IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A novel hybrid feature selection algorithm for hierarchical classification

HELEN C. S. C. LIMA[1], FERNANDO E. B. OTERO[2], LUIZ H. C. MERSCHMANN[3], AND MARCONE J. F. SOUZA[1]

[1]Department of Computing, Federal University of Ouro Preto, Ouro Preto 35400-000, Brazil (e-mail: {helen, marcone}@ufop.edu.br)
[2]School of Computing, University of Kent, Canterbury ME44AG, United Kingdom (e-mail: F.E.B.Otero@kent.ac.uk)
[3]Department of Applied Computing, Federal University of Lavras, Lavras 37200-000, Brazil (e-mail: luiz.hcm@ufla.br)

Corresponding author: Helen C. S. C. Lima (e-mail: helen@ufop.edu.br).

This work was supported in part by the Minas Gerais State Foundation for Research Support (FAPEMIG, grants CEX-PPM-00676-17 and CEX-APQ-02266-16), the National Council of Technological and Scientific Development (CNPq, grant 303266/2019-8), and the Coordination for the Improvement of Higher Education Personnel (CAPES, finance code 001).

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. Despite the benefits of feature selection for the classification task, to the best of our knowledge, few studies in the literature address feature selection for the hierarchical classification context. This paper proposes a novel feature selection method based on the general variable neighborhood search metaheuristic, combining a filter and a wrapper step, wherein a global model hierarchical classifier evaluates feature subsets. We used twelve datasets from the proteins and images domains to perform computational experiments to validate the effect of the proposed algorithm on classification performance when using two global hierarchical classifiers proposed in the literature. Statistical tests showed that using our method for feature selection led to predictive performances that were consistently better than or equivalent to that obtained by using all features with the benefit of reducing the number of features needed, which justifies its efficiency for the hierarchical classification scenario.

INDEX TERMS Feature selection, hierarchical single-label classification, variable neighborhood search, filter, wrapper.

I. INTRODUCTION

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

Feature selection aims to identify as many relevant features as possible and decrease the costs 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 execution time of classification are some of the benefits of feature selection [1].

Among data mining tasks, classification has received considerable attention from the scientific community [1]. 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 in which classes that label instances are organized into a hierarchical structure [2] represented by a tree or a directed acyclic graph (DAG), so-called hierarchical classification problems, exist.

Studies have proposed different methods to solve hierarchical classification problems. These methods are categorized as local or global approaches according to how the method handles the class hierarchy. In the local approach, classification is conducted 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 predicting only a single path of labels (single-label problem) or multiple paths of labels (multilabel problem).

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 applying conventional feature selection techniques and constructing classifiers by breaking down the hierarchical classification problem into several flat classification problems. This type of approach allowed researches to use feature selection techniques and classification algorithms traditionally adopted in flat classification [3]–[5].

Few recent approaches that also use a set of flat classifiers have proposed techniques based on recursive regularization that consider the hierarchical information of classes (e.g., parent-child, sibling, and graph relations) [6], [7]. In addition to structure information, another approach used a semantic description of class labels to select different feature subsets for each subclassifier [8]. It is worth mentioning that none of them conducted experiments using global hierarchical classifiers. Other ranked-based methods have proposed readjusting some existing popular filter feature selection algorithms to consider the hierarchical structure of classes [9], [10].

Unlike the previously mentioned studies, we propose a feature selection approach designed specifically for global model hierarchical classifiers that directly address class hierarchy relations. In the literature, several works propose modifications to existing flat classifiers to address the entire class hierarchy in a single step [11]–[18]. Given the relevance of global classifiers to the hierarchical classification scenario, one can see the importance of developing preprocessing techniques capable of handling the class hierarchy as a whole.

This paper presents a hybrid supervised feature selection method, combining filter techniques to form the ranking of features and metaheuristic techniques to search and evaluate feature subsets to construct solutions capable of improving the predictive performance of global hierarchical classifiers. This paper is an extension of a previous work [19] in the following aspects:

- We propose an algorithm that uses a variation of the variable neighborhood search (VNS) [20] metaheuristic, called general variable neighborhood search (GVNS) [21], that applies the basic variable neighborhood descent (B-VND) [22] procedure as a local search method.
- We characterize and compare the running time behavior of the GVNS algorithm to its previous version.
- We add experiments with a wrapper-based feature selection method to compare the effectiveness of the proposed algorithm.
- We include experiments with an additional hierarchical classifier that uses induction of clustering trees for hierarchical multi-label classification (CLUS-HMC).
- Finally, we conduct experiments considering a more extensive dataset collection that covers different domains.

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

- We propose another method that explores and takes advantage of joint a filter-based approach adapted to consider the hierarchical structure of classes and a search-based metaheuristic technique to find the best subset of features.
- We propose an efficient feature selection algorithm for the supervised hierarchical single-label classification task.
- We conduct extensive experiments on twelve real-world hierarchical datasets from protein and image domains to evaluate our approach's efficacy.
- The proposed method is consistently better than or equivalent to our previous algorithm [19].
- When we consider the running time behavior, the proposed method performs better than our previous method [19] since it achieved the improvements first.

The remainder of this work is organized as follows. Section II presents an overview of hierarchical classification and feature selection. In Section III we present the related work, and in Section IV we describe the problem addressed in this work. The proposed algorithm is detailed in Section V. Section VI presents the computational experiments and reports the results of the comparative experiments. Finally, conclusions and directions for future work are described in Section VII.

II. BACKGROUND

Throughout Sections II-A and II-B, we present an overview of hierarchical classification and feature selection methods, respectively.

A. 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 applications, the classes that label instances are organized into a hierarchical structure.

Different aspects can characterize hierarchical classification methods [2]. The first aspect is related to the type of hierarchical structure that the method can process, tree, or directed acyclic graph (DAG). Fig. 1 presents examples of a tree and a DAG, where the nodes represent the classes, and the edges indicate relationship between them. Basically, in a tree structure (Fig. 1a), each node (class) can possess only one parent node, while in a DAG (Fig. 1b), a child node (class) can have multiple parent nodes.

