# Hybrid feature selection approaches using metaheuristics for hierarchical classification

**Helen de Cássia Sousa da Costa Lima**


Universidade Federal de Ouro Preto
Departamento de Computação
Programa de Pós-Graduação em Ciência da Computação

Ouro Preto
2021

**Helen de Cássia Sousa da Costa Lima**

# Hybrid feature selection approaches using metaheuristics for hierarchical classification

Tese de doutorado apresentada ao Programa de Pós-graduação do Departamento de Computação da Universidade Federal de Ouro Preto para a obtenção do título de Doutora em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Marcone Jamilson Freitas Souza
Coorientador: Luiz Henrique de Campos Merschmann

Ouro Preto
2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

L732h Lima, Helen de Cassia Sousa da Costa.
Hybrid feature selection approaches using metaheuristics for hierarchical classification. [manuscrito] / Helen de Cassia Sousa da Costa Lima. - 2021.
71 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza.
Coorientador: Prof. Dr. Luiz Henrique de Campos Merschmann.
Tese (Doutorado). Universidade Federal de Ouro Preto. Departamento de Computação. Programa de Pós-Graduação em Ciência da Computação.
Área de Concentração: Ciência da Computação.

1. Classificação. 2. Wrapper. 3. Filtro. 4. Mineração de dados. I. Merschmann, Luiz Henrique de Campos. II. Souza, Marcone Jamilson Freitas. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO
PROGRAMA DE POS-GRADUACAO EM CIENCIA DA COMPUTACAO

**FOLHA DE APROVAÇÃO**

Helen de Cássia Sousa da Costa Lima

**Hybrid feature selection approaches using metaheuristics for hierarchical classification**

Tese apresentada ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Doutora em Ciência da Computação

Aprovada em 03 de setembro de 2021

Membros da banca

Prof. Dr. Marcone Jamilson Freitas Souza - Orientador - Universidade Federal de Ouro Preto
Prof. Dr. Luiz Henrique de Campos Merschmann - Universidade Federal de Lavras
Prof. Dr. Túlio Ângelo Machado Toffolo - Universidade Federal de Ouro Preto
Prof. Dr. Eduardo José da Silva Luz - Universidade Federal de Ouro Preto
Prof. Dr. Ricardo Cerri - Universidade Federal de São Carlos
Prof. Dr. Fernando Esteban Barril Otero - University o London

Prof. Dr. Marcone Jamilson Freitas Souza, orientador do trabalho, aprovou a versão final e autorizou seu depósito no Repositório Institucional da UFOP em 17/01/2022

Documento assinado eletronicamente por **Puca Huachi Vaz Penna**, **COORDENADOR(A) DE CURSO DE PÓS-GRADUACÃO EM CIÊNCIA DA COMPUTAÇÃO**, em 11/02/2022, às 13:31, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0275021** e o código CRC **0A8784F3**.



R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

# Agradecimentos

Em primeiro lugar agradeço a Deus, que me capacitou para a realização deste trabalho e me apontou o caminho para apreciar a jornada até aqui.

Ao meu marido, Marcos, por seu amor, companheirismo, renúncias e apoio na realização de sonhos que só brotaram no meu coração, mas que você soube respeitar do mesmo jeito.

Aos meus pais, Edna e Paulo, pelo amor, dedicação e por não medirem esforços pela minha educação. Espero sempre ter oportunidades de retribuir tudo que fizeram e fazem por mim.

Aos meus orientadores, Luiz e Marcone, por irem além de conhecimento compartilhado. Cada gesto de sabedoria e gentileza vindo de vocês contribuiu para me formar enquanto pesquisadora e orientadora. Perceber que transfiro para meus alunos tudo que recebi de vocês no doutorado é o maior legado dessa experiência.

Ao Fernando Otero, por aceitar me receber na University of Kent e ter feito contribuições essenciais para a defesa desta tese.

Aos amigos que fiz no período que vivi em Ouro Preto e na Inglaterra. Cada conversa honesta me fez ver o quão forte e resilientes vocês têm sido durante essa louca vida acadêmica.

Aos professores e técnicos do DECOM e do DECSI, pela disposição em ajudar sempre e por palavras de incentivo.

À CAPES e a UFOP, pelo apoio financeiro durante o período de doutorado sanduíche e afastamento.

*"O senhor... Mire veja: o mais importante e bonito, do mundo, é isto: que as pessoas não estão sempre iguais, ainda não foram terminadas - mas que elas vão sempre mudando. Afinam ou desafinam."*

*– João Guimarães Rosa (1908 - 1967),*
*in: Grande Sertão: Veredas.*

# Resumo

A seleção de atributos é uma etapa de pré-processamento amplamente difundida na área de mineração de dados. Um de seus objetivos é reduzir o número de atributos originais de uma base de dados para melhorar o desempenho de um modelo preditivo. No entanto, apesar dos benefícios da seleção de atributos para a tarefa de classificação, até onde sabemos, poucos estudos na literatura abordam a seleção de atributos para o contexto de classificação hierárquica. Este trabalho propõe duas abordagens principais de seleção híbrida de atributos supervisionada, combinando uma etapa filtro com uma *wrapper*, na qual um classificador hierárquico global avalia subconjuntos de atributos. A primeira abordagem usa a metaheurística Busca em Vizinhança Variável Geral com um ranqueamento de atributos construído com a medida Incerteza Simétrica Hierárquica. A segunda abordagem propõe uma adaptação da medida de seleção de atributos baseada em correlação adaptada para classificação hierárquica e utiliza o algoritmo *Best First Search* para pesquisar o espaço de subconjuntos de atributos. Doze bases de dados dos domínios de proteína e imagem foram usadas para realizar experimentos computacionais para validar o desempenho dos algoritmos propostos utilizando dois classificadores hierárquicos globais propostos na literatura. Testes estatísticos mostraram que o uso dos métodos de seleção de atributos propostos levaram a um desempenho preditivo consistentemente melhor ou equivalente ao obtido quando todos os atributos iniciais são utilizados, além do benefício de reduzir o número de atributos necessários, o que justifica a aplicação em cenários de classificação hierárquica.

**Palavras-chave:** Seleção de Atributos Híbrida. Classificação Hierárquica. Busca em Vizinhança Variável. *Wrapper*. Filtro. Seleção de Atributos baseada em correlação.

# Abstract

Feature selection is a widespread preprocessing step in the data mining field. One of its purposes is to reduce the number of original dataset features to improve a predictive model's performance. However, despite the benefits of feature selection for the classification task, as far as we are aware, few studies in the literature address feature selection for hierarchical classification context. This work proposes two main supervised hybrid feature selection approaches, combining a filter and a wrapper step, wherein a global model hierarchical classifier evaluates feature subsets. The first uses the General Variable Neighborhood Search metaheuristic and a feature ranking constructed with the Hierarchical Symmetrical Uncertainty measure. The second one proposes an extension of the Correlation-based Feature Selection measure for hierarchical classification and uses a Best First Search algorithm to search the feature subset space. We used twelve datasets from protein and image domains to perform computational experiments to validate the effect of the proposed algorithms on classification performance when using two global hierarchical classifiers proposed in the literature. Statistical tests showed that using our methods as a feature selection led to a predictive performance that is consistently better or equivalent to that obtained using all features, with the benefit of reducing the number of features needed, which justifies their use for the hierarchical classification scenario.

**Keywords:** Hybrid feature selection. Hierarchical classification. Variable Neighborhood Search. Wrapper. Filter. Correlation-based feature selection.

# List of Figures

# List of Tables

# List of Algorithms

# Contents

# Chapter 1

# Introduction

Data mining applications have become essential in recent years due to the massive increase in data generation and storage. The manipulation of data to transform it into understandable and advantageous information launches new research challenges.

Feature selection aims to identify as many relevant features as possible and decrease the cost requirements for processing data. Typically, data mining tasks use feature selection as a preprocessing step. In this paper, we will focus on feature selection approaches for the classification task. Therefore, we considered only datasets with labeled instances. Improving classifiers' predictive accuracy and reducing the classification's execution time are some of the benefits of feature selection (HAN; KAMBER; PEI, 2011).

Among data mining tasks, classification has received considerable attention from the scientific community (HAN; KAMBER; PEI, 2011). Classification predicts the class label(s) of examples based on the problem domain represented by its features. There are different complexity levels of classification problems in the literature. In traditional (flat) classification problems, one or more class labels are assigned to each dataset instance, and the classes are independent of each other. However, in many real applications, more complex classification problems exist, in which classes that label instances are organized into a hierarchical structure (SILLA; FREITAS, 2011) represented by a tree or a direct acyclic graph (DAG), so-called hierarchical classification problems.

Researches have proposed different methods to solve hierarchical classification problems, categorized into local or global approaches, according to how the method handles the class hierarchy. In the local approach, the classification is carried out using a set of flat classifiers. In contrast, the global approach uses a single classifier that considers the class hierarchy as a whole. Hierarchical classification methods may also be able to predict different numbers of paths of labels. A method can be restricted to predict only a single path of labels (single-label problem) or multiple paths of labels (multi-label problem).

## 1.1 Motivation

Despite the benefits of using feature selection methods as a preprocessing step for the classification task, many of the existing feature selection techniques in the literature cannot be directly applied to a hierarchical classification scenario. The initial efforts to solve feature selection for the hierarchical classification problem proposed to apply conventional feature selection techniques and construct classifiers by breaking down the hierarchical classification problem into several flat classification problems. This type of approach allowed them to use feature selection techniques and classification algorithms traditionally adopted in flat classification (KOLLER; SAHAMI, 1997; SECKER et al., 2010; PAES; PLASTINO; FREITAS, 2014).

Few recent approaches that also use a set of flat classifiers have proposed techniques based on recursive regularization that take into account the hierarchical information of classes (e.g., parent-child, sibling, and graph relations) (ZHAO et al., 2017; TUO; ZHAO; HU, 2019). In addition to structure information, another approach used a semantic description of class labels to select different feature subsets for each sub-classifier(HUANG; LIU, 2020). It is worth mentioning that none of them conducted experiments using global hierarchical classifiers. Other ranked-based methods have proposed to readjust some existing popular filter feature selection algorithms to take into account the hierarchical structure of classes (SLAVKOV et al., 2014; DIAS; MERSCHMANN, 2015).

In the literature, several studies propose modifications on existing flat classifiers to cope with the entire class hierarchy in a single step (CLARE; KING, 2003; CHEN; HU; TANG, 2009; SILLA; FREITAS, 2009; VENS et al., 2008; OTERO; FREITAS; JOHNSON, 2009; OTERO; FREITAS; JOHNSON, 2010; ZHENG; ZHAO, 2020), resulting in approaches that benefit from training a single classifier to solve hierarchical problems.. Given the relevance of global classifiers to the hierarchical classification scenario, one can see the importance of developing preprocessing techniques capable of dealing with the class hierarchy as a whole.

## 1.2 Goals and objectives

The goal of this thesis is to propose a feature selection approach specifically designed for global model hierarchical classifiers, dealing directly with the class hierarchy relations. To achieve this goal, we defined the following objectives:

- ❑ Combine filter measures and wrapper techniques to construct solutions to improve global hierarchical classifiers' predictive performance.
- ❑ Develop hybrid feature selection methods based on metaheuristics algorithms to efficiently search the feature subset space.

- ❏ Design a new search-based filter measure adapted for hierarchical classification problems capable of correctly evaluating the quality of feature subsets.
- ❏ Execute the wrapper evaluation only in strategic points of the search algorithms to reduce their computational cost.

## 1.3 Original contributions

To summarize, our major contributions in this thesis are as follows:

- ❏ We developed two hybrid feature selection methods based on the Variable Neighborhood Search (VNS) metaheuristic that combine the search-based step with a feature ranking constructed by the Hierarchical Symmetric Uncertainty ($SU_H$) measure, which evaluates features considering the hierarchical class structure.
- ❏ We designed a new Hierarchical Correlation-based Feature Selection (H-CFS) filter measure.
- ❏ We developed two hybrid feature selection methods based BFS and GA for global model hierarchical classifiers. These methods incorporate the H-CFS as an alternative fitness function to wrapper evaluations, reducing their computational costs.

## 1.4 Thesis organization

The remaining of this thesis is organized as follows. Chapter 2 presents an overview of hierarchical classification and feature selection. In Chapter 3 we describe the problem addressed in this work and present the related work. The proposed methods based on VNS metaheuristic are presented in Chapter 4, and the new approaches using the H-CFS measure are discussed in Chapter 5. Finally, conclusions and directions for future work are described in Chapter 6.

# Chapter 2

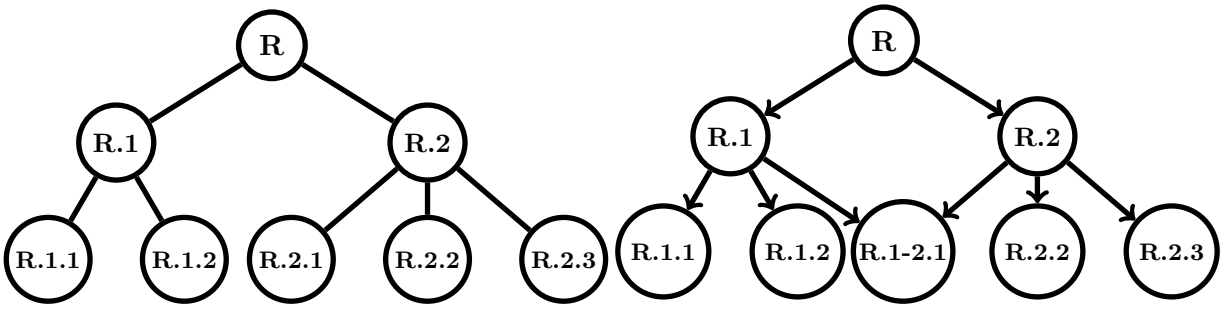
# Fundamentals

In this chapter we present throughout Sections 2.1.2 and 2.2 an overview on hierarchical classification and feature selection methods, respectively.

## 2.1 Hierarchical classification

Most classification studies in the data mining field are related to flat classification problems, in which the classes are independent of each other. However, in many real-world applications, the classes that label instances are organized into a hierarchical structure.

Different aspects can characterize hierarchical classification methods (SILLA; FREITAS, 2011). The first one is related to the type of hierarchical structure (tree or DAG) that the method can process. Fig. 1 presents a tree and a DAG example, where the nodes represent the classes, and the edges indicate a relationship between them. Basically, in a tree structure (Figure 1a), each node (class) can possess only one parent node, while in a DAG (Figure 1b), a child node (class) can have multiple parent nodes.




(a) Class hierarchy structured as tree (b) Class hierarchy structured as DAG

Figure 1 – Types of hierarchical structure

The second aspect is related to how deep in the class hierarchy the classification performs. A method can either perform mandatory leaf node prediction (MLNP) or non-mandatory leaf node predictions (NMLNP). In MLNP, the most specific class assigned to an instance must be one of the classes at a leaf node in the class hierarchy. Contrastingly,

in NMLNP, any class node in the hierarchy (internal or leaf) can be assigned to an instance.

The third aspect refers to the number of different paths of labels in the class hierarchy, in which the method can associate an instance. The methods may predict, to each instance, just a single path of labels in the class hierarchy (single-label problem), or they can be less restricted, predicting multiple paths of labels (multi-label problem).

Finally, the fourth aspect concerns how the classification method handles the class hierarchy. The classification methods can perform either flat or hierarchical classification (using a local or global model approach). In flat classification, the method ignores the class hierarchy and performs predictions considering only classes associated with leaf nodes. In the local model approach, the class hierarchy is explored through a local perspective, with the combination of classifiers that consider, in an isolated manner, different parts of the hierarchy. According to Silla and Freitas (2011), we can categorize local model approaches according to how they use the local information of the hierarchy structure and how they build their classifiers around it. There are three standard ways of using local information: local classifier per node, local classifier per parent node, and local classifier per the hierarchy level. The global model approach uses only one classifier, i.e., it builds a single model considering the class hierarchy as a whole.

