Towards a Method for Automatically Selecting and Configuring Multi-Label Classification Algorithms

Alex G. C. de Sá
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brazil
alexgcsa@dcc.ufmg.br

Gisele L. Pappa
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brazil
glpappa@dcc.ufmg.br

Alex A. Freitas
School of Computing
University of Kent
Canterbury, Kent, United Kingdom
a.a.freitas@kent.ac.uk

ABSTRACT

Given a new dataset for classification in Machine Learning (ML), finding the best classification algorithm and the best configuration of its (hyper)-parameters for that particular dataset is an open issue. The Automatic ML (Auto-ML) area has emerged to solve this task. With this issue in mind, in this work we are interested in a specific type of classification problem, called multi-label classification (MLC). In MLC, each example in the dataset can be associated to one or more class labels, making the task considerably harder than traditional, single-label classification. In addition, the cost of learning raises due to the higher complexity of the data. Although the literature has proposed some methods to solve the Auto-ML task, those methods address only the traditional, single-label classification problem. By contrast, this work proposes the first method (an evolutionary algorithm) for solving the Auto-ML task in MLC, i.e., the first method for automatically selecting and configuring the best MLC algorithm for a given input dataset. The proposed evolutionary algorithm is evaluated on three MLC datasets, and compared against two baseline methods according to four different multi-label predictive accuracy measures. The results show that the proposed evolutionary algorithm is competitive against the baselines, but there is still room for improvement.

CCS CONCEPTS

•**Computing methodologies** →**Machine learning;** *Machine learning approaches;*

KEYWORDS

Multi-label, Automatic Machine Learning, Evolutionary Algorithms.

ACM Reference format:
Alex G. C. de Sá, Gisele L. Pappa, and Alex A. Freitas. 2017. Towards a Method for Automatically Selecting and Configuring Multi-Label Classification Algorithms. In *Proceedings of the 19th Annual Conference Companion on Genetic and Evolutionary Computation, Berlin, Germany, 2017 (GECCO'17 Companion),* 8 pages.
DOI: http://dx.doi.org/10.1145/3067695.3082053

GECCO'17 Companion, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3067695.3082053

1 INTRODUCTION

Classification is one of the most important tasks in Machine Learning (ML) [2]. In a traditional single-label classification problem, the goal is to find/learn a model that expresses the relationships between a set of predictive attributes (features describing the example) and a predefined set of class labels. Each class label is represented by a discrete value. In a single-label classification problem, each example of the dataset is associated to a single label. In this work, we study a more general type of classification, namely multi-label classification [32]. In this case, each example in the dataset is associated to one or more class labels. Recently, this type of classification problem has attracted significant attention given the increasing number of applications from different sources [32, 41], such as bioinformatics, music categorization into emotions, semantic annotation of media and text mining.

It is important to mention that learning a suitable model in multi-label classification is more challenging than in a single-label problem. This can be justified by the fact that the learning algorithm should consider, in general, label correlations – detecting whether or not they exist – before building the classification model [41]. Additionally, the limited number of examples in the dataset for each class label may make it harder to generalize as the learning algorithm would need more examples to create a good model from such complex data [7].

Nevertheless, we also have in ML the fundamental issue of choosing appropriate algorithms and their parameters in order to properly solve a learning problem (a classification problem, in our case). Researchers and practitioners usually follow an ad-hoc approach, either in single-label or multi-label classification. For instance, when an untrained user or an expert decides to solve a classification problem associated to a dataset, this person will have to evaluate whether it is valid to apply any set of preprocessing steps and choose the classification algorithm(s) that will be executed. In the majority of cases, these decisions are based on trial and error when testing different methods from the literature or on the recommendation of other experienced data scientists.

This scenario makes ML solutions biased and inefficient, because the selected algorithm to solve the problem at hand is not usually the best. In order to solve this issue, the research area of Automatic Machine Learning (Auto-ML) emerged [9, 18, 27, 30]. The main idea of Auto-ML is to create personalized solutions to ML problems, aiming at recommending the best algorithm customized for each dataset provided by a user, with minimum human intervention.

Although there are a few solutions available in the literature to automatically generate complete ML solutions for classification, such as Auto-WEKA [30], Auto-SKLearn [9], TPOT (Tree-based

Pipeline Optimization Tool) [18] and RECIPE (REsilient ClassifIcation Pipeline Evolution) [27], none of them is able to provide solutions to multi-label classification problems. This fact can be justified by the higher difficulty to learn over multi-label data and by the computational cost involved. These particularities make the Auto-ML task for multi-label classification more challenging than the Auto-ML task for single-label classification.

Given the lack of Auto-ML techniques for multi-label data, the main objective of this work is to propose an automatic method for selecting and configuring multi-label classification (MLC) algorithms. Thus, we aim to generate and recommend the most appropriate algorithm to a given input multi-label dataset, based on a set of multi-label predictive accuracy measures.

To perform the Auto-ML task, we propose a new evolutionary algorithm (EA). The EA performs a search in a very large search space of many different types of MLC algorithms, while it avoids generating invalid solutions to the problem at hand. As already showed in Pappa *et al.* [20], EA approaches are generally suitable to provide customized algorithms to specific ML problems, although that work focused on traditional single-label classification, rather than multi-label classification as in this work.

The proposed EA was tested in three datasets extracted from the Mulan Repository [33], and compared to two efficient MLC algorithms (as recommended by Madjarov *et al.* [14]): Binary Relevance [31, 32] and Classifier Chain [24]. Four MLC measures, from different evaluation perspectives, were used to analyze the predictive performance of the proposed method and the baselines. The results showed that the proposed EA obtained the best result according to two out of the four predictive accuracy measures, and it was ranked as the second and third method (among the three methods) in the other two measures. Overall, considering all three datasets and all four predictive accuracy measures, the EA's result was statistically significantly better than another method in 10 cases, whilst the EA was statistically worse than another method in only 5 cases. However, there is still room for improvement of these results, e.g., by improving the definition of the search space of MLC algorithms.

