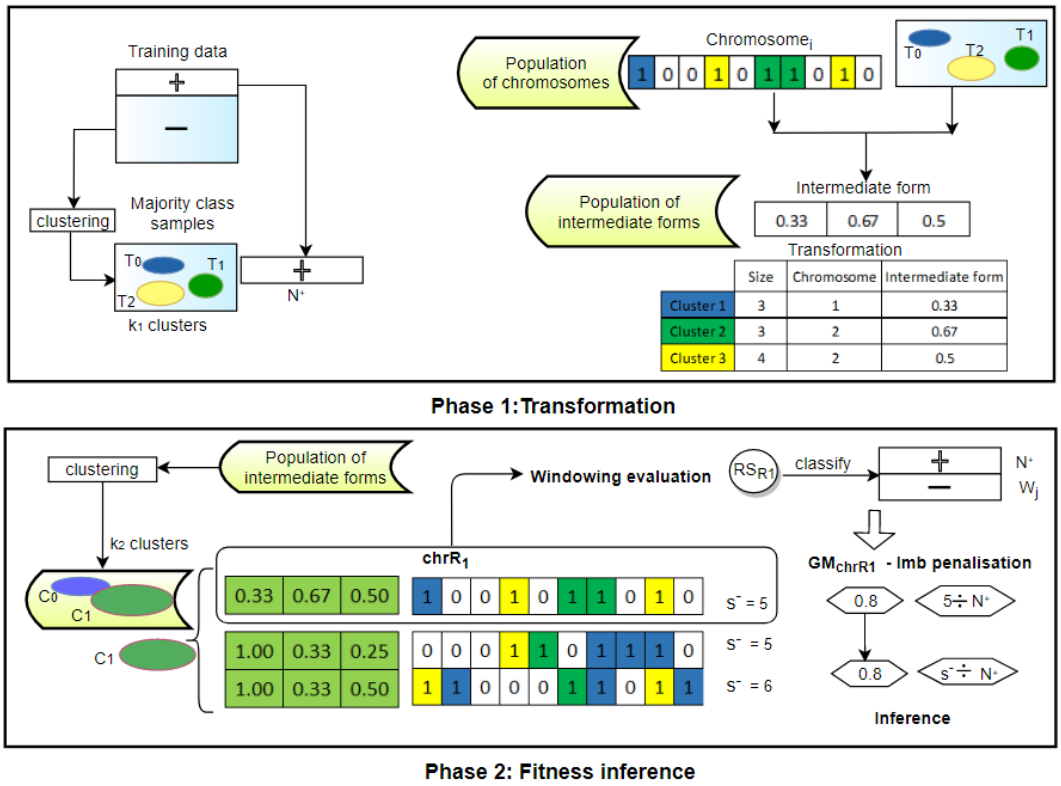


FIGURE 3.2: Workflow of EUSHC: Phase 1 conducts chromosome transformation; in the illustration 1 element out of 3 is selected from T_0 cluster, 2 out of 3 in T_1 cluster, and 2 out of 4 in T_2 cluster. Phase 2 performs fitness inference based on similarities between the transformed chromosomes. Only a representative chromosome from each cluster is evaluated using a windowing approach.

- Step 1: Before starting the evolutionary cycle, all the training samples that belong to the majority class are grouped into k_1 clusters $\{T_0, T_1, ...T_{k_1}\}$. In our experiments, we use the well-known k-means algorithm. The goal of this step is to quickly split the instance space into different regions based on the actual position (i.e. using their feature values) of the majority class examples. Note that this step is the most time-consuming one, but it is only applied once.

- Step 2: During the evolutionary cycle, we will transform binary chromosomes into an intermediate form using the previous clusters. This intermediate forms will have k_1 genes. Firstly, we count the number of selected instances (i.e. genes with a 1) that fall into each of the clusters. Each gene of the intermediate chromosome will be a real value which is computed as the division between the number of selected instances of this cluster and the original number of elements in the cluster. These values produce an intermediate form that tells us approximate information about the location in the instance space of the selected instances.

Phase 2: Fitness Inference When binary chromosomes have been transformed into real-coding ones, we can use similar ideas as implemented in the literature for continuous optimisation problems (Kim and Cho, 2001).

- Step 1: The population of chromosomes in their new intermediate form is fed into a clustering algorithm, which splits the different solutions into $k = k_2$ clusters, $C_0, C1, ..., C_{k_2}$. In this way, the clustering task conducted in the intermediate forms will also indirectly separate the chromosomes in the binary space.
- Step 2: Compute the fitness value of only k_2 chromosomes. To accelerate this step, we incorporate here the windowing approach. This means that for those chromosomes a subset of the training set is classified (as describe in the previous subsection) with the **RS**. We tested different approaches to decide which chromosomes should be evaluated with the fitness function. In this contribution we pick the centroid chromosome $chrR_i$ from each cluster $\{C_0, ..., C_{k_2}\}$ as a set of representative chromosomes $\{chrR_0, ..., chrR_{k_2}\}$.
- Step 3: Infer the fitness value of the remaining chromosomes. The GM values of the $\{chrR_0, ..., chrR_{k_2}\}$ (Algorithm 3.1) have been calculated in the previous step. To compute the fitness of the rest of the chromosomes, Equation 3.2 uses the GM value GM_{chr_j} of the centroid of the cluster. This means that all the members of a cluster simply inherit the same GM value. However, the component of the balance between classes of the fitness function is calculated based on the number of elements selected by the particular solution. We acknowledge that transferring the same GM to all members of a cluster may be an oversimplification, and more elaborated solutions could be adopted; however, our experiments show that this simple approach achieves good results.

In the experiments presented in the next section, the above fitness approximation is applied to all fitness evaluations, including the evaluation of the initial population.

3.5 Experimental Framework

In this section, we present the experimental framework in which our proposal, the original EUS and the EUS with Windowing evaluation will be compared. This section begins with the description of the used imbalanced datasets (Section 3.5.1) and is followed by the parameter configuration (Section 3.5.2). Finally, we briefly introduce the non-parametric statistical tests (Section 3.5.3) that will be used to analyse the results.

3.5.1 Datasets

In our experiments we consider numerous two-class imbalanced datasets with different imbalanced ratios, from low to high. The datasets are obtained from the KEEL dataset repository (Triguero et al., 2017) which has been used as a resource for many experiments in previous studies (Galar et al., 2013; García et al., 2018). Table 3.1 summarises the properties of the datasets and dataset entries are sorted by the ascending order of the imbalanced ratio. The range of imbalanced ratio goes from 1.5 to 9 for low imbalanced datasets and over 9 for highly imbalanced datasets. Each row represents a dataset showing its name (**Dataset**), the number of attributes (**Att**), the number of samples (**Samp**), the percentage of examples for each class (**%Class(min,maj)**) and the imbalanced ratio (**ImbR**). In total, there are 44 datasets in the experimental setup; each one is partitioned using 5-fold stratified cross-validation which delivers an adequate number of positive class samples in the test partitions. In a 5-fold cross-validation scheme, the original input data is randomly partitioned into 5 subsets. To compute the performance of a method, a single subset is retained to test the model, while the remaining 4 subsets are used as training data. This process is repeated until each one of the 5 subsets serves once as the test data. Thus, the overall performance of each dataset is given by an average of the 5 results from the 5 folds.

3.5.2 Parameter Configuration

EUS has been widely used in the literature and we employ the parameter values used in (García and Herrera, 2009), including the number of evaluations, population size, penalisation factor and others, as it has been done in most studies for a fair comparison. EUSC includes two additional parameters, namely k_1 and k_2. k_1 is the number of clusters when the majority class examples are divided in stage 1 of the algorithm, while k_2 is the number of groups into which intermediate chromosomes are categorised in stage 2. We explored a great number of combination pairs $\{k_1, k_2\}$ where each k_1 in {2, 4, 6, 8, 10, 12, 14, 20} is combined with each k_2 in {2, 4, 6, 8, 10, 12}, resulting in a total of 48 pairs of $\{k_1, k_2\}$. We analysed the performance of EUC on all datasets with these pairs and only found slight differences among them. Thus,

TABLE 3.1: Summary of datasets from low to highly imbalanced ratios.

Dataset	Att	Samp	%Class (min,maj)	ImbR	Dataset	Att	Samp	%Class (min,maj)	IR
glass1	9	214	(0.36, 0.64)	1.82	yeast-2_vs_4	8	514	(0.10, 0.90)	9.08
ecoli-0_vs_1	7	220	(0.35, 0.65)	1.86	yeast-0-5-6-7-9_vs_4	8	528	(0.10, 0.90)	9.35
wisconsinImb	9	683	(0.35, 0.65)	1.86	vowel0	13	988	(0.09, 0.91)	9.98
pimaImb	8	768	(0.35, 0.65)	1.87	glass-0-1-6_vs_2	9	192	(0.09, 0.91)	10.29
iris0	4	150	(0.33, 0.67)	2.00	glass2	9	214	(0.08, 0.92)	11.59
glass0	9	214	(0.33, 0.67)	2.06	shuttle-c0-vs-c4	9	1829	(0.07, 0.93)	13.87
yeast1	8	1484	(0.29, 0.71)	2.46	yeast-1_vs_7	7	459	(0.07, 0.93)	14.30
habermanImb	3	306	(0.26, 0.74)	2.78	glass4	9	214	(0.06, 0.94)	15.46
vehicle2	18	846	(0.26, 0.74)	2.88	ecoli4	7	336	(0.06, 0.94)	15.80
vehicle1	18	846	(0.26, 0.74)	2.90	page-blocks-1-3_vs_4	10	472	(0.06, 0.94)	15.86
vehicle3	18	846	(0.25, 0.75)	2.99	abalone9-18	8	731	(0.06, 0.94)	16.40
glass-0-1-2-3_vs_4-5-6	9	214	(0.24, 0.76)	3.20	glass-0-1-6_vs_5	9	184	(0.05, 0.95)	19.44
vehicle0	18	846	(0.24, 0.76)	3.25	shuttle-c2-vs-c4	9	129	(0.05, 0.95)	20.50
ecoli1	7	336	(0.23, 0.77)	3.36	yeast-1-4-5-8_vs_7	8	693	(0.04, 0.96)	22.10
new-thyroid1	5	215	(0.16, 0.84)	5.14	glass5	9	214	(0.04, 0.96)	22.78
new-thyroid2	5	215	(0.16, 0.84)	5.14	yeast-2_vs_8	8	482	(0.04, 0.96)	23.10
ecoli2	7	336	(0.15, 0.85)	5.46	yeast4	8	1484	(0.03, 0.97)	28.10
segment0	19	2308	(0.14, 0.86)	6.02	yeast-1-2-8-9_vs_7	8	947	(0.03, 0.97)	30.57
glass6	9	214	(0.14, 0.86)	6.38	yeast5	8	1484	(0.03, 0.97)	32.73
yeast3	8	1484	(0.11, 0.89)	8.10	ecoli-0-1-3-7_vs_2-6	7	281	(0.02, 0.98)	39.14
ecoli3	7	336	(0.10, 0.90)	8.60	yeast6	8	1484	(0.02, 0.98)	41.40
page-blocks0	10	5472	(0.10, 0.90)	8.79	abalone19	8	4174	(0.01, 0.99)	129.44

we decided not to report all results in this study for the sake of simplicity as they do not affect the conclusion of comparing the EUSC schemes to the EUS algorithm. In particular, we have chosen the pair $\{6, 6\}$ to present the outputs. Furthermore, as discussed in Section 3.2.1, EUS uses a heuristic search to find the subset of examples which can be different depending on the starting point of search. Hence, to make a fair comparison, we set up 10 fixed seeds to let each algorithm begin from the same point in 10 different executions. The results presented are the average of these ten executions.

To analyse the effectiveness of our EUSC proposal, we will use four different strategies (presented in Table 3.2), in which each is varied by two factors Representative (*Rep*), and Inference (*Infer_At_Init*). As discussed in Section 3.3.2, these two elements contribute their significant influence on the behaviour of the search. *Rep* is a boolean value to determine whether the representative is a random or the centroid chromosome. *Infer_At_Init* is a boolean value to determine whether fitness inference is used at both *Initialisation* and *Evolution* or merely at the *Evolution* phase. The results obtained from these four schemes will be contrasted against two benchmarks: (1) the original EUS, which is assumed to achieve the highest GM, but the most time-consuming approach; (2) EUS using windowing to evaluate its fitness function. Table 3.2 summarises all the parameters used in the experiments.

TABLE 3.2: Parameters used for different configurations of EUSC.

Rep	Infer_At_Init	Scheme	Shared parameters
Random	Initialisation & Evolution	R-IE	Population Size (NP) = 50, Max Evaluations = 10000
	Evolution	R-E	Probability of inclusion HUX = 0.25
Centroid	Initialisation & Evolution	C-IE	Measure = GM, $k_1 = 6$
	Evolution	C-E	$k_2 = 6$, Number of Runs = 10

3.5.3 Non-parametric Tests for Statistical Analysis

As there is not an established procedure to assert whether a classifier is better than another, statistical approaches have been adopted to determine whether the difference in performance obtained from the experiments is real or random. While a parametric test requires several assumptions to be satisfied, such as a normal distribution of inputs and homogeneity of variance, a non-parametric test is free from those requirements (Demšar, 2006; Derrac et al., 2011). In this chapter, we will employ non-parametric tests, specifically the Friedman Aligned-Ranks test (Hodges, Lehmann, et al., 1962) plus a Holm post-hoc test (Holm, 1979) to perform statistical analysis on the performance of the algorithms. Initially, the Friedman Aligned-Ranks test conducts multiple comparisons to detect statistical differences in the performance of multiple algorithms. This test will establish a ranking order of all compared algorithms. Then, the Holm post-hoc test is used to discover whether the highest ranking algorithm (the control algorithm) presents statistical differences with respect to the remaining methods. In our experiment, a level of significance of $\alpha = 0.05$ is adopted. Further information discussing these tests can be referred at http://sci2s.ugr.es/sicidm/.

3.6 Analysis of Results of EUSC

In this section, we present an examination of the results of EUSC. It begins with a detailed analysis of the behaviour of the EUSC scheme through an evolutionary search cycle (Section 3.6.1), and is followed by the report of the runtime as well as the reduction in the number of fitness function calls (Section 3.6.2). Finally, we statistically compare the performance of the different algorithms (Section 3.6.3).

3.6.1 Detailed Analysis of the Behaviour of EUSC

EUSC approximates the fitness values for some of the chromosomes during the evolutionary, which may change the behaviour of the original EUS. This section aims to investigate whether our fitness approximation approach produces values close to the real fitness ones and determine which of the four EUSC configurations perform better on the 44 used datasets. To do so, we carry out the following two experiments.

Experiment 1 The first experiment involves an analysis of the fitness difference between approximated and true fitness values through an evolutionary search cycle. Note that the fitness difference is reported as an average figure in relation to the number of chromosomes whose fitness is approximated in each generation (which may differ in every generation). This is due to the nature of CHC which does not

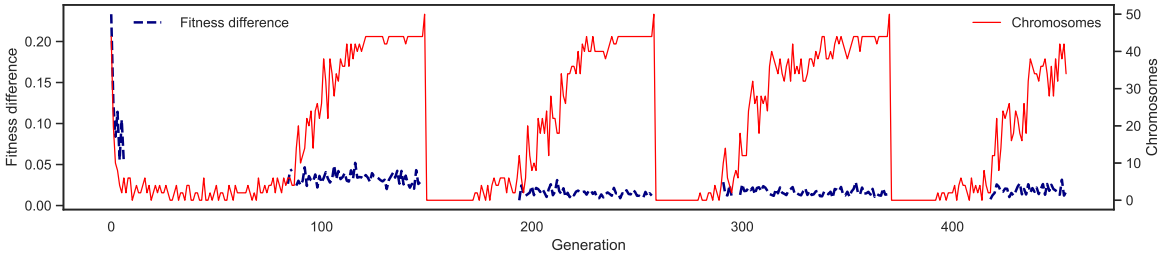


FIGURE 3.3: Behaviour of the EUSC through generations run by method R-IE on fold 1 of dataset ecoli2.

always produce NP offspring in every generation but the size of *newPop* depends on how many chromosome candidates in the current population are selected for conducting crossover. The selection mechanism is determined by the gene difference between two chromosome candidates, measured by the Hamming distance. The two candidates will be selected if their gene difference surpasses a user-defined threshold value. Hence, the surrogate model for fitness approximation is only employed when the quantity of chromosomes is greater than k_2. An example of an evolutionary search process run by scheme R-IE on fold 1 of dataset ecoli2 is plotted in Figure 3.3. Two y-axes are sharing the same x-axis. The left y-axis presents the fitness difference, while the right one expresses the number of chromosomes that are evaluated. Note that Figure 3.3 is only an example to characterise the behaviour of a specific EUSC on a dataset fold. However, a similar behaviour has been observed in other datasets and the different variants.

Analysing this figure in detail, we can see how the evolutionary search cycle begins with random initialisation of NP chromosomes. Due to random allocation, these chromosomes have diverse fitness values, resulting in a large value of fitness difference. Through generations, the value progressively decreases and remains low at less than 0.05 until the end of the search. Particularly, in generations 90-150, 190-260, 290-370, and 420-440 where the fitness inference is used, while the chromosome quantity follows an upward trend, the variation of fitness difference value remains mostly unchanged, fluctuates at somewhere between 0 and 0.05. This means that while the CHC algorithm introduces more diversity to the population, the EUSC approach is yet effective to divide the chromosomes into groups at which members of each group are approximate in fitness. There appear discontinuities in the fitness difference curve, indicating that fitness approximation is not applied in these generations as the number of generated offspring is lower than k_2. Furthermore, when the upward trend of the chromosome quantity reaches the top value (NP), it suddenly drops to the bottom due to the restart mechanism of the CHC algorithm. The algorithm uses a mutation process to generate a new generation from the current best chromosome. Taking the highest fitness chromosome in the current population

as the input, CHC mutates one or several genes to generate different chromosomes for a new population. A new similar pattern of the search restarts from this point as the population has been refreshed.

Experiment 2 With the previous experiment, we can say that at each generation of a search, the EUSC does not produce a large difference between approximated and true fitness values. If each EUSC scheme does not behave remarkably differently, it may end up with the same number of generations per search, resulting in a close fitness difference in total (note that the stopping criteria is based on number of evaluations, and not number of generations). In this experiment, we aggregate all the generated fitness differences throughout the entire search and from the 5 folds of each dataset, displayed in Figure 3.4. The aim of this experiment is to understand how different the search of four schemes performs regarding the total fitness difference. Higher fitness difference in an EUSC configuration infers that either more generations have been conducted in a search or fitness difference is large in some particular EUSC schemes and some datasets. The results have been sorted by the number of samples of datasets for the sake of studying the influence of their size.

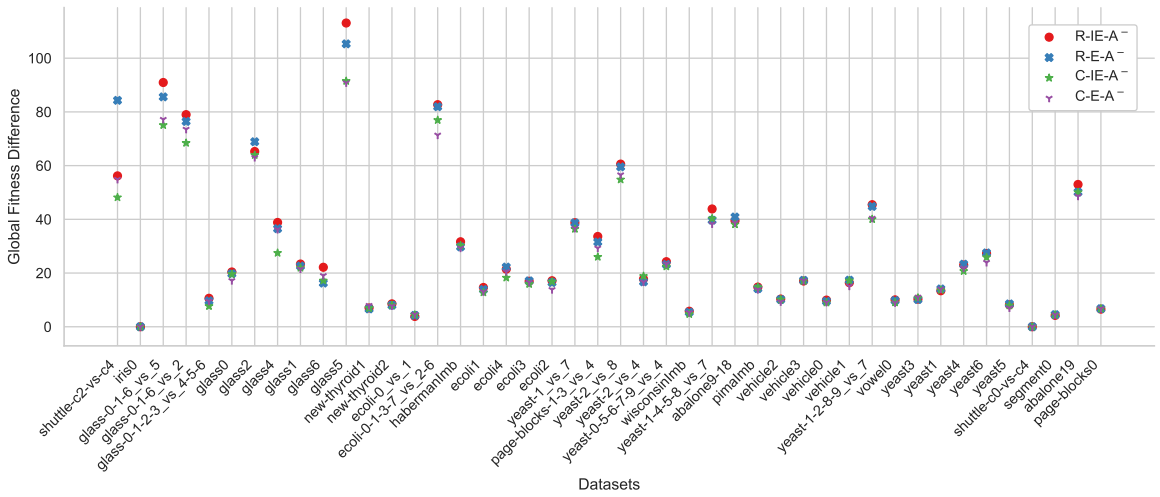


FIGURE 3.4: Total fitness difference aggregated from evolutionary search of each EUSC scheme over 5 folds of each dataset.

Centroid-based schemes (representative is the centroid chromosome) are likely to produce lower global fitness difference with respect to the random-based counterparts (representative is a random chromosome), denoted by a lower/nearly equal value of fitness difference in all datasets. The higher aggregated value in these random-based approaches can be attributed to the representative selection. The random-based schemes might obtain a very good solution and assign its fitness to other chromosomes, at which the least difference is generated. However, there is a probability that the selection of a representative falls into the worst situation or

nearby at which the inference task produces substantial fitness difference and thus adds a heavy burden to the global value. Note that if chromosomes in a cluster share approximate values, the difference may not be strongly affected by the selection mechanism.

In Figure 3.4 the four schemes tend to have a very similar behaviour in most of the datasets, producing a similar accumulated fitness difference value. On datasets that the range of total accumulated fitness difference is in between 40 and 50, the four EUSC schemes tend to separate each other, meaning the search tends to be slightly different. On datasets with the fitness difference values above 50, the four EUSC schemes show clear differences. These datasets are usually the ones having very high imbalance ratio. It is much more diffused on datasets with both high imbalance ratio and low samples, such as glass5 and glass-0-1-6_vs_5. As a result, with datasets having a low number of samples, it is reasonable not to use EUSC as the EUS does not suffer from high computational expense.

3.6.2 Runtime and Real Evaluations Reduction

In this section, we will compare the runtime needed by all the compared algorithms in every dataset. Figure 3.5 plots the comparison. For the sake of clarity and observing the influence of data size on the runtime, we group the first 22 datasets in the top sub-plot (smaller datasets), 15 next datasets in the middle sub-plot (medium-size datasets) and remaining 7 datasets in the bottom one (the larger datasets used in the experiments). Looking at that figure, we can observe that:

- Overall, four EUSC schemes demanded an insignificant amount of time to perform undersampling in comparison to the time required by the original EUS. This is also the case for EUS with windowing in about half of the datasets. However, as mentioned before, the EUS with windowing is strongly influenced by the IR. Thus, its runtime on datasets with high imbalance ratio was low and much lower when the size of those datasets is small. However, in other small datasets (low IR) this approach consumed a mostly comparable amount of time to EUS, and the runtime dramatically reduced in larger ones.
- Although the four EUSC schemes have a difference in the amount of time consumed, the variation is not significant in comparison with EUS. Over 44 datasets, the minimum and maximum percentage of the runtime saved are 16.76% and 83.24%, respectively. Additionally, on average in all datasets, EUS takes 30.63s, EUS_windowing takes 9.03s, and the four EUSC schemes consume from 6.66s to 7.25s.

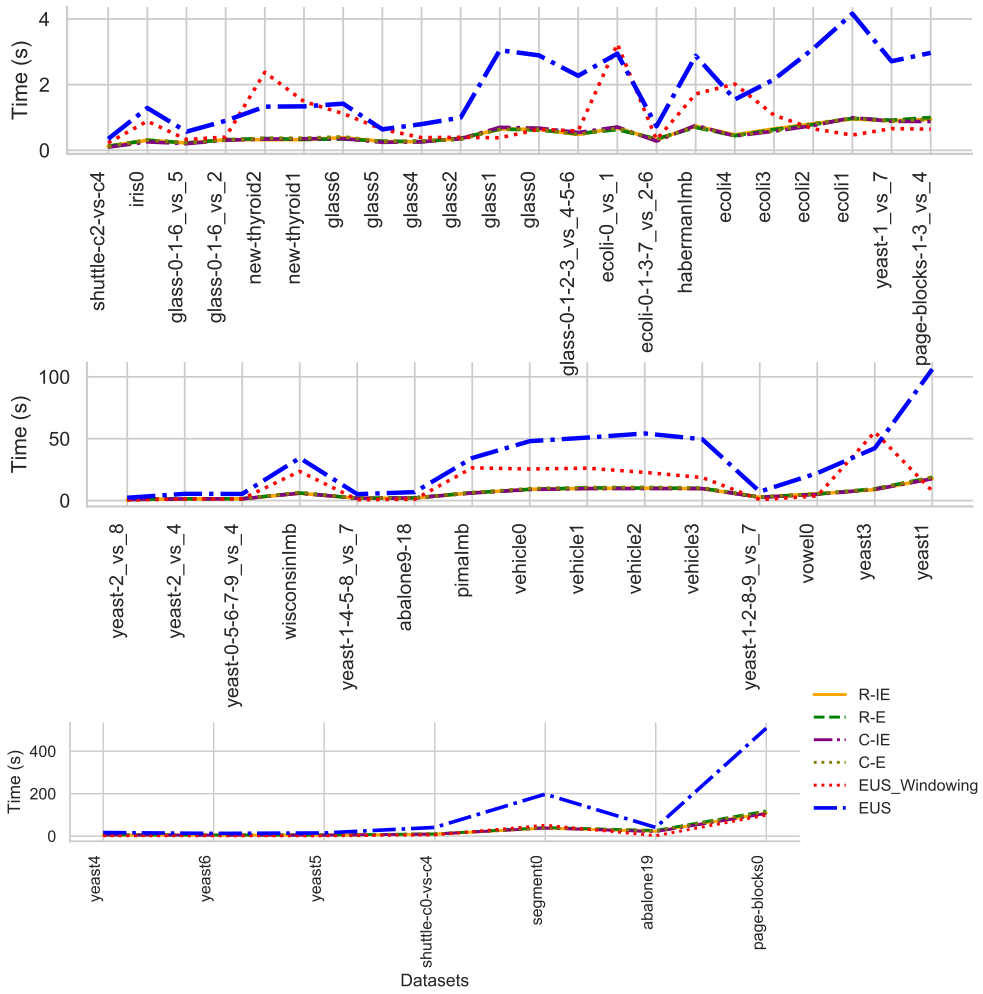


FIGURE 3.5: Comparison of the runtime (in seconds) of the involved algorithms in every single dataset. Runtime of the first 22 datasets (Top), next 15 datasets (Middle), last 7 datasets (Bottom).

In addition to runtime, we also report the number of evaluations that our method saved. As there is not a significant difference observed in the number of real fitness function calls among four EUSC schemes, we plot the histograms with the total number of evaluations of two representative schemes (C-E and R-IE) compared to the 10000 evaluations performed by the EUS for each dataset (Figure 3.6). As seen in the graph, the two schemes save a significant number of evaluations, up to 80% the evaluations demanded by the original EUS in most datasets. This infers a huge reduction of extensive computation in calculating the fitness for each chromosome.