In the literature, several works proposing modifications on existing flat classifiers to cope with the entire class hierarchy in a single step are available. Some examples of modifications of traditional flat classification algorithms are: HC4.5 (CLARE; KING, 2003) and HLC (CHEN; HU; TANG, 2009), modified versions of the C4.5; Global Model Naive Bayes (SILLA; FREITAS, 2009), a modified version of the Naive Bayes; CLUS-HMC (VENS et al., 2008), a method based on Predictive Cluster Trees; *h*Ant-Miner (OTERO; FREITAS; JOHNSON, 2009) and *hm*Ant-Miner (OTERO; FREITAS; JOHNSON, 2010), both adaptations of the Ant-Miner algorithm; and more recently the CSHCIC method (ZHENG; ZHAO, 2020), which integrates hierarchical classification and cost-sensitive learning to re-weight training data for the imbalanced class problem.

We used two global model hierarchical classifiers in this work, described in separate subsections in the following.

### 2.1.1 Global Model Naive Bayes

The flat Naive Bayes (NB) is an efficient and effective inductive learning algorithm for data mining (DUDA; HART, 1973). Despite its qualities, the flat NB is not designed to deal with hierarchical classification problems. For this reason, Silla and Freitas (2009) proposed the Global Model Naive Bayes (GMNB) algorithm, an adaptation of the flat NB to deal with the class hierarchy by considering the relationship between classes during the probabilities measurement.

Let $D = \{d^1, d^2, \ldots, d^t\}$ be a set of training instances, $A = \{A_1, A_2, \ldots, A_n\}$ be the set of predictive features of an instance $d^j \in D$, and let $C = \{c_1, \ldots, c_r\}$ be a set of classes that relate to each other through a hierarchical structure. Each instance $d^j$ is represented by its attribute vector $Y^j = \{y_1^j, y_2^j, \ldots, y_n^j\}$ with $y_i^j \in A_i$ and associated with a class $c_i \in C$.

Given a new instance $Y = \{y_1, y_2, \ldots, y_n\}$, where $y_1, y_2, \ldots, y_n$ are the predictive feature values associated with $A_1, A_2, \ldots, A_n$ respectively, the flat NB classifier simply assigns to this new instance the class $c_i$ associated with the maximum posterior probability, calculated as $P(c_i|Y) \propto P(Y|c_i)P(c_i)$, where $P(Y|c_i) = \prod_{j=1}^{n} P(y_j|c_i)$. Thus, the GMNB classifier's main difference is in calculating the probabilities $P(c_i)$ and $P(y_j|c_i)$, since it estimates them taking into account the class hierarchy.

More specifically, the GMNB considers that any instance of the class $c_i$ also belongs to all its parent classes. Considering the example in Figure 1a, if a training instance belongs to the class $R.1.2$, then it will be counted in the frequencies of all probabilities involving the class $R.1.2$ ($P(R.1.2)$ and $P(y_j|R.1.2)$) and also in all probabilities related to its parent class $R.1$ ($P(R.1)$ and $P(y_j|R.1)$). These adaptations allow the GMNB to predict classes at any level of the class hierarchy.

### 2.1.2 CLUS-HMC

The CLUS-HMC is a decision tree learner algorithm designed for hierarchical multi-label classification problems (VENS et al., 2008). It takes a set of instances and search for the best feature value test to be placed in a node. The algorithm then calls itself recursively to build a subtree for each cluster induced by the test using the training data. The best test is the one that maximally reduces the variance induced on the training instances. Maximizing variance reduction will maximize the homogeneity of the clusters, which improves the classification performance.

The variance is estimated according to Equation (1). If we take a set of instances $D$ and perform the arithmetic mean $\bar{v}$ of their label vectors (classes), the $j$th component of $\bar{v}$ will contain the proportion of instances in the set that are classified in the class $c_j$. The variance of a set of instances is defined as the mean square distance between the label vector $v_i$ of each instance and the mean label vector $\bar{v}$.

$$Var(D) = \frac{\sum_i d(v_i, \bar{v})^2}{|D|} \tag{1}$$

The distance used by CLUS-HMC is the weighted Euclidean distance, presented by Equation (2), where $v_{k,j}$ is the $j$th class of the class vector $v_k$ of a given instance $x_k$, and the class weights $w(c_j)$ decrease with the depth of the class in the hierarchy (e.g.,

$w(c_j) = w_0^{depth(c_j)}$, with $0 < w_0 < 1$). Thus, higher weights are assigned to class nodes at the smaller (more generic) levels in the hierarchy. This is natural, considering that similarities at smaller levels are more important than similarities at greater (deeper) levels.

$$d(v_1, v_2) = \sqrt{\sum_j w(c_j)(v_{1,j} - v_{2,j})^2} \tag{2}$$

Consider for example the class hierarchy shown in Figure 1a, and two examples $(Y^1, C_1)$ and $(Y^2, C_2)$ with $C_1 = \{R.1, R.2, R.2.2\}$ and $C_2 = \{R.2\}$. Using a vector representation with consecutive components representing membership of class R.1, R.1.1, R.1.2, R.2, R.2.1, R.2.2 and R.2.3, in that order, $d([1,0,0,1,0,1,0],[0,0,0,1,0,0,0]) = \sqrt{w_0 + w_0^2}$.

## 2.2 Feature selection

Feature selection has received increasing attention from researchers in recent years due to the continued rapid growth in data volume. Powerful as a preprocessing step, it selects a subset of predictive features to improve learning models' performance. Data containing irrelevant or redundant features can reduce classifiers' predictive capability and increase the classification processing time (HALL, 2000). Several research works have already shown that, in specific datasets, some of the features can be removed from the feature set without jeopardizing the predictive accuracy of the classifier (BLUM; LANGLEY, 1997). In practice, the use of feature selection in the classification task can result in the following benefits (LIU; MOTODA, 2007):

(i) Improvement of the predictive capability of classifiers.

(ii) Reduction of the running time spent in the classification learning process.

(iii) Development of simplified classification models, which allow for easier interpretation.

We can categorize feature selection methods according to different aspects. The first one is related to the use of the labeled class value. Feature selection methods can process datasets that have class values previously labeled, partially labeled, and non-labeled instances, leading to the development of supervised, semi-supervised, and unsupervised algorithms, respectively. A supervised feature selection algorithm determines the relevance of features by evaluating their existing correlation with the class feature. In this paper, we considered datasets with labeled instances. Therefore, we will focus on studies

that proposed feature selection approaches for the supervised learning context, specifically feature selection approaches for the classification task.

Another aspect is related to how the methods evaluate the quality of the predictive features. In this sense, we can consider different approaches, which, in general, can be categorized into embedded, filter, wrapper, or hybrid (involving possible combinations among embedded, filter, and wrapper) (LIU et al., 2010).

A method is categorized as a filter when it uses only intrinsic properties of the data. However, when a method uses a classifier to assess the quality of a given feature subset, it is categorized as a wrapper. Filter methods have the advantage of being independent of a classifier, generally faster than wrapper techniques. On the other hand, the wrapper approach usually has the advantage of reaching higher predictive performance than filters.

When we use an embedded feature selection approach, the classification model performs the feature selection simultaneously with its creation. Typical examples of these techniques are decision tree algorithms because they perform the selection of features placed into the nodes of the generated trees (COSTA et al., 2007; VENS et al., 2008; CHEN; HU; TANG, 2009).

### 2.2.1 Filter approaches

Filter approaches are independent of the classification algorithm that will be applied. They use the features' intrinsic properties (i.e., the "relevance" of the features) to evaluate the quality of features or subsets of features. Typically, one can divide techniques based on filter approaches into two groups: feature ranking-based approaches and search-based approaches.

The feature ranking-based approach applies statistical metrics to evaluate each feature individually, ranks features according to their relevance, and selects the top $k$ features from the ranked list (where $k$ is a predefined number). This approach's drawback is that it considers only one feature per evaluation (univariate method), ignoring the correlations between features. One feature that is irrelevant by itself can be significantly informative when considered together with other features (LIU; MOTODA, 2007). Examples of ranking-based methods are Information Gain Attribute Ranking (YANG; PEDERSEN, 1997), Symmetrical Uncertainty (SU) (LIU; MOTODA, 2007), Gain Ratio (LIU; MOTODA, 2007), and Chi-Squared (YANG; PEDERSEN, 1997).

The search-based approach considers the relationship between features in a feature subset (being a multivariate method), searching for the space of possible feature subsets. Each feature subset considered by the search method represents a candidate solution, which has its quality measured by an evaluation function. Assuming that the evaluation function penalizes redundant feature subsets, this approach has the advantage of feature redundancy elimination. On the other hand, this approach takes more time to generate and measure each feature subset's quality, which is usually slower than the univariate

approach. Recall that if there are $n$ possible features initially, then there are $2^n$ possible subsets, which turns the evaluation of every candidate feature subset prohibitive for all but a small fraction of the total number of possible subsets.

In this sense, one can apply various heuristic search strategies such as hill climbing and best first (LIU; MOTODA, 2007) to search the feature subset space in a reasonable time. Metaheuristic algorithms such as Simulated Annealing (SA) (DEBUSE; RAYWARD-SMITH, 1997), Genetic Algorithms (GA) (XUE et al., 2016), and Particle Swarm Optimization (PSO) (AGRAWAL et al., 2021) have also been applied efficiently as search-based feature selection approaches. Recently, researchers have explored strategies that design parallel algorithms to improve the running time of their feature selection approach, as proposed by Huang et al. (2019) for internet text classification. Examples of search-based methods are Correlation-Based Feature Selection (CFS) (HALL, 2000; JUNGJIT; FREITAS, 2015), and Consistency-based Feature Selection (LIU; SETIONO, 1996).

In the following subsections, we present two filter approaches that we used in this work. Section 2.2.1.1 describes the Hierarchical Symmetrical Uncertainty ($SU_H$) ranking-based approach, an adaptation of SU to hierarchical classification. Section 2.2.1.2 describes the CFS search-based method. Finally, in Section 2.2.1.3, we present the GA metaheuristic, used in Section 5.2 to search the feature subset space.

#### 2.2.1.1 Hierarchical Symmetrical Uncertainty

The classical SU filter approach is a non-linear measure of correlation widely used to evaluate features that combines the Entropy (E) and the Information Gain (IG) measures, described by Equation (3), where, for a given predictive feature $A_i \in A$ and a class $c_j \in C$, the value of $SU(A_i, c_j)$ quantifies the correlation between $A_i$ and a class $c_j$.

$$SU(A_i, c_j) \;=\; 2 \times \left( \frac{IG(c_j, A_i)}{E(c_j) \;+\; E(A_i)} \right) \tag{3}$$

The SU measure is not designed to deal with the hierarchical classification context. Thus, an adaptation was necessary to make the measure considers the relationship existent among the classes. The $SU_H$ measure (DIAS; MERSCHMANN, 2015) is the combination of the Hierarchical Entropy ($E_H$) and the Hierarchical Information Gain ($IG_H$) (CHEN; HU; TANG, 2009). The $E_H$ is described by Equation (4), and it is a weighted average of the entropy for each level of the class hierarchy.

More specifically, consider the hierarchical class structure $HC = \{N_1, N_2, \ldots, N_h\}$, where $N_i$ denotes the $i$th hierarchical level and $h$ the total number of levels in the hierarchy. Also, consider that the level $N_1$ is the one that contains the child nodes of the root node

of the hierarchical structure. In this way, the entropy is estimated for each hierarchy level of classes, starting from the first level ($i$ =1). Thus, the more specific is a class in the hierarchy, the deeper (greater) its level is. Finally, consider also $N_i = \{N_{(i,j)} \mid j = 1, ..., m_i \}$, where $N(i,j)$ corresponds to $j$th node (class) of level $i$ and $m_i$ is the total number of nodes (classes) of level $i$. Therefore, in Equation (4), $P(i,j)$ is the occurrence probability of the $j$th class of level $i$.

$$E_H(c_j) = -\sum_{i=1}^{h} \sum_{j=1}^{m_i} (P_{(i,j)} \times log_2 P_{(i,j)}) \times w_i \tag{4}$$

The weighting adopted by the $E_H$ is described in Equation (5), where $w_i$ is the weight assigned to level $i$ of the hierarchy. It is worth mentioning that $\sum_{i=1}^{h} w_i = 1$. Furthermore, it can be seen from Equation (5) that $w_1, w_2, \ldots, w_h$ corresponds to an arithmetical series where the highest weights are associated to the lowest (more generic) levels of the class hierarchy.

$$w_i = (h - i + 1) \times \left( \frac{2}{h \times (h+1)} \right), \; where \; i \geq 1 \tag{5}$$

The hierarchical Information Gain ($IG_H$) is described by Equation (6), where $E_H(c_j)$ is the hierarchical entropy of a class $c_j$ and $E_H(c_j \mid A_i)$ is the hierarchical entropy of a class $c_j$ after observing a feature $A_i$.

$$IG_H(c_j, A_i) = E_H(c_j) - E_H(c_j \mid A_i) \tag{6}$$

Therefore, Equation (7) describes the $SU_H$, where $IG_H(c_j, A_i)$ is the reduction caused in the hierarchical entropy of a class $c_j$ due to additional information provided by a feature $A_i$, $E_H(c_j)$ is the hierarchical entropy of a class $c_j$, and $E_H(A_i)$ is the entropy of a feature $A_i$.

$$SU_H(A_i, c_j) = 2 \times \left( \frac{IG_H(c_j, A_i)}{E_H(c_j) + E_H(A_i)} \right) \tag{7}$$

#### 2.2.1.2 Correlation-Based Feature Selection

Hall (2000) proposed the CFS, a well-known search-based filter method for single-label classification problems. It is a simple and fast-to-execute method, suitable for both

nominal class and continuous class problems (i.e., classification and regression problems, respectively).

The principle behind this measure states that a good feature subset should have two main properties: (1) the correlation between each feature and other features in the same subset should be low to minimize feature redundancy, and (2) the correlation between each feature in that subset and the class should be high.

Thus, Equation (12) describes the merit estimation of a feature subset, where $\overline{r_{FL}}$ is the average feature-label correlation over all the feature-label pairs for all features in the current feature subset, $\overline{r_{FF}}$ is the average feature-feature intercorrelation over all the pairs of features in the current feature subset $F$, and $k$ is the the number of features in $F$. Essentially, when $\overline{r_{FL}}$ increases and $\overline{r_{FF}}$ decreases, the quality of a feature subset increases regarding its ability to predict the labels of a class.

$$merit = \frac{k\overline{r_{FL}}}{\sqrt{k + k(k-1)\overline{r_{FF}}}} \tag{8}$$

#### 2.2.1.3 Genetic Algorithms

GA is a metaheuristic inspired by the process of natural selection, based on Darwin's evolutionary theory (GOLDBERG, 1989). The basic idea is that individuals in a population with better genetic features are more likely to survive and produce offspring more fittest each time while individuals that are less fit tend to disappear.

In a GA, we evaluate each individual (candidate solution) by a fitness function according to the target problem. In the context of a GA for feature selection, an individual is typically represented as a bit string where each bit takes the value 1 or 0 to indicate whether or not, respectively, a feature is included in the candidate feature subset.

Fig. 2 presents the basic structure of a GA. It starts with an initial population of individuals (candidate feature subsets), and iteratively performs the selection of individuals based on a measure fitness and creates new child individuals based on crossover and mutation of the parent individuals just selected. This process is iteratively repeated until a stopping criterion (e.g., a fixed number of iterations or generations) is satisfied.

There are two main types of genetic operators: crossover and mutation. Crossover or recombination merges information from two parents into one or two offspring. There are three main categories of crossover in the literature: one-point crossover, multi-point crossover, and uniform crossover. The mutation operator considers each gene (bit of a candidate solution) separately and allows each gene to flip (bit-flip mutation) according to the mutation rate (a user-specified parameter). Usually, a large value of mutation rate would lead the GA into a purely random search. To avoid this problem, the mutation rate is usually small, in the range of $0.5 - 5\%$. In contrast, the crossover operator is performed with a higher probability, e.g., 80%.