The reminder of this paper is organized as follows. Section 2 reviews related works on the MLC and Auto-ML areas. Section 3 details the proposed method, while Section 4 presents and discusses the preliminary results obtained. Finally, Section 5 draws some conclusions and discusses directions of future works.

2 RELATED WORK

This section is divided in two parts. The first part describes the main concepts of multi-label classification, and discusses different types of algorithms present in the literature. The second part refers to the Auto-ML task, i.e., related work on methods to generate learning algorithms customized to the input dataset.

2.1 Multi-label Classification

There is a wide range of studies on single-label classification in machine learning (ML) [2, 12]. In this type of task, each example of the dataset is associated to a single class $\lambda \in L$, where L is a set of disjoint classes. Given that $|L| > 1$, if $|L| = 2$ the problem is categorized as binary classification. Otherwise, if $|L| > 2$ the problem is categorized as multi-class classification.

Nevertheless, there are application domains where each example in the dataset can be associated to more than one class. In this case, we have a multi-label classification (MLC) problem, which is the focus of this work. Among the several domains that this problem encompasses, we can cite [32, 41]: tag recommendation, image and video annotation, bioinformatics, Web mining, and information retrieval. For instance, a Web document can be associated to two or more different class labels (e.g., sports and economics) at the same time.

More precisely, we consider that each example (or instance) X is a $d-$dimensional array, i.e., the examples are described by a list of d categorical and/or numerical features. If X is associated to a set of class labels, $Y \subseteq L$, this data is said to be multi-labeled.

According to Tsoumakas *et al.* [32], multi-label classification (MLC) is concerned with learning a model which returns a bipartition of the set of class labels. Given a query instance, this bipartition separates the labels into relevant and irrelevant ones. The majority of works in the literature follow the taxonomy proposed by Tsoumakas *et al.* [31], which divides MLC methods into problem transformation (PT) and algorithm adaptation (AA).

PT methods transform the multi-label dataset (task) into one or more single-label classification tasks. Using this concept, it is possible to apply single-label classifiers in order to obtain the classification outputs. However, a new step at the end of the process is required to map the single-label output(s) to a multi-label output. Among the several PT methods [3, 4, 32], it is important to mention binary relevance (BR), which learns $|L|$ independent binary classifiers, one for each label in the label set L; and label powerset (LP), which creates a single class for each unique set of labels that is associated with at least one example in a multi-label training set.

Additionally, there are some PT methods which modify the previously cited PT approaches in order to improve the predictive performance or to reduce the learning complexity. For example, pruned problem transformation (PPT) [23] extends LP by pruning label sets that occur less than a threshold. In another direction, random k-labelsets (RAkEL) [34] builds an ensemble of LP classifiers, training each classifier with a different and smaller random subset of the set of labels. Finally, classifier chain (CC) and ensemble of classifier chains (ECC) [24] change the BR method to take into account label correlations. To do this, CC creates L binary classifiers, like BR. However, unlike BR, the classifiers in CC are also linked along a chain, where each classifier deals with one BR problem. The attribute space of each link in the chain is increased with the classification outputs of all previous links. As the order of the classifiers in the chain is random, ECC tests different orders for CC in an ensemble fashion to improve its predictive performance.

On the other hand, AA methods aim to adapt traditional single-label classification algorithms to handle multi-label data. For instance, the multi-label version of C4.5 [5] for inducing decision trees modifies how the entropy is measured. Another example is BP-MLL, an adaptation of the back propagation algorithm to generate multi-label neural networks [39]. The main difference in BP-MLL is how it defines the error function to consider multi-labels. The literature presents many other adaptations of algorithms, such

as k-Nearest Neighbors [40], Naïve Bayes [38] and Support Vector Machines [8]. For more detail about MLC, see the surveys of Tsoumakas *et al.* [32] and Zhang & Zhou [41].

It is important to emphasize that there is a very large variety of MLC algorithms, each one having its own assumptions and biases. For example, when BR is chosen, the label correlations are disregarded. In the case of LP, RA*k*EL and CC, label correlations are considered differently. Different algorithm assumptions can lead to different predictive performances, depending on the characteristics of the dataset and the algorithm. As Tsoumakas *et al.* [32] remark, the label cardinality (average number of labels of the examples in the dataset) and the label density (average number of labels of the examples divided by the total number of labels in the dataset) differ for distinct MLC tasks. These two (and other) parameters related to the dataset tend to influence the way the MLC algorithm performs and, consequently, lead to good or poor class label predictions for different algorithms. Therefore, it is very difficult to choose the best MLC algorithm and its best parameter setting (i.e., the algorithm and parameter setting that maximize the predictive accuracy) for a particular dataset provided by a user. In this context, we propose an evolutionary algorithm for automatically selecting and configuring the best MLC algorithm for a given input dataset, which is a type of Automation of Machine Learning (Auto-ML) task, as discussed in the next Section.

2.2 Automatic Machine Learning (Auto-ML)

Auto-ML has recently become a growing research field as the ML community has been giving great attention to this topic. In spite of this fact, the field itself is not new, and has received different names in different works, such as constructive meta-learning, hyper-heuristics and hyper-parameter optimization [20]. The main reason for this is because Auto-ML emerged in the (super-)fields of ML and optimization in different time frames. Hence, the Auto-ML methods were developed mostly independently by both fields.

The Auto-ML methods have primarily focused on looking for the best combination of components of specific classification algorithms, instead of searching for complete classification pipelines with pre-processing, classification and post-processing methods. Six different types of classification algorithms had their components optimized by Auto-ML approaches: (i) artificial neural networks [17, 29, 37], (ii) rule induction algorithms [19], (iii) support vector machines [6, 15], (iv) decision trees [1], (v) Bayesian network classifiers [26] and (vi) Bayesian neural networks [28].

