Information Fusion 78 (2022) 1–19

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full length article

Auto-adaptive Grammar-Guided Genetic Programming algorithm to build Ensembles of Multi-Label Classifiers

Jose M. Moyano, Sebastián Ventura *

Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Córdoba, 14071, Córdoba, Spain
Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, 14071, Spain
Knowledge Discovery and Intelligent Systems in Biomedicine Laboratory, Maimonides Biomedical Research Institute of Córdoba, 14004, Spain

ARTICLE INFO

Keywords:
Multi-label classification
Ensemble learning
Evolutionary algorithm
Grammar-guided genetic programming

ABSTRACT

Multi-label classification has been used to solve a wide range of problems where each example in the dataset may be related either to one class (as in traditional classification problems) or to several class labels at the same time. Many ensemble-based approaches have been proposed in the literature, aiming to improve the performance of traditional multi-label classification algorithms. However, most of them do not consider the data characteristics to build the ensemble, and those that consider them need to tune many parameters to maximize their performance.

In this paper, we propose an Auto-adaptive algorithm based on Grammar-Guided Genetic Programming to generate Ensembles of Multi-Label Classifiers based on projections of k labels (AG3P-kEMLC). It creates a tree-shaped ensemble, where each leaf is a multi-label classifier focused on a subset of k labels. Unlike other methods in the literature, our proposal can deal with different values of k in the same ensemble, instead of fixing one specific value. It also includes an auto-adaptive process to reduce the number of hyper-parameters to tune, prevent overfitting and reduce the runtime required to execute it. Three versions of the algorithm are proposed. The first, *fixed*, uses the same value of k for all multi-label classifiers in the ensemble. The remaining two deal with different k values in the ensemble: *uniform* gives the same probability to choose each available value of k, and *gaussian* favors the selection of smaller values of k.

The experimental study carried out considering twenty reference datasets and five evaluation metrics, compared with eleven ensemble methods demonstrates that our proposal performs significantly better than the state-of-the-art methods.

1. Introduction

Classification is a machine learning task which aims to build a model able to predict one of the predefined categorical classes for a given input instance [1]. However, a wide range of real-world problems do not fit the restrictions of traditional classification, where each instance is associated to only one class. Examples of such problems are medical diagnosis (where each patient may have more than one disease) [2,3], image annotation (an image could be labeled with more than one item appearing in it) [4,5] and emotions detection (a person could be feeling more than one emotion at the same time) [6,7]. The Multi-Label Classification (MLC) paradigm appeared to solve classification problems where each instance may be associated to more than one class label simultaneously, and its attention has increased in the last decade [8,9].

Dealing with objects that may be labeled with more than one class provides the ability to solve a larger number of problems; however, some new challenges to be addressed show up. The output labels tend to be correlated among themselves, some of them appearing more frequently together than with others. For example, in image categorization, the labels *baby* and *toy* would be highly related, while the labels *baby* and *crocodile* would also be highly indirectly correlated. On the other hand, the labels *baby* and *sky* may not present any correlation, so they would not influence each other's modeling. Besides, labels do not appear with the same frequency in the dataset, so multi-label datasets tend to have a very imbalanced output space. Finally, the high dimensionality of the label space, where an output for all labels should be given, would make the problem far more difficult to handle. Several studies have demonstrated that by tackling these problems or challenges, the performance of the multi-label methods is improved [10–13].

The multi-label classification problem has been tackled from many different perspectives in the literature [14], but methods that are

* Corresponding author at: Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, 14071, Spain.
E-mail address: sventura@uco.es (S. Ventura).

https://doi.org/10.1016/j.inffus.2021.07.005
Received 19 October 2020; Received in revised form 25 June 2021; Accepted 25 July 2021
Available online 16 August 2021

based on ensembles (i.e., combination of several learners or classifiers) tend to outperform simpler methods, demonstrating very good performance [12,13,15,16]. Ensembles of Multi-Label Classifiers (EMLCs) combine the predictions of several multi-label classifiers to give their final prediction. Ensemble learners have been applied not only to classification but to a plethora of problems, and its application grows over the years [17]. However, the selection of base members (i.e., internal classifiers in the ensemble) is not trivial. Although formal dependency of this proof does not exist, many studies have stated that ensemble learners should include accurate and diverse members [18]. This should improve the generalization ability of each of the base members separately.

Despite the good performance of the EMLCs proposed in the literature, some of them do not consider the characteristics of the data to build the ensemble, but just generate diversity in the ensemble by following random procedures [10]. Two recent studies proposed to build EMLCs by using Evolutionary Algorithms (EAs) while considering the characteristics of the multi-labeled data, outperforming the rest of the methods [10,11]. However, these EAs require the configuration of a wide range of hyper-parameters that should be tuned to maximize their performance. Besides, the structure of the ensemble is fixed, being less capable to adapt each specific problem. Thus, the main aim of this paper is to overcome the drawbacks of previous approaches to build EMLCs. We present an algorithm for the optimization of a novel ensemble structure, with a method that reduces the number of hyper-parameters to tune.

In this paper, we propose a Grammar-Guided Genetic Programming (G3P) method to generate EMLCs. In contrast to most ensemble methods, where the final prediction is given by the majority voting of all classifiers [10–13,16], this method builds a tree-shaped ensemble, where each leaf is a multi-label classifier and each internal node is a combiner of the predictions of children nodes. In this way, classifiers that are deeper in the tree have a lower influence in the final prediction than shallower classifiers. Thus, the algorithm is flexible to adapt the structure of the tree to each specific problem, using different number of classifiers and combining them in an structure that maximizes its performance. This method also reduces the parameters to be tuned to the minimum, including an auto-adaptive procedure to modify the crossover and mutation probabilities during the execution, as well as a stop criterion based on the number of generations without improvement of the best individual.

The base members of our method are focused on predicting a subset of k labels, also known as k-labelset. Therefore, each member is able to model the dependencies among labels with a low complexity. Unlike other methods in the literature [10,11,13], it may include classifiers using different value of k in the ensemble, thus modeling dependencies among subsets of labels of different size at the same time. Thus, the user does not need to fix a value of k, which could not be optimal for the problem at hand. The experimental study carried out over 20 datasets and using 5 evaluation metrics demonstrates that our proposal obtains significantly better performance than state-of-the-art EMLCs.

It is worth noting that a previous version of this method was proposed in [19]. The main contributions of this paper in comparison to the previous one are following described. In order to prevent from overfitting, a stop criterion based on the non-improvement of the best individual has been defined. It reduces the required runtime to execute the algorithm, and removes the need to set a fixed value for the maximum number of generations (see Section 3.1). We also present two versions of the method with a variable value of k (*uniform* and *gaussian*). The former gives the same probability to all sizes of k, while the latter biases the search by smaller k-labelsets. The *gaussian* mode looks for a better trade-off between simpler models, avoiding including unnecessary noise between labels, and more complex models that can model the relationships among a greater number of labels (see Section 3.2). It is also noteworthy that the k-labelsets are not generated randomly as in the conference paper; it considers the relationship among labels and their frequency in the initial pool. Thus, it favors the selection of more related subsets of labels, but not setting aside those labels that are independent or infrequent (see Section 3.2). Besides, it ensures that all labels appear in the initial population (see Section 3.2).

An auto-adaptive process for the crossover and mutation operators is also defined. It allows the algorithm to automatically adapt each specific problem by increasing/decreasing the probabilities of crossover and mutation operators, thus adapting the exploration or exploitation of the evolutionary algorithm. Besides, it reduces the need to fix probabilities for these operators, which may differ greatly among problems (see Section 3.6). We have carried out an extensive study and analysis of the proposed methods to better understand its operation and performance. It includes analyzing the initial pool of classifiers, the convergence of the algorithms, the auto-adaptive process, and the size of the final EMLCs. A comparative study in terms of predictive performance and runtime has been also carried out (see Section 5.1). Finally, the experimental study has been extended, by including up to 20 multi-label datasets and comparing with 11 state-of-the-art EMLCs. An analysis of the runtime of the proposed methods versus the state-of-the-art ensembles has been also included (see Sections 4 and 5.2).

The rest of the paper is organized as follows: Section 2 includes related work and background in multi-label classification and G3P; Section 3 describes our auto-adaptive G3P method to build EMLCs; Section 4 presents the experimental studies carried out and their configuration, where 11 state-of-the-art EMLCs, 20 datasets, and 5 evaluation metrics are used; Section 5 analyzes the performance of our proposed method and discusses the results compared with the state-of-the-art in MLC; finally Section 6 ends with the conclusions attained from this work.

2. Related work

In this section, we first give a formal definition of MLC and present the state-of-the-art methods in MLC, and then we introduce the G3P paradigm.

2.1. Multi-label classification

Let $\mathcal{X} = X_1 \times \cdots \times X_d$ be a d-dimensional feature or input space, and $\mathcal{Y} = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$ the label or output space composed by $q > 1$ labels. A multi-label dataset $\mathcal{D}$ is composed by a set of m instances, and it is defined as $\mathcal{D} = \{(\mathbf{x}_i, Y_i) | 1 \leq i \leq m\}$. Each instance of $\mathcal{D}$ is composed by an input vector $\mathbf{x}$ and a set of relevant labels associated with it $Y \subseteq \mathcal{Y}$. Note that each different Y is also called labelset [14]. The goal of MLC is to build a predictive model $h : \mathcal{X} \to 2^{\mathcal{Y}}$ to provide a bipartition $\hat{b} = \left(\hat{Y}, \hat{Z}\right)$ of the output space for an unknown instance, distinguishing between relevant $(\hat{Y})$ and irrelevant $(\hat{Z})$ labels.

MLC methods are categorized into three main groups: problem transformation methods, algorithm adaptation methods, and EMLCs [8,15,20]. Problem transformation methods transform the multi-labeled data into one or several multi-class problems, then using traditional classification methods. Algorithm adaptation methods adapt the classification algorithms to directly handle with the multi-label data without the need of transforming it. Finally, EMLCs are defined as a set of n multi-label classifiers, either problem transformation or algorithm adaptation methods. Each of the classifiers provides predictions for all or part of the labels, and they are combined following any kind of aggregation rule [21]. In the following paragraphs, state-of-the-art EMLCs are described in more detail, and compared to our proposed ensemble schema. A wider description of problem transformation and algorithm adaptation methods can be found in [14].

Ensemble of Binary Relevance (EBR) [12] combines the predictions of n Binary Relevance (BR) models, each built over a random subset of the instances. Each BR builds q independent binary models, one for each label. So, although being simple and highly parallelizable, it is not able to model the dependencies among labels, which would harm its performance. Also, the fact of building a different model for each label may entail a high complexity in cases with a high number of labels. Similar to EBR, Ensemble of Classifier Chains (ECC) [12] relies its performance on binary classifiers. In this case they form a chain, so their input features are extended by including the predictions of previous labels in the chain, being able to model some of the dependencies among labels. In addition to using random subsets of the data for each base classifier, members also use a random chain ordering to increase diversity. Multi-Label Stacking (MLS) [22] was also proposed to overcome BR's drawbacks. It involves applying BR twice; first a BR model is built, and then, a second BR is built but including previous predictions of labels as extra input feature. Thus, it can model the relationship among labels via its stacking procedure.

Dynamic selection and Circulating Combination-based Clustering (D3C) [23] builds independent binary classifiers of different type for each label, and then uses clustering and dynamic selection to select a subset of accurate and diverse classifiers to combine their predictions. In this way, D3C is able to automatically select the classifiers that best fit to the problem in each case. Random Forest of Predictive Clustering Trees (RF-PCT) [24] generates an ensemble of Predictive Clustering Trees (PCTs). Each base classifier uses a random sample of the instances, and at each node of the tree, the data is partitioned into smaller clusters, dealing with small subsets of data at each node.

Ensemble of Pruned Sets (EPS) [16] combines the predictions of n Pruned Sets (PS), each built using different subsets of the data. PS is based on the Label Powerset (LP) approach, i.e., it transforms the multi-label data into multi-class, where each different labelset is considered as a different class [25]. Thus, it is able to deal with the relationship among labels, but the number of possible classes grows exponentially with the number of labels, which may generate extremely complex and imbalanced problems. In order to reduce the imbalance of the multi-class problem, PS substitutes the infrequent labelsets by subsets of labels that are more frequent in the dataset.

RAndom k-labELset (RAkEL) [13] is also based on the LP approach, but each of the base members is focused only on a small subset of k labels, also known as k-labelset. In this way, RAkEL is able to model the dependencies among groups of labels, but drastically reducing the imbalance and complexity of the output space that LP generates. Our proposal also uses subsets of k labels to model the dependencies among labels, but the selection of such k-labelsets is guided by the correlation among labels, and not just randomly, as RAkEL does. Besides, unlike RAkEL, ours can include classifiers using different values of k, modeling subsets of labels of different sizes.

A tree-shaped ensemble, Hierarchy Of Multi-label classifiERs (HOMER) is proposed in [26]. It transform the multi-label problem into a hierarchy of simpler models. In nodes with more than one label, similar labels are distributed together in sub-groups to improve the performance of subsequent models. Although our proposal and HOMER both use a tree structure, there are many structural differences among them. HOMER uses all labels at the root node, and it splits the label set into c children nodes according to a clustering algorithm, being c a predefined value. In our proposal, the number of children varies at each node. In HOMER the split is done until each leaf contains only one label, while in ours, leaves contain classifiers considering k labels each. In HOMER's prediction phase, each node passes the instances up to the children nodes, being the final output given by the leaves. In our method, the instances are given to the leaf nodes and they are passed up to the root node, which makes the final prediction. While in HOMER all the nodes contain trained classifiers, in our method only leaves contain classifiers, and internal nodes are just combiner nodes. We also propose a genetic programming algorithm that optimizes the ensemble structure for the problem at hand.

Most of the previously described ensemble methods construct the EMLCs by building base members separately and then joining all of them using a combination schema without considering their joint performance. D3C considers the performance of base members by dynamically selecting a subset of the classifiers for the ensemble; however, it focuses on building binary classifiers, thus not being able to learn from the dependencies among labels. So, contrary to most state-of-the-art methods, our proposal finds a promising ensemble structure by evaluating the performance of the whole ensemble, which is created by selecting a subset of previously built base classifiers. Besides, unlike D3C, our proposal benefits from learning the dependencies among groups of labels.

Recently, some evolutionary approaches have been proposed to select the ensemble structure in MLC. Evolutionary Multi-label Ensemble (EME) [10] proposes an evolutionary algorithm where each individual encodes an entire ensemble. Although the resulting ensemble is similar to the one of RAkEL, EME considers the characteristics of the data to guide the building process. Evolutionary AlGorithm for multi-Label Ensemble opTimization (EAGLET) [11], on the other hand, encodes base members of the ensemble in separate individuals, then combining accurate and diverse individuals from the population into the ensemble. Both EME and EAGLET are designed to use a fixed value of k for all classifiers, and they also need to tune a wide range of parameters to optimize their performance.

As EME and EAGLET, our proposal is based on learning base models over subsets of k labels. However, they use a fixed value of k for the entire ensemble, while our proposal may include base members with different values of k. EME and EAGLET fix an ensemble schema at the beginning of the execution, i.e., they use base members with $k = 3$ labels, $2q$ (EME) or $3.33q$ (EAGLET) classifiers in the ensemble, and a simple voting schema for prediction. In contrast, our proposal can deal with variable values of k in the ensemble, and the structure is adaptively selected during the evolution. Instead of choosing a fixed number of classifiers before executing, the algorithm chooses the structure and size of the tree-shaped ensembles to maximize its performance in each case. Besides, instead of giving the same weight in the final prediction to all members, the importance of each classifier in the final prediction depends on their depth in the tree as well as on the number of children of each combiner node.

