UNIVERSIDADE FEDERAL FLUMINENSE

EDUARDO CORRÊA GONÇALVES

NOVEL CLASSIFIER CHAIN METHODS FOR MULTI-LABEL CLASSIFICATION BASED ON GENETIC ALGORITHMS

NITERÓI

2015

UNIVERSIDADE FEDERAL FLUMINENSE

EDUARDO CORRÊA GONÇALVES

NOVEL CLASSIFIER CHAIN METHODS FOR MULTI-LABEL CLASSIFICATION BASED ON GENETIC ALGORITHMS

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Algoritmos e Otimização.

Orientador:
ALEXANDRE PLASTINO

Coorientador:
ALEX A. FREITAS

NITERÓI

2015

EDUARDO CORRÊA GONÇALVES

NOVEL CLASSIFIER CHAIN METHODS FOR MULTI-LABEL CLASSIFICATION BASED ON GENETIC ALGORITHMS

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Algoritmos e Otimização.

Aprovada em 15 de Setembro de 2015.

BANCA EXAMINADORA

Prof. Alexandre Plastino – Orientador, UFF

Prof. Simone de Lima Martins, UFF

Prof. José Viterbo Filho, UFF

Prof. Bianca Zadrozny, IBM Research Brasil

Prof. Eduardo Soares Ogasawara, CEFET-RJ

Niterói

2015

I take pleasure in dedicating this thesis to Glauce, Antonio, Ana and Joaquim.

Resumo

Classificação multirrótulo pode ser definida como a tarefa de associar automaticamente objetos a múltiplas categorias com base nas características dos mesmos. Existem muitas aplicações modernas e importantes para a tarefa como, por exemplo, categorização de músicas (associar músicas a diversos gêneros musicais) e genômica funcional (determinar as múltiplas funções biológicas de genes e proteínas). Proposto em 2009, o modelo denominado *classifier chains* (CC) se tornou um dos mais influentes métodos para classificação multirrótulo, destacando-se por sua abordagem simples e eficaz para explorar a questão da dependência entre rótulos. O método básico envolve o treinamento de q classificadores monorrótulo binários, onde q representa o número de rótulos. Cada um deles é responsável unicamente pela classificação de um rótulo de classe específico. Esses q classificadores são ligados em uma estrutura de cadeia, de maneira que cada classificador binário torna-se capaz de considerar os rótulos preditos pelos classificadores anteriores como informação adicional em tempo de classificação. O método CC é considerado um dos mais eficazes para classificação multirrótulo, demonstrando-se competitivo com o estado da arte nesta área. Entretanto, ele possui duas desvantagens: (i) determina a ordem dos rótulos na cadeia de maneira aleatória, embora diferentes ordenações possam influir de maneira significativa na acurácia do modelo; (ii) obriga todos os rótulos a participar da cadeia, mesmo que alguns contenham informação redundante e/ou irrelevante para a previsão dos vários outros rótulos.

O objetivo principal deste trabalho é a proposta de duas novas técnicas capazes de aprimorar a eficácia dos classificadores multirrótulo em cadeia através da busca por uma ordenação de cadeia otimizada (isto é, determinar uma ordenação capaz de aumentar a acurácia do classificador). Essas duas técnicas, denominadas GACC e GA-PartCC, são baseadas em Algoritmos Genéticos (AGs), que correspondem a métodos de busca e otimização inspirados no princípio da seleção natural. Uma das estratégias propostas (GA-PartCC) é capaz de avaliar cadeias de rótulos que variam não apenas na ordenação, mas também em comprimento. Os AGs propostos foram avaliados, em termos de desempenho preditivo, em diferentes bases de dados. Os resultados dos experimentos computacionais demonstraram que, em geral, os AGs propostos produzem resultados competitivos em relação a outros métodos de classificação multirrótulo em cadeia propostos na literatura.

Palavras-chave: Classificação multirrótulo, cadeias de classificadores, algoritmos genéticos.

Abstract

Multi-label classification (MLC) is the task of automatically assigning an object to multiple categories based on its characteristics. There are many important and modern applications of MLC such as music categorization (associating songs to various music genres) and functional genomics (determining the multiple biological functions of genes and proteins). First proposed in 2009, the classifier chains model (CC) has become one of the most influential methods for MLC. It is distinguished by its simple and effective approach to exploit label dependencies. The CC method involves the training of q single-label binary classifiers, where q represents the number of labels. Each one is solely responsible for classifying a specific label. These q classifiers are linked in a chain in random order, such that each binary classifier is able to consider the labels predicted by the previous ones as additional information at classification time. CC is considered one of the most effective MLC methods, in the sense that it has proved to be competitive with state-of-the-art techniques. However, the basic CC model suffers from two major drawbacks: (i) it decides the label sequence randomly, although different label sequences might have a strong effect on the predictive accuracy of the model; (ii) it forces all labels to be present in the chain, despite the fact that some of them might carry redundant and/or irrelevant information to predict the various other labels.

The main contribution of this thesis is the proposal of two novel techniques that enhance the effectiveness of multi-label chain classifiers by searching for a single optimized label sequence (i.e., a label sequence that leads to an improvement on the predictive accuracy of the CC model). These two techniques, named GACC and GA-PartCC, are based on Genetic Algorithms (GAs) which are search and optimization methods inspired by the principle of natural selection. One of the proposed strategies (GA-PartCC) is capable of evaluating chain sequences that vary not only in the ordering but also in length. The proposed GAs are evaluated, in terms of predictive performance, on diverse benchmark datasets. Overall, the results of our computational experiments have shown that the proposed GAs are competitive with well-known alternative multi-label classifier chain methods.

Keywords: Multi-label classification, classifier chains, genetic algorithms.

List of Figures

List of Tables

List of Abbreviations

BCC	:	Bayesian Chain Classifier
BR	:	Binary Relevance;
CC	:	Classifier Chains;
DRC	:	Donor-Receptor Crossover;
EA	:	Evolutionary Algorithm;
GA	:	Genetic Algorithm;
GACC	:	Genetic Algorithm for Optimizing Classifier Chains;
GA-PartCC	:	Genetic Algorithm for Optimizing Partially-Chained Models;
HBCC	:	Hybrid-Binary Chain Multi-Label Classifier;
LSOP	:	Label Sequence Optimization Problem;
MLC	:	Multi-Label Classification;
PartCC	:	Partially-Chained Model;
OC	:	Order Crossover;
OOCC	:	One-To-One Classifier Chains;
SLC	:	Single-Label Classification;
TSP	:	Traveling Salesman Problem;

Contents

Chapter 1

Introduction

Classification is one of the most active topics of research in the fields of data mining and machine learning. It consists in the task of automatically assigning objects to discrete classes (known as class labels or simply labels) based on the features of the objects. In other words: classifying is to predict the category(ies) to which an object belongs. There are several important real-world applications of classification. Some of them are traditional and well-known, such as spam detection (identifying an incoming e-mail as either "spam" or "regular"), fraud detection (identifying whether a credit card transaction is "fraudulent" or "genuine"), and loan risk prediction (classifying loan applicants as "low", "medium", or "high" credit risks). Others have arisen relatively recently and are less popular, such as functional genomics (determining the functions of genes and proteins) and automatic music categorization (associating songs to music genres).

In the majority of classification problems, each object must be associated to one and only one label within a predetermined set of class labels. These are known as single-label classification (SLC) problems [23, 47, 92, 105, 109]. For example, in the loan risk prediction problem, a loan applicant can be classified as "low", "medium" or "high" credit risk, but he or she will never be classified as two or three of these labels all together.

The single-label classification problem can be formally defined as follows.

Definition 1.1 *(Single-Label Classification). Let $X = \{X_1, ..., X_d\}$ be a set of d predictive attributes and $L = \{l_1, ..., l_q\}$ be a set of q class labels, where $q \geq 2$. Consider a training dataset D composed of N instances of the form $\{(x_1, c_1), (x_2, c_2), ..., (x_N, c_N)\}$. In this dataset, each x_i corresponds to a vector $(x_1, ..., x_d)$ that stores values for the d predictive attributes in X and each $c_i \in L$ corresponds to a single class label. The goal of the single-label classification task is to learn from D a function (a.k.a. classifier) y that,*

given an unlabeled instance $t = (x, ?)$, is capable of effectively predicting its class label c, i.e., $y(t) \rightarrow c$. When $|L| = 2$ the problem is called a binary SLC problem. Otherwise, it is called a multiclass SLC problem.

However, not all classification problems are single-label. An example is automatic music categorization, where the goal is to associate songs to music genres. For instance, most songs written by the Brazilian band Novos Baianos can be classified as belonging to "Rock", "Samba", and "Bossa Nova" genres at the same time. In the same way, a number of compositions by Tom Jobim are a mixture of both music genres: "Jazz" and "Bossa Nova". Hence, music categorization represents a multi-label classification (MLC) problem, [18, 38, 96, 97, 113] in which objects can be assigned to various labels within a predetermined set of class labels. Over the last few years, several other important and modern applications of MLC classification have emerged [38, 100], such as text categorization (associating documents to various subjects), direct marketing (recommendation of products for clients), automated medical diagnosis (identifying when patients are suffering from one or more diseases at the same time), functional genomics (determining the multiple biological functions of genes and proteins), and image and video annotation (assigning keywords to images and videos), just to name a few.

The multi-label classification problem can be formally defined as follows.