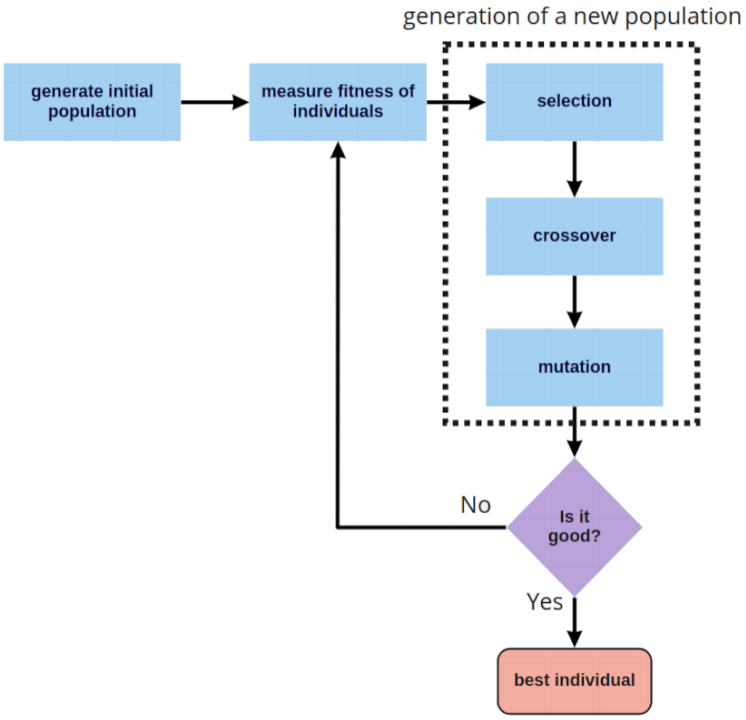



Figure 2 – Basic GA structure

### 2.2.2 Wrapper approaches

In wrapper approaches, the same classifier used in the classification step evaluates the quality of the feature subsets. Therefore, the "usefulness" of a given subset of features is measured by evaluating the trained classifier using only the features included in that subset. As search-based filter approaches, wrapper approaches need to promote searches among possible subsets of features. Each feature subset is then used to train a classification model evaluated according to some performance measure (KOHAVI; JOHN, 1997). The search process proceeds until it finds the subset with the highest evaluation in terms of the classifier's predictive performance.

Techniques that follow a wrapper approach generally produce better predictive performance results than those based on the filter approach since the classification algorithm itself drives the feature selection. However, in wrapper-based techniques, the classifier must be trained and evaluated multiple times during the search process, which could cause a very high computational cost, making it impractical for high dimensional datasets (BERMEJO; GÁMEZ; PUERTA, 2011).

### 2.2.3 Hybrid filter-wrapper approaches

In the last few years, hybrid filter-wrapper techniques have become the focus of many studies, as in this way, they aggregate the advantages of filter and wrapper approaches. Examples of hybrid filter-wrapper algorithms designed for flat classification problems

are HFS-C-P, a framework that integrates a correlation-guided clustering technique and PSO (SONG et al., 2021); BDE-X Rank, an approach that combines a wrapper method based on a Binary Differential Evolution (BDE) algorithm with a ranking-based filter method (APOLLONI; LEGUIZAMÓN; ALBA, 2016); MIMAGA, an algorithm that combines the Mutual Information Maximization (MIM) and the Adaptive Genetic Algorithm (AGA) (LU et al., 2017); and HI-BQPSO, a method that combines a filter technique with an improved quantum-behavior PSO algorithm (WU et al., 2019).

This work proposes two main supervised hybrid feature selection approaches, combining a filter and a wrapper step, wherein a global model hierarchical classifier evaluates feature subsets. The first one combines the GVNS metaheuristic as a search-based method with a feature ranking constructed by the $SU_H$ measure. The second approach is an adaptation of the CFS measure for hierarchical single-label classification and uses a best first algorithm to search the feature subset space.

# Chapter 3

# Problem description

Let $D = \{d_1, d_2, \ldots, d_t\}$ be a set of dataset instances. Let $A = \{A_1, A_2, \ldots, A_n\}$ be the set of predictive features of an instance $d_j \in D$ such that each $A_i$, $1 \leq i \leq n$, is a set of continuous or categorical values. Let $C = \{c_1, \ldots, c_r\}$ be a set of classes that relate to each other through a hierarchical structure, represented by a partial order $\prec_h$, i.e., for all $c_1, c_2 \in C, c_1 \prec_h c_2$ if and only if $c_1$ is a superclass of $c_2$. Each instance $d_j \in D$ is represented by the pair $(Y^j, c^j)$, in which $Y^j = \{y_1^j, y_2^j, \ldots, y_n^j\}$ is a list of feature values with $y_i^j \in A_i$, and $c^j \in C$ is the class of the instance $d_j$.

The Feature Selection for Hierarchical Classification (FSHC) problem identifies relevant features for the hierarchical classification task. It attempts to remove features from the dataset that do not increase or reduce the classification model's performance. Accordingly, a solution to the FSHC problem is a subset $X \subseteq A$ that can adequately classify new instances.

To exemplify, let $D$ be a set of academic papers, $A = \{$*Word count*, *Character count*, *Verb count*, *Noun count*$\}$ the feature set and $C = \{$*Computer science*, *Software engineering*, *Artificial intelligence*$\}$ the categorization of academic papers into defined topics in which *Computer science* is the superclass of *Software engineering* and *Artificial intelligence*. Let $d_j \in D$ be a paper with feature values recorded as $Y^j = \{500, 2000, 100, 200\}$ and categorization *Software engineering*. Thus, the pair $(Y^j, c^j) = (\{500, 2000, 100, 200\},$ *Software engineering*) represents this paper. The subset of features $X = \{$*Word count*, *Verb count*, *Noun count*$\}$ is an example of solution to this problem.

The time needed to train and execute a classifier, its complexity and the probability of over-fitting, and dataset dimensionality increase with the number of features. Thus, removing irrelevant and redundant features from datasets can improve the predictive classifier accuracy, simplify the generated classification model, and reduce the time spent to train a classifier. For this reason, feature selection is one of the most popular data preprocessing tasks in data mining literature.

## 3.1 Related work

Few studies in the literature discuss feature selection techniques for the hierarchical classification scenario as previously defined.

In the work of Koller and Sahami (1997), document classification (whose classes represent a hierarchy of topics) was addressed through the local model classification approach in association with feature selection using probabilistic methods for feature selection and classification. They construct a binary classifier for each node of the class hierarchy. A feature selection method is then applied to identify the most relevant features for constructing each local classifier. The feature selection method uses a measure of information theory previously proposed by Koller and Sahami (1995). As a result of this application, besides improving the predictive accuracy, reducing the number of features allowed more robust and simpler classifiers.

Secker et al. (2010) solved the problem of predicting protein functions by performing feature selection in conjunction with a local hierarchical classification approach. They used a top-down hierarchical classification strategy to select both classifiers and features for each dataset and each node of the hierarchy. Thus, in each node where a classifier has been constructed, a feature selection step is performed to reduce that particular node's dataset dimensionality. The proposed feature selection method uses the measure Correlation-based Feature Selection (CFS) and the Best First algorithm – both available in the WEKA data mining toolkit (GARNER, 1995; HALL et al., 2009). They conducted experiments to answer whether feature selection could improve computational efficiency without jeopardizing accuracy in predicting protein functions. Their experiments showed that this top-down system proposal significantly reduced the time required to train and test the classification model while maintaining predictive accuracy.

Paes, Plastino and Freitas (2014) explored the use of feature selection techniques to improve the predictive performance of two different hierarchical classification approaches, local per parent node and local per level. They proposed a method that produces a ranking of the features using the Information Gain (IG) measure (COVER; THOMAS, 1991). After forming the ranking, the $p$ best features are selected, with $p$ being an input parameter of the method. They used datasets from the bioinformatics area to conduct their experiments and concluded that the classifiers' best results occurred when some feature selection strategy was adopted.

In all of the works mentioned above, the application of feature selection techniques and classifiers' construction were performed by decomposing the hierarchical classification problem into several flat ones, which allowed them to use feature selection techniques and classification algorithms traditionally adopted in flat classification. Some recent approaches that use local model classifiers have proposed techniques based on recursive regularization that take the hierarchical structure of classes into account to select different feature subsets for each sub-classifier (ZHAO et al., 2017; TUO; ZHAO; HU, 2019;

HUANG; LIU, 2020).

Zhao et al. (2017) first propose a hierarchical feature selection technique based on recursive regularization using parent-child and sibling relations in a tree for hierarchical regularization. Experimental results showed that their algorithm efficiently selects different feature subsets for each node in a hierarchical tree structure. They achieved competitive results for both classification accuracy and computational efficiency compared with flat feature selection approaches.

Similarly, Tuo, Zhao and Hu (2019) proposed a hierarchical feature selection method with graph regularization. They sequentially used each internal node as the root node and the corresponding child nodes as leaf nodes, forming different subtrees. Then, they constructed parent-child relations as regularization of any two subtrees in the hierarchical tree structure. Their algorithm can also use the DAG label structure. They compared their method with different feature selection methods on six image datasets. The experimental results validate the efficiency and effectiveness of the proposed algorithm.

Huang and Liu (2020) proposed the most recent study that uses recursive regularization. It is the first attempt to explore a way to take advantage of the semantic description and the hierarchical structure of class labels in supervised feature selection. First, they represent the label descriptions as the semantic regularization via a vector of real numbers using sentence embedding techniques. Then, they propose a similarity score based on the attention mechanism to calculate the relevance between pairwise label vectors. Consequently, they could explore the semantic similarities of labels and use them to guide the feature selection. They also used parent-child and sibling relations as the structural regularization. Finally, they built a supervised learning model and imposed the semantic and structural regularization terms on each sub-classifier. Their proposed framework outperformed the state-of-the-art feature selection methods in the hierarchical classification domain.

Unlike those studies, our approach does not train one classifier per tree node but works in association with a global hierarchical classifier, dealing directly with the hierarchical structure of classes as a whole.

Other ranked-based methods propose to adapt some existing popular filter feature selection algorithms to handle the hierarchical structure of classes (SLAVKOV et al., 2014; DIAS; MERSCHMANN, 2015). The work of Slavkov et al. (2014) proposes a feature selection technique capable of dealing with the hierarchy of classes as a whole, without the decomposition of the hierarchical problem in several flat classification problems, considering hierarchical multi-label classification problems. They developed an adaptation of the ReliefF (ROBNIK-ŠIKONJA; KONONENKO, 2003) algorithm to the hierarchical multi-label context, called HMC-ReliefF. They employed *forward feature addition* (FFA) curves to evaluate their method, a stepwise filter-like procedure to construct classifiers for different numbers of top-k ranked features. By comparing the HMC-ReliefF curve

to an expected FFA curve obtained from a set of random rankings of features, their experiments showed that, for various datasets, the HMC-ReliefF algorithm performed well. Differently, we focus on solutions that deal with the hierarchical single-label classification scenario in this work.

Concerning hierarchical single-label classification, Dias and Merschmann (2015) proposed an adaptation of $SU$ filter measure to consider the hierarchical structure of classes. A comparative analysis between the ranking generated from the proposed measure, called $SU_H$, and another ranking randomly generated were performed. In this last one, the most relevant features are dispersed throughout the ranking positions. In this comparative evaluation, as expected, the $SU_H$ ranking resulted in higher predictive performances of the GMNB classifier than random rankings. In this work, we use the $SU_H$ filter measure (described in Section 2.2.1.1) to construct rankings combined with a wrapper step.

It is worth mentioning that Cerri et al. (2018) proposed to use the CLUS-HMC decision tree induction classifier as a feature selector, checking if the features selected to construct its tree are good enough to be used as input for two hierarchical multi-label classifiers based on neural networks and genetic algorithms. Their experimental results show that using CLUS-HMC as a feature selector led to better results than when using conventional flat multi-label methods, showing the need for developing feature selection methods specifically to consider hierarchical class relationships.

Chapter 4

# A novel hybrid feature selection based on VNS algorithms for hierarchical single-label classification

Next, we discuss the proposed algorithms based on VNS metaheuristic to solve the FSHC problem. The representation of a solution and its evaluation is presented in Section 4.1. Section 4.2 and Section 4.3 describe how to build an initial solution and to apply the neighborhood structures to explore the solution space of the problem, respectively. Section 4.4 and Section 4.5 provide a detailed description of the proposed algorithms, *Variable Neighborhood Search for Feature Selection in Hierarchical Classification* (VNS-FSHC) and *General Variable Neighborhood Search for Feature Selection in Hierarchical Classification* (GVNS-FSHC), respectively. Finally, Section 4.6 describes the experimental setup and reports the computational results, and Section 4.7 concludes the chapter.

## 4.1 Solution representation and evaluation

In this work, we propose hybrid feature selection methods based on the VNS metaheuristic, which first generate an initial solution $X \subseteq A$ and then explore the problem's solution space from this starting point.

To evaluate each solution $X' = \{x'_1, x'_2, \ldots, x'_m\}$, $m \leq n$ that is generated, we used the 5-fold cross validation strategy in association with the hierarchical $F$-measure ($hF$) (KIRITCHENKO; MATWIN; FAMILI, 2005) to evaluate the performance of each global hierarchical classifier adopted.

The $hF$ measure is an adaptation of the traditional $F$-measure, intensely used in flat classification problems, to take into account the class hierarchy. According to Silla and Freitas (2011), the main reason for using the $hF$ measure is that it can be effectively applied to any hierarchical classification scenario, i.e., tree, DAG, single-label, multi-label, MLNP, or NMLNP.

The quality of the solution $X$ is calculated according to the following equation:

$$hF(X) = \frac{2 \times hP(X) \times hR(X)}{hP(X) + hR(X)} \tag{9}$$

where $hP(X)$ and $hR(X)$ stand for the hierarchical Precision and the hierarchical Recall, respectively.

Considering $P_j$ as the set consisting of the most specific class predicted for the test instance $j$ and all its ancestor classes, and $T_j$ as the set consisting of the true most specific class of this same test instance and all its ancestor classes, the $hP(X)$ and $hR(X)$ of the solution $X$ can be defined according to (10) and (11):

$$hP(X) = \frac{\sum_j |P_j \cap T_j|}{\sum_j |P_j|} \tag{10}$$

$$hR(X) = \frac{\sum_j |P_j \cap T_j|}{\sum_j |T_j|}. \tag{11}$$

## 4.2 Building an initial solution

The initial solution is generated using the *Incremental Wrapper Subset Selection* (IWSS) approach (RUIZ; RIQUELME; AGUILAR-RUIZ, 2006), which works in two steps as follows:

(i) ***Filter***: a filter-based measure evaluates each predictive feature independently in regard to the dataset classes to create a ranking $R$ considering the $SU_H$ measure (DIAS; MERSCHMANN, 2015). Then, the ranking $R$ of all features is constructed using the roulette wheel method as in the survival selection phase in Genetic Algorithms (GOLDBERG, 1989). Thus, a feature's selection probability is proportional to its $SU_H$ value compared to this metric value for all other predictive features. That is, the best-evaluated features by the $SU_H$ metric are more likely to be selected in the first rounds of the roulette wheel method, occupying the initial ranking positions.

(ii) ***Wrapper***: the initial solution $X$ starts with the best-rated feature in the ranking $R$. Then we try to insert in $X$ the next feature $A_i \in R$ iteratively by evaluating the performance of that expanded subset $X' = X \cup \{A_i\}$. We evaluate the quality of each candidate subset $X'$ in a wrapper way (using a global model classifier). If $X'$ increases the classifier's predictive performance, $A_i$ is added to $X$ or discarded otherwise.

