Tree-Shaped Ensemble of Multi-Label Classifiers using Grammar-Guided Genetic Programming

Jose M. Moyano*, Eva L. Gibaja*, Krzysztof J. Cios†‡ and Sebastián Ventura*
*Dept. of Computer Science and Numerical Analysis
University of Córdoba, Córdoba, Spain
Email: jmoyano@uco.es, egibaja@uco.es, sventura@uco.es
†Dept. of Computer Science
Virginia Commonwealth University, Richmond, VA, USA
Email: kcios@vcu.edu
‡ Polish Academy of Sciences, Poland

***Abstract*—Multi-label classification paradigm has had a growing interest because of the emergence of a large number of classification problems where each of the instances of the data can be associated with several output labels simultaneously. Several ensemble methods were proposed to solve the multi-label classification problem. However, most of them simply create diversity in the ensemble by following a random procedure and give the same importance to all members. In this paper, we propose a Grammar-Guided Genetic Programming algorithm to build ensembles of multi-label classifiers. Given a pool of multi-label classifiers, each of them modeling dependencies among a subset of *k* labels, they are combined into a tree-shaped ensemble. At each node of the tree, predictions of its children nodes are combined, while each leaf represents a classifier from the pool. We propose two configurations for the method: using a fixed value of *k* for all classifiers in the pool, or using a variable value of *k* for each classifier, thus being able to capture relationships among groups of labels of different size in the ensemble. The experiments performed over sixteen multi-label dataset and using five evaluation metrics demonstrated that our method performs significantly better than the state-of-the-art ensembles of multi-label classifiers.**

***Index Terms*—Genetic programming, Multi-label classification, Ensemble learning**

I. Introduction

In recent years, classification problems where each of the instances of the data may belong to several classes/output labels simultaneously associated with it, are increasingly frequent. For example, in medical diagnosis systems, patients may have more than one disease, or complications, at the same time. Traditional classification methods are only able to deal with one class per instance. Thus, the Multi-Label Classification (MLC) paradigm emerged for addressing these situations [1]. MLC has been successfully applied to many real-world problems in addition to medical diagnosis [2], such as biology [3] and multimedia categorization [4].

Dealing with several output labels at the same time leads to emergence of new challenges such as modeling the compound dependencies among the labels, imbalance, and high dimensionality of the output space. Although a wide range of MLC methods has been proposed [1], we focus here our attention on Ensembles of Multi-Label Classifiers (EMLCs), which have been shown to outperform the base methods [5]–[7]. Ensemble classifiers are methods that combine predictions of several base classifiers, aiming to improve the overall generalization ability of each. Selection of classifiers to combine, however, is not trivial as they should not only be accurate but also diverse [8], [9].

Although the EMLCs outperform their base methods, they usually combine classifiers where the diversity is created by following any random procedure (such as randomly selecting instances or labels at each member of the ensemble), and the same weight or importance is given to all base classifiers. In this paper, we propose a Grammar-Guided Genetic Programming (G3P) algorithm able to generate EMLCs. G3P, which is an extension of Genetic Programming (GP), is an evolutionary learning technique that uses syntax trees to represent the individuals, and also a grammar to guide the learning process, such as the creation of initial individuals [10]. Using G3P a tree-shaped ensemble is obtained; at each node of the tree the predictions of children nodes are combined, while the leaves are the base multi-label classifiers. The use of G3P makes the selection of members of the ensemble more flexible, allowing to adapt to each particular problem, as well as to obtain an optimal structure of the ensemble.

In our method each of the base classifiers of the ensemble focuses only on a subset of k labels, also known as k-labelset. In this way, each member is able to consider the relationship among the labels, while drastically reducing the imbalance and high-dimensionality of the output space. In addition, our method is not only able to deal with fixed k for all base classifiers, but also is able to use different values of k in each base classifier, to capture the relationship among subsets of labels of different size. The experimental study using 16 datasets and five evaluation metrics, demonstrated that our method obtains significantly better performance than state-of-the-art MLC methods.

The rest of the paper is organized as follows. Section II provides background about MLC and G3P; Section III describes the G3P-based method for building the EMLCs; Section IV introduces the experimental study; Section V presents and discusses the results; and Section VI ends with conclusions.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

II. Background

In this section, we first describe MLC and state-of-the-art MLC methods, and then we introduce the G3P framework.

A. Multi-Label Classification

Let $\mathcal{X} = X_1 \times \cdots \times X_d$ be the d-dimensional input space, and $\mathcal{Y} = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$ the output space composed by $q > 1$ labels. Let $\mathcal{D}$ be a multi-label dataset composed of m instances, as $\mathcal{D} = \{(\mathbf{x}_i, Y_i) | 1 \leq i \leq m\}$, where each multi-label instance is a pair composed by an input feature vector $\mathbf{x} \in \mathcal{X}$ and a set of relevant labels $Y \subseteq \mathcal{Y}$ associated with it [1]. The goal of MLC is to construct a model able to provide a set of predicted relevant labels $\hat{Y}$ for an unknown instance $\mathbf{x}$.

MLC algorithms are categorized into three groups: I) problem transformation, which transform the multi-label problem into one or several single-label problems; II) algorithm adaptation, which directly adapt traditional classification methods to be able to deal with multi-label data without transforming it; and III) EMLCs, defined as a set of n multi-label classifiers, each of them providing prediction for all or part of the labels [7]. Finally, the prediction of all n classifiers are combined, usually by majority voting, although many other combining methods can be used [11].