Following another direction, Auto-Weka [30], Auto-SKLearn [9], TPOT (Tree-based Pipeline Optimization Tool) [18] and RECIPE (REsilient ClassifIcation Pipeline Evolution) [27] deal with the Auto-ML task by generating complete classification pipelines. In other words, these methods produce customized classification solutions for a given dataset by considering the steps of pre-processing, classification and/or post-processing.

Auto-WEKA and Auto-SKLearn are methods based on Bayesian optimization and their main idea is to find the most appropriate mapping between ML pipelines and their respective parameters. Auto-WEKA works just for classification, whereas Auto-SKLearn deals with classification and regression problems. Both methods follow a hierarchical approach to find the "best" pipeline to a dataset at hand. Thus, they firstly choose the classification algorithm (or the pre-processing method) and, only after that, its parameters are optimized. Additionally, it is important to mention that Auto-WEKA automates the process of selecting the "best" ML pipeline in WEKA [10], whereas Auto-SKLearn aims to optimize the pipelines in the SciKit-Learn library [21].

TPOT and RECIPE, in turn, use evolutionary algorithms to discover the "best" combination of ML pipelines and their parameters. TPOT performs a genetic programing (GP) search to choose the most suitable pipeline for a ML problem (classification and regression). It also searches for methods available in the SciKit-Learn library, but it has a smaller search space than Auto-SKLearn.

One of the major drawbacks of TPOT is that it can create ML pipelines that are arbitrary/invalid, i.e., it can create a ML pipeline that fails to solve a classification problem, as there are no constraints on which type of components can be combined. RECIPE handles this problem by creating a grammar which organizes the knowledge acquired from the literature on how successful ML pipelines look like. A grammar-based genetic programing (GGP) [16] is then applied to solve the Auto-ML task. RECIPE works only for classification problems, but it has a bigger search space of (valid) classification pipelines than Auto-SKLearn and TPOT. Although this makes the search more challenging, it also provides the opportunity for producing a greater variety of pipelines. This could improve the classification performance as the Auto-ML approach is more likely to find a better ML pipeline.

It should be noted that all the previously described Auto-ML methods were designed to solve the conventional single-label Auto-ML task. By contrast, in this work we propose the first Auto-ML method for the multi-label classification task (i.e. multi-label Auto-ML). In the next section, we describe in detail the proposed evolutionary algorithm for automatically selecting and configuring the best MLC algorithm for a given input dataset.

3 AUTOMATICALLY SELECTING AND CONFIGURING MULTI-LABEL CLASSIFICATION ALGORITHMS

This section presents the proposed method to automatically select and configure MLC algorithms as illustrated in Figure 1. The method receives as input a specific multi-label dataset (with the attribute space X and the class labels L_1 to L_q), and a set of possible components (essential functional parts and parameters) identified from previously designed MLC algorithms. After that, an EA (which performs the multi-label Auto-ML task) is used to exploit these algorithms and their components, outputting in the end an MLC algorithm tailored to the input dataset. In other words, the MLC algorithm is specifically selected and parameterized to this data, although it could be applied to any multi-label dataset.

In the proposed method, each individual represents an MLC algorithm (see Section 3.2) randomly generated from a combination of the available components in the search space (described in Section 3.1). During the evaluation of the individuals, a mapping between the individual and an MLC algorithm is performed (see Section 3.3).

In each iteration of the evolutionary process, after selecting individuals (MLC algorithms) with a probability proportional to

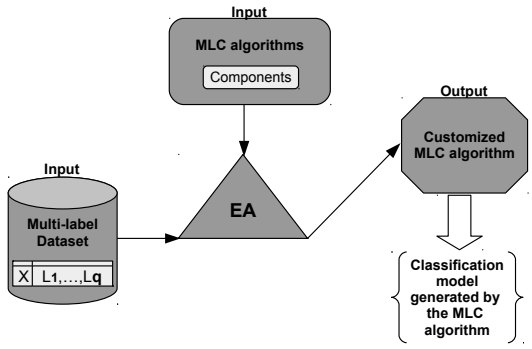

Figure 1: Proposed method to select and configure MLC algorithms.

their quality, uniform crossover and one-point mutation operations are applied to the selected individuals in order to generate a new population. Elitism is also used, copying the best individual from the current population to the next one. When a stopping criterion is reached (e.g., a predefined number of generations), the best MLC algorithm in the final population is returned, and its associated model is tested using a new data subset, which comes from the same application domain used for training.

3.1 Multi-label Classification Search Space

We developed an Auto-ML evolutionary approach for MLC using the MEKA framework [25], which is a multi-label extension to the WEKA software [10]. MEKA has a large variety of algorithms, focusing mainly on problem transformation methods. It also includes a variety of evaluation metrics from the literature, which are important to measure the performance of MLC algorithms from different classification perspectives.

The first step of this work was to understand the MLC search space in MEKA: the algorithms and their possible components, the constraints associated with different parameter settings for different types of data, the hierarchical nature of operations performed by problem transformation algorithms and meta-algorithms, and other issues. In total, we are using 31 MLC algorithms[1], which have their search spaces defined in Table 1. This table shows the algorithms, their numbers of parameters (#PAR.), and separates them into three types: algorithm adaptation (AA), problem transformation (PT) and Meta-Algorithms (Meta).

Most PT methods in Table 1 use a single-label classifier (SLC) in order to perform the transformed multi-label classification at a base level, except for MALC and MILC. Table 2 shows the used SLC algorithms, i.e., the possible algorithms at the base level and their (hyper-)parameters. We can observe that different types of SLC algorithms were selected to give a greater level of diversity to the evolutionary Auto-ML method. As explained in Section 2.1, the AA methods do not need to transform the data in a preprocessing step, applying their learning process in a straightforward way.