We used the same five folds of the cross-validation method in all wrapper evaluations to have fair comparisons. Also, we complement the IWSS method by adding a step

that verifies feature redundancy. When analyzing the inclusion of a feature $A_i$ in the initial solution, if its insertion in $X$ does not improve the classifier's performance, we try to swap it with each feature already inserted in $X$. Then if one of these temporary subsets increases the classifier's performance concerning $X$, the best-evaluated subset is maintained for the next iteration.

For instance, let be $X = \{A_1, A_2\}$ and $hF(X) = 0.70$. We try to insert $A_3$ in $X$, but it did not improve the classifier's performance. Therefore, we generate the temporary subsets $Y = \{A_3, A_2\}$ and $Z = \{A_1, A_3\}$, with $hF(Y) = 0.75$ and $hF(Z) = 0.60$. As $hF(Y)$ is greater than $hf(X)$, $Y$ is maintained for the next iteration, which would be try to include $A_4$ in $Y$. This procedure aims to revoke some previous decisions by identifying selected features that may become ineffective with the insertion of another one. Thus, this step follows the well-known Proximate Optimality Principle (POP) (GLOVER; LAGUNA, 1997).

## 4.3 Neighborhood structures

We considered three types of neighborhoods on a solution $X$ to search the problem solution space:

(i) *Neighborhood structure* $N_1$: It consists in removing a feature $X_j \in X$ from $X$, that is, $X = X \setminus \{X_j\}$.

(ii) *Neighborhood structure* $N_2$: It consists in inserting a feature $A_i \in (A \setminus X)$ into $X$, that is, $X = X \cup \{A_i\}$.

(iii) *Neighborhood structure* $N_3$: It consists in swapping a feature $X_j \in X$ with a feature $A_i \in (A \setminus X)$.

For the example described in Chapter 3, given the solution $X = \{$*Word count*, *Verb count*, *Noun count*$\}$, a swap movement consists of swap a feature in $X$ with another that is not already inserted in $X$. Thus, $X' = \{$*Word count*, *Character count*, *Noun count*$\}$ is a neighbor of $X$ considering the swap movement. Likewise, $X' = \{$*Word count*, *Verb count*, *Character count*, *Noun count*$\}$ is a neighbor example considering the insertion movement, and $X' = \{$*Verb count*, *Noun count*$\}$ is a neighbor of $X$ produced by the removal movement.

## 4.4 VNS approach for FSHC

The first proposed algorithm, so-called VNS-FSHC, is based on the VNS metaheuristic (MLADENOVIĆ; HANSEN, 1997). Its pseudo-code is outlined in Algorithm 1.

In Algorithm 1, VNSmax indicates the maximum number of iterations without improvement in the best solution found. $D$, $C$ and $M$ are the training set, the GMNB classifier and the $SU_H$ filter measure, respectively. Furthermore, $N_1$, $N_2$ and $N_3$ are the neighborhoods defined in Section 4.3.

An initial solution $X$ (line 2) is generated by applying the IWSS approach described in Section 4.2. Next, in line 4, $w$ is generated randomly ($2 \leq w \leq 4$). This parameter is the number of folds in which the evaluation ($hF$) of the tested solution must be higher than the evaluation of the current solution. This parameter is used in the Relevance function (line 12), a procedure that compares the evaluations of solutions $X'$ and $X''$.

The RandomDescent method (line 11) is adopted as an improvement step using the neighborhood $N_1$. It analyzes a neighbor that belongs to $N_1$ and accepts it only if it is strictly better than the current solution. If it is not true, the current solution remains unchanged and another neighbor is generated and analyzed. Additionally, the $SU_H$ filter measure is used to generate the ranking of features and the selection of features to swap is performed using the roulette wheel method as in the survival selection phase in Genetic Algorithms. Then, features that do not belong to the solution and have higher values of ranking have more chance of being chosen to swap with features that belong to the solution and have lower values of ranking. The procedure is interrupted after RDmax iterations without improvement in the best solution found.

**Algorithm 1** VNS-FSHC algorithm

```
1:  in: D, C, M, N_1(.), N_2(.), N_3(.)    out: X
2:  X ← InitialSolution(D, C, M);
3:  Calculate VNSmax based on X
4:  w ← random(2, 4);
5:  Iter ← 0;
6:  while Iter < VNSmax do
7:     k ← 2; Iter ← Iter + 1;
8:     while k ≤ 3 do
9:        Randomly generates a neighbor X' ∈ N_k(X);
10:       Calculate RDmax based on X'
11:       X'' ← RandomDescent(X', RDmax, D, C, M, w, N_1(.));
12:       if Relevance(X'', X, w) then
13:          X ← X''; k ← 2; Iter ← 0;
14:       else
15:          k ← k + 1;
16:       end if
17:    end while
18: end while
19: return X;
```

### 4.4.1 Relevance function

Algorithm 2 outlines the pseudo-code of the Relevance function. It starts averaging the $hF$ performance achieved on 5-fold cross-validation for each solution (lines 2 and 3). Then, solution $X''$ is considered better than $X$ if the average $hF(X'')$ is bigger than $hF(X)$ (line 5) and also if $w$ of the five $hF(X'')$ are bigger than $hF(X)$ (line 11). Thus, if both conditions are true, the function indicates that the solution $X''$ is better than the solution $X$.

**Algorithm 2** Relevance function

```
1: in: X'', X, w    out: boolean
2: X''.hF = ( Σ_{i=1}^{5} S'.hF⃗[i] )/5;
3: X.hF = ( Σ_{i=1}^{5} S''.hF⃗[i])/5;
4: counter = 0;
5: if X''.hF > X.hF then
6:    for i = 1 to 5 do
7:       if X''.hF⃗[i] > X.hF⃗[i] then
8:          counter++;
9:       end if
10:   end for
11:   if counter ≥ w then
12:      return true;
13:   end if
14: end if
15: return false;
```

## 4.5 GVNS approach for FSHC

This section presents the GVNS-FSHC algorithm, an adaptation of the General Variable Neighborhood Search (GVNS) metaheuristic (HANSEN et al., 2019) for the FSHC problem.

GVNS is a variation of the VNS metaheuristic, a framework for building heuristics based on neighborhoods' systematic changes. It is applied to find a local minimum in a descent step and escape from the corresponding valley in a perturbation step (HANSEN et al., 2019). GVNS differs from VNS concerning the local search method. While in VNS, the local search is conventional, in GVNS, the local search is done by the Variable Neighborhood Descent (VND) (MLADENOVIĆ; HANSEN, 1997) method.

In our GVNS-FSHC algorithm, we apply the basic sequential VND, named B-VND by Hansen et al. (2019). Algorithm 3 presents the pseudo-code of the proposed GVNS-FSHC.

In Algorithm 3, $D$, $C$, and $M$ are the training set, the hierarchical classifier, and the $SU_H$ filter measure, respectively. Furthermore, $N_1$, $N_2$, and $N_3$ are the neighborhoods

defined in Section 4.3. The $attempt_{\max}$, $RDrate$, and $w$ inputs are predefined parameters and will be explained below.

The $attempt_{\max}$ parameter defines the maximum number of attempts without improvement using the same level $k$ of perturbations of the Shake function. In a classical GVNS algorithm, the number of perturbations is increased whenever there is no improvement in the solution. Instead, in our algorithm, we only increase the number of perturbations after performing some local search attempts without improving the current solution. This strategy follows the ideas introduced by Reinsma, Penna and Souza (2018) and used successfully in Santos et al. (2020).

The $RDrate$ is a percentage rate used in the B-VND procedure of improvement, described in Section 4.5.1. Finally, $w$ is the number of folds in which the tested solution's evaluation ($hF$) must be greater than the current solution's evaluation. The Relevance function is described in Section 4.4.1.

The algorithm generates an initial solution $X$ (line 3) by applying the IWSS approach described in Section 4.2. In line 4, the variable $k$, which defines the number of random moves that will be applied in a given solution $X$ to generate a perturbed solution in the current neighborhood, is initialized. In line 5, the variable *attempt*, used to control the number of iterations using the same level $k$ of perturbations without improvement in the current solution $X$, is started.

A neighborhood structure is chosen randomly (line 8), and then the perturbed solution $X'$ is generated by the shaking procedure (line 9) that considers the neighborhood structure $N_l(.)$ to perform $k$ moves on the solution $X$. The solution $X'$ is subjected to the B-VND local search procedure, generating the solution $X''$. Next, the Relevance function verifies if $X''$ is better than the current solution $X$. If an improvement is detected, $X''$ is considered the best solution found so far, and $k$ is set to one. In lines 16 to 19, when no improvement is detected, if $attempt_{\max}$ iterations have already been run, the variable $k$ is increased by 1 and *attempt* is restarted. GVNS-FSHC ends when the given total running time $t_{\max}$ expires.

### 4.5.1 Variable neighborhood descent

The basic sequential VND procedure, named B-VND in Hansen et al. (2019), is the local search used in the GVNS-FSHC algorithm (line 10 of Algorithm 3). Our B-VND approach uses the following sequence of neighborhoods, in this order: $N_1$, $N_2$, and $N_3$. We ordered these neighborhoods by their size, which is a common strategy in VNS-based algorithms, according to ).

In Algorithm 4, $X$ is the current solution subjected to the B-VND local search procedure, and $RDrate$ is a percentage used to calculate the maximum number of iterations without improvement of the random descent improvement step. Furthermore, inputs $D$, $C$ , $M$, $w$, $N_1$, $N_2$, and $N_3$ are the same as defined in Algorithm 3.

**Algorithm 3** GVNS-FSHC algorithm

```
1:  in: D, C, M, N_1(.), N_2(.), N_3(.),
        attempt_max, RDrate, w
2:  out: X
3:  X ← InitialSolution(D, C, M);
4:  k ← 1;
5:  attempt ← 0;
6:  repeat
7:     attempt ← attempt + 1;
8:     Randomly choose a neighborhood structure N_l(.)
9:     X' ← Shake(X, N_l(.), k);
10:    X'' ← B-VND(X', RDrate, D, C, M, w, N_1(.), N_2(.), N_3(.));
11:    if Relevance(X'', X, w) then
12:       X ← X'';
13:       k ← 1;
14:       attempt ← 0;
15:    else
16:       if attempt > attempt_max then
17:          k ← k + 1;
18:          attempt ← 0;
19:       end if
20:    end if
21: until t ≤ t_max
22: return X;
```

In line 3, $l$ represents the current neighborhood structure used by the B-VND procedure. Initially, the maximum number of iterations without improvement ($RDmax$ in line 6), used by the random descent improvement step (lines 8 to 15), is defined. Considering $X'$ the current solution and $A$ the set of predictive features (Chapter 3), we will denote $|X'|$ as the number of elements of $X'$, and $RDmax = RDrate \times |X'| \times |A \setminus X'|$.

Our B-VND procedure has a random descent step (lines 8 to 15) in the same neighborhood and a step to change neighborhoods (lines 16 to 21). At the beginning of the B-VND procedure, the algorithm makes a copy $X'$ of the current solution $X$ (line 5). The random descent strategy starts by analyzing a neighbor $X''$ that belongs to the current neighborhood $N_l(X')$ (line 9) and accepts it as the new current solution if it is strictly better than $X'$ (line 10). Otherwise, $X'$ remains unchanged, and the algorithm generates and analyzes another neighbor. The algorithm repeats this random procedure until $RDmax$ iterations without improvement in the same neighborhood (line 8). After that, if the improved solution $X'$ is better than $X$, then $X'$ becomes the new current solution, and the random descend search returns to the first neighborhood (lines 17 and 18); otherwise, the search continues in the next neighborhood (line 20). The B-VND ends when there is no improvement in neither of the three neighborhoods.

It is worth mentioning that the $SU_H$ filter measure generates a feature ranking, used to direct the selection of features to swap, insert, or exclude from a candidate solution.

**Algorithm 4** B-VND algorithm

```
1: in: X, RDrate, D, C, M, w, N_1(.), N_2(.), N_3(.)
2: out: X
3: l ← 1;
4: while l ≤ 3 do
5:    X' ← X;
6:    RDmax ← RDrate percent of a predefined number of iterations
7:    iterRD ← 1;
8:    while iterRD ≤ RDmax do
9:       Randomly choose X'' ∈ N_l(X')
10:      if Relevance(X'', X', w) then
11:         iterRD ← 1;
12:         X' ← X'';
13:      end if
14:      iterRD = iterRD + 1;
15:   end while
16:   if Relevance(X', X, w) then
17:      X ← X';
18:      l ← 1;
19:   else
20:      l = l + 1;
21:   end if
22: end while
23: return X;
```

To do this, we perform the roulette wheel method, as used in the survival selection phase in Genetic Algorithms. Therefore, the probability of inserting a feature in a candidate solution is higher if it has higher ranking values. Similarly, features in a candidate solution set with low ranking values have a higher probability of being removed from the solution set.

## 4.6 Experimental results

The VNS-FSHC and GVNS-FSHC algorithms presented in Sections 4.4 and 4.5, respectively, were implemented in C++, using the compiler g++, version 4.8.5 for their execution. The experiments were performed on a computer with Intel Xeon(R) CPU E5620 @ 2.40GHz × 16, with 48 GB of RAM and CentOS Linux 7 operational system. Although the processor of this computer has more than one core, the algorithm is not optimized for multi-processing.

The proposed algorithms were designed to run as a preprocessing step for global hierarchical classifiers. In this sense, the computational experiments evaluate the proposed algorithm's efficacy for feature selection in the hierarchical single-label classification context. We used the GMNB and the CLUS-HMC hierarchical classifiers to evaluate the

quality of the selected features. It is worth mentioning that the CLUS-HMC deals with hierarchical multi-label problems, but it can also be used in the hierarchical single-label context. In the latter case, one needs only to consider single-label datasets as a particular case of multi-label classification in which the number of paths of labels in the class hierarchy is equal to one.

Based on the evaluation metrics, hierarchical Precision and hierarchical Recall, described in Section 4.1, we compared the proposed algorithms against the following strategies:

(i) **ALL**: we measured the performance of the classifier without any feature selection preprocessing step, i.e., using all features from the dataset.

(ii) **BF**: we implemented a bottom-up wrapper-based approach of the best first algorithm, a well-known heuristic search method (LIU; MOTODA, 2007). We first ranked all the features using the classifier performance evaluation in a descending manner. Then, starting with a subset containing only the first feature of the rank, the algorithm returns the best feature subset found by the heuristic search, measuring the quality of each candidate subset based on the classifier performance. Instead of evaluating all the subsets of features generated in the OPEN list, we chose a predefined number of backtracking to a candidate solution in the OPEN list without improvements as the stopping criterion of the algorithm.

### 4.6.1 Dataset description

The experiments use twelve public benchmark datasets with classes hierarchically organized in a tree structure, covering two domains, protein and image. The protein domain is represented by bioinformatic datasets[1] referring to the yeast genome (CLARE; KING, 2003).

The image datasets[2] were selected from the ImageCLEF 2007 competition for annotation of medical X-ray images. ImageCLEF aims to provide an evaluation forum for the cross-language annotation for images of the medical radiological domain (DIMITROVSKI et al., 2011).

These datasets are initially available as multi-label data. Since our method focus on dealing with the single-label scenario, we perform a preprocessing step to convert them into single-label data. Table 1 shows general characteristics of the datasets. For each dataset, the second column corresponds to the dataset domain, whereas the third column represents the total number of features. The fourth column represents the number of instances, and the fifth column represents the number of classes in each level of the tree hierarchy.