Ensemble of Binary Relevances (EBR) algorithm [5] combines the predictions of n Binary Relevance (BR) methods [12]. As BR creates an independent binary model for each label, EBR is not able to model the relationship among the labels. Also, diversity of classifiers in EBR is obtained by simply randomly selecting a subset of the instances for each of the members, which is a weakness.

Ensemble of Classifier Chains (ECC) [5] combines the predictions of n Classifier Chain (CC) methods. CC also creates binary models for each label, but they are not independent as in BR as they are chained in such a way that predictions of previous labels in the chain are introduced as additional input features in the subsequent binary classifiers. Therefore, ECC is able to model some of the relationship among labels. The diversity in ECC is obtained not only by selecting random subsets of the instances but also using different random chains at each member.

Ensemble of Pruned Sets (EPS) algorithm [13] combines the predictions of n Pruned Sets (PS) methods. PS follows the Label Powerset (LP) [14] approach, transforming the multi-label problem into a multi-class problem, where different combinations of labels (a.k.a. labelsets) are considered as different classes; then, PS prunes those instances associated with very infrequent classes, leading to less imbalanced problems. In this way, EPS is able to model the relationship among all labels; however, although it prunes infrequent labelsets, the resulting multi-class problem still tends to be imbalanced with a high number of classes, resulting in a still complex problem. As for the diversity, EPS selects random subset of instances at each member of the ensemble.

RAndom k-labELsets (RAkEL) [6] builds an ensemble of LP methods, but in this case each member of the ensemble focuses on a small subset of k labels, being k fixed for all members. Therefore, each member of the ensemble is able to deal with the relationships among k labels, leading to less imbalanced and lower dimensional problems than if all labels were considered at the same time. RAkEL selects the k-labelsets randomly to generate diversity in the ensemble.

Random Forest of Predictive Clustering Trees (RF-PCT) [15] builds an ensemble of Predictive Clustering Trees (PCTs) [16]. Each member uses a random selection of training instances, and at each node of the tree selects the best feature from a random subset of attributes.

In all described methods, the diversity among the members is generated following a random process, and the combination of labels is performed by majority voting, with the same weight given to each member of the ensemble.

B. Grammar-Guided Genetic Programming (G3P)

GP is an evolutionary and very flexible heuristic technique which allows the use of very complex individual representations. G3P is an extension of GP, where a free-context grammar is used to generate the individuals. Each individual is represented as a syntax tree, where internal or non-terminal nodes correspond to functions taking their children as arguments, and leaves or terminal nodes correspond to variables and constants. The use of the grammar provides ability of applying certain constraints at each node of the tree, such as the number or type of the child nodes, and also ensures that all generated individuals represent a valid solution [17].

G3P has been widely used in the literature on a large number of different problems, such as bankruptcy prediction [18], predicting student performance [19], and discovery of subgroups within a population [20], as well as it has been proven to work well on high-dimensional scenarios [21]. However, there are not many multi-label classification methods that are based on G3P. In [22] a G3P algorithm to build a rule-based multi-label classifier was proposed. In [23], an algorithm that automatically selects the most appropriate multi-label classifier is proposed using G3P. Nonetheless, to the best of our knowledge, no studies exist in applying either GP or G3P to the construction of EMLCs.

III. G3P-kEMLC

In this section we present the proposed method, called G3P-kEMLC. First, the main steps of the algorithm are presented, and then, the individuals, fitness function, and genetic operators are described.

A. Algorithmic strategy

The main steps of the G3P-kEMLC algorithm are shown in Fig. 1. First, a pool of n multi-label classifiers $\text{MLC}_1, \text{MLC}_2, \ldots, \text{MLC}_n$, where each is focused on predicting k labels, is created. Although any multi-label classifier could be used, we will use LP, in order to model the relationship among all k labels at the same time [6]. Unlike other methods, such as RAkEL, G3P-kEMLC is able to handle multi-label classifiers using different values of k. Therefore, two parameters k_{min} and k_{max} are given, so that each classifier

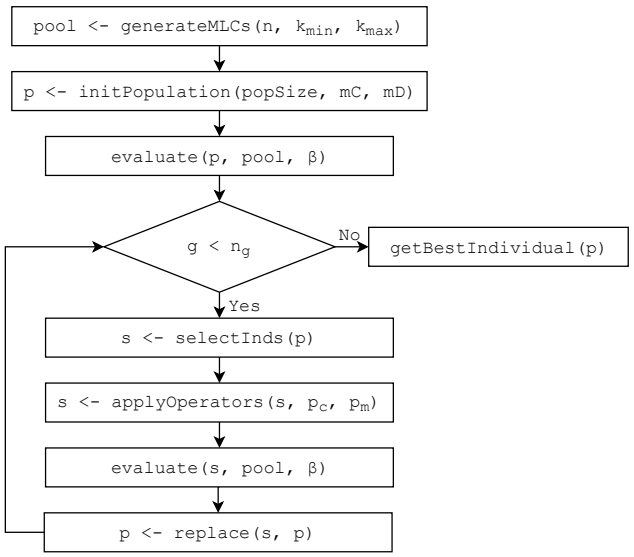


Fig. 1. Main steps of G3P-kEMLC.

will focus on a subset of k labels, being k a random value in the range $[k_{\text{min}},\ k_{\text{max}}]$. Note that repeated k-labelsets are not allowed, so if the generated random k-labelset is currently present in the pool, it is discarded and a new one is created. To increase diversity of the base classifiers, each of them is built over a random subset of the instances. Although building the base classifiers over random subsets of instances and labels, the difference between G3P-kEMLC and other EMLCs is that G3P-kEMLC will look for an optimal structure of the ensemble by means of the evolutionary procedure, instead of just giving the same weight to all base classifiers in the combination of predictions. In our tree-shaped ensemble, classifiers that are placed at a shallower depth have more importance in the final prediction than classifiers that are deeper in the tree.