The meta-algorithms (with IDs 24-31 in Table 1) have at most two base levels, the multi-label and the single-label base levels. Except when the meta-algorithm chooses BPNN and DBPNN, the other cases present both base levels. However, we perceived some constraints that MEKA meta-algorithms have, which are:

- MBR (31) only works for BR (6).
- MALC (3) and MILC (4) do not work for any meta-algorithm.
- BCC (5) only can be used at the multi-label base level for the methods CM (26), EM (27), RSML (29) and SM (30).
- the MLC algorithms that have the ID from 1 to 2 and from 6 to 23 showed a good compatibility to all possible meta-algorithms (from 24 to 30).
- PMCC (18) does not work well with CM (26) and EM (27).

The complete hierarchy of choices of MEKA's MLC algorithms and parameter constraints is shown in Figure 2, using the IDs of the

[1]For more details, see http://meka.sourceforge.net/api-1.9/index.html

Table 1: Multi-label algorithms and meta-algorithms [11, 14, 25, 32, 41] used in the MEKA data mining tool.

ID	Algorithm	Acronym	Type	#PAR.
1	Back Propagation Neural Network	BPNN	AA	4
2	Deep Back-Propagation Neural Network	DBPNN	AA	5
3	Majority Labelset Classifier	MALC	PT	0
4	Minority Labelset Classifier	MILC	PT	0
5	Bayesian Classifier Chains	BCC	PT	1
6	Binary Relevance	BR	PT	0
7	Binary Relevance – Random Subspace Version	BRq	PT	1
8	Classifier Chain	CC	PT	0
9	Classifier Chain – Random Subspace Version	CCq	PT	1
10	Conditional Dependency Networks	CDN	PT	2
11	Conditional Dependency Trellis	CDT	PT	5
12	Classifier Trellis	CT	PT	5
13	Four-class pairWise classification	FW	PT	0
14	Label Powerset	LP	PT	0
15	Hierarchical Label Sets	HASEL	PT	3
16	Monte-Carlo Classifier Chains	MCC	PT	2
17	Probabilistic Classifier Chains	PCC	PT	0
18	Population of Monte-Carlo Classifier Chains	PMCC	PT	5
19	Pruned Sets	PS	PT	2
20	Pruned Sets with Threshold	PST	PT	2
21	RAndom k-labEL pruned sets	RakEL	PT	4
22	RAndom k-labEL Disjoint pruned sets	RakELD	PT	3
23	Ranking and Threshold	RT	PT	0
24	Bagging of Multi-Label classifiers	BaggingML	Meta	2
25	Bagging of Multi-Label classifiers (Duplicate)	BaggingMLDup	Meta	2
26	Classification Maximization	CM	Meta	1
27	Expectation Maximization	EM	Meta	1
28	Ensemble of Multi-Label classifiers	EnsembleML	Meta	2
29	Random Subspace Multi-Label	RSML	Meta	3
30	Subset Mapper	SM	Meta	0
31	Meta-Learning for Binary Relevance	MBR	Meta	0

Table 2: Single-Label Classification (SLC) algorithms [10, 36] used in the WEKA data mining tool.

ID	Algorithm	Acronym	Type	#PAR.
1	Naïve Bayes	NB	Bayes	0
2	Tree Augmented Naïve Bayes	TAN	Bayes	2
3	Hill Climbing Bayesian Network Classifier	HC	Bayes	3
4	Logistic Regression	LR	Function	1
5	Support Vector Machine	SVM	Function	2
6	K-Nearest Neighbors	KNN	Lazy	1
7	C4.5	C4.5	Trees	2
8	Random Forests	RF	Trees	1
9	PART	PART	Rules	2
10	OneR	OneR	Rules	1
11	Multilayer Perceptron	MLP	Function	4

algorithms in Table 1. It is important to note in this figure and in the previous descriptions of these constraints that some algorithms have much fewer constraints than others. This must be considered by the EA when it is generating the MLC algorithms.

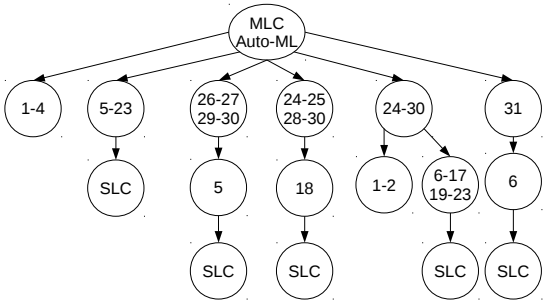

Figure 2: The proposed hierarchy of choices of algorithms and parameter constraints for the configuration of multi-label classification algorithms in MEKA.

Taking into account the constraints in the choices of algorithms' components and (hyper)-parameters in MEKA, we measured the size of the search space for the EA. In total, the search space of multi-label classification algorithms has 3.649×10^{20} possible MLC algorithm configurations. Next, we describe the proposed EA to perform the Auto-ML MLC task in this huge search space.

3.2 EA, Representation and Genetic Operators

In this work, we used a real-coded genetic algorithm (GA) [26] to search and explore the space of MLC algorithms. The GA is guided by the hierarchy of algorithm choices defined in the previous section and, thus, only generates valid individuals. A set of configuration files wraps the whole hierarchy.

Given the algorithms and their respective components, we followed the hierarchy to create a suitable representation for them. An individual is simply a real-valued array with 15 positions and its values are within the $[0, 1]$ interval. 15 is the maximum number of options an MLC algorithm can have: choosing MLP at the single-label base level (four parameters and the choice of the method itself), CDT, CT or PMCC at the multi-label base level (five parameters and the choice of the method itself) and RSML as the meta-algorithm (three parameters and the choice of the method itself). Each position of the array simply represents one component (parameter or the method itself). As mentioned earlier, different MLC algorithms may have different numbers of components. Note that for this reason, not all 15 positions in the array will be used, as some positions will refer to no component.