3.6.3 Classification Performance Comparison

In the previous section, we have shown that our proposal is more advantageous than the original EUS or EUS_Windowing in terms of runtime and the number of real evaluations used. However, these gains would not be of any value if the final

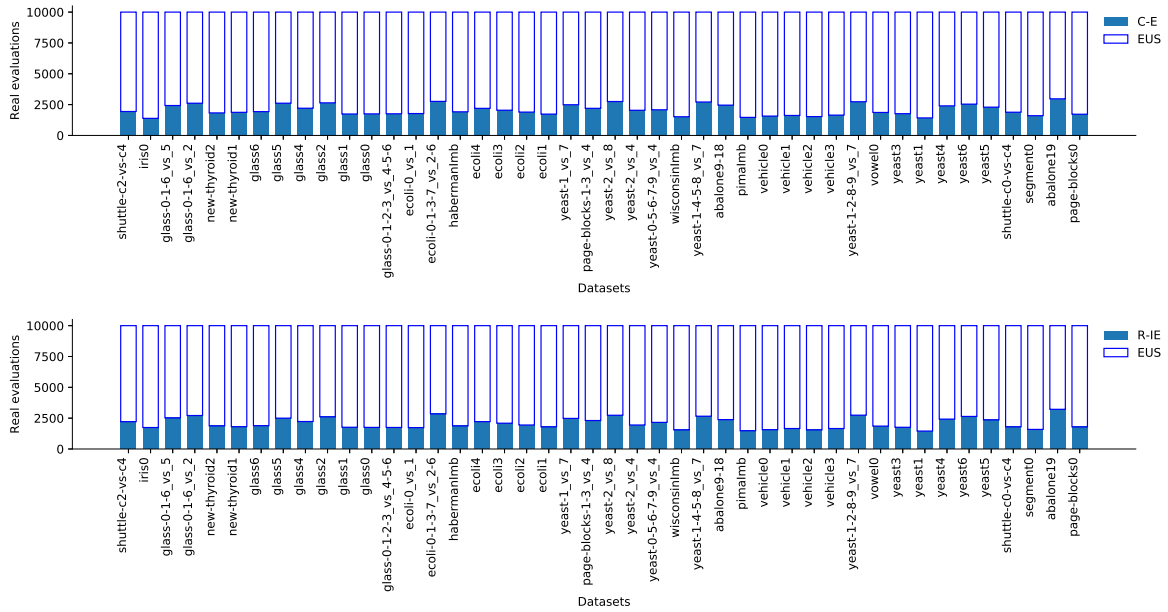


FIGURE 3.6: The number of real evaluations from the two EUSC schemes contrasted to the EUS algorithm over 44 imbalanced datasets.

goal of achieving high classification performance considerably decreased. This section thus aims to report the average GM of all the algorithms in test data. Table 3.3 shows the results in detail. The best result for each dataset is highlighted in boldface. The last row shows the number of times in which an algorithm has obtained the highest performance.

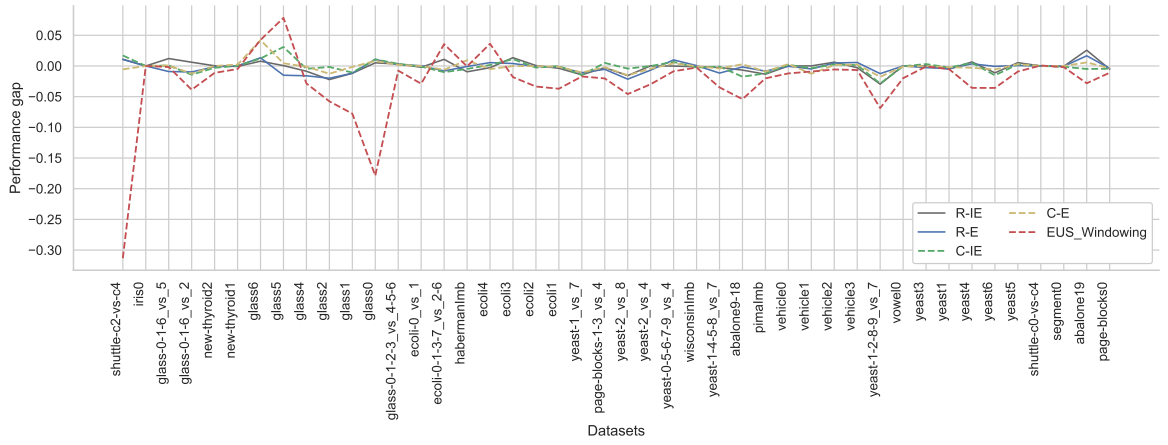


FIGURE 3.7: The difference in terms of GM of the EUSC schemes and EUS_Windowing constrated against the EUS.

Additionally, we also highlight how much difference in terms of GM of the four EUSC configurations and the EUS_Windowing against the original EUS. Every GM value of each EUSC scheme and EUS_Windowing subtracts the GM of the EUS to produce the curves, displayed in Figure 3.7. The plots lying above 0 signify better performance, while those under 0 mean worse. The distance measured from the

TABLE 3.3: Average GM of all compared algorithms over 44 datasets.

Dataset	R-IE	R-E	C-IE	C-E	EUS_Windowing	EUS
shuttle-c2-vs-c4	0.9690	0.9690	**0.9753**	0.9527	0.6449	0.9582
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
glass-0-1-6_vs_5	**0.9289**	0.9077	0.9175	0.9176	0.9151	0.9168
glass-0-1-6_vs_2	**0.6612**	0.6454	0.6409	0.6426	0.6164	0.6551
new-thyroid2	**0.9888**	0.9871	0.9856	0.9885	0.9773	0.9885
new-thyroid1	0.9851	0.9862	0.9862	**0.9882**	0.9809	0.9859
glass6	0.8724	0.8779	0.8768	0.9068	**0.9071**	0.8646
glass5	0.8298	0.8141	0.8602	0.8336	**0.9076**	0.8292
glass4	0.8712	0.8638	0.8752	0.8785	0.8513	**0.8798**
glass2	0.6879	0.6901	0.7085	0.6975	0.6525	**0.7101**
glass1	0.7668	0.7669	0.7680	0.7772	0.7010	**0.7787**
glass0	0.8015	0.8070	**0.8072**	0.8046	0.6176	0.7964
glass-0-1-2-3_vs_4-5-6	0.9493	0.9483	**0.9496**	0.9478	0.9385	0.9461
ecoli-0_vs_1	0.9580	**0.9605**	0.9591	0.9601	0.9312	0.9601
ecoli-0-1-3-7_vs_2-6	0.6800	0.6611	0.6591	0.6631	**0.7048**	0.6692
habermanImb	0.5547	0.5634	0.5594	**0.5736**	0.5635	0.5642
ecoli4	0.8972	0.9055	0.9016	0.8949	**0.9362**	0.9000
ecoli3	**0.8470**	0.8375	0.8447	0.8333	0.8153	0.8335
ecoli2	**0.9005**	0.8988	0.8971	0.8996	0.8663	0.8998
ecoli1	0.8639	0.8660	0.8676	0.8662	0.8306	**0.8677**
yeast-1_vs_7	0.7106	0.7140	0.7102	0.7144	0.7079	**0.7250**
page-blocks-1-3_vs_4	0.9575	0.9547	**0.9652**	0.9581	0.9399	0.9602
yeast-2_vs_8	0.7797	0.7739	0.7911	0.7802	0.7496	**0.7954**
yeast-2_vs_4	**0.9074**	0.9003	0.9070	0.9027	0.8774	0.9071
yeast-0-5-6-7-9_vs_4	0.7747	**0.7848**	0.7815	0.7792	0.7663	0.7749
wisconsinImb	0.9666	**0.9684**	0.9652	0.9674	0.9652	0.9678
yeast-1-4-5-8_vs_7	0.6412	0.6321	0.6427	0.6396	0.6088	**0.6437**
abalone9-18	0.7246	0.7297	0.7138	**0.7341**	0.6772	0.7313
pimaImb	0.6817	0.6867	0.6831	0.6859	0.6749	**0.6951**
vehicle0	0.9148	0.9143	0.9166	**0.9179**	0.9027	0.9148
vehicle1	**0.6716**	0.6667	0.6678	0.6588	0.6624	0.6715
vehicle2	**0.9294**	0.9280	0.9257	0.9247	0.9175	0.9232
vehicle3	0.7180	**0.7265**	0.7222	0.7232	0.7142	0.7208
yeast-1-2-8-9_vs_7	0.6469	0.6640	0.6479	0.6588	0.6078	**0.6765**
vowel0	**0.9921**	0.9918	0.9914	0.9910	0.9719	0.9918
yeast3	0.8751	0.8722	**0.8780**	0.8747	0.8740	0.8749
yeast1	0.6501	0.6509	0.6527	0.6532	0.6501	**0.6552**
yeast4	**0.8222**	0.8191	0.8206	0.8126	0.7799	0.8156
yeast6	0.8326	0.8432	0.8287	0.8381	0.8080	**0.8438**
yeast5	**0.9639**	0.9593	0.9611	0.9614	0.9494	0.9585
shuttle-c0-vs-c4	0.9960	0.9960	0.9960	0.9960	**0.9968**	0.9960
segment0	0.9884	0.9883	0.9876	**0.9890**	0.9870	0.9889
abalone19	**0.6600**	0.6509	0.6294	0.6401	0.6061	0.6343
page-blocks0	0.9096	0.9108	0.9103	0.9111	0.9038	**0.9148**
Wins	**13**	**5**	**6**	**6**	**6**	**13**

baseline $(y = 0)$ to a plot indicates how much an algorithm performs better/worse than the EUS. Looking at this table and figure, we can make the following observations:

- Despite using up to 80% more evaluations, EUS does not always achieve the highest GM. In contrast, using fitness approximation approaches does not only save a lot of computation but also might sometimes get even better performance. From Table 3.3, we can highlight the configuration R-IE, which appears very competitive in comparison to the EUS algorithm, obtaining the same number of wins out of the 44 datasets. Other algorithms find the best solution in 5 or 6 out of 44 datasets.
- In Figure 3.7, four EUSC schemes oscillate around the baseline in a confined

range of [-0.03, +0.04], while EUS_Windowing fluctuates with a larger distance. This infers stable performance of all configurations which is more consistent with the performance attained by the EUS technique.

Although the differences of GM among all involved algorithms are likely inconsiderable from what has been analysed so far, there is no evidence to confirm whether these differences are significant. Hence, we will employ the Friedman Aligned-Ranks test to discover if there exist statistical differences in the involved algorithms. After having the ranking table provided by the Friedman test, we then use Holm post-hoc test to find out whether the highest-ranking algorithm statistically outperforms the rest. Table 3.4 presents the results of this test. In this table, algorithms are sorted by their ranks, from the best to the worst and each algorithm is also associated with a p_{Holm} value at the same row.

TABLE 3.4: Average rankings of the algorithms over 44 datasets (Friedman Aligned-Ranks test and Holm post-hoc test).

Algorithm	Ranking	p_{Holm}
EUS	2.7386	-
C-E	3.0568	0.4881
C-IE	3.2841	0.4881
R-IE	3.2955	0.4881
R-E	3.4091	0.3711
EUS_Windowing	5.2159	0

- As expected, the original EUS gets the lowest ranking value and is established as the control algorithm. Our EUSC schemes are placed in the middle of the table, while EUS_Windowing with the largest ranking value lies in the last position.
- With the level of significance $\alpha = 0.05$, Holm's test does not report any significant difference between EUS and four EUSC schemes. However, the test reveals the EUS algorithm statistically outperforms the EUS_Windowing as $p\text{-}value = 0$.
- Although R-IE has the same number of wins with the EUS over 44 datasets, it does not show this advantage in the Ranking and p_{Holm} columns. This is because the benefits gained from winning in a number of datasets are deducted from the significant loss in the other datasets. At datasets where R-IE does not perform the best, its performance is usually ranked below C-E and C-IE.

Finally, to make sure that our proposed approach has not benefited from having the EUS_Windowing algorithm in the $1 * N$ comparison, we apply the Wilcoxon test between the original EUS and the different EUSC schemes. Table 3.5 shows rankings R^+ (sum of the positive ranks) and R^- (sum of the negative ranks) together with the

p-values obtained from the Wilcoxon test. As we can see, R^+ is slightly greater than R^- in the first two rows and moderately larger in the last two rows (meaning that EUS obtains slightly better results). However, the Wilcoxon test does not report significant differences between the EUS with a level of significance $\alpha = 0.05$.

TABLE 3.5: Results of the Wilcoxon test when the EUS algorithm is contrasted against the seven most effective EUSC schemes.

EUS vs	R^+	R^-	*p-value*
C-IE	582.5	407.5	≥ 0.2
R-IE	569.5	420.5	≥ 0.2
C-E	629.5	360.5	0.1185
R-E	656.5	333.5	0.0598

3.7 Analysis of Results of EUSHC

In this section, we analyse the performance of EUS related methods including EUS, EUS_windowing, EUSC (i.e. variant C-E is selected as it ranked the highest performance amongst others) and EUSHC (i.e. variant C-E with Windowing) with respect to several aspects, including runtime, number of evaluations, and classification performance.

3.7.1 Runtime

In this section, we compare the runtime required by each one of the methods in every single dataset. Figure 3.8 plots this comparison. For the sake of clarity, we sort the datasets by the runtime of EUS, and also apply logarithmic scale (base 10) on the vertical axis. We present two subplots, grouping 44 datasets into two halves. Looking at that figure, we can observe that:

- Overall, all the approximation methods consumed an insignificant amount of time to perform undersampling compared to the time demanded by the original EUS.
- Windowing spent a mostly equivalent amount of time to EUS in small datasets, but the time is dramatically reduced in larger ones. As stated above, the windowing approach is also affected by the imbalance ratio.
- Both EUSC and EUSHC show a very low runtime across the 44 examined datasets. As expected the hybrid approach is always faster than EUSC, and it is clearer on larger datasets. On average, EUSHC is roughly 52.84% faster than EUSC (3.15s vs. 6.68s, respectively).

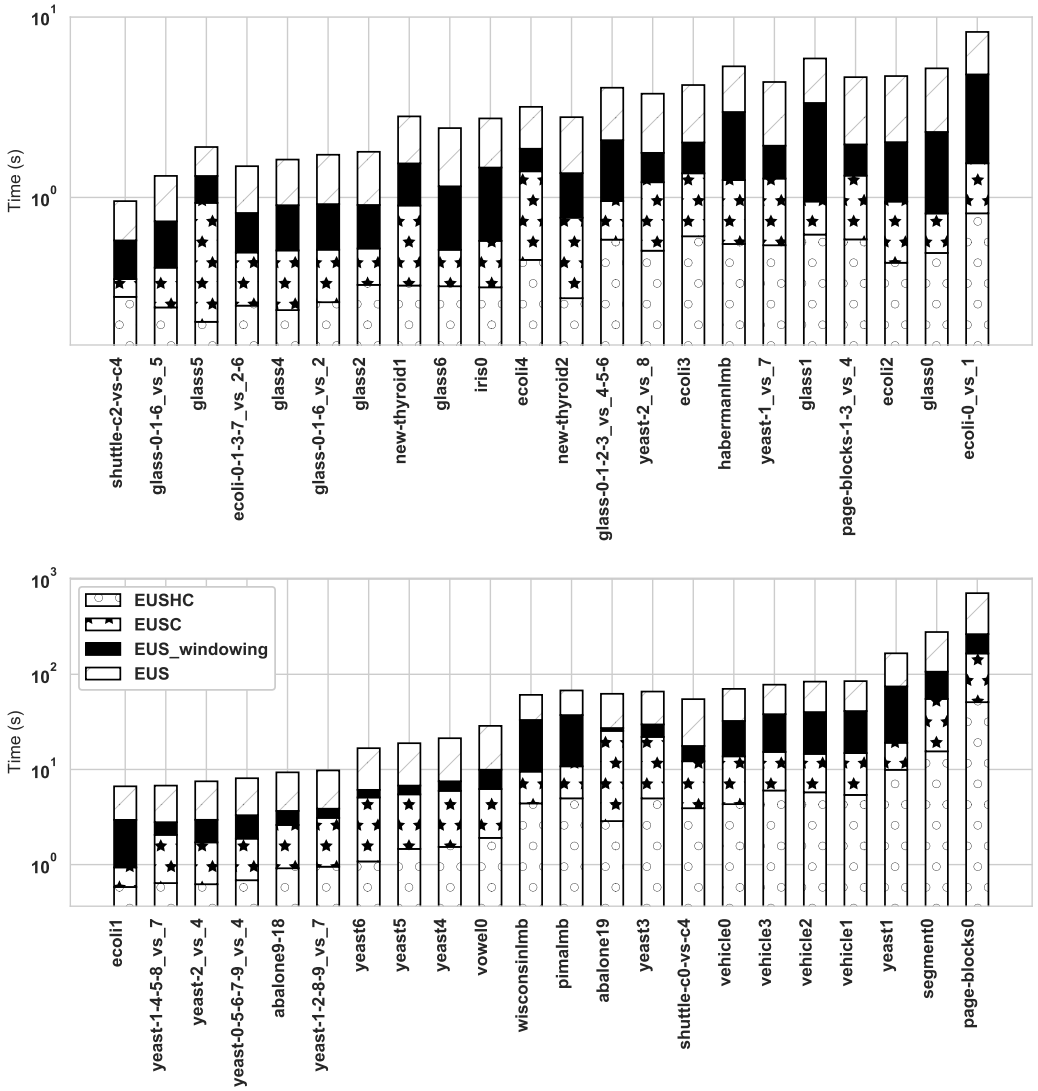


FIGURE 3.8: Comparison of the runtime (in seconds, base 10 logarithmic scale) of the different algorithms over 44 imbalanced datasets, sorted by the runtime of EUS. Runtime of the first 22 datasets (Top), last 22 datasets (Bottom). On average in the 44 datasets, EUS takes 26.44s, EUS_windowing 9.03s, EUSC 6.68s and EUSHC 3.15s.

3.7.2 Reduction of Evaluations

In addition to runtimes, we report the reduction of evaluations provided by the surrogate models. Figure 3.9 displays a histogram with the total number of evaluations of EUSC and EUSHC compared to the 10000 evaluations performed by EUS for each dataset.

- The two surrogate assisted schemes use a significant lower number of evaluations which is roughly a 20% of the 10000 evaluations used by EUS. The number of real evaluations avoided by the surrogate model varies among datasets. When we use fitness approximation (windowing and/or a surrogate model),

the behaviour of the CHC algorithm is also changed. The diversity in the population may be affected, so that, the proportion of chromosomes for which we infer the fitness may be changed in every generation.

Note that CHC does not necessarily create an offspring of NP elements in every generation. Hence, if the number of chromosomes to be evaluated in a generation is very low (less or equal than k_2), we would not take much advantage of the surrogate model. In the experiments, this effect is more noticeable on datasets with either a very small size or high imbalanced ratio.

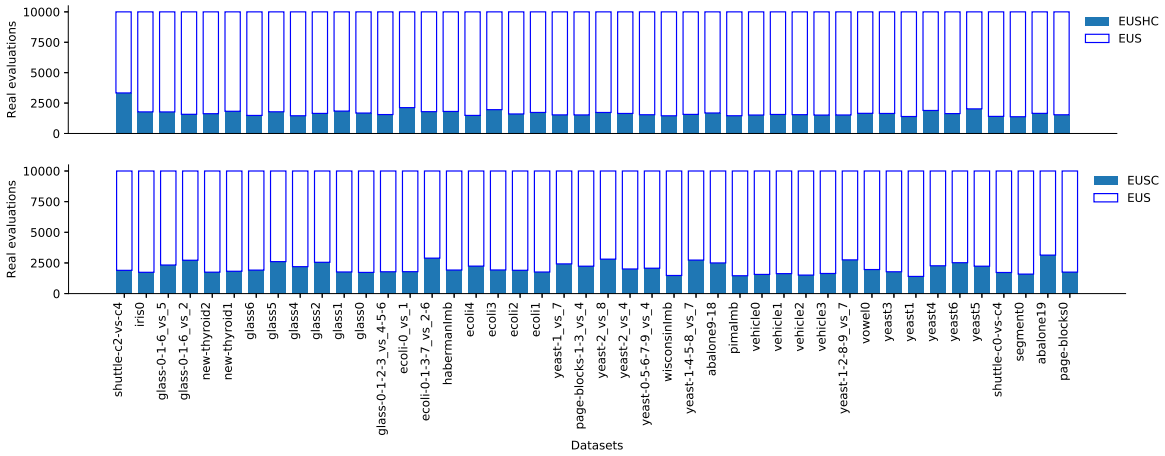


FIGURE 3.9: The number of fitness function calls in the original EUS, EUSC and EUSHC

- It is important to observe that there is a slight difference in the number of objective function calls between EUSC and EUSHC, and the EUSHC consistently saved more real evaluations in most of the datasets. This behaviour may seem unexpected as both methods are using a clustering-based surrogate model to reduce the number of evaluations, so that, both should report a similar number. However, the hybrid approach introduces windowing that changes the behaviour of EUS, and the search. Including windowing may also affect the quality of the chromosomes and, as explained above, the number of chromosomes to be evaluated in each generation.

3.7.3 Classification Performance Comparison

Until now, fitness approximation approaches has demonstrated its time-efficiency with respect to EUS. However, reducing the runtime would not be of any value if the classification performance is massively deteriorated. Table 3.6 shows the average GM performance of all the algorithms in test data. Values in bold indicate that the algorithm at the column achieves the highest GM in the dataset at the row.

Additionally, an extra row at the end displays the number of times that each algorithm wins over 44 datasets. We also compare the number of wins, ties and losses of EUSHC against each reference undersampling algorithm, displayed in Figure 3.10. Looking at the above table and figure, we can observe that:

TABLE 3.6: GM obtained by all comparison methods in 44 imbalanced datasets

Dataset	EUS	EUS_windowing	EUSC	EUSHC
shuttle-c2-vs-c4	**0.9577**	0.6449	0.9414	0.7365
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**
glass-0-1-6_vs_5	0.9214	0.9151	0.9160	**0.9501**
glass-0-1-6_vs_2	0.6383	0.6164	**0.6651**	0.5815
new-thyroid2	**0.9865**	0.9773	0.9831	0.9746
new-thyroid1	**0.9882**	0.9809	0.9859	0.9653
glass6	0.8889	0.9071	**0.9156**	0.9054
glass5	0.8105	0.9076	**0.9600**	0.9103
glass4	**0.8700**	0.8513	0.8613	0.8531
glass2	0.7194	0.6525	**0.7262**	0.6173
glass1	0.7773	0.7010	**0.7941**	0.7367
glass0	0.8009	0.6176	**0.8047**	0.6595
glass-0-1-2-3_vs_4-5-6	0.9525	0.9385	**0.9647**	0.9546
ecoli-0_vs_1	0.9583	0.9312	0.9581	**0.9615**
ecoli-0-1-3-7_vs_2-6	0.6700	**0.7048**	0.6625	0.6865
habermanImb	0.5475	**0.5635**	0.5521	0.5497
ecoli4	0.8984	0.9362	0.8857	**0.9645**
ecoli3	0.8348	0.8153	**0.8500**	0.8097
ecoli2	0.9000	0.8663	**0.9034**	0.8772
ecoli1	**0.8634**	0.8306	0.8554	0.8424
yeast-1_vs_7	**0.7176**	0.7079	0.7068	0.6669
page-blocks-1-3_vs_4	**0.9674**	0.9399	0.9471	0.9294
yeast-2_vs_8	**0.7931**	0.7496	0.7656	0.7668
yeast-2_vs_4	0.9042	0.8774	**0.9156**	0.8930
yeast-0-5-6-7-9_vs_4	0.7685	0.7663	**0.7901**	0.7535
wisconsinImb	**0.9690**	0.9652	0.9600	0.9590
yeast-1-4-5-8_vs_7	0.6569	0.6088	**0.6604**	0.6149
abalone9-18	**0.7269**	0.6772	0.7224	0.6559
pimaImb	0.6943	0.6749	0.6957	**0.7145**
vehicle0	**0.9164**	0.9027	0.9103	0.9016
vehicle1	0.6729	0.6624	0.6512	**0.6926**
vehicle2	0.9259	0.9175	**0.9265**	0.9173
vehicle3	**0.7280**	0.7142	0.7165	0.7204
yeast-1-2-8-9_vs_7	**0.6721**	0.6078	0.6704	0.6500
vowel0	**0.9897**	0.9719	0.9877	0.9831
yeast3	0.8728	0.8740	**0.8752**	0.8550
yeast1	0.6533	0.6501	**0.6600**	**0.6600**
yeast4	0.8050	0.7799	**0.8288**	0.7970
yeast6	**0.8357**	0.8080	0.8034	0.8031
yeast5	0.9634	0.9494	0.9455	**0.9653**
shuttle-c0-vs-c4	0.9960	**0.9968**	0.9960	0.9960
segment0	**0.9881**	0.9870	0.9876	0.9858
abalone19	0.6258	0.6061	**0.7214**	0.6556
page-blocks0	**0.9117**	0.9038	0.9096	0.9085
Wins	**18**	**4**	**18**	**8**

- Despite using about 80% more evaluations, EUS does not always provide the best classification performance. This result shows that the use of approximations may result in even better results, reducing overfitting of the training set. As stated in Section 5.2.1, using the training data is already an approximation of how well this data represents the concept to be learned.
- In this experiment, we can highlight EUSC, which seems to be very competitive with respect to EUS, obtaining the same number of wins out of the 44 datasets. EUSHC also finds the best solution in 8 out of 44 datasets.
- Over 44 datasets, EUSHC shows a greater number of wins with respect to EUS_windowing. It is predicted that EUSHC loses EUS and EUSC frequently

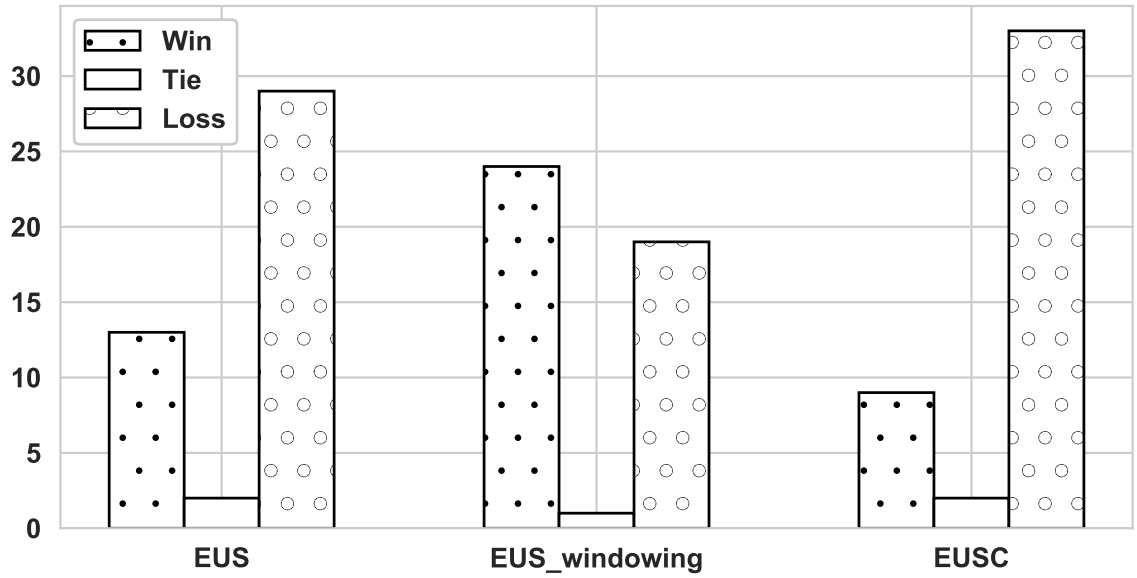


FIGURE 3.10: Comparison of EUSHC and reference undersampling algorithms with respect to the number of wins, ties, and losses over 44 imbalanced datasets.

as it applies two stages of approximation. However, figures in Table 3.6 show a very reduced difference in GM between our hybrid approach and the other algorithms. Note that the difference in GM is only more noticeable in those datasets either having high imbalance ratio or low number of samples with high IR.