Then, the initial population p of $popSize$ individuals is generated by using the grammar (see Section III-B). Note that mC and mD parameters indicate the maximum number of children of each node of the tree, and the maximum allowed depth of the tree, respectively. Once initial individuals are created, they are evaluated (see Section III-C).

Until the maximum number of generations n_g is reached, individuals are selected by tournament selection, genetic operators are applied (see Section III-D), and new individuals are evaluated. For the replacement of the population, the population at next generation is formed by all new children. However, if the best parent is better than the best child, it replaces the worst child, thus maintaining elitism. Once the maximum number of generations is reached, the best individual is returned as the optimal ensemble.

B. Individuals

Each individual in G3P-kEMLC is represented as a string encoding a tree-shaped ensemble. As non-terminal nodes, only one function called *Comb* is used, which combines the

```
<S>: <Comb>
<Comb>: <start> (<Comb>|<MLC>){2, mC} <end>
<start>: '('
<end>: ')'
<MLC>: MLC_1 | MLC_2 | ... | MLC_n
```

Fig. 2. Free-context grammar.

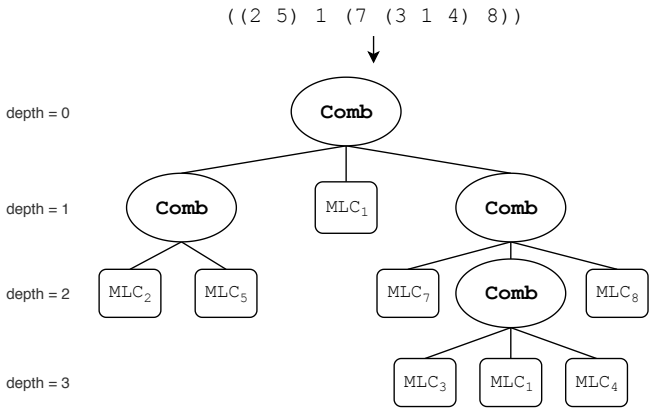


Fig. 3. Individual as string and its corresponding tree.

predictions of children nodes. For each label λ_l that is included in any of the children of the *Comb* node, it computes the ratio of positive predictions among all the children; if it is greater or equal than a given threshold t (as default $t = 0.5$ is used), the combined prediction for λ_l is positive, and negative otherwise. On the other hand, as terminal nodes, we use the previously generated pool of multi-label classifiers, represented as integers from 1 to n. Each of the multi-label classifiers gives prediction for the labels in their own k-labelsets. Note that, since each leaf is focused on a subset of k labels (and k could be different for each leaf), the number of labels considered at each *Comb* node can be different.

In Fig. 2 the grammar used to build individuals is shown. The initial node `S` is always replaced by a *Comb* node. Each *Comb* node starts and ends with '(' and ')' characters respectively, to represent (in the string) the hierarchy of the nodes. Each *Comb* node may contain from 2 to mC children nodes, each of them being either another *Comb* node or a *MLC* (leaf). The number of children is randomly selected in the given range $[2, mC]$ at each node, so each internal node could have a different number of children. However, the maximum depth (mD) of the tree is controlled in such a way that if the depth of the current *Comb* node is equal to $mD - 1$, only *MLC* nodes can be selected as children of this *Comb*. In Fig. 3 an example of an individual obtained using the grammar is shown, both as a string and as its corresponding tree.

The use of the grammar is helpful in two main aspects. On the one hand, it allows to determine the number of child nodes at each *Comb* node, as well as if these children are either other *Comb* nodes or leaves. On the other hand, if different types of combiner nodes for the predictions were used (as

we propose for future work), the use of the grammar would become essential, since it would select the appropriate type and/or number of child nodes depending on different type of combiner node.

C. Evaluation

For evaluation of the individuals, we use a fitness function (Eq. 1) which differentiates between incomplete and complete trees. Hereafter, it is indicated with $\uparrow$ and $\downarrow$ if metrics are maximized or minimized, respectively. L_t is the set of labels that the tree is considering among all the leaves. We define complete trees as those that include at least one vote for each label in the dataset ($|L_t| = l$), while incomplete trees are not able to give prediction for all labels ($|L_t| < l$).

$$\uparrow fitness = \begin{cases} -\left(l - |L_t|\right)/l & \text{if } |L_t| < l \\ \beta \cdot \text{ExF} + (1-\beta) \cdot \text{MaF} & \text{if } |L_t| = l \end{cases} \quad (1)$$

If the individual represents an incomplete tree, its fitness is a negative value, being closer to zero as the number of labels that are not considered is lower. We aim to remove incomplete individuals in the population, but in case when several incomplete individuals are chosen to compete with each other in the selection procedure, the one that is closer to be a complete tree is selected.

