THE POPULATION INITIALIZATION AFFECTS THE PERFORMANCE OF SUBGROUP DISCOVERY EVOLUTIONARY ALGORITHMS IN HIGH DIMENSIONAL DATASETS

By

VÍTOR DE ALBUQUERQUE TORREÃO

M.Sc. Dissertation

RECIFE/2019

Vítor de Albuquerque Torreão

THE POPULATION INITIALIZATION AFFECTS THE PERFORMANCE OF SUBGROUP DISCOVERY EVOLUTIONARY ALGORITHMS IN HIGH DIMENSIONAL DATASETS

A M.Sc. Dissertation presented to the Centro de Informática of Universidade Federal de Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Advisor: Renato Vimieiro

RECIFE
2019

Vítor de Albuquerque Torreão

The population initialization affects the performance of subgroup discovery evolutionary algorithms in high dimensional datasets/ Vítor de Albuquerque Torreão. – RECIFE, 2019-

64 p. : il. (algumas color.) ; 30 cm.

Advisor Renato Vimieiro

M.Sc. Dissertation – Universidade Federal de Pernambuco, 2019.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

Dissertação de mestrado apresentada por **Vítor de Albuquerque Torreão** ao programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **The population initialization affects the performance of subgroup discovery evolutionary algorithms in high dimensional datasets**, orientada pelo **Prof. Renato Vimieiro** e aprovada pela banca examinadora formada pelos professores:

RECIFE
2019

I dedicate this dissertation to all my family, friends and professors who gave me the necessary support to get here.

Acknowledgements

I would like to thank my parents, Roberto and Cláudia Torreão, for having invested all they got in me, and for teaching from a early age to value my education. Also, thanks to my sister, Maria da Graça, for her super powers of always leaving me more confident than she found.

Thanks to my beloved Marília Sabino, who endured my anxiety and insecurities throughout my undergrad and now grad school, and specially while I was working on this text.

My most sincere gratitude goes to the amazing professors, at both computer science departments of Universidade Federal Rural de Pernambuco and Universidade Federal de Pernambuco, for all the knowledge shared throughout the years.

I would also like to thank my colleagues at Acqio Payments for believing in me, for having patience with me and helping me conciliate work and research during these last two years.

Last but not least, I thank God for the calm and tranquility bestowed upon me when I needed most.

A learning experience is one of those things that says,
"You know that thing you just did? Don't do that."

—DOUGLAS ADAMS

Resumo

Descoberta de Conhecimento em Bases de Dados (KDD) é uma área ampla em Inteligência Artificial que se preocupa com a extração de informações e *insights* úteis a partir de um conjunto de dados. Dentre as diferentes metodologias de extração, uma importante subclasse de tarefas de KDD, chamada de Descoberta de Subgrupos (SD), lida com a descoberta de subconjuntos interessantes dentro dos dados.

Vários Algoritmos Evolucionários (EAs) foram propostos para resolver a tarefa de descobrir subgrupos com sucesso considerável em bases de dados de baixa dimensionalidade. A literatura já mostrou, no entanto, que alguns desses tem uma performance baixa em problemas de alta dimensionalidade. O algoritmo evolucionário para descoberta de subgrupos com, atualmente, a melhor performance em bases de alta dimensionalidade, SSDP, possui uma forma peculiar de inicializar sua população, limitando os indivíduos ao menor tamanho possível.

Assim como na maioria das técnicas baseadas em população, o resultado de um algoritmo evolucionário é, em geral, dependente do conjunto de soluções inicial, que é tipicamente gerado de forma aleatória. Escolher uma técnica de inicialização sob outra tem grande impacto na solução final apresentada, e este já foi o tópico de trabalhos publicados na área de computação evolucionária. Apesar disso, faltam trabalhos que estudem este tópico no caso específico de descoberta de subgrupos, especialmente quando são consideradas bases de alta dimensionalidade.

O objetivo final desta pesquisa é avaliar o impacto da geração da população inicial no resultado final de um algoritmo evolucionário no contexto de uma tarefa de descoberta de subgrupos em dados de alta dimensionalidade. Especificamente, são apresentados novos métodos de inicialização, projetados para as características específicas de tarefas de descoberta de subgrupos, que podem ser utilizadas em praticamente qualquer algoritmo evolucionário. Os experimentos conduzidos mostram que mudar o método de inicialização é o suficiente para aumentar a performance de algoritmos evolucionários do estado da arte em bases de dados de alta dimensionalidade.

Palavras-chave: Data Mining, Knowledge Discovery, Subgroup Discovery, Evolutionary Algorithms

Abstract

Knowledge Discovery in Databases (KDD) is a broad area in Artificial Intelligence concerned with the extraction of useful information and insights from a given dataset. Among the distinct extraction methodologies, an important subclass of Knowledge Discovery in Databases (KDD) tasks, called Subgroup Discovery (SD), undertakes the discovery of interesting subsets in the data.

Many Evolutionary Algorithms (EAs) have been proposed to solve the Subgroup Discovery task with considerable success in low dimensional datasets. Some of these, however, have been shown to perform poorly in high dimensional problems. The currently best performing Evolutionary Algorithm for Subgroup Discovery in high dimensional datasets, SSDP, has a peculiar way of initializing its populations, limiting the individuals to the smallest possible size.

As with most population-based techniques, the outcome of an Evolutionary Algorithm is usually dependent on the initial set of solutions, which are typically generated at random. The impact of choosing one initialization technique over another in the final presented solution has been the topic of many published works in the broad area of evolutionary computation. Despite this, there is still a lack of studies which approach this topic in the specific scenario of Subgroup Discovery tasks, especially when considering high dimensional datasets.

The ultimate goal of this research project is to evaluate the impact of initial population generation in the end result of the overall Evolutionary Algorithm used to solve a Subgroup Discovery task in high dimensional data. Specifically, we provide new initialization methods, designed for the specific characteristics of Subgroup Discovery tasks, which can be used in virtually any EA. Our conducted experiments show that, by just changing the initialization method, state of the art Evolutionary Algorithms have their performance increased in high dimensional datasets.

Keywords: Data Mining, Knowledge Discovery, Subgroup Discovery, Evolutionary Algorithms

List of Figures

List of Tables

List of Algorithms

List of Acronyms

BRACIS	Brazilian Conference on Intelligent Systems	40
DNF	Disjunctive Normal Form	28
EA	Evolutionary Algorithm	28
GA	Genetic Algorithm	28
KDD	Knowledge Discovery in Databases	15
MESDIF	Multi-objective Evolutionary Subgroup Discovery Fuzzy rules	28
NMEEF-SD	Non-dominated Multi-objective Evolutionary Algorithm Based on the Extraction of Fuzzy Rules for Subgroup Discovery	30
PDF	Probability Density Function	41
PRNG	Pseudo-Random Number Generator	36
RNG	Random Number Generator	36
SD	Subgroup Discovery	28
SSDP	Simple Search Discriminative Patterns	33
WRAcc	Weighted Relative Accuracy	30

Contents

1

Introduction

Data Mining is a broad area in Artificial Intelligence concerned with the extraction of novel, useful, and interesting patterns from potentially large datasets. Also known as Knowledge Discovery in Databases (KDD), its techniques can be divided into predictive and descriptive induction. The first is about extracting knowledge in order to predict the class label of unseen data, whereas the second is concerned with the search for interesting relationships between features of data.

Subgroup Discovery (SD) (WROBEL, 1997) is a subclass of Knowledge Discovery tasks classified as a descriptive data mining technique based on supervised learning. As the name suggests, its main objective is the search for statistically interesting subgroups in the dataset. A subgroup is a subset of objects in the dataset, and it is considered “interesting” if it is as large as possible and has unusual distributional characteristics with respect to a certain target property.

Subgroups must be described in explicit symbolic form and be humanly interpretable in order to guide the decision making process of some domain expert, e.g., doctors and software engineers (LAVRAČ, 2005). To that end, subgroups are typically represented as propositional rules, and their quality is assessed using measures from predictive rule learning (such as accuracy, sensitivity and specificity) and association rule learning (such as coverage and support) (LAVRAČ; FLACH; ZUPAN, 1999).

The first algorithms proposed to mine these subgroups were EXPLORA (KLÖSGEN, 1996) and MIDOS (WROBEL, 1997). These are known as exhaustive methods, meaning that they would go through the entire search space in order to find the exact optimal solution. As the datasets grew in number of attributes and objects, it became increasingly infeasible to search the complete space of solutions, and so researchers turned their attention to heuristic approaches.

It is possible to reduce subgroup discovery to a general search or optimization task, where a single rule is a valid solution, the set of all possible rules is the search space and the quality of any given solution can be calculated using one (or more) of the previously mentioned measures. That way, any general purpose search or optimization algorithm can be applied to solve SD.

Therefore, many heuristic search based approaches have been proposed to solve SD. The most common approaches can be divided into two categories: the ones based on beam search, and Evolutionary Algorithms (EAs). In the first category, we find algorithms such as SubgroupMiner (KLÖSGEN; MAY, 2002) and SD (GAMBERGER; LAVRAČ, 2002). In the second category, the focus of this research project, there are algorithms such as Non-dominated Multi-objective Evolutionary Algorithm Based on the Extraction of Fuzzy Rules for Subgroup Discovery (NMEEF-SD) (CARMONA et al., 2009), Multi-objective Evolutionary Subgroup Discovery Fuzzy rules (MESDIF) (JESUS; GONZALEZ; HERRERA, 2007) and Simple Search Discriminative Patterns (SSDP) (PONTES; VIMIEIRO; LUDERMIR, 2016).

EAs have been frequently applied to solve SD (CARMONA et al., 2014). However, even though they have had success in low dimensional problems, most of these EAs have poor performance in high dimensionality (LUCAS et al., 2017).

One of the common steps in all EAs is the creation of the starting population. This step provides an initial set of "guess" solutions, which will be iteratively improved during the optimization process. Good initial guesses can help the EA locate the optima, whereas bad initial guesses may even prevent EAs from finding it. There have been many discussions on the effect of population initialization in the outcome of general EA solutions (KAZIMIPOUR; LI; QIN, 2013, 2014a,b). However, to the best of our knowledge, there still lacks a work on such effects in the specific scenario of high dimensional SD problems.

The algorithm with the best results in high dimensional SD problems, SSDP, has a very peculiar way to initialize its population. This raises the question as whether it is possible to improve the performance of state of the art EAs, which perform poorly in high dimensionality, by just changing their initialization strategies.

Therefore, this study aims at starting the discussion about the impacts of selected population initialization strategies over the effectiveness of an EA for solving SD problems, especially in high dimensional datasets. In the scope of this dissertation, by "high dimensional", we mean to include datasets ranging from a few thousand, to tens of thousands of features.

1.1 Motivation

Subgroup Discovery is a general and broadly applicable technique for descriptive and exploratory data mining. As such, it possesses a wide range of real world applications.

LAVRAČ (2005) used SD in two case studies. In the medical domain, SD can be used to detect and describe Coronary Heart Disease risk groups. The available data collected includes physical examination and laboratory tests. In most cases with significant pathological symptoms, it is easy to detect the disease. To help prevent the disease, however, it is essential to be able to find endangered individuals with slightly abnormal symptoms. SD can help detect important risk factors and groups in the population.

The case study involved gene expression data, which is obtained through DNA microarrays, and help understand many biological processes. This DNA chip-based technology is able to measure the expression levels of thousands of genes, which produces a large amount of valuable high-dimensional data. Here, SD can be used to uncover interesting patterns which can help experts achieve a better understanding of the dependencies between diseases and gene expression values in a person's tissues (LIU et al., 2014).

LI; WONG (2002) have shown that the identification of groups of genes that are constrained to certain intervals of expression levels, in a way that these patterns only occur in a certain class of cells, but not in others, can help identify the best pharmacological intervention to treat a disease. For instance, SD could help assign patients with Acute Lymphoblastic Leukemia to specific risk groups, which is critical to tailor the intensity of the chemotherapy to treat them (YEOH et al., 2002).

Outside the bioinformatics and medical domains, RODRÍGUEZ et al. (2012) approached the problem of detecting defective software modules through a descriptive induction process using SD. The authors used an EA to generate understandable and useful rules, which could help project managers and quality assurance professionals to guide the testing effort and, ultimately, improve software quality.

JESUS et al. (2007), on the other hand, proposed their own genetic algorithm for SD to extract relevant and interesting information to help improve planning polices for trade fairs, which are basic instruments in industrial marketing.

As one can see, the varied possible applications of SD in real world problems make this data mining technique relevant. The most common approaches to solving SD problems are based on exhaustive or beam search. But because of these methods' performance in large-scale datasets, other methods based in soft computing techniques were investigated and applied. Among these, a common trend is to use Evolutionary Algorithms. When applied to high dimensional SD problems, however, the state of the art EAs for SD have been shown to either perform poorly or not at all (LUCAS et al., 2017). Such a poor performance raises the question of whether this behavior is due to the initialization procedure adopted in those heuristics, since most of the state of the art algorithms are straightforward adaptations of well-known optimization methods.

1.2 Problem Statement

Usually, EAs generate the starting population completely at random. For low dimensional problems that is, generally, not a problem, but for high dimensional datasets, those strategies generate very long propositional rules, which inevitably have poor (and very often zero) coverage and support. In contrast, the currently best performing EA in high dimensional datasets, designed specifically for that characteristic, SSDP (PONTES; VIMIEIRO; LUDERMIR, 2016), uses a very peculiar initialization method which restraints the starting population to rules of size 1. This raises the question as to whether other EAs could be led to perform better, in such high dimensional SD tasks, if their initialization operators were changed to, for example, prioritize rules of small sizes.

Based on this context, the main research question investigated by this dissertation is:

Research question *Is it possible to improve the performance of state of the art EAs, which thrive in low dimensional problems, but struggle in high dimensional ones, by just changing their initialization operators?*

Aiming to answer the above question, this dissertation presents new ways to initialize the starting population of EAs for SD problems, modifies the already existing state of the art EAs to use those operators and shows, through experiments, that these modified versions have superior performance in high dimensional SD problems.

1.3 Organization of the Dissertation

Chapter 2 reviews SD by giving a set theoretic definition of the problem. It also provides a description of many quality measures which can be used to evaluate subgroups. It finishes by reviewing many of the published approaches to discover subgroups, giving special attention to the evolutionary algorithms.

Chapter 3 describes in details the evolutionary algorithms used throughout the experiments conducted to test the hypothesis. In particular, it reviews how each algorithm initializes the population in order to motivate our proposed technique.

Chapter 4 presents the empirical tests designed to test our hypothesis in terms of convergence. We present each algorithm's convergence rate and then compare it with a new version where their initializations have been replaced by our proposed approach.

Chapter 5 focuses on comparing the algorithms in terms of the quality of output rules. It also presents new ways to initialize the population in an attempt to improve the results. The chapter finishes with a comparison of the various presented ways to initialize the population.

Finally, this document presents its final conclusions and results in Chapter 6. It is also in that chapter where we present some research questions that were left unanswered and will be addressed in our future works.

2

Background

This chapter reviews the specialized literature for Subgroup Discovery by first reviewing its definition, then the measures used to assert a subgroup's quality and finally describing existing approaches for discovering subgroups found in the literature.

Data mining is an area in Artificial Intelligence which includes many techniques, methods, technologies and systems. In particular, an important subclass of knowledge discovery tasks is the automated discovery of interesting subgroups in populations, also known as, Subgroup Discovery (SD). It was first defined by KLÖSGEN (1996) and WROBEL (1997) as: given a population of objects, described by a set of properties, and a specific property which we are interested in, the task is to discover subgroups of the population that are statistically most interesting with respect to that property of interest.