In summary, the proposed EUSHC highly reduces the computational cost of the EUS algorithm (about an 88.08% on average - from 26.44s to 3.15s), and the classification performance seems comparable to EUS and the other approximation methods. Looking at all the results presented in this contribution, when the number of instances is low, it is reasonable not to use a surrogate or windowing approach as the original EUS will not suffer from a high computational cost. However, in larger datasets, the benefits of the proposed approach are promising.

3.8 Summary

In this chapter, we have achieved our first research objective by proposing a novel evolutionary search strategy with the integration of two-stage clustering. The clustering-based surrogate model allows fitness to be approximated, which reduces the computational cost and thus accelerate the evolutionary search. Several characteristics of our proposed framework are stressed as follows:

- The transformation of binary chromosomes into real coding ones at which the overall location of the selected instances in that solution is summarised.

- Considering all of the instances at the fitness evaluation for the representative chromosomes, which is different from the Windowing method (Bacardit et al., 2004) using partial training set.
- Approximating fitness of other chromosomes based on the Euclidean distance correlation with the representative selected from the cluster they belong to.

To validate our approach, we have carried an extensive experimental framework on 44 imbalanced datasets to verify runtime, reduction rate and GM performance. The obtained numerical results can demonstrate the effectiveness of our proposal compared to the original EUS method. The experiments show that we are capable of drastically reducing the runtime required to perform undersampling without significantly losing classification performance. In comparison with alternative approaches to approximate fitness values in evolutionary undersampling (namely windowing), our method has demonstrated to consistently reduce the runtime and maintain high quality solutions. In addition, the hybrid surrogate model EUSHC has been examined under the same experimental setting. The proposed approach approximates fitness values of the chromosomes using a clustering-based surrogate model together with a windowing approach. The entire search is guided by approximate fitness values aiming to highly reduce the computational cost. From the obtained results, we can highly reduce the runtime required to perform EUS, especially in larger datasets, without incurring in a noticeable classification performance loss. As such, the proposed approach contributes towards the design of fitness approximation models based on surrogate models in EA for IS/undersampling.

We have completed the first phase of the dissertation with a valid solution tackling **IS** for the established Objective 1. This serves as a first achievement of accelerating population-based meta-heuristic search strategy. However, **IR** does not comprise only **IS** on the combinatorial search space, but also **IG** on the continuous domain. In the following chapter, we would develop another search strategy for IG, aiming to fulfil Objective 2 with a simple, yet effective and time-efficient pattern search approach.

Chapter 4

Single-Point Memetic Structure with Accelerated Local Search for IR

4.1 Introduction

In the first contribution, we have dealt with the acceleration for a population-based meta-heuristic search strategy. The proposed solution has addressed the high computational cost of fitness computation of an evolutionary search by the integration of two-stage clustering. The proposed idea of fitness approximation is applicable for a population-based search framework where multiple solution candidates are required to be evaluated at once. The proposed framework has offered a novel way of computing fitness at which the real fitness evaluation is conducted in only several representative solution candidates, while others can be approximated using the selected representative solutions. It is argued that a successful and validated solution has been offered to accelerate meta-heuristic search strategy and that is a novel search framework for **IS** in the combinatorial space. However, the introductory chapter has also discussed the Single-Point Memetic Structure as another prominent search method operating in the continuous domain. Hence, for a comprehensive set of solutions covering two types of **IR** (i.e. **IS** and IG), this chapter is devoted to Single-Point Memetic Structure, focusing on **IG** in the continuous search space.

The novel search framework in this chapter is characterised by a mechanism that while exploiting the structure of the optimisation algorithm allows a substantial reduction of the computational complexity (i.e. number of distance computations) of the objective function without approximations, see Section 4.2. Thus, the goal of this work is not to tackle big datasets and the memory limitations associated to it, but to devise a very fast and reliable **IR** process that could be combined together with the approaches provided in (Triguero et al., 2015b) when very big datasets need to be addressed. In this search framework, we will highlight two key components: pattern search and memetic computing.

Bearing in mind the elevated computational cost of the fitness function, we propose a simple and yet effective domain-specific MC approach for IR. The proposed MC approach is composed of a novel domain-specific implementation of local search hybridised with a global evolutionary operator. The local search exploits the logic of the Generalised Pattern Search that performs an implicit variable decomposition technique and perturbs the elements of a candidate solution one by one (Neri and Rostami, 2021). In contradistinction with existing population-based approaches that create new solutions perturbing multiple variables at once, we exploit the fact that the proposed local search produces candidate solutions that are only "slightly" different w.r.t the previous fitness evaluation. Based on this fact, we devise a mechanism to drastically reduce the cost of the objective function when using the NN algorithm as base classifier. The global search operator is a simple resampling mechanism followed by crossover while an elite memory slot retains the solution with the best performance. The key idea lies in keeping a single-point approach to highly accelerate the objective function evaluation while using a global operator to avoid getting stuck in local optima.

The remainder of this chapter is organised in the following way. Section 4.2 describes and justifies the proposed method. Section 4.3 presents the experimental setup while Section 4.4 shows and discusses the results. Finally, Section 4.5 provides the summary of this chapter.

4.2 Methodology

From the description in Section 2.4.2, we may characterise **IR** as an optimisation problem with the following considerations:

- the problem is large-scale and its number of variables ($p \times m$) can be extremely high depending on the size of the dataset
- due to the large number of variables, the problem is likely to be hard to solve and the fitness landscape could be highly multimodal
- even if it were multimodal, an excessive exploitation of the basin of attraction may yield an overfitted solution, that is a solution that performs well on the training set but not on the test set
- each objective function call (or fitness evaluation) is computationally expensive due to calculation of multiple Euclidean distances

In order to address the **IR** problem, a domain-specific MC approach that takes into account the considerations above is here proposed. The proposed MC approach, namely Single-Point Memetic Structure with Accelerated Local Search (SPMS-ALS)

is population-less and designed according to the bottom-up logic reported in (Iacca et al., 2012). SPMS-ALS perturbs a single solution and makes use of one more memory slot to store the elite solution, that is the best solution ever found. A novel domain-specific accelerated local search implementation is here proposed. Section 4.2.1 describes the local search operator employed in SPMS-ALS while Section 4.2.2 illustrates how the local search logic is exploited to accelerate the calculation of the objective function. The proposed SPMS-ALS makes also use of a simple global search operator illustrated in Section 4.2.3. Finally, Section 4.2.4 discusses and justifies the design of SPMS-ALS.

4.2.1 Local Search Operator

Given the candidate solution x as a 1-dimensional vector of n design variables, presented in 2.4.2. Let $\mathbf{e^i}$ be the i^{th} versor (that is a vector of modulus equal to 1) of a basis in an n-dimensional space, that is a vector whose elements are all zeros except from the i^{th} element which is one (Neri, 2019):

$$\mathbf{e^i} = (0, 0, \ldots, 1, \ldots, 0, 0)$$

The local search works on the candidate solution $\mathbf{x}$ to locally improve it. The following greedy implementation of a Generalised Pattern Search has been used, see (Neri and Rostami, 2021). The algorithm perturbs each feature value of an instance at a time in its feasible range and then check if any improvement is found. Specifically, let $\mathbf{x}$ be the base vector (the best solution found at the time), for each design variable i from 1 to n the algorithm explores at first

$$\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$$

where $\mathbf{x^t}$ is a trial vector and the scalar ρ is the step-size (exploratory radius). For each index i, the algorithm attempts to explore the opposite orientation of the direction identified by $\mathbf{e^i}$ if the first attempt fails, that is

$$\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$$

As a remark, the asymmetric step-size is designed to avoid to revisit the same solution (vector), see (Neri and Rostami, 2021). As soon as $\mathbf{x^t}$ outperforms $\mathbf{x}$, that is $f\left(\mathbf{x^t}\right) \geq f\left(\mathbf{x}\right)$, the trial vector $\mathbf{x^t}$ replaces the base vector $\mathbf{x}$.

Note that when applying the above perturbations, the resulting values in the vector x^t could be outside of the bounds $[x_{low}, x_{high}]$. On the basis of preliminary tests we employed a toroidal handling of the bounds, i.e. for $x_i \in [x_{low}, x_{high}]$, if $x_i > x_{high}$ it

is reinserted by reassignment:

$$x_i = x_{low} + \left((x_i - x_{high}) \lfloor \frac{(x_i - x_{high})}{(x_{high} - x_{low})} \rfloor (x_{high} - x_{low}) \right)$$

while if $x_i < x_{low}$ it is reinserted by reassignment

$$x_i = x_{high} - \left((x_{low} - x_i) - \lfloor \frac{(x_{low} - x_i)}{(x_{high} - x_{low})} \rfloor (x_{high} - x_{low}) \right)$$

where the parentheses $\lfloor \cdot \rfloor$ indicate the truncation to the lower integer.

As an example, if we are in the range [0,1], and the resulting value x_i of the perturbation is 1.1, the toroidal handling will begin from the beginning of the range, producing a 0.1. Conversely, if x_i were to be below 0, e.g. -0.1, this circular handling would provide 0.9. This ensures that the investigated values are within the range. Also, by forcing the perturbation to go to the other side of the bound, we increase the exploratory abilities of the method before reducing the radius ρ. This strategy provided good results in preliminary tests in comparison with other alternatives. If after the entire exploration along the n directions no improved solution $\mathbf{x^t}$ is found, then the radius ρ is reduced by a reduction rate. The local search is interrupted when either a budget condition is met or when the radius ρ is smaller than a pre-arranged precision. For sake of clarity Algorithm 4 shows the local search operator used in SPMS-ALS.

4.2.2 Accelerated Local Search

The proposed local search makes use of the search logic outlined in Algorithm 4 and integrates within it a domain-specific procedure (i.e. accelerating fitness computation) to reduce the computational time of the algorithm. As highlighted in Section 2.4.2, when the NN algorithm is used as based classifier, most of the high computational cost of the **IR** problem is due to the calculation of $l \times p$ Euclidean distances. However, the local search moves

$$\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$$

and

$$\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$$

affect only one design variable that is only one entry of the **RS** matrix.

Algorithm 4 Local Search of the family of Pattern Search used by SPMS-ALS

Input: $\mathbf{x}$
Output: $\mathbf{x}$ with different feature values

while local budget and precision conditions are not met **do**
  $\mathbf{x^t} = \mathbf{x}$
  for $i = 1 : n$ **do**
    $\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e^i}$
    Apply toroidal handling of the bounds
    if $f\left(\mathbf{x^t}\right) \geq f\left(\mathbf{x}\right)$ **then** ▷ Compute $f\left(\mathbf{x}\right)$, see Algorithm 5
      $\mathbf{x} = \mathbf{x^t}$
    else
      $\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e^i}$
      Apply toroidal handling of the bounds
      if $f\left(\mathbf{x^t}\right) \geq f\left(\mathbf{x}\right)$ **then**
        $\mathbf{x} = \mathbf{x^t}$
      end if
    end if
  end for
  if $\mathbf{x}$ has **not** been updated for N_{max} times **then**
    reduce ρ ▷ Reduced by reduction rate ρ_{Red} and stopped at thresholdρ_{Thr}
  end if
end while
RETURN $\mathbf{x}$

It is a common practice to just store the shortest distance and instance ID/number when accumulating the correct classifications for NN, and disregard any intermediate distance computations. Hence, most of the *IR* studies in the literature that use k-NN perform the calculation of $l \times p$ Euclidean distances, e.g. (Triguero, García, and Herrera, 2011). Unlike these studies, the accelerated objective function maintains a global *distance matrix*.

As a consequence, if we build a distance matrix $\mathbf{D}$ associated with $\mathbf{x^t}$, this differs by only one column from the matrix $\mathbf{D}$ associated with $\mathbf{x}$. When the objective function $f\left(\mathbf{x^t}\right)$ is calculated according to Algorithm 1, there is no need to recompute $l \times p$ Euclidean distances since $l \times (p-1)$ elements have already been computed and appropriately stored.

Thus, when Algorithm 4 is applied, each objective function call requires the calculation of only l Euclidean distances. This fact can be effectively represented as the modified objective function used by the local search outlined in Algorithm 5.

Our proposed local search performs once at the beginning the objective function call as in Algorithm 1 and then integrates Algorithm 5 into each $f\left(\mathbf{x^t}\right)$ function call for the rest of its execution.

Algorithm 5 Objective Function $f\left(\mathbf{x^t}\right)$ of the Accelerated Local Search

Input: matrix $\mathbf{TR} = \left[a_{i,j}\right]$, matrix $\mathbf{D}$ associated with the base vector $\mathbf{x}$, and trial vector $\mathbf{x^t}$
Output: the objective function value *Acc*

Build the matrix $\mathbf{RS} = \left[b_{i,j}\right]$ from $\mathbf{x^t}$
Update the matrix of Euclidean distances $\mathbf{D} = \left[D_{i,j}\right]$ by recalculating the l elements of the pertinent column
correct_classification = 0
for each row of the matrix **D** **do**
    Find the smallest figure and save its row and column indices
    Select, from **TR** and **RS**, the instances corresponding to the calculated indices
    Check the labels of the two instances
    if the labels coincide **then**
        correct_classification += 1
    end if
end for
Calculate *Acc*

4.2.3 Evolutionary Global Search Operator

At the beginning of the optimisation, a matrix **RS** (i.e. Equation 2.4) is randomly sampled from the matrix **TR** (i.e. Equation 2.3) and from **RS** the corresponding base vector $\mathbf{x}$ constructed and inputted into the local search operator. The local search is continued until the stopping criteria conditions on budget and precision are met. The local search returns a (possibly improved) solution $\mathbf{x}$. Then, a new solution $\mathbf{x^r}$ is generated by randomly sampling a new **RS** matrix from **TR** and constructing the corresponding vector. A uniform crossover is applied to $\mathbf{x^r}$ and $\mathbf{x}$ to generate a new trial vector $\mathbf{x^t}$.

In order to explain the functioning of this crossover, let us consider a candidate solution $\mathbf{x}$ and let us remind it that it corresponds to a matrix **RS** whose rows are instances and columns are features. By applying a matrix partitioning we may represent **RS** as a vector of row vectors

$$\mathbf{RS} = \begin{bmatrix} \mathbf{I_1} \\ \mathbf{I_2} \\ \dots \\ \mathbf{I_p} \end{bmatrix}$$

Similarly, we may consider the random solution $\mathbf{x^r}$ and represent the corresponding $\mathbf{RS^r}$ matrix

$$\mathbf{RS^r} = \begin{bmatrix} \mathbf{I_1^r} \\ \mathbf{I_2^r} \\ \dots \\ \mathbf{I_p^r} \end{bmatrix}$$

whose instances are randomly selected from **TR**.

The proposed crossover generates a trial vector $\mathbf{x^t}$ by randomly selecting some rows from **RS** and some rows from $\mathbf{RS^r}$. Each row of the resulting matrix $\mathbf{RS^t}$ has a gene-resampling probability Gr to be selected from **RS** and $1 - Gr$ probability to be selected from $\mathbf{RS^r}$. It must be remarked that a crossover that perturbs single elements of **RS** instead of entire rows would yield a candidate solution which could be noisy (i.e. not have the right class label), and therefore, not meaningful from a classification point of view.

The gene-resampling probability Gr expresses the rate of the instances in **RS** which are replaced by other instances sampled from **TR**. Algorithm 6 describes the crossover mechanism.

Algorithm 6 Crossover between $\mathbf{x}$ and $\mathbf{x^r}$

1: **INPUT** base vector $\mathbf{x}$ and random vector $\mathbf{x^r}$
2: Build the matrices $\mathbf{RS} = [\mathbf{I_i}]$ and $\mathbf{RS^r} = [\mathbf{I_i^r}]$
3: $\mathbf{RS^t} = [\mathbf{I_i^t}] = \mathbf{RS}$
4: **for** $i = 1 : p$ **do**
5: Generate a random number *rand*
6: **if** $rand < Gr$ **then**
7: $\mathbf{I_i^t} = \mathbf{I_i^r}$
8: **end if**
9: **end for**
10: From $\mathbf{RS^t}$ calculate $\mathbf{x^t}$
11: **OUTPUT** the trial vector $\mathbf{x^t}$

The local and global search operators are repeated until the global budget conditions are met. The framework of the proposed SPMS-ALS is illustrated in Algorithm 7.

4.2.4 Algorithmic Design

The proposed SPMS-ALS follows a bottom-up strategy as suggested in (Iacca et al., 2012): we implemented within the algorithmic operators the necessary countermeasures to address each challenge associated with the problem.

The structure of the local search has been selected to address the large scale nature of the **IR** problem, that is for a large dataset, the matrix **RS** can easily have hundreds if not thousands of rows. The proposed local search perturbs the variables separately

Algorithm 7 Framework of the SPMS-ALS for IR

Input: base vector $\mathbf{x}$
Output: $\mathbf{x}$ with new feature values

Randomly generate a base vector $\mathbf{x}$ in $[0,1]^n$ and calculate $f(\mathbf{x})$ according to Algorithm 1
Assign the elite $\mathbf{x^{elt}} = \mathbf{x}$
while global budget conditions are met **do**
    Apply the Accelerated Local Search to the base vector $\mathbf{x}$ according to Algorithm 4 with the objective function $f(\mathbf{x^t})$ calculated according to Algorithm 5
    if $f(\mathbf{x}) \geq f(\mathbf{x^{elt}})$ **then**
        Update the elite $\mathbf{x^{elt}} = \mathbf{x}$
    end if
    Randomly generate a vector $\mathbf{x^r}$ in $[0,1]^n$
    Apply Crossover between $\mathbf{x}$ and $\mathbf{x^r}$ according to Algorithm 6 and generate a new trial vector $\mathbf{x^t}$
    Calculate $f(\mathbf{x^t})$ according to Algorithm 5
    if $f(\mathbf{x^t}) \geq f(\mathbf{x^{elt}})$ **then**
        Update the elite $\mathbf{x^{elt}} = \mathbf{x^t}$
    end if
    Assign $\mathbf{x} = \mathbf{x^t}$
end while

and thus implicitly performs a variable decomposition. Approaches of this type have been proved effective for large scale problems, see (Ros and Hansen, 2008; Tseng and Chen, 2008; Li and Yao, 2012).

This observation was reported in the experimental study in (Caraffini, Neri, and Iacca, 2017). Large scale problems are by no means easier than low-dimensional problems. However, since in practice the computational budget cannot grow exponentially with the problem dimensionality, only a very limited portion of the decision space is explored. Under these experimental conditions, the algorithm "sees" the problem as separable: average Pearson and Spearman coefficients of the variables approach zero independently on the problem when the number of dimensions grows, see (Caraffini, Neri, and Iacca, 2017).

The high computational cost of each function call is addressed by the acceleration mechanism outlined above: only the elements of one column of the Euclidean matrix $\mathbf{D}$ and not those of the entire matrix are calculated at each function call. The population-less structure of SPMS-ALS has also been chosen taking into consideration the computational cost. The proposed SPMS-ALS naturally devotes most of the computational budget (in terms of function calls) to the local search. On the contrary, the global search operator performs only sporadic function calls. This logic perfectly suits the needs of reducing the computational cost since the global operator requires the expensive objective function as in Algorithm 1 the local search

operator uses its computationally cheaper version as in Algorithm 5.

In order to address the multimodality of the fitness landscape and prevent that the algorithm converges to a suboptimal solution, we combined the Accelerated Local Search with the simplistic global search described above. It must be noted that the global search makes use of part of design variables (genotype) of previously improved solutions. The best solution ever found is saved and stored in an elite slot and called $\mathbf{x}^{\mathbf{elt}}$. The elitism guarantees that previously detected promising solutions are available at the end of the run. Furthermore, the gene-resampling mechanism, happening at the instance level (considering the rows as building blocks) complements the local search that happens at the level of the elements of **RS**.

At last, the restarting local search logic combined with a limited local search budget is an important countermeasure to prevent from overfitting: an excessive local search budget is likely to yield an overly specialised solution that performs poorly when the solution is tested on a new dataset. This characteristic is experimentally analysed in Section 4.4.3.

4.3 Experimental Framework

This section presents the used datasets (Section 4.3.1) and introduces several **IR** techniques that will be used for comparison with our proposal (Section 4.3.2). Finally, the parameter configuration is explained (Section 4.3.3).

4.3.1 Datasets

In the experimental study, we have examined 40 small and 17 medium multi-class datasets from the KEEL dataset repository (Triguero et al., 2017). They are grouped into small and medium categories based on the number of samples. It is worth noting that the datasets used in this chapter are roughly balanced, which is different from the ones used for imbalanced classification in Chapter 3. This is because the purpose of this chapter is to do **IR** in general, not for a specific case study like imbalanced classification. As a result, samples of all classes are reduced rather than only the majority class. The properties of these datasets including name (**Dataset**), the number of samples (**Samp**), the number of attributes (**Att**), the number of classes (**%Class**) are summarised in Table 4.1. Each dataset is partitioned using a 10-fold stratified cross-validation (10-fcv) procedure, see (Refaeilzadeh, Tang, and Liu, 2009). Thus, the performance of each dataset is reported by an average of the 10 folds. All of the experiments with these datasets have been conducted on computers at which each has 2 x 20 core processors (Intel Xeon Gold 6138 20C 2.0GHz CPU) and 192 GB RAM.

TABLE 4.1: Summary description for small (Sample $<$ 2000) and medium (Sample $>=$ 2000) datasets.

Dataset	Samp	Att	Class	Dataset	Samp	Att	Class
Abalone	4174	8	28	Monks	432	6	2
Appendicitis	106	7	2	Movement_libras	360	90	15
Australian	690	14	2	Newthyroid	215	5	3
Autos	205	25	6	Nursery	12960	8	5
Balance	625	4	3	Page-blocks	5472	10	5
Banana	5300	2	2	Penbased	10992	16	10
Bands	539	19	2	Phoneme	5404	5	2
Breast	286	9	2	Pima	768	8	2
Bupa	345	6	2	Ring	7400	20	2
Car	1728	6	4	Saheart	462	9	2
Chess	3196	36	2	Satimage	6435	36	7
Cleveland	297	13	5	Segment	2310	19	7
Contraceptive	1473	9	3	Sonar	208	60	2
Crx	125	15	2	Spambase	4597	57	2
Dermatology	366	33	6	Spectheart	267	44	2
Ecoli	336	7	8	Splice	3190	60	3
Flare-solar	1066	9	2	Tae	151	5	3
German	1000	20	2	Texture	5500	40	11
Glass	214	9	7	Thyrod	7200	21	3
Haberman	306	3	2	Tic-tac-toe	958	9	2
Hayes-roth	133	4	3	Titanic	2201	3	2
Heart	270	13	2	Twonorm	7400	20	2
Hepatitis	155	19	2	Vehicle	846	18	4
Housevotes	435	16	2	Vowel	990	13	11
Iris	150	4	3	Wine	178	13	3
Led7digit	500	7	10	Wisconsin	683	9	2
Lymphography	148	18	4	Yeast	1484	8	10
Magic	19020	10	2	Zoo	101	16	7
Mammographic	961	5	2				

4.3.2 Comparison Algorithms

In order to understand the benefits of the proposed MC approach, we first define two baselines:

- Nearest Neighbour (1NN): we use the NN algorithm (k=1) employing the entire **TR** for training, without any pre-processing. The performance of the NN in **TR** is calculated following a *leave-one-out* validation scheme. This serves of a baseline to understand the benefits of IR.
- LSIR: the local search presented and used in (Neri and Triguero, 2020). LSIR is essentially the basic pattern search shown in Algorithm 4 without any acceleration, that is the local search by using the basic fitness function as in Algorithm 1.

In addition, we test the performance of the proposed approach against the current state-of-the-art in IR. SPMS-ALS belongs to the family of positioning adjustment

methods (see (Triguero et al., 2012)), which are, to date, the best performing IRs methods in the literature and follow a similar algorithmic structure to the proposed approach. In (Neri and Triguero, 2020), we showed the classification performance of the local search against the entire family of positioning adjustment methods. For the sake of simplicity, here we only report the comparison against the most competitive methods. SPMS-ALS can be categorised as a pure instance generation approach, as we perform a continuous search. Thus, we choose the following metaheuristics **IG** methods to compete against SPMS-ALS:

- Scale Factor Local Search Differential Evolution (SFLSDE): this memetic approach optimises the positioning of prototypes using an implementation of differential evolution (Triguero, García, and Herrera, 2011).
- Particle Swarm Optimisation (PSO): this algorithm modifies the position of an initial set using PSO rules, aiming to maximise the classification performance (Nanni and Lumini, 2009).

Additionally, to compare against more recent meta-heuristics, we have adapted a recent metaheuristic, proposed for the continuous domain, to tackle IR.

- Linear Population Size Reduction of the Success-History based Adaptive Differential Evolution (LSHADE) (Tanabe and Fukunaga, 2014): this approach is developed from Success-History based Adaptive Differential Evolution (SHADE) (Tanabe and Fukunaga, 2013) and Adaptive Differential Evolution with Optional External Archive JADE (Zhang and Sanderson, 2009). It makes use of success-history and also applies the population size reduction to progress the search. Note that this metaheuristic has not been previously used for IR, but due to its similarity to JADE, we used the design ideas from (Triguero, García, and Herrera, 2011) to adapt it to solve the **IR** problem.

These approaches evolve a population of solutions, whilst our method only evolves a single solution (or more precisely two solutions the trial solution $\mathbf{x^t}$ and the elite $\mathbf{x^{elt}}$). However, similar to our method, both approaches start off from a random (stratified) subset of the training set **TR** (one for each individual of their population), which keeps the original distribution of instances per class. Thus, the reduction rate is also defined by a parameter that determines how much we want to reduce **TR**.

As a further remark, while PSO, SFLSDE were existing meta-heuristics that have been adapted to solve the **IG** problem, the proposed SPMS-ALS has been expressly designed to solve this problem effectively in terms of accuracy efficiently in terms of runtime. This design approach follows the bottom-up design logic of MC (Iacca et al., 2012; Neri and Cotta, 2012) and can be observed in both accelerated local search logic and crossover operator. Another remark is that LSHADE was used in

the continuous domain to solve benchmark functions, this meta-heuristic design is first time adapted to solve the **IG** problem in this study.

In (Triguero, García, and Herrera, 2011), the authors showed that using a random selection as initialisation mechanism is not usually appropriate, and the hybridisation of an **IS** step followed by **IG** was suggested to replace this random initialisation. More specifically, the use of a Steady-State Memetic Algorithm (SSMA) (García, Cano, and Herrera, 2008) demonstrated empirically to provide an excellent starting point, which means a good selection of instances per class and a good reduction rate (automatically determined by the **IS** step). To the best of our knowledge, the hybrid **IR** algorithm, SSMA-SFLSDE (Triguero, García, and Herrera, 2011), remains the best performing method for **IR** in both accuracy and reduction rate. To establish a fair comparison against it, we will also hybridise the proposed MC approach and the local search with SSMA (see Section 4.4.5).

Whilst the hybrid IS/IG method, SSMA-SFLSDE has not been outperformed to date, in order to assess the potential of the proposed approach, we add a comparison with recently published algorithms belonging to the family of IS. We included an approach based on local sets (Leyva, González, and Pérez, 2015) and a method based on instance ranking (Cavalcanti and Soares, 2020). Note that these methods follow a completely different approach to produce a reduced set from the training set. Thus, we cannot set up the same computational budget that we do for the rest of the comparison algorithms, as they do not follow an optimisation-based approach (Section 4.3.3).