Hence, the GA uses a dynamic representation, but only at the phenotype level. The individual genotype representation is static, having always 15 positions. Crossover and mutation operations are then applied on the individual genotype (real-coded array) to avoid problems of individuals with different sizes.

We perform a uniform crossover operation, which builds a distributed binary mask with the same size of the individual genotype. In this case, if the value of the position in the mask is one, it means that the real-coded values of the genes – in both selected individuals – will be exchanged. Otherwise, the corresponding gene in the individuals is maintained and they do not suffer any modification in that position. Additionally, an one-point mutation operation is carried out into one of the 15 possible genes. For this operation, each gene has the same probability of being selected, and the value of the selected gene is replaced by a random real value in the same domain (which here varies from 0.0 to 1.0).

3.3 Fitness Function

In multi-label classification, researchers and practitioners typically evaluate the MLC algorithm using multiple measures because of the additional degrees of freedom the MLC setting introduces [14]. Usually, these measures follow different perspectives to quantify how good a classification algorithm is to a given dataset. For example, a measure may or may not consider correlations among the labels, or analyze the order (ranking) of the labels according to their relevance to a given instance. This performance evaluation for MLC algorithms differs significantly from single-label classification, where researchers and practitioners are used to evaluate the whole classification system using one single measure, such as F-measure [36].

Based on this MLC principle, we have to evaluate how effective the generated MLC algorithms are for a given dataset during the evolutionary process. In order to do this, the training set is divided into two parts: a learning set and a validation set. Each individual (MLC algorithm) builds a classification model from the learning set and measures its predictive accuracy on the validation set. Note that the test set is not accessed during the evolution, and it will be used only to measure the predictive accuracy of the MLC algorithm returned by the EA. Figure 3 illustrates the whole process of the evaluation of a given individual.

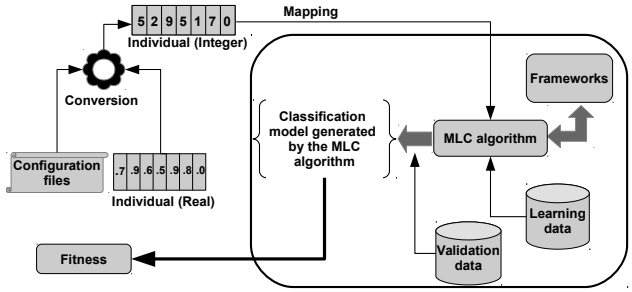

Figure 3: Evaluation process of one individual.

Initially, each individual in the GA is represented by a real-coded array, where each position of the genotype determines the use of a particular component option. Hence, the idea is to create a mapping between an individual position and an MLC component, using a set of configuration files that describe the algorithms and their respective components, and their complete hierarchy, as explained before in this section. The real-coded chromosome is therefore converted into an integer-coded chromosome of the same length, based on the configuration files.

In this conversion, the real number of a gene is multiplied by the maximum number of choices associated with that component, and the resulting value is rounded into an integer number that indicates

a component option in the configuration file. For instance, given that Table 2 is in one of the configuration files for a given PT method, suppose that the real value of a gene representing the single-label classification algorithm (for PT methods) is equal to 0.20, and, as we can see in this table, the total number of components for SLC algorithms is equal to 11. In this case, the rounded integer value will be 2, which means the second SLC algorithm – Tree Augmented Naïve Bayes (TAN) – in this configuration file will be selected. When the mapping process finishes, unused genes receive the value -1. The first position of the array is always used to choose one of the seven types of algorithms in the hierarchy, as we can observe in the number of tree leaves of Figure 2. This process guarantees that the GA will not generate invalid individuals.

Given the integer-coded individual, its chromosome is mapped to an MLC algorithm based on the configuration files. After that, we used the MEKA and WEKA frameworks to determine the MLC algorithm. In the next step, the algorithm is run on a learning set (part of the training set) to induce an MLC model, which is then evaluated using a validation set (the other part of the training set). The fitness function is generated from the validation set, using the average of four MLC measures [14, 32, 41]: Exact Match (EM), Hamming Loss (HL), F_1 Macro averaged by label (FM) and Ranking Loss (RL), as indicated in Equation 1:

$$Fitness = \frac{EM + (1 - HL) + FM + (1 - RL)}{4} \quad (1)$$

EM (also called subset accuracy) is considered a very strict evaluation metric, because it takes the value 1 when the predicted label set is an exact match to the true label set for an example, and value 0 otherwise. HL, on the other hand, calculates how many times an example-label pair is misclassified. In other words, it counts when a label not belonging to the example is predicted or when a label belonging to the example is not predicted. FM, in turn, is the harmonic mean between precision and recall, and its average is firstly calculated per label (across the examples in the dataset) and, after that, across all the labels in the dataset. This metric is interesting because it accounts for different levels of class imbalance of the data. Finally, RL measures the number of times that irrelevant labels are ranked higher than relevant labels, i.e., it penalizes the label pairs that are reversely ordered in the ranking for a given example. All four metrics are within the $[0, 1]$ interval. However, EM and FM are measures that should be maximized, whereas HL and RL should be minimized. In order for the fitness function to be maximized, in Equation 1, HL and RL are subtracted from one (1).

4 EXPERIMENTAL RESULTS

This section presents the preliminary experimental results of the proposed method in three datasets from the Mulan repository[2]. Table 3 presents the main characteristics of the datasets, including the number of instances (# instances), the number of features (# features) and classes labels (# labels), and the label cardinality and density. The relatively low number of datasets is due to the computational cost of the generated solutions.