[1] http://dtai.cs.kuleuven.be/clus/hmcdatasets/
[2] http://kt.ijs.si/DragiKocev/PhD/resources/doku.php?id=hmc_classification/

Table 1 – General characteristics of the datasets

| Dataset | Domain | # Features | # Instances | # Classes/Level |
|---|---|---|---|---|
| CellCycle | protein | 77 | 3723 | 15/14/14/8 |
| Church | protein | 27 | 3720 | 15/14/14/7 |
| Gasch2 | protein | 52 | 3742 | 15/14/14/8 |
| SPO | protein | 80 | 3653 | 15/13/14/7 |
| Phenotype | protein | 67 | 1551 | 12/13/12/6 |
| Eisen | protein | 79 | 2359 | 12/14/13/7 |
| Derisi | protein | 63 | 3677 | 15/13/14/7 |
| Gasch1 | protein | 173 | 3727 | 15/14/14/8 |
| Sequence | protein | 478 | 3874 | 15/14/14/8 |
| Expression | protein | 551 | 3742 | 15/14/14/8 |
| ImageCLEF07A | image | 80 | 11006 | 4/8/8 |
| ImageCLEF07D | image | 80 | 11006 | 8/7/11 |

The data preprocessing was carried out through four steps. In the first step, we selected, for each instance, the most frequent class considering the leaf nodes in the original dataset. In the second step, each missing value was replaced using the method *Hierarchical Supervised Imputation Method* (HSIM) (GALVÃO; MERSCHMANN, 2016). In the third step, every class with fewer than ten instances was merged with its parent class until all classes possessed at least ten instances. Finally, in the fourth step, we applied the unsupervised discretization method Equal Frequency Binning (YANG; WEBB, 2001) with 20 partitions to convert all continuous features into discrete values.

### 4.6.2 Parameter settings

Fig. 3 presents the 5-fold cross-validation setup used for all our experiments and comparisons The best feature subset for both algorithms was selected also using the 5-fold cross-validation procedure within the training set.

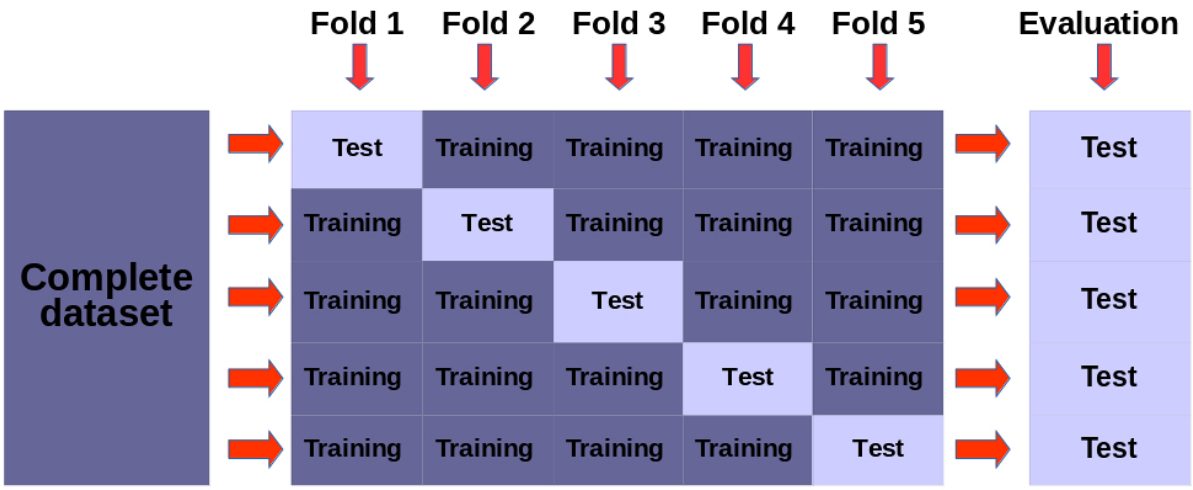



Figure 3 – 5-fold cross-validation setup

The parameters of the VNS-FSHC are those used by Costa et al. (2018), which were fixed at the following values: VNSmax = 0.1 × (number of features included in the initial solution) × (number of features excluded from the same solution), and RDmax = 0.1 × (number of features included in the current solution passed to the RandomDescent method) × (number of features excluded from the same solution).

Regarding the BF algorithm, we did preliminary experiments varying the stopping criterion from $\{5, 10, 15\}$ in all the datasets. Since we did not significantly improve the classifier's performance using the value 15 compared to 10, we fixed the stopping criterion as 10 in all datasets.

The parameter tuning of the GVNS-FSHC used the Irace package (LÓPEZ-IBÁÑEZ et al., 2016), an automatic algorithm configuration method. Table 2 shows the tuning setup, and we considered the 5-fold cross-validation procedure within the training set of the SPO dataset. The Irace generated three configurations, presented in Table 3. Configuration 1 ($w = 2$, $attempt_{\max} = 4$, and $RDrate = 0.02$) was chosen because it requires the lowest computational cost.

Table 2 – GVNS-FSHC tuning setup.

| **Parameter** | **Range** |
|---|---|
| $w$ | $\{2, 3, 4\}$ |
| $attempt_{\max}$ | $\{2, 4, 6, 8, 10\}$ |
| $RDrate$ | $\{0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20\}$ |

Table 3 – Irace best configurations.

| **Configuration** | $w$ | $attempt_{\max}$ | $RDrate$ |
|---|---|---|---|
| 1 | 2 | 4 | 0.02 |
| 2 | 2 | 10 | 0.10 |
| 3 | 4 | 10 | 0.20 |

## 4.6.3 Computational results

Section 4.6.3.1 presents the computational results using the GMNB classifier and Section 4.6.3.2 shows the CLUS-HMC results.

### 4.6.3.1 Results with the GMNB classifier

Considering the stochastic nature of the VNS-based algorithms, each one was applied 30 times to each dataset. To compare the GMNB performance using both the VNS-FSHC and the GVNS-FSHC algorithms, we first recorded the running time spent by each execution of the VNS-FSHC. Then, we executed the GVNS-FSHC with the same running time for a fairer comparison, considering the same dataset partition and seed for

Table 4 – $hF$ results (using the GMNB classifier)

| | **ALL** | **BF** | **VNS-FSHC** | **GVNS-FSHC** |
|---|---|---|---|---|
| ***Dataset*** | ***avg* (*sd*)** | ***avg* (*sd*)** | ***avg* (*sd*)** | ***avg* (*sd*)** |
| CellCycle | • 25.23 (1.1) | • 23.93 (3.2) | • 24.34 (3.5) | **25.27** (2.4) |
| Church | 14.59 (3.5) | 20.20 (4.3) | • 22.20 (4.4) | **22.28** (4.5) |
| SPO | 16.57 (1.2) | • **20.44** (0.3) | • 20.33 (0.6) | 20.42 (0.8) |
| Gasch2 | • **19.68** (1.3) | • 17.89 (1.4) | 17.48 (1.7) | 18.71 (1.3) |
| Phenotype | 10.09 (1.9) | 15.53 (1.4) | • 16.13 (1.3) | **16.70** (1.9) |
| Eisen | • **23.78** (2.1) | 19.98 (1.4) | 19.39 (1.7) | 21.84 (1.9) |
| Derisi | 14.75 (1.4) | • **16.75** (0.7) | • 16.62 (0.9) | 16.42 (1.1) |
| Gasch1 | 24.87 (1.7) | • **28.31** (0.9) | • 28.01 (1.7) | 28.04 (2.0) |
| Sequence | • **19.96** (0.7) | • 19.76 (1.2) | • 18.81 (1.6) | 19.46 (0.6) |
| Expression | 36.24 (1.5) | • 46.33 (2.2) | • 47.45 (2.6) | **48.12** (2.4) |
| ImageCLEF07A | • 80.37 (1.2) | • **80.79** (0.9) | • 80.27 (0.8) | 80.67 (0.8) |
| ImageCLEF07D | 63.04 (0.9) | • 66.58 (1.0) | • 66.65 (0.7) | **66.78** (0.7) |

generating random numbers of those metaheuristic algorithms. As the BF heuristic is deterministic, it was required only one execution for each dataset.

The results obtained in each dataset were compared by using two one-way hypothesis tests with a significance level of 0.05. To choose the most appropriate statistical test for each dataset result, we first verified if they were well-modeled by a normal distribution by applying the Shapiro-Wilk test (ROYSTON, 1982). If samples came from populations with normal distributions, we applied the ANOVA test (HEIBERGER, 2015), a parametric hypothesis test for two independent samples, or else, we applied the Kruskal-Wallis test (HOLLANDER; WOLFE, 1973), non-parametric analysis of variance which can compare several independent samples.

Table 4 shows the results obtained by using all features, the BF comparison algorithm, the VNS-FSHC, and the GVNS-FSHC algorithm concerning the $hF$ measure. From the second to the fifth column, we represent the $hF$ values reached by the GMNB classifier using all the dataset features (second column) and feature selection (other columns). In these columns, "*avg*" indicates the average result, with the standard deviation (*sd*) in parentheses. Bold results show the best absolute value, and a result preceded by • indicates no statistically significant difference between the specific result and the GVNS-FSHC result.

The experiments showed that the GVNS-FSHC algorithm obtained the best absolute average for five datasets (CellCycle, Church, Phenotype, Expression, and ImageCLEF07D). Moreover, the GVNS-FSHC is better than at least one comparison strategy with statistical significance for four of these five datasets. For the remaining ones (SPO, Gasch2, Eisen, Derisi, Gasch1, Sequence, and ImageCLEF07A), its performance was equivalent to the best result found, i.e., the difference was not statistically significant.

It is also important to mention that for the GMNB classifier, using a feature selection method improved the model's performance for most of the datasets (Church, SPO,

Table 5 – Number of features (using the GMNB classifier)

| | | **BF** | | **VNS-FSHC** | | **GVNS-FSHC** | |
|---|---|---|---|---|---|---|---|
| ***Dataset*** | ***All*** | ***avg*** | ***sd*** | ***avg*** | ***sd*** | ***avg*** | ***sd*** |
| CellCycle | 77.0 | 14.6 | 7.4 | 13.1 | 5.8 | 18.8 | 3.2 |
| Church | 27.0 | 3.2 | 0.4 | 3.1 | 0.7 | 3.0 | 0.0 |
| SPO | 79.0 | 4.0 | 1.0 | 4.0 | 0.9 | 3.7 | 0.7 |
| Gasch2 | 52.0 | 19.8 | 3.1 | 4.7 | 4.9 | 20.6 | 2.8 |
| Phenotype | 67.0 | 12.8 | 3.1 | 7.7 | 3.7 | 16.8 | 4.5 |
| Eisen | 79.0 | 1.6 | 0.5 | 1.6 | 0.5 | 27.0 | 1.8 |
| Derisi | 63.0 | 2.0 | 0.7 | 2.3 | 0.7 | 3.7 | 5.6 |
| Gasch1 | 173.0 | 17.8 | 2.2 | 15.8 | 2.7 | 23.4 | 4.9 |
| Sequence | 478.0 | 4.2 | 2.3 | 7.0 | 6.3 | 37.1 | 6.4 |
| Expression | 551.0 | 25.4 | 3.4 | 19.9 | 3.1 | 24.0 | 3.5 |
| ImageCLEF07A | 80.0 | 60.8 | 8.2 | 52.4 | 5.4 | 63.7 | 2.8 |
| ImageCLEF07D | 80.0 | 27.0 | 3.8 | 28.2 | 3.6 | 31.1 | 2.8 |

Phenotype, Derisi, Gasch1, Expression, and ImageCLEF07D).

Table 5 shows the comparison results between the algorithms concerning the number of features used by the GMNB classifier. The second column presents the number of features used without feature selection, and the remaining columns represent both the average (*avg*) and standard deviation (*sd*) of the number of features used by the algorithms.

When we compare the results of Table 5 concerning Table 4, we observed that when the GVNS-FSHC has not the best absolute performance concerning the $hF$ measure (SPO, Gasch2, Eisen, Derisi, Gasch1, Sequence, and ImageCLEF07A), it selects fewer features than the strategy with the best absolute performance value for four datasets. The only exceptions to this performance occur in Derisi, Gasch1, and ImageCLEF07A datasets, in which BF is the best strategy regarding best absolute performance and number of selected features.

Ultimately, these results show that the GVNS-FSHC algorithm with the GMNB classifier is consistently better or equivalent to the other comparison strategies regarding the $hF$ measure.

Aiming to characterize and compare the behavior of the GVNS-FSHC algorithm to its previous version (the VNS-FSHC algorithm) concerning the running times, we used the multiple time-to-target plot (mttt-plot) tool (REYES; RIBEIRO, 2018). The mttt-plot is an extension of time-to-target plot (FEO; RESENDE; SMITH, 1994) to sets of multiple instances.

Runtime distributions (ttt-plots) display on the ordinate axis the probability that an algorithm will find a solution at least as good as a given target value for a given problem instance within a given running time, shown on the abscissa axis (REYES; RIBEIRO, 2018). To build a ttt-plot, the algorithm $\mathcal{A}$ is run $q$ times on the fixed instance $\mathcal{I}$ and stops as soon as it finds a solution whose objective function is at least as good as the given target value $look4$. After concluding the $q$ independent runs, a Cumulative Distribution

Function (CDF) represents the solution times.

To build a mttt-plot, instead of one single instance and target value, $p$ instances $\mathcal{I}_j$ and their corresponding targets $look4_j$ are used, for $j = 1, 2, \ldots, p$. Let each $S_j \geq 0$ be a continuous random variable representing the time taken by algorithm $\mathcal{A}$ to find a solution as good as the target value $look4_j$ for instance $\mathcal{I}_j$ and $F_{S_j}(s) = P(S_j \leq s)$ be the cumulative distribution function of $S_j$. The mttt-plot is defined by a set of $z$ points $(\alpha_k, \hat{F}_{S_1+\ldots+S_p}(\alpha_k))$, for $k = 1, 2, \ldots, z$ and $z \gg q$, where each $\alpha_k$ is a sample of $S_1 + \ldots + S_p$, and $\hat{F}_{S_1+\ldots+S_p}$ is an estimator of $F_{S_1+\ldots+S_p}$. To generate these $z$ points, we sample $z$ occurrences of the sum of independent variables $S_1 + \ldots + S_p$ using the algorithm proposed by Reyes and Ribeiro (2018).

We considered one partition of each dataset (5) as instances. For each instance, two target values were considered ($a$ = mean of 30 runs of each dataset, and $b = a - 0.01 \times a$), making a total of $p = 10$ instance-target pairs. Each algorithm was run $q = 20$ times for each instance-target pair, until a solution at least as good as the corresponding target was found for each instance.

Fig. 4 shows the mttt-plot for each algorithm, resulting from the 10 individuals ttt-plots using $z = 2 \times 10^4$. We observed that the GVNS-FSHC performs better for this 10 instance-target pairs set. The GVNS-FSHC finds a target solution within $10^{7.4}$ milliseconds in approximately 70% of the times it ran. In contrast, the VNS-FSHC finds a solution in a longer time (within $10^{7.6}$ milliseconds), considering the same 70% of the times it ran. On the other hand, when we set a processing time, the GVNS-FSHC is more likely to reach the target value than VNS-FSHC. For example, at time $10^{7.4}$ milliseconds, the VNS-FSHC reaches the target value only in approximately 15% of the executions, while the proposed algorithm reaches the target value in about 75% of the executions.

Fig. 5 shows the evolution of the objective function ($hF$ measure) over time considering the pair (partition, seed) that generated the best result for the GVNS-FSHC in each dataset. It is possible to notice that, corroborating the mttt-plots results presented in Fig. 4, in most datasets, the GVNS-FSHC achieves the improvements before the VNS-FSHC algorithm.

#### 4.6.3.2 Results with the CLUS-HMC classifier

To see if our approach improved the performance of a classifier widely used in the literature, in this section, we compare the CLUS-HMC (VENS et al., 2008) performance with and without the feature selection generated by the GVNS-FSHC and the BF algorithm, using the same classifier. Worth emphasizing that the CLUS-HMC is a classifier based on decision trees. Specifically, it makes a feature selection embedded to optimize the objective function or performance of the learning model.