The study in (Leyva, González, and Pérez, 2015) contains three **IS** methods, namely Local Set Smoother (LSSm), Local Set Core (LSCo) and Local Set Border (LSBo). LSSm aims at achieving the highest accuracy regardless of the reduction, while LSCo seeks at obtaining the highest reduction with acceptable accuracy. LSBo addresses both accuracy and reduction rate with the same priority. For this reason, LSBo has been selected for comparison against the proposed memetic approach.

The main idea of (Cavalcanti and Soares, 2020) is to exploit the relationship among members in the training set by computing a rank for each element. A rank of an instance introduces the correlation between itself and others in the training set. Instances with higher ranks are likely to be selected compared to those holding a low rank. In Section 4.4.4, we report the performance of Ranking-based Instance Selection (RIS1) as it showed in (Cavalcanti and Soares, 2020) to display the best performance, among multiple variants, w.r.t both measures of accuracy and reduction rate.

4.3.3 Parameter Settings

This section presents the parameter configuration for all the methods employed in this chapter, including the accelerated local search outlined in Algorithms 4 and 5 and the entire memetic framework SPMS-ALS shown in Algorithm 7. We will discuss below the computational budget and different hyper-parameters used in SPMS-ALS.

Computational Budget: In the previous studies on IR, the computational budget has usually been set empirically to a number of iterations (in search-like algorithms), which remained fixed for all datasets (Triguero, García, and Herrera, 2011; Neri and Triguero, 2020). Just like in scalability studies for ordinary optimisation problems, in the **IR** problem, the complexity of the search space grows exponentially with the problem size, (Caraffini, Neri, and Iacca, 2017). On the other hand, **IR** poses a further challenge that is the risk of overfitting and underfitting. An incorrect local search budget allocation is likely to lead to overfitting in small datasets and underfitting appears in larger datasets. In order to overcome this challenge and propose a standard for setting the computational budget, we have here conducted an extensive experimental study. Note that keeping the same number of evaluations through the different comparison methods will also help establish a fairer comparison (which has not been the case in previous studies).

Among the various properties of a dataset, the number of instances in training data, i.e. the number of rows l of the matrix **TR** and the number of features (**Features**) in an instance at each dataset are the two important factors that define the size of the problem and need to be considered when the budget is allocated. We acknowledge that other factors may be also required into consideration such as the number of classes or the ratio of samples among claghsses. However, this simple yet effective approach of parameter setting has proven to significantly reduce the unnecessary allocated number of evaluations and thus can help mitigate overfitting in small datasets and underfitting in larger ones. Since RIS1 and LSBo do not perform any evaluation of their reduced set against the training set, we cannot apply a computational budget.

In the original setting based on 40 small datasets, SFLSDE (Triguero, García, and Herrera, 2011) and LSIR (Neri and Triguero, 2020) use approximately 20,000 and 30,000 evaluations, respectively. We took these values as a reference and set three levels in our experimental study: lower, comparable, and greater than the reference ones. Table 4.2 displays, for all the algorithms considered in this study that employ local search, the three local search budgets scenarios. From the total number of evaluations presented in Table 4.2, we split the evaluations into two parts when

SSMA is included. SSMA takes $3 \times l$ evaluations in Setting 1 and $5 \times l$ in Setting 2 and 3. The budget allocated to SSMA is indicated as SSMA_Eval. The rest of the evaluations is used for **IG** methods (LSIR, SPMS-ALS, PSO, SFLSDE, LSHADE).

TABLE 4.2: Changing the number of evaluations considering training size and features for fairer comparison.

	Computational Budget		
Algorithm	Setting 1	Setting 2	Setting 3
SSMA	SSMA_Eval = $3 \times l$	SSMA_Eval = $5 \times l$	SSMA_Eval = $5 \times l$
SFLSDE	$2.5 \times$ **Features** $\times l$ $-$ SSMA_Eval	$5 \times$ **Features** $\times l$ $-$ SSMA_Eval	$10 \times$ **Features** $\times l$ $-$ SSMA_Eval
LSIR			
SPMS-ALS			
PSO			
LSHADE			

Hyper-Parameters: The proposed SPMS-ALS contains some parameters to set to coordinate global and local search. In particular, the following parameters are fundamental to coordinate the interruption of accelerated local search and restart of global search.

- $N_{\max}$: the maximum number of times the local search accepts a new trial solution $\mathbf{x}^{\mathbf{t}}$ with the same objective function (*Acc*) as that of the previous trial solution i.e. maximum number of search moves allowed on a plateau
- ρ_{Red}: the reduction rate of the exploratory step ρ after the same fitness has been calculated $N_{\max}$ times, used in Algorithm 4
- ρ_{Thr}: the threshold after which the local search is stopped, $\rho \leq \rho_{Thr}$. When ρ reaches this defined threshold, we reset it to the original value, used in Algorithm 4, line 17
- Gr: the gene-resampling probability as in Algorithm 6

Since small datasets have only few samples per class, a large Gr value is required to make a significant refresh of the candidate solution. On the contrary, medium datasets inherently pose a highly multivariate problems. Hence smaller Gr values result into a major alternation of the candidate solution. We may consider this effect analogous to the setting of the crossover rate in Differential Evolution with respect to the number of dimensions of the problem, see (Neri, Iacca, and Mininno, 2011). On the other hand, numerous configurations have been examined to find a set of parameters that can guarantee a robust performance of SPMS-ALS on both small and medium datasets. In this study, we report the performance of SPMS-ALS using the following parameters: initial radius $\rho = 0.4$, $N_{\max} = 3$, $\rho_{Red} = 0.25$, $\rho_{Thr} = 0.005$

and Gr = 0.5 for small and 0.05 for medium datasets, respectively. Apart from the budget condition, which is investigated for all the comparison algorithms as described above, the rest of the parameters for all the algorithms are established as recommended by the authors. All the details are presented in Table 4.3. Following the experimental setup in (Triguero et al., 2012), the reduction rate parameter is set to 95% for small size datasets, and 98% for medium datasets.

TABLE 4.3: Parameters used for comparison algorithms

Algorithms	Parameter setting
SFLSDE	PopulationSize = 40, iterSFGSS =8, iterSFHC = 20, Fl = 0.1, Fu = 0.9
LSIR	initial ρ = 0.4
SSMA	Population = 40, Cross = 0.5, Mutation=0.001
PSO	SwarmSize = 40, C1 = 1, C2 = 3, Vmax = 0.25, Wstart = 1.5, Wend = 0.5
NN	k = 1, Euclidean distance
RIS	Thresholds = [0.0, 0.1, 0.2, ..., 0.9, 1.0]
LSBo	–
LSHADE	ArchiveSize = 1.4, PopulationSize = 40, MemorySize = 5

4.4 Analysis of Results

In this section, we analyse the results obtained from different sets of experiment, divided into multiple subsections, to empirically examine the individual effect of each component we propose in our algorithm. In the analysis, our aims are:

- To understand how well LSIR works in different settings of evaluations (Subsection 4.4.1). We discuss multiple aspects of LSIR such as the change in performance measured by accurate rate to see the overfitting or underfitting effects on the learning process. In addition, the number of evaluations that has been used and saved for each dataset is reported.
- To measure the actual savings in terms of runtime when the proposed acceleration is integrated within LSIR (Subsection 4.4.2). We report in detail the absolute and percentage figures of the runtime savings.
- To examine the performance enhancement due to the proposed memetic components (Subsection 4.4.3), in comparison with the local search. We report the accuracy rate depending on the number of evaluations and we analyse the statistical significance of the improvements.
- To compare the performance of SPMS-ALS with the state-of-the-art techniques in the family of IR, with a focus on **IG** (Subsection 4.4.4, Part A) considering the 1NN rule as a baseline and recent **IS** methods (Subsection 4.4.4, Part B).

In addition, the average runtime required by each algorithm is contrasted to highlight the substantial computational saving in the proposed method.

- To establish a fair comparison between the proposed approach and the state-of-the-art algorithm in the family of **IR** with hybrid **IS** and **IG** algorithm, SSMA-SFLSDE, using the same memetic **IS** algorithm as initialisation mechanism (Subsection 4.4.5).
- To contextualise the results presented in this chapter by comparing the performance of SPMS-ALS with a recently proposed classifier (obRaF(H)) which represents a robust algorithm in the field of classification (Katuwal, Suganthan, and Zhang, 2020) (Subsection 4.4.6).

For the sake of space, this section will only present summary results, and all the detailed results can be found in the Supplementary Material and the associated GitHub repository[1].

4.4.1 LSIR Running with Different Computational Budgets

The Local Search LSIR as shown in Algorithm 4 has been run with the three budget settings outlined in Table 4.2 to understand the influence of the budget allowance in the performance of this algorithm. Its average classification performance in the training and test phase is displayed in Table 4.4 on small and medium datasets. Note that the reported performance is obtained from using Algorithm 1 changing **TR** by **TS** to evaluate LSIR in the test phase. Analysing these average results and the detailed results in the supplementary material, we can make the following comments:

- In the training phase the computational budget has a major impact on the performance. However, while the performance grows consistently from Setting 1 to Setting 3, this improvement is not major when the performance of Setting 2 and Setting 3 are compared. This may infer that changing each feature value cannot help the search seek a better solution after a certain number of function calls.
- Regarding the test performance, we observe that the results are dramatically different to those achieved during the training phase: Setting 1 achieves most of the wins in test data overall. In small datasets, Setting 1 has 22 wins out of 40, while Setting 2 and 3 win 14 and 11 times, respectively. In medium datasets, Setting 1 has 9 wins out of 17, while Setting 2 and 3 win 6 and 8 times, respectively. We conclude that overfitting is likely to happen for LSIR (possibly due to its exploitative structure) in Setting 2 and 3. This tendency appears evident in small size datasets.

[1] https://github.com/lehoanglam20000/SPMS-ALS

TABLE 4.4: Average training and test accuracy performance in different settings of LSIR over small and medium datasets.

	Training		Test	
	Small	Medium	Small	Medium
Setting 1	0.8521 ± 0.0128	0.9005 ± 0.0039	0.7411 ± 0.0605	0.8612 ± 0.0139
Setting 2	0.8657 ± 0.0128	0.9049 ± 0.0039	0.7419 ± 0.0614	0.8610 ± 0.0133
Setting 3	0.8693 ± 0.014	0.9052 ± 0.0041	0.7415 ± 0.0607	0.8609 ± 0.0136

In summary, we can conclude that this local search does not seem to benefit from using a larger budget and, as possibly expected, may be falling into local optima, which do not generalise well in terms of classification performance. This is especially noticeable in medium size datasets in which the average training performance does not seem to increase much in respect to the number of evaluations.

To further illustrate the behaviour of the LSIR approach with respect to the number of evaluations, Table 4.5 shows the effect of its stopping criteria. More specifically, when LSIR does not succeed at enhancing upon the trial solution $\mathbf{x}^{\mathbf{t}}$, the exploratory step decreases by the factor ρ_{Red} until a threshold value ρ_{Thr} is met (see Algorithm 4). When these conditions are met the run of LSIR is interrupted. Table 4.5 displays the computational budget saving caused by the interruption of the run. The savings are shown for small and medium datasets and for each of the setting under consideration. For each configuration of dataset and setting the number of function calls used by the algorithm is also shown.

TABLE 4.5: Number of evaluations used and saved by LSIR in different settings and datasets.

	Small datasets			Medium datasets		
	Used	Saved	(%) Saved	Used	Saved	(%) Saved
Setting 1	15398	117	0.76	290777	0	0.00
Setting 2	30795	562	1.83	581554	54094	9.30
Setting 3	61591	2966	4.82	1163108	588637	50.61

Table 4.5 shows that Setting 1 mostly uses up the allocated number of evaluations, whilst Setting 2 saves 1.83% and 9.3% in small and medium datasets, respectively. Setting 3 spends most of the evaluations in small datasets but only consumes nearly half of the allocated number of evaluations. These figures may help optimise the number of evaluations used for each dataset based on their size and features. On the other hand, the allocation of a very large budget to the local search (like Setting 3) may not be always beneficial, and as mentioned above, the algorithm seems to get trapped into local optima.

4.4.2 Runtime Reduced in the Accelerated Version of LSIR

This subsection reports the runtime used by LSIR and how much it is reduced from its accelerated version using Algorithm 4 and the fitness in Algorithm 5, here referred to as Accelerated Local Search for **IR** (ALSIR).

TABLE 4.6: Average runtime (in seconds) saved in different settings of LSIR and ALSIR, smaller values are in bold.

	Small datasets			Medium datasets		
	LSIR	ALSIR	(%) Time saved	LSIR	ALSIR	(%) Time saved
Setting 1	6.98	**2.35**	66.25	8676.67	**856.05**	90.13
Setting 2	19.47	**3.60**	81.54	15957.94	**1508.85**	90.54
Setting 3	37.68	**5.92**	84.29	18061.49	**1784.64**	90.12

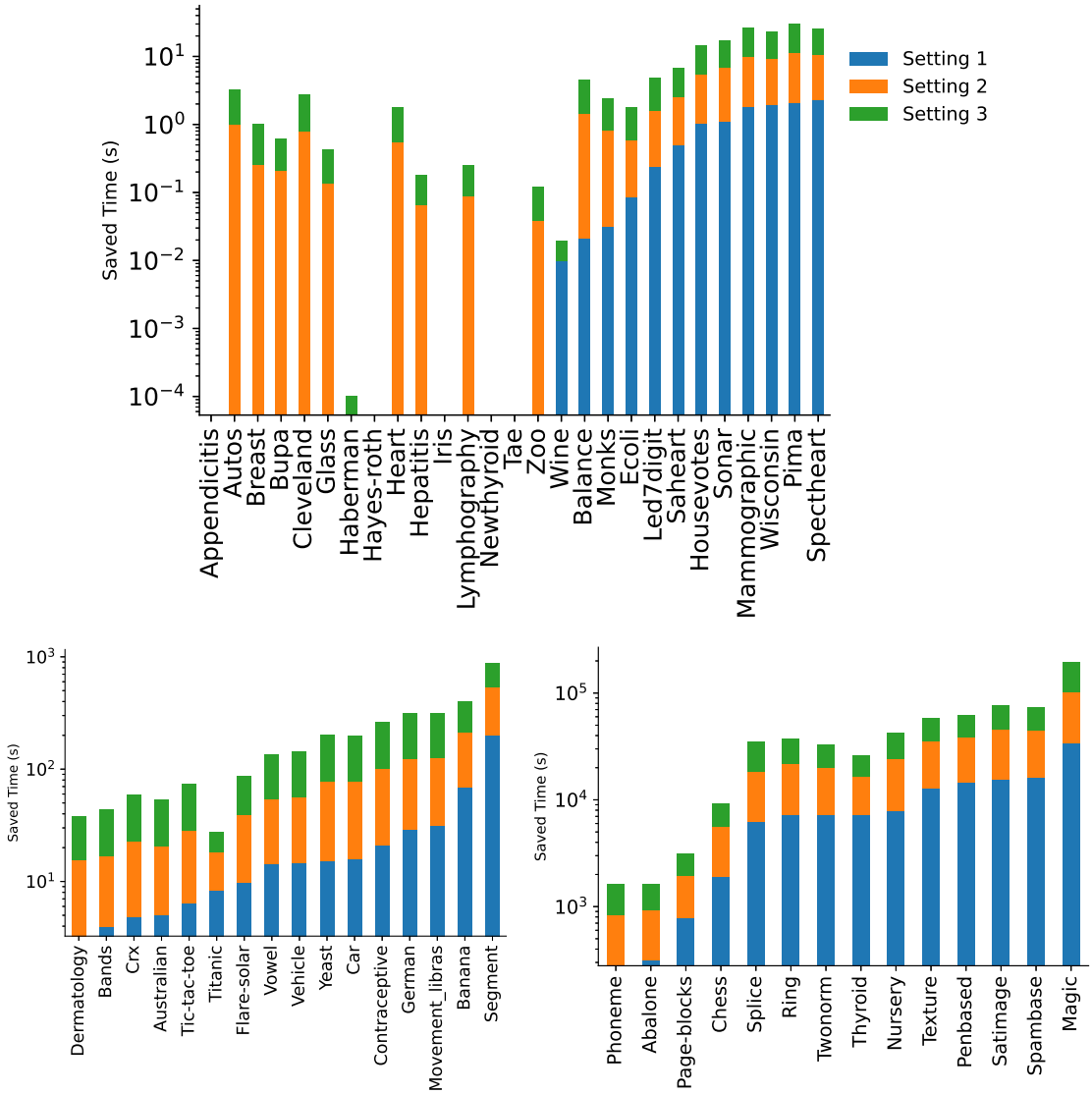


FIGURE 4.1: Runtime saved across 57 datasets, sorted by the ascending order of time gaps in Setting 1.

Of course, the time required by the local search depends directly on the allocated budget and when the stopping criteria is reached. It is also important to remember that ALSIR always provides exactly the same classification performance as LSIR, this is because ALSIR focuses on accelerating the execution of the proposed method but it does not change the behaviour of the algorithm at all. The objective of the section is therefore to show how much we can accelerate LSIR with the proposed acceleration strategy.

Details of the runtime of both LSIR vs ALSIR in small and medium datasets, respectively, with respect to each setting of the number of evaluations can be found in the Supplementary material. Table 4.6 summarises the average runtime for each setting and the average percentage of time saved by the proposed acceleration. On average in small datasets, the objective function of the accelerated local search as in Algorithm 5 enables a time reduction from at least 66% to 84.29%. However, the average runtime saved in medium datasets settles around 90% in the three settings. Thus, the larger the dataset the more we can benefit from the proposed acceleration strategy, as distance computations become the most dominant part of the execution of the local search.

To illustrate the runtime reduction depending on the dataset size, Figure 4.1 depicts the difference in runtime between LSIR and ALSIR for all the datasets, providing a graphical representation of the average time saving for each dataset. In order to enhance the readability of the diagram, the logarithmic scale has been used. Those datasets which appear to have no value represent those scenarios where the search can be completed is less than a second. Hence, the acceleration may not be essential in these cases.

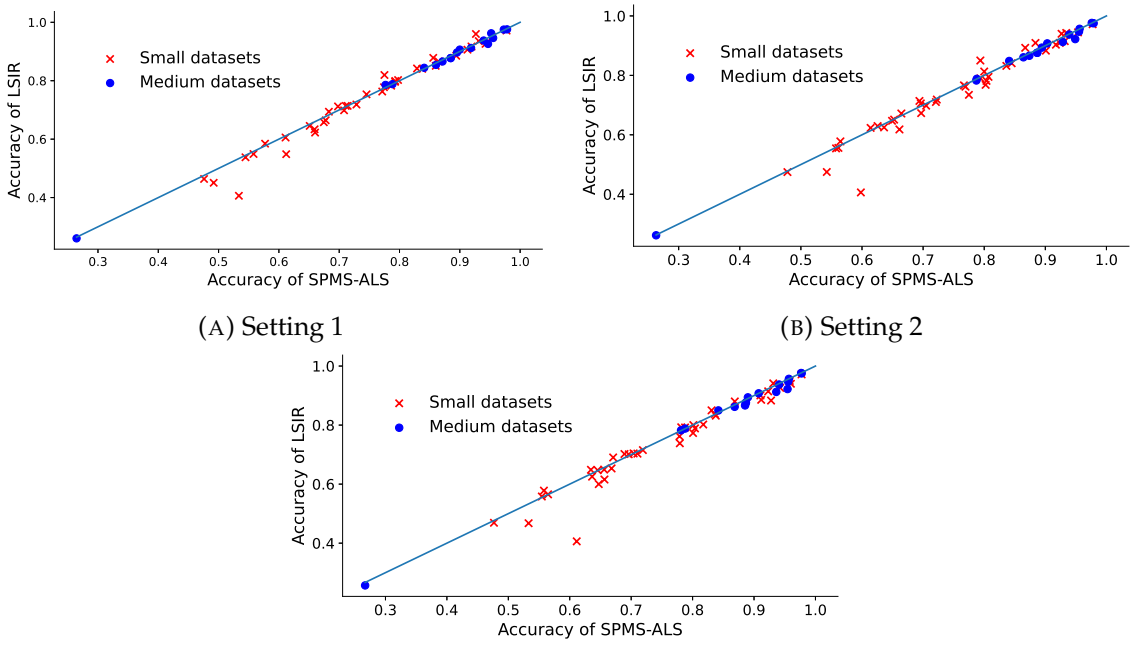
4.4.3 Validation of the Memetic Framework of SPMS-ALS

In this section, we compare the performance of LSIR and SPMS-ALS to demonstrate the effectiveness of the proposed memetic framework. Table 4.7 provides a full summary of this comparison, presenting the average accuracy values (over all the datasets) and the corresponding standard deviations in the three settings of computational budget in both training and test phases. The best average results in training and test are highlighted in bold face.

Furthermore, the Wilcoxon test (Wilcoxon, 1945) is also applied to detect the statistical differences between the two methods. The corresponding p-values are also shown in the last column of Table 4.7. When one algorithm significantly outperforms the other, the p-value is less than the confidence level 0.05. We highlight in italic these p-values.

TABLE 4.7: Comparison in average training and test performance between LSIR and SPMS-ALS over small and medium datasets. Wilcoxon p-value is obtained from the comparison between SPMS-ALS and LSIR.

		SPMS-ALS				LSIR				Wilcoxon
		TRAINING		TEST		TRAINING		TEST		p-value
SMALL	Evaluations	Acc	Std	Acc	Std	Acc	Std	Acc	Std	
	Setting 1	0.8598	0.0130	0.7477	0.0633	0.8521	0.0128	0.7411	0.0605	0.1015
	Setting 2	0.8665	0.0122	0.7512	0.0625	0.8657	0.0128	**0.7419**	0.0614	0.0867
	Setting 3	**0.8733**	0.0110	**0.7549**	0.0615	**0.8693**	0.0140	0.7415	0.0607	*0.0132*
MEDIUM	Setting 1	0.9126	0.0034	0.8625	0.0127	0.9005	0.0039	**0.8612**	0.0139	>0.2
	Setting 2	0.9129	0.0033	0.8626	0.0126	0.9049	0.0039	0.8610	0.0133	>0.2
	Setting 3	**0.9199**	0.0028	**0.8668**	0.0110	**0.9052**	0.0041	0.8609	0.0136	*0.0577*



(A) Setting 1

(B) Setting 2

(C) Setting 3

FIGURE 4.2: Accuracy scatter plots over 40 small and 17 medium datasets in the test phase.

Numerical results in Table 4.7 show that for both training and test phases, the memetic framework outperforms on a regular basis LSIR. We may observe that in training phase and small datasets, SPMS-ALS slightly outperforms LSIR while the difference in performance is larger for medium datasets. According to our interpretation, this shows the effectiveness of the global search component in complex spaces: while the local search exploits the space and is likely to achieve a suboptimal point (we may see that different computational budgets do not yield major changes in LSIR performance), the crossover allows the search a further chance to detect a solution closer to the global optimum. The results in the test phase display a consistent better performance of the memetic framework across the datasets. This finding can be

interpreted as a better performance of SPMS-ALS in terms of overfitting: the deterministic and exploitative nature of LSIR may lead to overfitting while the degree of randomisation introduced by the crossover-based global search element reduces the risk of overfitting hence improving upon the performance of the algorithm in test phase. Finally we may observe that SPMS-ALS statistically outperforms LSIR in Setting 3 in small datasets and shows improved progress in medium datasets. This fact is expected since longer runs tend to be more stable and thus be associated with lower standard deviation values. On the contrary, with Setting 1 and 2 we are more likely to observe "lucky" or "unlucky" runs that may jeopardise the statistical significance of the results.

The test results are also graphically presented in Figure 4.2 which contains scatter plots of the accuracy of the methods. Each point compares the test performance of SPMS-ALS and LSIR algorithm on a single dataset. The accuracy of SPMS-ALS is shown on the x-axis position of the point, while that of LSIR is on the y-axis position. Thus, points below the $y = x$ line correspond to datasets for which SPMS-ALS achieves better performance than the compared algorithm. In most of the cases, the points are plotted on or below the separating line, inferring greater performance of SPMS-ALS. In this plot, we can also see that the biggest improvements have been made in small datasets, but in turn, there are a few datasets in which SPMS-ALS performs slightly worse. However, in medium size datasets the improvements are less significant, especially in settings 1 and 2, but consistently better.

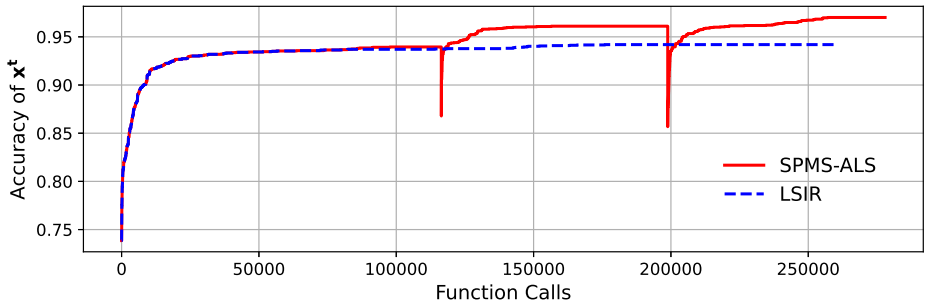


FIGURE 4.3: Accuracy progress of SPMS-ALS and LSIR on the Chess dataset.

In order to emphasise the different behaviour of LSIR vs SPMS-ALS we plot in Figure 4.3 the accuracy of the trial solution $\mathbf{x}^t$ against function calls (evaluations) of the two algorithms on the Chess dataset, using Setting 1. To show the functioning of the crossover the plot of SPMS-ALS refers to the local solution (and not the elite). We may observe that the crossover functions as a restart which then quickly reaches a solution with a good performance.

In conclusion, whilst the local search seemed to get stuck after a number of evaluations, the proposed MC approach, despite its simplicity, benefits from larger computation budgets, outperforming the local search.

4.4.4 Comparison with the State-of-the-art Methods in IG

This section consists of two parts: Part A covers the comparison of our proposal with related **IG** methods; Part B presents the comparison with recently published **IS** methods.

Part A: Comparison against Similar IG Techniques

In order to compare the performance of SPMS-ALS against that of the other global optimisers for **IG** (PSO (Nanni and Lumini, 2009), SFLSDE (Triguero, García, and Herrera, 2011), and the adapted LSHADE (Tanabe and Fukunaga, 2014)), we will focus on the maximum number of evaluations (Setting 3) for all the algorithms. As a baseline, we also include the 1NN algorithm as a comparison algorithm.

Table 4.8 summarises the performance of the comparison algorithm in all (57) datasets (small and medium). We have employed the Friedman procedure (García et al., 2010) plus a Holm post-hoc test to perform a ranking-based statistical analysis on the performance of the algorithms for small and medium datasets, respectively. The last two columns of Table 4.8 provide the results of these tests, including the rankings and the resulting p-values. Note that the control method will obtain the lowest ranking, and therefore, the p-value shows if the differences are significant comparing the control algorithm against the rest of the methods.

As shown in Table 4.8, LSHADE and SFLSDE are reported as the control method in small and medium datasets, respectively, since they hold the smallest ranking values. In small datasets, our proposal SPMS-ALS ranks third after SFLSDE, while it ranks second in the medium datasets and its ranking value is not far away from that of the control algorithm.