On the other hand, if the individual is a complete tree, first the predictions for the whole training set are obtained by combining predictions of internal nodes, until the final prediction of the ensemble is obtained at the root node. Then, the fitness function, composed of two terms, is computed. FMeasure is a robust evaluation metric that has been widely used to evaluate models in cases where the output space is imbalanced [24]. In MLC, several approaches are defined to compute the FMeasure, such as Example-based FMeasure (ExF) and Macro-averaged FMeasure (MaF) [25]. ExF (Eq. 2) computes the FMeasure of each instance as a whole, thus capturing the relationships among the labels in its calculation. On the other hand, MaF (Eq. 3) computes the FMeasure for each label independently and averages it by the number of labels, thus giving the same importance to all labels in its calculation. Note that tp_l, fp_l, and fn_l stand for the number of *true positives*, *false positives*, and *false negatives* of the l-th label, respectively.

$$\uparrow \text{ExF} = \frac{1}{m} \sum_{i=1}^{m} \frac{2|\hat{Y}_i \cap Y_i|}{|\hat{Y}_i| \cup |Y_i|} \quad (2)$$

$$\uparrow \text{MaF} = \frac{1}{q} \sum_{l=1}^{q} \frac{2 \cdot tp_l}{2 \cdot tp_l + fp_l + fn_l} \quad (3)$$

Therefore, using a combination of both ExF and MaF, we not only consider the relationship among labels when calculating the FMeasure, but also ensure that minority labels are also considered. Further, the fact of calculating the fitness function over the whole training set, while each base classifier is built over a subset of the same data, also offers an approximation of how each of them performs on unseen data.

D. Genetic operators

Each individual of the population is selected for crossover and mutation operators based on probabilities p_c and p_m respectively. Fig. 4 illustrates use of both operators.

1) Crossover operator: Given two parents, the crossover operator creates a new individual as follows: I) a random subtree st_1, not including the whole tree, is selected from the first parent; II) a random subtree st_2, including the possibility to select the whole tree, is selected from the second parent; III) if the maximum depth of st_2 is greater than the maximum allowed depth of the tree minus the depth of the root node of st_1, step II is repeated again but selecting a random subtree of st_2; IV) st_2 replaces st_1 in the first parent, obtaining the child individual. Crossed individuals include genetic material of both parents, and because of step III they are always feasible, as we ensure not to exceed the maximum allowed depth of the tree.

Regarding the example in Fig. 4a, consider that the subtree whose root is the shaded node of the first parent is selected as st_1. Then, the node marked with a dotted line in the second parent is first selected as st_2. However, considering $mD = 3$, if st_2 replaces st_1, the maximum depth constraint would not be met in the generated child; therefore, a new random subtree below this node is then selected as st_2 (shaded node of the second parent). Finally, the first child is created by replacing st_1 with st_2.

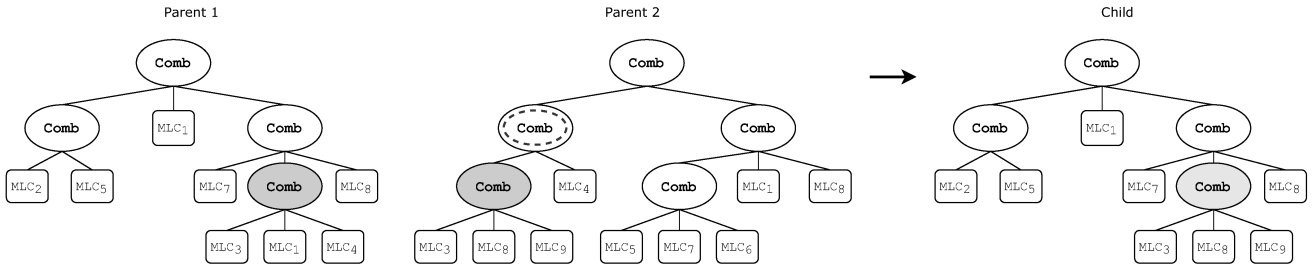
Given this crossover operator, just one child is obtained by each pair of parents. Proposing an operator where two children were obtained, for example swapping subtrees of the parents, would make more difficult the selection of these subtrees in such a way that they both fit in the other parent and do not break the maximum depth restriction in any of them. Therefore, the second child is obtained by following the same procedure but swapping the roles of each of the parents.

2) Mutation operator: For the mutation operator, the steps are: I) a random subtree st, not including the whole tree, is selected; II) a subtree is created following the grammar, but using as maximum depth the maximum allowed depth minus the depth of the root node of st; III) the generated tree replaces st. Therefore, a subtree of the individual is replaced by a random subtree, ensuring that it is feasible thanks to the use of the grammar and control of the maximum depth.

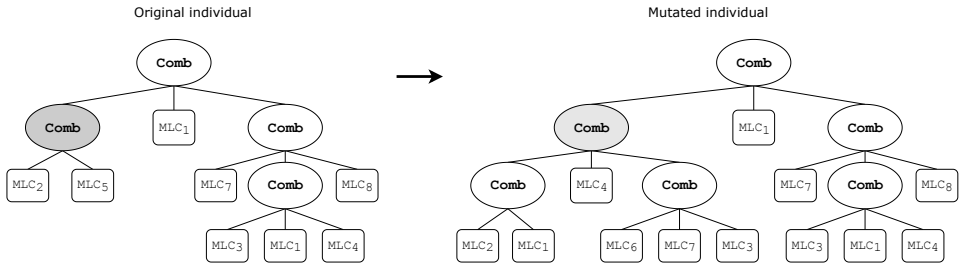
According to Fig. 4b, assume that the shaded node is randomly selected as root of the subtree to mutate. Then, a random new subtree is created following the grammar to replace it, creating the mutated individual.