Table 6 shows the results of the GVNS-FSHC using the CLUS-HMC classifier and following the same notation of Table 4. The results show that the feature selection step

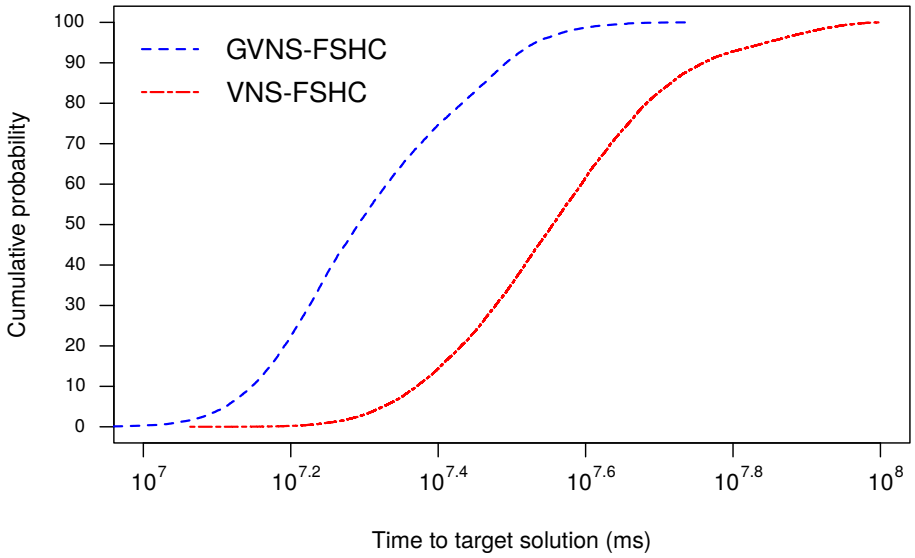


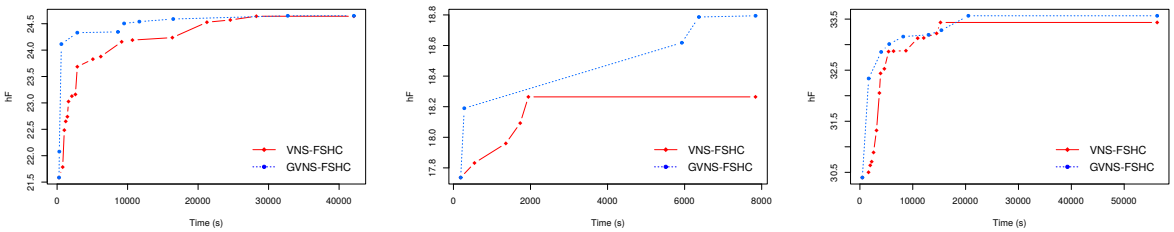

Figure 4 – Combined mttt-plot for VNS-FSHC × GVNS-FSHC

Table 6 – $hF$ results (using the CLUS-HMC classifier)

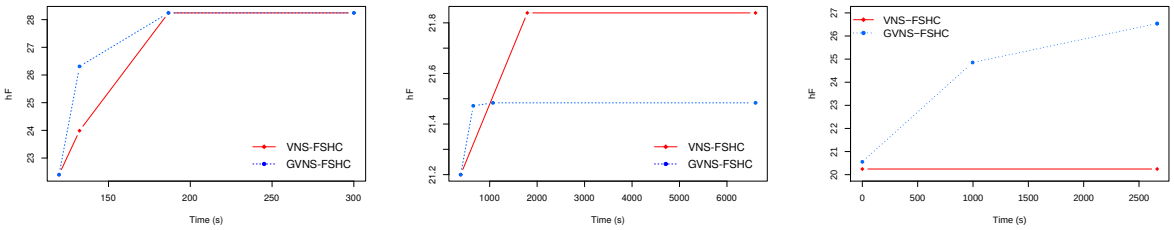
| | **ALL** | **BF** | **GVNS-FSHC** |
|---|---|---|---|
| ***Dataset*** | ***avg* (*sd*)** | ***avg* (*sd*)** | ***avg* (*sd*)** |
| CellCycle | • **22.39** (6.2) | • 22.05 (7.6) | 22.37 (5.8) |
| Church | 19.40 (7.6) | • 22.57 (4.3) | **22.74** (3.8) |
| SPO | • **21.79** (1.5) | • 21.65 (1.9) | 21.64 (1.3) |
| Gasch2 | • **17.47** (1.8) | • 16.67 (1.9) | 16.65 (2.1) |
| Phenotype | • **15.44** (1.1) | • 14.78 (1.2) | 15.10 (0.7) |
| Eisen | • **22.77** (1.6) | • 21.82 (2.1) | 22.43 (1.7) |
| Derisi | • 18.14 (1.2) | • **18.49** (0.8) | 16.82 (1.1) |
| Gasch1 | • 21.57 (1.5) | • **22.04** (1.0) | 21.69 (1.7) |
| Sequence | • 22.68 (0.9) | • 21.37 (1.8) | **22.95** (1.5) |
| Expression | • 42.11 (1.0) | • 42.22 (1.8) | **42.33** (2.0) |
| ImageCLEF07A | • **64.92** (1.2) | • 64.04 (1.0) | 64.78 (0.8) |
| ImageCLEF07D | • **66.11** (0.8) | • 65.01 (0.5) | 65.46 (1.0) |

using the GVNS-FSHC algorithm did not improve the CLUS-HMC classifier's performance with statistical significance (except for the Church dataset), confirming the power of decision trees as natural feature selectors. However, the GVNS-FSHC algorithm did not jeopardize the CLUS-HMC classifier's performance with statistical significance either.

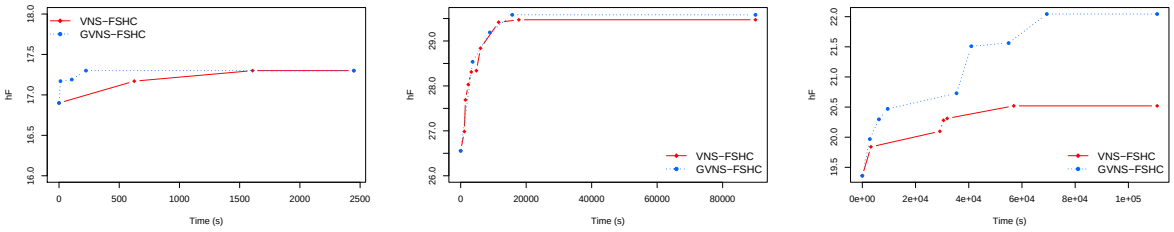
Considering the number of features used by the CLUS-HMC classifier with and without the feature selection step, shown in Table 7, one can see a significant reduction in the number of features. Thus, this feature selection can still be compelling since it can improve the model interpretability without losing accuracy.



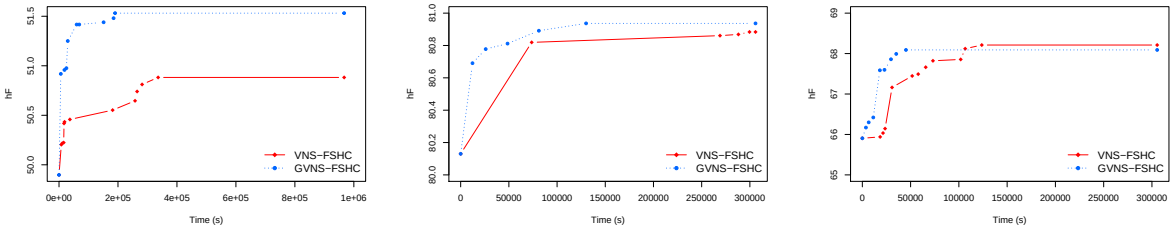

(a) Gasch2 (b) Pheno (c) CellCycle




(d) Church (e) SPO (f) Eisen




(g) Derisi (h) Gasch1 (i) Sequence




(j) Expression (k) ImageCLEF07A (l) ImageCLEF07D

Figure 5 – Evolution of the objective function (hF) over time considering the pair (partition, seed) that generated the best result for the GVNS-FSHC in each dataset

Table 7 – Number of features (using the CLUS-HMC classifier)

| | | BF | | GVNS-FSHC | |
|---|---|---|---|---|---|
| ***Dataset*** | ***All*** | ***avg*** | ***sd*** | ***avg*** | ***sd*** |
| CellCycle | 77.0 | 14.4 | 4.6 | 22.9 | 5.9 |
| Church | 23.0 | 5.6 | 1.5 | 6.1 | 1.3 |
| SPO | 79.0 | 3.2 | 0.8 | 21.1 | 6.4 |
| Gasch2 | 52.0 | 10.0 | 4.8 | 17.5 | 5.3 |
| Phenotype | 67.0 | 4.6 | 1.7 | 14.6 | 9.3 |
| Eisen | 79.0 | 1.6 | 0.5 | 27.0 | 6.6 |
| Derisi | 63.0 | 2.0 | 1.6 | 3.7 | 6.5 |
| Gasch1 | 173.0 | 5.6 | 2.2 | 23.4 | 4.9 |
| Sequence | 478.0 | 9.6 | 2.3 | 37.1 | 6.4 |
| Expression | 551.0 | 7.6 | 2.1 | 24.0 | 3.5 |
| ImageCLEF07A | 80.0 | 60.8 | 8.3 | 63.7 | 2.8 |
| ImageCLEF07D | 80.0 | 27.0 | 3.8 | 31.1 | 2.8 |

## 4.7 Final considerations

In this chapter, we presented a novel feature selection method tailored for global model hierarchical classifiers. We developed two hybrid filter-wrapper approaches based on the VNS metaheuristic, so-called VNS-FSHC and GVNS-FSHC, which use the $SU_H$ measure in a filter step, and the GMNB or the CLUS-HMC as the classifier of a wrapper step. We compare the GVNS-FSHC method with different strategies on twelve datasets (from protein and image context).

The experimental results using the GVNS-FSHC algorithm with the GMNB classifier showed that the predictive performance was consistently better or equivalent to the other comparison strategies. Furthermore, the GVNS-FSHC reduced the number of features in all datasets without negatively impacting the classification accuracy.

We also observed that the GVNS-FSHC is better or equivalent to the VNS-FSHC algorithm concerning the classifier's predictive performance. Moreover, when we considered the running times' behavior, the GVNS-FSHC performed better than the VNS-FSHC since it reached the improvements first.

Concerning the CLUS-HMC classifier, the GVNS-FSHC feature selection method did not improve the classification performance, showing decision trees' power as natural feature selectors. However, the GVNS-FSHC was able to select fewer features with no statistically significant difference in the performance results.

Chapter 5

# Hybrid hierarchical feature selection methods using a new correlation-based measure

Next, we discuss our proposed hybrid feature selection methods that combine an adaptation of the CFS search-based filter measure with wrapper evaluations to solve the FSHC problem. We aim to investigate the effectiveness of incorporating the adapted filter measure to evaluate feature subsets in a hybrid feature selection approach that runs the classifier (wrapper phase) less often, reducing computational costs.

This chapter is organized as follows. Section 5.1 presents the *Hierarchical Single-label Correlation-based Feature Selection* (H-CFS), our proposed measure. Section 5.3 and Section 5.2 provide a detailed description of the proposed algorithms, *Best First Search for Hierarchical Single-label Correlation-based Feature Selection* (BFS-H-CFS) and *Genetic Algorithm for Hierarchical Single-label Correlation-based Feature Selection* (GA-H-CFS), respectively. Finally, Section 5.4 describes the experimental setup, Section 5.5 reports the computational results, and Section 5.6 concludes the chapter.

## 5.1 Hierarchical Correlation-based Feature Selection

This section presents the H-CFS measure, an adaptation of the well-known CFS measure proposed by Hall (2000). As in CFS (Section 2.2.1.2), the evaluation of a feature subset (merit) is estimated using Equation (12), where $\overline{r_{FL}}$ is the quotient of the average feature-label correlation by the number of pairs feature-label, $\overline{r_{FF}}$ is the quotient of the average feature-feature correlation by the number of pairs feature-feature, $F$ is the evaluated feature subset, $L$ is the set of class labels, and $k_F$ is the number of features in

$F$.

$$merit = \frac{k_F \overline{r_{FL}}}{\sqrt{k_F + k_F(k_F - 1)\overline{r_{FF}}}} \tag{12}$$

Equation (13) describes the $\overline{r_{FF}}$ correlation. As in CFS, the correlation $r_{f_i f_j}$ is computed for each pair of features $(f_i, f_j)$ in the dataset, then the results are averaged dividing the total summation of all results by the number of pairs of features, denoted by $p$.

$$\overline{r_{FF}} = \frac{\sum_{i,j=1, i>j}^{k_F} |r_{f_i f_j}|}{p} \tag{13}$$

The difference of H-CFS to CFS is in the way that we estimate the term $\overline{r_{FL}}$. The main idea is that we calculate the average feature-label correlation using the arithmetic mean of all feature-label pairs considering the hierarchical relation among classes by using Equations (14) and (15). The conventional CFS measure computes the $\overline{r_{FL}}$ considering the mean of the correlation between each feature in $F$ and each single class, using only Equation (14). Differently, H-CFS uses Equation (15) to calculate the average value of the correlation coefficient between each feature in a candidate subset $F$ and each class in the set of all class labels, weighting each class by its level in the class hierarchy.

$$\overline{r_{FL}} = \frac{\sum_{f_i=1}^{k_F} |r_{f_i \bar{L}}|}{k_F} \tag{14}$$

$$r_{f\bar{L}} = \frac{\sum_{i=1}^{h} \sum_{j=1}^{k_{L_i}} w_i |r_{fL_{i,j}}|}{\sum_{i=1}^{h} w_i k_{L_i}} \tag{15}$$

where $L_i$ is the set of classes in level $i$, $h$ is the number of levels in the hierarchy, $k_{L_i}$ is the number of classes in $L_i$, and $w_i$ is the weight assigned to level $i$.

Equation (16) describes the calculation of the class weights $w_i$, with $0 \leq w_0 \leq 1$ and $1 \leq i \leq h$. The idea is that $w_i$ decreases with the depth of the level $i$ in the class hierarchy. Vens et al. (2008) proposed this weighting scheme in the Euclidean distance measure of the CLUS-HMC classifier. As in the referenced work, we have arbitrarily set $w_0 = 0.75$.

$$w_i = w_0^i \tag{16}$$

To estimate the correlation between two features $X$ and $Y$, we used the Pearson's correlation coefficient ($r_{XY}$). Equation (17) describes how to estimate $r_{XY}$ when $X$ and $Y$ are two continuous variables.

$$r_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{17}$$

where $x_i$ and $\bar{x}$ are the value of variable $X$ in the $i$th instance and the average value of $X$, respectively. Similarly, $y_i$ and $\bar{y}$ are the value of variable $Y$ in the $i$th instance and the average value of $Y$, and $n$ is the number of instances in the training set.

When one feature is continuous, and the other discrete, a weighted Pearson's correlation is calculated according to Equation (18). Specifically, for a discrete feature $X$ and a continuous feature $Y$, if $X$ has $k$ values, then $k$ binary features are correlated with $Y$. Each of the $X_{bi}$ binary features takes value 1 when the $i$th value of $X$ occurs and 0 for all other values. Each of the $r_{X_{bi}Y}$ is weighted by the prior probability that $X$ takes value $i$ (HALL, 2000).

$$r_{XY} = \sum_{i=1}^{k} p(X = x_i) r_{X_{bi}Y} \tag{18}$$

Equation (19) describes the Pearson's correlation when both features involved are discrete. We create binary features for both and calculate all weighted correlations for all combinations.

$$r_{XY} = \sum_{i=1}^{k} \sum_{j=1}^{l} p(X = x_i, Y = y_j) r_{X_{bi}Y_{bj}} \tag{19}$$

It is worth mentioning that we used the absolute values of the correlation coefficient in all $r_{XY}$ occurrences. As assumed by Jungjit et al. (2013), it is advisable to use absolute values of correlations because both positive and negative values can represent redundancy between a pair of features. For instance, if a feature subset contains one pair of features $(X, Y)$ with $r_{XY} = 0.8$ and another pair of features $(X, W)$ with $r_{XW} = -0.8$, these two values would cancel each other, resulting in an average $r$ over those two feature pairs of 0, a misleading value, since the two correlation values suggest a significant degree of redundancy in each of those feature pairs.