Definition 1.2 *(Multi-Label Classification). Let $X = \{X_1, ..., X_d\}$ be a set of d predictive attributes and $L = \{l_1, ..., l_q\}$ be a set of q class labels, where $q \geq 2$. Consider a training dataset D composed of N instances of the form $\{(x_1, Y_1), (x_2, Y_2), ..., (x_N, Y_N)\}$. In this dataset, each x_i corresponds to a vector $(x_1, ..., x_d)$ that stores values for the d predictive attributes in X and each $Y_i \subseteq L$ corresponds to a subset of labels. The goal of the multi-label classification task is to learn from D a classifier h that, given an unlabeled instance $t = (x, ?)$, is capable of effectively predicting its set of labels (a.k.a. labelset) Y, i.e., $h(t) \rightarrow Y$.*

Looking from a database theory angle, we might consider there is a unique difference between the SLC and MLC problems: the first corresponds to predicting the state of a single-valued class attribute, whereas the latter, the states of a multi-valued class attribute. Although the difference is subtle in theory, in practice MLC problems tend to be much more challenging. This is due to three main reasons. First, MLC applications usually need to handle a huge number of possible label combinations. Considering a problem involving q distinct class labels, the size of the output space in MLC is 2^q whereas it is just q

in SLC. Second, real-world MLC datasets (e.g., multimedia data, biological data, etc.) are usually larger and more complex in structure than SLC datasets (which often correspond to ordinary relational data). Finally, the third and most important challenge concerns the existence of correlations between labels in MLC. For example, a song is unlikely to be simultaneously labeled as "Heavy Metal" and "Jazz" because these two music genres have a strong negative correlation. Analogously, the likelihood of a song being labeled as "Pop" becomes stronger if it has been labeled as "Hip Hop" and "R&B". Thus, intuitively, we would expect that algorithms that are able to capture and model label correlations should be more accurate. Actually, exploiting label dependencies[1] has been a major concern in MLC research. A large body of recent work [11, 35, 37, 45, 81, 82, 83, 87, 98, 108] has primarily concentrated efforts to tackle this problem.

Proposed in [82, 83], the classifier chains (CC) method has become one of the most popular of such techniques. This method is mainly distinguished by its simple yet effective approach to incorporate label correlations into the classification process. It works as follows. First, a randomly-ordered chain containing all the q labels involved in the classification problem is generated, such as, for instance $C = \{l_1 \rightarrow l_2 \rightarrow ... \rightarrow l_q\}$. The CC's training phase consists in learning q binary classifiers, one for each label, following the chain ordering. The first binary classifier, y_1, is trained using solely the attributes that compose the feature set X as its input attributes. This classifier will be responsible for the prediction of the first label in the chain (l_1, considering the example chain C). The second binary classifier, y_2, is trained using a different feature space: X augmented with the training information of the first label in the chain (l_1, considering C). This classifier will be responsible for the prediction of the second label in the chain (l_2, considering C). Each subsequent classifier y_j is trained using X augmented with the training information of $j-1$ labels as its input attributes. I.e., the feature space of each classifier y_j is extended with the true label information of all previous labels in the chain. Once the model is built, the classification step is also performed in a chained way. To predict the labelset of a new object, q binary classifications are needed, with the process beginning at y_1 and going along the chain. In this procedure, the classifier y_j predicts the relevance of label l_j, given the features of the new object augmented by the predictions carried out by the previous $j-1$ classifiers.

The CC model for MLC has many appealing properties. First of all, it is theoretically simple. While several MLC methods invest in complex probabilistic techniques to

[1] In this work, the terms "correlation" and "dependence" are used interchangeably, as is typically done in the MLC literature.

model label dependencies, CC adopts a quite straightforward strategy: it just passes label information between classifiers. It is also relatively efficient, since it scales linearly with q. Finally, and more importantly, the method has proved to be highly effective. A comprehensive recent empirical study [66] comparing several state-of-the-art methods for MLC reported that CC is among the top best performing methods in terms of predictive performance.

Putting together simplicity, efficiency, and effective performance, CC has become one of the most adopted frameworks for MLC. Not surprisingly, a considerable number of variations of the original CC method have recently been proposed in the literature [16, 48, 58, 63, 78, 79, 80, 91, 110]. A common characteristic of these variations is that they try to eliminate a key drawback in the original proposal: the fact that the label ordering is decided at random. It is intuitive that an inadequate label ordering can potentially decrease accuracy, as the first binary classifiers could frequently output wrong predictions at classification time, thus resulting in significant error propagation along the chain. A simplistic solution to this problem would be to arrange the chain by placing the labels that are easiest to predict as the very first elements. Nevertheless, as argued by [58], this idea might not necessarily produce a label sequence that leads to an improvement on the predictive accuracy of the CC model (i.e., an optimized label sequence), since a label that is difficult to predict may make subsequent labels considerably easier to model. It is thus important to invest in algorithmic solutions to find an optimized chain order. Nonetheless, this is a difficult problem because of the enormous search space of $q!$ different existing label permutations.

The current extensions to the basic CC method make use of three different approaches to overcome the label sequence optimization problem (LSOP). The first and most widely adopted – proposed by the authors of CC themselves in [82, 83] – consists of combining random orders via an ensemble of classifier chains (ECC) in order to mitigate the effect of poorly ordered chains. The second category of CC variations [48, 63, 80, 91, 110] are guided by tests to assess label correlation between pairs of labels. These methods work by first running a preprocessing step that aims at identifying strongly correlated labels. Further, this information is employed to determine a restricted set of candidate chain sequences (basically, chains in which correlated labels are placed close to each other). Then, one of these candidate sequences should be randomly chosen or, optionally, ensembles can be built by randomly selecting some of the candidates. Finally, the third category of CC variations [16, 58, 78, 79] rely on the use of heuristic search techniques (such as beam search) that aim at finding a single optimized label sequence.

1.1 Contributions

This thesis proposes the use of genetic algorithms (GAs) [28, 30, 39, 85] as a new approach for optimizing multi-label chain classifiers. GAs are a powerful search technique inspired by Darwin's theory of natural evolution. In short, the GA search works as follows [28, 39]. In the first step, an initial population of individuals is created, where each one corresponds to a candidate solution to a given problem (in this thesis, the target problem is to find an optimized label sequence for a chain of classifiers). Next, these individuals are evaluated by a fitness function which assigns a numerical quality value to each of them. Then, the genetic algorithm produces a new generation of individuals by employing the notion of "survival of the fittest". This procedure consists of selecting the best (fittest) individuals to be combined so as to produce a new generation resembling them (using genetic operators such as crossover and mutation). This process goes on for many iterations, progressively producing better and better candidate solutions. Normally, the GA execution terminates after a user-specified number of generations.

Our main motivations for presenting a solution to the LSOP based on the evolutionary paradigm of genetic algorithms are listed below:

- GAs are a global search method capable of effectively exploring the extremely large search space associated to the LSOP problem. Hence, GAs are expected to discover correlations among labels that would be missed by a greedy algorithm [30, 32].
- GAs have been successfully employed to solve a large number of classification problems [30], varying from fraud detection [27] to the automated recognition of painting artists [60].
- GAs have also been successfully applied to solve optimization problems where a candidate solution is represented as a permutation, like the classical traveling salesman problem (TSP) [59, 68, 72] and the vehicle routing problem (VRP) [54, 64, 73]. Although both TSP and VRP do not constitute classification problems, they bear some resemblance to the LSOP: in essence, LSOP, VRP and TSP are permutation problems.

In this thesis, we propose two novel genetic algorithms: GACC (Genetic Algorithm for Optimizing Classifier Chains) and GA-PartCC (Genetic Algorithm for Optimizing Partially-Chained Models). The GACC strategy [42] represents the first proposed strategy that makes use of an evolutionary algorithm to optimize multi-label chain classifiers. In

this strategy, each GA individual encodes a different label permutation. Crossover works by transferring sub-chains of random length between two individuals whilst mutation swaps pairs of labels of an individual. At the end of the evolutionary cycle, GACC delivers to the user a single optimized chain ordering.

The GA-PartCC strategy [43] constitutes an extension to the GACC method proposed with the goal of investigating a potential drawback of the original CC method: the fact that CC forces all labels to be present in the chain. None of the extensions have yet explored the idea of generating models defined by optimized partial chains. In this context, the aim is to build a CC model in which one or more labels may be absent from the chain because their presence would lead to a decrease in the predictive accuracy. This is because some of the binary classifiers may pass redundant and irrelevant information, or wrongly predicted labels, along the chain, which might confuse the subsequent classifiers in the chain. Therefore, it might be interesting to remove these irrelevant or redundant labels from the chain structure (using independent binary classifiers for predicting each of them) and to create a partial chain with an optimized sequence using only the remaining labels.

Experiments on diverse benchmark datasets show that both GACC and GA-PartCC obtain, overall, higher predictive accuracies than CC and competitive results against other well-established chaining methods. Furthermore, our proposed genetic algorithms offer two advantages. The first lies in that they deliver an interpretable result, i.e., at the end of the process both GACC and GA-PartCC return a single optimized chain, reflecting the label dependencies. This characteristic makes the strategies suitable for applications that require the use of interpretable classifiers [33]. These kinds of classifiers explain their classification decisions and are mainly represented by decision trees [74] and associative classifiers [62]. This is a important advantage, since in some application scenarios of MLC, such as medical diagnosis, bioinformatics and direct marketing, the ability to interpret the classification result might be almost as important as the accuracy itself. The second advantage of the GA strategies is the fact that they are efficient at classification time (since they produce a single model), differently from some of the CC variations, such as the popular one based on combining several random orders trough an ensemble.