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 nonmandatory leaf node predictions (NMLNP). In MLNP, the most specific class assigned to an instance must be one of the classes at a

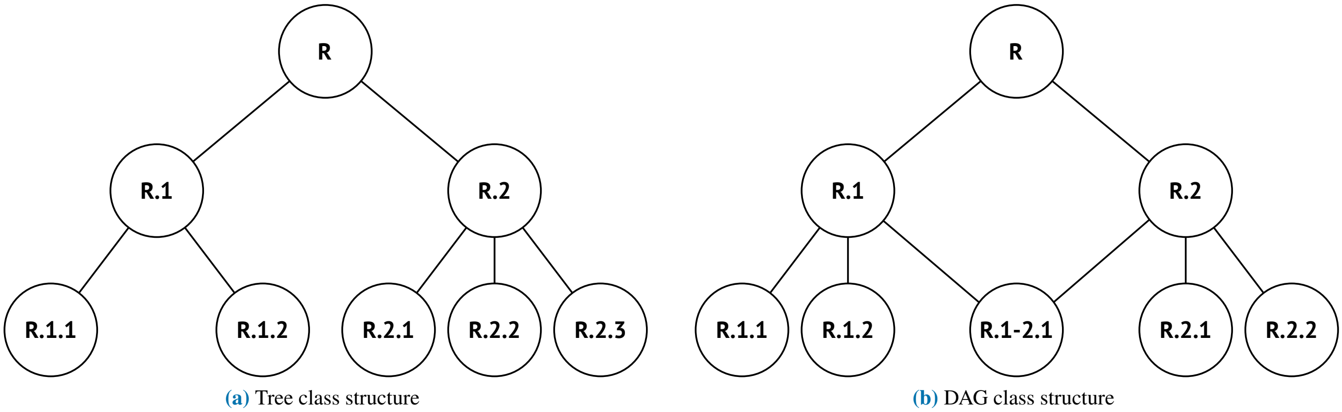


FIGURE 1: Different types of hierarchical structure. In a tree structure (a), nodes have a single parent; in a DAG structure (b), nodes can have multiple parents.

leaf node in the class hierarchy. In contrast, 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 just a single path of labels in the class hierarchy (single-label problem) or be less restricted, predicting multiple paths of labels (multi-label problem), for each instance.

Finally, the fourth aspect concerns how the classification method handles the class hierarchy. Classification methods can perform either flat or hierarchical classification (using a local or global model approach). In flat classification, the methods ignore the class hierarchy and make predictions considering only the classes associated with leaf nodes. In the local model approach, the class hierarchy is explored through a local perspective using a combination of classifiers that consider, in an isolated manner, different parts of the hierarchy. According to Silla Junior and Freitas [2], we can categorize local model approaches according to how they use the local information of the hierarchical structure and how they build their classifiers around it. There are three standard ways of using local information: a local classifier per node, a local classifier per parent node, and a local classifier per hierarchical 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 to existing flat classifiers to address the entire class hierarchy in a single step are available. Some examples of modifications of traditional flat classification algorithms are the following: HC4.5 [11] and HLC [12], modified versions of C4.5; global model naive bayes (GMNB) [13], a modified version of the naive bayes; CLUS-HMC [14], a method based on predictive clustering trees; $h$ant-miner [15] and hmant-miner [16], both adaptations of the ant-miner algorithm; HMC-LMLP [17], a neural network method based on multilayer perceptron; and, more recently, the CSHCIC method [18], which integrates hierarchical classification and cost-sensitive learning to reweight training data for the imbalanced class problem.

B. FEATURE SELECTION IN CLASSIFICATION

Feature selection has received increasing attention from researchers in recent years due to the continued rapid growth in the volume of data. Powerful as a preprocessing step, it selects a subset of predictive features to improve the performance of learning models. Data containing irrelevant or redundant features can reduce the predictive capability and increase the classification processing time of classifiers [23]. 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 a classifier [24]. In practice, the use of feature selection in the classification task can result in the following benefits [25]:

- (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 aspect is related to the use of labels (class value). Feature selection methods can process datasets that have previously labeled, partially labeled, and nonlabeled instances, leading to the development of supervised, semisupervised, 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 that generally can be categorized into embedded, filter, wrapper, or hybrid (involving possible combinations among embedded, filter, and wrapper) methods [26].

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 and are generally faster than wrapper techniques. Nevertheless, the wrapper approach usually has the advantage of achieving higher predictive performance than filters.

When we use an embedded feature selection approach, the classification model performs feature selection simultaneously with its creation. Typical examples of these techniques are decision tree algorithms because they select features placed into the nodes of the generated trees [12], [14], [27].

As outlined above, 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.

Feature ranking-based approaches apply statistical metrics to evaluate each feature individually, rank features according to their relevance, and select 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 [25]. Examples of ranking-based methods are information gain (IG) feature ranking [28], symmetric uncertainty [25], gain ratio [25], and chi-squared [28].

Search-based approaches consider the relationship between features in a feature subset (as a multivariate method) and search 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 eliminating feature redundancy. However, these approaches take more time to generate and measure each feature subset's quality, making them slower than univariate approaches. Recall that if there are n possible features initially, then there are 2^n possible subsets, which makes 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 [25] to search the feature subset space in a reasonable time. Metaheuristic algorithms such as simulated annealing (SA) [29], genetic algorithm (GA) [30], and particle swarm optimization (PSO) [31] 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. [32] for internet text classification. Examples of search-based methods are correlation-based feature selection (CFS) [23], [33], and consistency-based feature selection [34].

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 [35]. The search process proceeds until it finds the subset with the highest evaluation in terms of the classifier's predictive performance.

Methods that follow a wrapper approach generally produce better predictive performance results than those based on a filter approach since the classification algorithm itself drives feature selection. However, in wrapper-based methods, the classifier must be trained and evaluated multiple times during the search process, which could cause very high computational costs, making the method impractical for high-dimensional datasets [36]. Therefore, 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 [37]; BDE-X Rank, an approach that combines a wrapper method based on a binary differential evolution (BDE) algorithm with a ranking-based filter method [38]; MIMAGA, an algorithm that combines the mutual information maximization (MIM) and the adaptive genetic algorithm (AGA) [39], and HI-BQPSO, a method that combines a filter technique with an improved quantum-behavior PSO algorithm [40].