IV. Experimental Study

In this section, first the datasets and evaluation metrics used in the experiments are described, and then, the configuration of both G3P-kEMLC and the state-of-the-art MLC methods are presented.



(a) Crossover



(b) Mutation

Fig. 4. Genetic operators.

TABLE I
DATASETS AND CHARACTERISTICS. THE DATASETS ARE ORDERED BY THE NUMBER OF LABELS.

Dataset	m	d	q	*card*	*avgIR*	*rDep*
Emotions	593	72	6	1.868	1.478	0.933
Reuters1000	294	1000	6	1.126	1.789	0.667
Guardian1000	302	1000	6	1.126	1.773	0.667
Bbc1000	352	1000	6	1.125	1.718	0.733
3s-inter3000	169	3000	6	1.142	1.766	0.400
GnegativePseAAC	1392	1717	8	1.046	18.448	0.536
PlantPseAAC	978	440	12	1.079	6.690	0.318
Water-quality	1060	16	14	5.073	1.767	0.473
Yeast	2417	103	14	4.237	7.197	0.670
HumanPseAAC	3106	440	14	1.185	15.289	0.418
Birds	645	260	19	1.014	5.407	0.123
Genbase	662	1186	27	1.252	37.315	0.157
Medical	978	1449	45	1.245	89.501	0.039
NusWide[2]	2696	128	81	1.863	89.130	0.087
Stackex coffee	225	1763	123	1.987	27.241	0.017
CAL500	502	68	174	26.044	20.578	0.192

A. Datasets

A selection of 16 multi-label datasets from the KDIS repository[1], covering a wide range of characteristics, were used in the experiments and are shown in Table I. The number of instances (m), attributes (d), and labels (q) of each dataset are shown, as well as the cardinality or average number of labels associated with each instance (*card*), the average imbalance ratio (*avgIR*), and the ratio of dependent label pairs (*rDep*) [26].

B. Evaluation metrics

Five evaluation metrics are used to assess the performance of the multi-label methods [25]. The Adjusted Hamming loss (AHL) [27] was proposed because of the drawbacks of the Hamming loss, which tends to be zero in cases with a large number of labels but low cardinality. AHL, defined in Eq. 4, computes the ratio of misclassified labels divided by the number of positive labels (in both true and predicted sets), and averages it by the total number of instances. Note that Δ is the symmetric difference between two binary sets.

$$\downarrow \text{AHL} = \frac{1}{m}\sum_{i=1}^{m}\frac{|\hat{Y}_i \Delta Y_i|}{|\hat{Y}_i \cup Y_i|} \tag{4}$$

Subset accuracy (SA), Eq. 5, is a strict metric which evaluates the number of instances where the set of predicted labels exactly matches the set of true labels. Note that $[\![\pi]\!]$ returns 1 if predicate π is true, and 0 otherwise.

$$\uparrow \text{SA} = \frac{1}{m}\sum_{i=1}^{m}[\![\hat{Y}_i = Y_i]\!] \tag{5}$$

Also, we use different versions of the FMeasure, namely Example-based FMeasure (ExF; Eq. 2), Micro-averaged FMeasure (MiF; Eq. 6), and Macro-averaged FMeasure (MaF; Eq. 3). While ExF captures the relationship among the labels in its calculation, MiF and MaF give different weight to the labels; MiF is biased by more frequent labels, while MaF gives the same importance to all of them.

$$\uparrow \text{MiF} = \frac{\sum_{l=1}^{q} 2 \cdot tp_l}{\sum_{l=1}^{q} 2 \cdot tp_l + \sum_{l=1}^{q} fp_l + \sum_{l=1}^{q} fn_l} \tag{6}$$

[1]http://www.uco.es/kdis/mllresources

[2]In order to execute it in reasonable time, a random selection of instances of NusWide cVLAD+ dataset was performed.

C. Methods and configurations

G3P-kEMLC has been built using JCLEC [28], Mulan [29], and Weka [30] libraries, and the code is publicly available in a GitHub repository[3].

In the experiments, we use two different configurations of G3P-kEMLC: I) with a fixed value of $k = 3$ for all base classifiers; and II) with a variable value of k for each of them. For the second configuration, the size of the k-labelset of each multi-label classifier is in the range $[3, q/2]$. In this way, we observe how the method works both using fixed k value as in RAkEL, and also by considering the relationships among a large number of labels. Regarding the size of the pool of classifiers, Eq. 7 indicates how to calculate the number of classifiers needed to have, on average, $\overline{v}$ votes for each label in the pool. For n we also use two different configurations: I) n for $\overline{v} = 10$; and II) n for $\overline{v} = 20$. Note that in our approach, the fact of using a large number of base classifiers in the pool does not mean that all of them will be included in the ensemble, since the G3P algorithm selects the most suitable models. For each dataset, the results of the best of these two configurations are reported; the best configuration is the one with better ranking among all five evaluation metrics.

$$n = \frac{\overline{v}}{(k_{min} + k_{max})/2} \cdot q \tag{7}$$

Concerning the size of the tree, we use the maximum number of children at each node $mC = 7$, while the maximum depth is fixed to $mD = 3$ for most cases. However, when using fixed $k = 3$, for those datasets with more than 50 labels, $mD = 4$ is used. Note that for example for NusWide dataset, which has 81 labels, the size of the pool for $\overline{v} = 20$ is $n = 540$, while a tree of $mD = 3$ and $mC = 7$ can have, as most, $7^3 = 343$ leaves, so it could not include all classifiers in the pool if it was necessary. Using $mD = 4$, more complex trees including at most $7^4 = 2401$ leaves can be created, which is enough for these cases.