More formally, let $\boldsymbol{D} = \{\boldsymbol{E},\ \boldsymbol{A}\}$ be a dataset consisting of $\boldsymbol{E}$, which is the set of objects, and $\boldsymbol{A}$, the set of attributes. Then, we say the dataset has $n = |\boldsymbol{E}|$ objects, each described by $m = |\boldsymbol{A}|$ attributes, and so $\boldsymbol{D}$ is said to be a m-dimensional dataset.

Each attribute, $\boldsymbol{a}_j \in \boldsymbol{A}$ (for $j \in \{1,...,m\}$), is also a set: $\boldsymbol{a}_j = \{v_1^j,...,v_{k_j}^j\}$, consisting of $k_j = |\boldsymbol{a}_j|$ possible values. Each object $\boldsymbol{e}_i \in \boldsymbol{E}$ (for $i \in \{1,...,n\}$), is a set of m attribute-value pairs of the form: $\boldsymbol{e}_i = \{\langle 1, v_p^1\rangle,...,\langle m, v_q^m\rangle\}$, where the first element of each pair is the index of an attribute ($\boldsymbol{a}_1,...,\boldsymbol{a}_m$), and the second element is one of the possible values of that attribute, i.e., $v_p^1 \in \boldsymbol{a}_1$, where $p \in \{1,...,k_1\}$ and $v_q^m \in \boldsymbol{a}_m$, where $q \in \{1,...,k_m\}$.

This definition of a dataset does not contemplate continuous attributes, but it is simple and sufficient enough to formally define the task of SD and to mathematically describe the quality measures presented in Section 2.1.

Given a target attribute, $\boldsymbol{a}_T \in \boldsymbol{A}$, and one of its values in which we are interested in, $v_c^T \in \boldsymbol{a}_T$, possibly a class label, we can draw a subgroup $\boldsymbol{S}_R \subset \boldsymbol{E}$ induced by a rule, R, of the form:

$$R : Cond \rightarrow v_c^T \tag{2.1}$$

The antecedent part of (2.1), *Cond*, can be any boolean expression involving any number of attribute-value pairs. An object in the dataset, $\boldsymbol{e}_i$, is said to be covered by R if the expression in *Cond* is true for that object. An object covered by R is an element of the subgroup $\boldsymbol{S}_R$. As mentioned above, the consequent part of (2.1), v_c^T, is some value for the target attribute in which we are interested in. Usually, examples satisfying this target value are called "positive examples", while the remaining are known as "negative examples", since $\boldsymbol{a_T}$ is often a class label.

If *Cond* is true for that dataset object and the value in $\boldsymbol{e_i}$ for the target attribute is v_c^T, then $\boldsymbol{e_i}$ is called a "true positive". Whereas if it is any other value, then $\boldsymbol{e_i}$ is called a "false positive". We can subdivide $\boldsymbol{S_R}$ in terms of v_c^T: let $\boldsymbol{S_R^+} = \{\boldsymbol{e_i} \in \boldsymbol{S_R} \mid \langle T, v_c^T \rangle \in \boldsymbol{e_i}\}$ be the set of true positives and $\boldsymbol{S_R^-} = \boldsymbol{S_R} \setminus \boldsymbol{S_R^+}$ be the set of false positives. True and false positives will be used to calculate measures for the quality of subgroups, which are presented in the next section.

It is also possible to subdivide $\boldsymbol{E}$ in a similar fashion: let $\boldsymbol{E_c^+} = \{\boldsymbol{e_i} \in \boldsymbol{E} \mid \langle T, v_c^T \rangle \in \boldsymbol{e_i}\}$ be the set of positive objects in the dataset and $\boldsymbol{E_c^-} = \boldsymbol{E} \setminus \boldsymbol{E_c^+}$ be the set of negative objects in the dataset.

To illustrate our definitions, we now present an example. Suppose we want to discover subgroups in the toy dataset provided in Table 2.1, which was adapted from LUCAS et al. (2017).

Table 2.1: A toy example. In this dataset, the objective is to identify the characteristics which render a medical treatment a success or a failure.

Example	Gender	Age	Medicine	Label
e_1	M	Senior	B	Success
e_2	F	Senior	B	Success
e_3	M	Senior	A	Success
e_4	M	Adult	A	Success
e_5	F	Child	A	Success
e_6	F	Child	A	Failure
e_7	M	Child	B	Failure
e_8	F	Child	B	Failure
e_9	M	Adult	A	Failure
e_{10}	F	Adult	A	Failure

We want to find subgroups in the dataset which describe successful medical treatments for some disease. Following our established notation, $\boldsymbol{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{10}\}$ is the set of objects. $\boldsymbol{A}$, the set of attributes is $\{\boldsymbol{a}_{gender}, \boldsymbol{a}_{age}, \boldsymbol{a}_{medicine}, \boldsymbol{a}_{label}\}$, where $\boldsymbol{a}_{geder} = \{\text{M,F}\}$, $\boldsymbol{a}_{age} = \{\text{Senior, Adult, Child}\}$, $\boldsymbol{a}_{medicine} = \{\text{A, B}\}$ and $\boldsymbol{a}_{label} = \{\text{Success, Failure}\}$. So $\boldsymbol{D}$, represented in our table, is $\{\boldsymbol{E}, \boldsymbol{A}\}$. Moreover, the set of positive objects, $\boldsymbol{E}^+_{success}$, is given by $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_5\}$, while the set of negative objects, $\boldsymbol{E}^-_{success}$, is given by $\{\boldsymbol{e}_6, \ldots, \boldsymbol{e}_{10}\}$.

In this scenario, an example of a rule, $R1$, might be:

$$R1 = \{\langle age, v^{age}_{senior} \rangle\} \rightarrow v^{label}_{Success} \tag{2.2}$$

So the subgroup $S_{R1} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$, induced by rule $R1$, is an interesting subgroup, since the set of true positives is $S^+_{R1} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$, while the set of false positives is $S^-_{R1} = \emptyset$. That is, the frequency of the target class value in the subgroup is 100%, while the frequency of other class values is 0%.

The expression *Cond* can be built using any boolean operator, but conjunctions are more commonly used. See the example below:

$$R2 = \{\langle gender, v_{\mathrm{M}}^{gender}\rangle \wedge \langle medicine, v_{\mathrm{B}}^{medicine}\rangle\} \rightarrow v_{Success}^{label} \tag{2.3}$$

From $R2$, we induce the following subgroup: $S_{R2} = \{\boldsymbol{e}_1,\ \boldsymbol{e}_7\}$, which is not an interesting subgroup, because the frequency of objects in the target class value is the same as the other target class values: $S_{R2}^{+} = \{\boldsymbol{e}_1\}$ and $S_{R2}^{-} = \{\boldsymbol{e}_7\}$.

That way, it is possible to arbitrarily define subgroups for any given dataset simply by concatenating attribute-value pairs using some boolean operator. However, the SD task is concerned not with just any subgroup, but with "interesting" ones. To calculate the interestingness of a subgroup, it is common to use evaluation measures applied over the rules that define them.

2.1 Evaluation Measures

There are many ways to evaluate the subgroups. These measures are used both by SD strategies, to refine the final set of returned subgroups, and by SD researchers to compare different approaches. Typically, measures are applied locally over the rules which induce the subgroups, but some global measures evaluate an entire set of rules returned by a strategy's execution.

In the case of local measures, it is possible to obtain an evaluation of a strategy's entire execution by averaging the results of specific rules.

This section is aimed at reviewing the most used evaluation measures found in the literature. To that end, measures are grouped into "measures of complexity", "measures of generality", "measures of precision" and "hybrid measures", following the definition of HERRERA et al. (2011).

2.1.1 Measures of Complexity

The first set of measures that we review regard the understandability of the knowledge extracted from a successful execution of some SD approach. These are called measures of complexity.

The first is the Number of Subgroups discovered by a single execution of some SD algorithm. In particular, many approaches to SD are called *top-k* methods, which will always return a number of subgroups specified by the end user beforehand.

The second measure of complexity is applied to each individual rule. The Length of the Rule is simply the number of attribute-value pairs contained in it. The higher the number, the more complex a rule is. Rules with too many attribute-value pairs are typically harder to read even by domain experts.

2.1.2 Measures of Generality

Measures of generality are used to quantify how general a rule is in terms of how many examples of interest are covered. The two most noticeable measures in this category are Coverage and Support (LAVRAČ; FLACH; ZUPAN, 1999).

Coverage measures the fraction of examples in the dataset covered by the antecedent part of the rule. As such, it is calculated as:

$$Cov(R) = \frac{|\boldsymbol{S}_R|}{|\boldsymbol{E}|} \tag{2.4}$$

The Support of a rule is the fraction of examples covered by both the antecedent and consequent parts of the rule. It can be calculated by:

$$Supp(R) = \frac{|\boldsymbol{S}_R^+|}{|\boldsymbol{E}|} \tag{2.5}$$

2.1.3 Measures of Precision

The confidence (LAVRAČ; FLACH; ZUPAN, 1999) of a rule is the relative frequency of true positive objects in the subgroup induced by it, which can be mathematically defined as:

$$Cnf(R) = \frac{|\boldsymbol{S}_R^+|}{|\boldsymbol{S}_R|} \tag{2.6}$$

Another measure of precision is the Q_g metric (GAMBERGER; LAVRAČ, 2002), which measures the trade-off between the number of true positive examples and the unusualness of their distribution. It can be computed as:

$$Q_g(R,g) = \frac{|\boldsymbol{S}_R^+|}{|\boldsymbol{S}_R^-| + g} \tag{2.7}$$

Where g is a generalization parameter and is usually configured to a value between 0.5 and 100.

Another possible way to measure the precision of a rule is to calculate a straightforward difference between the number of true positives and false positives:

$$\text{SUB}(R) = |\boldsymbol{S}_R^+| - |\boldsymbol{S}_R^-| \tag{2.8}$$

2.1.4 Measures of Interest

This category includes measures aimed at selecting and ranking rules according to their potential interest to the user. Such measures are: Novelty and Significance.

Novelty (LAVRAČ; FLACH; ZUPAN, 1999) is a measure able to detect how unusual a subgroup is. It can be computed by:

$$Nov(R) = Supp(R) - \frac{|\boldsymbol{S}_R|}{|\boldsymbol{E}|} \times \frac{|\boldsymbol{E}_c^+|}{|\boldsymbol{E}|} \tag{2.9}$$

Another measure of interest is Significance (LAVRAČ et al., 2004), which indicates how significant a finding is compared to the null hypothesis of statistical independence. To calculate Significance, we first need to extend our framework.

For a given rule $R : Cond \rightarrow v_c^T$, the actual subgroup is induced by applying the boolean expression, *Cond*, to the objects in the dataset. The consequent part does not affect the induced subgroup, but is essential to calculate the quality, as it contains the target class of the SD task. Therefore, we can write $R_u : Cond \rightarrow v_u^T$, for $u \in \{1, ..., k_T\}$, as rules with the same antecedent as R, but with a different value for the target attribute in the consequent.

With that, the Significance is given by:

$$Sig(R) = 2 \times \sum_{u=1}^{k_T} |\boldsymbol{S}_{R_u}^+| \times log \frac{|\boldsymbol{S}_{R_u}^+|}{|\boldsymbol{E}_u^+| \times Cov(R_u)} \tag{2.10}$$

2.1.5 Hybrid Measures

When searching for subgroups in SD, we typically face a trade-off between complexity, generality, precision and interest. Because of that, evaluation measures that blend these qualities are included in this hybrid category.

The first hybrid measure is the Sensitivity of a rule, defined as the proportion of examples correctly covered. It can be computed as:

$$Sen(R) = \frac{|\boldsymbol{S}_R^+|}{|\boldsymbol{E}_c^+|} \tag{2.11}$$

Another hybrid measure is Specificity, which is the proportion of examples correctly not covered. It is calculated by the following expression:

$$Spec(R) = \frac{|\boldsymbol{E}_c^- \setminus \boldsymbol{S}_R|}{|\boldsymbol{E}_c^-|} \tag{2.12}$$

Finally, we have the Unusualness evaluation metric, also known as Weighted Relative Accuracy (WRAcc) (LAVRAČ; FLACH; ZUPAN, 1999), which is the balance between rule coverage and its accuracy gain. It can be computed as:

$$WRAcc(R) = Cov(R) \times \left(\frac{|\boldsymbol{S}_R^+|}{|\boldsymbol{S}_R|} - \frac{|\boldsymbol{E}_c^+|}{|\boldsymbol{E}|} \right) \tag{2.13}$$

It is important to note that WRAcc is perhaps the most fundamental of all the presented rule evaluation measures and that is due to two main reasons demonstrated by LAVRAČ; FLACH; ZUPAN (1999). First, WRAcc and Novelty are equivalent, i.e., rules with high WRAcc also have high novelty, and *vice versa*. Second, it also provides a trade-off between accuracy and other predictive measures shown above, namely, Sensitivity and Specificity.

In Table 2.2, we present a summary of the quality measures presented in this chapter for the reader's convenience.

Table 2.2: A summary of the quality measures for SD.

Name	**Equation**	**Category**	**Characteristics**
Coverage	2.4	Complexity	Fraction of instances covered by the antecedent
Support	2.5	Complexity	Fraction of true positives covered by the antecedent
Confidence	2.6	Precision	Frequency of true positives among the covered set
Q_g	2.7	Precision	Trade-off between true positives and the unusualness of their distribution
SUB	2.8	Precision	Straightforward difference between true positives and false positives
Novelty	2.9	Interest	Measures how unusual the subgroup is regarding the target value of the class
Significance	2.10	Interest	Measures how unusual the subgroup is regarding all values of the class
Sensitivity	2.11	Hybrid	The true positive rate of a rule
Specificity	2.12	Hybrid	The true negative rate of a rule
WRAcc	2.13	Hybrid	Trade-off between accuracy and Sensitivity/Specificity. Equivalent to novelty

2.2 Existing Approaches

Now that we have reviewed what Subgroup Discovery is, as well as how to evaluate the subgroups found by its algorithms, this section reviews the existing strategies for discovering subgroups found in the literature. We subdivide them into: exhaustive approaches, beam search based strategies, adapted from classification rule learning and evolutionary algorithms.

2.2.1 Exhaustive approaches

The first published approach for Subgroup Discovery, EXPLORA (KLÖSGEN, 1996), is a search algorithm for multidimensional spaces. It explores the space of target subgroups, which are represented as conjunctions of taxonomical values for target variables. The search starts in more general subgroups and moves towards more specific ones. Furthermore, the search is constrained by redundancy filters, pruning successor subgroups off of the search space. The ordering of the exploration can be changed, allowing EXPLORA to follow a breadth-first, depth-first, best-first or even heuristic search strategies.

Following that, the MIDOS (WROBEL, 1997) algorithm is an adaptation of EXPLORA designed to work with multi-relational datasets. As such, it performs a top-down, general-to-specific search which can use breath-first, depth-first or best-first strategies just by changing the selection ordering of hypotheses. During its execution, MIDOS considers the size of the subgroups and their unusualness as evaluation metrics.

SD-MAP (ATZMUELLER; PUPPE, 2006) is an algorithm for SD which adapts the well-known FP-growth method for mining association rules. It is an exhaustive method which implements depth-first search for candidate solution generation and, as such, is guaranteed to find the best subgroups for the given dataset, according to the chosen evaluation metric, which the user can select from a wide variety of options.