This paper designs a hybrid feature selection method for the hierarchical classification context based on the GVNS [21] metaheuristic. It combines a filter step, wherein a feature ranking is constructed based on the hierarchical symmetrical uncertainty (SU_H) measure [10], with a wrapper step, wherein a global model classifier evaluates feature subsets. We used two classifiers of this type, the GMNB [13] and the CLUS-HMC [14].

III. 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 [3], document classification (whose classes represent a hierarchy of topics) was addressed through the local model classification approach combined 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 [41]. 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. [4] 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 the dataset dimensionality of that particular node. The proposed feature selection method uses the CFS and the best first algorithm – both available in the WEKA data mining toolkit [42], [43]. They conducted experiments to determine 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 the predictive accuracy.

Paes et al. [5] explored the use of feature selection techniques to improve the predictive performance of two different hierarchical classification approaches, the local per parent node and local per level approaches. They proposed a method that produces a ranking of the features using the IG measure [44]. After forming the ranking, the p best features are selected, where p is 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 feature selection techniques and classifier construction were performed by decomposing the hierarchical classification problem into several flat ones, which allowed the researchers 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 consider the hierarchical structure of classes to select different feature subsets for each subclassifier [6]–[8].

Zhao et al. [6] 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 in both classification accuracy and computational efficiency compared with flat feature selection approaches.

Similarly, Tuo et al. [7] 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 [8] proposed the most recent study that uses recursive regularization. This is the first attempt to explore a method 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 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 explore the semantic similarities of labels and use them to guide feature selection. They also used parent-child and sibling relations as structural regularization. Finally, they built a supervised learning model and imposed semantic and structural regularization terms on each subclassifier. 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, directly addressing 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 [9], [10]. The work of Slavkov et al. [9] proposes a feature selection technique capable of handling the hierarchy of classes as a whole without the decomposition of the hierarchical problem in several flat classification problems for hierarchical multilabel classification problems. They developed an adaptation of the ReliefF [45] algorithm to the hierarchical multilabel context, called HMC-ReliefF. They employed forward feature addition (FFA) curves, a stepwise filter-like procedure to construct classifiers for different numbers of top-k ranked features, to evaluate their method. 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. Our approach is different because we address the hierarchical single-label classification scenario.

Concerning hierarchical single-label classification, Dias and Merschmann [10] proposed an adaptation of the symmetrical uncertainty (SU) filter measure to consider the hierarchical structure of classes. Comparative analysis between the ranking generated from the SU_H and another ranking randomly generated was performed. In the random ranking, the most relevant features were 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. We use the SU_H filter measure to construct rankings and combine it with a wrapper step in our approach.

This work is an extension of a previous study [19] in which we proposed a hybrid algorithm based on the VNS metaheuristic, named VNS-FSHC, using the SU_H measure in a filter step and the GMNB as the classifier of a wrapper step. The present work builds on this preliminary effort by providing a more efficient framework based on the VNS metaheuristic. Furthermore, we include experiments using two hierarchical classifiers (GMNB and CLUS-HMC) and consider a more extensive dataset collection covering different domains. We also add a wrapper hierarchical feature selection method to compare the effectiveness of our approach.

It is worth mentioning that Cerri et al. [46] proposed using the CLUS-HMC decision tree induction classifier as a feature selector and checked if the features selected to construct its tree were sufficiently good to be used as input for two hierarchical multilabel 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 multilabel methods, showing the need to develop feature selection methods specifically to consider hierarchical class relationships.

IV. PROBLEM STATEMENT

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, where $1 \leq i \leq n$, 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), where $Y^j = \{y_1^j, y_2^j, \ldots, y_n^j\}$ is a list of feature values with $y_i^j \in A_i$, and $c^j \in C$ is the class of 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 this, let D be a set of academic papers, A = {*word count*, *character count*, *verb count*, *noun count*} be the feature set and C = {*computer science*, *software engineering*, *artificial intelligence*} be 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 a solution to this problem.

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

V. PROPOSAL

Next, we discuss our proposed hybrid feature selection method, which uses the GVNS metaheuristic to solve the FSHC problem. The representation of a solution and its evaluation are presented in Section V-A. Sections V-B and V-C describe how to build an initial solution and how to apply the neighborhood structures to explore the solution space of the problem, respectively. Finally, Section V-D provides a detailed description of the proposed algorithm, general variable neighborhood search for feature selection in hierarchical classification (GVNS-FSHC).

A. SOLUTION REPRESENTATION AND EVALUATION

Our method's first step is to 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 and the hierarchical F-measure (hF) [47] 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, used to consider the class hierarchy.

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

$$hF(X) = \frac{2 \times hP(X) \times hR(X)}{hP(X) + hR(X)} \tag{1}$$

where $hP(X)$ and $hR(X)$ represent 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, $hP(X)$ and $hR(X)$ of solution X can be defined according to (2) and (3):

$$hP(X) = \frac{\sum_j |P_j \cap T_j|}{\sum_j |P_j|} \tag{2}$$

$$hR(X) = \frac{\sum_j |P_j \cap T_j|}{\sum_j |T_j|}. \tag{3}$$

B. BUILDING AN INITIAL SOLUTION

As in Costa et al. [19], we used the incremental wrapper subset selection (IWSS) approach [48] to generate the initial solution. IWSS has two steps:

(i) *Filter*: A filter-based measure evaluates each predictive feature independently regarding the dataset classes to create a ranking R. We used the SU_H measure [10] to consider the hierarchical context. Then, the ranking R of all features is constructed using the roulette wheel method as in the survival selection phase in GA [49]. 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 according to 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 set initial solution X starts with the best-rated feature in the ranking R. Then, we try to insert the next feature $A_i \in R$ into X 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; otherwise, it is discarded.