For the selection of the rest of parameters of the algorithm, we performed a brief preliminary study. Due to space constraints, additional material including this study is available at the KDIS Group Webpage[4]. We use $popSize = 50$ individuals; $n_g = 150$ generations; probabilities $p_c = 0.7$ and $p_m = 0.2$ for genetic operators; $\beta = 0.5$ for the fitness function; and 75% of instances are sampled without replacement at each multi-label classifier. Finally, not only in G3P-kEMLC but also in other EMLCs, the C4.5 [31] decision tree algorithm is used as a single-label classifier (except for RF-PCT, which uses PCTs).

A comparison with RAkEL, which also uses subsets of k labels at each base classifier, shall demonstrate whether the fact of evolving an optimal structure for the ensemble instead of just giving the same weight to each classifier improves (or not) predictive performance. The recommended configuration for RAkEL is with $k = 3$ and $n = 2q$, so that each label has, on average, 6 votes. However, to perform a fair comparison, RAkEL was executed with $n = 2q$ (so $\overline{v} = 6$, as recommended), $n = 3.33q$ (meaning $\overline{v} = 10$, as in other EMLCs), and also with n calculated in such a way that $\overline{v}$ is the same as in the best configuration of G3P-kEMLC; we report the results of the best configuration. In this way, we aim to show that the performance of our method is not only biased by the number of base classifiers.

A second comparison involving state-of-the-art EMLCs was also performed. The best EMLCs according to the study in [7] were selected for the comparison. For all EMLCs, the default parameters proposed by their authors were used. EBR, ECC and RF-PCT use sampling with replacement of the original training dataset at each member, while EPS uses samples without replacement of 66% of the instances. Note that for CAL500 dataset, which has as many different labelsets as instances, EPS was executed without pruning the infrequent labelsets. Finally, all of them use $n = 10$.

In all cases, the datasets were partitioned using random 5-fold cross-validation, and all methods were executed using 6 different seeds for random numbers. Finally, the results were averaged over 30 runs.

V. Results and Discussion

In this section, the results of the experimental study are presented and discussed. First, analysis and comparison of G3P-kEMLC and RAkEL is performed; then, G3P-kEMLC is compared to other state-of-the-art EMLCs. Hereafter, the two versions of our proposed method are indicated as G3P-kEMLC-3 when $k = 3$ is used, and as G3P-kEMLC-V when a variable value of k in the range $[3, q/2]$ is used.

A study of the efficiency of G3P-kEMLC was also carried out; however, due to space constraints, the results of this study are available in the additional material, as well as detailed results of the experiments and also p-values of statistical tests.

A. G3P-kEMLC vs RAkEL

In this section, we compare the performance of G3P-kEMLC and RAkEL, as methods that use base classifiers focused on k-labelsets. In Table II, the average ranking values for each metric, computed for all datasets, are shown. For each pair dataset-metric, the method that performs best obtains a ranking of 1, the next a ranking of 2, and so on; the lower the value the better. We can see that in four of the metrics G3P-kEMLC-V has the best average ranking, while G3P-kEMLC-3 is the best in one. Further, both methods are ahead of RAkEL in all cases except in SA, where RAkEL has a better average ranking than G3P-kEMLC-3.

In order to determine if significant differences exist among algorithms, Friedman's [32] and Holm's [33] tests were performed, using a confidence value $\alpha = 0.05$. Friedman's test has shown that significant differences existed for all metrics except for SA. Further, Holm's test has shown that for the rest of the metrics, the performance of both configurations of G3P-kEMLC is statistically the same, and both are significantly better than RAkEL. Fig. 5 shows the critical diagrams for

[3]https://www.github.com/kdis-lab/G3P-kEMLC

[4]http://www.uco.es/kdis/G3P-kEMLC/

TABLE II
AVERAGE RANKINGS IN THE COMPARISON AMONG G3P-KEMLC AND RA*k*EL.

	G3P-kEMLC-3	G3P-kEMLC-V	RA*k*EL
AHL	1.72	**1.47**	2.81
SA	2.19	**1.81**	2.00
ExF	1.72	**1.47**	2.81
MiF	1.76	**1.65**	2.59
MaF	**1.53**	1.78	2.69

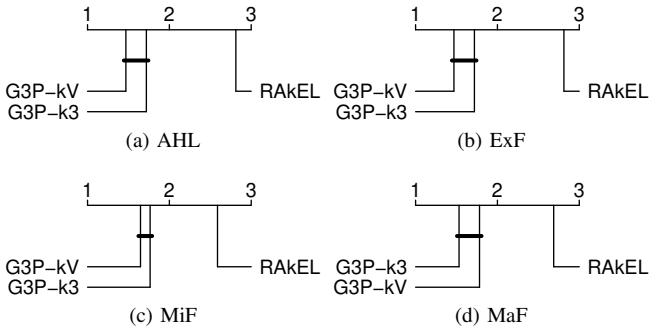


Fig. 5. Critical diagrams of Holm's test at 95% confidence for the comparison between G3P-kEMLC and RA*k*EL. G3P-kEMLC-3 and G3P-kEMLC-V are indicated as G3P-k3 and G3P-kV, respectively.

the four metrics comparing G3P-kEMLC with RA*k*EL; in these diagrams, a line linking two methods indicates that their performance is statistically the same at 95% confidence.