In summary, the main contribution of this thesis is the proposal of two novel chain methods for multi-label classification based on genetic algorithms, which occupy an important niche: they are competitive on diverse multi-label problems, yet being suitable for use with interpretable classifiers. One of these strategies (GA-PartCC) is capable of searching for a single optimized label ordering, while at the same time taking into

consideration the utilization of partial chains.

Secondary contributions of the thesis aim at improving the fundamental understanding of the underlying principles of the classifier chains model. In order to accomplish this goal, we report and discuss the results of an experiment that, for the first time, investigated in depth the influence of the label sequence in the predictive accuracy of CC models. Additionally, we perform a set of empirical comparisons involving different chaining methods and propose a group of baseline heuristics for the determination of optimized label sequences.

Significant parts of the research presented in this thesis have appeared in the following publications.

- [42] Eduardo Corrêa Gonçalves, Alexandre Plastino and Alex A. Freitas. *A Genetic Algorithm for Optimizing the Label Ordering in Multi-Label Classifier Chains.* In Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2013). Washington, D.C., USA, November 2013 (currently classified as A2 by Qualis-CAPES.).
  - This paper presents preliminary results obtained with our proposed GACC algorithm, which is itself the main subject of Chapter 5.

- [43] Eduardo Corrêa Gonçalves, Alexandre Plastino and Alex A. Freitas. *Simpler is Better: a Novel Genetic Algorithm to Induce Compact Multi-label Chain Classifiers.* In Proceedings of the 2015 Genetic and Evolutionary Computation Conference (GECCO 2015). Nominated for best paper award. Madrid, Spain, July 2015. (currently classified as A1 by Qualis-CAPES.).
  - This paper comprises the core of Chapter 6.

- [16] Pablo Nascimento da Silva, Eduardo Corrêa Gonçalves, Alexandre Plastino and Alex A. Freitas. *Distinct Chains for Different Instances: An Effective Strategy for Multi-label Classifier Chains.* In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2014). Nancy, France, September 2014 (currently classified as A2 by Qualis-CAPES.).
  - Some of the results of the exhaustive experiment presented in Chapter 4 of this thesis were published in the above paper.

1.2 Thesis Organization

This thesis is structured as follows:

- Chapter 2: Background on Multi-Label Classification. This chapter presents an overview of multi-label classification, covering the following topics: basic approaches for inducing multi-label classifiers; properties of multi-label datasets; performance evaluation metrics for MLC; the notion of label dependence.

- Chapter 3: Genetic Algorithms for Permutation Problems. This chapter provides a succinct overview of GAs. We focus our discussion on GAs designed for permutation problems, as this thesis primarily deals with the development of methods for discovering optimized label sequences (i.e., optimized permutations of labels) for multi-label chain classifiers.

- Chapter 4: Multi-Label Chain Classifiers. This chapter is devoted to the classifier chains method. First, the original CC method is explained and formalized in pseudocode. Next, we present an experiment that investigated the influence of the label sequence in the predictive accuracy of CC models [16]. The results of this experiment demonstrate that the use of an optimized label sequence actually corresponds to a key factor in inducing effective multi-label chain classifiers. Motivated by this issue, we propose a set of baseline heuristics for the determination of optimized label sequences. Additionally, we present and compare the variations of the CC method currently proposed in the literature.

- Chapter 5: Genetic Algorithm for Optimizing Classifier Chains (GACC). This chapter addresses one of the main contributions of this thesis: the GACC method [42]. We describe in detail the designed GA and experimentally study the performance of our proposed method.

- Chapter 6: Genetic Algorithm for Optimizing Partially-Chained Models (GA-PartCC). This chapter addresses the second main contribution of this thesis: the GA-PartCC method [43]. Initially, we formally define the concept of partially chained (PartCC) model for MLC and describe the GA-PartCC method. Next, experimental results of GA-PartCC and diverse CC variations are presented.

- Chapter 7: Conclusions. In this chapter we outline and discuss the achievements of this thesis and identify directions for future research.

Chapter 2

Background on Multi-Label Classification

This chapter gives an overview of multi-label classification (MLC). It starts in Section 2.1 with an introduction to the basic approaches for inducing multi-label classifiers. Next, Section 2.2 provides a study of the properties of multi-label data. Section 2.3 is devoted to evaluation aspects of multi-label classifiers, introducing and comparing different kinds of performance measures proposed in the literature. Finally, Section 2.4 addresses the label dependence issue, which, as discussed in the previous chapter, represents an important aspect to be taken into account when designing effective multi-label methods.

2.1 Basic Approaches for Multi-label Learning

According to the literature [18, 96, 97, 113], existing methods for MLC can be primarily categorized into two fundamental families: problem transformation and algorithm adaptation. In what follows, each of the two families is introduced with emphasis given on the first, since the contributions proposed in this thesis aim at improving a problem transformation strategy. For a recent and comprehensive survey on MLC methods, the reader is referred to [37].

2.1.1 Problem Transformation Methods

Problem transformation (a.k.a. algorithm independent) methods work by transforming the original multi-label problem into one or more single-label problems. Then, any existing single-label algorithm can be directly applied by simply mapping back its single label predictions into multi-label predictions. Problem transformation methods are flexible, as they enable abstraction from the underlying base (single-label) classification algorithm.

This constitutes an important advantage, because different single-label techniques (such as decision trees, SVM, Naïve Bayes, etc.) are more or less effective according to the different application domains.

There are a few distinct strategies to perform the transformation of a multi-label problem into one or more single-label problems [18, 96, 97]. Nonetheless, three are most widely used: label combination (LC), ranking by pairwise comparison (RPC) and binary relevance (BR). In the remainder of this subsection, these three strategies are explained with the aid of the hypothetical training dataset for music categorization illustrated in Figure 2.1. Following the notation introduced in Chapter 1, in this example consider $L = \{$*"Metal", "Jazz", "Bossa", "Pop"*$\}$ as the set of non-disjoint class labels (music genres). Also consider that each instance i in the dataset (in this case, a song) is associated with a vector x_i – which stores values for an arbitrary number of predictive attributes – and a subset of labels $Y_i \subseteq L$.

L = {Metal, Jazz, Bossa, Blues}

Instance	Features	Metal	Jazz	Bossa	Pop
1	x_1	•			•
2	x_2		•	•	
3	x_3		•		
4	x_4	•			
5	x_5		•	•	•

Figure 2.1: Music categorization dataset

2.1.1.1 Label Combination (LC)

The LC approach [7] – also referenced in the literature as "label powerset" and "label creation" – reduces the multi-label problem to a unique multiclass single-label problem. This is accomplished through the definition of a new compound class attribute whose values correspond to all possible label combinations present in the original training dataset. Figure 2.2 illustrates the resulting (single-label) dataset generated from the LC transformation of the original (multi-label) dataset from Figure 2.1. Observe that the two labels of the first instance in the original dataset ("Metal" and "Pop") were combined to create a new single-label "Metal-Pop" in the resulting dataset. Analogously, the new single-labels "Jazz-Bossa" and "Jazz-Bossa-Pop" were created from the two and three labels respectively associated to the instances 2 and 5 of the original dataset. It is worth noting that the transformations only affect the label space, i.e., the feature space is preserved in

its original form (in fact, this is a characteristic common to all problem transformation methods).

Inst.	Feat.	Class
1	x_1	Metal-Pop
2	x_2	Jazz-Bossa
3	x_3	Jazz
4	x_4	Metal
5	x_5	Jazz-Bossa-Pop

Figure 2.2: LC transformation of the example music categorization dataset

Once the transformation has been applied to the original dataset, the induction of an LC model is straightforward, corresponding merely to training a multiclass single-label classifier using the transformed dataset. The classification of a new instance is also trivial: the LC model simply outputs a compound class, which actually corresponds to a labelset in the original dataset.

The LC method is simple and offers the advantage of implicitly taking into account the dependencies between labels. However, it suffers from two important drawbacks. First, it is not capable of predicting labelsets that are not present in the training set. For instance, an LC model induced from the transformed dataset shown in Figure 2.2 would not be able to classify a new song as "Bossa" or "Jazz-Pop" because these combinations do not exist in the original multi-label dataset. Second, and more importantly, the LC transformation can generate an exponential number of compound classes, some of them with very few instances compared to the rest. The maximum number of single-label classes is given by $min(N, 2^q)$, where N corresponds to the number of training instances and q represents the number of labels. Although in practice the actual number is usually much smaller than the maximum possible number, it is normally much larger than q (as will be shown in Section 2.2). Thus, the LC method is impractical for several real-world problems.

2.1.1.2 Ranking by Pairwise Comparison (RPC)

The RPC approach [35, 49] works by transforming the original multi-label dataset into diverse binary single-label datasets. Each derived dataset is associated to a distinct pair of labels $\{l_i, l_j\}, 1 \leq i < j \leq q$ and must contain those instances in the training set D which are labeled either as l_i or l_j, but not labeled as both. Figure 2.3 shows the six binary datasets resulting from the RPC transformation of the music categorization dataset illustrated in Figure 2.1. Observe that the first dataset refers to the pair of labels "Metal" and "Jazz". All instances in this dataset can be associated to either of the following two class values:

"Metal(1)-Jazz(0)" or "Metal(0)-Jazz(1)". The first and fourth instances have the class value "Metal(1)-Jazz(0)" because in the original dataset these respective instances are labeled as "Metal" but are not labeled as "Jazz". Analogously, the second, third and fifth instances have the class value "Metal(0)-Jazz(1)", because these are labeled as "Jazz" and not labeled as "Metal" in the original dataset.