We used the same 5-fold cross-validation method in all wrapper evaluations to ensure fair comparisons. Additionally, we complement the IWSS method by adding a step that verifies the 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 $X = \{A_1, A_2\}$ and $hF(X) = 0.70$. We inserted A_3 in X, but it did not improve the classifier's performance. Therefore, we generated the temporary subsets $Y = \{A_3, A_2\}$ and $Z = \{A_1, A_3\}$, where $hF(Y) = 0.75$ and $hF(Z) = 0.60$. As $hF(Y)$ is greater than $hf(X)$, Y is maintained for the next iteration, which would seek to include A_4 in Y. This procedure aims to revoke some previous decisions by identifying selected features that may become ineffective after the insertion of another feature. Thus, this step follows the well-known proximate optimality principle (POP) [50].

C. NEIGHBORHOOD STRUCTURES

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

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

(ii) *Neighborhood structure* N_2: It consists of 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 of swapping a feature $X_j \in X$ with a feature $A_i \in (A \setminus X)$.

For the example described in Section IV, given the solution $X = \{$*word count*, *verb count*, *noun count*$\}$, a swap movement consists of swapping 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.

D. GVNS APPROACH TO SOLVE FSHC

This section presents the GVNS-FSHC algorithm, an adaptation of the GVNS metaheuristic [22] to solve 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 [21]. GVNS differs from VNS in the local search method. While the local search is conventional in VNS, in GVNS, the local search is performed by the variable neighborhood descent (VND) [20] method.

In our GVNS-FSHC algorithm, we apply the basic sequential VND, named B-VND in Hansen et al. [21]. Algorithm 1 presents the pseudocode of the proposed GVNS-FSHC.

In Algorithm 1, 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 V-C. 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 of perturbations k in the Shake function. In a classical GVNS algorithm, the level of perturbations is increased whenever there is no improvement in the solution. Instead, in our algorithm, we only increase the level of perturbations after performing some local search attempts without improving the current solution. This strategy follows the ideas introduced by Reinsma et al. [51] and used successfully in Santos et al. [52].

$RDrate$ is a percentage rate used in the B-VND improvement procedure, described in Section V-D2. 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 $\mathrm{Relevance}$ function is described in Section V-D1.

The algorithm generates an initial solution X (line 3) by applying the IWSS approach described in Section V-B. 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 of perturbations k without improvement in the current solution X, is started.

Algorithm 1 GVNS-FSHC algorithm

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

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 whether 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 occurred, the variable k is increased by 1 and $attempt$ is restarted. GVNS-FSHC ends when the given total running time $t_{\max}$ expires.

1) Relevance function

Algorithm 2 outlines the pseudocode of the Relevance function. It starts by measuring the average hF performance achieved using the 5-fold cross-validation procedure for each solution (lines 3 and 4). The solution X'' is considered better than X if the average $hF(X'')$ is larger than $hF(X)$ (line 6) and if w-fold measures of the five $X''.\overrightarrow{hF}$ are greater than or equal to the corresponding measure of $X.\overrightarrow{hF}$ (line 12). Thus, if both conditions are true, X'' is considered better than X.

2) Variable neighborhood descent

B-VND [21] is the local search used in the GVNS-FSHC algorithm (line 10 of Algorithm 1). 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 Hansen et al. [21].

Algorithm 2 Relevance function

1: *in:* X'', X, w
2: *out:* boolean
3: $X''.hF = (\sum_{i=1}^{5} X''.\overrightarrow{hF}[i])/5$;
4: $X.hF = (\sum_{i=1}^{5} X.\overrightarrow{hF}[i])/5$;
5: $counter = 0$;
6: **if** $X''.hF > X.hF$ **then**
7: **for** $i = 1$ **to** 5 **do**
8: **if** $X''.\overrightarrow{hF}[i] > X.\overrightarrow{hF}[i]$ **then**
9: $counter = counter + 1$;
10: **end if**
11: **end for**
12: **if** $counter \geq w$ **then**
13: **return** $TRUE$;
14: **end if**
15: **end if**
16: **return** $FALSE$;

Algorithm 3 B-VND algorithm

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

In Algorithm 3, 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 1.

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 (Section IV), 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 there are $RDmax$ iterations without improvement in the same neighborhood (line 8). Then, if the improved solution X' is better than X, then X' becomes the new current solution, and the random descent 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 features from a candidate solution. To do this, we perform the roulette wheel method, as used in the survival selection phase in GAs. 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.

VI. EXPERIMENTAL RESULTS

The GVNS-FSHC algorithm presented in Section V-D was implemented in C++ using the compiler g++ version 4.8.5 for its execution. The experiments were performed on a computer with an Intel Xeon(R) CPU E5620 @ 2.40 GHz $\times$ 16, 48 GB of RAM, and a CentOS Linux 7 operating system. Although this computer processor has more than one core, the algorithm was not optimized for multicore-processing.

GVNS-FSHC is a preprocessing step designed specifically for global hierarchical classifiers. In this sense, computational experiments evaluate the efficacy of the proposed algorithm for feature selection in the hierarchical single-label classification context. We used the GMNB and CLUS-HMC hierarchical classifiers to evaluate the quality of the selected features. It is worth mentioning that the CLUS-HMC handles hierarchical multilabel 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 multilabel classification in which the number of labels is equal to one.

Based on the evaluation metrics, the hierarchical precision and hierarchical recall, described in Section V-A, we compared the proposed GVNS-FSHC algorithm to 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) **VNS-FSHC**: A previous version of this approach so-called variable neighborhood search for feature selection in hierarchical classification (VNS-FSHC) [19].

(iii) **BF**: We implemented a bottom-up wrapper-based approach of the best first algorithm, a well-known heuristic search method [25]. 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 and measures 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 steps to a candidate solution in the OPEN list without improvements as the stopping criterion of the algorithm.

Section VI-A presents the dataset description and preprocessing steps. Section VI-B presents the parameter configuration. Section VI-C details the computational results of the proposed method using the GMNB and CLUS-HMC classifiers.

A. DATASET DESCRIPTION

The experiments use twelve public benchmark datasets with classes hierarchically organized in a tree structure, covering two domains, proteins and images. The protein domain is represented by bioinformatic datasets[1] referring to the yeast genome [11].

The image datasets[2] were selected from the ImageCLEF 2007 competition for annotating medical X-ray images. ImageCLEF aims to provide an evaluation forum for the cross-language annotation of the medical radiological images [53].