Fig. 6 shows the average number of votes per label, $\overline{v}$, of the best configuration in each case. Note that, although RA*k*EL was executed with a large number of classifiers (such as n calculated for $\overline{v} = 10$ and for $\overline{v}$ equal to the best configuration of G3P-kEMLC), in most cases it performed better just with 6 votes on average for each label. We see that G3P-kEMLC is able to model and adjust the number of classifiers depending on each dataset, and is flexible to adapt to each specific case. In many cases, G3P-kEMLC obtained better results than RA*k*EL using less classifiers. Therefore, we also show that the number of classifiers used in the ensemble is not the only characteristic that contributes for our method to achieve good performance, but the selection of classifiers into an optimal tree-shaped ensemble structure is decisive for its performance.

B. G3P-kEMLC vs state-of-the-art

Once we have shown that the performance of G3P-kEMLC is significantly better than RA*k*EL, we next compare it with other state-of-the-art EMLCs. As in the previous experiment, Table III shows the average ranking values of each method on all datasets. In it, G3P-kEMLC-3 and G3P-kEMLC-V are indicated as G3P-3 and G3P-V respectively. We see that both G3P-kEMLC configurations always have better average ranking than the rest of EMLCs, except for SA, where EPS is better ranked.

Friedman's test determined that there existed significant differences in the performance of the algorithms for all metrics. Fig. 7 shows the critical diagrams of the Holm's test for all metrics. For AHL, ExF, and MiF, the performance of both G3P-kEMLC configurations is significantly better than all other methods. For MaF, the performance of G3P-kEMLC-V is statistically comparable to EBR, while for SA, RF-PCT is the only method that perform significantly worse than both EPS and G3P-kEMLC-V.

Therefore, we demonstrate that given the optimal structure of the ensemble that G3P-kEMLC obtains, independently of the configuration used, it outperformed state-of-the-art EMLCs.

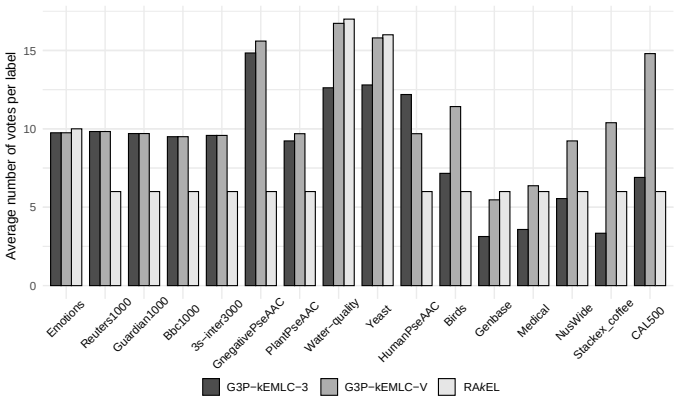


Fig. 6. Average number of votes per label in each dataset.

TABLE III
AVERAGE RANKINGS IN THE COMPARISON AMONG G3P-KEMLC AND STATE-OF-THE-ART EMLCS.

	G3P-3	G3P-V	ECC	EBR	EPS	RF-PCT
AHL	1.81	**1.44**	4.59	4.69	3.81	4.66
SA	3.22	2.88	3.22	4.09	**2.66**	4.94
ExF	2.00	**1.63**	4.41	4.34	3.88	4.75
MiF	1.75	**1.50**	4.72	4.16	4.25	4.63
MaF	**1.47**	1.97	4.41	3.75	4.50	4.91

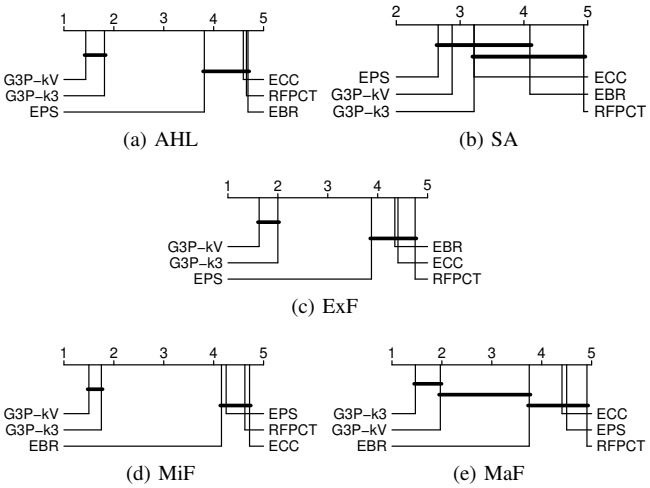


Fig. 7. Critical diagrams of Holm's test at 95% confidence for the comparison between G3P-kEMLC and state-of-the-art EMLCs. G3P-kEMLC-3 and G3P-kEMLC-V are indicated as G3P-k3 and G3P-kV, respectively.