Our proposal, AG3P-kEMLC, includes mechanisms to auto-adapt some of the hyper-parameters of the evolutionary process (such as the number of generations and probabilities for crossover and mutation operators). It not only reduces the number of parameters that the user needs to set, but also makes it easier to automatically adapt to each specific problem. In EME and EAGLET, these parameters, which completely bias the final performance of the method, need to be set before the execution. Thus, the auto-adaptive process of our proposal provides it with greater generalization ability across a wide range of problems. Regarding the evaluation process, EME needs to build the whole ensemble to be able to give a prediction for the evaluation data set and thus calculate the corresponding fitness function. EAGLET reduces the complexity of EME since it needs to build a single classifier to evaluate each individual. Although both of them include mechanisms to avoid building the same classifier twice, the high amount of different classifiers during the evolution make them computationally complex. On the other hand, our proposal creates a pool of classifiers at the beginning, and they do not change during the evolution. Thereby, it does not need to re-build them to evaluate the individuals, but only to combine previously obtained predictions from classifiers in the pool, making the process much less complex. A summary of this comparison is presented in Table 1.

2.2. Grammar-Guided Genetic Programming

Genetic Programming (GP) [27] is a technique based on EAs, being the representation of individuals their main characteristic and difference compared to other evolutionary techniques [28,29]. In GP, individuals are encoded as variable-length hierarchical structures, where

Table 1
Comparison between AG3P-kEMLC, and state-of-the-art evolutionary methods to build EMLCs.

	EME [10]	EAGLET [11]	AG3P-kEMLC
Size of the subsets of labels	Fixed	Fixed	Fixed or variable
Ensemble structure	Simple voting	Simple voting	Tree-shaped with voting on each internal node
Ensemble size	Fixed	Fixed	Adaptive
Weight of classifiers in final prediction	Same for all classifiers	Same for all classifiers	Depends on the structure
Selection of hyper-parameters	Needs previous optimization	Needs previous optimization	Adaptive during execution
Cost of evaluation	Needs to build and evaluate whole ensemble (high complexity)	Needs to build and evaluate a single classifier (medium complexity)	Only combines predictions, classifiers are not re-built (low complexity)

the shape or size of the trees may not be constrained a priori, but they evolve towards optimal structures that best fit the problem. As other EAs, individuals in the population are proposed to be recombined and mutated in each generation, and the population for the next generation is chosen taking into account both the parent and children individuals, considering the fitness of individuals to guide the evolution; this process is performed until a stop criterion is fulfilled, e.g., a maximum number of generations.

Grammar-Guided Genetic Programming (G3P) is an extension of GP, which enables the use of a grammar to constraint the GP process [30, 31]. A context-free grammar [32] is a 4-tuple (V, T, R, S), where:

- V is a finite set of *variables*,
- T is a finite set of *terminals*, disjoint from V,
- R is a finite set of *rules*, each comprising a variable in the antecedent, and a consequent that may be composed of variables and terminals,
- $S \in V$ is the start variable.

Context-free grammars are generally used to create strings following derivations. Given a start variable, the derivation begins by looking for a rule with the start variable in the antecedent (usually, the first rule). Then, for each variable in the consequent, it finds a rule that starts with this variable, and replaces it with the consequent of the rule. This step is repeated until all variables are replaced, so the final string only contains terminal symbols.

In G3P, each individual represents a solution that is derived from the grammar by applying productions up to leaf nodes. This grammar enables the GP process to ensure certain constraints at each node, such as the type or number of children nodes, also ensuring that generated individuals are feasible, i.e., represent a valid solution. The internal nodes of the tree are called non-terminal nodes and they correspond to functions taking children as arguments, while the leaves are terminal nodes, corresponding to variables of the problem or constant values.

G3P has been successfully applied to classification problems [33]; however, the related work of both GP and G3P in MLC is scarce. A G3P algorithm is proposed in [34] to build a rule-based multi-label classifier from scratch. Recently, some works in Automated Machine Learning (AutoML) have been proposed for multi-label classification problems, where a G3P-based procedure is used to choose the most suitable multi-label classification algorithm [35,36]. The AutoML approach and our proposal have similarities: both use a G3P algorithm as an optimization process, looking for an individual which maximizes a fitness function based on a combination of multi-label evaluation measures. However, while the AutoML approach is centered on looking for the best method from a pre-defined set of classifiers, ours looks for a novel ensemble structure that could not be obtained from the AutoML approach. Besides, both AutoML and our approach can co-exist without a problem, and AutoML methods could incorporate our proposal as an option of multi-label classifier in the future.

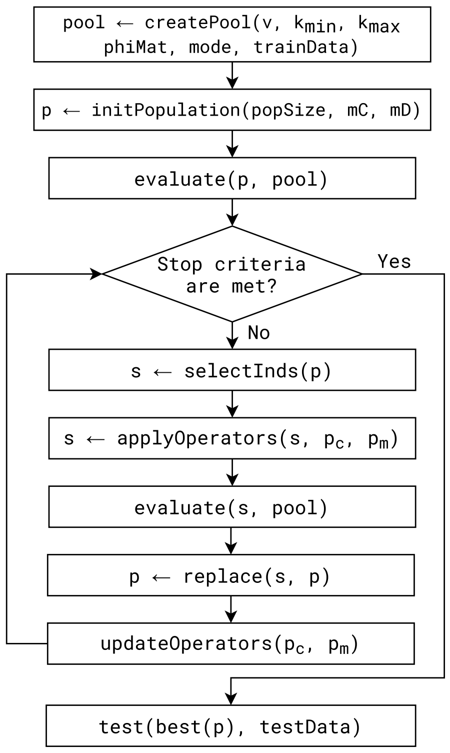
pool ← createPool(v, k_min, k_max phiMat, mode, trainData)
p ← initPopulation(popSize, mC, mD)
evaluate(p, pool)
Stop criteria are met?
Yes
No
s ← selectInds(p)
s ← applyOperators(s, p_c, p_m)
evaluate(s, pool)
p ← replace(s, p)
updateOperators(p_c, p_m)
test(best(p), testData)

Fig. 1. Main flow of AG3P-kEMLC.

3. AG3P-kEMLC

In this section, we present our proposed Auto-adaptive algorithm based on Grammar-Guided Genetic Programming to generate Ensembles of Multi-Label Classifiers based on projections of k labels (AG3P-kEMLC). First, we present the algorithmic strategy. Then, it is presented how the initial pool of classifiers is created, as well as the encoding of individuals. Later, the fitness function and evaluation process is described. Finally, the genetic operators are presented, as well as the auto-adaptability procedures for the hyper-parameters.

3.1. Algorithmic strategy

The main flow of the AG3P-kEMLC algorithm is presented in Fig. 1. First, a pool of classifiers, each of them focused on $k \in [k_{min}, k_{max}]$ labels is created. Let $\overline{v}$ be the average number of votes per label expected in the pool, *phiMat* the matrix with coefficients ϕ measuring the relationship among labels, *mode* the way in which the value of k is selected for each classifier, and *trainData* the training dataset used to build and evaluate the models (see Section 3.2).

Then, the initial population of *popSize* individuals, each of them representing an EMLC, is created by following the grammar, being *mC* the maximum number of children at each node, and mD the maximum depth of the tree (see Section 3.3). The population is evaluated (see Section 3.4), and until the stop criteria are met, individuals are selected, the crossover and mutation operators are applied with probabilities p_c and p_m respectively (see Section 3.5), and the new individuals are evaluated. Individuals are replaced maintaining elitism, i.e., the best parent is maintained if it is better than all new individuals, and the probabilities of applying crossover and mutation operators are updated in the auto-adaptive process (see Section 3.6).

Once any of the stop criteria is met, the best individual, i.e., the best EMLC, is returned, being the one tested with test data. The algorithm receives as stop criterion a maximum number of generations; however, it also has a stopping criterion based on the improvement of the best individual. Given that previous similar approaches needed between 50 and 100 generations in total [11] we considered that 10 is a sufficient number of generations to consider that the evolution is stuck. If the best individual did not improve in some generations, the evolutionary process finishes and the best ensemble is returned and tested later. This not only reduces the need of tuning the maximum number of generations for each specific case, but also avoids overfitting and reduces the required runtime.

Unlike other evolutionary approaches to build EMLCs (i.e., EME and EAGLET), the base classifiers are built at the beginning and remain unchanged during the evolution. Since the classifiers do not need to be re-built for the evaluation process, the complexity of the method is drastically reduced. The ensemble schema is not a simple and fixed voting process, but an adaptive tree-structure is created. The structure depends on each scenario and the algorithm select the one that achieves the best performance. In addition, AG3P-kEMLC gets rid of the need to select many of the hyper-parameters, such as the number of classifiers in the ensemble, a predefined number of generations, and the probabilities of crossover and mutation operators. It enhances end-user usability and adaptability to each problem.

3.2. *Pool of classifiers*

Each classifier in the pool is focused on modeling a different k-labelset, then using LP to build the model, and not restricting k to be a fixed value for all of them. LP has been successfully used in the literature previously [11,13,16], since it is able to model the compound dependencies among groups of labels. Although any multi-label classifier might be used, using others such as Classifier Chains (CC) would introduce extra complexity in the search process. Since its chain order should be also considered in the individual codification, it would become more difficult to find an optimal solution. Anyway, note that CC may be used with the current implementation, but with a random order. On the other hand, using different values of k makes the ensemble consider groups of different size, avoiding the need to fix a specific value for each case.

For the selection of k we propose three different modes or versions. First, we propose the *fixed* mode, which as other state-of-the-art methods uses a fixed value of k, usually being $k = 3$ [10,11,13]. In this way, small subsets of labels are considered, so it avoids highly complex and imbalanced LP models. However, in some cases it would not be enough to model the dependencies among just three labels at a time, encouraging to select subsets of different size.

Second, we propose the *uniform* mode, which selects k in the range $[k_{min}, k_{max}]$, where all values of k have the same probability to be selected. We propose to use $k_{min} = 3$ and $k_{max} = q/2$. Thus, the ensemble would include classifiers modeling different subsets of labels, including smaller subsets that will generate simpler LP methods, and bigger subsets which would lead to more complex base models but being able to model the relationship between a higher number of labels.

The third approach, known as *gaussian* mode, considers variable values of k as previous approach, but in this case smaller k values have higher probability to be selected than bigger ones. In some cases, the labels are not so dependent among themselves as to create such big k-labelsets. Therefore, with this approach we aim to have a better trade-off between simpler LP models and more complex ones, favoring the selection of smaller k-labelsets while still giving chance to create bigger ones. The probability to select the size of the k-labelsets is modeled as the right part of a gaussian function, with values $\mu = k_{min}$ and $\sigma = \left(k_{max} - k_{min}\right)/2.5$, so the 98.8% of labelsets are in the selected range. To select each k, a random value is generated following this distribution and rounded; if the selected k was lower than k_{min} or higher than k_{max}, it is set to one of these extremes.

Once the value of k is selected, the k-labelset is created considering both the relationship among the labels and the number of times that each label appears in the pool so far. In this way, we aim to model together labels that are more related among themselves as well as not disregard any of the labels in the dataset. For building each k-labelset, first one label is randomly selected. Then, until k labels are selected, the vector of weights $\boldsymbol{w}$ for each non-selected label λ_l is calculated as in Eq. (1). Then, a label is selected randomly considering $\boldsymbol{w}$, where labels with higher weights have more chance to be selected next.

$$w_l = \left(\varepsilon + \sum_{a \in kL} |\phi_{l,a}|\right) \cdot \left(\exp\left(\frac{1}{1+f_l}\right)\right) \tag{1}$$

The first term of the equation considers the relationship among labels to calculate the weight for label λ_l. The ϕ coefficient measures the relationship among pairs of labels in the range $[-1, 1]$, being -1 total indirect correlation, 1 total direct correlation, and 0 absence of correlation [37]. As we aim to consider the correlation among labels, regardless of whether it is positive or negative, the absolute value of ϕ is used. The ϕ value between λ_l and the rest of currently active labels in the k-labelset kL is added, as well as a small $\epsilon = 1\text{E} - 3$ value to avoid probabilities of 0.

The second term of the Eq. (1) considers the number of times that the label λ_l appears in the current pool (f_l) to calculate its probability of being selected. Therefore, those labels that have been rarely selected for a k-labelset get their probability increased.

Algorithm 1 presents the pseudo-code of the process to generate the pool of classifiers. Until the expected number of average votes per label in the pool ($\overline{v}$) is not reached, different k-labelsets are created following a specific mode (lines 1–9). The generation of each k-labelset (line 6) is guided by the formula in Eq. (1). If any label λ was not selected by the previous process for any of the k-labelsets, the ensembles will not be able to predict it; thus, a random label in a random k-labelset is removed, and λ is added, so all labels are ensured to appear in the pool (lines 10–15). Then, a multi-label classifier (in this case, a LP model) is built for each of the k-labelsets over a random subset of the training instances (lines 16–19). Since classifiers in the pool are independent from each other, they are built in parallel. Finally, the pool of classifiers is returned (line 20).

With this process, we ensure that all k-labelsets are different, and that all labels are present in the pool. Besides, the fact of using random subsets of the instances to build the base classifiers increases the diversity of the ensemble, which is also enhanced by selecting different k-labelsets.

3.3. *Individuals*

Each individual of the population encodes an EMLC in a tree shape, as the example in Fig. 2. It consists of combiner nodes (non-terminal) and base multi-label classifiers from the pool (terminal). Each classifier gives predictions for the labels in its k-labelset, and the internal nodes combine these predictions by majority voting for each label in any of the children nodes. Note that a given classifier may appear more than once in the tree (as *MLC1* in the example).

In Fig. 3 an example of a small tree giving prediction for an instance is presented. Note that in the example all classifiers have $k = 3$.

Algorithm 1 Creation of the pool of classifiers.

```
Input
    $k_{min}$: Minimum size for the $k$-labelsets.
    $k_{max}$: Maximum size for the $k$-labelsets.
    $mode$: Approach to select the $k$-labelsets.
    $q$: Number of labels in the dataset.
    $\overline{v}$: Average number of votes per label in the pool.
    $\mathcal{D}$: Multi-label dataset.
    $\boldsymbol{\phi}(\mathcal{D})$: Matrix of phi correlations among labels.
    $r$: Ratio of training instances to use in each classifier.
Output
    $\boldsymbol{pool}$: Pool of multi-label classifiers.
 1: $\boldsymbol{kLs} \leftarrow \emptyset$
 2: $vTotal \leftarrow 0$
 3: while $(vTotal/q) < \overline{v}$ do
 4:     $k \leftarrow selectK(k_{min}, k_{max}, mode)$
 5:     repeat
 6:         $kL \leftarrow generateKLabelset(k, \boldsymbol{\phi}(\mathcal{D}))$
 7:     until $kL \notin \boldsymbol{kLs}$
 8:     $\boldsymbol{kLs} \leftarrow \boldsymbol{kLs} \cup \{kL\}$
 9: end while
10: for each label $\lambda$ in $\mathcal{D}$ do
11:     if $\lambda \notin \boldsymbol{kLs}$ then
12:         $kL \leftarrow selectRandomKLabelset(\boldsymbol{kLs})$
13:         $replace(kL, randomLabel(kL), \lambda)$
14:     end if
15: end for
16: $\boldsymbol{pool} \leftarrow \emptyset$
17: for $kL$ in $\boldsymbol{kLs}$ do
18:     $\boldsymbol{pool} \leftarrow \boldsymbol{pool} \cup buildMLC(\mathcal{D}, kL, r)$
19: end for
20: return $\boldsymbol{pool}$
```

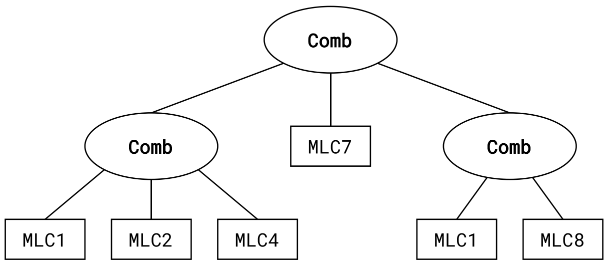

Fig. 2. Example of individual.