2.2.2 Beam search based strategies

SubgroupMiner (KLÖSGEN; MAY, 2002) is an extension of the previous subgroup discovery systems EXPLORA and MIDOS. It uses interactive beam search in the space of possible solutions and uses the same approach as EXPLORA to eliminate redundant subgroups. As for evaluation measures, SubgroupMiner uses the classical binomial test to verify if the frequency of the target class in the subgroup is significantly different than in the entire population.

SD (GAMBERGER; LAVRAČ, 2002) is a heuristic Subgroup Discovery algorithm based on beam search. Instead of using an evaluation metric to automatically select the final set of subgroups to be returned to the user, the objective of SD is to guide a domain expert in making effective searches on a variety of solutions. During its search step, however, it uses the Q_g metric to evaluate partial solutions. The search is performed by keeping a fixed width beam of subgroups and, in each iteration, a conjunction is added to every subgroup. Each new solution generated that way is kept for the next iteration if they are better in terms of the Q_g metric and their relevance. A new subgroup is irrelevant if there exists another subgroup in the new beam such that the true positives of the new subgroup are a subset of the true positives of the old one and false positives of the new are a superset of the false positives in the old.

2.2.3 Approaches adapted from classification rule learning

APRIORI-SD (KAVŠEK; LAVRAČ; JOVANOSKI, 2003) is an adaptation of the classification rule learning algorithm called APRIORI-C. APRIORI-C induces rules such that they are above a established minimal confidence and minimal support. It is often the case that the number of induced rules is too large, so a post-processing step is necessary to find only the best rules. APRIORI-SD uses a modified version of the WRAcc evaluation metric for this step. The weights of each instance decreases each time a rule is found to cover it. When an instance's weight drops below a given threshold, the instance is removed. APRIORI-SD keeps inducing rules until all instances in the dataset have been covered or until it can't induce any more rules.

Another algorithm for SD obtained from the adaptation of a classification rule learning is CN2-SD (LAVRAČ et al., 2004), which adapts the classical algorithm CN2. The later originally uses the covering algorithm for constructing the set of rules, where the first constructed rules are induced from the entire set of instances from the dataset, but subsequent ones are induced from a biased example subset, which includes only the positive examples in the dataset that were not covered by the first rules. CN2-SD modifies this approach by instead lowering the probability of covering non-positive, already covered examples instead of excluding them. It also uses the modified version of WRAcc as evaluation measures.

Most of the algorithms presented in the three subsections above perform either an exhaustive or heuristic search for mining rules and the subgroups they represent. These category of methods have been shown in past works to perform poorly in high dimensional datasets (SANTOS; VIMIEIRO, 2017). For this reason, many strategies applying soft-computing techniques have been studied. Among these, the literature has given much attention to Evolutionary Algorithms (EAs). Below we will review these categories of techniques. In particular, we give special attention to the state of the art EAs for the SD task.

2.2.4 Evolutionary algorithms

In this subsection we will review EAs designed to solve SD. These are the focus of this project, as we are studying the impact of population initialization in their outcome. Although some of these are relatively old algorithms, they are still widely regarded as the best representatives for this category of algorithms (HERRERA et al., 2011; HELAL, 2016; CARMONA; JESUS; HERRERA, 2018).

SDIGA (JESUS et al., 2007) is a mono-objective evolutionary algorithm for inducing fuzzy rules. The disjunctive normal form is used to implement these fuzzy rules and serve as a description language to represent subgroups. The algorithm evaluates the quality of subgroups by means of weighted average of many possible measures chosen by the user. The search for the fuzzy rules is carried by a genetic algorithm, but a post-processing step is done afterwards which applies local search to find one simple and interpretable fuzzy rule as the algorithm's output.

After each run of SDIGA's genetic algorithm, a rule is obtained and a post-processing step performs a hill climb to try to improve the rule. This step makes the algorithm run significantly slower than the others presented in this subsection. Even though SDIGA allows the user to configure multiple evaluation measures, it is not a real multi-objective algorithm, it just performs a weighted average of the configured measures in order to obtain the final fitness of each individual, so the user also has to select the weights of each measure.

MESDIF (JESUS; GONZALEZ; HERRERA, 2007) is a multi-objective genetic algorithm for inducing fuzzy rules based on the SPEA2 and many evaluation measures can be combined to evaluate the generated subgroups. During its search, MESDIF keeps a separate elite population, the size of which is determined by the user. By the end of the algorithm's execution, the user can retrieve this elite population and use it as the set of returned subgroups. MESDIF provides an objective based on novelty to avoid redundancy in the elite population.

NMEEF-SD (CARMONA et al., 2009) is also a multi-objective genetic algorithm designed to induce fuzzy or crisp rules for the SD task. Therefore, it can also be used with a wide variety of evaluation measures. Different from the MESDIF, NMEEF-SD is based on the NSGA-II. Another difference is that NMEEF-SD performs a reinitialization based on coverage after a full iteration of the genetic algorithm is over. This behavior is designed to increase the diversity in the population.

Both MESDIF and NMEEF-SD are adaptations of general-purpose EAs, namely SPEA2 and NSGA-II. As such, they were not designed from the group up to solve SD. This is reflected in their initialization operators, which will often generate individuals with too many attribute-value pairs, representing rules which are too specific and that will most likely have poor quality. One advantage they have when compared to SDIGA is that they are true multi-objective algorithms, so the user can configure which quality measures are desired and leave for the algorithms to balance them during the search.

Finally, SSDP (LUCAS et al., 2017) is a mono-objective EA designed specifically for solving SD tasks in high dimensional datasets. One of the most discerning characteristics of the algorithm, when compared to other EAs, is that it only has one parameter which needs to be set by the user: the number of rules to be returned. This is due to its particular way to initialize the population: SSDP generates a starting population with every possible rule with a single attribute-value pair. As the algorithm is designed to return the top-k rules found during the search, it records the most relevant individuals found in a separate, elite, population. Individuals are only inserted in this elite population in case they are relevant, and SSDP uses the same concept of relevance as the SD algorithm.

Although SSDP has been shown by its authors to work well in both low and high dimensional SD tasks, it has two limitations. The first is the fact that it does not support continuous attributes. So any dataset must have its attributes transformed to discrete intervals before SSDP can be executed. The second lies in its initialization technique. Because it generates all possible individuals with exactly one attribute-value pair, and it is designed for high dimensional datasets, it is common to initialize SSDP with tens of thousands of individuals. This may quickly drain the computational resources available.

Now that the existing approaches to solve SD have been described and the state of the art EAs have been listed, the next chapter will discuss each of these last three algorithms in detail. Because SDIGA runs significantly slower, and we could not spare the time or computational resources, we decided to leave it out of our list of algorithms selected to study and answer this project's research question. After MESDIF, NMEEF-SD and SSDP are discussed in detail, Chapter 4 will describe how their initializations were modified to test and verify the hypothesis.

3

Evolutionary Algorithms for Subgroup Discovery

In this chapter, we will cover some of the state of the art Evolutionary Algorithms (EAs) for Subgroup Discovery (SD) in detail. The first two, MESDIF and NMEEF-SD were designed for low dimensional SD problems and have been shown to perform poorly in high dimensional ones (LUCAS et al., 2017). The last one, called SSDP, on the other hand, was specifically designed for high dimensional problems. These details will prepare for the next chapter, where we will review our hypothesis, explain the intuition behind it and how we will go about testing it.

3.1 MESDIF

This section describes the data mining algorithm called Multi-objective Evolutionary Subgroup Discovery Fuzzy rules (MESDIF) (JESUS et al., 2007). It is a multi-objective Genetic Algorithm (GA) for the extraction of rules which describe subgroups. It works for discrete target variables, and also supports both continuous and nominal attributes.

When extracting the rules, the algorithm keeps the consequent of the rule fixed in some value of the target variable and the search is done for the antecedent part only. In practice this means that, to extract knowledge in all possible classes of the problem, the algorithm needs to be run once for each class.

The rules extracted can be crisp and represented in canonical form or fuzzy and described in the Disjunctive Normal Form (DNF). The first is a conjunction of attribute-value pairs, such as the ones described in Chapter 2, while the second is used as description language to specify subgroups, which permits a disjunction for the values of any variable present in the antecedent of a rule.

MESDIF is based on the SPEA-2 strategy and, as such, it applies the concept of elitism in its search for rules, keeping an elite population of non-dominated individuals. In the end of its execution, MESDIF returns the rules present in this elite population, the size of which is predetermined by the user.

The outline of the algorithm is shown in Figure 3.1. Next, we explain the most important components of its design in more detail.

Figure 3.1: Scheme of the MESDIF algorithm.

```
Step 1. *Initialization*:
    Generate an initial population $P_0$ and create an empty elite population $P'_0 = \emptyset$. Set $t = 0$.
Repeat
    Step 2. *Fitness assignment*:
        calculate fitness values of the individuals in $P_t$ and $P'_t$.
    Step 3. *Environmental selection*:
        copy all non-dominated individuals in $P_t$ and $P'_t$ to $P'_{t+1}$. As the size of $P'_{t+1}$ must
        be exactly the number of individuals to store (N), we may have to use truncation or a filling
        function.
    Step 4. *Mating selection*:
        perform binary tournament selection with replacement on $P'_{t+1}$ applying later crossover
        and mutation operators in order to fill the mating pool (obtaining $P_{t+1}$).
    Step 5. *Increment generation counter*:
        $(t = t + 1)$
While stop condition is not verified.
Step 6.
    Return the non-dominated individuals in $P'_{t+1}$.
```

Source: adapted from (JESUS et al., 2007)

3.1.1 Individual Representation

The choice of how to represent the solutions being searched for by the algorithm is perhaps the most important step in GA design. The encoding approach used in MESDIF is the "Chromosome = Rule", in which each individual encodes a single rule (instead of, for instance, representing a set of rules).

Because the consequent of the rule is kept to a fixed value during the algorithm's execution, the individual only represents the antecedent part of the rule. Each individual has a chromosome, which, in MESDIF, is implemented using an array of integers. Each position in the array represents an attribute (excluding the target) in the dataset. The i-th position in the array contains the value paired with the i-th attribute. So each position in the array encodes an attribute-value pair and the rule is obtained by the conjunction of each element in the array. If the value in the i-th position is higher than the number of possible values for the dataset's i-th attribute, then it means, the i-th attribute is not a part of the rule represented by that array.

3.1.2 Population Initialization

MESDIF initializes its starting population at random. For each attribute available in the dataset, a random value will be drawn in a uniform manner to pair with that attribute. That way, if the dataset has m attributes, then every rule in the starting population will have m attribute-value pairs. This means the initial rules will be very specific, i.e., they will be a conjunction of so many attribute-value pairs, that the rule ends up covering few examples in the dataset, if any at all.

For datasets with a small number of attributes, this might not be a problem. The evolutionary operators might tweak the individuals into covering more examples. For high dimensional datasets, however, the individuals in the starting population will often cover zero examples, and the tweaks made by the mutation and crossover operators may not be enough to lead the population to better individuals, since both old and new generations of individuals will have similarly low values for quality measures, and the evolutionary process stagnates.

After we present how the other algorithms initialize their starting population, we will discuss the different approaches to population initialization in more depth and present our own strategies.

3.1.3 Objectives optimized

Being a multi-objective EA, MESDIF can be configured to optimize any number of evaluation metrics. However, the main implementation of the algorithm, made by its authors and available in the software package KEEL (ALCALÁ-FDEZ et al., 2009), provides as default the Sensitivity, Weighted Relative Accuracy (WRAcc) and Fuzzy Confidence measures. The first two were introduced in Section 2.1 and the later is a fuzzy version of Confidence introduced by the authors.

3.1.4 Reproduction Model

The way MESDIF moves from one generation to the next is the following: first, the original population and the elite population are joined together and the non-dominated individuals of the joined population are determined. Domination is defined in terms of Pareto fronts. Second, binary tournament selection is applied to the non-dominated individuals. Last, recombination is applied to the resulting population by two-point crossover and biased mutation, where half of the mutations carried out have the effect of eliminating one of the attribute-value pairs in the rule, to increase its generality.

3.2 NMEEF-SD

This section describes the data mining algorithm called Non-dominated Multi-objective Evolutionary Algorithm Based on the Extraction of Fuzzy Rules for Subgroup Discovery (NMEEF-SD) (CARMONA et al., 2009). Similarly to MESDIF, NMEEF-SD is a multi-objective GA for the extraction of rules which describe subgroups, however, it is based on the NSGA-II strategy.

It is designed to work with discrete target attributes, but supports both continuous and discrete descriptive features. When working with continuous variables, NMEEF-SD uses fuzzy rules. For discrete attributes, however, the algorithm uses crisp rules. Like MESDIF, it can work with DNF or canonical representations.

Its search is performed by means of evolving the population of individuals based on the non-dominated sort in fronts of dominance. The first front contains the non-dominated solutions in the population (also called the Pareto front). The second contains solutions dominated by one solution, while the solutions in the third front are dominated by two, and so on.

The pseudo-code for the algorithm is reproduced in Figure 3.2 and, we now describe the important elements of its design in detail.

Figure 3.2: The NMEEF-SD algorithm.

```
BEGIN
    Create $P_0$ with biased initialization
    REPEAT
        $Q_t \leftarrow \emptyset$
        Tournament Selection ($P_t$)
        $Q_{tc} \leftarrow$ Multi-point Crossover ($P_t$)
        $Q_{tm} \leftarrow$ Biased Mutation ($Q_{tc}$)
        $Q_t \leftarrow Q_{tc} + Q_{tm}$
        $Q_t \leftarrow Q_t +$ descendants
        $R_t \leftarrow$ Join($P_t$,$Q_t$)
        Fast-non-dominated-sort($R_t$)
        IF $F_1$ evolves
            Introduce fronts in $P_{t+1}$
        ELSE
            Re-initialization based on coverage $P_{t+1}$
    WHILE ($num\_eval < max\_eval$)
END
```

Source: adapted from (CARMONA et al., 2009)

3.2.1 Individual Representation

The individuals in NMEEF-SD are encoded according to the "Chromosome = Rule" approach, just as MESDIF does. As such, only the antecedent part of the rule is represented in the individual. In each execution of the algorithm, a value of the target attribute is fixed and the entire search process must be performed once for each of the desired values.

3.2.2 Population Initialization

NMEEF-SD's population initialization technique is less random than MESDIF's. It generates 75% of the individuals in the population using only 25% of the available attributes in the database. The remaining 25% of the starting population is randomly generated the same way as in MESDIF.

According to the authors, the intent behind this technique is to obtain a set of rules with high generality, because most of the generated rules have a short antecedent expression, in terms of the number of attribute-value pairs. Though the technique might have achieved its purpose in low dimensional SD problems, this is not the case for high dimensionality.

Indeed, if the dataset has a high enough number of attributes, this restriction to 25% of the available attributes will not achieve the objective of generating starting individuals with high generality, as 25% of the available attributes will still be too many, and this initialization technique will suffer from the same problems as MESDIF's. We discuss this further in Section 3.4.

3.2.3 Objectives Optimized

Being a multi-objective algorithm, NMEEF-SD can be configured to use many different evaluation metrics. However, the implementation provided by its authors, as part of the software package KEEL (ALCALÁ-FDEZ et al., 2009), uses as default objectives the Sensitivity and WRAcc metrics, which were introduced in Section 2.1.

3.2.4 Reproduction Model

As outlined in Figure 3.2, NMEEF-SD obtains a descendant population (called Q_t in the outline), with the same size as the original one, through its genetic operators. Namely, the algorithm uses Tournament Selection, multi-point crossover and biased mutation.