These datasets were initially available as multilabel data. Since our method focuses on addressing the single-label scenario, we perform a preprocessing step to convert the datasets into single-label data. Table 1 shows the general characteristics of the datasets. For each dataset, the second column corresponds to the dataset domain, and 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.

Data preprocessing was conducted in four steps. In the first step, we selected the most frequent class considering the leaf nodes in the original dataset for each instance. In the second step, each missing value was replaced using the hierarchical supervised imputation method (HSIM) [54]. In

[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 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 equal frequency binning [55] method with 20 partitions to convert all continuous features into discrete values.

B. PARAMETER SETTINGS

Our experiments and comparisons use 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.

The parameters of the VNS-FSHC are those used by Costa et al. [19], which were fixed at the following values: $\mathrm{VNSmax} = 0.1 \times$ (number of features included in the initial solution) $\times$ (number of features excluded from the same solution), and $\mathrm{RDmax} = 0.1 \times$ (number of features included in the current solution passed to the $\mathrm{RandomDescent}$ method) $\times$ (number of features excluded from the same solution).

Regarding the BF algorithm, we performed 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 [56], an automatic algorithm configuration method. Table 2 shows the tuning setup, and we applied the 5-fold cross-validation procedure to the training set of the SPO dataset. 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 costs.

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

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)

C. COMPUTATIONAL RESULTS

Subsection VI-C1 presents the computational results using the GMNB classifier, and Subsection VI-C2 shows the CLUS-HMC results.

1) GVNS-FSHC results with the GMNB classifier

Considering the stochastic nature of the VNS-based algorithms, each algorithm 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 generating random numbers of those metaheuristic algorithms. As the BF heuristic is deterministic, it 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 whether they were well modeled by a normal distribution by applying the Shapiro-Wilk test [57]. If samples came from populations with normal distributions, we applied the ANOVA test [58], a parametric hypothesis test for two independent samples; otherwise, we applied the Kruskal-Wallis test [59], a nonparametric analysis of variance that can compare several independent samples.

Table 4 shows the hF measure results obtained by using all features, the GVNS-FSHC algorithm, and the two comparison algorithms. The second to fifth column represent the hF values achieved by the GMNB classifier using all the dataset features (second column) and feature selection methods (other columns). In these columns, "*avg*" indicates the average result, and the standard deviation (*sd*) is 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 datasets (SPO, Gasch2,

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

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, 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 averages (*avg*) and standard deviations (*sd*) of the number of features used by the algorithms.

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

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

Aiming to characterize and compare the running time behavior of the GVNS-FSHC algorithm to its previous version (the VNS-FSHC algorithm), we used the multiple time-to-target plot (mttt-plot) tool [60]. The mttt-plot is an extension of the time-to-target plot [61] to sets of multiple instances.

Runtime distributions (ttt-plots) display the probability that an algorithm will find a solution at least as good as a given target value for a given problem instance, on the ordinate axis, within a given running time, shown on the abscissa axis [60]. 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 an 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 $look4_j$, such as $\mathcal{I}_j$; 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 α_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 [60].

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

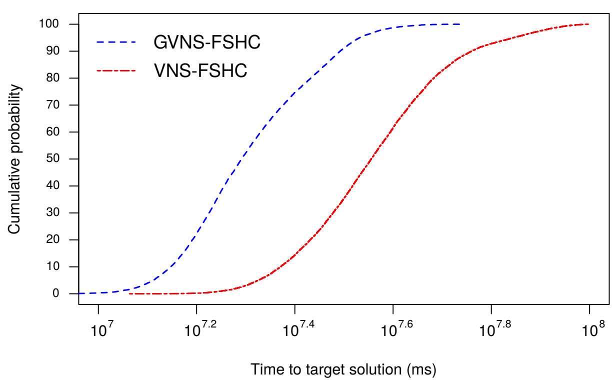
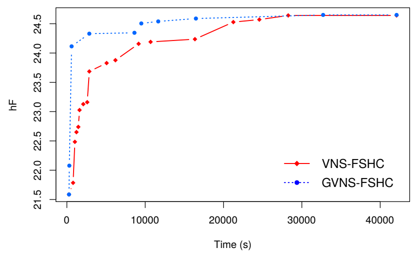
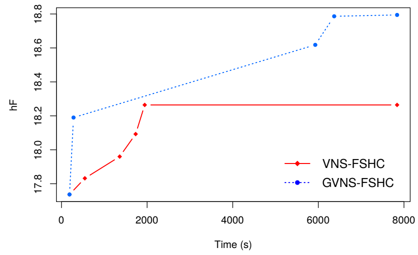


FIGURE 2: Combined mttt-plot for the VNS-FSHC × the GVNS-FSHC

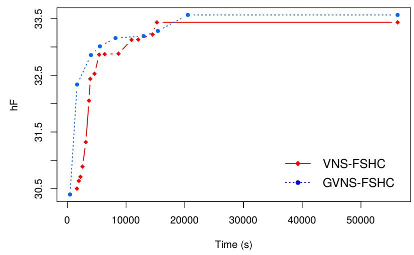
Fig. 3 shows the evolution of the objective function (hF measure) over time considering the pair (partition, seed) that generated the best result for the GVNS-FSHC in each dataset. The figure shows that, on most datasets, the GVNS-FSHC algorithm achieves improvements before the VNS-FSHC algorithm.