First, classifiers in the leaves give their own prediction for the labels in their k-labelset. In the figure, predictions are in the form $\lambda_l : \hat{y}_l$. Then, combiner nodes collect the predictions for each label among their children nodes, and create their own prediction by majority voting. Let us consider the combiner node in the left; the classifiers *MLC1* and *MLC4* consider that label λ_2 is relevant ($\lambda_2 : 1$), while *MLC2* considers that it is irrelevant ($\lambda_2 : 0$). Thus, as 2 out of 3 votes consider that the label λ_2 is relevant, the prediction of the combiner is that it is relevant ($\lambda_2 : \frac{2}{3} \rightarrow 1$). On the other hand, as λ_7 has only one vote in the children nodes and it is irrelevant, for the combiner node this label is irrelevant. These predictions are combined in subsequent combiner nodes until the root node is reached, where the final ensemble prediction is created as majority voting of its children nodes (in the example, the final prediction is created by combining the predictions of *MLC7* and the two other combiner nodes). The bipartition for this given instance would be $\hat{b}_i = \left(\hat{Y}_i : \{\lambda_1, \lambda_2, \lambda_4, \lambda_6\}, \hat{Z}_i : \{\lambda_3, \lambda_5, \lambda_7\}\right)$

Individuals are created by following the grammar in Fig. 4. The root node is always a combiner node (*Comb*), followed by the symbol

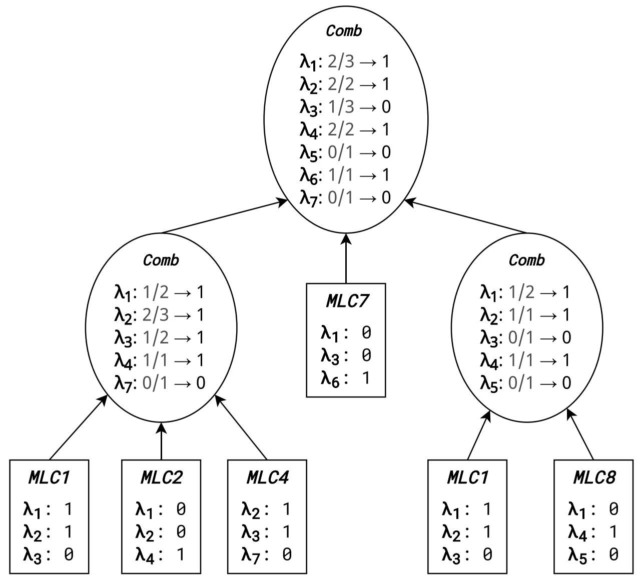

Fig. 3. Example of combination of predictions in an individual. Each MLC gives prediction for its k-labelset and internal nodes combine predictions of children nodes. Each node shows its predictions in the form $\lambda_l : \hat{y}_l$.

```
<S> → <Comb> ';'
<Comb> → '(' (<Comb>|<MLC>){2, mC} ')'
<MLC> → MLC1 | MLC2 | ... | MLCn
```

Fig. 4. Grammar for creating individuals.

of end of individual ";". Each *Comb* node is substituted by a number of children nodes (randomly selected between 2 and mC), each of them being either another combiner node or a classifier from the pool (MLC_i). These children are between parenthesis to indicate the hierarchy of nodes in the string genotype. For ease of representation, *MLCs* are represented just by their index. Note that in the event that the maximum allowed depth for the tree is going to be reached (i.e., if the combiner node is in depth $mD-1$), only *MLC* nodes can be selected as children of a combiner node, in order not to violate this restriction. The example individual in Fig. 2 would be coded with the following genotype: "*((1 2 4) 7 (1 8));*".

3.4. Evaluation

The evaluation of individuals is carried out by evaluating their predictions, which are obtained by combining the predictions of leaves up to the root node. Although each base classifier is built using a sampled set of the training instances, the evaluation of individuals is made using the whole training set; thus, each classifier gives prediction to a percentage of unseen instances, also providing an estimation of how well they would perform on test set.

For the fitness function, we differentiate between two types of individuals: incomplete and complete trees. Even though all labels are ensured to appear in the pool of classifiers, individuals that are not able to give prediction for all labels in the dataset could be created. Regarding the example in Fig. 3, consider that an individual including only *MLC1*, *MLC2*, and *MLC4* is created, not being able to predict labels λ_5 and λ_6. These trees are called incomplete trees, while complete trees are those that can give prediction for all labels in the dataset.

Being L_t the set of labels that appear in the tree at least once, we propose the fitness function in Eq. (2), where incomplete trees ($|L_t| < q$) have negative fitness and complete trees ($|L_t| = q$) have positive value.

In this way, we aim to get rid of incomplete trees. As the fitness for incomplete trees is closer to zero as the number of labels that do not appear in the ensemble is lower, in case that several incomplete trees appear in the population, those that are closer to be complete are maintained.

$$\uparrow fitness = \begin{cases} -\left(q - |L_t|\right)/q & \text{if } |L_t| < q \\ (\text{ExF} + \text{MaF})/2 & \text{if } |L_t| = q \end{cases} \tag{2}$$

$$\uparrow \text{ExF} = \frac{1}{m}\sum_{i=1}^{m} \frac{2|\hat{Y}_i \cap Y_i|}{|\hat{Y}_i| \cup |Y_i|} \tag{3}$$

$$\uparrow \text{MaF} = \frac{1}{q}\sum_{l=1}^{q} \frac{2 \cdot tp_l}{2 \cdot tp_l + fp_l + fn_l} \tag{4}$$

On the other hand, for complete trees, a combination of two evaluation metrics is computed: Example-based FMeasure (ExF) [38] and Macro-averaged FMeasure (MaF) [39]. FMeasure is a robust and widely used evaluation metric in classification, even more in imbalanced scenarios [40]. In MLC there exist several approaches to calculate this metric [41]. The ExF (Eq. (3)) computes the metric for each instance, considering the multi-label prediction as a whole, thus being able to capture the relationship among labels in its calculation. On the other hand, MaF (Eq. (4)) computes the metric using the confusion matrix of each label independently, then averaging the value. Therefore, MaF gives the same importance to all labels in its calculation, regardless of their frequency. Note that tp_l, fp_l, and fn_l stand for the number of *true positives*, *false positives*, and *false negatives* of λ_l, respectively.

Although ExF and MaF are partly correlated [42], they treat the dependencies and importance of labels in different ways, which plays an important role on the algorithm. ExF is able to detect the dependencies among labels in its calculation, and it does not care about the imbalance of labels. MaF does not consider the dependency among labels, but gives the same importance to all labels, not neglecting minority labels in the calculation. Therefore, this combination provides AG3P-kEMLC with the ability to evolve the ensemble structure while considering both the correlation among labels and their imbalance.

3.5. Genetic operators

The genetic operators comprise a crossover operator aiming to combine useful genetic material between two individuals, trying to exploit the current population, and a mutation operator, which aims to include new genetic material in the population to better explore the search space. Following operators are based on previously proposed ones, which have been widely used in the literature [30,31]. Each individual is applied the crossover or mutation operator with probabilities p_c and p_m respectively.

Algorithm 2 presents the steps followed by the crossover operator. First, a random subtree st_1, not considering the whole tree, is selected from the first parent (line 1). Then, a random subtree st_2, including the possibility of selecting the whole tree, is selected from the second parent (line 2). If the replacement of st_2 in st_1 would produce a violation of the maximum depth of the tree, a subtree is recursively selected from st_2 (lines 3–5). Finally, st_2 replaces st_1 in the first parent and it is returned (line 6).

Since the crossover operator must fit some restrictions, such the maximum depth of the trees, it would be more difficult and restrictive to find a recombination point that produces two feasible trees. Thus, the crossover operator is defined to generate one individual given the two parents. Then, this operation is repeated but swapping the role of the original parents, in order to obtain two offspring individuals. The offspring contain genetic material from both parents, as well as they are always feasible, since the maximum depth restriction is fulfilled.

In Fig. 5, an example of application of the crossover operator is presented, where shaded nodes are proposed to be recombined. Note that in the example, in the second parent, first the node with the dotted line was selected; however, given that the maximum depth restriction would be violated if it was replaced (considering $mD = 3$), a random subtree below it was selected, finally obtaining a feasible individual fulfilling the restrictions.

Algorithm 2 Crossover operator.

```
Input
    ind_1: First parent.
    ind_2: Second parent.
    mD: Maximum depth of trees.
Output
    child: Crossed individual.
1: $st_1 \leftarrow subtree(ind_1, selectRoot{=}False)$
2: $st_2 \leftarrow subtree(ind_2, selectRoot{=}True)$
3: while $depth(st_2) > (mD - depth(st_1))$ do
4:     $st_2 \leftarrow subtree(st_2, selectRoot{=}False)$
5: end while
6: return $replace(ind_1, st_1, st_2)$
```

On the other hand, the operation of the mutation operator is presented in Algorithm 3. First, a random subtree st is selected from the original individual (line 1). With a probability of 0.5, a random terminal (i.e., a classifier from the pool) is selected to replace st, and the mutated individual is returned (lines 2–3). If the depth of the root node of st in the original individual is mD, in order to not violate the maximum depth restriction, a random terminal is directly selected too. Otherwise, a subtree st_2 is created by following the grammar, where the maximum allowed depth for the subtree is mD minus the depth of st. Then, st_2 replaces st in the individual, which is returned (lines 4–7).

Algorithm 3 Mutation operator.

```
Input
    ind: Original individual.
    mD: Maximum depth of trees.
    grammar: Grammar to build the trees.
Output
    child: Mutated individual.
1: $st \leftarrow subtree(ind, selectRoot{=}False)$
2: if $depth(st, ind) == mD$ or $random(0,1) < 0.5$ then
3:     return $replace(ind, st, randomTerminal())$
4: else
5:     $st2 \leftarrow createSubtree(grammar, mD - depth(st))$
6:     return $replace(ind, st, st_2)$
7: end if
```

An example of the mutation operator is shown in Fig. 6, where the shaded node is the chosen as mutation point. The resulting individual is always feasible, because the restrictions are always considered in the process. Unlike in crossover, the mutation operator allows to introduce new genetic material that was not previously present in the population, increasing the diversity in the population and making it able to better explore the search space.

3.6. Auto-adaptability

One of the main drawbacks of the EAs is the wide range of hyper-parameters that need to be tuned to maximize their performance, such as the number of individuals, maximum number of generations, or probabilities of crossover and mutation. AG3P-kEMLC is able to reduce the need of tuning some of these parameters by including an auto-adaptive procedure for crossover and mutation probabilities [43,44].

The probabilities of crossover and mutation are related in such a way that $p_c = 1 - p_m$, and both are 0.5 at the beginning of the evolution. When each generation is completed, the average fitness of the current

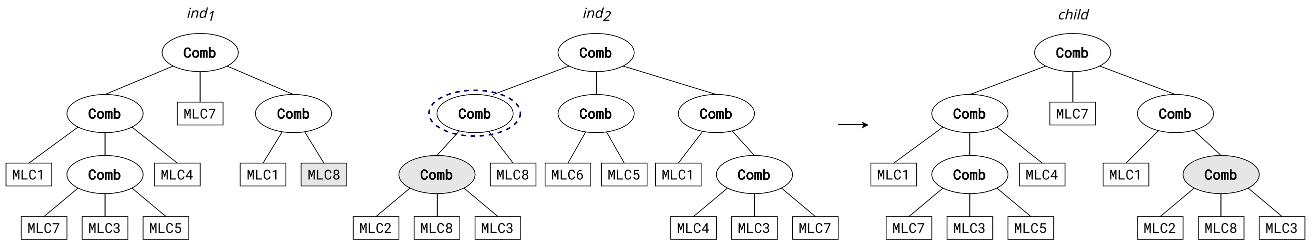

Fig. 5. Crossover operator.

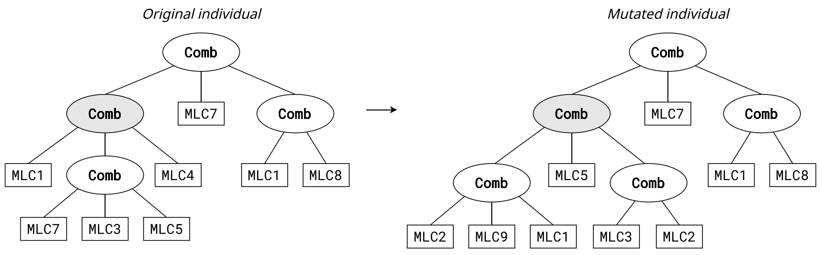

Fig. 6. Mutation operator.

population is compared to the best average fitness of a population so far. If the current population is better on average than the best one, the crossover probability is increased in 0.02, and the mutation probability is therefore decreased, since depth search or exploitation is preferred. On the other hand, if the average population do not improve the best so far, the crossover probability is decreased in 0.02 and the mutation increased. It favors the inclusion of new genetic material in the population for a better diversity and exploration of the search space. The idea behind this auto-adaptive process is that, when the population does not improve, i.e., the population is stuck in a local optimum, the mutation operator may create individuals that explore new areas in the search space, increasing the exploration [43,44]. In the case when the population is improving in average, the exploitation of these solutions is encouraged with the aim to reach their optimum. With this process, where exploitation and exploration are favored depending on the scenario, we obtain a trade-off, which is necessary to reach good solutions.

Besides, a stop criterion based on the improvement of the best individual of the population is defined. If the best individual does not improve in some generations, the algorithm could be stalled, or it may have already found an optimal. Therefore, in order to avoid possible overfitting and to reduce the required runtime, the algorithm is stopped after some generations without improvement of the best individual, avoiding the need for setting the maximum number of generations.

3.7. Time complexity

For defining the time complexity of AG3P-kEMLC, we divide the algorithm in two independent processes: the creation of the initial pool, and the G3P evolution.

In the creation of the initial pool, the time complexity will be given by the complexity of the classifier used. We use C4.5 decision tree as single-label classifier, which complexity is $\mathcal{O}\left(m \times d^2\right)$, m being the number of training instances and d the number of input attributes [45]. In the worst case, $\frac{q \cdot \bar{v}}{k_{min}}$ classifiers are created in the initial pool; however, in practice when $k_{max} > k_{min}$ much less classifiers are needed. Besides, this process may be performed in parallel, since classifiers are independent from each other. Thus, the time complexity of the first part of the algorithm in the worst case would be $\mathcal{O}\left(m \times d^2 \times \frac{q \cdot \bar{v}}{k_{min}}\right)$.

On the other hand, the most complex part of EAs is usually the evaluation of individuals. To evaluate an individual in AG3P-kEMLC, the predictions of base classifiers for all the training instances are combined in the tree, resulting in a complexity of $\mathcal{O}(m)$. As the number of total evaluations is given by the population size (*popSize*) and the number of generations (G), the complexity of the evolution process is $\mathcal{O}(m \times popSize \times G)$.

4. Experimental studies

In this section the different experiments and studies carried out in this work are presented. First, the multi-label datasets and evaluation metrics used in the experiments are described, and then, the settings of the different experiments as well as the methods and their parameters are presented.

4.1. Datasets

A set of 20 multi-label datasets obtained from the KDIS research group repository[1] have been selected. In Table 2 the main characteristics of the datasets are presented, including number of instances (d), number of input features (m), number of labels (q), cardinality, i.e., the average number of labels associated with each instance (*Card*), average imbalance ratio per label (*avgIR*), and ratio of dependent labels pairs (*rDep*). The characterization of datasets has been performed using MLDA tool [46].

These datasets cover a wide range of characteristics, not only regarding the number of labels, which range from 6 to 174, but also considering the imbalance (ranging from near to 1, i.e., low imbalance of labels, to almost 300, meaning that on average, the most frequent label appears up to 300 times more than the rest of labels) and relationship among labels (ranging from near to 0 in Stackex coffee meaning almost no dependency of labels, to almost 1 in Emotions). Therefore, as our proposal considers the imbalance, relationship, and size of the output space to build the ensemble, this set is considered as diverse and consistent enough to carry out the experiments.

4.2. Evaluation metrics

We use five evaluation metrics to assess the performance of methods in the experiments. Given that in MLC the predictions may be not only

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

Table 2
Datasets and their characteristics, including number of instances (m), number of attributes (d), number of labels (q), cardinality (*card*), average imbalance ratio (*avgIR*), and ratio of dependent label pairs (*rDep*). 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
Gnegative	1392	1717	8	1.046	18.448	0.536
Plant	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
Human	3106	440	14	1.185	15.289	0.418
Birds	645	260	19	1.014	5.407	0.123
tmc2007–500	2860	500	22	2.230	17.225	0.364
Ohsumed	1393	1002	23	1.682	8.738	0.103
Yahoo arts	1497	500	26	1.671	25.486	0.138
Genbase	662	1186	27	1.252	37.315	0.157
Medical	978	1449	45	1.245	89.501	0.039
NusWide	2696	128	81	1.895	77.443	0.089
Mediamill	2195	120	101	4.430	294.599	0.116
Stackex coffee	225	1763	123	1.987	27.241	0.017
CAL500	502	68	174	26.044	20.578	0.192

totally correct or incorrect, but partially correct, specific evaluation metrics must be used.