We run all the experiments following a 10-fold cross-validation procedure with five repetitions varying the EA's random seed. We evaluated the results with four MLC measures from different perspectives, the same measures used in the EA's fitness function (Section 3.3). The results reported in this section correspond to the average and standard deviation obtained for the 50 executions (10-fold cross-validation × 5 repetitions) of the methods in the test set. Results were compared using the Wilcoxon Signed-Rank test [35] with 5% of significance. We also checked the general predictive performance using the achieved average ranking of the methods.

[2]Datasets are available in http://mulan.sourceforge.net/datasets-mlc.html

Table 3: Datasets used in the experiments.

dataset	# instances	# features	# labels	cardinality	density
flags	194	19	7	3.392	0.485
scene	2407	294	6	1.074	0.179
birds	645	260	19	1.014	0.053

The proposed EA for the MLC Auto-ML task was run using the following parameters: 100 individuals evolved for 100 generations, tournament selection of size two, elitism of five individuals and uniform crossover and mutation probabilities of 0.9 and 0.1, respectively. If the best individual remains the same for five generations and the current generation number is greater than 20, we stop the evolutionary process and return its respective MLC algorithm. The learning and validation sets are also resampled every five generations in order to avoid overfitting. Additionally, we assigned time and memory budgets for each MLC algorithm generated by the EA. Based on the size of the datasets used in the experiments, these budgets were set to 450 seconds (7.5 minutes) and 2GB of RAM, respectively.

The methods Binary Relevance (BR) and Classifier Chain (CC) were used as the baselines. We chose and set the parameters of these baselines following the recommendations of Madjarov *et al.* [14]). All baselines used at the single-label base level an SVM classifier with a radial basis kernel. The kernel parameter γ and the penalty C were optimized for each cross-validation run by using the training set only (i.e., choosing the best parameter value on the validation set). The values $\{2^{-15}, 2^{-13}, 2^{-11}, ..., 2^{1}, 2^{3}\}$ and $\{2^{-5}, 2^{-3}, 2^{-1}, ..., 2^{13}, 2^{15}\}$ were used to set γ and C, respectively. We performed an exhaustive search for the best parameter configuration of each baseline method considering these candidate parameter settings. It is also important to emphasize that identical time and memory budgets were applied in the baselines in order to make a fair comparison.

4.1 Comparison with the baseline methods

Table 4 shows the results of average and standard deviation of Exact Match (EM), Hamming Loss (HL), Ranking Loss (RL) and F_1 Macro averaged by label (FM). In this table, we are comparing the proposed method with the two aforementioned baselines. For the comparisons, the symbol ▲ denotes a statistically significant positive variation for the baseline in that column relative to the proposed EA and ▼ a statistically significant negative variation relative to the proposed method according to the Wilcoxon Signed Rank test.

Considering the average ranking for the EM measure, Table 4 shows that CC achieved the best average rank, followed by the proposed EA. Additionally, four out of six comparisons present statistically significant differences, with the proposed EA being

Table 4: Comparison of EM, HL, RL and FM obtained by the proposed EA and the baseline methods in the test set.

Exact Match (EM) – to be maximized			
Datasets	**EA**	**BR**	**CC**
Flags	0.184 (0.100)	0.134 (0.072) ▼	0.159 (0.073)
Scene	0.716 (0.039)	0.668 (0.040) ▼	0.738 (0.039) ▲
Birds	0.526 (0.066)	0.540 (0.063) ▲	0.545 (0.073)
Average Ranking	2.000	2.667	**1.333**
Hamming Loss (HL) – to be minimized			
Datasets	**EA**	**BR**	**CC**
Flags	0.260 (0.030)	0.262 (0.032)	0.283 (0.037) ▼
Scene	0.079 (0.009)	0.074 (0.008) ▲	0.076 (0.011) ▲
Birds	0.043 (0.007)	0.042 (0.008)	0.042 (0.009)
Average Ranking	2.333	**1.500**	2.167
Ranking Loss (RL) – to be minimized			
Datasets	**EA**	**BR**	**CC**
Flags	0.223 (0.052)	0.291 (0.056) ▼	0.314 (0.062) ▼
Scene	0.098 (0.032)	0.122 (0.016) ▼	0.120 (0.018) ▼
Birds	0.067 (0.036)	0.151 (0.035) ▼	0.147 (0.032) ▼
Average Ranking	**1.000**	2.333	2.667
F_1 Macro averaged by label (FM) – to be maximized			
Datasets	**EA**	**BR**	**CC**
Flags	0.617 (0.065)	0.613 (0.073)	0.606 (0.072)
Scene	0.785 (0.025)	0.788 (0.026)	0.794 (0.031) ▲
Birds	0.409 (0.050)	0.384 (0.083) ▼	0.398 (0.068)
Average Ranking	**1.667**	2.333	2.000

better than a baseline method in two comparisons and worse in the other two comparisons. Furthermore, the good result for CC and the bad one for BR were expected because this measure considers the maximum degree of correlation among class labels, whilst BR ignores correlations among class labels.

For the HL measure, BR showed the best average rank. As this metric disregards the class label correlations, this result is consistent. The proposed EA obtained the second best result. In the dataset Scene, BR and CC showed significantly better performances against the proposed EA. On the other hand, CC was statistically inferior to the proposed EA in the dataset Flags.

The best result for the proposed EA was achieved with the RL measure. As we can observe, the proposed EA was the best method in the ranking, followed by BR and CC. In addition, it presented a significantly better performance in all the six statistical comparisons for this measure.

Finally, the proposed EA reached the first place in the average ranking for the FM measure, followed by CC and BR. CC presented a significantly better result in the dataset Scene when compared to the proposed EA. The EA reported a statistically significant positive variation in the comparison only against BR for the dataset Birds.