VI. Conclusions

In this paper, we introduced a method based on G3P for building EMLCs. Given a pool of multi-label classifiers, each focused on a subset of k labels, where k could be different for each, the algorithm evolves individuals representing a tree-shaped ensemble. At each node of the ensemble, predictions of children nodes are combined, while the leaves represent any multi-label classifier of the pool. The experimental study on 16 multi-label dataset using 5 evaluation metrics yielded promising results, demonstrating that the fact of building a tree-shaped ensemble where not all members have the same importance in the final prediction not only outperformed RAkEL (which also is focused on small subsets of the labels) but also the other state-of-the-art EMLCs, obtaining the model in acceptable time. Moreover, the proposed method is flexible and able to adapt the number of members of the ensemble according to each specific case.

In the future, we will explore other types of non-terminal nodes to combine the predictions, which also may use different types or number of child nodes. Further, we will develop a heuristic or evolutionary process to get a pool of multi-label classifiers that are accurate, diverse, and have an optimal size of the k-labelsets.

Acknowledgment

This research was supported by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund, project TIN2017-83445-P. This research was also supported by the Spanish Ministry of Education under FPU Grant FPU15/02948.

References

[1] E. Gibaja and S. Ventura, "Multi-label learning: a review of the state of the art and ongoing research," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 6, pp. 411–444, 2014.

[2] H. Shao, G. Li, G. Liu, and Y. Wang, "Symptom selection for multi-label data of inquiry diagnosis in traditional chinese medicine," *Science China Information Sciences*, vol. 56, no. 5, pp. 1–13, 2013.

[3] J. Xu, "Fast multi-label core vector machine," *Pattern Recognition*, vol. 46, no. 3, pp. 885–898, 2013.

[4] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *Proceedings of the ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, 2008, pp. 53–59.

[5] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, pp. 335–359, 2011.

[6] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multi-label classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[7] J. M. Moyano, E. L. Gibaja, K. J. Cios, and S. Ventura, "Review of ensembles of multi-label classifiers: Models, experimental study and prospects," *Information Fusion*, vol. 44, pp. 33 – 45, 2018.

[8] G. Tsoumakas, I. Partalas, and I. Vlahavas, "A taxonomy and short review of ensemble selection," in *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*, 2008, pp. 1–6.

[9] Y. Bi, "The impact of diversity on the accuracy of evidential classifier ensembles," *International Journal of Approximate Reasoning*, vol. 53, no. 4, pp. 584 – 607, 2012.

[10] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 2, pp. 121–144, 2009.

[11] O. Gharroudi, H. Elghazel, and A. Aussem, "Ensemble Multi-label Classification: A Comparative Study on Threshold Selection and Voting Methods," in *IEEE International Conference on Tools with Artificial Intelligence,*, 2015, pp. 377–384.

[12] G. Tsoumakas, I. Katakis, and I. Vlahavas, *Data Mining and Knowledge Discovery Handbook, Part 6*. Springer, 2010, ch. Mining Multi-label Data, pp. 667–685.

[13] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 995–1000.

[14] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[15] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Ensembles of multi-objective decision trees," in *European conference on machine learning*, 2007, pp. 624–631.

[16] H. Blockeel, L. D. Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 55–63.

[17] R. I. Mckay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'neill, "Grammar-based genetic programming: a survey," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 365–396, 2010.

[18] A. Tsakonas, G. Dounias, M. Doumpos, and C. Zopounidis, "Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming," *Expert Systems with Applications*, vol. 30, no. 3, pp. 449–461, 2006.

[19] A. Zafra and S. Ventura, "Multi-instance genetic programming for predicting student performance in web based educational environments," *Applied Soft Computing*, vol. 12, no. 8, pp. 2693–2706, 2012.

[20] J. M. Luna, J. R. Romero, C. Romero, and S. Ventura, "On the use of genetic programming for mining comprehensible rules in subgroup discovery," *IEEE transactions on cybernetics*, vol. 44, no. 12, pp. 2329–2341, 2014.

[21] A. Tahmassebi and A. H. Gandomi, "Genetic programming based on error decomposition: A big data approach," in *Genetic Programming Theory and Practice XV*. Springer, 2018, pp. 135–147.

[22] A. Cano, A. Zafra, E. L. Gibaja, and S. Ventura, "A grammar-guided genetic programming algorithm for multi-label classification," in *European Conference on Genetic Programming*. Springer, 2013, pp. 217–228.

[23] A. G. C. de Sá, A. A. Freitas, and G. L. Pappa, "Automated selection and configuration of multi-label classification algorithms with grammar-based genetic programming," in *Parallel Problem Solving from Nature – PPSN XV*, 2018, pp. 308–320.

[24] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information sciences*, vol. 250, pp. 113–141, 2013.

[25] E. Gibaja and S. Ventura, "A tutorial on multilabel learning," *ACM Computing Surveys*, vol. 47, no. 3, 2015.

[26] J. M. Moyano, E. L. Gibaja, and S. Ventura, "MLDA: A tool for analyzing multi-label datasets," *Knowledge-Based Systems*, vol. 121, pp. 1–3, 2017.

[27] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, "Mlenn: A first approach to heuristic multilabel undersampling," in *Intelligent Data Engineering and Automated Learning – IDEAL 2014*, 2014, pp. 1–9.

[28] S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervás, "JCLEC: a Java framework for evolutionary computation," *Soft Computing*, vol. 12, no. 4, pp. 381–392, 2008.

[29] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A Java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, pp. 2411–2414, 2011.

[30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[31] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[32] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, no. 1, pp. 86–92, 03 1940.

[33] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.