Hamming loss has been a widely-used metric in MLC. However, in cases with a large number of labels but low cardinality, it tends to be 0 in all cases. Therefore, the Adjusted Hamming loss (AHL) was proposed in [47], and it is defined in Eq. (5). Hereafter, it is indicated with ↓ if the metric is minimized and with ↑ if it is maximized. AHL computes the ratio of misclassified labels divided by the number of positive labels in both the true and predicted sets, averaging it by the total number of instances. Note that Δ is the symmetric difference between two binary sets.

$$\downarrow \text{AHL} = \frac{1}{m} \sum_{i=1}^{m} \frac{|\hat{Y}_i \Delta Y_i|}{|\hat{Y}_i \cup Y_i|} \tag{5}$$

Subset accuracy (SA) [48] is a strict metric which measures the ratio of instances whose prediction exactly matches the ground truth, including both relevant and irrelevant labels. It is defined in Eq. (6), $[\![\pi]\!]$ returning 1 if predicate π is true, and 0 otherwise.

$$\uparrow \text{SA} = \frac{1}{m} \sum_{i=1}^{m} [\![\hat{Y}_i = Y_i]\!] \tag{6}$$

The FMeasure is a metric that combines both precision (ratio of true relevant labels among those predicted as relevant) and recall (ratio of relevant labels correctly predicted as relevant), and it has been widely used in classification, especially in imbalanced problems. In multi-label scenarios, there are several ways to calculate the FMeasure, such as example-based FMeasure (ExF, Eq. (3)) [38], micro-averaged FMeasure (MiF, Eq. (7)), and macro-averaged FMeasure (MaF, Eq. (4)) [39]. ExF computes the metric for each instance separately, so it is able to capture the relationship among labels in its calculation. Both MiF and MaF calculate the metric based on the confusion matrix. MiF first joins all confusion matrices and then computes the metric, giving more importance in the calculation to more frequent labels. MaF computes the metric for each label independently and then averages the values, so all labels have the same importance in the calculation, and infrequent labels are not neglected.

$$\uparrow \text{MiF} = \frac{\sum_{l=1}^{q} 2 \cdot tp_l}{\sum_{l=1}^{q} 2 \cdot tp_l + \sum_{l=1}^{q} fp_l + \sum_{l=1}^{q} fn_l} \tag{7}$$

4.3. Experimental settings

AG3P-kEMLC has been built using the JCLEC [49], Mulan [50], and Weka [51] libraries, and the code is publicly available in a GitHub repository.[2] The first purpose of the experimental study is to analyze and carefully study the performance of AG3P-kEMLC. It comprises the analysis of the pool of classifiers, the convergence of the algorithm, the auto-adaptive genetic operators, the structure of the EMLCs, and the algorithm runtime. Then, an experimental comparison between the different versions of AG3P-kEMLC is proposed to verify if any of them performs significantly better than the rest. Finally, some experimental studies involving other EMLCs in the literature are carried out, to determine if the proposed method outperforms state-of-the-art methods.

Table 3
Average size of the k-labelsets in the pool.

	AG3P-k3	AG3P-ku	AG3P-kg
Emotions	3.00	3.75	3.49
Birds	3.00	5.93	4.44
Medical	3.00	12.91	8.55

For the experiments, we use three different versions of AG3P-kEMLC: with a fixed $k = 3$ value for all labels (hereafter named as AG3P-k3), with a variable value of k following the *uniform* mode (hereafter named as AG3P-ku), and with a variable value of k following the *gaussian* mode (hereafter named as AG3P-kg). In all cases, AG3P-kEMLC is executed using $\bar{v} = 20$, so there are enough votes per label in the initial pool; that does not mean that the final ensemble has this number of votes per label, since the evolutionary process selects those classifiers that best fit in the tree. The number of individuals is fixed to 50, while the maximum number of children at each node is set to $mC = 7$, the maximum depth of the tree to $mD = 3$, and each classifier in the pool is built using a random sample of 75% of the instances. Given preliminary results, we fixed the stopping criterion to 10 generations without improvement of the best individual.

Note that in cases where fixed $k = 3$ is used, in bigger datasets the size of the tree would not be enough to include all classifiers in the pool if needed. For example, if $q = 50$, there would be $(20 * 50)/3 \approx 333$ classifiers in the pool, while the tree allows up to $7^3 = 343$ leaves in the most complex tree. Therefore, when fixed $k = 3$ mode is being used and $q > 50$, the maximum depth is set to $mD = 4$ to allow bigger trees. On the other hand, in cases with a low number of labels, such as for $q = 6$, all versions are supposed to use only $k = 3$ (since those using variable

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

k use up to $q/2$). Given the reduced number of different k-labelsets, in these cases when using fixed k, only $\overline{v} = 10$ is used; while when using variable k, $\overline{v} = 20$ is still used but a maximum size $k = 5$ is allowed to include variety in the size of the k-labelsets.

All the methods use C4.5 decision tree [52] as single-label classifier (except RF-PCT, which uses PCTs), given its generalized use in EMLCs [12,13,53]. The rest of the EMLCs are executed with the parameters proposed by their authors. Unless other specified, they include $n = 10$ classifiers. EBR and ECC use sample with replacement of the original dataset for each classifier. HOMER generates $c = 3$ clusters at each node and uses the *balanced k-means* clustering method. EPS prunes the instances with labelsets appearing less than 3 times, and reintroduces them using the top two best ranked subsets.[3] RA*k*EL includes $n = 2q$ classifiers and uses $k = 3$ for all base members. EME has been executed with 50 individuals, the number of generations varies from 110 to 300 generations depending on the dimensionality of the dataset, the probability of crossover was set to 0.9 ($q \leq 8$) and 0.8 ($q > 8$), while the probability of mutation was set to 0.2. The EMLC of EME contains $n = 2n$ members and fixed $k = 3$ Finally, EAGLET is executed with up to 50 generations, the EMLC contains $n = 3.33q$ members with $k = 3$, and the size of the population is equal to $2n$. In EAGLET, the probability of crossover is set to 0.7 and probability of mutation to 0.2 ($q \leq 30$) and 0.1 ($q > 30$).

For all the experiments, the datasets are partitioned following a random 5-fold cross-validation procedure, and the algorithms are executed using 10 different seeds for random numbers, so the results are averaged among 50 executions. The evaluation metrics are reported over the test set in each case. For comparing the performance of the state-of-the-art methods, Skillings-Mack's [54] and Holm's [55] statistical tests are performed, using Skillings Mack R package [56] and the KEEL software tool [57].

5. Results and discussion

In this section, we present and discuss the results obtained in the experiments. First, the performance of AG3P-kEMLC is analyzed, and then, it is compared with the state-of-the-art EMLCs. Further results are provided in the supplementary material in the KDIS Research Group webpage.[4] It includes results using different aggregation criteria, different single-label base classifiers, comparison versus standard non-ensemble multi-label classifiers, and more examples of single runs of different variants of AG3P-kEMLC.

5.1. Analysis of AG3P-kEMLC

For the analysis of the performance of the different AG3P-kEMLC versions, we select 3 datasets of different size and characteristics. Emotions, Birds, and Medical, with 6, 19, and 45 labels respectively are selected for being analyzed in detail. These datasets also have incrementally more imbalanced output space and decreasing dependency among labels. Therefore, we cover a wide range of characteristics to show the behavior of the proposed algorithm in different scenarios. The results on these three datasets are presented just for understanding purposes, but they do not bias the performance of the algorithm at all, since they are not used for selecting any of the hyper-parameters.

Unless otherwise indicated, the results presented in this section are averaged among the 50 executions of each method over each dataset. For length restrictions, in some of the experiments, the results of single executions for only one of the variants are presented. However, the results for all variants are included in the supplementary material.

[3] CAL500 dataset has as many different labelsets as instances, so it was executed without pruning the infrequent labelsets.

[4] https://www.uco.es/kdis/AG3P-kEMLC/.

Table 4
Average number of classifiers in the pool.

	AG3P-k3	AG3P-ku	AG3P-kg
Emotions	20.0	32.1	34.7
Birds	127.0	65.0	85.2
Medical	300.0	69.5	104.5

Table 5
Average number of generations that AG3P-kEMLC needs to converge.

	AG3P-k3	AG3P-ku	AG3P-kg
Emotions	50	51	51
Birds	62	54	57
Medical	81	57	61

First the size of the k-labelsets and number of appearances of each label in the pool of classifiers are analyzed. Then, the convergence of the algorithm is studied, including the convergence of the best individual, the variation in the probabilities of the genetic operators, and the size of the final ensembles. Finally, the runtime of each of the versions is analyzed and their predictive performance is compared using statistical tests.

5.1.1. Pool of classifiers

We have proposed different versions of AG3P-kEMLC: using fixed $k = 3$ value (AG3P-k3), and using variable k in the members either with same probabilities for all available k values (AG3P-ku) or giving more probability to smaller k-labelsets (AG3P-kg). In this section, we analyze the initial pool of classifiers, according to the size of the k-labelsets and the number of classifiers in the pool.

For the analysis of the size of the k-labelsets of the classifiers in the pool, only AG3P-ku and AG3P-kg are analyzed; for AG3P-k3 this analysis does not make sense since all labelsets have the same size. Fig. 7 compares the ratio of k-labelsets of each size in the initial pool for the three datasets. As observed, it follows the distribution described in Section 3.2, in AG3P-kg more k-labelsets of smaller size are included, while in AG3P-ku, the number of k-labelsets of each size is similar. Note that for AG3P-ku in Emotions, the distribution of k-labelsets is not so uniform with a low number of k-labelsets of size $k = 5$, since there is only 6 different possible combinations of subsets of 5 labels (Emotions has 6 labels in total).

Besides, Table 3 shows the average value of k among all classifiers in the pool. As expected, the classifiers in AG3P-ku are able to model, on average, the relationships among bigger subsets of labels than AG3P-kg, since the latter favors the selection of smaller k-labelsets. Thus, while AG3P-ku would be able to better exploit the relationships among labels, AG3P-kg would increase the diversity by having a wider range of classifiers in the pool to combine in the ensemble.

In Table 4, the average number of classifiers in the pool for each dataset and method are shown. It can be observed as the higher ratio of smaller k-labelsets, the higher the number of classifiers in the pool to obtain an average $\overline{v} = 20$ (except AG3P-k3 in Emotions, which is executed with $\overline{v} = 10$). This will also influence in the efficiency of the methods, since the lower the number of classifiers to build in the initial pool, the lower should be the runtime required to execute the algorithm.

Fig. 8 shows the number of appearances or votes of each label in the initial pool. The number of votes per label in both versions is very similar, since they only differ in the size of the k-labelsets but not in the way the labels in each k-labelsets are selected. Not all labels appear in the classifiers in the pool the same number of times, but they are biased by their average phi coefficient with the rest of labels, which is also plotted in the figure. The higher the relationship of a label with the rest, the higher the probability to appear in the k-labelsets (see Section 3.2). Besides, it makes more sense that a label which is almost independent

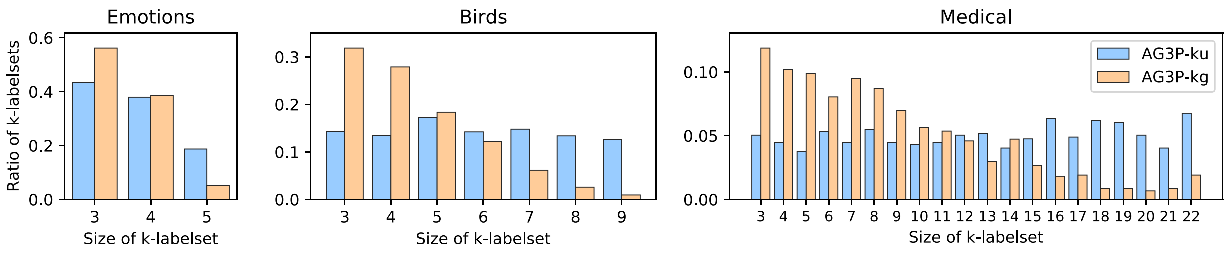

Fig. 7. Ratio of k-labelsets of each size in the initial pool in AG3P-ku and AG3P-kg.

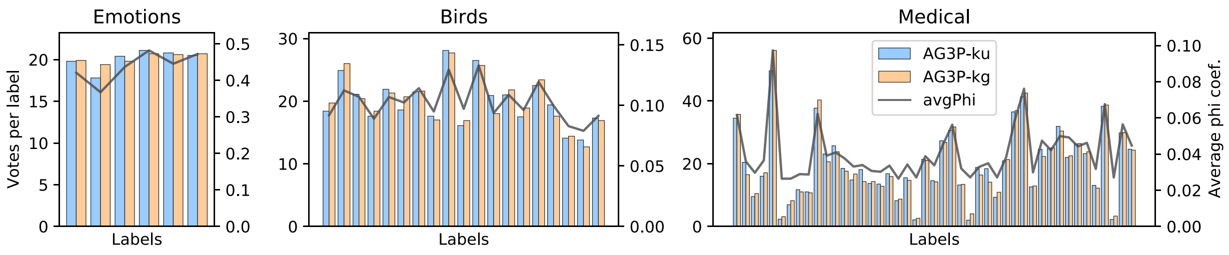

Fig. 8. Number of average votes per label in the pool.

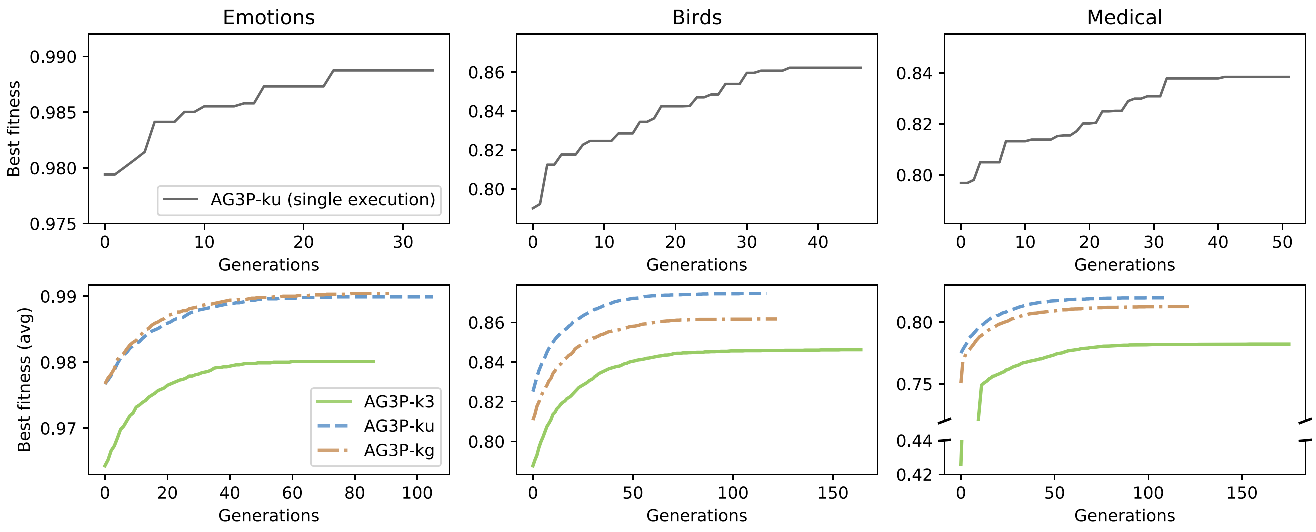

Fig. 9. Convergence of AG3P-kEMLC, including a single execution of AG3P-ku (top) and average of executions for all versions (bottom).

of the rest (has low average phi value) appears in a lower number of classifiers, since it would be similarly modeled independent of the rest of labels that appear in the same k-labelset.