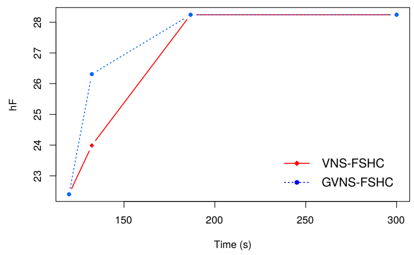
(a) Gasch2



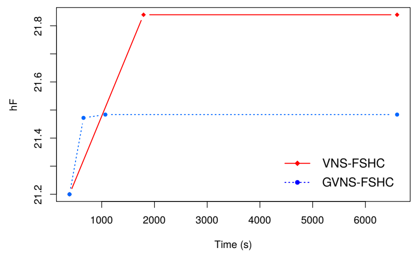
(b) Pheno



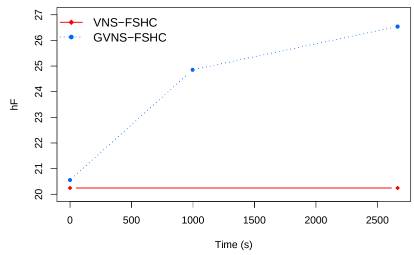
(c) CellCycle



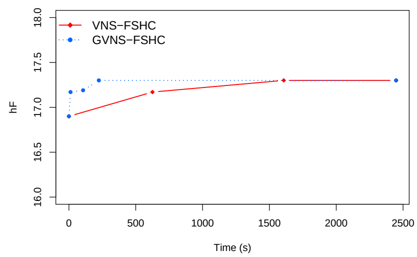
(d) Church



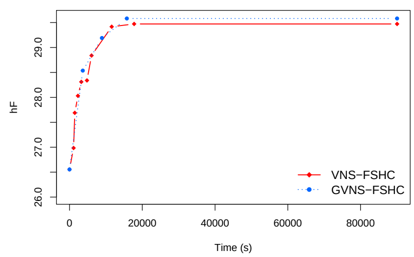
(e) SPO



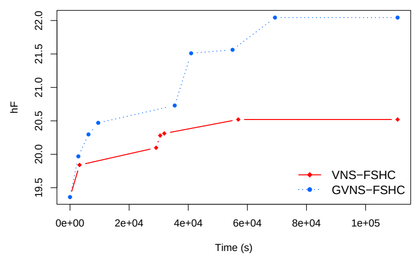
(f) Eisen



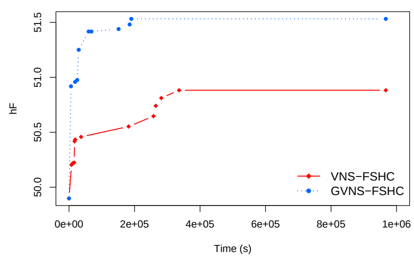
(g) Derisi



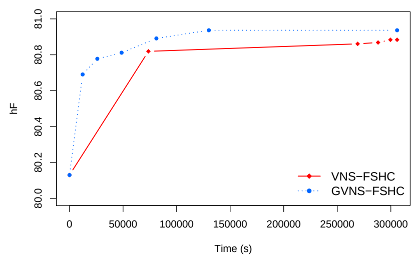
(h) Gasch1



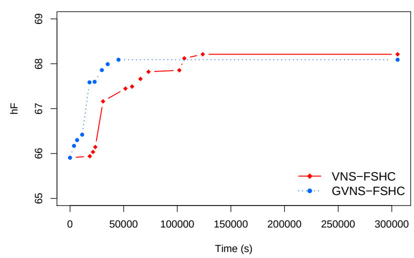
(i) Sequence



(j) Expression



(k) ImageCLEF07A



(l) ImageCLEF07D

FIGURE 3: 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 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)

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

2) GVNS-FSHC 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 [14] 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 embeds feature selection 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 as Table 4. The results show that the feature selection step using the GVNS-FSHC algorithm did not statistically significantly improve the performance of the CLUS-HMC classifier (except for the Church dataset), confirming the power of decision trees as natural feature selectors. However, the GVNS-FSHC algorithm did not statistically significantly jeopardize the performance of the CLUS-HMC classifier.

Considering the number of features used by the CLUS-HMC classifier with and without the feature selection step, Table 7 shows 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.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel feature selection method tailored for global model hierarchical classifiers. We developed a hybrid filter-wrapper approach based on the VNS metaheuristic, the so-called GVNS-FSHC, which uses 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 feature selection strategies on twelve datasets (from proteins and images contexts).

The experimental results showed that the method using the GVNS-FSHC algorithm with the GMNB classifier achieved predictive performance that was consistently better than 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 predictive performance of the GVNS-FSHC is better than or equivalent to the VNS-FSHC algorithm. Moreover, when we considered the running time behavior, the GVNS-FSHC performed better than the VNS-FSHC since it achieved the improvements first.

Concerning 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, the GVNS-FSHC was able to select fewer features with no statistically significant difference in the performance results.

We intend to investigate and develop subset filter-based measures adapted to treat the class hierarchy in future work. The goal is to incorporate the measures in a hybrid approach that runs the classifier less often in the wrapper phase of the feature selection to reduce its computational costs.

ACKNOWLEDGMENT

The authors would like to thank the Federal University of Ouro Preto (UFOP), the Federal University of Lavras (UFLA), and the University of Kent for supporting the development of the present study.

REFERENCES

[1] Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[2] C. N. Silla Junior and A. A. Freitas. A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery, 22(1-2):31–72, 2011.

[3] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In 14th International Conference on Machine Learning, pages 170–178, San Francisco, 1997. Morgan Kaufmann Publishers Inc.

[4] Andrew Secker, Matthew N. Davies, Alex Alves Freitas, Edward Clark, Jonathan Timmis, and Darren R. Flower. Hierarchical classification of G-Protein-Coupled Receptors with data-driven selection of attributes and classifiers. International Journal of Data Mining and Bioinformatics, 4:191–210, 2010.

[5] Bruno C. Paes, Alexandre Plastino, and Alex Alves Freitas. Exploring attribute selection in hierarchical classification. Journal of Information and Data Management, 5(1):124–133, 2014.

[6] Hong Zhao, Pengfei Zhu, Ping Wang, and Qinghua Hu. Hierarchical feature selection with recursive regularization. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, pages 3483–3489, Melbourne, 2017. AAAI Press.

[7] Qianjuan Tuo, Hong Zhao, and Qinghua Hu. Hierarchical feature selection with subtree based graph regularization. Knowledge-Based Systems, 163:996–1008, 2019.

[8] Hai Huang and Huan Liu. Feature selection for hierarchical classification via joint semantic and structural information of labels. Knowledge-Based Systems, 195:105655, 2020.

[9] Ivica Slavkov, Jana Karcheska, Dragi Kocev, Slobodan Kalajdziski, and Saso Dzeroski. Relieff for hierarchical multi-label classification. In New Frontiers in Mining Complex Patterns, Lecture Notes in Computer Science, pages 148–161. Springer International Publishing, 2014.