The Holm post-hoc test is used to detect if there is any significant statistical differences between the control algorithm (LSHADE and SFLSDE) with respect to the remaining methods. Considering a level of significance of $\alpha = 0.05$, LSHADE statistically outperforms only 1NN in small datasets, and PSO in medium datasets. The statistical tests have not reported significant differences between our proposal and the control algorithm in either small or medium datasets.

TABLE 4.8: Summary of the performance of SPMS-ALS against SFLSDE, PSO, LSHADE and 1NN for **IR** over 57 datasets. The best performance in the column is shown in bold.

		TRAINING		TEST		Friedman+Holm	
	Algorithm	Acc	Std	Acc	Std	Ranking	p_{Holm}
SMALL	LSHADE	0.8401	0.0165	0.7541	0.0612	**2.425**	–
	SFLSDE	0.8480	0.0092	**0.7615**	0.0634	2.525	0.7773
	SPMS-ALS	**0.8733**	0.0110	0.7549	0.0615	3.125	0.1017
	PSO	0.8147	0.0156	0.7414	0.0606	3.175	0.1017
	1NN	0.7369	0.0088	0.7369	0.0088	3.750	**0.0007**
MEDIUM	SFLSDE	0.8887	0.0048	0.8608	0.0122	**2.177**	–
	SPMS-ALS	**0.9199**	0.0028	**0.8668**	0.0110	2.353	0.7448
	LSHADE	0.8859	0.0138	0.8503	0.0147	3.294	0.1180
	1NN	0.8316	0.0045	0.8316	0.0045	3.294	0.1180
	PSO	0.8537	0.0066	0.8319	0.0137	3.882	**0.0066**

According to our interpretation, in training phase an exploitative action guarantees a better performance of the algorithm especially in high dimensions (Caraffini, Neri, and Iacca, 2017). However, the exploitative pressure should be counterbalanced by a certain degree of randomisation to prevent the algorithm from overfitting and pay off with a deteriorated performance in test phase. This feature of the **IG** problem makes it especially suitable to be tackled by memetic frameworks. Albeit reasonable, the excessively exploratory nature of PSO does not appear to effectively address the large dimensional space.

In comparison with the baseline, 1NN, which uses all the data to classify the test set, we have conducted the Wilcoxon test to conduct a pairwise comparison to our method. Although SPMS-ALS shows better average performance , the Wilcoxon test compute p-value 0.0415 for small datasets and > 0.2 for medium datasets. These numeric p-values indicates that our algorithm statistically outperform 1NN in small dataset, but has no significant different in medium datasets, considering a level of significance of $\alpha = 0.05$.

Finally, Table 4.9 displays the average runtime of the four global optimisers for the small and medium datasets, respectively. On average, the runtime spent for small datasets is 6.25s, saving 92.52%, 83.93% and 28.17% with respect to LSHADE, SFLSDE and PSO, respectively. For medium datasets, the percentage of saving time is slightly higher than what it did in small datasets, but the absolute value is more meaningful. Specifically, SPMS-ALS only consumes roughly 5000s on average, while SFLSDE and PSO experience about 35000s, and LSHADE used up to approximate 160.000s. In other words, for medium datasets, SPMS-ALS achieves similar if not better results of SFLSDE and LSHADE in one seventh and less than one thirtieth of the runtime, respectively.

TABLE 4.9: Comparison of the runtime (in seconds) consumed in SPMS-ALS and other approaches. Min values are in bold.

	Small datasets	Medium datasets
SFLSDE	38.89 ± 1.31	34738.82 ± 188.55
PSO	19.63 ± 0.80	34941.65 ± 188.86
LSHADE	83.63 ± 4.12	159009.10 ± 1154.75
SPMS-ALS	**6.25** ± 0.39	**5006.93** ± 405.33

Part B: Comparison against Recent IS

As mentioned in Section 4.3.2, two recent **IS** algorithms LSBo (Leyva, González, and Pérez, 2015) and RIS1 (Cavalcanti and Soares, 2020) have been selected for comparison with our proposal. This section reports the experimental results of the these two algorithms against SPMS-ALS over 57 datasets with reference to test performance and reduction rate.

Details of the classification performance of RIS1, LSBo and SPMS-ALS in the test phase on small and medium datasets can be found in the Supplementary Material, while the summary information is displayed at Table 4.10. Overall, RIS1 obtains a majority of wins in both small and medium datasets, SPMS-ALS ranks second and LSBo lies at the lowest position. Particularly, RIS1 has 25 wins (18 small and 7 medium), SPMS-ALS achieves the best results 18 times (15 small and 3 medium), while LSBo obtains 15 wins (8 small and 7 medium). However, the average test performance of LSBo and SPMS-ALS are the highest for small and medium datasets, respectively.

The overall goal of **IR** is to reduce the original dataset as much as possible whilst keeping (or improving) the accuracy. Therefore, to establish a fairer comparison between the two **IS** methods and our method, we will also provide an additional metric to consider both test accuracy and reduction as equally important. Following (Triguero, García, and Herrera, 2011), we simply multiply the accuracy in test **Acc** and the reduction rate **Red** to form a new metric **Acc*Rec**. Table 4.10 presents the overall average performance in accuracy **Acc**, reduction rate **Red** and both metrics **Acc*Rec**. Furthermore, Table 4.11 provides the set of rankings and p-values obtained from Friedman+Holm tests for the three contrasted algorithms.

In the previous section, all algorithms (PSO, SFLSDE, LSHADE, and SPMS-ALS) yield the same reduction rate. In particular, in this experiment, SPMS-ALS fixes the rate up to 95% and 98% for small and mediums datasets, respectively. However, LSBo and RIS1 do not specify the reduction rate as a parameter, but their reduction depends on a particular dataset. RIS1 yields an average reduction rate of 59.06% and 70.52% in small and medium datasets, respectively; whilst LSBo reduces 78.59%

and 87.39% in small and medium datasets. Thus, both algorithms achieve a smaller reduction rate than the **IG** approach investigated in this chapter.

TABLE 4.10: Summary of the performance of SPMS-ALS against RIS1 and LSBo considering **Acc** in the test phase, **Red** and **Acc*Red** measures for **IR** over 57 datasets. The best performance in the column is shown in bold.

	Algorithm	Acc (Test)	Std	Wins	Red	Acc*Red
SMALL	RIS1	0.7319	0.0583	18	0.5906	0.4499
	LSBo	**0.7605**	0.0576	8	0.7859	0.6064
	SPMS-ALS	0.7549	0.0615	15	**0.9500**	**0.7172**
MEDIUM	RIS1	0.7972	0.0141	7	0.7052	0.6071
	LSBo	0.8540	0.0099	7	0.8739	0.7639
	SPMS-ALS	**0.8668**	0.0110	3	**0.9800**	**0.8495**

On average, the test performance of LSBo is the highest on small datasets, while SPMS-ALS reports the highest average on medium datasets. However, apart from having the best reduction rate, SPMS-ALS obtains the best balance between accuracy and reduction. The results from the non-parametric tests (Friedman+Holm) in Table 4.11 reveal the advantage of SPMS-ALS looking at the Ranking and p-value columns. The p-values reported for the comparison in terms of accuracy column do not reflect any significant differences between the three methods in either small or medium datasets. However, when the reduction rate is taken into consideration the proposed technique stands out significantly.

TABLE 4.11: Friedman+Holm statistical test results in both **Acc** and **Acc*Rec** metrics for small and medium datasets. The best performance in the column is shown in bold.

	Acc			Acc * Red		
	Algorithm	Ranking	p-value	Algorithm	Ranking	p-value
SMALL	SPMS-ALS	**1.912**	–	SPMS-ALS	**1.175**	–
	RIS1	1.938	0.9110	LSBo	2.000	**0**
	LSBo	2.150	0.2882	RIS1	2.825	**2.25E-04**
MEDIUM	LSBo	**1.882**	–	SPMS-ALS	**1.176**	–
	RIS1	2.000	0.7316	LSBo	2.059	**1.01E-02**
	SPMS-ALS	2.117	0.4927	RIS1	2.765	**4.00E-06**

4.4.5 Hybridisation with Instance Selection

As stated in Section 4.3.2, to perform a fair comparison against hybrid **IR** method SSMA-SFLSDE (Triguero, García, and Herrera, 2011), the initialisation process of the proposed algorithm must be replaced with a smarter approach. In particular, we use the same **IS** algorithm, SSMA (García, Cano, and Herrera, 2008), as tested

in (Triguero, García, and Herrera, 2011). This section compares LSIR, SFLSDE, LSHADE and SPMS-ALS after using SSMA as initialisation. The resulting algorithms are indicated as SSMA-LSIR, SSMA-SPMS-ALS, SSMA-LSHADE and the state-of-the-art algorithm SSMA-SFLSDE (Triguero, García, and Herrera, 2011). The detailed accuracy results for small and medium datasets in both training and test can be found in the Supplementary Material. Table 4.12 shows the average results obtained from the compared algorithms in conjunction with SSMA and the ranking plus p-values from the Friedman + Holm test.

TABLE 4.12: Summary performance between four hybrid models over 57 datasets.

		TRAINING		TEST		Friedman + Holm	
	Algorithm	Acc	Std	Acc	Std	Ranking	p_{Holm}
SMALL	SSMA-LSHADE	0.8687	0.0101	**0.7792**	0.0570	**2.000**	–
	SSMA-SFLSDE	0.8684	0.0108	0.7767	0.0594	2.200	0.4884
	SSMA-SPMS-ALS	0.8727	0.0134	0.7670	0.0574	2.700	0.0306
	SSMA-LSIR	**0.8911**	0.0148	0.7642	0.0604	3.100	0.0004
MEDIUM	SSMA-SPMS-ALS	**0.9264**	0.0033	0.8700	0.0107	**2.265**	–
	SSMA-SFLSDE	0.9059	0.0040	0.8675	0.0125	2.441	1.0000
	SSMA-LSHADE	0.9069	0.0035	**0.8706**	0.0118	2.647	1.0000
	SSMA-LSIR	0.9245	0.0039	0.8682	0.0127	2.647	1.0000

The detailed results show that for small datasets and in training phase SSMA-SPMS-ALS outperforms SSMA-SFLSDE and SSMA-LSHADE, and is outperformed by SSMA-LSIR. However these results are not confirmed in test phase where SSMA-LSHADE achieves the best performance, SSMA-SPMS-ALS the third best performance after SSMA-SFLSDE, and SSMA-LSIR the worst performance over the four algorithms considered in this section. This ranking is statistically significant and confirmed by the Friedman + Holm test. According to our interpretation, the deterministic and exploitative local search logic in LSIR causes overfitting. The restriction of the search space caused by SSMA increases the risk of overfitting. In the proposed memetic framework, the resampling and crossover mechanism seems to mitigate the overfitting.

Our interpretation is confirmed by the results for medium datasets. Since the search space is naturally large, the exploitative local search is beneficial (Caraffini, Neri, and Iacca, 2017) and is improved by the memetic framework. Hence, SSMA-SPMS-ALS achieves the best performance in training phase. This ranking is confirmed in test phase where SSMA-SPMS-ALS slightly outperforms SSMA-SFLSDE and is established as the control algorithm in the Friedman test.

In summary, and similar to what we saw when comparing against purely **IG** methods, the proposed memetic framework can obtain a very competitive classification performance, especially in larger datasets, whilst reducing drastically the required runtime.

At last we report some considerations about future improvements that can be applied. We will investigate the extension of our approach to big data frameworks (Triguero et al., 2015b). In addition, we plan to expand our approach to new promising classifiers, such as Heterogeneous oblique random forest (Katuwal, Suganthan, and Zhang, 2020). An initial comparison with this kind of classifier can be found in the Supplementary Material. Further investigation is however required to perform an appropriate **IR** for those classifiers.

4.4.6 Contextualising the Results and Limitations of SPMS-ALS

Experimental results from Sections 4.4.1 to 4.4.5 show that the proposed SPMS-ALS and its hybrid form, SSMA-SPMS-ALS, are effective at reducing the size of the training data whilst maintaining, or even improving, the performance of the base classifier; in our case, the 1NN rule. The goal of this section is to contextualise the classification results presented in this chapter with the 1NN as the base classifier, and let the reader know where we are going in our future research. To do so, we compare the results of the proposed SPMS-ALS and 1NN against the popular Random Forest (RaF) algorithm and a state-of-the-art classifier also based on Trees, obRaF(H) (Katuwal, Suganthan, and Zhang, 2020).

It is important to note that the classification performance of a classifier is not only influenced by the pre-processing techniques but also its inherent robustness. For example, an ensemble classifier is likely to outperform a single model (Rokach, 2010), or tree-based approaches handle categorical attributes better than the NN classifier. Thus, whilst the reader may expect the results of RaF and obRaF(H) to be superior to the ones presented with the 1NN rule, we believe it is beneficial to still observe the performance gap and understand potential future research lines for preprocessing technique for more robust classifiers.

To establish a fair comparison against methods, we have re-run SPMS-ALS over the 121 UCI datasets used in (Fernández-Delgado et al., 2014; Katuwal, Suganthan, and Zhang, 2020), following the exact same experimental framework, including depth of a tree (i.e. 57), number of trees (i.e. 500), and a 4-fold cross validation scheme. Details of the comparison on each single dataset can be found in the Supplementary Material, while the summary information and results of the statistical test are displayed in Table 4.13. As expected, both RaF and obRaF(H) display a higher performance than NN-based results. On the other hand, the proposed data reduction

does not only reduce the storage need but also becomes much more efficient in terms of runtime as we only preserve 2 to 5% of the training data (i.e. followed the experimental setup in (Triguero et al., 2012). For this reason, it is essential to highlight that the contribution of our proposal lies in the reduction of the training set, as a preprocessing technique, whilst maintaining (or improving) the classification performance of 1NN.

TABLE 4.13: Summary the average accuracy performance of 4-fold cross validation between the basic models (1NN and RaF) and their improved versions (SPMS-ALS and obRaF(H)) over 121 datasets.

			Friedman + Holm	
	TRAINING	TEST	Ranking	p_{Holm}
obRaF(H)	–	**0.8336**	146.24	–
RaF(Scikit-learn)	0.9892	0.8286	173.13	0.05
SPMS-ALS	0.8926	0.7597	324.99	0.03
1NN	0.7487	0.7534	325.64	0.02

The reader might wonder if the result of the preprocessing performed by the technique proposed in this chapter could be used directly by any other classifier like RaF or obRaF(B). Although this pre-processing approach is intended for 1NN, as highlighted in (Cano, Herrera, and Lozano, 2003) the resulting set could potentially be used by any other classifiers. However, it is not straightforward to directly use the reduced set obtained from SPMS-ALS in another classifier. We have performed some preliminary experiments using the resulting reduced set as training data for RaF in 89 small datasets (from the set of 121). The average results in the test phase sets at 0.5656, whilst it is 1.000 on training. This suggests that RaF is overfitting the training data with the parameters used (e.g. depth of the trees, or number of trees). This could be expected as the reduction on small datasets may end up having as few as 2-15 samples in some extremely small datasets. Whilst NN technique would work well with such amount of data, Tree-like technique will not. Thus, as future work we plan to explore the interaction between the proposed SPMS-ALS and more robust classifiers like obRaF(B), for example, by adding it as base classifier or fine tuning the parameters to use smaller training datasets without overfitting.

4.5 Summary

In this chapter, we have achieved research Objective 2 by designing a simple and yet effective single-point memetic pattern search for IG, which is competitive to the other complex designs of state-of-the-art methods. The proposed search framework is composed of a novel accelerated local search and a crossover based global search.

The local search is deterministic and exploitative belonging to the family of Pattern Search methods whilst the global search is stochastic, based on resampling and crossover. By making some considerations about the functioning of the NN classifier in instance generation and exploiting the search logic of Pattern Search, the local search has been redesigned and implemented in an accelerated version. The accelerated local search uses most of the calculations performed at the previous step and thus lead to a major saving in terms of runtime with respect to the existing algorithms in the literature.

Numerical results performed with and without **IS** as initialisation mechanism show that the proposed single-point memetic approach tends to be slightly worse than only one **IR** algorithm in small datasets. On the other hand, on medium datasets, our proposal achieves the best accuracy performance in both training and test phases. These results are extremely valuable when we consider that the proposed approach is up to seven times faster than the other algorithms. Besides the proposed domain-specific MC approach this article offers an extra contribution about experimentalism in data reduction. More specifically, in this chapter we perform a thorough parameter setting of the computational budget and display the results in multiple scenarios. These results aim to offer some guidelines to data scientists to set their experimental conditions in a fair and effective manner to detect the desired trade-off between accuracy and runtime.

In the first two stages of our research, we have concentrated on two different tasks of **IS** and IG, constructing different innovative search solutions on combinatorial and continuous spaces. Since the available information of integrating **IS** and **IG** into a single search framework is very limited or non-existed (to the best of our knowledge), we are motivated to propose a novel search framework to handle **IS** and **IG** simultaneously. In the following chapter, we will introduce this new search strategy which aims to cover Objective 3.

Chapter 5

Accelerated Pattern Search for Simultaneous IS and IG

5.1 Introduction

In previous chapters, we have dealt with different search strategies at which each has a different accelerated mechanism to speed up the optimisation process. Concretely, Chapter 3 has introduced a two-stage clustering method in a population-based meta-heuristic search strategy, then Chapter 4 has presented Single-Point Memetic Structure with an accelerated objective function evaluation. The contribution made in Chapter 3 provides a solution for IS, while the counterpart made in Chapter 4 is designed for IG. Considering both reduction rate and accuracy, state-of-the-art performance in **IR** techniques is achieved by hybrid approaches in which **IS** and **IG** combine their effectiveness to produce a final reduced dataset (Le, Neri, and Triguero, 2021). In practice, all hybrid techniques proposed so far in the literature, make use of cascade approaches which perform **IS** and **IG** in separated and subsequent stages. More specifically, **IS** reduces the size of the dataset while **IG** refines the results on the reduced data. This is because **IS** is usually employed to decide the best distribution of instances per class, providing a good starting point to let **IG** optimise the positions of the instances. Optimisation-based **IG** approaches search for a solution in the continuous space (generate instances) while keeping the number of instances fixed, i.e., deciding at first then the desired number of instances and then searching for suitable instances within the continuous space defined (Triguero et al., 2012). In contrast, a method that determines the most appropriate number of instances while performing **IG** has not been explored yet.

Popular hybrid approaches combine a Steady State Memetic Algorithm (*SSMA*) (García, Cano, and Herrera, 2008) plus an **IG** method (Triguero, García, and Herrera, 2011; Derrac, García, and Herrera, 2012; Le, Neri, and Triguero, 2021). SSMA is a population-based memetic search approach developing on the evolutionary algorithm including an *LS* component. The process of sample selection is explored

globally by a variety of solution candidates in the population and is also exploited locally by the *LS* association. As a result, the search seeks the reduced set at different regions in the training instance space. After optimising the number of samples, hybrid approaches feed the subset of well-distributed samples to an **IG** algorithm to refine their positions. The subset of samples are further optimised to be capable of filling regions that do not have any representative in the training instance space. The best performing **IG** techniques reported in the literature usually belong to the family of population-based evolutionary algorithms (Le, Neri, and Triguero, 2021).

Despite its effectiveness, this type of hybrid methods is typically time-consuming due to the expensive cost of computing the objective function, not only caused by SSMA but also by the **IG** phase (if tackled by evolutionary methods). A recent study has introduced a simple, yet effective memetic approach which is able to yield major savings in computational overhead during the **IG** phase (Le, Neri, and Triguero, 2021). However, this memetic approach still relies on SSMA feeding in a subset of compact and well-rep resentative samples among classes. Thus, the computational cost of the overall **IR** process is still high to handle medium sized datasets, taking hours or even days to complete it (See Figure 5.4).

From the mentioned advantages and disadvantages, there is the necessity for an approach which can achieve the same high accuracy performance of hybrid methods while addressing their drawback of excessive runtime. This technique should be capable of searching globally and exploiting locally the instance space like SSMA, but also simultaneously adjusts the samples at the feature level. Besides from obtaining a performance as good as that of hybrid methods, more importantly, the proposed technique has to be fast to be applicable in solving real-world problems. To the best of our knowledge, ideas of using simultaneously **IS** and **IG** in a single approach has never been explored.

In this chapter, we propose an **IR** algorithm, called Accelerated Pattern Search with Variable Solution Size (APS-VSS), that performs a fast reduction of the data by simultaneously doing **IS** and IG. Our proposal stems from the SPMS, proposed in (Neri and Triguero, 2020; Le, Neri, and Triguero, 2021). SPMS algorithm is a domain-specific technique for **IG** belonging to the family of Memetic Computing, which features an accelerated objective function evaluation. Conversely, the proposed APS-VSS integrates two domain-specific *LS* components to optimise the total number of instances appearing in the reduced set within the **IG** local search of (Le, Neri, and Triguero, 2021). These two *LS* components can adjust the size of a candidate solution during the optimisation process, to shrink the size, so that the **IG** local search becomes more effective. The remainder of this chapter is as follows: Section 5.2 details the proposed APS-VSS algorithm. Section 5.3 describes the experimental setup

while Section 4.4 shows and discusses the results. Finally, Section 5.5 provides the summary of this final research stage.

5.2 Accelerated Pattern Search with Variable Solution Size

This section describes the proposed algorithm APS-VSS. Subsection 5.2.1 highlights the novelty of the proposed algorithm while, Subsection 5.2.2 presents the implementation details of APS-VSS.

5.2.1 Novelty of the proposed approach

Two main branches of **IR** may belong to different search space domains. While **IS** may be considered in the combinatorial search space, **IG** is in the continuous domain. Though there are **IR** techniques working in both search spaces but they are applied sequentially, usually **IS** first and then **IG** (e.g. SSMA-SPMS-ALS (Le, Neri, and Triguero, 2021)). Typically, **IS** searches for the best distribution of instances per class to feed in **IG** for further optimisation. The current state-of-the-art approaches are accurate but may be extremely computationally expensive (Triguero, García, and Herrera, 2011). For example, with the dataset "Magic" (19020 samples, 10 features), on average of 10 runs, SSMA-SFLSDE consumed 68.3 hours to complete (Triguero, García, and Herrera, 2011), while SSMA-LSHADE took 92.6 hours (Le, Neri, and Triguero, 2021). Broadly speaking, the novelty of our proposal lies in introducing an **IR** algorithm which handles simultaneously an optimisation problem on both continuous and combinatorial search spaces. Unlike previous studies which address **IR** in separated stages, APS-VSS performs the selection and generation within a single framework.

5.2.2 Algorithmic Description

The proposed APS-VSS partly makes use of the successful search logic of SPMS-ALS (Le, Neri, and Triguero, 2021) but completely reinterprets the **IR** problem representation and hence the search space associated with it. APS-VSS employs two novel extra *LS* components, named LS^{eli} and LS^{asc}, respectively. More specifically, a candidate solution x is considered to have variable length and its design variables to continuously vary within intervals. The variable length of the candidate solution encodes the **IS** and is optimised by LS^{eli} where *eli* stands for "elimination" of the non-representative instances. On the other hand, the values in the present design variables encode the **IG** and are optimised by feature adjustment of LSIR. Furthermore, the proposed APS-VSS attempts to remove the instance just modified by the

LSIR. If the accuracy of the candidate solution is not worsened after the removal of the instance, then the instance is removed. In other words this LS^{asc} has to "ascertain" that the perturbed instance is indeed necessary to represent the dataset. The pseudo-code of APS-VSS is presented in Algorithm 8.

Algorithm 8 Accelerated Pattern Search with Variable Solution Size

Input: Training Data (TR)
Output: An optimised RS

1: x = SelectSamples (TR) ▷ Select samples from TR, tuned by parameter P_{init}
2: **while** local budget and precision conditions are not met **do**
3: **for** $h = 1 : p$ **do**
4: $\mathbf{x}_h$ = **x** after removing the elements $b_{h1}, b_{h2}, \ldots, b_{hm}$
5: **if** $f(\mathbf{x_h}) \geq f(\mathbf{x})$ **then**
6: $x = \mathbf{x_h}$
7: **end if**
8: **end for**
9: **for** $i = 1 : n \ (n = m \cdot p)$ **do**
10: $\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e}^i$
11: **if** $f(\mathbf{x^t}) \geq f(\mathbf{x})$ **then**
12: $\mathbf{x} = \mathbf{x^t}$
13: **else**
14: $\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e}^i$
15: **if** $f(\mathbf{x^t}) \geq f(\mathbf{x})$ **then**
16: $\mathbf{x} = \mathbf{x^t}$
17: **end if**
18: **end if**
19: **if** $\mod(i, m) = 0$ **then**
20: j = i / m ▷ Get index of the generated example
21: $\mathbf{x}_t^{'}$ = $\mathbf{x}_t$ after removing the elements $b_{j1}, b_{j2}, \ldots, b_{jm}$
22: **if** $f\left(\mathbf{x}_t^{'}\right) \geq f(\mathbf{x}_t)$ **then**
23: $\mathbf{x} = \mathbf{x}_t^{'}$
24: **end if**
25: **end if**
26: **end for**
27: **if** x has **not** been updated **then**
28: halve the exploratory radius ρ
29: **if** $\rho < \epsilon$ **then**
30: Randomly generate a candidate solution $\mathbf{x^r}$
31: Apply Crossover between x and $\mathbf{x^r}$ to generate a new trial vector x^t
32: Reinitialise $\mathbf{x} = \mathbf{x^t}$
33: **end if**
34: **end if**
35: **end while**
36: **RETURN** x
37:

LS^{eli}: LS in the combinatorial space

$LSIR$: LS in the continuous space

LS^{asc}: LS in the combinatorial space

The pseudo-code of the LS^{eli} is presented in lines 3-8 of Algorithm 8. The main contribution of LS^{eli} is to discard any elements in **RS** whose absence does not affect the current solution quality. Although this simple idea of elimination relies heavily on the quality of initial sampling, the output is yet promising as the **RS** can be enhanced by the generation phase of the greedy LS (lines 9-26). Since feature perturbation, i.e., IG), is empirically more effective on a small set of samples (that is a smaller search space), LS^{eli} prepares a compact representative **RS** for the **IG** operator. In the results

presented in Section 5.4.1, a consistently steady improvement of accuracy is made during the optimisation, indicating that the shrunk **RS** is well-shaped by the LS^{eli}.

The pseudo-code of the LS^{asc} is shown in lines 19-25, which ascertain the new positioned sample is important in the current processing of RS. Different to LS^{eli}, LS^{asc} is called only occasionally as its main role is to confirm the importance of the adjusted sample. It is necessary because the feature perturbation process may adjust it into an already existing (redundant) sample in RS.

Finally, the gene-resampling mechanism (lines 27-34), taken from SPMS-ALS, is an important countermeasure to prevent the sample adjustment from overfitting. An excessive effort on sample adjustment is likely to yield an overfitted RS, which results in poor performance on unseen data. This characteristic is controlled by multiple parameters introduced in (Le, Neri, and Triguero, 2021), and experimentally analysed in Section 5.4.2.

In summary, apart from what was described for SPMS-ALS in (Le, Neri, and Triguero, 2021), APS-VSS introduces two LS components attempting to shrink the **RS** when possible to maximise the effectiveness of the single-point pattern search structure. Besides, it is important to note that it is usually expensive to check the **RS** quality when the size of **TR** is large. Our proposal does not suffer from this computational burden because it takes advantage of fast computation of the objective function, described in Section 4.2.2. One iteration of APS-VSS can be summarised by the following operators:

- LS^{eli} shrinks the initial RS, discarding any element whose absence does not deteriorate the solution quality and passes the shrunk solution to LSIR.
- LSIR perturbs features and searches for an accurate solution (one exploration of LSIR).
- LS^{asc} is embedded within LSIR and confirms whether the presence of the newly generated instance is necessary.
- The crossover re-initialises the candidate solution to explore another search region when the LS seems to be no longer effective.