## 5.2 GA approach using H-CFS

This section presents the *Genetic Algorithm for Hierarchical Single-label Correlation-based Feature Selection* (GA-H-CFS) approach. GAs are stochastic (non-deterministic) global search methods inspired by the process of natural selection, based on Darwin's evolutionary theory (EIBEN; SMITH, 2015). This metaheuristic has been applied efficiently as a search-based feature selection approach for flat classification problems (XUE et al., 2016).

The main idea of GAs is that the individuals better adapted to their environment will survive in nature, while the remaining ones will vanish with time. GA starts with a random individual initialization process which generates a population of individuals, where each individual is a candidate solution to the target problem. Next, GA selects parents from the set of individuals using some selection approach, as tournament selection or roulette wheel approach. Then, crossover and mutation operations are applied to selected parents to create new individuals. Finally, GA selects survival individuals from offspring individuals and goes to the next generation.

In the proposed GA-H-CFS, each individual (candidate feature subset) is represented by a string of $n$ bits, where $n$ is the number of features. The $i$th bit ($i = 1, \ldots, n$) takes the value 1 if a feature is selected or 0 otherwise. Each individual is evaluated by a fitness function, given by H-CFS, described in Section 5.1. At each generation (iteration), individuals are selected by the roulette wheel selection operator, which selects individuals with a probability proportional to their fitness (quality) values. The selected individuals then undergo one-point crossover or bit-flip mutation with a predefined probability. The selection, crossover, and mutation operators are conventional GA operators (EIBEN; SMITH, 2015).

The main novelty of the proposed GA is the use of the H-CFS as a fitness function. Additionally, we add a wrapper phase using the GMNB classifier, and therefore the $hF$ measure (described in Section 4.1) works as a second fitness function. We run this evaluation measure only to the best solution found in each generation and return the best subset found by the wrapper phase when the search terminates. The reasoning behind this idea is to execute the wrapper evaluation, which is the most computational costly step, only in strategic points of the search.

## 5.3 BF approach using H-CFS

This section presents the *Best First Search for Hierarchical Single-label Correlation-based Feature Selection* (BFS-H-CFS) approach.

The Best First Search (BFS) is a well-known heuristic search method used for feature selection (LIU; MOTODA, 2007). The BFS forward model starts with an empty set of features and generates all possible single subset feature candidates. Then, the subset with the highest evaluation is chosen and expanded by adding a single feature to generate all possible subset candidates. Expanded subsets go to the CLOSED list of nodes. Additionally, the remaining subsets are added to the OPEN list of candidates nodes to expand. If expanding the current subset results in no improvement, the search backtracks to the best-unexpanded subset in the OPEN list and continues from there. Given enough time, the BFS algorithm will explore the entire feature subset space and terminate when the OPEN list is empty. Thus, it is common to limit backtracking to an unexpanded subset that results in no improvement. The best subset found is returned when the search terminates.

The main novelty of the proposed BFS is the use of the H-CFS as an evaluation function. Additionally, we add a wrapper phase with the GMNB classifier, using the $hF$ measure (described in Section 4.1) to evaluate candidate subsets of features. We run this second evaluation measure only when the BFS decides to backtrack to a subset in the OPEN list considering the evaluations performed by the H-CFS measure. At that point, we use the wrapper evaluation to the current candidate subset and also to the

best-unexpanded subset in OPEN. The BFS will expand the best-unexpanded subset from OPEN if the wrapper evaluation endorses the backtracking decision. If not, it will expand the current subset and remain in the same branch. Similar to the GA-H-CFS approach, the reasoning behind this idea is to execute the wrapper evaluation only in strategic points of the search. The best subset found is returned when the search terminates. We chose a predefined number of backtracking without improvements as the stopping criterion of the algorithm.

## 5.4 Experimental setup

The GA-H-CFS and BFS-H-CFS algorithms were implemented in C++, using the compiler g++, version 4.8.5 for its execution. The experiments were performed on a computer with Intel Xeon(R) CPU E5620 @ 2.40GHz × 16, 48 GB of RAM, and CentOS Linux 7 operational system. Although this computer processor has more than one core, the algorithm was not optimized for multi-processing.

We used the GMNB hierarchical classifier to evaluate the quality of the selected features. Based on the evaluation metrics, hierarchical Precision and hierarchical Recall, described in Section 4.1, we compared the proposed algorithms against the following feature selection strategies:

(i) **ALL**: we measured the performance of the classifier without any feature selection preprocessing step, i.e., using all features from the dataset.

(ii) **BFS-NW**: a version of the BFS-H-CFS algorithm described in Section 5.3 with No-Wrapper (NW) evaluations to investigate the performance of the H-CFS measure as a unique evaluation function of this hybrid feature selection approach. We chose a predefined number of backtracking without improvements as the stopping criterion of the algorithm.

Our experiments and comparisons use the same datasets and prepossessing steps described in Section 4.6.1. We used the same 5-fold cross-validation setup for each dataset. The best feature subset for both algorithms was selected using the 5-fold cross-validation procedure within the training set when using the wrapper evaluation. Moreover, we used each training set fold to estimate the H-CFS evaluation.

The parameter settings of GA-H-CFS were: population size = 500, number of generations = 300, crossover probability = 0.9, mutation probability = 0.01. We did preliminary experiments varying the number of generations from $\{100, 200, 300\}$ in all datasets and chose the parameter value that generated the best results.

Regarding the BFS-H-CFS algorithm, we did preliminary experiments varying the stopping criterion, which is a predefined number of backtracking without improvements,

Table 8 – $hF$ results for BFS-H-CFS and GA-H-CFS

| | **ALL** | **BFS-NW** | **BFS-H-CFS** | **GA-H-CFS** |
|---|---|---|---|---|
| ***Dataset*** | ***avg* (*sd*)** | ***avg* (*sd*)** | ***avg* (*sd*)** | ***avg* (*sd*)** |
| CellCycle | • 25.23 (1.1) | 12.99 (0.8) | 24.90 (1.9) | • **25.79** (1.6) |
| Church | 14.59 (3.5) | 13.91 (1.6) | 18.05 (4.3) | • **18.24** (5.2) |
| SPO | 16.57 (1.2) | 12.75 (1.9) | **19.10** (0.6) | • 19.0 (1.4) |
| Gasch2 | • 19.68 (1.3) | 15.04 (0.8) | **18.55** (1.1) | • 18.21 (1.0) |
| Phenotype | 10.09 (1.9) | • 15.08 (2.3) | **16.11** (1.2) | • 15.85 (2.2) |
| Eisen | • 23.78 (2.1) | 18.0 (1.6) | **24.35** (1.8) | 22.07 (0.7) |
| Derisi | • **14.75** (1.4) | 12.75 (1.1) | 14.08 (0.9) | • 14.26 (0.6) |
| Gasch1 | 24.87 (1.7) | • 27.16 (1.8) | **29.40** (1.7) | 26.31 (1.5) |
| Sequence | 19.96 (0.7) | 10.85 (1.2) | **25.02** (1.6) | 20.86 (0.2) |
| Expression | 36.24 (1.5) | 25.46 (3.9) | **45.36** (2.6) | 34.07 (4.6) |
| ImageCLEF07A | **80.37** (1.2) | 54.68 (1.9) | 67.92 (1.2) | 77.40 (1.6) |
| ImageCLEF07D | • 63.04 (0.9) | 55.37 (3.8) | 62.37 (1.7) | • **63.31** (0.5) |

Table 9 – Number of selected features for BFS-H-CFS and GA-H-CFS

| | | **BFS-NW** | | **BFS-H-CFS** | | **GA-H-CFS** | |
|---|---|---|---|---|---|---|---|
| ***Dataset*** | ***All*** | ***avg*** | ***sd*** | ***avg*** | ***sd*** | ***avg*** | ***sd*** |
| CellCycle | 77.0 | 4.2 | 6.4 | 17.0 | 5.7 | 27.8 | 6.2 |
| Church | 27.0 | 6.0 | 0.7 | 9.6 | 0.5 | 8.8 | 1.0 |
| SPO | 79.0 | 6.4 | 1.2 | 6.8 | 0.8 | 7.6 | 1.6 |
| Gasch2 | 52.0 | 6.4 | 3.1 | 19.0 | 0.9 | 18.6 | 1.5 |
| Phenotype | 67.0 | 13.6 | 3.1 | 34.0 | 9.4 | 24.2 | 3.6 |
| Eisen | 79.0 | 10.8 | 0.5 | 46.6 | 0.8 | 34.0 | 3.0 |
| Derisi | 63.0 | 9.8 | 2.7 | 16.4 | 3.5 | 23.0 | 2.2 |
| Gasch1 | 173.0 | 11.0 | 2.2 | 24.4 | 6.1 | 78.6 | 2.7 |
| Sequence | 478.0 | 46.6 | 2.3 | 91.0 | 4.4 | 225.8 | 6.5 |
| Expression | 551.0 | 12.7 | 3.4 | 36.2 | 11.5 | 268.4 | 14.1 |
| ImageCLEF07A | 80.0 | 7.4 | 4.8 | 46.2 | 2.5 | 36.8 | 3.8 |
| ImageCLEF07D | 80.0 | 7.0 | 1.2 | 15.2 | 3.6 | 32.0 | 3.2 |

from $\{10, 20, 30\}$ in all datasets and chose the parameter value that generated the best results (stopping criterion = 30). Additionally, for the BFS-NW algorithm, we fixed the stopping criterion = 30.

## 5.5 Computational results

All the GA-H-CFS results are averages of 10 runs with a different random seed used to create the initial population in each run. As the BFS-H-CFS and the BFS-NW heuristics are deterministic, only one execution was required for each training set.

We compared the results obtained in each dataset using two one-way hypothesis tests with a significance level of 0.05. We chose the most appropriate statistical test for each dataset result as described in Section 4.6.3.

In general, the BFS-H-CFS algorithm obtained the best predictive performances in our experiments. Differently, the BFS-NW performed worst, showing the importance of the wrapper evaluation phase to improve the performance of a feature selection approach. Table 8 presents the $hF$ values reached by the GMNB classifier using all the dataset features (second column) and feature selection (third to fifth columns). In these columns, "$avg$" indicates the average result, with the standard deviation ($sd$) in parentheses. Bold results show the best absolute value, and a result preceded by • indicates no statistically significant difference between the specific result and the BFS-H-CFS result.

The experiments showed that the BFS-H-CFS algorithm obtained the best absolute average for seven datasets (SPO, Gasch2, Phenotype, Eisen, Gasch1, Sequence, and Expression). For the remaining ones (CellCycle, Church, Derisi, ImageCLEF07A, and ImageCLEF07D), its performance was equivalent to the best result found for four datasets (CellCycle, Church, Derisi, and ImageCLEF07D), i.e., the difference was not statistically significant. The only exception to this occurs for the ImageCLEF07A dataset, in which any feature selection approach improved the performance of the GMNB classifier using all features.

Table 9 shows the comparison results between the algorithms concerning the number of features used by the GMNB classifier. The second column presents the number of features used without feature selection, and the remaining ones (third to eighth columns) represent both the average ($avg$) and standard deviation ($sd$) of the number of features used by the algorithms.

When we compare the results of Table 9 regarding Table 8, we observed that when the BFS-H-CFS has equivalent performance concerning the $hF$ measure with the GA-H-CFS (CellCycle, Church, SPO, Gasch2, Phenotype, Derisi, and ImageCLEF07D), it selects fewer features for four of those seven datasets, confirming its advantage over the GA-H-CFS approach.

Moreover, when the dataset's initial number of features increases, the number of features selected by the GA-H-CFS increases much faster than the number selected by the BFS-H-CFS (e.g., Gasch1, Sequence, and Expression). One possible explanation for this is that the BFS heuristic adds one feature in the current candidate feature subset at a time. On the contrary, the GA metaheuristic search may add many features to a given individual at once using the crossover operator. In theory, this gives the GA the advantage of coping better with feature interactions, but this comes with the disadvantage that the extra features added to a candidate subset in a single crossover operation may include some features less relevant to class prediction.

### 5.5.1 GVNS-FSHC × BFS-H-CFS

Table 10 shows the results obtained by our best approaches, the GVNS-FSHC, and the BFS-H-CFS algorithms. From the second to the third column, we represent the $hF$ values

Table 10 – $hF$ results for GVNS-FSHC and BFS-H-CFS with GMNB classifier

| | **ALL** | **GVNS-FSHC** | **BFS-H-CFS** |
|---|---|---|---|
| ***Dataset*** | ***avg* (*sd*)** | ***avg* (*sd*)** | ***avg* (*sd*)** |
| CellCycle | 25.23 (1.1) | • **25.27** (2.4) | • 24.90 (1.9) |
| Church | 14.59 (3.5) | **22.28** (4.5) | 18.05 (4.3) |
| SPO | 16.57 (1.2) | • **20.42** (0.8) | • 19.10 (0.6) |
| Gasch2 | **19.68** (1.3) | • 18.71 (1.3) | • 18.55 (1.1) |
| Phenotype | 10.09 (1.9) | • **16.70** (1.9) | • 16.11 (1.2) |
| Eisen | 23.78 (2.1) | • 21.84 (1.9) | • **24.35** (1.8) |
| Derisi | 14.75 (1.4) | **16.42** (1.1) | 14.08 (0.9) |
| Gasch1 | 24.87 (1.7) | • 28.04 (2.0) | • **29.40** (1.7) |
| Sequence | 19.96 (0.7) | 19.46 (0.6) | **25.02** (1.6) |
| Expression | 36.24 (1.5) | **48.12** (2.4) | 45.36 (2.6) |
| ImageCLEF07A | 80.37 (1.2) | **80.67** (0.8) | 67.92 (1.2) |
| ImageCLEF07D | 63.04 (0.9) | **66.78** (0.7) | 62.37 (1.7) |

reached by the GMNB classifier using all the dataset features (second column) and feature selection (other two columns). In these columns, "*avg*" indicates the average result, with the standard deviation (*sd*) in parentheses. Bold results show the best absolute value, and a result preceded by • indicates no statistically significant difference between the GVNS-FSHC and the BFS-H-CFS performances.

Table 10 shows that both algorithms have equivalent performance for six datasets (CellCycle, SPO, Gasch2, Phenotype, Eisen, and Gasch1). For the remaining ones, the GVNS-FSHC performs better for five of six datasets compared to BFS-H-CFS (Church, Derisi, Expression, ImageCLEF07A, and ImageCLEF07D). This result was already expected since the GVNS-FSHC runs the classifier evaluation in the wrapper phase more times than BFS-H-CFS, with the drawback of being more computationally expensive.

## 5.6 Final considerations

This chapter presented two novel hybrid filter-wrapper approaches based on GA and BFS algorithms that combine the H-CFS filter measure with a wrapper step to work as fitness functions for FSHC.

The experimental results using the BFS-H-CFS algorithm with the GMNB classifier showed that the predictive performance was consistently better or equivalent than other comparison strategies (except for the ImageCLEF07A dataset). Furthermore, the BFS-H-CFS reduced the number of features in eleven of twelve datasets without negatively impacting the classification performance.

Concerning the BFS-NW algorithm results, we observed that the H-CFS feature selection measure alone could not improve the classifier performance for most datasets (the only exceptions are Phenotype and Gasch1 datasets). On the other hand, when we compare

the BFS-NW results with the BFS-H-CFS approach, which differs from the BFS-NW only in the addition of the wrapper phase, it is possible to observe a consistency improvement of the method in terms of classifier performance.

Regarding the GA-H-CFS approach, we observed that the number of features selected tends to increase rapidly with the problem size (number of initial features of datasets). At the same time, the BFS-H-CFS is less sensitive to this issue.

As future work, we plan to develop an evolutionary computation approach that will use a fitness function with two objectives, the candidate subset evaluation and the number of selected features, to prevent the search strategy from selecting too many features at once.