The mutation is applied to a given gene according to a configured probability. If mutation is to be carried out, then the operator can work in one of two ways: the first eliminates an attribute-value pair from the individual, aiming to create a more general rule; the second randomly mutates the value paired with a given attribute. Both the first and second ways to mutate a gene can be applied in each mutation with the same probability.

After a new population is obtained through the genetic operators, both the parent and descendant populations are joined in a new population, called R_t in the outline. Subsequently, the non-dominated sort is applied to this resulting population in order to classify each individual in fronts of dominance.

Lastly, NMEEF-SD may re-initialize its population before moving on to the next generation. First, the Pareto front of the new population (P_{t+1} in the outline) is checked to see whether or not it evolves. It is considered to evolve if it covers at least one more instance of the dataset than the Pareto front of the previous population (P_t). Then, if the Pareto front does not evolve during more than 5% of the evolutionary process (measured by the number of individual evaluations), the population is re-initialized.

The re-initialization is based on coverage. The individuals in the Pareto front which cover the same instances of the dataset are considered repeated and are replaced by new individuals generated to cover previously uncovered instances.

3.3 SSDP

Simple Search Discriminative Patterns (SSDP) (LUCAS et al., 2017) is a mono-objective evolutionary approach for SD. SSDP stands out especially due to two of its characteristics: being designed for high dimensional data, and having few parameters to adjust. Another important characteristic of SSDP is its lack of support for continuous attributes.

SSDP does not allow its user to set parameters commonly tuned in traditional EAs, such as mutation and crossover rate and the population size. Instead, the algorithm has only two parameters: the number of returned subgroups (represented as rules), and the evaluation metric to be used as fitness function. In theory, SSDP allows the use of any of the already presented evaluation measures as the fitness function.

The outline for the SSDP algorithm is depicted in Figure 3.3. In the next subsections, we describe the important elements of SSDP in detail.

Figure 3.3: SSDP pseudo-code.

```
Require: $k$, evaluationMetric
  Initialize $P$ with every possible individual of size 1.
  $P_k \leftarrow$ kBestRelevantes($P$)
  reinitializationCount $\leftarrow 0$
  mutationRate $\leftarrow 0.4$
  crossOverRate $\leftarrow 0.6$
  while reinitializationCount < 2 do
      generations $\leftarrow 0$
      while $P_k$ does not improve after 3 consecutive generations with mutationRate $= 1$ do
          if generations == 1 then
              $P_{new} \leftarrow crossoverAND(P)$
          else
              $P_{new} \leftarrow$ evolutionaryOperator(P, mutationRate, crossOverRate, evaluationMetric)
          end if
          $P^* \leftarrow$ best($P$, $P_{new}$)
          $P_k \leftarrow$ kBestRelevants($P_k$, $P^*$)
          update(mutationRate, crossOverRate)
          $P \leftarrow P^*$
          generations $\leftarrow$ generations $+1$
      end while
      reinitializationCount $\leftarrow$ reinitializationCount $+1$
      Restart $P$
  end while
  return $P_k$
```

Source: adapted from (LUCAS et al., 2017)

Like MESDIF, SSDP keeps a separate population for the best individuals found during the search step. The size of it is one of the two necessary parameters, named *k* in Figure 3.3. Furthermore, when there is no change to the top-*k* individuals in this elite population for three consecutive generations and the mutation rate has already been updated to its maximum value, 1, then SSDP performs a reinitialization of the population. After the population has been reinitialized twice, the algorithm stops.

The mutation and crossover rates start at 0.4 and 0.6, respectively, and are updated every generation depending on how the search process affected the elite population P_k. If, at the end of an evolutionary step, the new generation improved P_k, then the crossover rate is increased by 0.2 and the mutation rate is decreased by 0.2, making the search process more exploitative. In case there was no improvement to P_k, however, the mutation rate is increased by 0.2 and the crossover rate is decreased by 0.2, thus making the search a more exploratory process. Regardless of how the rates are updated, they always sum to 1.

The elite population P_k is only updated by means of inserting the best, in terms of fitness, and relevant individuals from either the starting population or the descendants. Relevance is measured as follows: an individual outside P_k is considered irrelevant with respect to the population P_k if there is an individual, in P_k, that covers more examples of the target class and less examples of the other classes than the outsider.

3.3.1 Individual Representation

Similarly to MESDIF and NMEEF-SD, SSDP encodes its individuals according to the "Chromosome = Rule" approach. Therefore, only the antecedent part of the rule is encoded in the individual representing it, and so an entire execution of SSDP is required for each different target value.

3.3.2 Population Initialization

SSDP generates its population with every possible individual of size 1, i.e. individuals representing rules with just one attribute-value pair. As mentioned before, this means the user need not to provide the algorithm with a population size parameter as it is automatically determined. It also means that, for high dimensional problems, it is not uncommon to have a starting population with tens or hundreds of thousands of individuals.

Although appearing to be very limiting, this deterministic initialization scheme guarantees the starting population will be somewhat evenly spread around the search space, leaving the task of expanding the population to individuals of greater sizes to the evolutionary step. This strategy also makes sure that every attribute-value pair is considered in the search, granting that not all combinations of pairs will be considered.

Initializing the search from all possible one-dimensional solutions was a novelty among evolutionary strategies, even though it was already widely used by many algorithms based on Beam Search.

Because the number of possible attribute-value pairs will be very high, for high dimensional datasets, it is expected that SSDP's population will consume a lot of computational resources. Therefore, it is important to test whether it is really necessary to have such an enormous initial population, or just decreasing the average size of each individual generated is enough to operate an EA in high dimensional SD problems.

3.3.3 Objectives Optimized

As a mono-objective algorithm, SSDP can only use one evaluation metric at a time for discovering subgroups. As mentioned before, it can be used with any of the evaluation metrics found in the literature. The default implementation, provided by its authors (LUCAS et al., 2017), however, includes three metrics: Q_g, WRAcc and SUB.

3.3.4 Reproduction Model

Because SSDP starts with individuals of size 1, it is the responsibility of the evolutionary operators to move the search towards individuals of greater sizes. For that end, SSDP uses binary tournament as a way to select individuals for breeding, has two different crossover and one mutation operators which are described next.

The first crossover operator, called *crossOverAND* (see Figure 3.3), is used only in the first generation, in which all individuals have just one attribute-value pair. It generates a new individual from the union of the two attribute-value pairs of its parents.

In the $crossOverUniform$, SSDP's second crossover operator, two parent individuals generate two new children by combining their attribute-value pairs uniformly with a 50% mixing ratio. That is, for each attribute-value pair in a parent individual, that pair can be inherited by the first or second individuals with equal probability, but only one of them will inherit that particular attribute-value pair. The crossover finishes once all attribute-value pairs in both parents have been inherited by exactly one of the children.

Lastly, the mutation operator for SSDP has equal probability of doing either one of three possible operations: (1) a random attribute-value pair is added to the individual; (2) a random attribute-value pair is replaced by another; (3) a random attribute-value pair is removed from the individual.

Now that we reviewed three state of the art EAs for SD, the next section will discuss how population initialization has a significant impact on the performance of these algorithms.

3.4 Initializing Populations for High Dimensional SD

Generating a starting population is the first algorithmic step in most population-based search (or optimization) strategies. In EAs, this first step is responsible for creating the first estimates of the solutions to the problem. It stands to reason that good initial solutions may provide a head start for an algorithm and improve its overall performance while saving computational resources. Conversely, a bad set of estimates could get the algorithms "stuck" on poor regions of the search space, causing the output of bad-quality answers.

Therefore, selecting a proper population initialization technique is an important part of EA design. If there is not enough information about the problem, however, it is almost impossible for an algorithm designer to identify which are good estimates for building the initial population before evaluating the whole optimization process. Consequently, it has become a trend between researchers to use uniformly distributed random numbers as a tool for building the initial population (KAZIMIPOUR; LI; QIN, 2014b). The intuition behind this technique is that, by generating a starting population which is uniformly distributed, the chance of missing a considerable portion of the search space during the optimization phase of the algorithm is decreased.

The popularity of uniform random initialization techniques can also be justified by its simplicity. Fast Pseudo-Random Number Generator (PRNG) tools are available in the standard library of almost all modern programming languages and there is no restriction in the number of points to be drawn (in our case, population size) nor in the dimension size (in our case, number of attributes in the database). Therefore, these techniques can be easily applied to a large number of problems.

Both MESDIF's and NMEEF-SD's population initialization techniques revolve around generating uniformly distributed random individuals. MESDIF randomly selects a value to pair with each of the available attributes from an uniform distribution. This technique is represented in Algorithm 1.

Algorithm 1 MESDIF's initialization technique (reuses the notation set in Chapter 2).

```
Require: $\boldsymbol{A}$: set of attributes
 1: function MESDIF-Initialize(popSize, $\boldsymbol{A}$)
 2:     $\boldsymbol{P} \leftarrow \emptyset$                                        ▷ Population
 3:     for $i \leftarrow 1$ to popSize do
 4:         $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{$ MESDIF-Generate($\boldsymbol{A}$) $\}$
 5:     end for
 6:     return $\boldsymbol{P}$
 7: end function
 8: function MESDIF-Generate($\boldsymbol{A}$)
 9:     $\boldsymbol{I} \leftarrow \emptyset$                                        ▷ Individual
10:     for $attr \leftarrow 1$ to $|\boldsymbol{A}|$ do
11:         $val \leftarrow$ Uniform-Random(1, $|\boldsymbol{a}_{attr}|$)
12:         $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{\langle attr, v_{val}^{attr}\rangle\}$
13:     end for
14:     return $\boldsymbol{I}$
15: end function
```

NMEEF-SD does the same for 25% of its population. For the other 75%, however, only 25% of the attributes (chosen at random) are paired with randomly selected values. Both use the uniform distribution to draw its random numbers. This technique is represented in Algorithm 2.

Both algorithm's initialization techniques suffer from two drawbacks. The first lies on the fact that, due to the deterministic nature of computers, Random Number Generators (RNGs) cannot produce a perfect uniform distribution of points, hence why RNG algorithms are called Pseudo-Random Number Generators.

Algorithm 2 NMEEF-SD's initialization technique (reuses the notation set in Chapter 2).

```
Require: $\boldsymbol{A}$: set of attributes
 1: function NMEEF-INITIALIZE(popSize, $\boldsymbol{A}$)
 2:     $\boldsymbol{P} \leftarrow \emptyset$                                  ▷ Population
 3:     for $i \leftarrow 1$ to popSize do
 4:         if $i \leq 0.25 \times$ popSize then
 5:             $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{$ MESDIF-GENERATE($\boldsymbol{A}$) $\}$
 6:         else
 7:             $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{$ NMEEF-GENERATE($\boldsymbol{A}$) $\}$
 8:         end if
 9:     end for
10:     return $\boldsymbol{P}$
11: end function
12: function NMEEF-GENERATE($\boldsymbol{A}$)
13:     $\boldsymbol{I} \leftarrow \emptyset$                                  ▷ Individual
14:     $n \leftarrow 0.25 \times |\boldsymbol{A}|$        ▷ number of attribute-value pairs for this individual
15:     while $|I| < n$ do
16:         $attr \leftarrow$ UNIFORM-RANDOM(1, $|\boldsymbol{A}|$)
17:         $val \leftarrow$ UNIFORM-RANDOM(1, $|\boldsymbol{a}_{attr}|$)
18:         $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{\langle attr, v_{val}^{attr}\rangle\}$
19:     end while
20:     return $\boldsymbol{I}$
21: end function
```

Furthermore, they suffer from the Curse of Dimensionality. That is, when the dimension of the problem is not very high and the population size is high enough, PRNGs can provide initial populations with satisfactory level of uniformity (MA; VANDENBOSCH, 2012). As the dimensionality of the problem grows, however, PRNGs cannot produce perfect, evenly distributed points.

The second drawback regards a characteristic of the problem at hand. As we mentioned, initializing the population uniformly distributed across the search space is generally a good guess when there is scarce information about the problem. In SD, we are mining the database for rules which describe subpopulations in the data. As mentioned, it is of high importance that these rules be comprehensible for a human expert user. It stands to reason, then, that the rules which we are looking for need to be simple, and so, contain few attribute-value pairs.

MESDIF's initialization starts every individual with one pair for each possible attribute. So we know that these starting individuals represent rules with too many attribute-value pairs and are bad starting points for the optimization step. Furthermore, the tweaks made by the crossover and mutation operators may not be enough to lead the search process away from these enormous, useless rules and the convergence of the evolutionary process is questionable. Knowing this characteristic of SD problems, we could accelerate and improve the search process by starting from shorter rules.

NMEEF-SD's initialization represents a step in that direction, as 75% of the initial population consists of individuals representing rules with just 25% of the number of attribute-value pairs that the MESDIF's initial population would have. This remedies the second drawback to a certain degree. If given a database with tens of thousands of attributes, however, 25% of that may still be too high a number of attribute-value pairs. Therefore, for a high enough number of available attributes, NMEEF-SD's initial population is still expected to suffer the same problem as MESDIF's.

SSDP uses, instead, a deterministic initialization of its starting population. That is, every time SSDP initializes the population for a given database, it generates the same individuals. Each individual has exactly one attribute-value pair and there is one individual for every possible pair. This strategy is illustrated in Algorithm 3.

Algorithm 3 SSDP's initialization technique (reuses the notation set in Chapter 2).

```
Require: $\boldsymbol{A}$: set of attributes
 1: function SSDP-INITIALIZE($\boldsymbol{A}$)
 2:     $\boldsymbol{P} \leftarrow \emptyset$                              ▷ Population
 3:     for $attr \leftarrow 1$ to $|\boldsymbol{A}|$ do
 4:         for $val \leftarrow 1$ to $|\boldsymbol{a}_{attr}|$ do
 5:             SSDP-GENERATE($attr$, $val$)
 6:         end for
 7:     end for
 8:     return $\boldsymbol{P}$
 9: end function
10: function SSDP-GENERATE($attr$, $val$)
11:     $\boldsymbol{I} \leftarrow \{\langle attr, v_{val}^{attr} \rangle\}$          ▷ Individual
12:     return $\boldsymbol{I}$
13: end function
```

This initialization technique avoids both drawbacks mentioned above. First, the initial individuals are evenly spread around the search space. Second, because every individual has size 1, they are easily interpretable. It stands to reason, however, that many of these individuals will have poor quality, because they are too general, i.e. it is likely that the rules represented by these individuals will cover too many instances in the database, both from the target class and not. To remedy that, SSDP also makes sure that the second generation of individuals are only created by combining the individuals from the first generation, through its crossover operator.

Although SSDP has been shown to perform very well in high dimensional databases (LUCAS et al., 2017), this initialization technique forces the algorithm to require a rather large amount of computational resources as its population size is a function of the number of available attributes in the database. Since it was designed for high dimensionality, it is not uncommon to initialize the population containing hundreds of thousands of individuals.

This scenario raised the following hypothesis to be verified in this work: the state *of the art EAs designed to work in low dimensional databases do not work well in high dimensional problems, because their initialization scheme creates rules with too many attribute-value pairs, that is, the rules are too specific. By decreasing the average rule size in the starting population, these algorithms can be made to work in high dimensional SD problems.*

To test this hypothesis, we need to replace the initialization technique of these algorithms and verify the impact of that change in the outcome of the algorithm. In the next chapter, we propose new population initialization operators and demonstrate, through empirical tests, what was the impact in the convergence of MESDIF and NMEEF-SD.