These four steps are iteratively repeated until the computational budget is exhausted or a precision condition is reached.

5.3 Experimental Framework

This section presents the experimental setup of this study. Section 5.3.1 introduces the datasets used in this chapter. Section 4.3.2 presents the **IR** techniques that have

been used for comparison against APS-VSS. Finally, Section 5.3.3 describes the parameter configuration.

5.3.1 Datasets

In the experimental study, as we carry on a methodology of **IR** for balanced classification, we keep using 57 multi-class datasets with various sizes from the KEEL dataset repository (Triguero et al., 2017) presented in Section 4.3.1 and summarised in Table 4.1.

5.3.2 Comparison Algorithms

In order to understand the benefits of the proposed APS-VSS, we first define the baselines.

- Nearest Neighbour (1NN): we use the NN algorithm (k=1) employing the entire **TR** for training, without any pre-processing. This is the first baseline to verify if an **IR** solution outperforms it.
- LSIR and SPMS-ALS, refer to the previous chapter, Sections 4.2.1 and 4.2.4, respectively.

State-of-the-art approaches are hybrid techniques including SSMA-LSHADE (Tanabe and Fukunaga, 2014; Le, Neri, and Triguero, 2021), SSMA-SFLSDE (Triguero, García, and Herrera, 2011), SSMA-SPMS-ALS (Le, Neri, and Triguero, 2021). These approaches start with **IS** employing SSMA to intelligently select samples. The output **RS** is then fed into generation techniques based on differential evolution (i.e. LSHADE (Tanabe and Fukunaga, 2013), SFLSDE (Triguero, García, and Herrera, 2011)), or a single-point memetic struture (i.e. SPMS-ALS (Le, Neri, and Triguero, 2021)). To claim the effectiveness of APS-VSS, the performance needs to be better than baselines and equivalently as good as several state-of-the-art methods (but faster).

5.3.3 Parameter Settings

This section presents the parameter configuration for all the methods mentioned in Section 5.3.2. The same parameters suggested by the authors in their original works have been used. Apart from the shared parameters obtained from SPMS-ALS including ρ, $N_{\max}$, ρ_{Red}, ρ_{Thr} and Gr (Le, Neri, and Triguero, 2021), APS-VSS requires the tuning of the P_{init} parameter, which is the percentage of initialisation. As described in Section 5.2.2, LS^{eli} relies on the quality of initial sampling, P_{init} is tuned in a wide range, from 5% up to 70%. P_{init} = 5 meaning 5% of samples of a class are randomly allocated to the initial RS.

TABLE 5.1: Parameters used for comparison algorithms

Algorithms	Parameter setting
SFLSDE	PopulationSize = 40, iterSFGSS =8, iterSFHC = 20, Fl = 0.1, Fu = 0.9
LSIR	initial ρ = 0.4
SSMA	Population = 40, Cross = 0.5, Mutation=0.001
NN	k = 1, Euclidean distance
LSHADE	ArchiveSize = 1.4, PopulationSize = 40, MemorySize = 5
SPMS-ALS	ρ = 0.4, $N_{\max}$ = 3, ρ_{Red} = 0.25, ρ_{Thr} = 0.005, Gr = 0.5 (small), 0.05 (medium)
APS-VSS	P_{init} = 10, ρ = 0.4, $N_{\max}$ = 3, ρ_{Red} = 0.5, ρ_{Thr} = 0.005, Gr = 0.75 (small), 0.05 (medium)

Numerous combinations of these parameters have been examined to conclude a set of values that can produce a robust performance of APS-VSS on both small and medium datasets. Values of parameters for all algorithms are presented in Table 5.1. To make a fair comparison, all compared algorithms in this study are allocated the same computational budget, particularly $10 \times \#Samples \times \#Attributes$ evaluations, as discussed in (Le, Neri, and Triguero, 2021) .

5.4 Analysis of results

In this section, we present the analysis of the results to provide insights of APS-VSS. Our aims are:

- To understand the effectiveness of LS^{eli} and LS^{asc}. A plot displaying the search behaviour of APS-VSS on a dataset fold are presented. We also discuss the causes of improvement to explain why APS-VSS can achieve better performance with respect to the previous algorithmic designs (Subsection 5.4.1).
- To establish a fair comparison of the accuracy performance between APS-VSS with several baseline models, followed by competition with the state-of-the-art algorithms in the family of **IR** (Subsection 5.4.2).
- To present multiple advantageous aspects of APS-VSS such as reduction rate and runtime. We report in detail the absolute and percentage figures of the runtime savings, the class proportion of all examined algorithms over 57 datasets.

Numerical results of all comparisons between algorithms at each dataset can be found in the Supplementary Material at the associated GitHub repository[1].

5.4.1 Detailed Analysis of the Search Behaviour

This section presents a detailed analysis of the search behaviour and the number of function calls to the two local searches.

[1] https://github.com/lehoanglam20000/APS-VSS

Search Behaviour:

In this section, we present the search behaviour of APS-VSS in comparison to SPMS-ALS. We describe the limitations of SPMS-ALS and solutions to address them, which may be helpful to understand the reason why our proposal outperforms previous similar-structured algorithms. As an example, Figure 5.1 characterises the search behaviour of APS-VSS and SPMS-ALS on the zoo dataset.

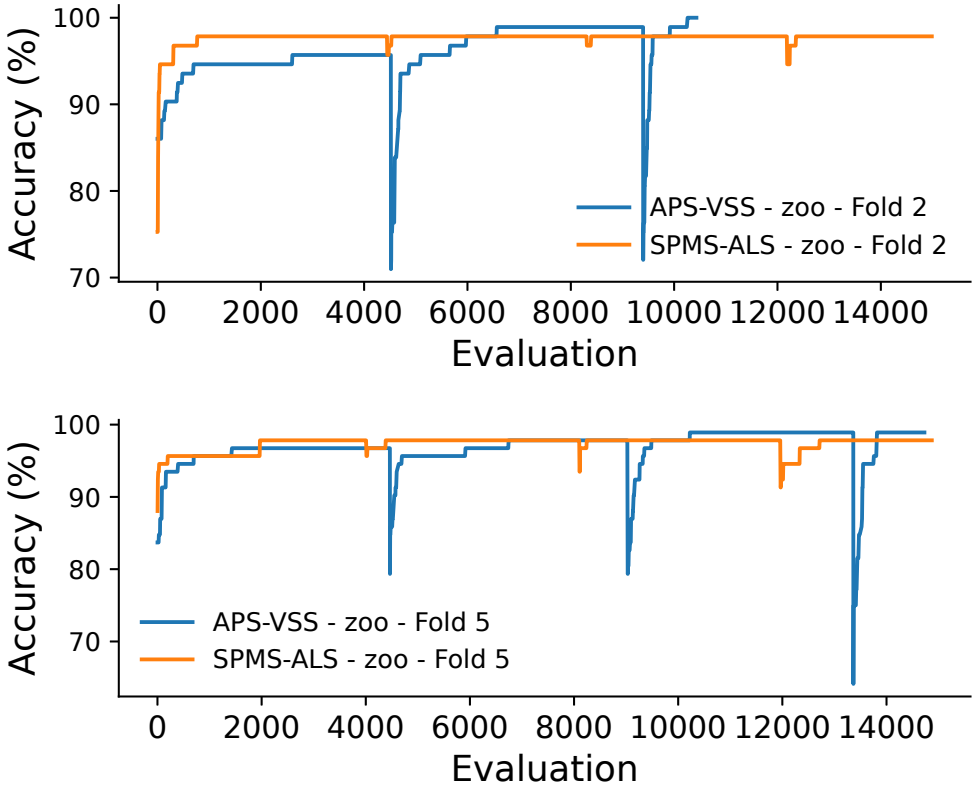


FIGURE 5.1: Search behaviour of APS-VSS and SPMS-ALS at dataset zoo, folds 2 and 5.

The search consists of several cycles, separated by a straight drop. The first cycle begins with random initialisation of RS, and end after about 4,500 evaluations. APS-VSS and SPMS-ALS do not share the same initial percentage of samples, as they start at different positions. Then, the global evolutionary operator introduces new materials to allow the search to start at a new region, initialising the start of the second cycle. Note that the search does not necessarily use up the allocated computational budget, but it might stop earlier if the stopping conditions are met (i.e. accuracy hits 100%, ρ is below the threshold ρ_{Thr}).

At Fold 2, through the feature perturbation, the accuracy value of SPMS-ALS progressively increases and remains unchanged at 97-98% until the end of the first search cycle, while APS-VSS only reaches 94-95%. Around evaluation 4500, the upward trends at both algorithms suddenly drop sharply due to the effect of the

global gene-resampling crossover operator. It depends on the parameters used for the operator to observe how far the accuracy drops down. As reported in Table 5.1, APS-VSS has $Gr = 0.75$ while SPMS-ALS has $Gr = 0.5$, which means there are more samples replaced in APS-VSS and thus resulting in a significant drop.

A new similar pattern of the search restarts from evaluation 4500 as the elements in the **RS** have been refreshed. APS-VSS gradually develops the accuracy whilst SMPS-ALS goes back to its previous peak, which can be attributed to the impact of the two new *LS* components in APS-VSS. As discussed earlier, the effectiveness of the single-point perturbation *LS* will appear more obvious in a smaller sized RS. In Fold 2, it is likely APS-VSS discards more samples to let the search progress further, while SPMS-ALS does not show this characteristic. In the end of the second search cycle, APS-VSS achieves a higher accuracy rate while SPMS-ALS remains unchanged until the end of the run. APS-VSS continues to develop a greater performance in the next cycle and stops the search at evaluation 10,500 as the stop conditions are met.

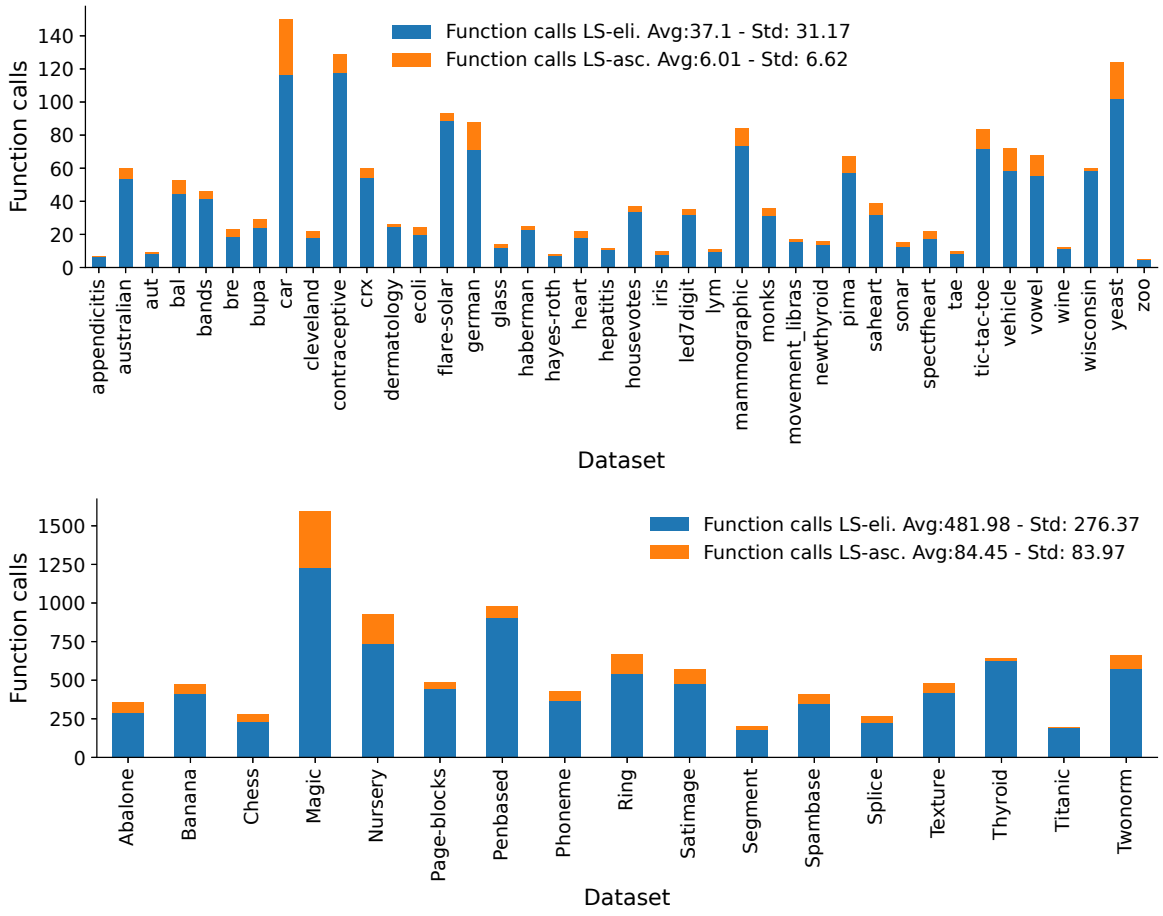


FIGURE 5.2: The average number of function calls of the two LS components over small (top) and medium (bottom) datasets.

Different from Fold 2, the two algorithms used up the computational budget in Fold

5. SPMS-ALS shows a better performance at a few first cycles, but eventually APS-VSS obtains higher accuracy. APS-VSS is likely more consistent to make progression after each restart than SPMS-ALS. As a memory slot has been allocated to store the **RS** having the highest score, exploration in a new region does not affect the final outcome when either algorithm gets lower performance at different search regions.

Average Function Calls of LS^{eli} and LS^{asc}: From the understanding of the search behaviour of APS-VSS, it may be interesting to know about the number of function calls that the two added LS are used. In general, the number of calls of LS^{eli} is more frequent than that of LS^{asc} due to two reasons. Firstly, LS^{eli} has more opportunities to be called frequently to discard unimportant samples after initialisation. Second, as $Gr = 0.75$ at a restart, LS^{eli} has to consider more samples for elimination.

On the other hand, LS^{asc} only focuses on checking whether the perturbation of all features at an instance can contribute to any improvement. As explained in the algorithm description, the adjusted instance will be only removed if its absence does not deteriorate the current performance of the RS. Since the **RS** is usually shrunk before conducting instance perturbation, samples surviving until this generation phase are likely essential for 1NN classification, so that the elimination is not expected to occur frequently. Figure 5.2 summarises the number of function calls of LS^{eli} and LS^{asc}. On average, while there are about 37 calls for LS^{eli} on small datasets, LS^{asc} gets 6, which is less than LS^{eli} about 6 times. Likewise, the ratio of LS^{eli} over LS^{asc} is approximately 6 in the medium datasets. Note that, the standard deviation is large in any numerical values, indicating the two LS components are called significantly more frequent in one dataset than in another.

5.4.2 Comparison with Other IR Algorithms

There are two comparisons in this section, Part A presents an examination of our proposal against baseline models including 1NN, LSIR and SPMS-ALS. Part B shows the comparison with the state-of-the-art algorithms including SSMA-LSHADE (Le, Neri, and Triguero, 2021), SSMA-SFLSDE (Triguero, García, and Herrera, 2011) and SSMA-SPMS-ALS (Le, Neri, and Triguero, 2021). We employ Wilcoxon test with a confidence level $\alpha = 0.05$ to analyse the significant difference among models.

Before discussing these results, it is important to note that as said earlier the goal of this work was not to improve upon the accuracy results of the state-of-the-art **IR** solutions, as they are usually more complex and time-consuming, but to verify that with a simpler and faster strategy we can remain competitive in terms of accuracy.

TABLE 5.2: Summary performance between APS-VSS and other baseline models over small and medium datasets.

		TRAINING	TEST	APS-VSS vs			p-value
		Acc $\pm$ Std	Acc $\pm$ Std	Win	Tie	Lose	APS-VSS vs
SMALL	1NN	0.7367 $\pm$ 0.012	0.7388 $\pm$ 0.061	27	1	**12**	**0.031**
	LSIR	0.8693 $\pm$ 0.014	0.7415 $\pm$ 0.061	**28**	0	**12**	**0.001**
	SPMS-ALS	0.8733 $\pm$ 0.011	0.7549 $\pm$ 0.062	25	0	15	**0.044**
	APS-VSS	**0.8753** $\pm$ 0.015	**0.7656** $\pm$ 0.064	–	–	–	–
MEDIUM	1NN	0.8322 $\pm$ 0.006	0.8308 $\pm$ 0.017	**11**	11	**3**	0.08
	LSIR	0.9052 $\pm$ 0.004	0.8609 $\pm$ 0.014	8	0	9	0.644
	SPMS-ALS	0.9199 $\pm$ 0.003	0.8668 $\pm$ 0.011	6	1	10	0.431
	APS-VSS	**0.9271** $\pm$ 0.005	**0.8682** $\pm$ 0.014	–	–	–	–

Part A: Comparison with Baseline Models

The performance of our proposal and the baseline models is reported in Table 5.2. We look into the average training and test of the 10-folds cross-validation, the number of Wins, Ties and Losses of APS-VSS with the algorithm in the row in both small and medium datasets. The $p-$value is calculated to confirm if APS-VSS statistically outperforms the algorithm in the row. Several observations can be drawn from the numerical results:

- APS-VSS has obtained the highest average test accuracy in either small or medium datasets. p-values at the last column have confirmed the difference is significant in small datasets, while no significant difference found in medium ones.
- APS-VSS has a substantially greater number of Wins than Losses with respect to the three baseline models in small datasets. Specifically, it is more than 2 times compared with LSIR and 1NN, and more than 1.5 times compared with SPMS-ALS. On the contrary, those figures are distributed mostly equal im medium datasets which does not help distinguish which one outperforms the other.

Part B: Comparison with state-of-the-art techniques

APS-VSS statistically outperforms baseline models in small datasets, but no significant difference is found in medium datasets. However, APS-VSS is proposed to work on both combinatorial and continuous space, it is, thus, necessary to contrast its performance with state-of-the-art approaches. Note that, the three hybrid comparison methods share the same **IS** approach of SSMA as initialisation. Table 5.3 shares the same design of Table 5.2. The only difference is at the last column, where p-value refers to the Wilcoxon statistical test of the algorithm in the row to APS-VSS. This is because APS-VSS does not likely outperform these algorithms and p-value in that comparison direction is reported > 0.2. As a result, $p-$value $< \alpha(0.05)$ indicates APS-VSS is outperformed by the algorithm in the row. Several observations

can be made:

- Though average training performance goes up in the order of SSMA-LSHADE, SSMA-SFLSDE, SSMA-APMS-ALS and APS-VSS, the average test performance does not follow the same pattern in both small and medium datasets.
- In either small or medium datasets, APS-VSS has no significant difference with and SSMA-SFLSDE, SSMA-APMS-ALS. This information shows the robustness of APS-VSS with respect to several state-of-the-art methods in **IR** techniques.
- SSMA-LSHADE outperforms statistically APS-VSS in small datasets but the significant difference is rejected in medium datasets. Note that SSMA-LSHADE is a very complicated meta-heuristic technique, demanding an excessive amount of runtime to complete a run. We will report this aspect in Section 5.4.3.

TABLE 5.3: Summary performance between APS-VSS and other state-of-the-art models over small and medium datasets.

		TRAINING	TEST	APS-VSS vs			*p*-value
		Acc ± Std	Acc ± Std	Win	Tie	Lose	vs APS-VSS
SMALL	SSMA-LSHADE	0.8687 ± 0.010	**0.7792** ± 0.057	11	0	29	0.008
	SSMA-SFLSDE	0.8684 ± 0.011	0.7767 ± 0.059	15	0	25	0.096
	SSMA-SPMS-ALS	0.8727 ± 0.013	0.7670 ± 0.057	**19**	0	**21**	0.979
	APS-VSS	**0.8815** ± 0.015	0.7623 ± 0.061	–	–	–	–
MEDIUM	SSMA-LSHADE	0.9069 ± 0.004	**0.8706** ± 0.012	5	0	12	0.145
	SSMA-SFLSDE	0.9059 ± 0.004	0.8675 ± 0.013	**6**	0	**11**	0.712
	SSMA-SPMS-ALS	0.9264 ± 0.003	0.8700 ± 0.011	5	0	12	0.145
	APS-VSS	**0.9271** ± 0.005	0.8682 ± 0.014	–	–	–	–

5.4.3 Advantages of APS-VSS

From Section 5.4.2, we have seen that APS-VSS is only statistically outperformed by SSMA-LSHADE in small datasets. For other comparisons, our proposal outperforms or is statistically equivalent in either small or medium datasets. While maintaining a competitive performance, APS-VSS is more advantageous when looking at the reduction rate and runtime. In general, APS-VSS is capable of achieving similar reductions rates to existing hybrid approaches based on SSMA, but reducing hugely the required runtime to obtain the resulting reduced set. These two advantages will be reported in detail in the below Part A and Part B.

Part A: Reduction Rate

Figure 5.3 plots a bar chart starting with the original size in TR, followed by the reduction rate of SPMS-ALS, Hybrid approaches, and APS-VSS. At each algorithm, three pieces of information are written above each bar. At topmost, it is the average

size of the reduced set, then the percentage of the average size in relation to the original size, and at last the percentage of reduction. The second and third pieces of information are complementary to each other. At the top right corner, we also place a text mentioning the quantity of 1% of the original size for reference,

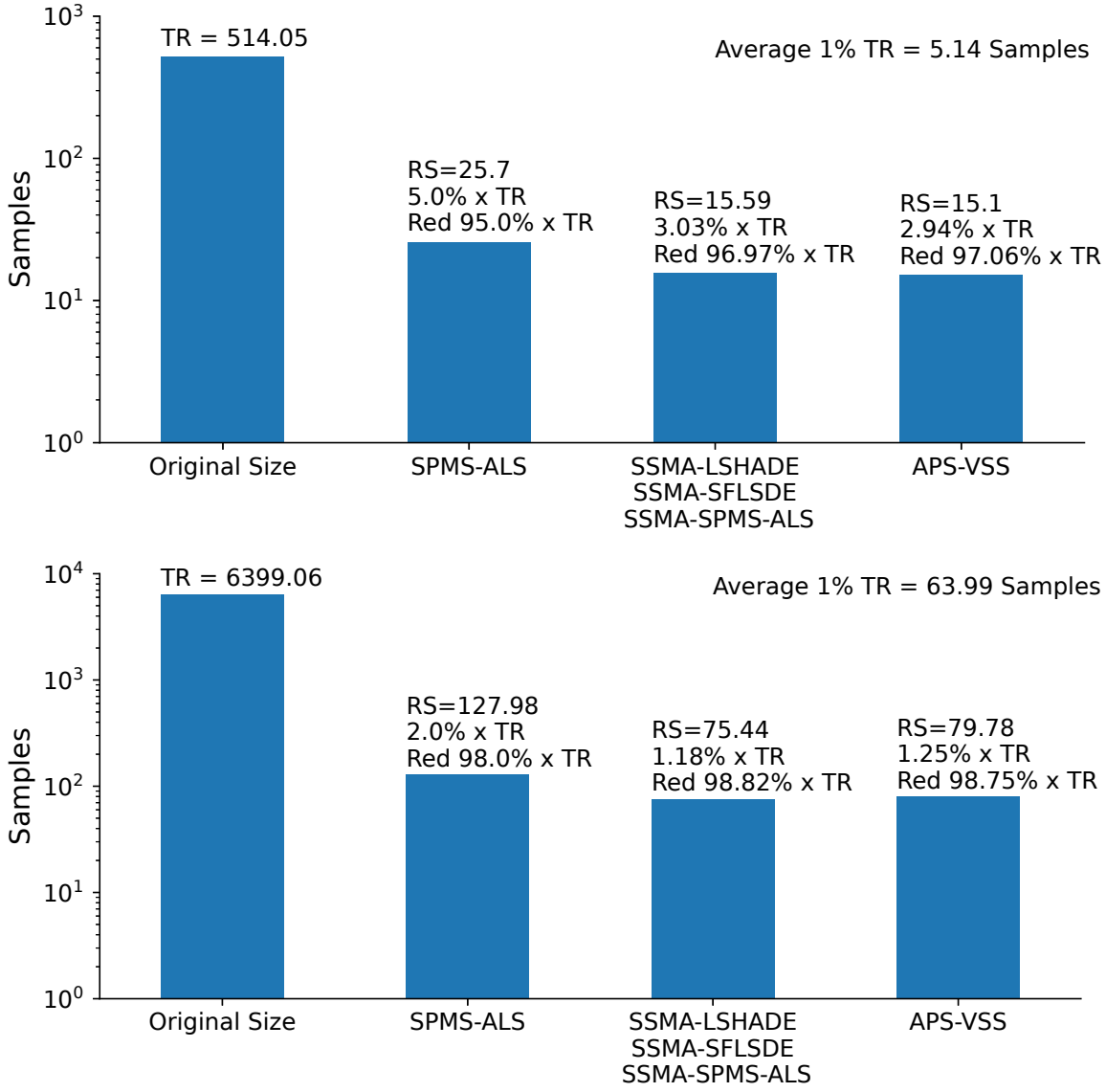


FIGURE 5.3: Reduction rate of APS-VSS against other examined algorithms over 57 datasets. The values on y-axis is displayed on the logarithmic scale.

SPMS-ALS has a fixed reduction rate of 5% and 2% for small and medium datasets. APS-VSS mostly shares the same reduction rate with hybrid methods, though it shrinks slightly more in small datasets. On average, APS-VSS has reduced 97.06% and 98.75% on small and medium datasets, respectively, while the hybrid approaches have saved 96.97% and 98.82% the original data size on small and medium datasets, respectively.

Part B: Runtime

Figure 5.4 shows how much faster the proposed APS-VSS is against all the comparison algorithms in small and medium datasets, respectively. This barplot shows the total average runtime required to perform IR. Given the runtime of APS-VSS as T(s), we also display, on top of each bar, the proportion of runtime that other algorithms take in relation to T(s) in both types of datasets.