4.2 Analyzing the selected MLC algorithms

We also analyze the MLC algorithms selected by the EA in order to accomplish the MLC Auto-ML task. Table 5 presents the percentage of selection at three different levels: the single-label classification (SLC) base level, multi-label classification (base) level and meta level. This table refers to the algorithm acronyms in Tables 1 and 2. At the SLC base level, we can observe that RF is overall the most frequently selected algorithm across the three datasets, especially for the datasets Scene (58% of the times) and Birds (100% of the times). For the dataset Flags, the algorithm PART had the highest percentage (46%) of selection by the EA.

Table 5: Percentage of selection of MLC algorithms at the single-label classification (SLC) base level, multi-label classification (ML base) level and meta level.

Dataset	SLC base level	%	MLC (base) level	%	Meta level	%
Flags	PART	46	LP or RakEL	28	None	60
	RF	24	BR or CC(q)	26	BaggingML	20
	C4.5	18	FW	22	EnsembleML	12
	Others	12	Others	24	Others	8
Scene	RF	58	LP or RakEL	58	None	80
	TAN	26	PS(t) or RT	30	MBR or RSML	12
	SVM	12	BR or CC(q)	12	Others	8
Birds	RF	100	BR	74	None	54
	Others	0	LP	20	MBR	34
	-	-	Others	6	Others	12

The multi-label level is a base level when it is associated to a meta-algorithm or it is simply the multi-label level when this does not occur. At this level, the selected algorithms are more diverse than at the SLC base level. For this reason, we grouped some algorithms together in the table. For the datasets Flags and Scene, the algorithms LP or RAkEL were selected in 28% of the cases for the dataset Flags and in 58% for the dataset Scene. BR was selected in 74% of the cases in the dataset Birds.

Finally, the majority of the selected MLC algorithms do not have the meta level. For example, in the dataset Scene the EA did not choose a meta-algorithm in 80% of the cases. This is mainly due to the complexity of such algorithms as they have to train and run many classifiers considering the same time budget. The meta-algorithms BaggingML (Dup or not) or EnsembleML were selected in 32% of the cases for the Flags dataset. And for the dataset Birds, MBR was the most chosen algorithm (34% of the times).

5 CONCLUSIONS AND FUTURE WORKS

This work introduced the first version of an evolutionary algorithm (EA) for automatically selecting and configuring multi-label classification (MLC) algorithms. Each EA individual represents a full MCL algorithm with a particular parameter configuration. The EA received as input an MLC dataset and a list of MLC algorithms' components, and returns an MLC algorithm and its parameter configuration customized to (i.e., maximizing the predictive accuracy on) that particular dataset.

The EA was tested in three MLC datasets from the Mulan repository, and compared to two well-known baseline MLC methods, namely Binary Relevance (BR) and Classifier Chain (CC). The predictive accuracy comparisons were based on four different MLC measures (Exact Match, Hamming Loss, Ranking Loss and F_1 Macro). Overall, considering that the proposed EA's results were compared against the results of two (2) baseline methods across three (3) datasets and four (4) predictive accuracy measures, 24 (2 x 3 x 4) statistical tests of significance were performed. The results of these tests were that the EA significantly outperformed another method in 10 cases, the EA was significantly outperformed by another method in five (5) cases, and in the remaining nine (9) cases there was no significant difference. In terms of the results for each predictive accuracy measure separately, the EA obtained the best result according to two out of the four measures, and in particular it obtained statistically better results than the other two methods

on all three datasets according to the Ranking Loss measure. The EA was the worst method only according to the Hamming Loss measure, although the statistical significance results are less strong in this case: the EA's performance was statistically worse than the other two methods on only one of the three datasets, whilst the EA statistically outperformed the Classifier Chain algorithm on one of the datasets. However, there is room for improvement as we can improve the definition of the MLC search space.

The next step for this work is to test the algorithm in more datasets available in the Mulan repository. This evaluation will help us to understand the performance of the proposed method in a larger number of domains. As another future work, the number of MLC algorithms and components will be extended, and we will measure the predictive performance of the EA using different search spaces. In addition, we aim to generate complete MLC pipelines. Therefore, our method will choose not just the MLC algorithm, but also the preprocessing methods (e.g., the feature selection algorithm [22] and its associated parameters). Finally, we plan to compare the proposed EA to an automated parameter optimization tool, such as irace [13].

ACKNOWLEDGMENTS

This work was partially supported by the following Brazilian Research Support Agencies: CNPq, CAPES (especially, the grant number 88881.136011/2016-01) and FAPEMIG.

REFERENCES

[1] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas. 2013. Automatic Design of Decision-Tree Algorithms with Evolutionary Algorithms. *Evolutionary Computation* 21, 4 (2013), 659–684.

[2] S. Boucheron, O. Bousquet, and G. Lugosi. 2005. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics* 9 (2005), 323–375.

[3] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition* 37, 9 (2004), 1757–1771.

[4] W. Chen, J. Yan, B. Zhang, Z. Chen, and Q. Yang. 2007. Document Transformation for Multi-label Feature Selection in Text Categorization. In *Proc. of the IEEE International Conference on Data Mining (ICDM)*. 451–456.

[5] A. Clare and R. D. King. 2001. Knowledge Discovery in Multi-label Phenotype Data. In *Proc. of the European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*. Springer-Verlag, London, UK, 42–53.

[6] L. Dioşan, A. Rogozan, and J.-P. Pecuchet. 2012. Improving classification performance of Support Vector Machine by genetically optimising kernel shape and hyper-parameters. *Applied Intelligence* 36, 2 (2012), 280–294.

[7] P. Domingos. 2012. A Few Useful Things to Know About Machine Learning. *Commun. ACM* 55, 10 (Oct. 2012), 78–87.