4

On the convergence of evolutionary algorithms for high dimensional subgroup discovery

This chapter aims at demonstrating, through empirical tests, that decreasing the average number of attribute-value pairs in rules represented by individuals in the starting population, increases the convergence rate for the state of the art EAs, described in the last chapter, when solving high dimensional SD problems. This chapter is a textual extension of what was published in the seventh Brazilian Conference on Intelligent Systems (BRACIS) (TORREÃO; VIMIEIRO, 2018).

As we mentioned in the last chapter, the first requirement to make such a demonstration is an initialization technique to replace the ones from the presented EAs. This new strategy must be designed in such a way as to allow us to tune the average number of attribute-value pairs in the generated individuals, so that we can observe the effect of reducing or increasing it in the outcome of the algorithms.

This new initialization technique will require the user to input the number of individuals the starting population must have. With that information provided, the technique will loop through the following procedure, generating one individual at a time, until the population is complete. First, a random number, x, is drawn from some PRNG. x must be any real number between 1 and 100 (both inclusive). Then, the technique randomly selects attributes and values to be paired and added to the new individual until it reaches a number of pairs equal to x% of the total available attributes. Finally, the new individual is appended to the population.

This approach is very similar to how NMEEF-SD initializes its starting population, except that instead of fixing the number of attribute-value pairs to 25% of the available attributes, a different proportion is randomly selected for each new individual. Also, because we want to keep control over the average number of attribute-value pairs in the population, instead of using the uniform distribution, we are drawing random numbers from Beta Distributions (WEISSTEIN, 2003), from which we also borrow the name for our initialization technique. The final difference from NMEEF-SD's strategy is that our technique generates all of the starting population individuals the same way, while NMEEF-SD generates 25% of the population in the same way as MESDIF.

It is important to note that we are using Beta Distributions for determining the proportion of attribute-value pairs only. That is, for choosing the attribute and value to pair together, we

Algorithm 4 The proposed beta initialization technique (reuses the notation set in Chapter 2).

```
Require: $\boldsymbol{A}$: set of attributes
Require: $\alpha$: the $\alpha$ parameter for the beta distribution
Require: $\beta$: the $\beta$ parameter for the beta distribution
function BETA-INITIALIZE(popSize, $\boldsymbol{A}$, $\alpha$, $\beta$)
    $\boldsymbol{P} \leftarrow \emptyset$                                    ▷ Population
    for $i \leftarrow 1$ to popSize do
        $\boldsymbol{P} \leftarrow P \cup \{$ BETA-GENERATE($\boldsymbol{A}$, $\alpha$, $\beta$) $\}$
    end for
    return $\boldsymbol{P}$
end function
function BETA-GENERATE($\boldsymbol{A}$, $\alpha$, $\beta$)
    $\boldsymbol{I} \leftarrow \emptyset$                                    ▷ Individual
    $rand \leftarrow$ BETA-RANDOM($\alpha$, $\beta$)
    $n \leftarrow rand \times |\boldsymbol{A}|$          ▷ number of attribute-value pairs for this individual
    while $|I| < n$ do
        $attr \leftarrow$ UNIFORM-RANDOM(1, $|\boldsymbol{A}|$)
        $val \leftarrow$ UNIFORM-RANDOM(1, $|\boldsymbol{a}_{attr}|$)
        $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{\langle attr, v_{val}^{attr} \rangle\}$
    end while
    return $\boldsymbol{I}$
end function
```

are still using the uniform distribution, as can be observed in Algorithm 4, which contains the pseudo-code for the proposed technique.

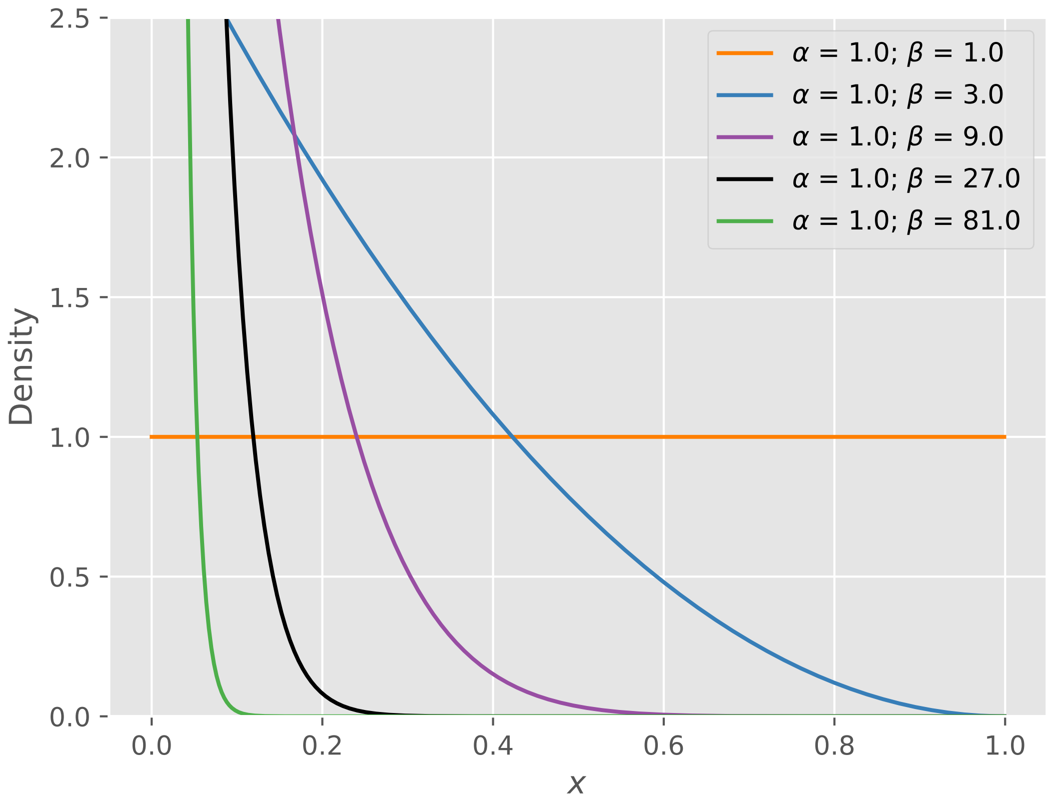
The use of the beta distribution provides the technique with flexibility, as it has two parameters: α and β. By changing these values, it is possible to obtain many different Probability Density Functions (PDFs). For instance, with $\alpha = 1$ and $\beta = 1$, we get the uniform distribution.

Figure 4.1 depict five different PDFs obtained from varying the values of β, while keeping α fixed at 1. It is interesting to observe how just changing the values of β can already cause significant differences in the resulting PDF.

In our initialization technique, we do not fix the number of attribute-value pairs an individual will have like MESDIF and NMEEF-SD do ($|\boldsymbol{A}|$ and $0.25 \times |\boldsymbol{A}|$, respectively). The Beta initialization will generate the individuals with a random proportion of the available attributes in the dataset. Each time a new individual must be generated, we first randomly pick a number, w, between 0 and 1, from a beta distribution obtained with the given α and β parameters. Then, we build an individual with $w \times |\boldsymbol{A}|$ attribute-value pairs. The attributes and values are randomly picked from a uniform distribution.

Therefore, the obtained distribution has a high impact in the generated individuals. The average number of attribute-value pairs will depend on three parameters: α, β and the set of attributes in the dataset, $\boldsymbol{A}$. α and β control the average proportion between the number of attribute-value pairs in a generated individual and the number of attributes available in the dataset. This means that $|\boldsymbol{A}|$ is the maximum number of attribute-value pairs a Beta initialization

Figure 4.1: A few examples of PDFs for the Beta Distribution. α is always 1, while β varies from 1 to 81.



generated individual could have, while the minimum is 1.

For example, if we set $\alpha = \beta = 1$, we get the uniform distribution. Hence, on average, a generated individual will have a number of attribute-value pairs equal to 50% of the available attributes in the dataset ($0.5 \times |\boldsymbol{A}|$), because the mean value of the Beta distribution with $\alpha = \beta = 1$ is 0.5.

In this chapter, we aim at evaluating the outcome of the selected EAs, in high dimensional SD tasks when the average number of attribute-value pairs contained in the generated individuals for the starting population is decreased. In the context of our initialization technique, decreasing the average number of attribute-value pairs means changing α and β in such a way that the mean value for the resulting distribution is decreased.

With an initialization technique suitable for checking our hypothesis, the next section presents the methodology for the empirical tests executed to that end.

4.1 Methodology

In order to check the hypothesis, all three EAs described in Chapter 3 (MESDIF, NMEEF-SD and SSDP) were modified to initialize the population using the technique described above. Henceforth, the algorithms as they were originally proposed will be referenced as “original” or “original version”, to differentiate from the case when the initialization step was modified by replacing their original initialization step with the technique proposed in this chapter, which will be referenced as “modified version” or yet “beta initialization version”.

For this experiment, the beta initialization technique was instanced with the following values for α and β: (1) $\alpha = 1$, $\beta = 1$; (2) $\alpha = 1$, $\beta = 3$; (3) $\alpha = 1$, $\beta = 9$; (4) $\alpha = 1$, $\beta = 27$; (5) $\alpha = 1$, $\beta = 81$. Figure 4.1 depicts the PDFs for the distributions obtained with these parameters, while Table 4.1 contains the statistics of each. We could have selected more values for the α and β parameters, but as we discuss later in this section, each new value for these parameters required many thousand executions of our experiments. We did not have the computational or time resources for that.

Table 4.1: Statistics for the five different beta distributions

Parameters	Mean	Mode	Std. Dev.	Variance
$\alpha = 1.0, \beta = 1.0$	0.5	-	0.2886	0.0833
$\alpha = 1.0, \beta = 3.0$	0.25	0	0.1936	0.0375
$\alpha = 1.0, \beta = 9.0$	0.1	0	0.0904	0.0081
$\alpha = 1.0, \beta = 27.0$	0.0357	0	0.0344	0.0011
$\alpha = 1.0, \beta = 81.0$	0.0121	0	0.0120	0.0001

As the reader can see from Table 4.1, keeping the value of α fixed at 1 causes the probability of generating an individual representing a rule with a high number of attribute-value pairs to decrease, as the value of β increases. In fact, the average number of attribute-value pairs in the generated individuals should be approximately the mean value of the obtained distribution, which means: 50% of the total number of attributes in the dataset when $\beta = 1$, 25% for $\beta = 3$, 10% in case $\beta = 9$, about 3% when $\beta = 27$ and approximately 1% for $\beta = 81$. If we varied the value of α, we would obtain severely different shapes for the PDFs and would loose this effect, which is essential for testing how our initialization technique affects the algorithms as the average number of attribute-value pair decrease.

Having a way to decrease the average number of attribute-value pairs in the individuals forming the starting population, the second requirement to test our hypothesis are datasets to run the algorithms in and measure the outcome. Because we are interested in high dimensional SD problems, the experiment was conducted in 10 microarray datasets available in the package called *datamicroarray* from R software, which are described in detail in Table 4.2.

For each dataset, the majority class was considered the target value, and thus labeled “positive”, whereas the other class values were labeled “negative”. This was done due to fact that

Table 4.2: Detailed information of the microarray datasets.

Name	# Examples	# Attributes	# Labels
alon	62	2,000	2
burczynski	127	22,283	3
chiaretti	111	12,625	2
chin	118	22,215	2
christensen	217	1,413	3
gravier	168	2,905	2
nakayama	105	22,283	10
sun	180	54,613	4
tian	173	12,625	2
yeoh	248	12,625	6

each of the EAs can only consider one target value per execution, so considering only one of the values considerably simplifies our measurements.

In addition to that, because SSDP does not support continuous attributes, all attributes in the datasets were discretized. Discussions on discretization techniques are outside the scope of this work, so we used the two most simple techniques available. From each dataset, two new were generated: one using the equal frequency discretization method and another using equal width, for a total of 20 available datasets. In both cases, the attributes were discretized into two possible values.

These datasets alone are not enough to test the hypothesis, as we need some way to increase the dimensionality to observe the outcome of the algorithms as it grows. To achieve that, for each of the 20 datasets, eight new ones were generated by randomly selecting a subset of features. Datasets with 1%, 2%, 5%, 10%, 15%, 25%, 50% and 100% of the original number of features were generated for this experiment. Each algorithm was executed 30 times on each of these 160 datasets.

As the last requirement to test our hypothesis, we need a way to measure the number of times each algorithm's execution converged. We have considered that an algorithm's execution converged if the average support among the resulting subgroups is greater than zero, i.e., if at least one of the subgroups cover at least one of the dataset instances.

4.2 Algorithm parameters

In this section, we go through the parameters with which each EAs were configured during this experiment.

SSDP has only two parameters to configure: the number of rules to be returned at the end of execution and the objective being optimized. In these experiments, the configured number of rules to be returned was 10 and the selected objective was the Q_g evaluation metric, with generalization parameter $g = 1$, as these are the default values in SSDP's implementation provided by the authors. Because, unlike the original initialization method, our proposed technique generates individuals with varying sizes, we have included the population size as a configurable parameter for the modified versions of SSDP. In our experiments, the population size for these modified versions of SSDP was reduced to 100, which is also the value used for this parameter in the other algorithms, both in the modified and original versions.

Both NMEEF-SD and MESDIF had its parameters configured to the following values, which are defaults in their implementations provided by their authors:

- The canonical representation for rules was used instead of the DNF;
- The population size was set to 100;
- The crossover and mutation probabilities were configured to 0.6 and 0.1 respectively.

Since both are multi-objective EAs, the objectives configured in this experiment for MESDIF were confidence, support and original support. NMEEF-SD was configured with WRAcc and support measures. These are the objectives specified in the documents where each algorithm was originally published.

4.3 Results

In this section, we examine how well the algorithms fared in terms of convergence. It is important to remember we consider that an algorithm's execution converged if at least one of the returned rules cover at least one of the instances in the dataset.

As mentioned, we obtained 160 datasets by randomly selecting attributes, in 8 different proportions, from the 20 datasets we had after applying discretization. Here, we break down this set of datasets into 8 different "buckets" according to the proportion of attributes selected. The x axis of the charts in Figure 4.2 depict these "buckets". Since each algorithm was executed 30 times on each dataset, we have a total of 600 executions in each "bucket". So, the y axis of Figure 4.2 depicts the number of executions in which the algorithm converged.