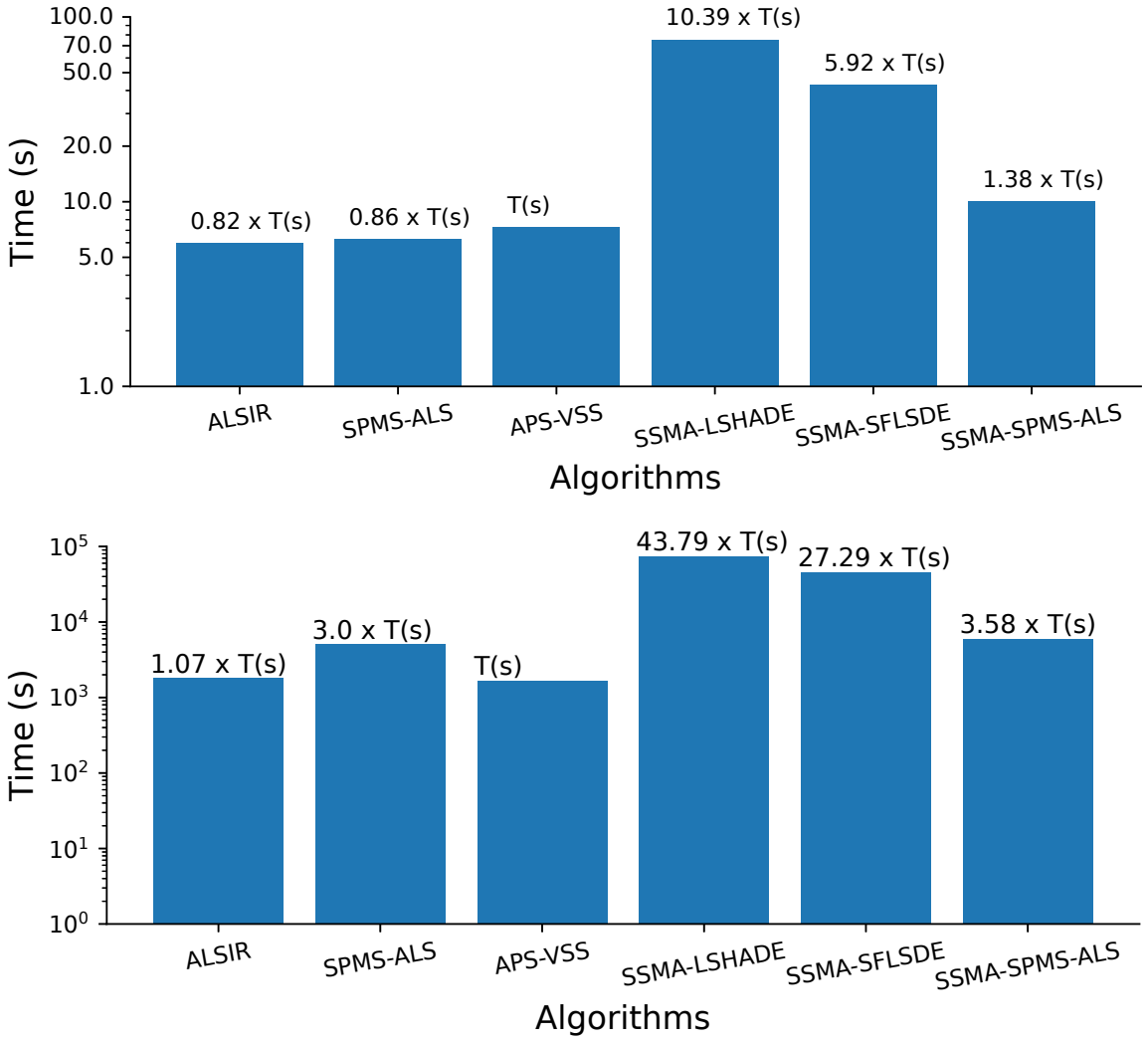


FIGURE 5.4: Average runtime (in seconds) of all compared algorithms and their multiplication to the runtime of APS-VSS T(s) over small (top) and medium (bottom) datasets. The values on y-axis is displayed on the logarithmic scale.

On small datasets, Figure 5.4 shows that although the runtime of APS-VSS is not always smaller than SPMS-ALS, its average absolute value is not far away. This is because APS-VSS has to spend fixed computation costs on multiple components. Considering the time consumed for the baselines, APS-VSS spends around 9 seconds while LSIR and SPMS-ALS are 1(s) or 2(s) lower. Other hybrid approaches substantially consume more runtime than APS-VSS. Particularly, SSMA-LSHADE

uses more than 10 times, SSMA-SFLSDE takes approximate 6 times, and SSMA-SPMS-ALS is 1.38 times the runtime used in APS-VSS. The benefit of the runtime is more obviously observed in medium datasets, where any compared algorithm requires more runtime than our solution. Specifically, APS-VSS reduces 3.58 times consumed in SSMA-SPMS-ALS and 43.79 times in SSMA-LSHADE. With respect to LSIR and SPMS-ALS, the range is from 1.07 times and 3 times, respectively.

Taking into consideration the accuracy results reported in the previous section, we would advise to use the proposed APS-VSS when using medium size datasets. We have observed that no significant differences can be found in terms of accuracy compared to the best hybrid **IR** algorithm (i.e. SSMA-LSHADE), but we can reduce the size of the training data in a much faster way, up to 43.79 times.

5.5 Summary

This chapter has provided a thorough examination of the proposed search framework for handling **IS** and **IG** simultaneously, which has covered our research objective 3. The proposed algorithm is a single point memetic structure that perturbs candidate solutions in the continuous space, i.e., performing the **IG** endowed with two LS mechanisms that attempt to shorten the length of the solutions, i.e., removing instances from the RS. The first LS attempts to remove instances from the solution, i.e. to perform IS, right before an **IG** cycle. The second LS attempts to remove potentially just generated redundant instances. This algorithmic approach is new in the domain-specific body of literature of IR.

Having considerations about the functioning of the NN classifier, APS-VSS is designed to take more advantage of a fast objective function computation. Thus, the acceleration is effectively used in guiding the search compared to other previous designs, which results in a competitive accuracy performance and more time-saving. The robustness of the proposed method with respect to performance, runtime and reduction rate is proved competitive to a few state-of-the-art hybrid methods existing in the literature, and statistically better than published algorithms using single-point search structure. The outputs of APS-VSS are valuable in the context of real-world problems when an application is required to be processed in a timely fashion. Also, these preliminary results may offer initial empirical evidence for investigating mixed continuous/combinatorial optimisation in data science.

As the final stage of the thesis, the developed methodology of this chapter has leveraged the achievements obtained in the preceding stage, thus confirming the covering of different accelerated search strategies for **IS** and **IG** completed. To conclude this thesis, in the next chapter we outline the final conclusions for the dissertation,

highlight key aspects of several work packages conducted and summarise the contributions. In addition, we also point out limitations of the research and some potential lines for future work.

Chapter 6

Conclusion

This chapter presents the final conclusions from different research works conducted in this study. Firstly, Section 6.1 outlines a set of contributions corresponding to the research gaps and objectives identified in the Introduction. Then, Section 6.2 points out several limitations that have been observed but unresolved in this dissertation. Finally, Section 6.3 draws some reflections on potential lines of future work.

6.1 Summary of Contributions

Meta-heuristic frameworks like EAs and different other derivative free search methods have been effectively employed to handle complex multimodal optimisation problems such as IR. Accelerating a search process has arrived to the forefront of scientific research as a valuable approach to quickly transform data into **Smart Data**. It is more valuable when the size of a dataset is relatively large, at which memetic algorithms or evolutionary-based optimisation methods are likely unable to produce high quality output in a reasonable amount of time. Although these search strategies have been successful to address IR in either continuous or combinatorial or both domains, they carry several drawbacks that hinder their deployment in different scenarios. It has been shown that there is a need of novel strategies to accelerate these effective derivative-free search methods, so that meta-heuristics, memetic computing with pattern search can fully demonstrate their potentials.

The research developed in this dissertation has contributed towards the improvement of different robust search strategies to handle IR problems in either continuous or combinatorial or both domains. A large number of datasets and multiple hyper-parameter settings designed in the experiments, together with a thorough analysis and the verification of different statistical tests have reinforced the insights concluded in each chapter. The thesis began with a thorough literature review of diverse approaches to address IR problem, then identified the strengths and drawbacks of those solutions, with an emphasis on those that are considered state-of-the-art, from which the research gaps and objectives have been established. Next,

details of different backgrounds and related works related to the solving problems have been completed, including key sections like meta-heuristics, fitness approximation, and memetic computing with pattern search. After this, three contributions presented in the three research stages have been developed covering two types of solutions for IR (i.e. IS and IG), in which early findings contribute to the later research outcomes. The key contributions deployed through the dissertation are revisited in the following paragraphs, where workable solutions are connected to the research gaps and objectives initially established in the introductory chapter.

- The first contribution (Chapter 3) has addressed the high computational cost of fitness evaluation in a population-based evolutionary search algorithm (i.e. CHC). The key novelty of the proposed approach lies in the acceleration of the fitness evaluation in the context of imbalanced classification. The acceleration mechanism was conducted in two stages: Firstly, a preliminary clustering stage of majority examples transforms a binary chromosome into another form which can describe the overall location of the existing samples in that solution. This stage defines another equivalent encoding with new features that can effectively group similar chromosomes. Secondly, another clustering stage exploits the newly defined chromosome representation to categorise similar elements into groups, and thus each chromosome's fitness can be approximated based on its distance to a selected representative of the cluster it belongs to. This chapter has introduced a new population-based meta-heuristic search framework composing an evolutionary search and a surrogate model for fitness evaluation. The reported results with the verification of different statistical tests have confirmed the achievement of Objective 1, as the runtime has been greatly reduced while maintaining the best so far achieved performance.

- After proposing successfully an effective and time-efficient solution for IS in Chapter 3, Chapter 4 has introduced a movel derivative-free search method for IG, which consists of Single-Point Search and MC. Various aspects of the proposed single-point memetic search structure have been analysed with a large number of datasets, including the number of evaluations (i.e. computational budget), different forms of a single-point search starting from a naive structure to a more complex one. As a results, SPMS-ALS has been successful to demonstrate its effectiveness and time-efficiency when reporting a significantly low runtime to achieve mostly equivalent high performance in comparison with the state-of-the-art approaches. These achievements have covered Objective 2 of the thesis: a fast and yet effective domain-tailored memetic approach for IG. In addition, fruitful insights obtained in this chapter are key achievements for the development of a novel methodology handling both IS and IG in the subsequent research stage.

- From the success of accelerating different search strategies in Chapters 3 and 4, the output RS has been time-efficiently obtained from a selection or generation process, providing fundamentals to conduct the first attempt of integrating those two processes into one single search framework (Chapter 5). A new search design has been developed adopting the well-established algorithm in Chapter 4 plus several newly designed LS operators tailored to the problem domain. We have demonstrated the improvement of the runtime used in this new framework with respect to that of the previous algorithmic designs and state-of-the-art search approaches. Achievements in Chapter 5 contains Objective 2, given that the algorithmic design is simple and effective in handling IG. However, they mainly cover Objective 3 when presenting an IR solution to work on both combinatorial and continuous search spaces, which is different from other studies conducting either IS or IG separately. Being consistent with the main theme of accelerating search algorithms in this thesis, this novel algorithmic design is associated with an acceleration mechanism for the objective function, and thus can speed up the search.

In conclusion, the contributions summarised throughout different chapters above have thoroughly fulfilled the three defined thesis objectives. Each chapter has contributed in its own way towards accelerating a search strategy for IR.

6.2 Limitations

In relation to the contributions summarised above, the research work completed in this dissertation also contains limitations that need to be acknowledged. In this section, we discuss those shortcomings which would reinforce the research conducted and may also suggest ideas for future work.

- The experiments in which two-stage clustering was employed for fitness approximation may benefit from further an analysis of parameter tuning. The analysis on the effect of parameters (i.e. k_1, k_2) in the two-stage clustering may provide more in-depth understanding. In addition, features transformed from the binary chromosome representation to an intermediate form, presented in Section 3.3.2, was preliminary examined by only k-means. Employing other powerful clustering methods or even supervised learning models may explore more insights and/or provide solutions for other applications. Hence, an analysis of using these features in such different scenarios can provide more understanding and highlight their significance.
- As stated in the thesis the proposed search frameworks are designed for 1-NN, a well-known instance-based learning approach. Though the resulting

RS can be employed in other classifiers, other aspects such as tuning hyper-parameters may be required to obtain the right RS for a specific classifier. In this regard, this thesis will limit its discussion to the use of the proposed search frameworks for 1-NN only, leaving the deployment of the RS with other classifiers unexamined. It is noted that the accelerated computation of some search frameworks (i.e. APMS-ALS, APS-VSS) may be infeasible in learning with other classifiers as they are specifically designed for k-NN.

- The entire conducted research has employed the case study of classification with small and medium datasets. Other scenarios like Big Data classification or feature reduction have not been examined with the proposed search frameworks, though empirical numerical results in those scenarios may provide more understanding and can further demonstrate their robustness. The time constraint of completing those experiments and a high demand of computational resources have limited the scope of experimental design of the dissertation.

The limitations discussed above also represent for a set of enhancement that may suggest immediate lines of work in the next research stage. In the next section, we relate our current research stage to other sectors of investigation that would either develop further the research work or deploy the found insights in other applications.

6.3 Future work

The research work conducted in this dissertation has aimed to enhance the speed of derivative-free search techniques, bridging the gaps in the intersection of the three domains of study, namely: ML, data mining and optimisation. The proposed fast and effective search methods can extend their applicability to other research fields or applications. In the following, we remark several ideas for future work derived from our understanding about the findings and limitations. These ideas may be applicable in deploying the success of our search frameworks in other applications or in digging further the insights that have not been exploited.

- The curse of dimensionality has caused a lot of challenges to many learning models, which has captured a lot of attention in recent works (Ayesha, Hanif, and Talib, 2020). This problem has been tackled by various approaches as discussed in Chapter 1. To participate in the family of solutions for feature reduction, our different search frameworks present a wide range of advantages to address the problem. The frameworks are now focusing on features

(i.e. columns in the matrix of TR) to shrink the size of features or to generate artificial feature values.

- As mentioned above, learning with more powerful models like ensemble may reach higher classification accuracy (Rokach, 2010; Galar et al., 2011). Ensemble-based learning methods (e.g. Bagging, Boosting) combine multiple weak classifiers to form ensembles of diverse base models (Chawla et al., 2003), which have been proved more accurate than a non-ensemble learning approach (Galar et al., 2011). To adopt the proposed search designs into other classifiers, more investigation of understanding the learning models is necessary. While SPMS-ALS or APS-VSS needs re-designing to maintain the acceleration mechanism, EUSC and EUSHC are applicable for any classifier. Considering this important characteristic of EUSC and EUSHC, a continuation of the research on these two robust evolutionary search frameworks would be promising with ensemble learning to obtain higher accuracy performance for imbalanced classification with respect to EUS. From the insights gained in Chapter 3, the accelerated search mechanism is highly possible to speed up EUSBoost (Galar et al., 2013), the state-of-the-art method in the family of undersampling techniques for addressing imbalanced classification.

- Broaden the topic to Big data scenario where the examined datasets are not small or medium, but huge. Learning algorithms may need to be re-designed to capture the insights from the datasets through distributed Big Data technologies depending on what approach to be used (i.e. global and local). While a global approach considers the entire dataset as one distribution, a local method develops a solution with the split dataset locally available at a computing node, which our search frameworks can be employed without any modification to handle Big Data classification problem. Other local approaches employing evolutionary-based search may require more split from the original source because excessive time is consumed to process a large volume of samples with these techniques. In addition, the search may be misled with a small portion of the original source. Our search frameworks have a significant lower runtime to handle the same number of samples, so that, the original data can be split to fewer chunks. On top of having lower possibility of misleading the search, this deployment can contribute to time and energy saving as it allows a lower number of computing nodes to handle a larger volume for dataset within a reasonable amount of time.

- For EUSC, the features defined in the intermediate representation can be examined in supervised learning as follows: Initially, chromosomes are vastly generated to cover the fitness space. Next, these chromosome samples are transformed into an equivalent form where the features were constructed as

guided in Section 3.3.2. The transformed vectors will be fed to other powerful learning models for learning. Lastly, the learned model will be used globally for fitness prediction when the search requires an evaluation. As a result, constructing a supervised learning model may have a positive impact on the search performance (i.e. runtime or accuracy), and thus may extend the contribution of our research in the literature.

- The reader might wonder how well the classification performance is when the resulting RS is used directly by any other classifier like ensemble model, decision tree or SVM. Although all of the proposed search frameworks are intentionally designed for 1NN, the resulting set can be potentially employed by another learning model. Parameter tuning may be investigated to obtain the right size of an RS with respect to the decision boundary of a classifier. Our first preliminary experiments using directly the resulting RS as training data for other learning models were reported in Section 4.4.6 and the results showed a clear overfitting issue at the learned model. However, it is promising that learning with other classifiers exhibits several areas of improvement that could uphold and extend the insights presented in this dissertation.

Bibliography

Acampora, G., V. Loia, and M. Gaeta (2010). "Exploring e-Learning Knowledge Through Ontological Memetic Agents". In: *IEEE Computational Intelligence Magazine* 5.2, pp. 66–77.

Ahmad, Iftikhar (2015). "Feature selection using particle swarm optimization in intrusion detection". In: *International Journal of Distributed Sensor Networks* 11.10, p. 806954.

Aichholzer, O et al. (2002). "Evolution strategy and hierarchical clustering". In: *IEEE Transactions on Magnetics* 38.2, pp. 1041–1044.

Alexandropoulos, Stamatios-Aggelos N, Sotiris B Kotsiantis, and Michael N Vrahatis (2019). "Data preprocessing in predictive data mining". In: *The Knowledge Engineering Review* 34.

Algethami, Haneen and Dario Landa-Silva (2017). "Diversity-based adaptive genetic algorithm for a Workforce Scheduling and Routing Problem". In: *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, pp. 1771–1778.

Algethami, Haneen, Anna Martínez-Gavara, and Dario Landa-Silva (2019). "Adaptive multiple crossover genetic algorithm to solve workforce scheduling and routing problem". In: *Journal of Heuristics* 25.4, pp. 753–792.

Alpaydin, Ethem (2014). *Introduction to machine learning*. MIT press.

Amalina, Fairuz et al. (2019). "Blending big data analytics: Review on challenges and a recent study". In: *IEEE Access* 8, pp. 3629–3645.

Amaya, Jhon Edgar et al. (2020). "Deep memetic Models for Combinatorial Optimization Problems: Application to the Tool Switching Problem". In: *Memetic Computing* 12.1, pp. 3–22.

Andréasson, N., M. Patriksson, and A. Evgrafov (2020). *An introduction to continuous optimization: foundations and fundamental algorithms*. Courier Dover Publications.

Arnaiz-González, Álvar et al. (2016). "Instance selection for regression: Adapting DROP". In: *Neurocomputing* 201, pp. 66–81.

Atkeson, Christopher G, Andrew W Moore, and Stefan Schaal (1997). "Locally weighted learning". In: *Lazy learning*, pp. 11–73.

Atla, Abhinav et al. (2011). "Sensitivity of different machine learning algorithms to noise". In: *Journal of Computing Sciences in Colleges* 26.5, pp. 96–103.

Ayesha, Shaeela, Muhammad Kashif Hanif, and Ramzan Talib (2020). "Overview and comparative study of dimensionality reduction techniques for high dimensional data". In: *Information Fusion* 59, pp. 44–58.

Bacardit, Jaume et al. (2004). "Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy". In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 1021–1031.

Back, T., M. Emmerich, and O. Shir (2008). "Evolutionary algorithms for real world applications". In: *IEEE Computational Intelligence Magazine.* 3.1.

Barandela, Ricardo et al. (2003). "Strategies for learning in class imbalance problems". In: *Pattern Recognition* 36.3, pp. 849–851.

Bartel, Christopher J (2021). "Data-centric approach to improve machine learning models for inorganic materials". In: *Patterns* 2.11, p. 100382.

Bartz-Beielstein, Thomas and Martin Zaefferer (2017). "Model-based methods for continuous and discrete global optimization". In: *Applied Soft Computing* 55, pp. 154–167.

Batista, Gustavo EAPA, Ronaldo C Prati, and Maria Carolina Monard (2004). "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD Explorations Newsletter* 6.1, pp. 20–29.

Bertini, Ilaria et al. (2011). "Soft computing based optimization of combined cycled power plant start-up operation with fitness approximation methods". In: *Applied Soft Computing* 11.6, pp. 4110–4116.

Bianchi, Leonora and Marco Dorigo (2006). "Ant colony optimization and local search for the probabilistic traveling salesman problem: a case study in stochastic combinatorial optimization". PhD dissertation. Université libre de Bruxelles.

Box, MJ (1969). "Non-linear optimization techniques". In: *Nomograph No. 5, IC I* 20.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge university press.

Bradley, Andrew P (1997). "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern Recognition* 30.7, pp. 1145–1159.

Brent, Richard P (1973). *Algorithms for Minimization without Derivatives, chap. 4*.

Brighton, Henry and Chris Mellish (2002). "Advances in instance selection for instance-based learning algorithms". In: *Data mining and knowledge discovery* 6.2, pp. 153–172.

Brownlee, Alexander EI and Jonathan A Wright (2015). "Constrained, mixed-integer and multi-objective optimisation of building designs by NSGA-II with fitness approximation". In: *Applied Soft Computing* 33, pp. 114–126.

Buche, Dirk, Nicol N Schraudolph, and Petros Koumoutsakos (2005). "Accelerating evolutionary algorithms with Gaussian process fitness function models". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35.2, pp. 183–194.

Bui, Lam T, Hussein A Abbass, and Daryl Essam (2005). "Fitness inheritance for noisy evolutionary multi-objective optimization". In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. ACM, pp. 779–785.

Burke, E. K. et al. (2013). "Hyper-heuristics: A survey of the state of the art". In: *Journal of the Operational Research Society.* 64.12, pp. 1695–1724.

Burke, Edmund K et al. (2010). "A classification of hyper-heuristic approaches". In: *Handbook of metaheuristics*. Springer, pp. 449–468.

Cano, J. R., F. Herrera, and M. Lozano (2003). "Using Evolutionary Algorithms as Instance Selection for Data reduction in KDD: an experimental study". In: *IEEE Transactions on Evolutionary Computation* 7.6, pp. 561–575.

Cano, José Ramón, Francisco Herrera, and Manuel Lozano (2005). "Stratification for scaling up evolutionary prototype selection". In: *Pattern Recognition Letters* 26.7, pp. 953–963.

Caponio, A. et al. (2007). "A Fast Adaptive Memetic Algorithm for On-line and Off-line Control Design of PMSM Drives". In: *IEEE Transactions on System Man and Cybernetics-part B, Special Issue on Memetic Algorithms* 37.1, pp. 28–41.

Caraffini, F. et al. (2014). "Re-sampled Inheritance Search: High Performance Despite the Simplicity". In: *Soft Computing* 17.12, pp. 2235–2256.

Caraffini, Fabio, Ferrante Neri, and Michael G. Epitropakis (2019). "HyperSPAM: A Study on Hyper-heuristic Coordination Strategies in the Continuous Domain". In: *Information Sciences* 477, pp. 186–202.

Caraffini, Fabio, Ferrante Neri, and Giovanni Iacca (2017). "Large Scale Problems in Practice: The Effect of Dimensionality on the Interaction Among Variables". In: *Applications of Evolutionary Computation*. Ed. by Giovanni Squillero and Kevin Sim. Springer, pp. 636–652.

Caraffini, Fabio, Ferrante Neri, and Lorenzo Picinali (2014). "An analysis on separability for memetic computing automatic design". In: *Information Sciences* 265, pp. 1–22.

Cartis, Coralia and Lindon Roberts (2019). "A derivative-free Gauss–Newton method". In: *Mathematical Programming Computation* 11.4, pp. 631–674.

Cartis, Coralia et al. (2019). "Improving the flexibility and robustness of model-based derivative-free optimization solvers". In: *ACM Transactions on Mathematical Software (TOMS)* 45.3, pp. 1–41.

Cavalcanti, George DC and Rodolfo JO Soares (2020). "Ranking-based Instance Selection for Pattern Classification". In: *Expert Systems with Applications* 150, p. 113269.

Cervantes, Alejandro, Inés María Galván, and Pedro Isasi (2009). "AMPSO: A New Particle Swarm Method for Nearest Neighborhood Classification". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.5, pp. 1082–1091.

Chang-Yong Lee and Xin Yao (2004). "Evolutionary Programming Using Mutations Based on the Levy Probability Distribution". In: *IEEE Transactions on Evolutionary Computation* 8.1, pp. 1–13.

Chawla, Nitesh V et al. (2002). "SMOTE: synthetic minority over-sampling technique". In: *Journal of Artificial Intelligence Research* 16, pp. 321–357.

Chawla, Nitesh V et al. (2003). "SMOTEBoost: Improving prediction of the minority class in boosting". In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pp. 107–119.

Chen, X. et al. (2011). "A Multi-Facet Survey on Memetic Computation". In: *IEEE Transactions on Evolutionary Computation* 15.5, pp. 591–607.

Chen, Zhenxiang et al. (2018). "Machine learning based mobile malware detection using highly imbalanced network traffic". In: *Information Sciences* 433, pp. 346–364.

Choong, S. S., L. Wong, and C. P. Lim (2018). "Automatic design of hyper-heuristic based on reinforcement learning". In: *Information Sciences* 436, pp. 89–107.

Chugh, Tinkle et al. (2020). "Surrogate-Assisted Evolutionary Optimization of Large Problems". In: *High-Performance Simulation-Based Optimization*. Springer, pp. 165–187.

Conn, Andrew R, Katya Scheinberg, and Luis N Vicente (2009). *Introduction to derivative-free optimization*. SIAM.

Cover, T. M. and P. E. Hart (1967). "Nearest Neighbor Pattern Classification". In: *IEEE Transactions on Information Theory* 13.1, pp. 21–27.

Curtis, Frank E and Katya Scheinberg (2017). "Optimization methods for supervised machine learning: From linear models to deep learning". In: *Leading Developments from INFORMS Communities*. INFORMS, pp. 89–114.

Czarnowski, Ireneusz (2012). "Cluster-based instance selection for machine classification". In: *Knowledge and Information Systems* 30.1, pp. 113–133.

Dai, Hong-Liang (2015). "Imbalanced protein data classification using ensemble FTM-SVM". In: *IEEE Transactions on Nanobioscience* 14.4, pp. 350–359.

Dean, Jeffrey and Sanjay Ghemawat (2010). "MapReduce: a flexible data processing tool". In: *Communications of the ACM* 53.1, pp. 72–77.

Demšar, Janez (2006). "Statistical Comparisons of Classifiers over Multiple Data sets". In: *Journal of Machine Learning ReSearch* 7, pp. 1–30.

Deng, Yong, Yang Liu, and Deyun Zhou (2015). "An improved genetic algorithm with initial population strategy for symmetric TSP". In: *Mathematical Problems in Engineering* 2015.

Derrac, Joaquín, Salvador García, and Francisco Herrera (2010). "Stratified prototype selection based on a steady-state memetic algorithm: a study of scalability". In: *Memetic Computing* 2.3, pp. 183–199.

– (2012). "A survey on evolutionary instance selection and generation". In: *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends*. IGI Global, pp. 233–266.

Derrac, Joaquín et al. (2011). "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms". In: *Swarm and Evolutionary Computation* 1.1, pp. 3–18.

Diaz-Gomez, Pedro A and Dean F Hougen (2007). "Initial Population for Genetic Algorithms: A Metric Approach." In: *GEM*, pp. 43–49.

Domingos, Pedro (1999). "Metacost: A general method for making classifiers cost-sensitive". In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 155–164.

Dorigo, M. and L. M. Gambardella (1997). "Ant colony system: a cooperative learning approach to the traveling salesman problem". In: *IEEE Transactions on Evolutionary Computation* 1.1, pp. 53–66.

Elola, Andoni et al. (2017). "Hybridizing Cartesian Genetic Programming and Harmony Search for Adaptive Feature Construction in Supervised Learning Problems". In: *Applied Soft Computing* 52, pp. 760–770.

Erhan, Dumitru et al. (2014). "Scalable object detection using deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2154.

Eshelman, Larry J (1991). "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination". In: *Foundations of Genetic Algorithms* 1, pp. 265–283.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996a). "The KDD process for extracting useful knowledge from volumes of data". In: *Communications of the ACM* 39.11, pp. 27–34.

Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth (1996b). "From data mining to knowledge discovery in databases". In: *AI magazine* 17.3, pp. 37–37.