5.1.2. Convergence of best individual

For studying the convergence of AG3P-kEMLC, first the evolution of the best individual fitness for a given execution in AG3P-ku is shown (so the auto-stop condition given the non-improvement of the best individual is seen in a specific execution). Then, the value of fitness of the best individual in each generation is also presented averaged among all executions for the three versions of AG3P-kEMLC. As not all executions need the same number of generations, the figures include as much generations as the execution that needed most; for those executions that stopped earlier, the fitness value in the last generation is used up to the end to calculate the average of each generation.

Fig. 9 shows the convergence of AG3P-kEMLC in the three datasets. First, it can be observed (top of the figure) how the algorithm automatically stops when the best individual does not improve in ten generations, allowing to reduce the runtime and avoiding overfitting of the final model.

On the other hand, the fitness of the best individual in each generation averaged among all executions is presented in the bottom subplots. In general, AG3P-k3 is the one obtaining the worst results according to the fitness of the best individual reached (note that these results are obtained only using the training set), while AG3P-kg and AG3P-ku tend to obtain better results. Also, AG3P-k3 is the one which needs more generations to converge. This is to be expected, since the lower the size of the k-labelsets, the more the number of classifiers in the pool, and therefore, it is more complex to combine them in an ensemble because of the increased search space. Note that it does not apply to the case of Emotions because AG3P-k3 was executed with $\overline{v} = 10$ instead of $\overline{v} = 20$, so the search space is much smaller. Besides, in Medical we can observe that AG3P-k3 obtains very bad fitness values in early generations on average, given the great amount of incomplete trees in first generations; this also affects the need of a higher number of generations to converge.

Although these figures show the fitness values up to the execution that needed more generations, in Table 5 the average number of generations to converge is shown, to have a better picture of how many generations it needs to obtain the best solution. It supports our hypothesis that the lower the size of the k-labelsets, the higher the

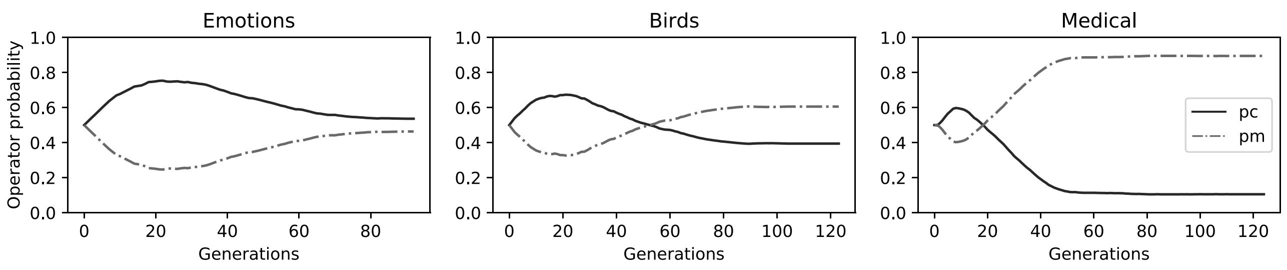

Fig. 10. Average value of the probabilities of crossover and mutation of AG3P-kg.

Table 6
Average number of leaf nodes in the best tree obtained by AG3P-kEMLC. In parenthesis, the percentage of leaf nodes used out of the total leaf nodes in the largest possible tree.

	AG3P-k3	AG3P-ku	AG3P-kg
Emotions	72.26 (21.1%)	77.50 (22.6%)	72.84 (21.2%)
Birds	61.68 (18.0%)	61.56 (17.9%)	63.38 (18.5%)
Medical	72.56 (21.2%)	51.82 (15.1%)	51.88 (15.1%)

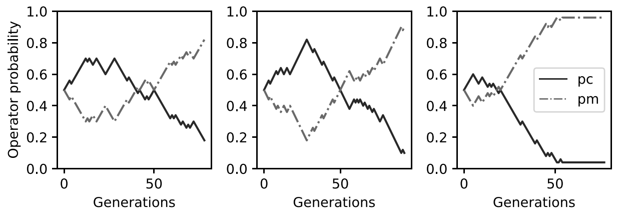

Fig. 11. Probabilities of crossover and mutation operators in different executions of AG3P-kg in Birds dataset.

number of generations to converge. It can be observed also when comparing AG3P-ku and AG3P-kg; since AG3P-kg allows a higher number of smaller k-labelsets, it tends to need slightly more generations to converge.

5.1.3. Genetic operators

One of the main advantages of AG3P-kEMLC is the ability to auto-adapt some of its hyper-parameters during the execution, both reducing the number of parameters that the user has to set and also being able to adapt to each specific problem. In Fig. 10 the values of the crossover and mutation probabilities are plotted along the generations for the three datasets in AG3P-kg. These figures show the values of p_c and p_m averaged among all executions, similar to previous section. It can be observed that in general, in early generations the crossover probability increases, meaning that the current population is improving each generation, and therefore the exploitation of current solutions is preferred. Then, as generations pass, the population does not improve (on average), so the probability to apply the mutation operator is increased, even surpassing the probability of applying the crossover operator. Although there exists a point when the population does not improve on average (so the mutation probability increases), it still allows the algorithm to not to converge prematurely and to obtain better solutions by widely exploring the search space, which is proven by the fact that a large number of generations passes between the moment in which the probability of mutation starts to increase and the moment when the algorithm auto-stops given the non-improvement of the best.

We can observe as the fact of adapting to each specific dataset is beneficial, since they all do not behave equally as seen in Fig. 10. However, the variation in these probabilities is still quite dependent on each specific run, even on the same dataset. Fig. 11, presents the evolution of the probabilities over three different single executions on Birds dataset. The tendency is similar in all cases, but here we could observe that: (1) the change in the probabilities in each execution is not as smoothly as seen in the average plot, but the probabilities for each operator increase and decrease all over the generations, as the average population improves or not the best one; (2) the point in which the probability of mutation is higher than the crossover one is different in each case; and (3) in some cases, as in the right subplot, there is a large number of generations in which the mutation operator has the higher probability and it still is improving the performance of the best individual.

5.1.4. Size of the final EMLC

In this section, we analyze the size of the final EMLCs. We study the number of leaf nodes in the best tree, as well as the number of different votes for each label in the ensemble, i.e., the number of times that each label appears in the classifiers used in the best tree.

In Table 6, the average number of leaf nodes in the best tree of each execution are reported for the three variants. Considering that for these cases the trees may include up to $7^3 = 343$ leaf nodes, the percentage of leaves out of the maximum possible is also reported. It can be noted that, in average, the best tree has 19% of possible leaf nodes, meaning a size much lower than the maximum allowed. In this way, we demonstrate that: (1) the maximum size of the trees defined in the experiments seems appropriate, since it does not reach the maximum size by far; and (2) bigger trees do not necessarily mean greater performance of the ensemble.

On the other hand. Fig. 12 presents the number of votes for each label in the final ensemble, as well as the average number of votes per label. In Emotions, all labels appear a similar number of times in the final ensemble, but in Birds and even more in Medical each label appears in a different number of classifiers; but all labels are ensured to appear, being able to give prediction for all labels.

However, the imbalance in the number of appearances of each label in the ensemble is not just random, but is slightly biased by the frequency of labels, as compared with Fig. 13, where the frequency of labels in the data is presented. Since infrequent labels appear in a lower number of instances, it is usually easier to learn them than those labels that appear in a higher number of instances; more frequent labels would also have more complex correlations with other labels, so the algorithm tends to include more classifiers comprising the more frequent labels. Therefore, it is not strange that in Medical the number of votes among labels is more imbalanced, given the imbalance of its output space, with an $avgIR = 89.501$, while for Birds is of 5.407, meaning a much less imbalanced label space.

It can be also noted the adaptability of our proposal to each specific problem; while most state-of-the-art methods fix the number of classifiers in the ensemble, our method is able to automatically adapt the number of classifiers to each case. For Emotions, over 18 different votes per label are used on average. On the other hand, AG3P-k3 builds much simpler ensembles for Medical, with just over 4 votes per label

Table 7
Results obtained by the three versions of AG3P-kEMLC for five evaluation metrics and runtime. Best results in each dataset and metric are marked in bold.

	↓ AHL			↑ SA			↑ ExF		
	AG3P-k3	AG3P-ku	AG3P-kg	AG3P-k3	AG3P-ku	AG3P-kg	AG3P-k3	AG3P-ku	AG3P-kg
Emotions	0.469	**0.449**	0.450	0.264	**0.285**	0.282	0.619	**0.638**	0.637
Reuters1000	0.835	**0.824**	0.828	0.116	0.129	**0.130**	0.183	**0.193**	0.187
Guardian1000	0.843	**0.828**	0.837	0.105	**0.129**	0.116	0.176	**0.188**	0.179
Bbc1000	0.808	0.802	**0.790**	0.130	0.148	**0.157**	0.215	0.216	**0.228**
3s-inter3000	**0.898**	0.922	0.926	**0.054**	0.049	0.047	**0.119**	0.088	0.083
Gnegative	**0.409**	0.415	0.415	0.531	**0.536**	0.528	**0.611**	0.602	0.604
Plant	**0.805**	0.826	0.809	0.148	0.145	**0.151**	**0.212**	0.184	0.205
Water-quality	0.573	0.574	**0.572**	0.012	**0.013**	**0.013**	**0.571**	0.570	**0.571**
Yeast	0.497	**0.488**	0.493	0.124	**0.141**	0.129	0.619	**0.625**	0.622
Human	**0.726**	0.742	0.730	0.185	**0.186**	**0.186**	**0.306**	0.284	0.299
Birds	0.419	**0.393**	0.413	0.474	**0.502**	0.481	0.620	**0.644**	0.627
tmc2007–500	0.474	**0.451**	0.459	0.217	**0.241**	0.233	0.628	**0.650**	0.643
Ohsumed	0.672	**0.626**	0.631	0.137	**0.177**	0.170	0.400	**0.446**	0.442
Yahoo arts	0.918	**0.803**	0.814	0.056	**0.122**	0.113	0.093	**0.229**	0.218
Genbase	**0.014**	**0.014**	0.015	0.971	**0.972**	0.969	**0.990**	**0.990**	0.989
Medical	0.230	0.224	**0.223**	0.650	**0.671**	0.665	0.810	0.811	**0.815**
NusWide	0.781	0.757	**0.751**	0.121	**0.163**	**0.163**	0.262	0.276	**0.284**
Mediamill	0.604	0.577	**0.576**	0.044	**0.063**	0.061	0.523	0.548	**0.550**
Stackex coffee	**0.784**	0.860	0.842	**0.076**	0.046	0.058	**0.275**	0.177	0.199
CAL500	0.750	0.736	**0.725**	**0.000**	**0.000**	**0.000**	0.395	0.411	**0.425**
Avg ranking	2.38	**1.80**	1.83	2.60	**1.48**	1.93	2.35	**1.83**	**1.83**

	↑ MiF			↑ MaF			↓ Runtime (s)		
	AG3P-k3	AG3P-ku	AG3P-kg	AG3P-k3	AG3P-ku	AG3P-kg	AG3P-k3	AG3P-ku	AG3P-kg
Emotions	0.657	**0.674**	**0.674**	0.644	0.660	**0.663**	**4.01E+00**	5.01E+00	5.07E+00
Reuters1000	0.229	**0.245**	0.239	0.196	**0.211**	0.209	**4.88E+00**	6.87E+00	6.82E+00
Guardian1000	0.231	**0.246**	0.236	0.193	**0.199**	0.190	**4.93E+00**	6.85E+00	7.17E+00
Bbc1000	0.273	0.280	**0.290**	0.236	0.239	**0.254**	**5.37E+00**	7.73E+00	8.01E+00
3s-inter3000	**0.157**	0.122	0.112	**0.132**	0.095	0.084	**5.47E+00**	7.29E+00	7.51E+00
Gnegative	**0.648**	0.645	0.645	**0.431**	0.409	0.423	3.49E+01	3.51E+01	**3.42E+01**
Plant	**0.268**	0.243	0.260	**0.153**	0.132	0.144	4.53E+01	**4.48E+01**	4.57E+01
Water-quality	0.599	0.597	**0.600**	**0.556**	0.549	0.555	1.14E+01	**9.44E+00**	1.02E+01
Yeast	0.637	**0.643**	0.640	0.435	0.431	**0.436**	3.34E+01	**3.21E+01**	3.28E+01
Human	**0.358**	0.344	0.354	**0.163**	0.148	0.159	2.07E+02	**1.84E+02**	2.14E+02
Birds	0.452	**0.479**	0.461	0.321	0.330	**0.336**	1.41E+01	**1.21E+01**	1.31E+01
tmc2007–500	0.639	**0.657**	0.651	0.504	0.507	**0.511**	2.08E+02	**1.03E+02**	1.36E+02
Ohsumed	0.453	**0.491**	0.490	0.288	**0.349**	0.348	2.43E+02	**2.27E+02**	2.57E+02
Yahoo arts	0.122	**0.220**	0.216	0.128	0.147	**0.148**	**7.49E+01**	7.99E+01	8.48E+01
Genbase	**0.988**	**0.988**	0.987	**0.931**	0.927	0.929	1.46E+01	**7.95E+00**	9.51E+00
Medical	**0.815**	0.811	0.814	0.632	0.630	**0.636**	5.17E+01	**3.20E+01**	3.88E+01
NusWide	**0.272**	0.258	0.268	**0.147**	0.139	0.146	**2.94E+02**	5.18E+02	4.93E+02
Mediamill	0.536	0.561	**0.562**	0.289	0.294	**0.300**	**2.74E+02**	5.12E+02	4.95E+02
Stackex coffee	**0.314**	0.223	0.243	**0.627**	0.604	0.604	6.81E+01	**2.96E+01**	3.23E+01
CAL500	0.397	0.414	**0.429**	**0.181**	0.173	0.177	1.28E+02	**1.08E+02**	1.14E+02
Avg ranking	2.18	1.93	**1.90**	1.95	2.33	**1.73**	2.33	**1.47**	2.20

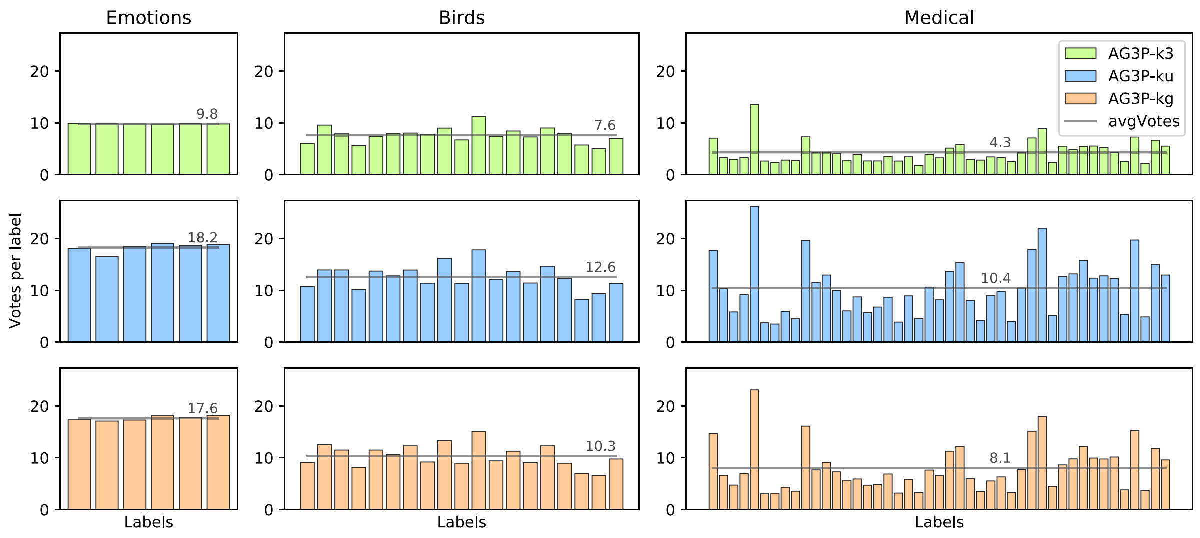

Fig. 12. Number of votes for each label in the final EMLC obtained by the different versions of AG3P-kEMLC.