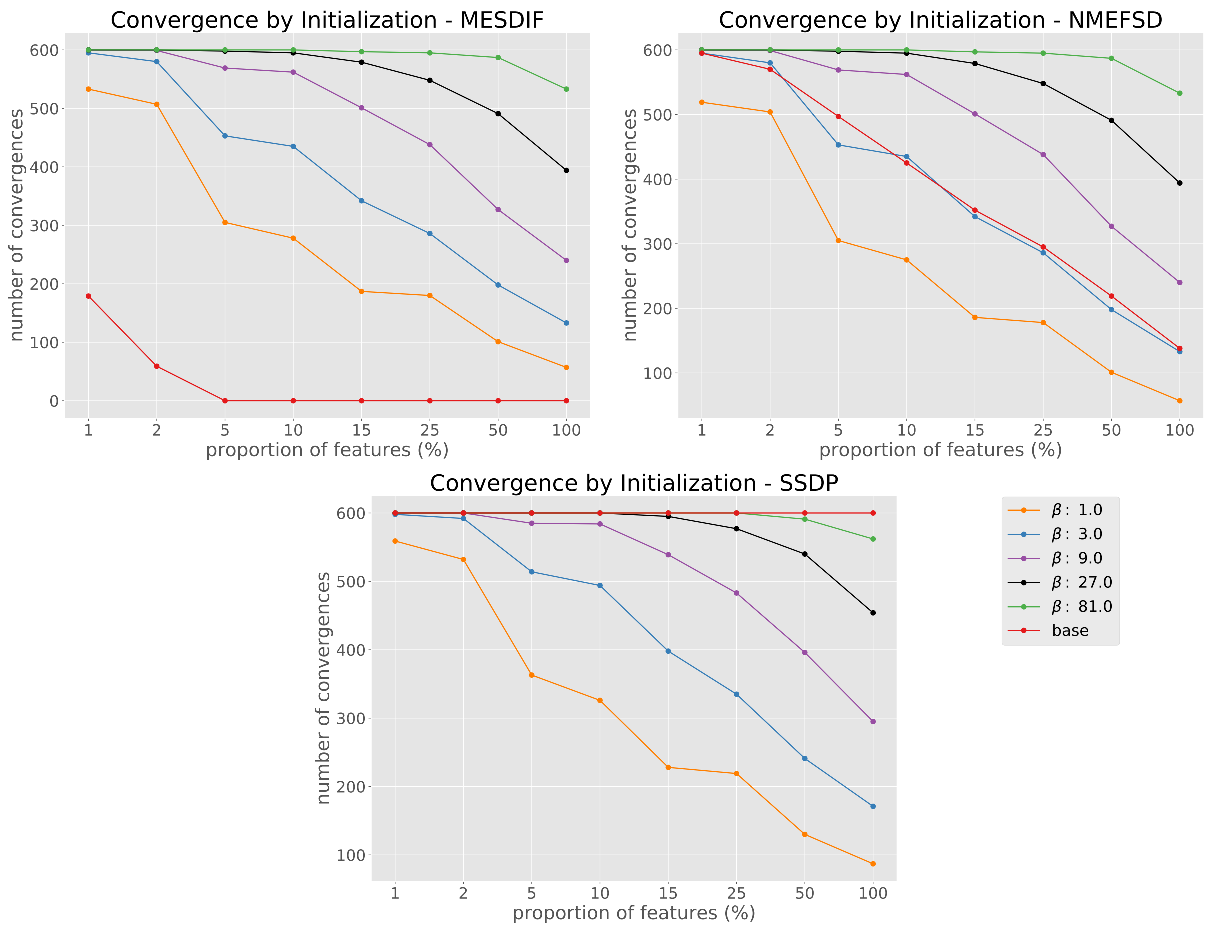
As one can see from the first chart in Figure 4.2, the original MESDIF algorithm quickly becomes ineffective as we select more attributes. In fact, as soon as the proportion of selected attributes hits 5%, the algorithm stops converging. The beta initialization versions, however, have better results. The uniform beta initialization version ($\alpha = 1$ and $\beta = 1$) already presents a significant gain in terms of number of converging executions. And as we reduce the average number of attribute-value pairs further (by increasing β), the results improve proportionately. Indeed, the modified version with $\beta = 81$, where the average number of attribute-value pairs in the generated rules is only 1% of the total of attributes in the dataset, converged in more than 500 out of the 600 executions in the datasets containing all attributes.

As expected, the original version of NMEEF-SD had a considerably better performance in terms of convergence, when compared to MESDIF's original version. Its limitation of rule sizes to 25% of the total number of attributes seems to be the determining factor, since the algorithm's original version had a remarkably similar performance to that of the modified version of MESDIF with $\alpha = 1$ and $\beta = 3$. In fact, the mean for the beta distribution with those parameters is 0.25.

With $\alpha = 1$ and $\beta = 1$, the modified initialization version of NMEEF-SD had a performance worse than that of the original version. Indeed, for NMEEF-SD, we can also observe from Figure 4.2 that, the smaller the number of attribute-value pairs in the generated individuals, the more the algorithm's executions converged.

SSDP, the algorithm with the lowest starting rule size among the three original versions, converged in 100% of the executions. When the average number of attribute-value pairs in the generated rules was increased, however, the number of times it could not converge increased as well, to the point where the beta version with $\alpha = \beta = 1$ led the algorithm to converge in less than 100 out of the 600 executions in datasets with 100% of the available attributes.

Figure 4.2: Number of times the state of the art EAs converged in high dimensional SD problems.



In addition to those conclusions, Figure 4.2 also suggests that the convergence numbers for MESDIF and NMEEF-SD, are remarkably similar when using the same initialization technique, other than their respective originals. Although SSDP modified versions' lines have the same shape as MESDIF's and NMEEF-SD's, the convergence number for SSDP are always higher than the other two, when using the same initialization technique.

As the reader can observe, the lower the average number of attribute-value pairs in the generated individuals, the greater the number of times all three algorithms converged, in their modified initialization versions. The curves presented in Figure 4.2 suggest that, had we increased the value of beta even further, beyond 81, the modified initialization versions of the algorithms would have converged in even more executions.

These findings corroborate our hypothesis that, by reducing the average number of attribute-value pairs of the generated rules for the initial population, one can adapt the MESDIF and NMEEF-SD algorithms to work on high dimensional SD problems.

Having showed that by applying our knowledge of SD to the design of initialization techniques, we have improved the convergence of EAs in high dimensional SD problems, in the next chapter, we analyze how changing the initialization technique impacts the quality of resulting rules.

5

On the result quality of evolutionary algorithms for high dimensional subgroup discovery

The first step of this research project, described in the previous chapter, was concerned with the impact of initialization techniques in the frequency of convergence for EAs in high dimensional SD problems. The next step, described in this chapter, is concerned with the impact of such techniques in the quality of rules returned by EAs in high dimensional SD problems. To that end, we set a new methodology and new empirical tests, both presented below.

Similarly to how we proceeded in Chapter 4, we will replace the initialization techniques for the MESDIF and NMEEF-SD algorithms and observe the consequences in the quality of the resulting rules. Therefore, in order to run our experiments, we need replacement initialization techniques and a quality measure from the list presented in Chapter 2.

For the initialization technique, we cannot start with the original ones from the MESDIF and NMEEF-SD algorithms, since they have a very low convergence rate in high dimensional problems. Therefore, we will start with the one presented in Chapter 4, which we call "Beta Initialization". We will modify both MESDIF and NMEEF-SD to use that technique and compare them with one another and with SSDP in terms of the quality of rules returned. Then, we will work from that initialization technique and try to improve the results by imbuing more knowledge of the problem in the way we initialize the populations.

Among the possible choices of rule evaluation measures, WRAcc is perhaps the most fundamental one in the context of knowledge discovery. High WRAcc means a rule has high accuracy, which is desirable in predictive tasks. For descriptive tasks, such as SD, accuracy alone is not enough, as we also need the rules to be as general as possible. LAVRAČ; FLACH; ZUPAN (1999) have shown that WRAcc provides this trade-off between generality and rule accuracy. They also demonstrated that WRAcc incorporates other measures, such as Sensitivity and Specificity. For these reasons, we have decided to use WRAcc as our evaluation measure when observing the quality of rules obtained.

5.1 Methodology

With an initialization technique and a quality measure selected, we now present the methodology used in this new set of experiments.

Since we are still interested in high dimensional SD, we use the same 10 datasets presented in Table 4.2. Because each algorithm selected for the experiments can only consider one target value at a time, for each dataset, the majority class was considered the target value, and thus labeled “positive”, while the instances in the remaining classes were all interpreted as belonging to a single “negative” class.

Because SSDP does not support datasets with continuous attributes, we need to convert every attribute in these datasets from continuous to discrete values. For this set of experiments, we used a single discretization technique based on quartiles. Therefore, for each attribute in each dataset, the values were broken down in 3 bins: the values lower than the first quartile, Q_1, were placed in the first bin, the ones between Q_1 and Q_3 were placed in the second and, finally, the values higher than Q_3 were placed in the third bin.

We subdivided the empirical testing into two phases. In the first phase, we run many different configurations of the algorithms in order to tune their parameters. Then, we execute each one again, with the best parameters found, and actually compare one another to analyze the impact of the initialization technique in the outcome of the algorithms. These two phases are explained in more detail below.

For each pair of evolutionary algorithm and initialization technique involved in the experiments, we first tune the algorithm’s parameters in the hope that our quality analysis of each algorithm’s output will not be affected by a bad choice of parameters. The parameters tuned were: mutation and crossover probabilities, number of evaluations (the stopping criteria for MESDIF and NMEEF-SD) and, finally, population length. This tuning phase was not applied to SSDP since it does not expose those parameters to the user. For this tuning phase, a grid search is performed with the following variations for each parameter:

- the mutation probability varied between 0.01, 0.05, 0.1 and 0.2;
- the crossover probability varied between 0.6, 0.7, 0.8 and 0.9;
- the number of evaluations varied between 10000, 40000, 70000 and 100000;
- the population length varied between 100, 500, 1000 and 2000.

Consequently, there are $4 \times 4 \times 4 \times 4 = 256$ different configurations to test in our grid search. In addition to that, each configuration is executed 30 times in each dataset. Therefore, there is a total of $256 \times 30 \times 10 = 76800$ executions for each pair of algorithm and initialization technique.

Due to the limited computational resources available during this research, we reduced the number of attributes in the datasets for this tuning phase in order to decrease the running time of each algorithm’s execution. From a dataset’s list of attributes, 1000 were selected according to their mutual information regarding the class attribute. This was done using the Python package scikit-learn’s `SelectKBest` and `mutual_info_classif` functions (PEDREGOSA et al., 2011).

After executing each configuration 30 times on each of the 10 datasets, we used the 300 samples to calculate the average ranking of each configuration, and decided for the one with the best average rank.

Then, having found the best configuration for each pair of algorithm and initialization technique, we execute that configuration again in the complete versions of the datasets (with all the attributes) for our comparison phase, which is explained next.

For comparing the pairs of algorithms and initialization techniques with one another, we run each 30 times in each dataset and compute the WRAcc, forming a sample of 300 observations. Then, following the suggestions in DEMŠAR (2006), we execute a Friedman statistical test (FRIEDMAN, 1937) to determine if there is statistically significant difference between the considered pairs and if so, execute a Nemenyi post-doc test (NEMENYI, 1963) to determine which ones are statistically different. In both tests, we use significance level of $\alpha = 5\%$.

In addition to the Nemenyi post-doc test, we also analyze the data obtained from our experiments. To that end, we take the average WRAcc of the returned set of rules. With that, we have a WRAcc value for each of the 300 executions (30 in each of the 10 datasets). Then, we also present the average and standard deviation of the 30 WRAcc scores each algorithm obtained in each dataset to illustrate our analysis.

5.1.1 Non-tuned algorithm parameters

Because we did not have enough computational resources to search for the best configuration of every parameter in each algorithm, in this subsection, we go through the values with which we configured the remaining parameters in each algorithm.

As SSDP has only two parameters to configure, namely, the number of rules to be returned and the fitness measure. We did not tune any of those parameters. Instead, we set the number of rules to 10 and the fitness measure as the Q_g metric with $g = 1$, as that is the default configuration in SSDP's implementation provided by the authors.

Both NMEEF-SD and MESDIF have a default implementation provided by their authors available in the KEEL software package. We use the default configuration, provided by KEEL, for the non-tuned parameters. Specifically, for NMEEF-SD, we set the rule representation to "canonical" and the fitness measures as sensitivity and WRAcc. For MESDIF, we also set the rule representation to "canonical" and the fitness measures to sensitivity, WRAcc and fuzzy confidence.

It is important to note that NMEEF-SD does not have a configurable number of returned rules: it returns the Pareto front of the last generation. MESDIF, on the other hand, has an elite population whose length was configured to 10.

Every time a Beta distribution was needed by an initialization technique, we used the parameters $\alpha = 1$ and $\beta = 81$ obtained in the previous chapter.

5.2 The Beta initialization

In this section we explore the quality of the results obtained by the MESDIF and NMEEF-SD algorithms with the Beta initialization (TORREÃO; VIMIEIRO, 2018) presented in Chapter 4.

First, we need to find the best parameter configuration for each algorithm. Following our methodology, we found the parameters described in the first two rows of Table 5.1. The remaining rows of Table 5.1 describe the best parameters found for other initialization techniques presented later in this chapter. We will refer the reader back to this table as we present them.

Table 5.1: Configured parameters found by grid search.

Technique	# evaluations	pop. length	mutation prob.	crossover prob.
MESDIF Beta	100,000	2,000	0.2	0.6
NMEEF-SD Beta	70,000	2,000	0.01	0.7
MESDIF Pos. Cov.	100,000	2,000	0.2	0.6
NMEEF-SD Pos. Cov.	100,000	2,000	0.01	0.9
MESDIF Roulette	100,000	2,000	0.2	0.6
NMEEF-SD Roulette	70,000	2,000	0.05	0.9

As the reader can see, for both algorithms, the best value for population length is 2000. This is expected, as one way of dealing with high dimensional problems is to increase the population length (KAZIMIPOUR; LI; QIN, 2013). However, it is not possible to increase the population length indefinitely, as that will eventually exhaust the available computational and time resources.

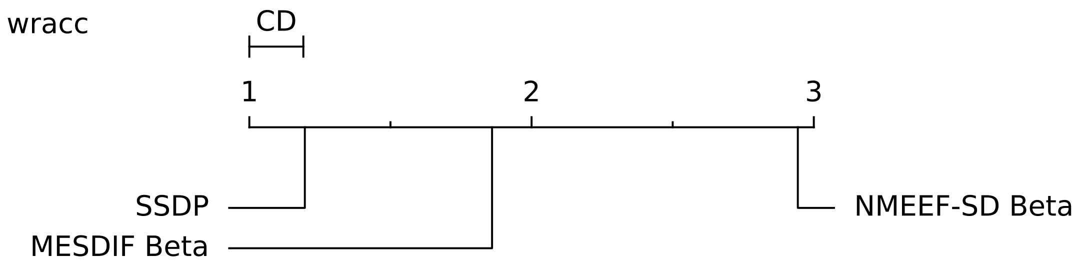
From the obtained number of evaluations, the reader can see that both algorithms needed large numbers. This is also expected as that gives the algorithms more generations to search for the optimal rules.

Finally, from the mutation and crossover probability results, it is interesting to observe that the algorithms had fairly opposite configurations: MESDIF was configured to a more exploratory behavior (from a higher mutation probability), while NMEEF-SD was configured to a more exploitative behavior (even though the crossover probabilities are similar).

Following our methodology, now that we have tuned values for the algorithm's parameters, we move to comparing the results in terms of WRAcc. Since we executed each algorithm 30 times in each of the 10 datasets, we now have 300 samples for each. We also execute SSDP with the parameters configured as explained in Section 5.1.1. We proceed with a Friedman test to see if there is significant difference between the three techniques. The obtained p-value was $5.157E^{-102}$. We are using significance level of $\alpha = 5\%$, so we can conclude that there is indeed significant difference between the average quality of rules returned by the three techniques.

Then, we must run a Nemenyi post-doc test to verify which techniques are different from one another. For that, we plot the critical difference graph, which is shown in Figure 5.1.

Figure 5.1: Nemenyi critical difference graph comparing the MESDIF and NMEEF-SD techniques modified with the Beta initialization along with the unmodified SSDP technique. The quality measure used is the WRAcc



The reader can observe in Figure 5.1 that all three techniques are different. Also, SSDP was the best ranking technique, followed by MESDIF and NMEEF-SD. This is an interesting result as NMEEF-SD had been reported to have better performance than MESDIF on low dimensional (CARMONA et al., 2009) and also high dimensional (TORREÃO; VIMIEIRO, 2018) SD problems. This contradiction could be due to the difference in the datasets used, discretization techniques, as well as the configuration of parameters. To investigate this, it would be necessary to experiment further with these techniques.