[8] A. Elisseeff and J. Weston. 2001. A Kernel Method for Multi-labelled Classification. In *Proc. of the International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS)*. MIT Press, Cambridge, MA, USA, 681–687.

[9] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Proc. of International Conference on Neural Information Processing Systems*. MIT Press, 2755–2763.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explorations Newsletter* 11, 1 (Nov. 2009), 10 – 18.

[11] F. Herrera, F. Charte, A. J. Rivera, and M. J. del Jesus. 2016. *Multilabel Classification : Problem Analysis, Metrics and Techniques* (1 ed.). Springer International Publishing.

[12] S. B. Kotsiantis. 2007. Supervised Machine Learning: A Review of Classification Techniques. In *Proc. of the Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. IOS Press, 3–24.

[13] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, T. Stützle, and M. Birattari. 2016. The irace package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives* 3 (2016), 43–58.

[14] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Dzeroski. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition* 45, 9 (2012), 3084–3104.

[15] R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl, and A. C. P. L. F. de Carvalho. 2015. Effectiveness of Random Search in SVM hyper-parameter tuning. In *Proc. of the International Joint Conference on Neural Networks*. 1–8.

[16] R. I. McKay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'Neill. 2010. Grammar-based Genetic Programming: a survey. *Genetic Programming and Evolvable Machines* 11, 3 (2010), 365–396.

[17] H. Mendoza, A. Klein, M. Feurer, J. Springenberg, and F. Hutter. 2016. Towards Automatically-Tuned Neural Networks. In *Proc. of the ICML AutoML Workshop*.

[18] R. Olson, R. Urbanowicz, P. Andrews, N. Lavender, L. Kidd, and J. H. Moore. 2016. Automating Biomedical Data Science Through Tree-Based Pipeline Optimization. In *Proc. of the European Conference on the Applications of Evolutionary Computation*. Springer International Publishing, 123–137.

[19] G. L. Pappa and A. A. Freitas. 2009. *Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach*. Springer.

[20] G. L. Pappa, G. Ochoa, M. R. Hyde, A. A. Freitas, J. Woodward, and J. Swan. 2014. Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. *Genetic Programming and Evolvable Machines* 15, 1 (2014), 3–35.

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

[22] R. B. Pereira, A. Plastino, B. Zadrozny, and L. H. C. Merschmann. 2016. Categorizing feature selection methods for multi-label classification. *Artificial Intelligence Review* (2016), 1–22.

[23] J. Read. 2008. A pruned problem transformation method for multi-label classification. In *Proceedints of the New Zealand Computer Science Research Student Conference (NZCSRS)*. 143–150.

[24] J. Read, B. Pfahringer, G. Holmes, and E. Frank. 2011. Classifier Chains for Multi-label Classification. *Machine Learning* 85, 3 (Dec. 2011), 333–359.

[25] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes. 2016. MEKA: A Multi-label/Multi-target Extension to Weka. *Journal of Machine Learning Research* 17, 21 (2016), 1–5.

[26] A. G. C. Sá and G. L. Pappa. 2014. A Hyper-heuristic Evolutionary Algorithm for Learning Bayesian Network Classifiers. In *Proc. of Ibero-American Conference on Artificial Intelligence*. 430–442.

[27] A. G. C. Sá, W. J. G. S. Pinto, L. O. V. B. Oliveira, and G. L. Pappa. RECIPE: A Grammar-based Framework for Automatically Evolving Classification Pipelines. In *Proc. of the European Conference on Genetic Programming (EuroGP)*. Springer International Publishing, 246–261.

[28] J. T. Springenberg, A. Klein, S.Falkner, and F. Hutter. 2016. Bayesian optimization with robust Bayesian neural networks. In *Proc. of the Conference on Neural Information Processing Systems*.

[29] K. O. Stanley and R. Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary Computation* 10, 2 (2002), 99–127.

[30] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2013. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proc. of KDD*. ACM, 847–855.

[31] G. Tsoumakas and I. Katakis. 2007. Multi-label classification: An overview. *International Journal on Data Warehousing and Mining* 3, 3 (2007), 1–13.

[32] G. Tsoumakas, I. Katakis, and I. Vlahavas. 2010. Mining Multi-label Data. In *Data Mining and Knowledge Discovery Handbook*, Oded Maimon and Lior Rokach (Eds.). Springer US, Boston, MA, 667–685.

[33] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. 2011. Mulan: A Java Library for Multi-Label Learning. *Journal of Machine Learning Research* 12 (2011), 2411–2414.

[34] G. Tsoumakas and I. Vlahavas. 2007. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *Proc. of the European Conference on Machine Learning (ECML)*. Springer-Verlag, Berlin, Heidelberg, 406–417.

[35] F. Wilcoxon, S. K. Katti, and R. A Wilcox. 1970. Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. *Selected tables in mathematical statistics* 1 (1970), 171–259.

[36] Ian H. Witten, Eibe Frank, and Mark A. Hall. 2011. *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Morgan Kaufmann Publishers Inc.

[37] X. Yao. 1999. Evolving artificial neural networks. *Proc. of the IEEE* 87, 9 (1999), 1423–1447.

[38] M.-L. Zhang, J. M. Pena, and V. Robles. 2009. Feature selection for multi-label naive Bayes classification. *Information Sciences* 179, 19 (2009), 3218–3229.

[39] M.-L. Zhang and Z.-H. Zhou. 2006. Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE Transactions on Knowledge and Data Engineering* 18, 10 (2006), 1338–1351.

[40] M.-L. Zhang and Z.-H. Zhou. 2007. ML-KNN: A Lazy Learning Approach to Multi-label Learning. *Pattern Recognition* 40, 7 (2007), 2038–2048.

[41] M.-L. Zhang and Z. H. Zhou. 2014. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1819–1837.