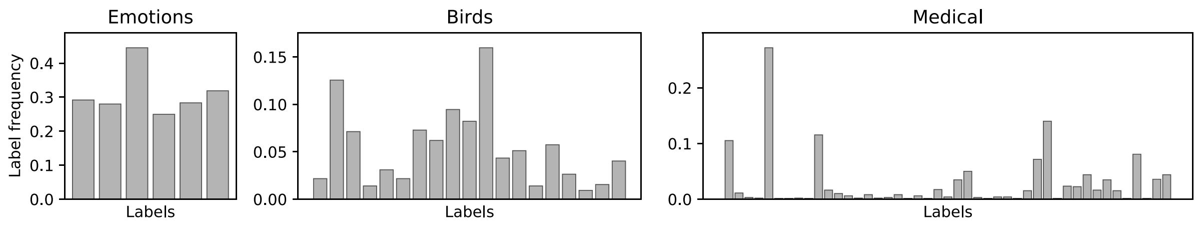

Fig. 13. Frequency of labels.

Table 8
Results of the Skillings-Mack's test comparing the state-of-the-art methods. Significant differences at 95% are marked in bold.

	Skillings-Mack's statistic	p-value
AHL	82.953	**1.13E−12**
SA	74.601	**4.37E−11**
ExF	85.973	**2.96E−13**
MiF	87.058	**1.83E−13**
MaF	75.287	**3.24E−11**

on average. The creation of simpler trees would mean not only that the complexity of the EMLC would be lower in testing, but also that it would be prevented from overfitting the training data. In general, AG3P-kEMLC needs only between 4 and 10 votes for the ensemble in Medical, and between 7 and 13 for Birds; therefore, the best ensemble does not use all classifiers in the pool (since the pool contains enough classifiers to have 20 votes per label on average), selecting those that perform better together.

5.1.5. *Runtime analysis*

In this section we analyze and compare the runtime of the three versions of AG3P-kEMLC. In Fig. 14 the runtime (in seconds) of each version is presented for the three datasets. The results are presented as boxplots. The conclusions attained from these plots are similar to the ones obtained in the study of the convergence. AG3P-k3 usually needs more generations to converge than the rest of versions, leading to a greater required runtime.

However, although the number of generations affects the required runtime, the number (and complexity) of the classifiers in the pool also determines the runtime of AG3P-kEMLC, since the building phase of these classifiers is a time-consuming process. In this way, AG3P-k3 is also the version that more classifiers has in the pool, spending more time than the rest in this phase, leading to be the most complex version (remember that in Emotions, AG3P-k3 is executed with $\overline{v} = 10$ and not 20). On the other hand, AG3P-ku allows a greater number of classifiers with a higher k value, so less classifiers need to be built to obtain 20 votes on average in the initial pool. This, together with the fact that it is also the one that needs a lower number of generations to converge, makes it to be the fastest version.

5.1.6. *Experimental comparison*

To finish with the study of the different proposed versions of AG3P-kEMLC, in this section an experimental comparison of the three methods is carried out. For that, all 20 datasets and 5 evaluation metrics are considered, and statistical tests are then performed to analyze whether any of the versions performs significantly better than the rest.

Table 7 presents the results of the three proposed versions for all metrics, and the average ranking of the methods is also included for each metric. In general, AG3P-ku is the one that obtains the best results in most cases in almost all metrics, except in MaF. AG3P-kg, although not being the best in as many cases as AG3P-ku, has a great consistency; for example, in MiF AG3P-kg is the best in 5 datasets, while AG3P-ku in 9, but the average ranking of AG3P-kg is the best. As for the runtime, AG3P-ku is the fastest in most methods, even more considering that in the first five datasets AG3P-k3 was executed using less classifiers in the pool than in the rest of cases, which biases its runtime in these cases.

In order to determine if there exist significant differences in performance among the different versions, the Skillings-Mack's test has been performed over the five evaluation metrics. It determines that there exist differences in the performance for SA (with p-value 1.64E−03), so the Holm's post-hoc test has been also carried out for this metric. The Holm's test determines that the performance of AG3P-ku is significantly better than AG3P-k3 with p-value 7.49E−04, while the p-value for AG3P-kg is 1.55E−01, so its performance is no significantly different to AG3P-ku. These results demonstrate that by using a variable value of k, allowing to model bigger subsets of labels, the performance is improved rather than just using fixed $k = 3$. Therefore, for further comparisons AG3P-k3 is not used, since it has been demonstrated that its performance is significantly worse than the rest of versions in at least one metric.

5.2. *Comparison vs state-of-the-art*

In this section, the performance of AG3P-ku and AG3P-kg is compared with other state-of-the-art EMLCs, in order to determine whether the method proposed in this paper outperforms other methods for MLC. Tables A.1, A.2, A.3, A.4 and A.5 present the results of all methods over all datasets and metrics, also including the average ranking for each method. Table A.6 also provides the required runtime of each of the methods. In all tables, methods that were not able to finish their execution due to memory overhead[5] are marked as DNF, and best results for each dataset are marked in bold.

The best results for each dataset are very spread for all metrics. AG3P-ku is the best in between one and six cases in each metric, while AG3P-kg is the best in between 4 and 6 cases in each metric. However, although not being the best methods in more datasets, their consistency in overall for all cases is widely demonstrated: AG3P-ku has the best average ranking in three metrics (and second in the rest) and AG3P-kg is the best in two metrics (and second in two more). Other EMLCs also demonstrate a good performance in some cases, but they perform much worse in the rest, leading to a poor consistency and a worse performance on average. For example, EMLS obtains competitive results in ExF and MaF, but in SA it is the worst method; RF-PCT obtains the best results in several datasets in almost all metrics, however, its average ranking is one of the worst always due to its weak performance in many other datasets; and ECC obtains good average ranking values in general, however, these values are usually far from the best, and in MaF it demonstrates much worse performance being the 8th method on average.

In order to determine if there exist significant differences among the methods, the Skillings-Mack's test is performed, and the results are presented in Table 8. The Skillings-Mack's test determined that in all metrics, there exist significant differences in the performance of the methods; then, the Holm's post-hoc test is also performed for all

[5] The experiments were performed on a machine with Intel Xeon E5645 Processor (6 × 2.40 GHz) and 24.GB RAM.

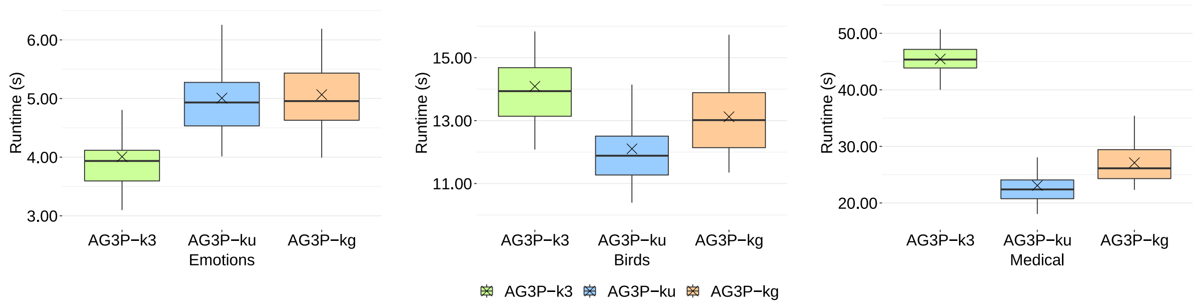

Fig. 14. Comparison of runtime of AG3P-kEMLC.

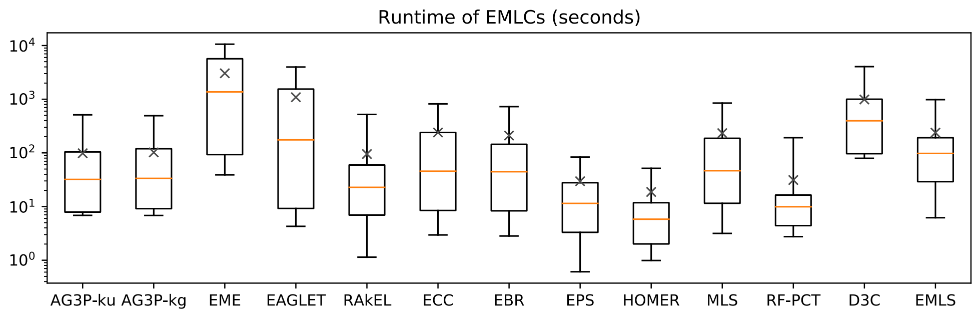

Fig. 15. Boxplots including the runtime in seconds of the EMLCs for all datasets (Y-axis is in logarithmic scale). The "×" symbol indicate the mean value.

metrics. The method with best average ranking in each metric is used as control algorithm, and the adjusted p-values are reported in Table 9.

Both AG3P-ku and AG3P-kg demonstrate to perform significantly better than state-of-the-art algorithms. It is noteworthy that for AHL, ExF, and MiF they perform significantly better than all the rest of methods, and for no metric any of them perform significantly worse than the control algorithm. Besides, as also introduced in Section 5.1.6 both versions of AG3P-kEMLC perform statistically the same in all cases.

5.3. Discussion

AG3P-kEMLC has demonstrated to have a good predictive performance, in any of the three versions that have been proposed. We observed that AG3P-k3, which uses a fixed $k = 3$ value for all classifiers as other state-of-the-art EMLCs, has the worst average ranking in 4 out of 5 metrics. Besides, in only one metric it performs significantly worse than the best version. Although it still has a great performance (obtains the best results in many cases), it is demonstrated that the fact of using k-labelsets of variable size in the ensemble improves the predictive performance. AG3P-k3 is also the most complex version, since it needs to build a higher number of classifiers in the pool and also needs more generations to converge, while AG3P-ku, which builds the lower number of classifiers in the pool is the fastest one.

Compared with state-of-the-art EMLCs, AG3P-ku and AG3P-kg obtain extremely good results. In almost all evaluation metrics, they are the two best methods (except AG3P-kg which is the 5th in SA), demonstrating their consistency independently of the evaluation metric. Then, in 3 of the metrics, both versions perform significantly better than all the rest of methods, being the only methods that do not perform significantly worse than any of the methods in any case.

According to the runtime, although not being the fastest methods, they obtain competitive results too, as observed in Fig. 15. AG3P-ku and AG3P-kg are the 5th and 6th methods (out of 13) according to the runtime in overall, being very similar to most state-of-the-art methods. It is noteworthy the drastic reduction in runtime compared to other EAs to build ensembles, such as EME and EAGLET. For a deeper analysis, the percentage of time that AG3P-ku and AG3P-kg reduce compared

Table 9
Adjusted p-values of Holm's post-hoc test comparing the state-of-the-art methods. The control algorithm in each case is indicated with "-". Significant differences with the control algorithm at 95% are marked in bold.

	AHL	SA	ExF	MiF	MaF
AG3P-ku	–	–	9.35E−01	–	6.01E−01
AG3P-kg	9.35E−01	1.76E 00	–	9.68E−01	–
EME	**5.30E−05**	9.42E−02	**8.90E−05**	**2.26E−04**	9.42E−02
EAGLET	**2.81E−02**	1.76E 00	**1.02E−02**	**1.35E−02**	2.36E−01
RAkEL	**3.48E−03**	9.42E−02	**4.99E−03**	**6.17E−03**	5.21E−01
ECC	**2.97E−02**	1.76E 00	**9.12E−03**	**1.35E−02**	**4.82E−03**
EBR	**1.40E−05**	**3.14E−02**	**3.00E−06**	**2.00E−06**	**1.53E−04**
EPS	**1.45E−04**	1.76E 00	**6.00E−06**	**1.00E−06**	**1.00E−05**
HOMER	**6.70E−05**	**6.93E−04**	**1.10E−04**	**6.10E−05**	9.42E−02
MLS	**6.00E−06**	**1.44E−03**	**6.00E−06**	**1.70E−05**	6.18E−02
RF-PCT	**0.00E 00**	**5.89E−04**	**0.00E 00**	**0.00E 00**	**0.00E 00**
D3C	**5.00E−06**	**1.63E−02**	**2.00E−06**	**3.30E−05**	**2.94E−03**
EMLS	**1.45E−04**	**1.00E−06**	**1.63E−03**	**3.46E−04**	6.01E−01

with other EAs is computed. Both versions of AG3P-kEMLC reduce the 91% of required runtime of EME, and over 47% the runtime required by EAGLET, on average. EAGLET is faster than AG3P-kEMLC in those datasets with only 6 labels; but in 13 out of 20 datasets, AG3P-kEMLC reduces more than 75% the runtime of EAGLET.

Therefore, our proposal not only obtains a very consistent predictive performance, being significantly better than the rest of state-of-the-art EMLCs, but also comprises a competitive solution in terms of complexity, considerably reducing the complexity of other similar approaches based on EAs.

6. Conclusions

In this paper, an auto-adaptive grammar-guided genetic programming method to build ensembles of multi-label classifiers has been proposed, called AG3P-kEMLC. Each of the multi-label classifiers that form the ensemble is focused on a subset of k labels. Unlike other EMLCs in the literature, it is able to include classifiers using different value of k. The ensemble is built in a tree-shape structure, where the leaves are the multi-label classifiers, and the internal nodes combine

Table A.1
Results of AG3P-kEMLC and state-of-the-art EMLCs for ↓AHL. Best results in each dataset are marked in bold. DNF values indicate that the method did not finish its execution.

	AG3P-ku	AG3P-kg	EME	EAGLET	RA*k*EL	ECC	EBR	EPS	HOMER	MLS	RF-PCT	D3C	EMLS
Emotions	0.449	0.450	0.504	0.463	0.492	0.454	0.490	0.471	0.544	0.560	**0.433**	0.440	0.482
Reuters1000	**0.824**	0.828	0.867	0.851	0.837	0.913	0.950	0.863	0.870	0.843	0.999	0.963	0.837
Guardian1000	**0.828**	0.837	0.884	0.883	0.867	0.917	0.953	0.844	0.852	0.877	0.999	0.970	0.834
Bbc1000	0.802	**0.790**	0.845	0.831	0.827	0.898	0.933	0.828	0.820	0.859	0.999	0.946	0.808
3s-inter3000	0.922	0.926	0.929	0.932	0.924	0.929	0.965	0.945	0.880	**0.859**	1.000	0.994	0.867
Gnegative	0.415	0.415	0.482	0.446	0.459	0.427	0.479	0.472	0.508	0.516	0.566	**0.394**	0.448
Plant	0.826	0.809	0.867	0.871	0.844	0.852	0.903	0.901	0.845	0.821	0.983	0.838	**0.770**
Water-quality	0.574	0.572	0.612	0.593	0.599	0.585	0.605	0.798	0.607	0.652	**0.556**	0.576	0.589
Yeast	0.488	0.493	0.516	0.500	0.517	0.491	0.514	0.504	0.566	0.600	**0.481**	0.501	0.534
Human	0.742	**0.730**	0.796	0.800	0.768	0.786	0.833	0.828	0.807	0.784	0.934	0.827	0.739
Birds	**0.393**	0.413	0.421	0.410	0.423	0.401	0.412	0.419	0.451	0.424	0.442	0.540	0.441
tmc2007–500	**0.451**	0.459	0.591	0.491	0.512	0.482	0.484	0.504	0.531	0.541	0.559	0.497	0.507
Ohsumed	**0.626**	0.631	0.701	0.695	0.694	0.709	0.710	0.718	0.757	0.727	0.947	0.821	0.837
Yahoo arts	**0.803**	0.814	0.954	0.939	0.965	0.825	0.971	0.812	0.811	0.981	1.000	0.958	0.936
Genbase	0.014	0.015	0.019	**0.012**	0.019	0.017	0.018	0.036	0.018	0.019	1.000	0.037	0.710
Medical	0.224	**0.223**	0.254	0.246	0.260	0.242	0.262	0.255	0.261	0.263	0.964	0.477	0.922
NusWide	0.757	0.751	0.755	0.753	0.769	**0.748**	**0.748**	0.756	0.824	0.816	0.763	0.901	0.980
Mediamill	0.577	**0.576**	0.598	0.598	0.605	0.580	0.582	0.605	0.648	0.656	0.579	0.752	0.939
Stackex coffee	0.860	0.842	DNF	**0.797**	0.814	0.916	0.907	0.913	DNF	0.853	1.000	0.816	0.988
CAL500	0.736	**0.725**	DNF	0.774	0.767	0.762	0.781	DNF	0.761	0.779	DNF	0.818	0.831
Avg ranking	**2.58**	2.68	8.13	5.78	6.68	5.58	8.50	7.78	8.03	8.73	10.00	8.80	7.78

Table A.2
Results of AG3P-kEMLC and state-of-the-art EMLCs for ↑SA. Best results in each dataset are marked in bold. DNF values indicate that the method did not finish its execution.