Table 5.2 complements our analysis. The reader can see from the first four columns in the table that MESDIF, using the Beta initialization technique, obtained results close to SSDP in some datasets, having a higher average WRAcc score in the Alon and Gravier datasets. This is significant, as we have taken an algorithm that did not even converge on high dimensional SD problems and, by simply changing the way it initializes the population, we have brought its results close to an algorithm designed specifically for such high dimensional SD instances.

Furthermore, the fact that SSDP initializes its population with the lowest possible number of attribute-value pairs and it achieved the best overall results in our experiments, further corroborates our hypothesis. Indeed, our results suggest that by lowering the average number of attribute-value pairs in the initial population we can, in fact, improve the results of EAs in high dimensional SD problems.

We established in Chapter 2 that good solutions to SD are simple rules that must be humanly comprehensible, and as such, must be made out of few attribute-value pairs. Through the Beta initialization technique, we used that knowledge of SD to bias the initial population to regions of the search space which we considered to have high potential. The next section follows this idea further by creating a new initialization technique with more knowledge of the problem at hand to see if we can get better results still.

5.3 Positive coverage-based Beta initialization

Following the trend set by the Beta initialization, in this section we present an initialization technique which incorporates more knowledge about SD problems.

Table 5.2: Average and Standard Deviation values of WRAcc for each dataset across the SSDP, MESDIF and NMEEF-SD algorithms. The suffix “Beta” means the column regards the modified version of the algorithm with the Beta initialization technique. The “Cov” suffix means the algorithm was modified with the Positive coverage-based Beta initialization. Finally, the “Rou” suffix means the algorithm was modified with the roulette selection with positive coverage Beta initialization. In each cell, the first number is the average WRAcc score, while the number after the “±” is the respective standard deviation. The best score in each dataset is displayed in bold for the reader’s convenience.

Dataset	**SSDP**	**MESDIF Beta**	**NMEEF Beta**	**MESDIF Cov**	**NMEEF Cov**	**MESDIF Rou**	**NMEEF Rou**
alon.quartile	0.0943±0.004	**0.0972±0.005**	0.0719±0.015	0.0946±0.009	0.0785±0.012	0.0969±0.008	0.0815±0.008
burczynski.quartile	**0.1055±0.000**	0.0749±0.010	0.0349±0.018	0.0755±0.008	0.0366±0.016	0.0779±0.009	0.0408±0.015
chiaretti.quartile	**0.0877±0.000**	0.0660±0.004	0.0338±0.011	0.0663±0.004	0.0389±0.012	0.0661±0.004	0.0432±0.010
chin.quartile	**0.1072±0.001**	0.0780±0.008	0.0356±0.012	0.0801±0.010	0.0422±0.016	0.0813±0.009	0.0425±0.012
christensen.quartile	0.1211±0.001	0.1336±0.004	0.1157±0.012	**0.1353±0.003**	0.1204±0.012	0.1342±0.004	0.1197±0.012
gravier.quartile	**0.0820±0.001**	0.0750±0.003	0.0605±0.009	0.0749±0.003	0.0633±0.007	0.0740±0.002	0.0661±0.009
nakayama.quartile	**0.1010±0.002**	0.0791±0.007	0.0378±0.017	0.0780±0.007	0.0332±0.014	0.0779±0.006	0.0400±0.014
sun.quartile	**0.1127±0.001**	0.0849±0.008	0.0316±0.022	0.0868±0.007	0.0315±0.019	0.0857±0.008	0.0334±0.018
tian.quartile	**0.0538±0.001**	0.0413±0.003	0.0265±0.008	0.0408±0.004	0.0261±0.009	0.0403±0.003	0.0250±0.008
yeoh.quartile	**0.1451±0.001**	0.1053±0.010	0.0494±0.031	0.1061±0.008	0.0498±0.027	0.1080±0.008	0.0439±0.016

In addition to being simple and humanly comprehensible, it is also desired that the instances in a subgroup have an unusual statistical distribution with regards to a particular target variable in which we are interested in, so good subgroups will have higher frequency of instances belonging to the positive class.

Therefore, for the next initialization technique, in addition to lowering the average number of attribute-value pairs in the generated individuals, the technique will also make sure that every individual in the starting population will cover at least one of the instances belonging to the positive class.

Algorithm 5 illustrates the pseudo-code for this approach which is called “Positive coverage-based Beta initialization”. The technique is fairly similar to the Beta initialization. It also generates individuals with a number of attribute-value pairs drawn from a Beta distribution. It also selects attributes randomly (in uniform fashion) to build the pairs. However, instead of randomly selecting a value to pair with each attribute, it instead will always copy the value from one of the instances belonging to the positive class.

Algorithm 5 The proposed positive coverage-based beta initialization technique

```
Require: $\boldsymbol{E}$: set of instances
Require: $\boldsymbol{A}$: set of attributes
Require: $\alpha$: the $\alpha$ parameter for the beta distribution
Require: $\beta$: the $\beta$ parameter for the beta distribution
 1: function POS-COV-BETA-INITIALIZE(popSize, $\boldsymbol{E}$, $\boldsymbol{A}$, $\alpha$, $\beta$)
 2:     $\boldsymbol{P} \leftarrow \emptyset$                                                  ▷ Population
 3:     while (popSize $-$ $|\boldsymbol{P}|$) $\geq |\boldsymbol{E}^+|$ do
 4:         for $\boldsymbol{e} \in \boldsymbol{E}^+$ do
 5:             $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{$ COV-BETA-GENERATE($\boldsymbol{e}$, $\boldsymbol{A}$, $\alpha$, $\beta$) $\}$
 6:         end for
 7:     end while
 8:     for $i \leftarrow |\boldsymbol{P}|$ to popSize do
 9:         $\boldsymbol{e} \leftarrow$ UNIFORM-RANDOM-SELECT($\boldsymbol{E}^+$)        ▷ select random element from $\boldsymbol{E}^+$
10:         $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{$ COV-BETA-GENERATE($\boldsymbol{e}$, $\boldsymbol{A}$, $\alpha$, $\beta$) $\}$
11:     end for
12:     return $\boldsymbol{P}$
13: end function
14: function COV-BETA-GENERATE($\boldsymbol{e}$, $\boldsymbol{A}$, $\alpha$, $\beta$)
15:     $\boldsymbol{I} \leftarrow \emptyset$                                                  ▷ Individual
16:     $rand \leftarrow$ BETA-RANDOM($\alpha$, $\beta$)
17:     $n \leftarrow rand \times |\boldsymbol{A}|$                 ▷ number of attribute-value pairs for this individual
18:     while $|\boldsymbol{I}| < n$ do
19:         $attr \leftarrow$ UNIFORM-RANDOM(1, $|\boldsymbol{A}|$)
20:         $v_{val}^{attr} \leftarrow v_x^{attr} \in \boldsymbol{a}_{attr} \mid \langle attr, v_x^{attr} \rangle \in \boldsymbol{e}$          ▷ value of attribute $\boldsymbol{a}_{attr}$ in instance $\boldsymbol{e}$
21:         $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{\langle attr, v_{val}^{attr} \rangle\}$
22:     end while
23:     return $\boldsymbol{I}$
24: end function
```

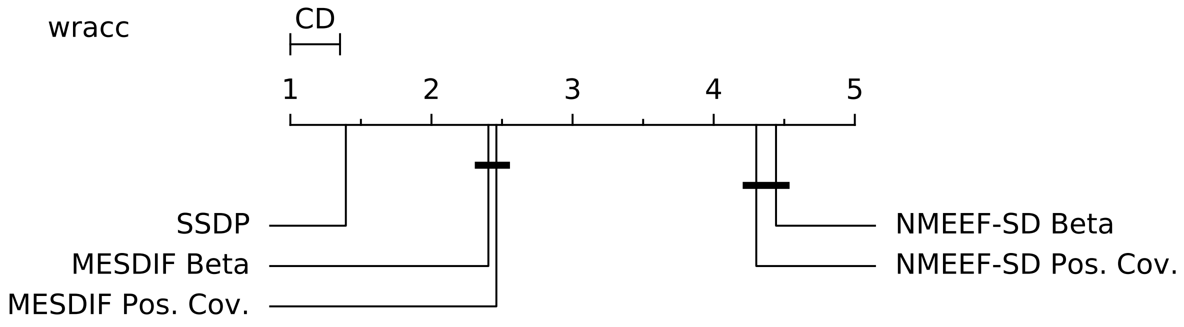
In case there are enough individuals to go through all of the instances with a positive class label, it will do so to guarantee all positive instances will be covered by the initial population. If there is more instances than individuals, the technique selects instances randomly.

Following the steps in the methodology, we replaced the initialization technique in both MESDIF and NMEEF-SD with this new one. Then, we tuned the parameters using the datasets with 1000 attributes. The resulting configuration is shown in the third and fourth rows of Table 5.1. This time we see more clearly that MESDIF relies more on exploring the search space, while NMEEF-SD requires a higher probability of crossover, presenting a more exploitative behavior.

Finally, we moved to the comparison phase of our methodology, where we first executed a Friedman test to see if there is any significant difference between the positive coverage-based Beta initialization version of MESDIF and NMEEF-SD when compared to one another, to the Beta initialization versions of each and to the SSDP algorithm.

The Friedman test's p-value was $6.94E^{-181}$, so there was significant difference between the 5 methods. Thus, we needed to run a Nemenyi post-doc test. Figure 5.2 illustrates those results.

Figure 5.2: Nemenyi critical difference graph comparing the MESDIF and NMEEF-SD techniques modified with the Beta and positive coverage-based initialization techniques along with the unmodified SSDP algorithm. The quality measure used is the WRAcc



As the reader can see in Figure 5.2, there was no significant difference between the positive coverage-based Beta initialization and the former technique. There was, however, significant differences between each EA, with the best ranking algorithm being SSDP. Therefore, we can conclude that by biasing the initial population individuals to cover the positive labeled instances in the dataset, we obtained no significant improvement from just reducing the average number of attribute-value pairs in the starting individuals.

As shown in Table 5.2 we can see that the average WRAcc scores obtained by both MESDIF and NMEEF-SD configured with the positive coverage-based Beta initialization are indeed very similar to the ones obtained by those algorithms configured with the Beta initialization.

5.4 Roulette selection with positive coverage Beta initialization

Following up with the idea of introducing more knowledge about the SD problem in order to bias the initial population and, hopefully, increase the quality of the resulting rules, in this section, we start biasing the initial population towards the attribute-value pairs with higher WRAcc scores.

This new initialization technique builds upon our last strategies. Therefore, it will use a Beta distribution to bias the average number of attribute-value pairs to lower values, and also make sure that every individual in the starting population covers at least one instance belonging to the positive class. Instead of selecting the attributes from a uniform distribution, just as the positive coverage-based Beta initialization approach, this time we will use the WRAcc as fitness function and choose the attributes using Fitness Proportionate Selection, also known as Roulette selection (LUKE, 2013).

Just like the previous initialization technique, this approach will try to generate individuals covering all instances belonging to the positive class, if there are enough individuals in the population. If the number of positive instances is greater than the population length, then the technique will pick positive instances at random. Once a number of attribute-value pairs has been obtained from a Beta distribution and we have selected a positive instance to be covered, the technique will use Roulette selection to pick one of the attribute-value pairs in the positive instance until we have reached the desired number of pairs for this individual. The technique generates individuals this way until the population has reached the desired length. This technique is illustrated in Algorithm 6.

Following our methodology, we once again replaced the initialization technique in the MESDIF and NMEEF-SD algorithms. The first step, then, is to tune the parameters. The resulting configuration is shown in the last two rows of Table 5.1. The configured parameters are very similar to the ones obtained by the other initialization techniques.

Next, in the comparison phase, we executed the Friedman test to see if there was any significant difference between the quality of the rules returned by MESDIF and NMEEF-SD modified to use this new technique when compared to the SSDP algorithm and the previously modified versions of MESDIF and NMEEF-SD. The p-value obtained was $4.04E^{-257}$. So there is indeed significant difference between the 7 approaches. Therefore, we move to the Nemenyi test results presented in Figure 5.3.

As the reader can see from the figure, the difference in the average WRAcc scores obtained by the MESDIF and NMEEF-SD algorithms using the new initialization technique was not significantly different from the ones obtained by each algorithm with the previous initialization techniques. Therefore, we can conclude that further biasing the initial population towards the best attribute-value pairs did not result in any improvement in terms of output rule quality.

Table 5.2 reinforces that conclusion. The average WRAcc scores for MESDIF using the

Algorithm 6 The proposed positive coverage-based beta initialization with Roulette selection.

```
Require: $\boldsymbol{E}$: set of instances
Require: $\boldsymbol{A}$: set of attributes
Require: $\alpha$: the $\alpha$ parameter for the beta distribution
Require: $\beta$: the $\beta$ parameter for the beta distribution
 1: function ROULETTE-BETA-INITIALIZE(popSize, $\boldsymbol{E}$, $\boldsymbol{A}$, $\alpha$, $\beta$)
 2:     $\boldsymbol{P} \leftarrow \emptyset$                                         ▷ Population
 3:     while (popSize $- |\boldsymbol{P}|) \geq |\boldsymbol{E}^+|$ do
 4:         for $\boldsymbol{e} \in \boldsymbol{E}^+$ do
 5:             $r \leftarrow$ BUILD-ROULETTE($\boldsymbol{e}$) ▷ Builds a Roulette with the attribute-value pairs in $\boldsymbol{e}$
 6:             $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{$ ROULETTE-BETA-GENERATE($r$, $\boldsymbol{A}$, $\alpha$, $\beta$) $\}$
 7:         end for
 8:     end while
 9:     for $i \leftarrow |\boldsymbol{P}|$ to popSize do
10:         $\boldsymbol{e} \leftarrow$ UNIFORM-RANDOM-SELECT($\boldsymbol{E}^+$)          ▷ select random element from $\boldsymbol{E}^+$
11:         $r \leftarrow$ BUILD-ROULETTE($\boldsymbol{e}$)    ▷ Builds a Roulette with the attribute-value pairs in $\boldsymbol{e}$
12:         $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{$ ROULETTE-BETA-GENERATE($r$, $\boldsymbol{A}$, $\alpha$, $\beta$) $\}$
13:     end for
14:     return $\boldsymbol{P}$
15: end function
16: function ROULETTE-BETA-GENERATE($r$, $\boldsymbol{A}$, $\alpha$, $\beta$)
17:     $\boldsymbol{I} \leftarrow \emptyset$                                         ▷ Individual
18:     $rand \leftarrow$ BETA-RANDOM($\alpha$, $\beta$)
19:     $n \leftarrow rand \times |\boldsymbol{A}|$                 ▷ number of attribute-value pairs for this individual
20:     while $|\boldsymbol{I}| < n$ do
21:         $\langle attr, v_{val}^{attr} \rangle \leftarrow$ ROULETTE-SELECT($r$)
22:         $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{\langle attr, v_{val}^{attr} \rangle\}$
23:     end while
24:     return $\boldsymbol{I}$
25: end function
```

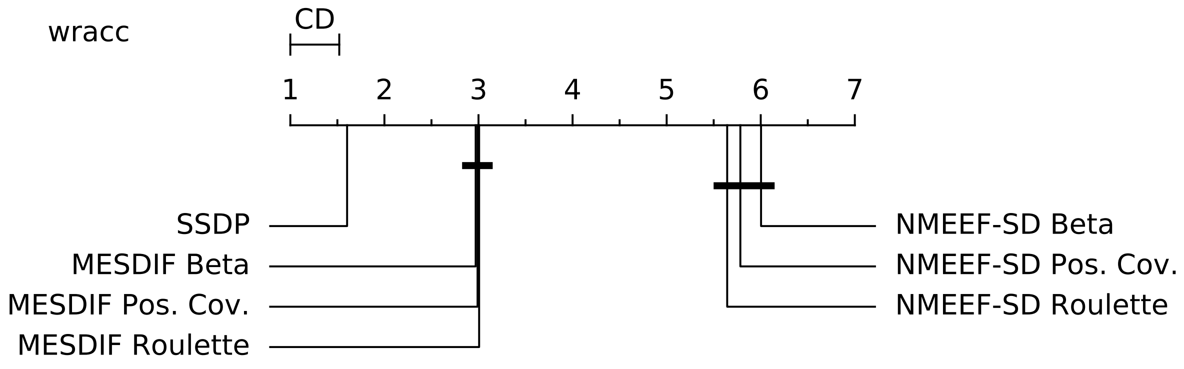
initialization technique with roulette selection had very close results when compared to either positive coverage-based initialization or plain Beta initialization techniques. For NMEEF-SD, although the average WRAcc scores for this new initialization technique were higher than the previous two in all but the last two datasets, we can see they are still very close and the Nemenyi statistical test has confirmed that the difference is not statistically significant.