Inst.	Feat.	Class Metal-vs-Jazz
1	x_1	Metal(1)-Jazz(0)
2	x_2	Metal(0)-Jazz(1)
3	x_3	Metal(0)-Jazz(1)
4	x_4	Metal(1)-Jazz(0)
5	x_5	Metal(0)-Jazz(1)

Inst.	Feat.	Class Metal-vs-Bossa
1	x_1	Metal(1)-Bossa(0)
2	x_2	Metal(0)-Bossa(1)
4	x_4	Metal(1)-Bossa(0)
5	x_5	Metal(0)-Bossa(1)

Inst.	Feat.	Class Metal-vs-Pop
4	x_4	Metal(1)-Pop(0)
5	x_5	Metal(0)-Pop(1)

Inst.	Feat.	Class Jazz-vs-Bossa
3	x_3	Jazz(1)-Bossa(0)

Inst.	Feat.	Class Jazz-vs-Pop
1	x_1	Jazz(0)-Pop(1)
2	x_2	Jazz(1)-Pop(0)
3	x_3	Jazz(1)-Pop(0)

Inst.	Feat.	Class Bossa-vs-Pop
1	x_1	Bossa(0)-Pop(1)
2	x_2	Bossa(1)-Pop(0)

Figure 2.3: RPC transformation of the example music categorization dataset

The induction of an RPC model consists in training one binary single-label classifier for each derived dataset. Thus, each classifier is trained for a pair of labels $\{l_i, l_j\}$,

forming a decision boundary for these two labels. Unlike in the LC method, in RPC the classification step cannot be directly performed. In RPC, a new instance t to be classified must first be submitted to all binary models. Then, the "votes" received by each label are counted and used to compute a label ranking. The final predicted labelset for t is obtained with the application of a threshold function for separating the highest-rated labels from the lowest-rated ones.

The RPC method is able to predict label combinations that are not present in the original training set, while still retaining the LC's advantage of implicitly incorporating label correlations into the classification model. However, it has important disadvantages. First, its classification process is dependent on a threshold function, which might need to be calibrated according to the different datasets. Second, it may achieve quadratic complexity in terms of space and time, since, in the worst case, a total of $q(q-1)/2$ binary classifiers must be trained and kept in memory. All of them need to be queried at classification time. Due to this, the RPC method is usually intractable for several real-world problems in which q is not small (further discussion is given in Section 2.2).

2.1.1.3 Binary Relevance (BR)

BR [52, 55] is the most well-known and widely adopted problem transformation method for MLC [113]. In this approach, the original multi-label dataset is decomposed into q binary single-label datasets, one for each label. As an example, Figure 2.4 shows the four binary datasets generated from the BR transformation of the music categorization dataset from Figure 2.1.

Inst.	Feat.	Metal
1	x_1	•
2	x_2	
3	x_3	
4	x_4	•
5	x_5	

Inst.	Feat.	Bossa
1	x_1	
2	x_2	•
3	x_3	
4	x_4	
5	x_5	•

Inst.	Feat.	Jazz
1	x_1	
2	x_2	•
3	x_3	•
4	x_4	
5	x_5	•

Inst.	Feat.	Pop
1	x_1	•
2	x_2	
3	x_3	
4	x_4	
5	x_5	•

Figure 2.4: BR transformation of the example music categorization dataset

The induction of a BR model consists in training one binary classifier for each derived dataset. Consequently, in the music categorization example, four independent binary models would have to be trained, being each one solely and exclusively responsible for classifying a specific music genre. Once the BR model has been induced, the classification process is quite straightforward: new instances are predicted by simply combining the

outputs produced by each binary classifier. This process is illustrated in the example given in Figure 2.5. It shows the classification process of a new multi-label instance t (the song "The Girl from Ipanema") considering a hypothetical BR model trained using the group of derived datasets from Figure 2.4. In this figure, y_1, y_2, y_3 and y_4 respectively represent the trained binary classifiers to predict the genres "Metal", "Jazz", "Bossa" and "Pop" and x represent the set of features describing t. Observe that in order to carry out the classification, the BR model outputs the aggregation of the labels positively predicted by all of the independent binary classifiers. Since in this example labels "Metal" and "Bossa" were predicted as non-relevant whilst "Jazz" and "Pop" were predicted as relevant, the labelset {"Jazz", "Pop"} was assigned to t.

t = "The Girl from Ipanema" – Tom Jobim & Frank Sinatra

Classifiers	Classifications
y_1: x → {Metal, ~Metal}	~Metal
y_2: x → {Jazz, ~Jazz}	Jazz
y_3: x → {Bossa, ~Bossa}	~Bossa
y_4: x → {Pop, ~Pop}	Pop
Predicted Labelset	***{Jazz, Pop}*** $\subseteq L$

Figure 2.5: BR classification of the song "The Girl from Ipanema"

The BR strategy offers important advantages. First, like LC, it is simple and intuitive. Second, like RPC, it is capable of predicting labelsets that are not present in the training set. Third, unlike LC and RPC, the BR method has relatively low computational complexity, since it scales linearly with q. Nonetheless, an obvious and important drawback of the strategy lies in the fact that a trained BR model completely ignores the possible correlations among labels, as the binary classifiers take decisions independently from each other. Thus, in theory, the method tends to be more suitable for problems where only a small number of labels exhibit correlation with each other. Chapter 4 is devoted to the classifier chains model [82, 83], a direct extension of the binary relevance method which is capable of taking label dependencies into consideration.

2.1.2 Algorithm Adaptation Methods

Algorithm adaptation methods extend or adapt an existing single-label algorithm for the task of multi-label classification. E.g., in the recent work of [87], the authors introduce a slightly modified Bayes formula which enables the traditional Naïve Bayes technique to be applied in the MLC context. This formula is presented in Equation 2.1. Given an

unlabeled instance t, the probability of a label l_j being relevant is computed taking into consideration two separate contributions: the posterior probability of l_j conditioned on x (the set of features describing t) and also the posterior probability of l_j conditioned on $Z - l_j$, which denotes a set containing the estimated 0/1 relevance for each class label involved in the MLC problem, excluding l_j.

$$\Pr(l_j \mid x, Z - l_j) = \frac{\Pr(l_j \mid x) \times \Pr(l_j \mid Z - l_j)}{\Pr(l_j)} \tag{2.1}$$

Besides Naïve Bayes, other classic single-label techniques adapted for MLC include k-NN [90, 112, 108], neural networks [14, 111], decision trees [12], SVM [29, 107], and Bayesian networks [6, 102]. Unlike problem transformation strategies, it is noticeable that algorithm adaptation methods are normally designed to be used in specific problem domains. For instance, the adaptation of the Naïve Bayes technique presented in [87] was originally proposed to be applied to the task of text categorization, since it constitutes an area where Naïve Bayes very often yields much better results than other algorithms.

2.2 Multi-label Data

The performance of the different multi-label learning methods over a dataset may be affected by a number of distinct properties of the dataset. This section provides a study of such properties, which define the complexity and degree of "multi-labelled-ness"[1] of multi-label data. In order to facilitate the discussion, we present examples obtained from the benchmark datasets utilized in the experiments carried out in this thesis. The text is structured as follows. First, in Subsection 2.2.1, each dataset is briefly described in terms of its target classification task and basic characteristics (number of labels, attributes and instances). Following this introductory presentation, in Subsection 2.2.2, we show a set of statistics (the properties themselves) extracted from the collection of datasets, discussing how this information can be employed to give indication of the performance of the different multi-label methods.

2.2.1 Benchmark Datasets

The collection of datasets used in this thesis is presented below. All of them contain real-world data from distinct application domains. We present the name of each dataset,

[1] This term was coined by Jesse Read in [77], having become popular among the MLC community.

followed by its application domain(s) and a brief description of its characteristics. The datasets and further information about them can be obtained at the sources referenced in the text.

- Emotions [95] (Music Categorization). Classification of songs into six kinds of emotions: "sad-lonely", "angry-aggressive", "amazed-surprised", "relaxing-calm", "quiet-still", and "happy-pleased". The dataset comprises pieces of 593 distinct songs described by 72 extracted numeric features.

- Scene [7] (Image Annotation). Classification of images into six different contexts: "beach", "sunset", "field", "fall-foliage", "mountain", and "urban". The dataset is formed by 2407 images described by 294 numeric features.

- Flags [61] (Image Annotation). This is a simple dataset containing information about 194 countries and their national flags. Each instance is described by 19 attributes (such as area in km^2, language and predominant religion). The MLC task consists in predicting the colors present in the national flags ("red", "green", "blue", "yellow", "white", "black", and "orange").

- University [61] (Social Research). This dataset contains information related to 242 North American universities, which are characterized by 14 input attributes (e.g.: number of students, male/female ratio, whether it is under either public or private control, etc.). The MLC task is to predict the courses offered by a university. The ten most popular courses were selected to form this dataset.