	AG3P-ku	AG3P-kg	EME	EAGLET	RA*k*EL	ECC	EBR	EPS	HOMER	MLS	RF-PCT	D3C	EMLS
Emotions	0.285	0.282	0.248	0.290	0.250	0.297	0.274	0.292	0.182	0.186	**0.337**	0.315	0.195
Reuters1000	0.129	**0.130**	0.111	0.123	0.129	0.064	0.040	0.115	0.078	0.112	0.001	0.031	0.068
Guardian1000	0.129	0.116	0.086	0.083	0.092	0.063	0.037	**0.130**	0.069	0.076	0.001	0.023	0.075
Bbc1000	0.148	**0.157**	0.120	0.145	0.134	0.086	0.057	0.142	0.102	0.088	0.001	0.043	0.086
3s-inter3000	0.049	0.047	0.040	0.041	0.037	0.050	0.025	0.044	0.042	**0.077**	0.000	0.006	0.041
Gnegative	0.536	0.528	0.487	0.523	0.493	**0.548**	0.497	0.513	0.421	0.397	0.426	0.530	0.386
Plant	0.145	**0.151**	0.113	0.113	0.127	0.140	0.089	0.095	0.094	0.109	0.017	0.118	0.073
Water-quality	0.013	0.013	0.014	0.016	0.013	0.017	0.016	0.015	0.004	0.008	0.020	**0.022**	0.015
Yeast	0.141	0.129	0.137	0.151	0.112	0.171	0.131	0.168	0.076	0.051	**0.181**	0.146	0.077
Human	**0.186**	**0.186**	0.159	0.163	0.167	0.174	0.141	0.140	0.105	0.122	0.060	0.102	0.076
Birds	0.502	0.481	0.496	0.500	0.490	**0.522**	0.516	0.515	0.457	0.491	0.505	0.327	0.435
tmc2007–500	0.241	0.233	0.230	0.233	0.212	0.240	0.229	0.240	0.167	0.186	0.164	**0.251**	0.194
Ohsumed	0.177	0.170	0.165	0.175	0.164	0.158	0.157	**0.182**	0.095	0.132	0.036	0.116	0.001
Yahoo arts	0.122	0.113	0.037	0.053	0.029	0.145	0.023	**0.153**	0.135	0.015	0.000	0.035	0.000
Genbase	0.972	0.969	0.966	**0.976**	0.965	0.968	0.967	0.937	0.970	0.967	0.000	0.926	0.000
Medical	0.671	0.665	0.649	0.658	0.641	0.671	0.650	**0.674**	0.654	0.637	0.022	0.222	0.000
NusWide	0.163	0.163	0.184	0.188	0.165	0.213	0.211	0.210	0.109	0.108	**0.222**	0.028	0.000
Mediamill	0.063	0.061	0.064	0.065	0.054	0.078	0.067	0.079	0.025	0.031	0.072	**0.223**	0.000
Stackex coffee	0.046	0.058	DNF	0.085	**0.088**	0.030	0.032	0.014	DNF	0.058	0.000	0.022	0.000
CAL500	**0.000**	**0.000**	DNF	**0.000**	**0.000**	**0.000**	**0.000**	DNF	**0.000**	**0.000**	DNF	**0.000**	**0.000**
Avg ranking	**4.15**	5.10	7.13	4.85	7.13	4.43	7.65	4.75	9.05	8.80	9.13	7.95	10.90

Table A.3
Results of AG3P-kEMLC and state-of-the-art EMLCs for ↑ExF. Best results in each dataset are marked in bold. DNF values indicate that the method did not finish its execution.

	AG3P-ku	AG3P-kg	EME	EAGLET	RA*k*EL	ECC	EBR	EPS	HOMER	MLS	RF-PCT	D3C	EMLS
Emotions	0.638	0.637	0.577	0.618	0.593	0.627	0.587	0.608	0.548	0.529	**0.643**	0.639	0.621
Reuters1000	0.193	0.187	0.141	0.159	0.175	0.094	0.054	0.145	0.153	0.173	0.001	0.040	**0.200**
Guardian1000	0.188	0.179	0.126	0.129	0.147	0.090	0.051	0.164	0.178	0.141	0.001	0.032	**0.202**
Bbc1000	0.216	0.228	0.167	0.178	0.186	0.107	0.071	0.182	0.209	0.161	0.001	0.057	**0.234**
3s-inter3000	0.088	0.083	0.082	0.077	0.090	0.079	0.038	0.059	0.149	0.167	0.000	0.006	**0.171**
Gnegative	0.602	0.604	0.529	0.565	0.558	0.581	0.530	0.533	0.518	0.515	0.437	**0.633**	0.615
Plant	0.184	0.205	0.140	0.134	0.166	0.150	0.100	0.100	0.179	0.205	0.017	0.178	**0.298**
Water-quality	0.570	0.571	0.528	0.549	0.544	0.556	0.534	0.298	0.538	0.489	**0.586**	0.562	0.551
Yeast	**0.625**	0.622	0.593	0.609	0.599	0.616	0.597	0.602	0.552	0.526	**0.625**	0.609	0.594
Human	0.284	0.299	0.220	0.213	0.254	0.228	0.176	0.182	0.227	0.251	0.069	0.202	**0.338**
Birds	**0.644**	0.627	0.610	0.622	0.609	0.626	0.615	0.607	0.585	0.607	0.578	0.516	0.606
tmc2007–500	**0.650**	0.643	0.591	0.602	0.582	0.610	0.611	0.583	0.574	0.554	0.536	0.590	0.595
Ohsumed	**0.446**	0.442	0.349	0.354	0.359	0.339	0.339	0.319	0.303	0.325	0.059	0.202	0.251
Yahoo arts	**0.229**	0.218	0.049	0.065	0.038	0.188	0.031	0.203	0.210	0.021	0.000	0.046	0.115
Genbase	0.990	0.989	0.985	**0.991**	0.985	0.988	0.986	0.972	0.986	0.985	0.000	0.974	0.433
Medical	0.811	**0.815**	0.778	0.785	0.773	0.787	0.768	0.769	0.768	0.770	0.041	0.640	0.141
NusWide	0.276	**0.284**	0.272	0.273	0.260	0.268	0.270	0.259	0.208	0.219	0.245	0.137	0.037
Mediamill	0.548	**0.550**	0.521	0.522	0.516	0.538	0.540	0.509	0.478	0.466	0.542	0.259	0.113
Stackex coffee	0.177	0.199	DNF	**0.260**	0.226	0.105	0.117	0.116	DNF	0.181	0.000	0.259	0.023
CAL500	0.411	**0.425**	DNF	0.363	0.372	0.377	0.352	DNF	0.380	0.356	DNF	0.302	0.286
Avg ranking	2.58	**2.48**	7.85	5.93	6.45	6.13	8.75	8.55	7.75	8.60	10.18	8.88	6.90

Table A.4
Results of AG3P-kEMLC and state-of-the-art EMLCs for ↑MiF. Best results in each dataset are marked in bold. DNF values indicate that the method did not finish its execution.

	AG3P-ku	AG3P-kg	EME	EAGLET	RA*k*EL	ECC	EBR	EPS	HOMER	MLS	RF-PCT	D3C	EMLS
Emotions	0.674	0.674	0.630	0.665	0.637	0.671	0.649	0.654	0.600	0.582	**0.694**	0.693	0.652
Reuters1000	**0.245**	0.239	0.186	0.209	0.220	0.139	0.087	0.189	0.186	0.216	0.002	0.069	0.244
Guardian1000	**0.246**	0.236	0.185	0.192	0.205	0.135	0.085	0.213	0.230	0.194	0.003	0.059	0.242
Bbc1000	0.280	**0.290**	0.230	0.239	0.248	0.154	0.114	0.238	0.250	0.211	0.001	0.101	0.276
3s-inter3000	0.122	0.112	0.115	0.107	0.125	0.120	0.062	0.082	0.194	0.196	0.000	0.010	**0.204**
Gnegative	0.645	0.645	0.604	0.638	0.613	0.646	0.625	0.612	0.556	0.556	0.577	**0.662**	0.589
Plant	0.243	0.260	0.200	0.196	0.224	0.210	0.159	0.153	0.227	0.245	0.031	0.245	**0.304**
Water-quality	0.597	0.600	0.558	0.578	0.573	0.585	0.565	0.300	0.569	0.516	**0.614**	0.594	0.583
Yeast	0.643	0.640	0.622	0.636	0.622	0.640	0.625	0.629	0.575	0.549	**0.651**	0.635	0.606
Human	0.344	**0.354**	0.291	0.286	0.314	0.298	0.250	0.255	0.277	0.298	0.112	0.260	0.347
Birds	**0.479**	0.461	0.416	0.439	0.423	0.437	0.414	0.379	0.390	0.422	0.334	0.431	0.471
tmc2007–500	**0.657**	0.651	0.614	0.624	0.606	0.632	0.632	0.606	0.586	0.581	0.560	0.617	0.606
Ohsumed	**0.491**	0.490	0.418	0.424	0.427	0.413	0.412	0.372	0.345	0.389	0.083	0.264	0.276
Yahoo arts	**0.220**	0.216	0.063	0.082	0.051	0.180	0.044	0.197	0.200	0.029	0.000	0.059	0.118
Genbase	0.988	0.987	0.986	**0.990**	0.985	0.987	0.986	0.956	0.985	0.986	0.000	0.970	0.450
Medical	0.811	**0.814**	0.809	0.807	0.804	0.809	0.801	0.771	0.791	0.802	0.077	0.615	0.145
NusWide	0.258	**0.268**	0.216	0.208	0.213	0.156	0.159	0.136	0.183	0.204	0.068	0.168	0.040
Mediamill	0.561	**0.562**	0.536	0.538	0.532	0.552	0.555	0.519	0.492	0.484	0.552	0.105	0.117
Stackex coffee	0.223	0.243	DNF	**0.303**	0.267	0.146	0.159	0.138	DNF	0.235	0.000	0.291	0.023
CAL500	0.414	**0.429**	DNF	0.365	0.375	0.379	0.354	DNF	0.383	0.360	DNF	0.304	0.289
Avg ranking	**2.50**	2.55	7.58	5.85	6.40	6.00	8.88	9.00	7.95	8.38	10.33	8.18	7.40

Table A.5
Results of AG3P-kEMLC and state-of-the-art EMLCs for ↑MaF. Best results in each dataset are marked in bold. DNF values indicate that the method did not finish its execution.

	AG3P-ku	AG3P-kg	EME	EAGLET	RA*k*EL	ECC	EBR	EPS	HOMER	MLS	RF-PCT	D3C	EMLS
Emotions	0.660	0.663	0.616	0.651	0.625	0.623	0.631	0.637	0.592	0.577	0.664	**0.668**	0.648
Reuters1000	0.211	0.209	0.157	0.169	0.178	0.093	0.058	0.139	0.153	0.164	0.001	0.052	**0.215**
Guardian1000	0.199	0.190	0.163	0.167	0.177	0.090	0.056	0.160	**0.219**	0.147	0.002	0.056	0.212
Bbc1000	0.239	**0.254**	0.211	0.210	0.219	0.108	0.082	0.199	0.233	0.174	0.001	0.089	0.244
3s-inter3000	0.095	0.084	0.085	0.076	0.104	0.087	0.044	0.061	**0.160**	0.159	0.000	0.007	**0.160**
Gnegative	0.409	0.423	0.396	0.425	0.408	0.381	0.346	0.343	0.366	0.333	0.233	**0.472**	0.382
Plant	0.132	0.144	0.106	0.107	0.121	0.092	0.076	0.069	0.139	0.157	0.010	0.206	**0.208**
Water-quality	0.549	**0.555**	0.502	0.521	0.522	0.528	0.503	0.171	0.524	0.469	0.536	0.532	0.549
Yeast	0.431	0.436	0.393	0.408	0.409	0.403	0.381	0.377	0.394	0.391	0.370	0.438	**0.462**
Human	0.148	0.159	0.124	0.124	0.136	0.113	0.094	0.087	0.139	0.155	0.025	0.193	**0.197**
Birds	0.330	0.336	0.270	0.290	0.284	0.268	0.250	0.228	0.289	0.305	0.203	**0.376**	0.367
tmc2007–500	0.507	**0.511**	0.463	0.461	0.463	0.476	0.479	0.343	0.442	0.445	0.152	0.358	0.505
Ohsumed	**0.349**	0.348	0.253	0.255	0.261	0.228	0.228	0.222	0.220	0.238	0.035	0.128	0.306
Yahoo arts	0.147	**0.148**	0.115	0.120	0.113	0.125	0.109	0.138	0.132	0.106	0.092	0.110	0.132
Genbase	0.927	0.929	0.930	**0.933**	0.926	0.925	0.922	0.762	0.915	0.930	0.215	0.911	0.839
Medical	0.630	0.636	**0.644**	0.631	0.639	0.637	0.636	0.603	0.604	0.640	0.322	0.504	0.430
NusWide	0.139	0.146	**0.156**	0.145	0.152	0.141	0.142	0.137	0.125	0.153	0.136	0.082	0.064
Mediamill	0.294	0.300	0.308	0.307	0.311	0.301	0.310	0.277	**0.344**	0.304	0.284	0.140	0.172
Stackex coffee	0.604	0.604	DNF	**0.633**	0.631	0.618	0.619	0.607	DNF	0.628	0.597	0.576	0.158
CAL500	0.173	0.177	DNF	0.148	0.163	0.139	0.137	DNF	0.185	0.158	DNF	0.145	**0.203**
Avg ranking	4.55	**3.70**	6.68	6.03	5.38	7.93	9.03	9.75	6.65	6.70	11.30	8.13	4.98

Table A.6
Runtime of AG3P-kEMLC and state-of-the-art EMLCs. Best results in each dataset are marked in bold. DNF values indicate that the method did not finish its execution.