[10] T. N. Dias and L. H. C. Merschmann. 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, pages 142–149, Natal, 2015.

[11] Amanda Clare and Ross D King. Predicting gene function in saccharomyces cerevisiae. Bioinformatics, 19(suppl 2):ii42–ii49, 2003.

[12] Yen-Liang Chen, Hsiao-Wei Hu, and Kwei Tang. Constructing a decision tree from data with hierarchical class labels. Expert Systems with Applications, 36(3, Part 1):4838 – 4847, 2009.

[13] C. N. Silla Junior and A. A. Freitas. 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), pages 182–196, Miami Beach, December 2009. IEEE Press.

[14] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. Machine Learning, 73(2):185, 2008.

[15] Fernando E. B. Otero, Alex A. Freitas, and Colin G. Johnson. A Hierarchical Classification Ant Colony Algorithm for Predicting Gene Ontology Terms. Springer Berlin Heidelberg, Berlin, 2009.

[16] Fernando E. B. Otero, Alex A. Freitas, and Colin G. Johnson. A hierarchical multi-label classification ant colony algorithm for protein function prediction. Memetic Computing, 2(3):165–181, 2010.

[17] R. Cerri, R. C. Barros, and A. C. P. L. F. de Carvalho. Hierarchical classification of gene ontology-based protein functions with neural networks. In 2015 International Joint Conference on Neural Networks (IJCNN), pages 1–8, Killarney, 2015.

[18] Weijie Zheng and Hong Zhao. Cost-sensitive hierarchical classification for imbalance classes. Applied Intelligence, 50:2328–2338, 2020.

[19] H. Costa, L. R. Galvão, L. H. C. Merschmann, and M. J. F. Souza. A vns algorithm for feature selection in hierarchical classification context. Electronic Notes in Discrete Mathematics, 66:79–86, 2018. 5th International Conference on Variable Neighborhood Search.

[20] N. Mladenović and P. Hansen. Variable neighborhood search. Computers & Operations Research, 24(11):1097–1100, November 1997.

[21] Pierre Hansen, Nenad Mladenović, Raca Todosijević, and Saìd Hanafi. Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization, 5:423–454, 2017.

[22] Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A. Moreno Pérez. Variable neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, Handbook of Metaheuristics, pages 57–97. Springer International Publishing, Cham, 2019.

[23] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In Proceedings of the Seventeenth International Conference on Machine Learning, pages 359–366, San Francisco, 2000. Morgan Kaufmann Publishers Inc.

[24] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. Artif. Intell., 97(1-2):245–271, December 1997.

[25] Huan Liu and Hiroshi Motoda. Computational Methods of Feature Selection. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC, 2007.

[26] Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao. Feature selection: An ever evolving frontier in data mining. In Proceedings of the Fourth International Workshop on Feature Selection in Data Mining, volume 10 of Proceedings of Machine Learning Research, pages 4–13, Hyderabad, 21 Jun 2010. PMLR.

[27] Eduardo P. Costa, Ana C. Lorena, André C. P. L. F. Carvalho, Alex A. Freitas, and Nicholas Holden. Comparing several approaches for hierarchical classification of proteins with decision trees. In Advances in Bioinformatics and Computational Biology, pages 126–137, Berlin, 2007. Springer Berlin Heidelberg.

[28] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Proceedings of the Fourteenth International Conference on Machine Learning, pages 412–420, San Francisco, 1997. Morgan Kaufmann Publishers Inc.

[29] Justin C W Debuse and Victor J Rayward-Smith. Feature subset selection within a simulated annealing data mining algorithm. Journal of Intelligent Information Systems, 9:57–81, 1997.

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

[31] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed. Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). IEEE Access, 9:26766–26791, 2021.

[32] Changqin Huang, Jia Zhu, Yuzhi Liang, Min Yang, Gabriel Pui Cheong Fung, and Junyu Luo. An efficient automatic multiple objectives optimization feature selection strategy for internet text classification. International Journal of Machine Learning and Cybernetics, 2019.

[33] Suwimol Jungjit and Alex Freitas. 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, pages 989–996, Madrid, 2015. Association for Computing Machinery.

[34] Huan Liu and Rudy Setiono. A probabilistic approach to feature selection - a filter solution. In Proceedings of the Thirteenth International Conference on Machine Learning, pages 319–327, San Francisco, 1996. Morgan Kaufmann.

[35] Ron Kohavi and George H. John. Wrappers for feature subset selection. Artificial Intelligence, 97(1-2):273–324, December 1997.

[36] Pablo Bermejo, José A. Gámez, and José M. Puerta. A grasp algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets. Pattern Recognition Letters, 32(5):701 – 711, 2011.

[37] Xian-Fang Song, Yong Zhang, Dun-Wei Gong, and Xiao-Zhi Gao. A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data. IEEE Transactions on Cybernetics, pages 1–14, 2021.

[38] Javier Apolloni, Guillermo Leguizamón, and Enrique Alba. Two hybrid wrapper-filter feature selection algorithms applied to high-dimensional microarray experiments. Appl. Soft Comput., 38(C):922–932, 2016.

[39] Huijuan Lu, Junying Chen, Ke Yan, Qun Jin, Yu Xue, and Zhigang Gao. A hybrid feature selection algorithm for gene expression data classification. Neurocomputing, 256(C):56–62, 2017.

[40] Qing Wu, Zheping Ma, Jin Fan, Gang Xu, and Yuanfeng Shen. A feature selection method based on hybrid improved binary quantum particle swarm optimization. IEEE Access, 7:80588–80601, 2019.

[41] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In 13th International Conference on Machine Learning, pages 284–292, San Francisco, 1995. Morgan Kaufmann Publishers Inc.

[42] S. Garner. Weka: The waikato environment for knowledge analysis. In Proceedings of the New Zealand Computer Science Research Students Conference, pages 57–64, 1995. Available at https://www.cs.waikato.ac.nz/~ml/publications/1995/Garner95-WEKA.pdf.

[43] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. ACM SIGKDD Explorations Newsletter, 11(1):10–18, 2009.

[44] Thomas M. Cover and Joy A. Thomas. Elements of Information Theory. Wiley-Interscience, New York, NY, USA, 1991.