- Yeast [29] (Functional Genomics). This is a biological dataset where yeast genes can be associated with a set of 14 functional categories belonging to the top level of the Functional Catalog (FunCat) [88]. The dataset is composed of 2417 instances described by 103 numeric attributes, representing micro-array expressions and phylogenetic profiles of the yeast genes.

- CES-16 [41] (Social Research). This dataset contains 904 observations collected from a household survey called Consumer Expenditure Survey (CES). This survey has been conducted by a Brazilian institute of research since 1947 to, among other goals, support the analysis of food consumption of Brazilian families. Each instance concerns a different family, which is characterized by 3 attributes: monthly income, number of members and city of residence. The MLC task is to predict the collection of products acquired by each family on their last visit to a supermarket. Sixteen

distinct products (presented in the examples of [40, 41]) were selected to compose the dataset.

- Birds [8] (Audio Classification and Biology Research). Classification of the bird species present in a 10-second audio record. The dataset comprises 645 instances, 260 features and 19 labels (species of birds).

- Thyroid [61, 74, 76] (Medical Diagnosis). This dataset keeps information about examinations performed on 9172 patients from 1984 to 1987. Each instance is described by 29 attributes and the classification task is to perform the diagnosis of thyroid conditions according to a set of 25 distinct class labels.

- Genbase [24] (Functional Genomics). Classification of proteins into 27 distinct functions. The dataset comprises 662 instances, each one corresponding to a protein chain represented by a motif sequence vocabulary (binary array of length 1186).

- Medical [70] (Text Categorization and Medical Diagnosis). Classification of clinical reports into 45 distinct codes, representing different diseases or clinical conditions. This dataset keeps information about clinical reports regarding 978 patients. Each report is described by 1149 binary attributes which indicate if a specific term (word) is either present or absent.

- Enron [56, 81] (Text Categorization). Classification of e-mails into 54 categories. The dataset contains 1702 e-mail messages exchanged between employees of the Enron Corporation, that were made available during a legal investigation. Each e-mail is described by 1001 binary attributes which indicate if a term is either present or absent.

- LLog [77] (Text Categorization). Classification of free text into 75 topics. The dataset was generated from the Language Log forum[2], and contains 1460 instances, each described by 1004 binary attributes. As in the case of the datasets Medical and Enron, each binary attribute is used to indicate if a specific term appears in a text document.

- Cal500 [101] (Music Categorization). Classification of songs into 174 different kinds of music concepts, which, among others, may represent music genres (e.g.: “Soul”, “Jazz”, “Pop”, etc.) and the presence of certain instruments (e.g.: “Synthesizer”, “Saxophone”, “Piano”, etc.). The dataset comprises 502 well-known songs recorded

[2]http://languagelog.ldc.upenn.edu/nll/

by distinct artists in the last 50 years. Each song is described by 68 extracted numeric features.

2.2.2 Dataset Statistics

In what follows, we present a collection of statistics extracted from the 13 benchmark datasets introduced in the previous subsection. These reveal a series of interesting and useful properties associated with multi-label data.

Table 2.1 presents a collection of properties that can be used to roughly indicate the overall complexity associated to each dataset. The first column informs the name of each dataset whilst the second, third, and fourth columns (N, d, and q) respectively show the number of instances, features and labels (which have been previously presented in Subsection 2.2.1). Note that the datasets are arranged in ascending order of number of labels. The fifth column indicates the dataset complexity, as proposed in [77], which is given by the product $N \times d \times q$. Nonetheless, it is worth reminding the reader that methods based on the transformation of the multi-label problem in diverse binary single-label problems, such as RPC and BR, may be more affected by the size of q (parameter that actually defines the number of binary models to be induced) than by the complexity itself.

Table 2.1: List of benchmark multi-label datasets and their statistics concerning complexity

Dataset	N	d	q	***Complexity***
emotions	593	72	6	256,176
scene	2,407	294	6	4,245,948
flags	194	19	7	25,802
university	242	14	10	33,880
yeast	2,417	103	14	3,485,314
ces-16	904	3	16	43,392
birds	645	260	19	3,186,300
thyroid	9,172	29	25	6,649,700
genbase	662	1186	27	21,198,564
medical	978	1449	45	6,377,0490
enron	1,702	1001	53	90,296,206
llog	1,460	1004	75	109,938,000
cal500	502	68	174	5,939,664

Table 2.2 presents a set of properties which give a reasonable indication of the degree of "multi-labelled-ness" of the datasets. The first column informs the name of each dataset

whilst the second and third reproduce, respectively, the number of instances and labels. The values in the fourth column ($LCard$) present the label cardinality associated to each dataset, which corresponds to the average number of labels per instance. The lowest value is close to 1.0 in the "birds" dataset, where most instances are associated with only one label and some of them might not be associated to any label. On the other hand, the highest value is superior to 26.0 in the "cal500" dataset. Complementary, the fifth column ($LDens$) gives the label density, which corresponds to $LCard$ divided by q. Both measures were introduced in [97]. The sixth column (NC) gives the number of distinct label combinations present in each dataset. The seventh column (NU) presents the number of "unique labelsets", i.e., the number of label combinations that have frequency equal to 1 in the dataset. These two measures were proposed in [77]. We further introduce two new measures, which are presented in the last two columns. The first is NU/NC, in the eighth column. This measure gives the proportion of unique labelsets to the total number of distinct labelsets. The second is the *number of RPC pairs* (NP), presented in the last column. It indicates the total number of distinct pairs of labels $\{l_i, l_j\}, 1 \leq i < j \leq q$ that are associated to at least one instance in the dataset, considering that the instance must be labeled as l_i but not as l_j or vice-versa. In other words: it gives the number of binary classifiers that would need to be trained in order to induce an MLC model employing the RPC method.

Table 2.2: List of benchmark multi-label datasets and their statistics concerning the degree of "multi-labelled-ness"

Dataset	N	q	$LCard$	$LDens$	NC	NU	NU/NC	NP
emotions	593	6	1.87	0.31	27	4	0.15	15
scene	2,407	6	1.07	0.18	15	3	0.20	15
flags	194	7	3.39	0.49	54	24	0.44	21
university	242	10	1.58	0.16	68	36	0.53	45
yeast	2,417	14	4.24	0.30	198	77	0.43	91
ces-16	904	16	3.17	0.20	475	358	0.75	120
birds	645	19	1.01	0.05	133	73	0.55	171
thyroid	9,172	25	7.00	0.28	31	5	0.16	247
genbase	662	27	1.25	0.05	32	10	0.31	350
medical	978	45	1.25	0.03	94	33	0.35	984
enron	1,702	53	3.38	0.06	753	573	0.76	1,378
llog	1,460	75	1.18	0.02	304	189	0.62	2,774
cal500	502	174	26.04	0.15	502	502	1.00	15,028

As stated in the beginning of this section, the properties of a dataset will influence the performance of the different multi-label methods. For instance, consider the LC method which treats each label combination that exists in the multi-label dataset as a distinct

single-label class. The information presented in Table 2.2 demonstrates that this method is actually impractical for most MLC problems. This is due to two reasons. First, the value of the NC property (number of distinct label combinations) is superior to 30 in the majority of the datasets, which can be considered a large number of class values for a multiclass single-label problem. Second, observe that several datasets have a large number of label combinations that are associated to only one instance (given by the NU property). This is, for example, the case of "university", "ces-16", "birds", "enron", "llog" and "cal500", where the proportion NU/NC is superior to 50% (i.e., among the total number of distinct label combinations, more than 50% are associated to only one instance). Hence, the LC method would have to deal with a large number of compound classes that appear only once in the dataset. In the extreme case of the "cal500" dataset, the number of label combinations is equal to the number of instances ($NC = N$), so the method would have to deal with only one instance per each compound class.

The information in Table 2.2 also allows for a comparison between the RPC method (where a binary classifier must be trained for each pair of labels of interest) and the BR method (where a binary classifier is trained for each label) in terms of time and space complexity. In this regard, the NP property (last column Table 2.2) reveals that the number of binary models that need to be trained and kept in memory considering the RPC method is equal or very close to the upper bound of $q(q-1)/2$ for all datasets. On the other hand, the BR method has a much lower complexity, since it only requires the use of a fixed number of q classifiers. This is the reason why the BR method has been preferred over RPC and LC in many real-world problems.

2.3 Performance Evaluation

Over the last few years, several evaluation measures specifically designed for MLC have been proposed in the literature. The platforms MULAN [99] and MEKA [84] – both widely adopted for research projects in MLC – make available more than twenty different metrics to their users. This subsection introduces and compares some of the most commonly used evaluation measures. In the examples and definitions throughout the text the following notation was adopted:

- n : number of test instances,
- q : number of labels,

- Y_i: actual labelset of the ith test instance,
- Z_i: predicted labelset of the ith test instance.

The most trivial evaluation metric for MLC algorithms is the Exact Match (EM), defined in Equation 2.2. This measure assesses the proportion of instances that were fully correctly predicted in the test set. Consider that $I(true) = 1$ and $I(false) = 0$.

$$EM = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i) \tag{2.2}$$

Although it provides essential information, the EM measure is regarded as too strict to be applied in a standalone manner. This is because in MLC problems a result can be, very often, partially correct, i.e., the classifier may predict some of the correct labels, but it can either miss some of them or include wrong predictions. Hence, it is necessary to adopt other evaluation metrics so as to complement the Exact Match metric.