5.5 Final remarks

In this chapter, we have analyzed the quality, in terms of WRAcc scores, of the rules obtained by the MESDIF and NMEEF-SD algorithms when their respective initialization techniques were replaced by our proposed Beta technique. We have also presented new ways to bias the population initialization in different ways in an attempt to improve the quality of the resulting rules.

The results of our empirical tests show that there was no improvement on the quality of the

Figure 5.3: Nemenyi critical difference graph comparing the MESDIF and NMEEF-SD techniques modified with each initialization technique presented in this text, along with the unmodified SSDP algorithm. The quality measure used is the WRAcc



obtained rules after biasing the initial population further than simply lowering the average number of attribute-value pairs in the generated individuals, as was proposed initially in Chapter 4.

The fact that the unmodified version of the SSDP algorithm obtained the best quality results does not rule out our hypothesis, but actually further reinforces it, as its initialization technique generates all individuals with just one attribute-value pair.

Therefore, it stands to reason that, indeed, biasing the initial population to individuals containing few attribute-value pairs can lead EAs to higher quality results in high dimensional SD problems. Moreover, further biasing the initial population has proven to be fruitless.

6

Conclusion

Subgroup Discovery (KLÖSGEN, 1996; WROBEL, 1997) is a descriptive data mining technique, based on supervised learning. It has many real world applications ranging from describing risk groups for the Coronary Heart Disease (LAVRAČ, 2005), using gene expression data, to detecting defective software modules (RODRÍGUEZ et al., 2012) and improving policies for trade fairs in industrial marketing (JESUS et al., 2007).

SD's objective is to search for statistically interesting subsets of instances in the dataset. By "interesting", it is meant being as large as possible and having an unusual statistical distribution regarding a certain property of interest.

Because subgroups must be described in explicit symbolic form and be humanly interpretable, they are usually represented as propositional rules, and their quality is assessed using measures borrowed from predictive and association rule learning (LAVRAČ, 2005).

These characteristics have lead researchers to use general purpose optimization algorithms for SD. In particular, the research community has given special attention to EAs for this task (CARMONA et al., 2014).

Among the state of the art EAs for SD, MESDIF (JESUS; GONZALEZ; HERRERA, 2007) and NMEEF-SD (CARMONA et al., 2009) have been shown to work well in low dimensional problems. However, when introducing SSDP, an EA designed specifically for high dimensional SD problems, PONTES; VIMIEIRO; LUDERMIR (2016) reported that these algorithms performed very poorly in high dimensional datasets.

Observing the three algorithms, it stands out that though having very similar mutation and crossover operators, these algorithms have very distinct initialization techniques. MESDIF generates an individual by appending an attribute-value pair for each possible attribute in the dataset, while NMEEF-SD limits that to 25% of the available attributes for 75% of the generated individuals in the starting population. SSDP, on the other hand, initializes its population with every possible individual containing only a single attribute-value pair.

This distinction, together with the knowledge that the rules discovered by these algorithms must be simple enough for a human to read, raised the research question for this work: since rules with too many attribute-value pairs are too complicated for a human reader and, therefore, inappropriate for SD, could changing EAs to use an initialization technique which bias the initial population towards individuals with a small number of attribute-value pairs increase the performance of said algorithms in high dimensional SD problems?

To answer that question, we first broke down “performance” in terms of convergence frequency and output rule quality, using the WRAcc (LAVRAČ; FLACH; ZUPAN, 1999) evaluation measure for the former. Then, we selected two of the state of the art EAs with source code readily available online, allowing us to just modify the initialization technique. Next, we introduced a new initialization technique that did not necessarily restrict the initial population to individuals with few attribute-value pairs, but biased the population towards that characteristic using random numbers drawn from specific Beta distributions. Subsequently, we selected high dimensional datasets on which to make empirical tests. Finally, we analyzed the results obtained from a convergence frequency and rule quality perspective.

Throughout our experiments we observed that, indeed, with the replacement initialization technique both algorithms were able to achieve higher convergence frequencies. In fact, the original version of MESDIF converged in 0% of the executions in the tested high dimensional datasets, while NMEEF-SD’s original version converged in less than 25% of its executions. By simply replacing their initialization techniques for our proposed “Beta initialization”, not only did both algorithms converge in most executions, but both modified algorithms had a remarkably similar convergence frequency when configured to the same initialization technique. This further corroborates our conclusion that it was in fact the initialization technique that was responsible for this improvement.

In terms of rule quality we observed that, for both MESDIF and NMEEF-SD, the modified versions achieve positive scores of WRAcc. MESDIF in particular, had results very similar to that of SSDP, and better average WRAcc scores in two of the tested high dimensional datasets. This is significant, as we started from an algorithm that did not even converge in high dimensional datasets, and were able to bring its results close to an algorithm designed specifically for these conditions, by simply biasing the initial population towards individuals with small numbers of attribute-value pairs.

Furthermore, the fact that SSDP still had the better overall results in terms of WRAcc only corroborates our hypothesis since it is the algorithm that restricts the initial population to individuals with the lowest possible number of attribute-value pairs.

In fact, because the observed results seemed to suggest that incorporating more knowledge about the specifics of SD into the initialization technique could result in even better results, we tested the algorithms with two other proposed initialization techniques. In our second technique, the initial population was biased towards covering the instances belonging to the positive class. In addition to covering positive instances, the third also biased the initial population towards the attribute-value pairs with higher WRAcc scores.

The obtained results, however, showed that, for both algorithms, there was no significant difference in the output rule quality when replacing their initialization techniques with either one of the three techniques proposed. Therefore, we can conclude that biasing the initial population further than just towards individuals with a small number of attribute-value pairs yielded no significant improvement to the output rule quality.

Finally, we confirmed, through the empirical tests described in this document, our hypothesis that, by biasing the initial population towards individuals with a small number of attribute-value pairs, we can improve the performance of EAs in high dimensional SD problems.

Because our initialization technique does not restrict the generated individuals to having few attribute-value pairs, but rather bias them in that direction, we intend to explore, in future works, the impact of using this initialization technique equipped with distributions other than Beta, such as Chaotic distributions.

It is also our intention to investigate whether or not it is possible to further improve the performance of EAs in high dimensional SD problems by replacing the mutation and crossover operators. Once we apply the Beta initialization technique, we know the general region in the search space where the population starts, so perhaps we can build crossover and mutation operators which leverage this knowledge. Many mutation operators, for instance, allow the individual to grow in the number of attribute-value pairs, which might be counter-productive since we are starting the population with a small number of attribute-value pairs per individual on purpose.

References

ALCALÁ-FDEZ, J. et al. KEEL: a software tool to assess evolutionary algorithms for data mining problems. **Soft Computing**, [S.l.], v.13, n.3, p.307–318, Feb 2009.

ATZMUELLER, M.; PUPPE, F. SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery. In: KNOWLEDGE DISCOVERY IN DATABASES: PKDD 2006, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2006. p.6–17.

CARMONA, C. J. et al. Non-dominated Multi-objective Evolutionary Algorithm Based on Fuzzy Rules Extraction for Subgroup Discovery. In: HYBRID ARTIFICIAL INTELLIGENCE SYSTEMS, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2009. p.573–580.

CARMONA, C. J. et al. Overview on Evolutionary Subgroup Discovery: analysis of the suitability and potential of the search performed by evolutionary algorithms. **Wiley Int. Rev. Data Min. and Knowl. Disc.**, New York, NY, USA, v.4, n.2, p.87–103, Mar. 2014.

CARMONA, C.; JESUS, M. del; HERRERA, F. A Unifying Analysis for the Supervised Descriptive Rule Discovery via the Weighted Relative Accuracy. **Know.-Based Syst.**, Amsterdam, The Netherlands, The Netherlands, v.139, n.C, p.89–100, Jan. 2018.

DEMŠAR, J. Statistical Comparisons of Classifiers over Multiple Data Sets. **J. Mach. Learn. Res.**, [S.l.], v.7, p.1–30, Dec. 2006.

FRIEDMAN, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. **Journal of the American Statistical Association**, [S.l.], v.32, n.200, p.675–701, 1937.

GAMBERGER, D.; LAVRAČ, N. Expert-guided Subgroup Discovery: methodology and application. **J. Artif. Int. Res.**, USA, v.17, n.1, p.501–527, Dec. 2002.

HELAL, S. Subgroup Discovery Algorithms: a survey and empirical evaluation. **Journal of Computer Science and Technology**, [S.l.], v.31, n.3, p.561–576, May 2016.

HERRERA, F. et al. An overview on subgroup discovery: foundations and applications. **Knowledge and Information Systems**, [S.l.], v.29, n.3, p.495–525, Dec 2011.

JESUS, M. J. del et al. Evolutionary Fuzzy Rule Induction Process for Subgroup Discovery: a case study in marketing. **IEEE Transactions on Fuzzy Systems**, [S.l.], v.15, n.4, p.578–592, Aug 2007.

JESUS, M. J. del; GONZALEZ, P.; HERRERA, F. Multiobjective Genetic Algorithm for Extracting Subgroup Discovery Fuzzy Rules. In: IEEE SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE IN MULTI-CRITERIA DECISION-MAKING, 2007. **Anais...** [S.l.: s.n.], 2007. p.50–57.

KAVŠEK, B.; LAVRAČ, N.; JOVANOSKI, V. APRIORI-SD: adapting association rule learning to subgroup discovery. In: ADVANCES IN INTELLIGENT DATA ANALYSIS V, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2003. p.230–241.

KAZIMIPOUR, B.; LI, X.; QIN, A. K. Initialization methods for large scale global optimization. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION, 2013. **Anais...** [S.l.: s.n.], 2013. p.2750–2757.

KAZIMIPOUR, B.; LI, X.; QIN, A. K. A review of population initialization techniques for evolutionary algorithms. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION (CEC), 2014. **Anais...** [S.l.: s.n.], 2014. p.2585–2592.

KAZIMIPOUR, B.; LI, X.; QIN, A. K. Effects of population initialization on differential evolution for large scale optimization. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION (CEC), 2014. **Anais...** [S.l.: s.n.], 2014. p.2404–2411.

KLÖSGEN, W. Advances in Knowledge Discovery and Data Mining. In: FAYYAD, U. M. et al. (Ed.). . Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996. p.249–271.

KLÖSGEN, W.; MAY, M. Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database. In: PRINCIPLES OF DATA MINING AND KNOWLEDGE DISCOVERY, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2002. p.275–286.

LAVRAČ, N. Subgroup Discovery Techniques and Applications. In: PACIFIC-ASIA CONFERENCE ON ADVANCES IN KNOWLEDGE DISCOVERY AND DATA MINING, 9., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2005. p.2–14. (PAKDD'05).

LAVRAČ, N. et al. Decision Support Through Subgroup Discovery: three case studies and the lessons learned. **Machine Learning**, [S.l.], v.57, n.1, p.115–143, Oct 2004.

LAVRAČ, N. et al. Subgroup discovery with CN2-SD. **Journal of Machine Learning Research**, [S.l.], v.5, n.Feb, p.153–188, 2004.

LAVRAČ, N.; FLACH, P. A.; ZUPAN, B. Rule Evaluation Measures: a unifying view. In: INTERNATIONAL WORKSHOP ON INDUCTIVE LOGIC PROGRAMMING, 9., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 1999. p.174–185. (ILP '99).

LI, J.; WONG, L. Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. **Bioinformatics**, [S.l.], v.18, n.5, p.725–734, 2002.

LIU, X. et al. Discriminative pattern mining and its applications in bioinformatics. **Briefings in bioinformatics**, [S.l.], v.16, n.5, p.884–900, 2014.

LUCAS, T. et al. A New Evolutionary Algorithm for Mining Top-k Discriminative Patterns in High Dimensional Data. **Appl. Soft Comput.**, Amsterdam, The Netherlands, The Netherlands, v.59, n.C, p.487–499, Oct. 2017.

LUKE, S. **Essentials of Metaheuristics**. second.ed. [S.l.]: Lulu, 2013. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

MA, Z.; VANDENBOSCH, G. A. E. Impact of Random Number Generators on the performance of particle swarm optimization in antenna design. In: EUROPEAN CONFERENCE ON ANTENNAS AND PROPAGATION (EUCAP), 2012. **Anais...** [S.l.: s.n.], 2012. p.925–929.

NEMENYI, P. **Distribution-free Multiple Comparisons**. [S.l.: s.n.], 1963.

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

PONTES, T.; VIMIEIRO, R.; LUDERMIR, T. B. SSDP: a simple evolutionary approach for top-k discriminative patterns in high dimensional databases. In: BRAZILIAN CONFERENCE ON INTELLIGENT SYSTEMS (BRACIS), 2016. **Anais...** [S.l.: s.n.], 2016. p.361–366.

RODRÍGUEZ, D. et al. Searching for rules to detect defective modules: a subgroup discovery approach. **Information Sciences**, [S.l.], v.191, p.14 – 30, 2012. Data Mining for Software Trustworthiness.

SANTOS, M. A. S.; VIMIEIRO, R. A quantitative comparison of exhaustive discriminative pattern mining algorithms. In: ENCONTRO NACIONAL DE INTELIGêNCIA ARTIFICIAL E COMPUTACIONAL, 14. **Anais...** [S.l.: s.n.], 2017.

TORREÃO, V.; VIMIEIRO, R. Effects of Population Initialization on Evolutionary Techniques for Subgroup Discovery in High Dimensional Datasets. In: BRAZILIAN CONFERENCE ON INTELLIGENT SYSTEMS (BRACIS), 2018. **Anais...** [S.l.: s.n.], 2018. p.25–30.

WEISSTEIN, E. W. **Beta distribution**. [S.l.]: Wolfram Research, Inc., 2003.

WROBEL, S. An Algorithm for Multi-relational Discovery of Subgroups. In: FIRST EUROPEAN SYMPOSIUM ON PRINCIPLES OF DATA MINING AND KNOWLEDGE DISCOVERY, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 1997. p.78–87. (PKDD '97).

YEOH, E.-J. et al. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. **Cancer cell**, [S.l.], v.1, n.2, p.133–143, 2002.