[45] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. Machine Learning, 53(1-2):23–69, 2003.

[46] Ricardo Cerri, Rafael G Mantovani, Márcio P. Basgalupp, and André C. P. L. F. de Carvalho. Multi-label feature selection techniques for hierarchical multi-label protein function prediction. In International Joint Conference on Neural Networks (IJCNN), pages 1–7, Rio de Janeiro, 2018.

[47] Svetlana Kiritchenko, Stan Matwin, and A. Fazel Famili. 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, 2005.

[48] Roberto Ruiz, José C. Riquelme, and Jesús S. Aguilar-Ruiz. Incremental wrapper-based gene selection from microarray data for cancer classification. Pattern Recognition, 36(12):2383–2392, 2006.

[49] David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[50] F. Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers, Norwell, Massachusetts, 1997.

[51] Jordi Alves Reinsma, Puca Huachi Vaz Penna, and Marcone Jamilson Freitas Souza. 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, 2018. SOBRAPO.

[52] Mario S. Santos, Thomas V. B. Pinto, Enio Lopes Júnior, Luciano P. Cota, Marcone J. F. Souza, and Thiago A. M. Euzébio. Simheuristic-based

decision support system for efficiency improvement of an iron ore crusher circuit. Engineering Applications of Artificial Intelligence, 94:103789, 2020.

[53] Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Deroski. Hierarchical annotation of medical images. Pattern Recogn., 44(10–11):2436–2449, 2011.

[54] Leandro Galvão and Luiz H. C. Merschmann. HSIM: A supervised imputation method for hierarchical classification scenario. In Proceedings of the 19th International Conference on Discovery Science, pages 134–148, Bari, 2016.

[55] Ying Yang and Geoffrey I. Webb. Proportional k-interval discretization for naive-bayes classifiers. In 12th European Conference on Machine Learning, volume 2167 of LNCS, pages 564–575, Berlin, 2001. Springer.

[56] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3:43–58, 2016.

[57] J. P. Royston. An extension of shapiro and wilk's w test for normality to large samples. Journal of the Royal Statistical Society. Series C (Applied Statistics), 31(2):115–124, 1982.

[58] Richard M. Heiberger. Computation for the analysis of designed experiments. Wiley series in probability and statistics. John Wiley & Sons, New York, NY, USA, 2015.

[59] Myles Hollander and Douglas A. Wolfe. Nonparametric Statistical Methods. John Wiley & Sons, New York, 1973.

[60] Alberto Reyes and Celso C. Ribeiro. Extending time-to-target plots to multiple instances. International Transactions in Operational Research, 25(5):1515–1536, 2018.

[61] Thomas A. Feo, Mauricio G. C. Resende, and Stuart H. Smith. A greedy randomized adaptive search procedure for maximum independent set. Operations Research, 42(5):860–878, 1994.

HELEN C. S. C. LIMA received her B.Sc. degree in computer science from São Paulo State University (UNESP), Campus Rio Claro, Brazil in 2010 and an M.Sc. degree in computer science from Federal University of Ouro Preto (UFOP), Brazil in 2013. She is currently pursuing a Ph.D. degree in computer science in the Department of Computing, Federal University of Ouro Preto (UFOP), Brazil. She was a student visitor at the University of Kent, Canterbury, UK in 2019. Currently, she is an assistant professor in the Department of Computing and Systems, UFOP. Her main research interest is machine learning and optimization algorithms, focusing on feature selection for classification techniques using metaheuristics. Her research areas also include data mining, natural language processing, and social media analysis. She has published over ten peer-reviewed papers in journals and conferences.

FERNANDO E. B. OTERO received his B.Sc. in computer science from PUC-PR, Curitiba, Brazil in 2002 and his Ph.D. in computer science (in data mining and ant colony optimization algorithms) from the University of Kent, Canterbury, UK in 2010. He is currently a senior lecturer in computational intelligence and deputy head of the School of Computing, University of Kent. He also serves as an associate editor for the Artificial Intelligence Review journal, Springer. His research interests include data mining, focusing on interpretability and fairness; and biologically inspired computational intelligence. He has published over 50 peer-reviewed papers in journals and conferences. He received peer recognition for his work published in conferences, winning a best conference track paper award at both the 2013 and 2016 editions of the ACM International Genetic and Evolutionary Computation Conference.

LUIZ H. C. MERSCHMANN received a B.Sc. degree in mining engineering from the Federal University of Ouro Preto (UFOP), Brazil in 1999; an M.Sc. degree in production engineering from the Federal University of Rio de Janeiro (UFRJ), Brazil in 2002; and a Ph.D. degree in computer science from the Fluminense Federal University (UFF), Brazil in 2007. In 2012, he spent one year as a postdoctoral researcher at the University of Kent, UK. Currently, he is a professor in the Department of Applied Computing, Federal University of Lavras, Brazil. His research interests include data mining, machine learning, and natural language processing. He has authored/coauthored over 50 scientific publications.

MARCONE J. F. SOUZA received his B.Sc. degree in metallurgical engineering from Federal University of Ouro Preto (UFOP), Brazil in 1982; and his M.Sc. and Ph.D. degrees in systems engineering and computing from the Federal University of Rio de Janeiro, Brazil in 1989 and 2000, respectively. He also performed a postdoctoral internship at the Institute of Computing, Fluminense Federal University, Brazil in 2008. Currently, he is a full professor at the Department of Computing, UFOP. His research interests include metaheuristics, scheduling, timetabling, open-pit mining, vehicle routing, public transport, machine learning applications, and operations research in the health area. He has authored/coauthored 72 full papers in journals, 18 book chapters, and 228 full papers in conferences. Prof. Souza's awards and honors include a research productivity fellowship granted by the Brazilian Council for Scientific and Technological Development (CNPq) in transport and production engineering, the winner's award in the competition on solution methods for the biobjective traveling thief problem (10th International Conference on Evolutionary Multi-Criterion Optimization, EMO 2019), the winner's prize in the International Timetabling Competition 2011-2012 (9th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2012), and best paper award certificates in the area of artificial intelligence and decision support systems (16th and 21st International Conference on Enterprise Information Systems, ICEIS 2014 and 2019).