The Accuracy (ACC) and F-Measure (FM) metrics, respectively defined in Equations 2.3 and 2.4, can be seen as "less harsh" versions of EM. Both metrics provide the user with information about the proportion of correct predictions, thus taking into consideration results that are partially correct.

$$ACC = \frac{1}{n}\sum_{i=1}^{n} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{2.3}$$

$$FM = \frac{1}{n}\sum_{i=1}^{n} \frac{2 \times |Z_i \cap Y_i|}{|Z_i| + |Y_i|} \tag{2.4}$$

Tables 2.3 and 2.4 illustrate, respectively, the use of Accuracy and F-Measure in a toy problem of six test instances and four labels. Observe that the measures are simple and intuitive: the higher the value, the better the classification. However, the cases illustrated in the examples E_5 and E_6 in both tables highlight a drawback associated with these measures: they are not very severe on penalizing wrong predictions (either false positives or false negatives).

In order to cope with this problem, it is possible to employ the Hamming Loss measure (HL), defined in Equation 2.5. This metric informs the average number of incorrect binary predictions per instance. The expression $|Y_i \ \Delta \ Z_i|$ represents the symmetric difference between Y_i and Z_i. Thus, the smaller the HL value, the better the performance.

$$HL = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \,\Delta\, Z_i|}{q} \tag{2.5}$$

The use of HL in a new toy problem of six test instances and four labels is illustrated in Table 2.5. Note that the most important advantage of HL over ACC and FM is that it is more suitable to penalize wrong predictions. Nevertheless, in many practical situations this characteristic ends up becoming a disadvantage. For instance, observe the four last examples in Table 2.5. They show that, considering a pair of cases with the same number of wrong predictions (such as E_3 and E_4 or E_5 and E_6), the HL measure does not distinguish the case with a greater number of correct predictions from the one with a smaller number of correct predictions.

Table 2.3: Illustration of Accuracy for six examples

	$\boldsymbol{Y_i}$	$\boldsymbol{Z_i}$	$\boldsymbol{ACC}$	**Comments**
E_1	l_1, l_2, l_3, l_4	l_1, l_2, l_3, l_4	1.00	perfect classification
E_2	l_1	l_2, l_3, l_4	0.00	worst case
E_3	l_1	l_2	0.00	worst case
E_4	l_1, l_3	l_1, l_2	0.33	
E_5	l_1	l_1, l_2	0.50	
E_6	l_1, l_2	l_1, l_2, l_3, l_4	0.50	more false positives than E_5, but the value is still 0.50

Table 2.4: Illustration of F-Measure for six examples

	$\boldsymbol{Y_i}$	$\boldsymbol{Z_i}$	$\boldsymbol{FM}$	**Comments**
E_1	l_1, l_2, l_3, l_4	l_1, l_2, l_3, l_4	1.00	perfect classification
E_2	l_1	l_2, l_3, l_4	0.00	worst case
E_3	l_1	l_2	0.00	worst case
E_4	l_1, l_3	l_1, l_2	0.50	
E_5	l_1	l_1, l_2	0.66	
E_6	l_1, l_2	l_1, l_2, l_3, l_4	0.66	more false positives than E_5, but the value is still 0.66

It is important to emphasize that despite some idiosyncrasies, all the four evaluation measures presented in this subsection are important since they provide complementary information about MLC processes. Another important aspect related to the Exact Match, Accuracy, F-Measure and Hamming Loss metrics is that they all work by first evaluating the performance of the algorithm on each test instance separately, and then averaging the obtained result over the entire test set. Due to this, they are referred to as *instance-based* or *example-based* measures. We may still observe that these four metrics summarize

Table 2.5: Illustration of Hamming Loss for six examples

	$\boldsymbol{Y_i}$	$\boldsymbol{Z_i}$	$\boldsymbol{HL}$	**Comments**
E_1	l_1, l_2, l_3, l_4	l_1, l_2, l_3, l_4	0.00	perfect classification
E_2	l_1	l_2, l_3, l_4	1.00	worst case
E_3	l_1	l_2	0.50	
E_4	l_1, l_3	l_1, l_2	0.50	better classification than E_3, but the value is still 0.50
E_5	l_1	l_1, l_2	0.25	
E_6	l_1, l_2, l_3, l_4	l_1, l_2, l_3	0.25	higher number of correct predictions than E_5, but the value is still 0.25

the overall performance of the algorithm taking into account all labels in the dataset. Nevertheless, in most multi-label scenarios the frequency distribution of the labels in the dataset is imbalanced. This naturally makes some labels easier to predict than others. For example, in the music categorization task, genres like "Pop" and "Dance" would certainly be much more frequent than "Folk". To cope with this situation, it might be also interesting to employ evaluation metrics capable of assessing the effectiveness of the classifier on each label separately. These kinds of metrics are called *label-based*. Two examples of label-based metrics are the True Positive Rate (TPR) and the True Negative Rate (TNR), respectively defined in Equations 2.6 and 2.7. In the definitions, l_j represents some specific label of interest and TP_{l_j}, TN_{l_j}, FP_{l_j}, FN_{l_j} the number of true positives, true negatives, false positives, and false negatives after the binary evaluation of l_j.

$$TPR_{(l_j)} = \frac{TP_{l_j}}{TP_{l_j} + FN_{l_j}} \tag{2.6}$$

$$TNR_{(l_j)} = \frac{TN_{l_j}}{TN_{l_j} + FP_{l_j}} \tag{2.7}$$

The TPR of label l_j measures the percentage of actual positive instances that were classified as positive whilst the TNR of l_j assesses the percentage of actual negative instances that were indeed classified as negative. To demonstrate the importance of label-based measures, consider the information presented in Table 2.6. In this table, the first column stores the actual labelsets of six test instances and the second contains the respective labelsets predicted by a hypothetical multi-label learner. In this example, we have $TPR_{(l_1)} = 3/(3+0) = 1.00$ and $TNR_{(l_1)} = 1/(1+2) = 0.33$. Therefore, the learning algorithm obtained the maximum value for the TPR of l_1, but its performance was far worse with respect to the TNR. Let us now examine the label l_2. The values of both

measures are given by $TPR_{(l_2)} = 2/(2+1) = 0.66$ and $TNR_{(l_2)} = 2/(2+1) = 0.66$. Thus, the TPR and TNR values of l_2 are more balanced.

Table 2.6: Illustration of actual and predicted labelsets of six examples

	$\boldsymbol{Y_i}$	$\boldsymbol{Z_i}$
E_1	l_1, l_3, l_4	l_1, l_4
E_2	l_2, l_4	l_1, l_4
E_3	l_1, l_4	l_1, l_2
E_4	l_2	l_2
E_5	l_3, l_4	l_1, l_3, l_4
E_6	l_1, l_2	l_1, l_2

Besides TPR and TNR, there are many other label-based measures for MLC. In fact, any binary measure for SLC classification, as the various presented in [51], can be directly applied in the MLC context. In spite of being label focused, these metrics can also be easily summarized for all labels by applying different kinds of averaging operations, such as micro-averaging and macro-averaging [97, 113].

As a final remark, it is worth mentioning that some studies also make use of *ranking-based* measures [97] to evaluate multi-label methods. Nonetheless, unlike example-based and label-based measures, this kind of metric requires the use of a method that can output a score for each class label (e.g., probability) along with the predicted 0/1 relevance. An example of such measure is One Error (OE), defined in Equation 2.8. It assesses the proportion of test instances where the label predicted with the highest confidence score (denoted by $best(Z_i)$ in the formula) does not correspond to an actual label. Consider that $H(true) = 1$ and $H(false) = 0$. Smaller values indicate better performance.

$$OE = \frac{1}{n}\sum_{i=1}^{n} H(best(Z_i) \notin Y_i) \tag{2.8}$$

It is important pointing out that OE and other ranking-based measures (such as the ones presented in [97]) can be considered more suitable for use in the evaluation of multi-label ranking methods. Multi-label ranking is a related classification task where the goal is to construct a classification model that provides, for each unseen instance, a list of preferences (i.e., a ranking) on the labels [65, 100]. In other words: given a test instance t, a multi-label ranking classifier produces a ranking, such as $r(l_4) < r(l_1) < r(l_3) < r(l_2)$, where $r(l_i)$ denotes the position of label l_i in the ranking (the smaller the better). As an example of practical application, consider a system for classification of scientific papers

into keywords. Suppose the number of keywords is fixed to k (i.e., each paper must be associated with exactly k keywords). In this case, to obtain the keywords for a new instance, a multi-label ranking method could simply select the top k labels in the output ranking.

2.4 The Label Dependence Issue

There seems to be a consensus in the literature that multi-label algorithms capable of identifying and modeling the dependencies among labels tend to be more effective [11, 35, 37, 45, 81, 82, 83, 87, 98]. In practice, two different types of label dependencies can be taken into consideration [21]: unconditional and conditional.

The unconditional dependence between a pair of labels l_i and l_j occurs when the actual joint probability $\Pr(l_i, l_j)$ is different from the expected joint probability: $\Pr(l_i, l_j) \neq \Pr(l_i) \times \Pr(l_j)$. This type of dependence is quite easy to be measured, since it is based only on the frequency of the labels. Any standard statistical measure of correlation can be used for this purpose, such as the Pearson correlation coefficient [92], the chi-squared test for correlation [9], and the mutual information [34, 80].