Feng, L. et al. (2015). "Memetic Search With Interdomain Learning: A Realization Between CVRP and CARP". In: *IEEE Transactions on Evolutionary Computation* 19.5, pp. 644–658. ISSN: 1941-0026.

Fernandes, Everlandio et al. (2019). "Ensemble of Classifiers based on MultiObjective Genetic Sampling for Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 14, pp. 1041–4347.

Fernández, Alberto et al. (2014). "Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4.5, pp. 380–409.

Fernández, Alberto et al. (2018a). *Learning from imbalanced data sets*. Springer.

Fernández, Alberto et al. (2018b). "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary". In: *Journal of Artificial Intelligence Research* 61, pp. 863–905.

Fernández-Delgado, Manuel et al. (2014). "Do we need hundreds of classifiers to solve real world classification problems". In: *Journal of Machine Learning Research* 15.1, pp. 3133–3181.

Filippone, Maurizio et al. (2008). "A survey of kernel and spectral methods for clustering". In: *Pattern recognition* 41.1, pp. 176–190.

Fister, Iztok et al. (2019). "Novelty Search for global Optimization". In: *Applied Mathematics and Computation* 347, pp. 865–881.

Galar, Mikel et al. (2011). "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4, pp. 463–484.

Galar, Mikel et al. (2013). "EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling". In: *Pattern Recognition* 46.12, pp. 3460–3471.

García, S., Julián Luengo, and F. Herrera (2015). *Data Preprocessing in Data Mining*. Springer.

García, S. et al. (2012). "Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3, pp. 417–435.

García, Salvador, José Ramón Cano, and Francisco Herrera (2008). "A Memetic Algorithm for Evolutionary Prototype Selection: A Scaling up Approach". In: *Pattern Recognition* 41.8, pp. 2693–2709.

García, Salvador and Francisco Herrera (2009). "Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy". In: *Evolutionary Computation* 17.3, pp. 275–306.

García, Salvador et al. (2009). "Diagnose effective evolutionary prototype selection using an overlapping measure". In: *International Journal of Pattern Recognition and Artificial Intelligence* 23.08, pp. 1527–1548.

García, Salvador et al. (2010). "Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power". In: *Information Sciences* 180.10, pp. 2044–2064.

Garcia, Salvador et al. (2012). "Prototype selection for nearest neighbor classification: Taxonomy and empirical study". In: *IEEE transactions on pattern analysis and machine intelligence* 34.3, pp. 417–435.

García, Salvador et al. (2018). "Dynamic ensemble selection for multi-class imbalanced datasets". In: *Information Sciences* 445, pp. 22–37.

García-Gil, Diego et al. (2019). "Enabling smart data: noise filtering in big data classification". In: *Information Sciences* 479, pp. 135–152.

Glover, F. (1989). "Tabu search—part I". In: *ORSA Journal on Computing*. 1.3, pp. 190–206.

Gündüz, Deniz et al. (2019). "Machine learning in the air". In: *IEEE Journal on Selected Areas in Communications* 37.10, pp. 2184–2199.

Gupta, Abhishek and Yew-Soon Ong (2019). *Memetic Computation. The Mainspring of Knowledge Transfer in a Data-Driven Optimization Era*. Adaptation, Learning, and Optimization. Springer.

Haixiang, Guo et al. (2017). "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems with Applications* 73, pp. 220–239.

Hajjaji, Yosra et al. (2021). "Big data and IoT-based applications in smart environments: A systematic review". In: *Computer Science Review* 39, p. 100318.

Han, Jiawei, Jian Pei, and Micheline Kamber (2011). *Data mining: concepts and techniques*. Elsevier.

Hart, Peter (1968). "The condensed nearest neighbor rule (corresp.)" In: *IEEE Transactions on Information Theory* 14.3, pp. 515–516.

He, Haibo and Edwardo A Garcia (2008). "Learning from imbalanced data". In: *IEEE Transactions on Knowledge & Data Engineering* 9, pp. 1263–1284.

Hedjazi, Lyamine et al. (2015). "Membership-margin based feature selection for mixed type and high-dimensional data: Theory and applications". In: *Information Sciences* 322, pp. 174–196.

Hillier, Frederick S (2012). *Introduction to operations research*. Tata McGraw-Hill Education.

Hodges, JL, Erich L Lehmann, et al. (1962). "Rank methods for combination of independent experiments in analysis of variance". In: *The Annals of Mathematical Statistics* 33.2, pp. 482–497.

Holm, Sture (1979). "A simple sequentially rejective multiple test procedure". In: *Scandinavian Journal of Statistics*, pp. 65–70.

Iacca, G. et al. (2012). "Ockham's Razor in Memetic Computing: Three Stage Optimal Memetic Exploration". In: *Information Sciences* 188, pp. 17–43.

Iafrate, Fernando (2014). "A journey from big data to smart data". In: *Digital Enterprise Design & Management*. Springer, pp. 25–33.

Jana, N. D., J. Sil, and S. Das (2018). "Continuous fitness landscape analysis using a chaos-based random walk algorithm". In: *Soft Computing* 22, pp. 921–948.

Jensi, R and Dr G Wiselin Jiji (2014). "A survey on optimization approaches to text document clustering". In: *arXiv preprint arXiv:1401.2229*.

Jin, Yaochu (2005). "A comprehensive survey of fitness approximation in evolutionary computation". In: *Soft Computing* 9.1, pp. 3–12.

Jin, Yaochu (2011). "Surrogate-assisted evolutionary computation: Recent advances and future challenges". In: *Swarm and Evolutionary Computation* 1.2, pp. 61–70.

Jin, Yaochu, Markus Olhofer, and Bernhard Sendhoff (2000). "On evolutionary optimization with approximate fitness functions". In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., pp. 786–793.

Jin, Yaochu and Bernhard Sendhoff (2004). "Reducing fitness evaluations using clustering techniques and neural network ensembles". In: *Genetic and Evolutionary Computation Conference*. Springer, pp. 688–699.

Katuwal, Rakesh, Ponnuthurai Nagaratnam Suganthan, and Le Zhang (2020). "Heterogeneous Oblique Random Forest". In: *Pattern Recognition* 99, p. 107078.

Kaupe Jr, Arthur F (1963). "Algorithm 178: direct search". In: *Communications of the ACM* 6.6, pp. 313–314.

Kennedy, J. and R. Eberhart (1995). "Particle swarm optimization". In: *Proceedings of the International Conference on Neural Networks*. Vol. 4, pp. 1942–1948.

Kim, Hee-Su and Sung-Bae Cho (2001). "An efficient genetic algorithm with less fitness evaluation by clustering". In: *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. Vol. 2. IEEE, pp. 887–894.

Kordos, Mirosław, Marcin Blachnik, and Rafał Scherer (2022). "Fuzzy clustering decomposition of genetic algorithm-based instance selection for regression problems". In: *Information Sciences* 587, pp. 23–40.

Krawczyk, Bartosz, Michał Woźniak, and Gerald Schaefer (2014). "Cost-sensitive decision tree ensembles for effective imbalanced classification". In: *Applied Soft Computing* 14, pp. 554–562.

Krawczyk, Bartosz et al. (2016). "Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy". In: *Applied Soft Computing* 38, pp. 714–726.

Kumari, A Charan, K Srinivas, and MP Gupta (2013). "Software module clustering using a hyper-heuristic based multi-objective genetic algorithm". In: *Advance Computing Conference (IACC), 2013 IEEE 3rd International*. IEEE, pp. 813–818.

Larose, Daniel T and Chantal D Larose (2014). *Discovering knowledge in data: an introduction to data mining*. Vol. 4. John Wiley & Sons.

Le, H. L., F. Neri, and I. Triguero (2021). "SPMS-ALS: A Single-Point Memetic structure with accelerated local search for instance reduction". In: *Swarm and Evolutionary Computation*, p. 100991.

Le, M. N. et al. (2009). "Lamarckian memetic algorithms: local optimum and connectivity structure analysis". In: *Memetic Computing Journal* 1.3, pp. 175–190.

Lenk, Alexander et al. (2015). "Towards a taxonomy of standards in smart data". In: *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 1749–1754.

Leung, Yiu-Wing and Yuping Wang (2001). "An orthogonal genetic algorithm with quantization for global numerical optimization". In: *IEEE Transactions on Evolutionary Computation* 5.1, pp. 41–53.

Leyva, Enrique, Antonio González, and Raúl Pérez (2015). "Three New Instance Selection Methods Based on Local Sets: A Comparative Study with Several Approaches from a Bi-objective Perspective". In: *Pattern Recognition* 48.4, pp. 1523–1537.

Li, Kuan et al. (2014). "Boosting weighted ELM for imbalanced learning". In: *Neurocomputing* 128, pp. 15–21.

Li, Rui et al. (2008). "Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis". In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, pp. 2764–2771.

Li, Xiaodong and Xin Yao (2012). "Cooperatively Coevolving Particle Swarms for Large Scale Optimization". In: *Evolutionary Computation, IEEE Transactions on* 16.2, pp. 210–224.

Lin, Lin and Mitsuo Gen (2018). "Hybrid evolutionary optimisation with learning for production scheduling: state-of-the-art survey on algorithms and applications". In: *International Journal of Production Research* 56.1-2, pp. 193–223.

Liu, Huan and Hiroshi Motoda (2007). *Computational methods of feature selection*. CRC Press.

Liu, Yu et al. (2019). "A data-centric Internet of Things framework based on azure cloud". In: *IEEE Access* 7, pp. 53839–53858.

Lloyd, S. (1982). "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137.

López, Victoria et al. (2013). "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics". In: *Information Sciences* 250, pp. 113–141.

López-García, Pedro et al. (2016). "GACE: A Meta-heuristic Based in the Hybridization of Genetic Algorithms and Cross Entropy Methods for Continuous Optimization". In: *Expert Systems with Applications* 55, pp. 508–519.

Lourenço, H. R., O. C. Martin, and T. Stützle (2003). "Iterated local search". In: *Handbook of Metaheuristics*. Kluwer Academic Publishers, pp. 320–353.

Lu, Jie et al. (2018). "Learning under concept drift: A review". In: *IEEE Transactions on Knowledge and Data Engineering* 31.12, pp. 2346–2363.

Luengo, Julián, Salvador García, and Francisco Herrera (2012). "On the choice of the best imputation methods for missing values considering three groups of classification methods". In: *Knowledge and information systems* 32.1, pp. 77–108.

Ma, L. et al. (2020). "Cost-Aware Robust Control of Signed Networks by Using a Memetic Algorithm". In: *IEEE Transactions on Cybernetics*. to appear, pp. 1–14.

Ma, X. et al. (2019). "A Survey on Cooperative Co-Evolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation* 23.3, pp. 421–441.

MacQueen, James et al. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA, pp. 281–297.

Magyar, Gábor, Mika Johnsson, and Olli Nevalainen (2000). "An adaptive hybrid genetic algorithm for the three-matching problem". In: *IEEE Transactions on Evolutionary Computation* 4.2, pp. 135–146.

Malan, K. M. and A. P. Engelbrecht (2013). "A survey of techniques for characterising fitness landscapes and some possible ways forward". In: *Information Sciences* 241, pp. 148 –163.

Marchiori, Elena (2009). "Class conditional nearest neighbor for large margin instance selection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.2, pp. 364–370.

Martinez, Aritz D. et al. (2019). "Hybridizing differential evolution and novelty Search for multimodal Optimization Problems". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pp. 1980–1989.

Martínez-Estudillo, Alfonso C et al. (2005). "Hybridization of evolutionary algorithms and local search by means of a clustering method". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 36.3, pp. 534–545.

Mitchell, Tom M (1997). "Machine learning (mcgraw-hill international editions computer science series)". In.

Mockus, J. (2012). *Bayesian approach to global optimization: theory and applications*. Vol. 37. Springer Science & Business Media.

Moraglio, Alberto and Ahmed Kattan (2011). "Geometric generalisation of surrogate model based optimisation to combinatorial spaces". In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, pp. 142–154.

Moscato, Pablo (1989). *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*. Tech. rep. 826.

Moscato, Pablo and Michael Norman (1989). *A Competitive and Cooperative Approach to Complex Combinatorial Search*. Tech. rep. 790.

Nanni, Loris and Alessandra Lumini (2009). "Particle Swarm Optimization for Prototype Reduction". In: *NeuroComputing* 72.4-6, pp. 1092–1097.

Narendra, Patrenahalli M. and Keinosuke Fukunaga (1977). "A branch and bound algorithm for feature subset selection". In: *IEEE Transactions on computers* 9.C-26, pp. 917–922.

Nareyek, Alexander (2003). "Choosing search heuristics by non-stationary reinforcement learning". In: *Metaheuristics: Computer decision-making*. Springer, pp. 523–544.

Nelder, JA (1965). "Mead. R.,"A simplex method for function minimum"". In: *Computer Journal* 7, pp. 308–313.

Neri, F. and C. Cotta (2012). "Memetic algorithms and memetic computing optimization: A literature review". In: *Swarm and Evolutionary Computation* 2, pp. 1–14.

Neri, F. and S. Rostami (2021). "Generalised Pattern Search Based on Covariance Matrix Diagonalisation". In: *SN Comput. Sci.* 2.3, p. 171.

Neri, F. and I. Triguero (2020). "A Local Search with a Surrogate Assisted Option for Instance Reduction". In: *Applications of Evolutionary Computation*. Vol. 12104. Springer, pp. 578–594.

Neri, Ferrante (2019). *Linear Algebra for Computational Sciences and Engineering*. second. Springer.

Neri, Ferrante, Giovanni Iacca, and Ernesto Mininno (2011). "Disturbed Exploitation Compact Differential Evolution for Limited Memory Optimization Problems". In: *Information Sciences* 181.12, pp. 2469 –2487.

Nguyen, Quang Huy et al. (2009). "Adaptive Cellular Memetic Algorithms no access". In: *Evolutionary Computation* 17.2, pp. 231–256.

Nogueras, Rafael and Carlos Cotta (2016). "Studying Self-balancing Strategies in Island-based Multimemetic Algorithms". In: *Journal of Computational and Applied Mathematics* 293, pp. 180–191.

Oh, Sang-Hoon (2011). "Error back-propagation algorithm for classification of imbalanced data". In: *Neurocomputing* 74.6, pp. 1058–1061.

Ougiaroglou, Stefanos and Georgios Evangelidis (2016). "RHC: a non-parametric cluster-based data reduction for efficient k-NN classification". In: *Pattern Analysis and Applications* 19.1, pp. 93–109.

Özcan, Ender, Burak Bilgin, and Emin Erkan Korkmaz (2008). "A comprehensive analysis of hyper-heuristics". In: *Intelligent Data Analysis* 12.1, pp. 3–23.

Pelikan, Martin and Kumara Sastry (2004). "Fitness inheritance in the Bayesian optimization algorithm". In: *Genetic and Evolutionary Computation Conference*. Springer, pp. 48–59.

Poikolainen, Ilpo, Ferrante Neri, and Fabio Caraffini (2015). "Cluster-based population initialization for differential evolution frameworks". In: *Information Sciences* 297, pp. 216–235.

Polyzotis, Neoklis et al. (2018). "Data lifecycle challenges in production machine learning: a survey". In: *ACM SIGMOD Record* 47.2, pp. 17–28.

Raja, Rohit et al. (2022). *Data Mining and Machine Learning Applications*. John Wiley & Sons.

Ramentol, Enislay et al. (2012). "SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory". In: *Knowledge and Information Systems* 33.2, pp. 245–265.

Ramírez-Gallego, Sergio et al. (2018). "Big Data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce". In: *Information Fusion* 42, pp. 51–61.

Refaeilzadeh, Payam, Lei Tang, and Huan Liu (2009). "Cross-Validation". In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, pp. 532–538.

Reina, Daniel G et al. (2016). "A survey on the application of evolutionary algorithms for mobile multihop ad hoc network optimization problems". In: *International Journal of Distributed Sensor Networks* 12.2, p. 2082496.

Rijn, Sander van et al. (2015). "Optimizing Highly Constrained Truck Loadings Using a Self-Adaptive Genetic Algorithm." In: *CEC*, pp. 227–234.

Rokach, Lior (2010). "Ensemble-based Classifiers". In: *Artificial Intelligence Review* 33.1, pp. 1–39.

Ros, Raymond and Nikolaus Hansen (2008). "A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity". In: *Parallel Problem Solving from Nature – PPSN X*. Ed. by Günter Rudolph et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 296–305.

Rosales-Pérez, Alejandro et al. (2015). "Surrogate-assisted multi-objective model selection for support vector machines". In: *Neurocomputing* 150, pp. 163 –172. ISSN: 0925-2312.

Ross, Peter, Dave Corne, and Hsiao-Lan Fang (1994). "Improving evolutionary timetabling with delta evaluation and directed mutation". In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 556–565.

Ruder, Sebastian (2016). "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747*.

Runkler, Thomas A (2020). *Data analytics*. Springer.

Saha, Soumitra et al. (2022). "Cluster-Oriented Instance Selection for Classification Problems". In: *Information Sciences*.

Salami, Mehrdad and Tim Hendtlass (2003). "A fast evaluation strategy for evolutionary algorithms". In: *Applied Soft Computing* 2.3, pp. 156–173.

Sánchez, José Salvador (2004). "High training set size reduction by space partitioning and prototype abstraction". In: *Pattern Recognition* 37.7, pp. 1561–1564.

Sastry, Kumara, David E Goldberg, and Martin Pelikan (2001). "Don't evaluate, inherit". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., pp. 551–558.

Schwartz, R et al. (2019). "Green AI; 2019". In: *arXiv preprint arXiv:1907.10597*.

Sharma, Prinkle and Hong Liu (2020). "A machine-learning-based data-centric misbehavior detection model for internet of vehicles". In: *IEEE Internet of Things Journal* 8.6, pp. 4991–4999.

Shi, L and K Rasheed (2010). "A survey of fitness approximation methods applied in evolutionary algorithms". In: *Computational intelligence in expensive optimization problems*. Springer, pp. 3–28.

Silver, Edward Allen (2004). "An overview of heuristic solution methods". In: *Journal of the operational research society* 55.9, pp. 936–956.

Sindhya, Karthik (n.d.). "An Introduction to Evolutionary Algorithms". In: *Industrial Optimization Group, Department of Mathematical Information Technology, University of Tyvaskyla* ().

Smith, Robert E, Bruce A Dike, and SA Stegmann (1995). "Fitness inheritance in genetic algorithms". In: *Proceedings of the 1995 ACM Symposium on Applied computing*. ACM, pp. 345–350.

Smith, Robert E and Ellen Smuda (1993). "Adaptively resizing populations: Algorithm, analysis, and first results". In.

Song, H., I. Triguero, and E. Ozcan (2019). "A review on the self and dual interactions between machine learning and optimisation." In: *Progress in Artificial Intelligence* 8, 143–165.

Sörensen, K. and F. W. Glover (2013). "Metaheuristics". In: *Encyclopedia of Operations Research and Management Science*. Springer, pp. 960–970.

Srisawat, Anantaporn, Tanasanee Phienthrakul, and Boonserm Kijsirikul (2006). "SV-kNNC: An algorithm for improving the efficiency of k-nearest neighbor". In: *Pacific rim international conference on artificial intelligence*. Springer, pp. 975–979.

Stojanovic, Vladimir, Shuping He, and Baoyong Zhang (2020). "State and parameter joint estimation of linear stochastic systems in presence of faults and non-Gaussian noises". In: *International Journal of Robust and Nonlinear Control* 30, pp. 6683–6700.

Streichert, Felix et al. (2003). "A clustering based niching method for evolutionary algorithms". In: *Genetic and Evolutionary Computation Conference*. Springer, pp. 644–645.

Sun, Bo et al. (2018). "Evolutionary under-sampling based bagging ensemble method for imbalanced data classification". In: *Frontiers of Computer Science* 12.2, pp. 331–350.

Sun, Y. et al. (2019). "Surrogate-Assisted Evolutionary Deep Learning Using an End-to-End Random Forest-based Performance Predictor". In: *IEEE Transactions on Evolutionary Computation*, pp. 1–1. ISSN: 1941-0026.

Sundar, R and M Punniyamoorthy (2019). "Performance enhanced Boosted SVM for Imbalanced datasets". In: *Applied Soft Computing*, p. 105601.

Taleb, Ikbal, Mohamed Adel Serhani, and Rachida Dssouli (2018). "Big data quality: A survey". In: *2018 IEEE International Congress on Big Data (BigData Congress)*. IEEE, pp. 166–173.

Taleb, Ikbal et al. (2016). "Big data quality: A quality dimensions evaluation". In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP /SmartWorld)*. IEEE, pp. 759–765.

Tanabe, Ryoji and Alex Fukunaga (2013). "Success-history Based Parameter Adaptation for Differential Evolution". In: *2013 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 71–78.

Tanabe, Ryoji and Alex S Fukunaga (2014). "Improving the Search Performance of SHADE Using Linear Population Size Reduction". In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1658–1665.

Tao, Hongfeng et al. (2020). "An unsupervised fault diagnosis method for rolling bearing using STFT and generative neural networks". In: *Journal of the Franklin Institute* 357, pp. 7286–7307.

Thabtah, Fadi et al. (2020). "Data imbalance in classification: Experimental evaluation". In: *Information Sciences* 513, pp. 429 –441. ISSN: 0020-0255.

Ting, Chuan-Kang et al. (2018). "Mining Fuzzy Association Rules Using a Memetic Algorithm Based on Structure Representation". In: *Memetic Computing* 10.1, pp. 15–28.

Triguero, I., S. García, and F. Herrera (2010). "A Preliminary Study on the Use of Differential Evolution for Adjusting the Position of Examples in Nearest Neighbor Classification". In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1–8.

Triguero, I., S. García, and F. Herrera (2011). "Differential Evolution for Optimizing the Positioning of Prototypes in Nearest Neighbor Classification". In: *Pattern Recognition* 44.4, pp. 901–916.

Triguero, I. et al. (2012). "A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification". In: *IEEE Transactions on Systems, Man, and Cybernetics–Part C* 42.1, pp. 86–100.

Triguero, I. et al. (2014). "A combined MapReduce-windowing Two-level Parallel Scheme for Evolutionary Prototype Generation". In: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 3036–3043.

Triguero, I. et al. (2015a). "Evolutionary Undersampling for Imbalanced Big Data Classification". In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 715–722.

Triguero, I. et al. (2015b). "MRPR: A MapReduce Solution for Prototype Reduction in Big Data Classification". In: *NeuroComputing* 150, pp. 331–345. ISSN: 0925-2312.

Triguero, I. et al. (2017). "A first attempt on global evolutionary undersampling for imbalanced big data". In: *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2054–2061.

Triguero, I. et al. (2017). "KEEL 3.0: an open source software for multi-stage analysis in data mining". In: *International Journal of Computational Intelligence Systems* 10, pp. 1238–1249.

Triguero, I. et al. (2019). "Transforming Big Data into Smart Data: An Insight on the Use of the k-nearest Neighbors Algorithm to Obtain Quality Data". In: *WIREs Data Mining and Knowledge Discovery* 9.2, p. 1289.

Tseng, Lin-Yu and Chun Chen (2008). "Multiple trajectory search for Large Scale Global Optimization". In: *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 3052–3059.

Vallejo, Carlos G, José A Troyano, and F Javier Ortega (2010). "InstanceRank: Bringing order to datasets". In: *Pattern recognition letters* 31.2, pp. 133–142.

Večerík, Matej et al. (2017). "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards". In: *arXiv preprint arXiv:1707.08817*.

Wang, Jin et al. (2020). "Big data service architecture: a survey". In: *Journal of Internet Technology* 21.2, pp. 393–405.

Watkins, C. and P. Dayan (1992). "Q-learning". In: *Machine Learning* 8.3-4, pp. 279–292.

Wilcoxon, Frank (1945). "Individual comparisons by ranking methods". In: *Biometrics Bulletin* 1.6, pp. 80–83.

Wilson, D Randall and Tony R Martinez (2000). "Reduction Techniques for Instance-Based Learning Algorithms". In: *Machine Learning* 38.3, pp. 257–286.

Wilson, Dennis L (1972). "Asymptotic properties of nearest neighbor rules using edited data". In: *IEEE Transactions on Systems, Man, and Cybernetics* 3, pp. 408–421.

Witten, I. H. and E. Frank (2017). *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier.

Wong, Ka-Chun (2017). "Evolutionary algorithms: Concepts, designs, and applications in bioinformatics". In: *Nature-Inspired Computing: Concepts, Methodologies, Tools, and Applications*. IGI Global, pp. 111–137.

Xu, Dongkuan and Yingjie Tian (2015). "A comprehensive survey of clustering algorithms". In: *Annals of Data Science* 2.2, pp. 165–193.

Xu, Rui and Donald Wunsch (2005). "Survey of clustering algorithms". In: *IEEE Transactions on neural networks* 16.3, pp. 645–678.

Yao, Xin, Yong Liu, and Guangming Lin (1999). "Evolutionary Programming Made Faster". In: *IEEE Transactions on Evolutionary Computation* 3.2 (2), pp. 82–102.

Yen, Chen-Wen, Chieh-Neng Young, and Mark L Nagurka (2004). "A vector quantization method for nearest neighbor classifier design". In: *Pattern Recognition Letters* 25.6, pp. 725–731.

Yen, Show-Jane and Yue-Shi Lee (2009). "Cluster-based under-sampling approaches for imbalanced data distributions". In: *Expert Systems with Applications* 36.3, pp. 5718–5727.

Yusta, Silvia Casado (2009). "Different metaheuristic strategies to solve the feature selection problem". In: *Pattern Recognition Letters* 30.5, pp. 525–534.

Zadrozny, Bianca and Charles Elkan (2001). "Learning and making decisions when costs and probabilities are both unknown". In: *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 204–213.

Zaharia, M et al. (2012). "A Fault-Tolerant Abstraction for In-Memory Cluster Computing". In: *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12)*.

Zaher, Amer Abu et al. (2019). "An Adaptive Memetic Algorithm for Feature Selection Using Proximity Graphs". In: *Computational Intelligence* 35.1, pp. 156–183.

Zebari, Rizgar et al. (2020). "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction". In: *Journal of Applied Science and Technology Trends* 1.2, pp. 56–70.

Zhan, Zhi-Hui et al. (2009). "Adaptive particle swarm optimization". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.6, pp. 1362–1381.

Zhang, Huaxiang and Jing Lu (2008). "Adaptive evolutionary programming based on reinforcement learning". In: *Information Sciences* 178.4, pp. 971–984.

Zhang, Jingqiao and Arthur C Sanderson (2009). "JADE: adaptive differential evolution with optional external archive". In: *IEEE Transactions on Evolutionary Computation* 13.5, pp. 945–958.

Zhang, Jun, Henry Shu-Hung Chung, and Wai-Lun Lo (2007). "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms". In: *IEEE Transactions on Evolutionary Computation* 11.3, pp. 326–335.

Zhang, Jun et al. (2011). "Evolutionary computation meets machine learning: A survey". In: *IEEE Computational Intelligence Magazine* 6.4, pp. 68–75.

Zhu, Bing, Bart Baesens, and Seppe KLM vanden Broucke (2017). "An empirical comparison of techniques for the class imbalance problem in churn prediction". In: *Information Sciences* 408, pp. 84–99.

Zhu, Z., S. Jia, and Z. Ji (2010). "Towards a Memetic Feature Selection Paradigm [Application Notes]". In: *IEEE Computational Intelligence Magazine* 5.2, pp. 41–53.