	AG3P-ku	AG3P-kg	EME	EAGLET	RA*k*EL	ECC	EBR	EPS	HOMER	MLS	RF-PCT	D3C	EMLS
Emotions	5.01E+0	5.07E+0	1.24E+1	1.56E+0	8.44E−1	1.79E+0	1.88E+0	**6.10E−1**	8.63E−1	2.81E+0	4.86E+0	5.00E+1	6.20E+0
Reuters1000	6.87E+0	6.82E+0	5.09E+1	4.28E+0	6.11E+0	6.94E+0	7.03E+0	4.72E+0	**1.96E+0**	1.11E+1	2.75E+0	7.93E+1	2.14E+1
Guardian1000	6.85E+0	7.17E+0	5.37E+1	4.33E+0	6.14E+0	7.35E+0	7.53E+0	4.98E+0	**2.03E+0**	1.16E+1	2.78E+0	8.31E+1	2.24E+1
Bbc1000	7.73E+0	8.01E+0	5.96E+1	5.30E+0	8.41E+0	9.48E+0	1.03E+1	6.78E+0	**2.26E+0**	1.60E+1	3.06E+0	9.71E+1	2.99E+1
3s-inter3000	7.29E+0	7.51E+0	1.04E+2	6.28E+0	7.21E+0	8.82E+0	8.60E+0	4.00E+0	**2.24E+0**	1.24E+1	3.02E+0	9.53E+1	3.05E+1
Gnegative	3.51E+1	3.42E+1	5.83E+2	3.72E+1	3.63E+1	8.23E+1	8.13E+1	2.22E+1	**8.72E+0**	9.22E+1	1.02E+1	4.52E+2	1.87E+2
Plant	4.48E+1	4.57E+1	1.44E+3	2.01E+2	5.50E+1	9.81E+1	1.12E+2	2.71E+1	1.09E+1	1.19E+2	**1.05E+1**	3.32E+2	1.61E+2
Water-quality	9.44E+0	1.02E+1	3.90E+1	1.02E+1	1.14E+0	2.96E+0	2.83E+0	**9.17E−1**	9.88E−1	3.16E+0	1.09E+1	1.23E+2	5.94E+0
Yeast	3.21E+1	3.28E+1	4.15E+2	1.49E+2	2.43E+1	3.76E+1	5.60E+1	1.21E+1	**4.81E+0**	4.83E+1	4.43E+1	5.24E+2	1.19E+2
Human	1.84E+2	2.14E+2	5.55E+3	1.42E+3	2.39E+2	6.39E+2	5.93E+2	8.35E+1	4.37E+1	5.72E+2	**3.83E+1**	1.91E+3	9.82E+2
Birds	1.21E+1	1.31E+1	2.13E+2	7.10E+1	8.57E+0	1.03E+1	8.94E+0	**1.23E+0**	1.51E+0	1.12E+1	9.64E+0	8.66E+1	2.67E+1
tmc2007–500	1.03E+2	1.36E+2	6.05E+3	3.95E+3	6.79E+2	2.02E+3	1.92E+3	4.57E+1	1.78E+2	2.10E+3	**1.15E+1**	3.38E+3	1.61E+3
Ohsumed	2.27E+2	2.57E+2	6.16E+3	3.98E+3	5.21E+2	8.19E+2	7.28E+2	2.25E+2	5.16E+1	8.47E+2	**1.13E+1**	4.05E+3	6.03E+2
Yahoo arts	7.99E+1	8.48E+1	2.42E+3	8.67E+2	9.72E+1	2.47E+2	2.23E+2	7.37E+1	1.01E+1	2.47E+2	**6.36E+0**	4.19E+3	1.56E+2
Genbase	7.95E+0	9.51E+0	1.30E+3	8.30E+1	2.28E+0	3.58E+0	4.04E+0	**6.67E−1**	1.44E+0	5.32E+0	2.63E+0	1.91E+2	4.82E+1
Medical	3.20E+1	3.88E+1	8.12E+3	5.68E+2	5.21E+1	9.66E+1	7.91E+1	1.08E+1	1.45E+1	1.04E+2	**5.15E+0**	3.41E+2	2.06E+2
NusWide	5.18E+2	4.93E+2	1.06E+4	1.79E+3	6.16E+1	3.82E+2	1.55E+2	1.78E+1	**6.81E+0**	1.87E+2	2.19E+2	5.81E+2	2.98E+2
Mediamill	5.12E+2	4.95E+2	1.16E+4	1.51E+3	5.88E+1	2.38E+2	1.41E+2	2.12E+1	**8.11E+0**	1.87E+2	1.92E+2	2.03E+3	1.04E+2
Stackex coffee	2.96E+1	3.23E+1	DNF	1.65E+3	2.13E+1	3.31E+1	3.04E+1	**5.02E−1**	DNF	4.44E+1	5.21E+0	4.79E+2	9.15E+1
CAL500	1.08E+2	1.14E+2	DNF	5.49E+3	1.81E+1	5.35E+1	3.32E+1	DNF	**4.32E+0**	4.48E+1	DNF	6.97E+2	5.62E+1
Avg ranking	6.00	6.70	12.58	8.90	4.45	7.40	6.80	2.85	**2.33**	8.15	3.90	11.95	9.00

the prediction of children nodes, thus avoiding the typical ensemble structure in MLC where all members have the same weight in the final prediction.

The auto-adaptive process embedded in AG3P-kEMLC avoids one of the main drawbacks of EAs, i.e., the need to tune a plethora of hyper-parameters to enhance the performance of the method. In this way, AG3P-kEMLC automatically adapts the crossover and mutation probabilities depending on whether the population improves on average or not, thus favoring the exploitation of current individuals (increasing the crossover probability) or the exploration of the search space (increasing the mutation probability). Furthermore, a stop criterion based on the non-improvement of the best individual is also set, reducing the required runtime and avoiding overfitting.

Three versions of AG3P-kEMLC have been proposed. AG3P-k3 uses a fixed value of $k = 3$ in all base classifiers, as other methods in the literature. On the other hand, AG3P-ku and AG3P-kg allow both the use of k-labelsets of different size, in the range $k \in [3, q/2]$; while the former gives the same probability to each value of k, the latter gives higher probability to include smaller k-labelsets in the initial pool. The experimental results demonstrate that AG3P-k3 is the most complex version, due to the high number of classifiers that it has to built in the pool, also obtaining the worst results according to the predictive performance. On the other hand, AG3P-ku and AG3P-kg have demonstrated to perform significantly better than the state-of-the-art EMLCs. The fact of using an ensemble shape that is able to adapt to each specific problem (both in size and in the weight given to each classifier) as well as the auto-adaptive process, lead AG3P-kEMLC to obtain a very good predictive performance and consistency, achieving very good results on average for all evaluation metrics and datasets. Besides, its complexity is similar to other state-of-the-art EMLCs, much reducing the required runtime of other similar approaches to build EMLCs based on evolutionary algorithms.

Since the proposed method has achieved good and promising results, we still intend to look for ways to enhance its performance in the future. One of the lines of future work would be not only to use a heuristic to create the initial pool (as the modes that AG3P-ku and AG3P-kg use to select the k-labelsets based on the correlation among labels), but to evolve the initial pool towards accurate and diverse classifiers that would then be combined in the ensemble. Other aggregation approaches to combine the predictions in non-leaf nodes, like the combination of confidences instead of bipartitions, should be further studied. Other approach that might be studied in order to improve the auto-adaptability process would be looking at the fitness of the best individuals (i.e., check whether good genes are maintained in the population) instead of the average fitness of the whole population. Finally, the use of techniques to prevent the algorithm from being stuck in local optima, such as population restarts, might be analyzed.

CRediT authorship contribution statement

Jose M. Moyano: Formal analysis, Investigation, Software, Validation, Writing – original draft, Writing – review & editing. **Sebastián Ventura:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the Spanish Ministry of Science and Innovation and the European Regional Development Fund, project PID2020-115832GB-I00, and by the University of Córdoba, Spain and Andalusian Council of Economy, Knowledge, Business, and Universities, Spain, project 1262678-F. The authors are also grateful to the University of Córdoba, Spain / CBUA for funding open access publication charges.

Appendix. Tables with results

This appendix includes the detailed results of the proposed AG3P-kEMLC method and the rest of state-of-the-art EMLCs. Tables A.1, A.2, A.3, A.4, A.5 and A.6 include the results for AHL, SA, ExF, MiF, MaF, and runtime, respectively.

References

[1] O. Maimon, L. Rokach, Data Mining and Knowledge Discovery Handbook, Springer, 2005.
[2] I. Allaouzi, M.B. Ahmed, A novel approach for multi-label chest X-ray classification of common thorax diseases, IEEE Access 7 (2019) 64279–64288.
[3] H. Shao, G. Li, G. Liu, Y. Wang, Symptom selection for multi-label data of inquiry diagnosis in traditional Chinese medicine, Sci. China Inf. Sci. 56 (5) (2013) 1–13.
[4] T. Gong, B. Liu, Q. Chu, N. Yu, Using multi-label classification to improve object detection, Neurocomputing 370 (2019) 174–185.
[5] G. Nasierding, G. Tsoumakas, A.Z. Kouzani, Clustering based multi-label classification for image annotation and retrieval, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11-14 October 2009, 2009, pp. 4514–4519.
[6] K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas, Multi-label classification of music into emotions, in: Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008), 2008, pp. 325–330.
[7] J. Yu, L. Marujo, J. Jiang, P. Karuturi, W. Brendel, Improving multi-label emotion classification via sentiment classification with dual attention transfer network, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, ACL, 2018, pp. 1097–1102.
[8] F. Charte, M.J. del Jesus, A.J. Rivera, Multilabel Classification: Problem Analysis, Metrics and Techniques, Springer, 2016.
[9] M.-L. Zhang, Z.-H. Zhou, A review on multi-label learning algorithms, IEEE Trans. Knowl. Data Eng. 26 (8) (2014) 1819–1837.
[10] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, An evolutionary approach to build ensembles of multi-label classifiers, Inf. Fusion 50 (2019) 168–180.
[11] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, Combining multi-label classifiers based on projections of the output space using Evolutionary algorithms, Knowl.-Based Syst. 196 (2020) 105770.
[12] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. 85 (3) (2011) 335–359.
[13] G. Tsoumakas, I. Katakis, I. Vlahavas, Random k-labelsets for multi-label classification, IEEE Trans. Knowl. Data Eng. 23 (7) (2011) 1079–1089.
[14] E. Gibaja, S. Ventura, Multi-label learning: a review of the state of the art and ongoing research, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 4 (6) (2014) 411–444.
[15] G. Madjarov, D. Kocev, D. Gjorgjevikj, S.D. zeroski, An extensive experimental comparison of methods for multi-label learning, Pattern Recognit. 45 (9) (2012) 3084–3104.
[16] J. Read, B. Pfahringer, 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.
[17] O. Sagi, L. Rokach, Ensemble learning: A survey, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 8 (4) (2018) e1249.
[18] Y. Bi, The impact of diversity on the accuracy of evidential classifier ensembles, Internat. J. Approx. Reason. 53 (4) (2012) 584–607.
[19] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, Tree-shaped ensemble of multi-label classifiers using grammar-guided genetic programming, in: 2020 IEEE Congress on Evolutionary Computation, CEC 2020, 2020, pp. 1–8.
[20] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, Review of ensembles of multi-label classifiers: Models, experimental study and prospects, Inf. Fusion 44 (2018) 33–45.
[21] O. Gharroudi, H. Elghazel, 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.
[22] G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, I. Vlahavas, Correlation-based pruning of stacked binary relevance models for multi-label learning, in: 1st International Workshop on Learning from Multi-Label Data (MLD'09), 2009, pp. 101–116.

[23] C. Lin, W. Chen, C. Qiu, Y. Wu, S. Krishnan, Q. Zou, LibD3C: Ensemble classifiers with a clustering and dynamic selection strategy, Neurocomputing 123 (2014) 424–435.
[24] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Ensembles of multi-objective decision trees, in: European Conference on Machine Learning, 2007, pp. 624–631.
[25] M. Boutell, J. Luo, X. Shen, C. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (2004) 1757–1771.
[26] G. Tsoumakas, I. Katakis, 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.
[27] A.E. Eiben, J.E. Smith, et al., Introduction To Evolutionary Computing, Vol. 53, Springer, 2003.
[28] J.R. Koza, Genetic Programming: On the Programming of Computers By Means of Natural Selection, MIT Press, 1992.
[29] J.R. Koza, Introduction to genetic programming: tutorial, in: Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation, 2008, pp. 2299–2338.
[30] R.I. McKay, N.X. Hoai, P.A. Whigham, Y. Shan, M. O'neill, Grammar-based genetic programming: a survey, Genet. Program. Evol. Mach. 11 (3) (2010) 365–396.
[31] P.A. Whigham, et al., Grammatically-based genetic programming, in: Proceedings of the Workshop on Genetic Programming: From Theory To Real-World Applications, 1995, pp. 33–41.
[32] M. Sipser, Context-free languages, in: Introduction to the Theory of Computation, Cengage Learning, 2012, pp. 101–163.
[33] P.G. Espejo, S. Ventura, F. Herrera, A survey on the application of genetic programming to classification, IEEE Trans. Syst. Man Cybern. C 40 (2) (2009) 121–144.
[34] A. Cano, A. Zafra, E.L. Gibaja, S. Ventura, A grammar-guided genetic programming algorithm for multi-label classification, in: European Conference on Genetic Programming, Springer, 2013, pp. 217–228.
[35] A.G. de Sá, A.A. Freitas, G.L. Pappa, Automated selection and configuration of multi-label classification algorithms with grammar-based genetic programming, in: International Conference on Parallel Problem Solving from Nature, Springer, 2018, pp. 308–320.
[36] A.G. de Sá, C.G. Pimenta, G.L. Pappa, A.A. Freitas, A robust experimental evaluation of automated multi-label classification methods, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, 2020, pp. 175–183.
[37] J. Cohen, P. Cohen, S.G. West, L.S. Aiken, Applied Multiple Regression / Correlation Analysis for the Behavioral Sciences, Psychology Press, 2002.
[38] S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, in: Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2004, pp. 22–30.
[39] G. Tsoumakas, I. Katakis, I. Vlahavas, Mining multi-label data, in: Data Mining and Knowledge Discovery Handbook, Part 6, Springer, 2010, pp. 667–685.
[40] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, Inform. Sci. 250 (2013) 113–141.
[41] E. Gibaja, S. Ventura, A tutorial on multilabel learning, ACM Comput. Surv. 47 (3) (2015).
[42] R.B. Pereira, A. Plastino, B. Zadrozny, L.H. Merschmann, Correlation analysis of performance measures for multi-label classification, Inf. Process. Manage. 54 (3) (2018) 359–369.
[43] J.M. Luna, J.R. Romero, C. Romero, S. Ventura, On the use of genetic programming for mining comprehensible rules in subgroup discovery, IEEE Trans. Cybern. 44 (12) (2014) 2329–2341.
[44] J.M. Luna, J.R. Romero, C. Romero, S. Ventura, Reducing gaps in quantitative association rules: A genetic programming free-parameter algorithm, Integr. Comput.-Aided Eng. 21 (4) (2014) 321–337.
[45] J. Su, H. Zhang, A fast decision tree learning algorithm, in: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI'06, 2006, pp. 500–505.
[46] J.M. Moyano, E.L. Gibaja, S. Ventura, MLDA: A tool for analyzing multi-label datasets, Knowl.-Based Syst. 121 (2017) 1–3.
[47] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, MLeNN: a first approach to heuristic multilabel undersampling, in: International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2014, pp. 1–9.
[48] N. Ghamrawi, A. McCallum, Collective multi-label classification, in: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005, pp. 195–200.
[49] S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás, JCLEC: a Java framework for evolutionary computation, Soft Comput. 12 (4) (2008) 381–392.
[50] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, I. Vlahavas, Mulan: A Java library for multi-label learning, J. Mach. Learn. Res. 12 (2011) 2411–2414.
[51] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: An update, SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[52] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., 1993.
[53] L. Rokach, A. Schclar, E. Itach, Ensemble methods for multi-label classification, Expert Syst. Appl. 41 (16) (2014) 7507–7523.
[54] M. Chatfield, A. Mander, The Skillings–Mack test (Friedman test when there are missing data), Stata J. 9 (2) (2009) 299–305.
[55] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70.
[56] P. Srisuradetchai, Skillings.Mack: The Skillings-Mack Test Statistic for Block Designs with Missing Observations. https://CRAN.R-project.org/package=Skillings.Mack. (Last access 07 October 2020).
[57] I. Triguero, S. González, J.M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M.J. del Jesús, L. Sánchez, F. Herrera, KEEL 3.0: an open source software for multi-stage analysis in data mining, Int. J. Comput. Intell. Syst. 10 (1) (2017) 1238–1249.

Jose M. Moyano obtained his Ph.D. in Computer Science from the University of Córdoba, Córdoba, Spain, and the Virginia Commonwealth University, Richmond, VA, USA, in 2020. He also received his B.Sc. and M.Sc. degrees in Computer Science from the University of Córdoba, in 2014 and 2016, respectively. He is member of the Knowledge Discovery and Intelligent Systems Research Group of the University of Córdoba, and its research is mainly focused on ensemble methods for multi-label classification.

Sebastián Ventura is currently a Full Professor in the Department of Computer Science and Numerical Analysis at the University of Córdoba, where he heads the Knowledge Discovery and Intelligent Systems Research Laboratory. He received his B.Sc. and Ph.D. degrees in sciences from the University of Córdoba, Spain, in 1989 and 1996, respectively. He has published more than 170 papers in journals and scientific conferences, and he has edited three books and several special issues in international journals. He has also been engaged in 14 research projects (being the coordinator of six of them) supported by the Spanish and Andalusian governments and the European Union. His main research interests are in the fields of soft-computing, machine learning, data mining, and their applications. Dr. Ventura is a senior member of the IEEE Computer, the IEEE Computational Intelligence and the IEEE Systems, Man and Cybernetics Societies, as well as the Association of Computing Machinery (ACM).