However, the strength of the dependence between two labels may dramatically change in presence of one or more variables from the input feature set X. In order to support this claim, consider the following example, originally presented in the study of [41] and obtained from the CES dataset. As described in Section 2.2, the CES dataset keeps information about purchases made by families residing in different Brazilian cities. In this dataset, the pair of labels *beer* and *salami* is unconditionally independent if we consider all instances. Thus, the following relation holds: $\Pr(beer, salami) \approx \Pr(beer) \times \Pr(salami)$. However, an example obtained with the use of a technique to extract exception rules proposed in [41] revealed that the correlation between these two items becomes not only positive but also considerably strong if we only consider the subset of the CES dataset defined by people who live alone. More clearly, in the subset of the CES dataset defined by instances where the value of the input attribute "number of members in the family" is equal to 1, the following relation holds: $\Pr(beer, salami) > \Pr(beer) \times \Pr(salami)$.

Despite being very simple, the above example is enough to disclose the importance of the exploitation of conditional dependencies in MLC processes. This type of dependence takes into consideration the attributes from the input feature space X in order to capture the dependence among labels. Two labels l_i and l_j are said to be condition-

ally dependent given x (a vector that stores values for the predictive attributes in X) if: $\Pr(l_i, l_j|x) \neq \Pr(l_i|x) \times \Pr(l_j|x)$. Clearly, the measurement of conditional dependence is more appropriate than the measurement of unconditional dependence, since any multi-label classifier will predict the labelset of an object conditioned on its features. Nonetheless and unfortunately, measuring conditional dependence is much more difficult due to the fact that, in most datasets, there are many input variables to be taken into consideration in the feature set X.

Chapter 3

Genetic Algorithms for Permutation Problems

The goal of this chapter is to review the concepts of genetic algorithms (GAs) particularly relevant to this thesis. The text is divided into two sections. Section 3.1 starts by providing a succinct overview of GAs. Section 3.2 expands upon the theory relating to GAs by specifically discussing the issues surrounding the design of GAs for solving permutation problems.

3.1 GA Basics

Evolutionary Algorithms (EAs) are stochastic search methods that work by simulating the principles of natural selection and natural genetics [2, 19, 28, 32]. The field of EAs encompasses different techniques, such as, among others, genetic algorithms [39, 85], genetic programming [57], and evolution strategies [5]. This thesis focuses on genetic algorithms (GAs), which are distinguished by their wide range of applications in the field of data mining [4, 30, 32, 46, 47].

GAs work based on the application of Darwinian principles (natural selection, reproduction and mutation) to solve high-dimensional problems. The following is a brief explanation of how the GA search works [28, 39]. In the first step, an initial population of individuals (also named chromosomes) is created, where each one corresponds to a candidate solution to a given problem. Next, these individuals are evaluated by a fitness function which assigns a numerical quality value to each of them. Then, the genetic algorithm produces a new generation of individuals by employing the notion of "survival of the fittest". This procedure consists in selecting individuals to be combined, with probability

proportional to their fitness values, so as to produce a new generation resembling them (using genetic operators such as crossover and mutation). The process goes on for many iterations, progressively producing better and better candidate solutions. Normally, the GA execution terminates either when a sufficiently fit individual emerges or after a user-specified maximum number of generations has been performed. Algorithm 1 (adapted from [28, 32]) outlines the above described scheme in pseudocode.

Algorithm 1 Generic pseudocode for a genetic algorithm

```
1: create an INITIAL POPULATION of individuals (candidate solutions)
2: EVALUATE each individual
3: repeat
4:    SELECT parents based on fitness
5:    apply GENETIC OPERATORS to selected individuals, creating new individuals
6:    EVALUATE each new individual
7:    UPDATE the current population (new individuals replace old individuals)
8: until (termination condition is satisfied)
```

GAs constitute a very generic heuristic search paradigm, which has proven to deliver good (though not necessarily optimal) solutions within acceptable time for a wide range of problems [28, 32, 39, 85]. In the next section, we examine the constituent parts of the GA search in detail. We focus our discussion on GAs designed for permutation problems, as this thesis primarily deals with the development of methods for discovering optimized label sequences (i.e., optimized permutations of labels) for multi-label chain classifiers.

3.2 GAs for Permutation Problems

Within the past 30 years, GAs have become quite popular as a means of solving hard combinatorial optimization problems [28, 85, 89]; many of them correspond to problems that take the form of deciding on the order in which a sequence of events should occur (permutation problems). Examples vary from the classical traveling salesman problem [59, 72] to the analysis of genome rearrangements [36], also including the personnel assignment problem [94], the vehicle routing problem [64, 73], and several others.

For instance, consider the traveling salesman problem (TSP), a well-known NP-hard problem where the objective is to find the shortest route for a traveling salesman who, starting from his home city, needs to visit a set of n cities and then return to his departure city. This problem can be formally stated as follows. Let G be a complete weighted

undirected graph composed of $n + 1$ nodes. In this graph, each node corresponds to a city (where node 0 stands for the traveling person's home city) and the weighted edges represent the distances between each pair of cities. The goal of TSP is to design a tour that starts and ends at node 0, includes all other nodes exactly once, and has minimum total weight. Figure 3.1 shows an example involving four cities ($n = 4$). Note that although the TSP is easy to describe and represent, it is very difficult to solve due to the enormous number of $n!/2$ possible tours. To overcome the prohibitive computational cost for the exact solution, heuristic approaches – such as GAs – are often used in practice [59, 72, 85, 89].

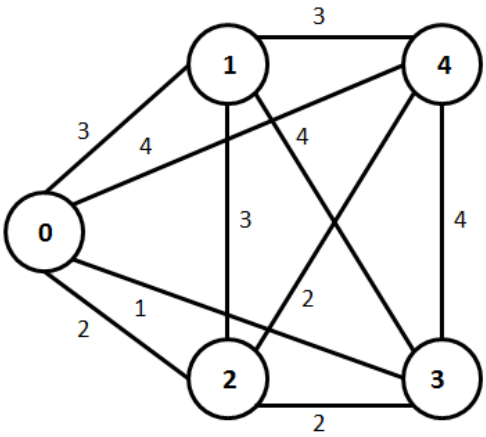


Figure 3.1: An example of TSP with $n = 4$. The optimal tour is 0-1-4-2-3-0 (or 0-3-2-4-1-0) with total weight 11

In the remainder of this section we show how to design a GA to solve permutation problems. In order to accomplish this task, we will look at the operation of Algorithm 1 in detail, by separately examining each of its component parts: population initialization (line 1), fitness computation (lines 2 and 6), parent selection (line 4), genetic operators (line 5) and population replacement scheme (line 7). The discussion is illustrated with examples that specifically regard the TSP, since it shares many similarities with the LSOP. Before that, however, we discuss the individual representation of the proposed GA.

3.2.1 Individual Representation

The design of any GA starts with the definition of the representation of a solution to the problem at hand in the form of a *chromosome*. In essence, this corresponds to choosing an adequate *data structure* for representing a solution. As a pragmatic rule of thumb, [39] advises "choosing a simple representation that is as close as possible to the natural representation of solutions in the target problem".

With regard to permutation problems, the most natural and popular approach is to represent a chromosome as a vector of integers, according to one of the following two standard encodings [28]: *path representation* and *ordinal representation*. In the first encoding scheme, the ith element of the permutation is stored at the ith position in the array. In the latter, the value at the ith position in the array denotes the position of the ith element in the permutation. For instance, Figures 3.2a and 3.2b show, respectively, the chromosome representation of the TSP tour 0-1-4-2-3-0 according to the path and the ordinal representations. Note that, in both cases, the home city is not included in the chromosome to avoid redundancy[1]. Other, less popular, methods for representing permutations can be found in [72].

(a) (b)

Figure 3.2: Representation of the TSP tour 0-1-4-2-3-0 using two distinct chromosome encoding schemes: (a) path representation and (b) ordinal representation

3.2.2 Population Initialization

As shown in Algorithm 1, the first step of the GA search process corresponds to the creation of an initial population of chromosomes (i.e., a set of possible solutions to the given problem) which will evolve over successive generations in an attempt to find an optimized solution to the problem. In most GAs, the initial population is simply generated at random, although it is also possible to make use of heuristics that ensure that, at least, a fraction of the chromosomes have some desirable characteristics [73]. Typical real-world problems have a population of several dozen or even hundreds chromosomes [4, 28, 39].

3.2.3 Fitness Computation

The second step of the GA process consists in evaluating each member of the initial population by a fitness function, which is responsible for assigning a numerical quality value to each of them. This enables chromosomes to be compared against each other. Subsequently, during the actual evolutionary process (lines 3–8 of Algorithm 1), the fitness function provides a basis for the creation of new populations of candidate solutions. Basically, individuals that will form a new population are generated from the "genetic

[1] It is also to simplify the design of genetic operators, as will be shown in Section 3.2.5.

material" of the fittest members in the current population, mimicking the behavior of natural selection in nature.

In most optimization problems, the definition of the fitness function is straightforward: it usually corresponds to a well defined mathematical function. For instance, in the case of the TSP, the fitness function is simply defined as the distance (length) of the tour encoded in a chromosome. The smaller the value, the better the individual. Nonetheless, there exist certain problems in which more than one objective must be taken into account in order to determine the quality of a candidate solution [31]. In such multi-objective problems, the definition of the fitness evaluation function is less trivial. This theme is discussed in Chapter 6.