Chapter 6

# Conclusions and future works

This work has focused on the development of feature selection methods tailored for global hierarchical classifiers. In Chapter 4, we have proposed two hybrid algorithms based on the VNS metaheuristic, named VNS-FSHC and GVNS-FSHC, which use the $SU_H$ measure in a filter step and two global model classifiers (GMNB or CLUS) in a wrapper step.

Computational experiments were carried out with twelve datasets from protein and image domains to evaluate the effect of the proposed algorithms on classification performance when using global hierarchical classifiers.

When using the GMNB classifier, the GVNS-FSHC algorithm was consistently better or equivalent to the other comparison strategies. Furthermore, the GVNS-FSHC reduced the number of features in all datasets without negatively impacting the classification accuracy.

Regarding the CLUS-HMC classifier, the GVNS-FSHC feature selection method did not improve the classification performance, showing the power of decision trees as natural feature selectors. However, it was able to select fewer features with no statistically significant difference in the performance results.

Even though we obtained positive results with this approach, we faced the drawback of scalability to high-dimensional datasets. Aiming to tackle this issue, we focused on developing a new hybrid feature selection approach capable of running the wrapper evaluation less often to reduce computational cost.

In Chapter 5, we presented two novel hybrid filter-wrapper approaches based on GA and BFS algorithms, so-called GA-H-CFS and BFS-H-CFS, respectively. These methods combine the H-CFS filter measure, an adaptation of the traditional CFS measure, with a wrapper step, working as an alternative fitness function for FSHC. The reasoning behind this approach is to execute the wrapper evaluation, which is the most computational costly step, only in strategic points of the search algorithms.

The experimental results showed that the BFS-H-CFS algorithm performed better with the GMNB classifier compared to the other strategies. Furthermore, the BFS-H-CFS

reduced the number of features in eleven of twelve datasets without negatively impacting the classification performance.

Considering the GA-H-CFS approach, we faced the drawback that this method tends to select subsets with a higher number of features that increase rapidly with the database size (number of initial features of datasets).

As future work, we plan to develop an evolutionary computation approach that will use a fitness function with two objectives, the candidate subset evaluation and the number of selected features, to prevent the search strategy from selecting too many features at once. Additionally, we intend to carry out experiments with the GA-H-CFS approach using other types of crossover and mutation operators.

We also intend to investigate if the BFS-H-CFS algorithm remains efficient for datasets with a significantly higher number of features.

Furthermore, given the positive results of this work for hierarchical single-label problems, we can conclude that an extended version for multi-label problems should be investigated. Also, our feature selection method can be useful to solve extreme multi-label classification problems (BHATIA et al., 2016), in which the objective is to learn feature architectures and classifiers that can automatically tag a data point with the most relevant subset of labels from an extremely large label set.

## 6.1 Academic publications

The research described in this thesis has led to the production of two peer-reviewed papers. The first one was published in the proceedings of an international conference. The second paper was published in an international journal. Additionally, one last paper is in preparation for submission to an international conference.

- COSTA, H.; GALVÃO, L. R. ; MERSCHMANN, L. H. C. ; SOUZA, Marcone J. F.(2018). A VNS algorithm for feature selection in hierarchical classification context. Electronic Notes in Discrete Mathematics, v. 66, p. 79-86. Proceedings of the 5th International Conference on Variable Neighborhood Search, 2017. [Qualis B3 - journal]

- LIMA, H. C. S. C.; OTERO, F. E. B.; MERSCHMANN, L. H. C.; SOUZA, Marcone J. F. A novel hybrid feature selection algorithm for hierarchical classification. IEEE Access, 2021. [Qualis A2 - journal]

- LIMA, H. C. S. C.; OTERO, F. E. B.; MERSCHMANN, L. H. C.; SOUZA, Marcone J. F. A new hierarchical single-label correlation-based hybrid feature selection. [In preparation]

# References

AGRAWAL, P. et al. Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). **IEEE Access**, v. 9, p. 26766–26791, 2021.

APOLLONI, J.; LEGUIZAMÓN, G.; ALBA, E. Two hybrid wrapper-filter feature selection algorithms applied to high-dimensional microarray experiments. **Appl. Soft Comput.**, v. 38, n. C, p. 922–932, 2016.

BERMEJO, P.; GÁMEZ, J. A.; PUERTA, J. M. A grasp algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets. **Pattern Recognition Letters**, v. 32, n. 5, p. 701 – 711, 2011.

BHATIA, K. et al. **The extreme classification repository: Multi-label datasets and code**. 2016. Available from Internet: <http://manikvarma.org/downloads/XC/XMLRepository.html>.

BLUM, A. L.; LANGLEY, P. Selection of relevant features and examples in machine learning. **Artif. Intell.**, Elsevier Science Publishers Ltd., Essex, UK, v. 97, n. 1-2, p. 245–271, dez. 1997.

CERRI, R. et al. Multi-label feature selection techniques for hierarchical multi-label protein function prediction. In: **International Joint Conference on Neural Networks (IJCNN)**. Rio de Janeiro: [s.n.], 2018. p. 1–7.

CHEN, Y.-L.; HU, H.-W.; TANG, K. Constructing a decision tree from data with hierarchical class labels. **Expert Systems with Applications**, v. 36, n. 3, Part 1, p. 4838 – 4847, 2009.

CLARE, A.; KING, R. D. Predicting gene function in saccharomyces cerevisiae. **Bioinformatics**, Oxford Univ Press, v. 19, n. suppl 2, p. ii42–ii49, 2003.

COSTA, E. P. et al. Comparing several approaches for hierarchical classification of proteins with decision trees. In: **Advances in Bioinformatics and Computational Biology**. Berlin: Springer Berlin Heidelberg, 2007. p. 126–137.

COSTA, H. et al. A vns algorithm for feature selection in hierarchical classification context. **Electronic Notes in Discrete Mathematics**, v. 66, p. 79–86, 2018. 5th International Conference on Variable Neighborhood Search.

COVER, T. M.; THOMAS, J. A. **Elements of Information Theory**. New York, NY, USA: Wiley-Interscience, 1991. ISBN 0-471-06259-6.

DEBUSE, J. C. W.; RAYWARD-SMITH, V. J. Feature subset selection within a simulated annealing data mining algorithm. **Journal of Intelligent Information Systems**, Springer Nature, v. 9, p. 57–81, 1997.

DIAS, T. N.; MERSCHMANN, L. H. C. Adaptation of the symmetric uncertainty measure for feature selection in single-label hierarchical classification context (in portuguese). In: **Annals of the National Meeting of Artificial Intelligence and Computational**. Natal: [s.n.], 2015. p. 142–149.

DIMITROVSKI, I. et al. Hierarchical annotation of medical images. **Pattern Recogn.**, Elsevier Science Inc., v. 44, n. 10–11, p. 2436–2449, 2011.

DUDA, R. O.; HART, P. E. **Pattern recognition and scene analysis**. [S.l.]: Wiley, New York, 1973.

EIBEN, A. E.; SMITH, J. E. **Introduction to Evolutionary Computing**. 2nd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2015.

FEO, T. A.; RESENDE, M. G. C.; SMITH, S. H. A greedy randomized adaptive search procedure for maximum independent set. **Operations Research**, v. 42, n. 5, p. 860–878, 1994.

GALVÃO, L.; MERSCHMANN, L. H. C. Hsim: A supervised imputation method for hierarchical classification scenario. In: **Proceedings of the 19th International Conference on Discovery Science**. Bari: [s.n.], 2016. p. 134–148.

GARNER, S. Weka: The waikato environment for knowledge analysis. In: **Proceedings of the New Zealand Computer Science Research Students Conference**. [s.n.], 1995. p. 57–64. Available from Internet: <https://www.cs.waikato.ac.nz/~ml/publications/1995/Garner95-WEKA.pdf>.

GLOVER, F.; LAGUNA, M. **Tabu Search**. Norwell, Massachusetts: Kluwer Academic Publishers, 1997.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

HALL, M. et al. The weka data mining software: An update. **ACM SIGKDD Explorations Newsletter**, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, 2009.

HALL, M. A. Correlation-based feature selection for discrete and numeric class machine learning. In: **Proceedings of the Seventeenth International Conference on Machine Learning**. San Francisco: Morgan Kaufmann Publishers Inc., 2000. p. 359–366.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

HANSEN, P. et al. Variable neighborhood search. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. Cham: Springer International Publishing, 2019. p. 57–97.

HEIBERGER, R. M. **Computation for the analysis of designed experiments**. New York, NY, USA: John Wiley & Sons, 2015. (Wiley series in probability and statistics).

HOLLANDER, M.; WOLFE, D. A. **Nonparametric Statistical Methods**. New York: John Wiley & Sons, 1973.

HUANG, C. et al. An efficient automatic multiple objectives optimization feature selection strategy for internet text classification. **International Journal of Machine Learning and Cybernetics**, Springer Nature, 2019.

HUANG, H.; LIU, H. Feature selection for hierarchical classification via joint semantic and structural information of labels. **Knowledge-Based Systems**, v. 195, p. 105655, 2020.

JUNGJIT, S.; FREITAS, A. A lexicographic multi-objective genetic algorithm for multi-label correlation based feature selection. In: **Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation**. Madrid: Association for Computing Machinery, 2015. p. 989–996.

JUNGJIT, S. et al. Two extensions to multi-label correlation-based feature selection: A case study in bioinformatics. In: **2013 IEEE International Conference on Systems, Man, and Cybernetics**. [S.l.: s.n.], 2013. p. 1519–1524.

KIRITCHENKO, S.; MATWIN, S.; FAMILI, A. F. Functional annotation of genes using hierarchical text categorization. In: **Proceedings of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology (in association with ISMB-05)**. Detroit: [s.n.], 2005.

KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. **Artificial Intelligence**, Elsevier Science Publishers Ltd., Essex, UK, v. 97, n. 1-2, p. 273–324, dez. 1997.

KOLLER, D.; SAHAMI, M. Toward optimal feature selection. In: **13th International Conference on Machine Learning**. San Francisco: Morgan Kaufmann Publishers Inc., 1995. p. 284–292.

KOLLER, D.; SAHAMI, N. Hierarchically classifying documents using very few words. In: **14th International Conference on Machine Learning**. San Francisco: Morgan Kaufmann Publishers Inc., 1997. p. 170–178.

LIU, H.; MOTODA, H. **Computational Methods of Feature Selection**. [S.l.]: Chapman & Hall/CRC, 2007. (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series).

LIU, H. et al. Feature selection: An ever evolving frontier in data mining. In: **Proceedings of the Fourth International Workshop on Feature Selection in Data Mining**. Hyderabad: PMLR, 2010. (Proceedings of Machine Learning Research, v. 10), p. 4–13.

LIU, H.; SETIONO, R. A probabilistic approach to feature selection - a filter solution. In: **Proceedings of the Thirteenth International Conference on Machine Learning**. San Francisco: Morgan Kaufmann, 1996. p. 319–327.

LÓPEZ-IBÁÑEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016.

LU, H. et al. A hybrid feature selection algorithm for gene expression data classification. **Neurocomputing**, v. 256, n. C, p. 56–62, 2017.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, Elsevier Science Ltd., Oxford, UK, UK, v. 24, n. 11, p. 1097–1100, nov. 1997. ISSN 0305-0548.

OTERO, F. E. B.; FREITAS, A. A.; JOHNSON, C. C. A hierarchical multi-label classification ant colony algorithm for protein function prediction. **Memetic Computing**, Springer Berlin Heidelberg, Berlin, v. 2, n. 3, p. 165–181, 2010.

OTERO, F. E. B.; FREITAS, A. A.; JOHNSON, C. G. **A Hierarchical Classification Ant Colony Algorithm for Predicting Gene Ontology Terms**. Berlin: Springer Berlin Heidelberg, 2009. 68–79 p.

PAES, B. C.; PLASTINO, A.; FREITAS, A. A. Exploring attribute selection in hierarchical classification. **Journal of Information and Data Management**, v. 5, n. 1, p. 124–133, 2014.

REINSMA, J. A.; PENNA, P. H. V.; SOUZA, M. J. F. A simple and efficient algorithm to solve the generalized traveling salesman problem (in portuguese). In: **Proceedings of the L Brazilian Symposium of Operations Research**. Rio de Janeiro: SOBRAPO, 2018.

REYES, A.; RIBEIRO, C. C. Extending time-to-target plots to multiple instances. **International Transactions in Operational Research**, v. 25, n. 5, p. 1515–1536, 2018.

ROBNIK-ŠIKONJA, M.; KONONENKO, I. Theoretical and empirical analysis of relieff and rrelieff. **Machine Learning**, Kluwer Academic Publishers, Hingham, MA, USA, v. 53, n. 1-2, p. 23–69, 2003.

ROYSTON, J. P. An extension of shapiro and wilk's w test for normality to large samples. **Journal of the Royal Statistical Society. Series C (Applied Statistics)**, Wiley, Royal Statistical Society, v. 31, n. 2, p. 115–124, 1982.

RUIZ, R.; RIQUELME, J. C.; AGUILAR-RUIZ, J. S. Incremental wrapper-based gene selection from microarray data for cancer classification. **Pattern Recognition**, v. 36, n. 12, p. 2383–2392, 2006.

SANTOS, M. S. et al. Simheuristic-based decision support system for efficiency improvement of an iron ore crusher circuit. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 94, p. 103789, 2020.

SECKER, A. et al. Hierarchical classification of G-Protein-Coupled Receptors with data-driven selection of attributes and classifiers. **International Journal of Data Mining and Bioinformatics**, v. 4, p. 191–210, 2010.

SILLA, C. N. J.; FREITAS, A. A survey of hierarchical classification across different application domains. **Data Mining and Knowledge Discovery**, Springer US, v. 22, n. 1-2, p. 31–72, 2011.

SILLA, C. N. J.; FREITAS, A. A. A global-model naive bayes approach to the hierarchical prediction of protein functions. In: **Proceedings of the Ninth IEEE International Conference on Data Mining (ICDM-2009)**. Miami Beach: IEEE Press, 2009. p. 182–196.

SLAVKOV, I. et al. Relieff for hierarchical multi-label classification. In: **New Frontiers in Mining Complex Patterns**. [S.l.]: Springer International Publishing, 2014, (Lecture Notes in Computer Science). p. 148–161.

SONG, X.-F. et al. A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data. **IEEE Transactions on Cybernetics**, p. 1–14, 2021.

TUO, Q.; ZHAO, H.; HU, Q. Hierarchical feature selection with subtree based graph regularization. **Knowledge-Based Systems**, v. 163, p. 996–1008, 2019.

VENS, C. et al. Decision trees for hierarchical multi-label classification. **Machine Learning**, v. 73, n. 2, p. 185, 2008.

WU, Q. et al. A feature selection method based on hybrid improved binary quantum particle swarm optimization. **IEEE Access**, v. 7, p. 80588–80601, 2019.

XUE, B. et al. A survey on evolutionary computation approaches to feature selection. **IEEE Transactions on Evolutionary Computation**, v. 20, n. 4, p. 606–626, 2016.

YANG, Y.; PEDERSEN, J. O. A comparative study on feature selection in text categorization. In: **Proceedings of the Fourteenth International Conference on Machine Learning**. San Francisco: Morgan Kaufmann Publishers Inc., 1997. p. 412–420.

YANG, Y.; WEBB, G. I. Proportional k-interval discretization for naive-bayes classifiers. In: **12th European Conference on Machine Learning**. Berlin: Springer, 2001. (LNCS, v. 2167), p. 564–575.

ZHAO, H. et al. Hierarchical feature selection with recursive regularization. In: **Proceedings of the 26th International Joint Conference on Artificial Intelligence**. Melbourne: AAAI Press, 2017. p. 3483–3489.

ZHENG, W.; ZHAO, H. Cost-sensitive hierarchical classification for imbalance classes. **Applied Intelligence**, Springer Nature, v. 50, p. 2328–2338, 2020.