3.2.4 Parent Selection

Parent selection is the step in the GA evolutionary cycle responsible for defining the individuals in the current generation that will be combined in order to produce offspring (new individuals). As mentioned in the previous subsection, the essential idea is that the fittest individuals of a population must have higher probability of being selected as parents, thus passing their genetic material to later generations. There are three popular kinds of selection methods [28, 32]: proportionate selection (a.k.a. roulette wheel), ranking and tournament. These are introduced and compared below.

In proportionate selection, each member of the population is assigned a probability of selection that is simply given by its fitness value divided by the sum of the fitness of all other chromosomes in the population. In spite of being simple and intuitive, the method is seldom used in practice because it has several drawbacks [32]. First, it assumes a maximization problem, requiring the fitness function to be modified if it is not the case. Second, it requires the computation of a global statistic: the sum of the fitness of all chromosomes in the population. This reduces the potential for parallel implementation of the GA. Third, in problems where most individuals have similar fitness values, selection will be almost random. On the other hand, when there exist one or very few "superperformer" individuals in a population (maybe representing local optimal solutions), these will have a much greater probability of selection. As a result, the population may prematurely converge to be dominated by copies of such individuals.

Ranking selection is an alternative method that works in two steps. First, the individuals of the population are sorted in descending (for a maximization problem) or ascending (for a minimization problem) order according to their fitness values. Next, selection is

performed with probability proportional to their rank positions (ignoring the actual fitness value), by following a predetermined probability distribution function, such as the ones shown in Figure 3.3 (example obtained from [50]). Although the method overcomes most problems associated to proportionate selection, it also requires the computation of a global statistic (rank position of individuals), difficulting a parallel implementation of the GA.

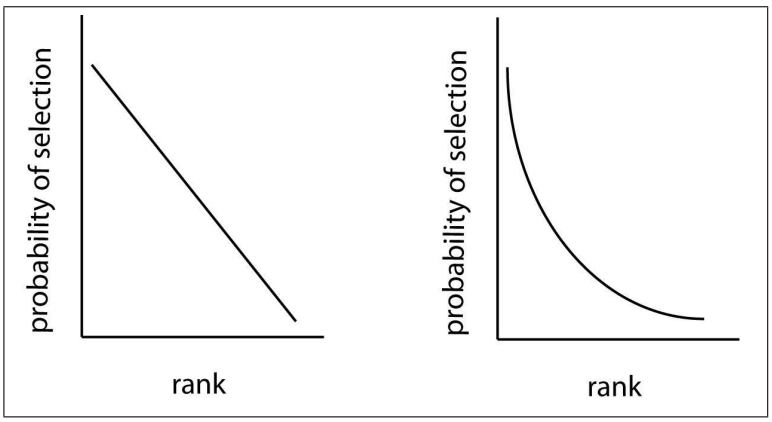


Figure 3.3: Two examples of probability distribution functions for ranking selection

Tournament selection is the most adopted parent selection technique nowadays [28]. In this approach, the GA randomly chooses k individuals from the population, where k is a user-specified parameter called *tournament size*. These individuals "play a tournament" which consists of a comparison of their fitness values. The winner is the individual with the best fitness among the k participants. In spite of being conceptually similar to the ranking method, tournament selection offers the advantage of not requiring the computation of a global statistic. Another appealing property is that the parameter k allows for a more direct control over the *selective pressure* [28, 32], a measure of how often the top individuals are selected to be parents in comparison with the weaker ones. The larger the value of k, the stronger the selective pressure.

3.2.5 Genetic Operators

Once the parents have been selected, the subsequent step in the GA cycle is to generate a new set of candidate solutions (the offspring) by applying the crossover and mutation genetic operators. There are many distinct crossover and mutation techniques available in the literature [19, 28, 39, 72]. Each one is suitable for a specific kind of chromosome representation. In the subsections below, we present examples of strategies suitable for permutation problems where chromosomes are encoded using the path representation.

3.2.5.1 Crossover

Crossover is the most important genetic operator in GAs. It is applied to a selected pair of parents in order to produce one or more children (new individuals), which inherit "genetic material" from both parents. The rationale is that by "mating" two individuals P_1 and P_2 with different but desirable characteristics, it will be possible to produce a new fitter individual that combines the best characteristics of P_1 and P_2. In the following, we describe two different crossover techniques designed for permutation problems, namely donor-receptor crossover (DRC) [64] and order crossover (OC) [17]. A few other crossover strategies proposed for permutation problems using the path representation and other encoding schemes can be found in [28, 72].

DRC is a simple method that generates one child from two parents. An example illustrating how it works is given in Figure 3.4. One of the selected parents plays a role of donor of genetic material. The second parent is cloned and its copied version acts as a receptor. The crossover operation is performed in three steps. First, a sub-chain is chosen at random on the donor individual. Next, the elements of the sub-chain are removed from the receptor individual. At last, the child is generated by inserting the donor's sub-chain at a random position into receptor.

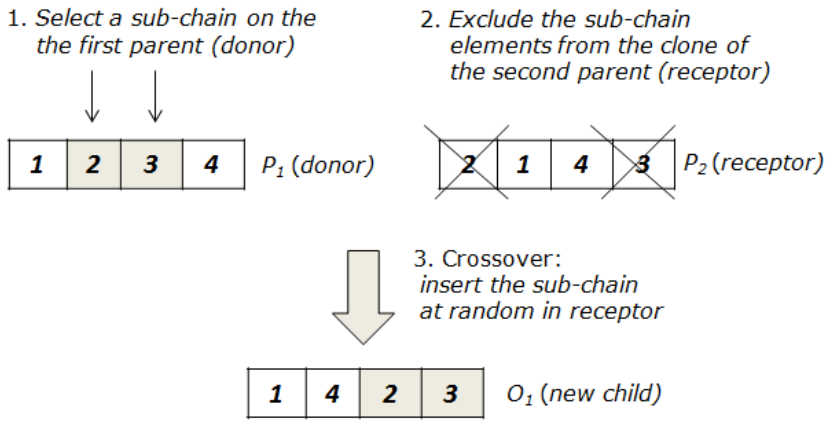


Figure 3.4: DRC Crossover

Unlike DRC, the OC technique generates two children from two parents. This is, probably, the kind of crossover method that has been most used in permutation problems [28]. Although OC is a bit more complex than DRC, it offers the important advantage of producing children that preserve the relative order of the permutations encoded in both of their parents. To explain how the method operates, consider the example shown in Figure 3.5. As aforementioned, the OC approach generates two children (represented by O_1 and O_2 in Figure 3.5) from two parents (represented by P_1 and P_2). Initially, two crossover

points (represented by the two vertical thick lines in Figure 3.5) are chosen at random. The first step to generate O_1 is to copy the segment between the crossover points from P_1 into O_1 (Figure 3.5a). The second step consists in filling the remainder empty positions in O_1 with genetic material from P_2 (Figure 3.5b). The procedure works as follows. Starting from the position next to the second crossover point (the fourth position in our example), the values that are present in P_2 but are not contained in O_1 are transferred to the empty positions in O_1, wrapping around when the last position of both chromosomes is reached. As shown in Figure 3.5, the second child, O_2, is analogously generated.

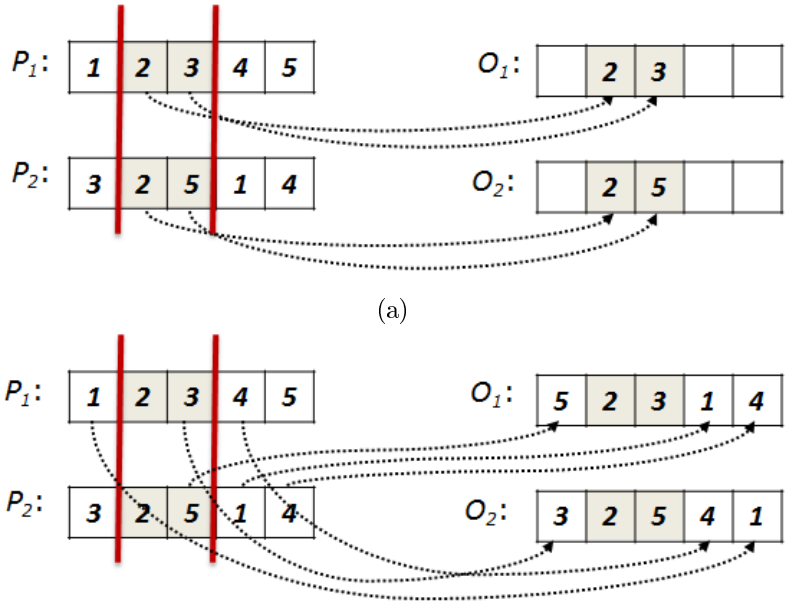


Figure 3.5: Order Crossover: (a) step 1 and (b) step 2

GAs make use of a parameter named *crossover rate* (p_c) [28] to control the frequency with which crossover is applied. This parameter is usually defined in the range between 0.5 and 1.0 and is employed as follows. Each selected pair of parents is given a random number r from [0,1). If $r \leq p_c$, then the crossover operation is normally applied. Otherwise, the children are created "asexually", by simply generating clones of the parents. In the case of the DRC approach, the single child is created as a clone of the donor individual.

As a final remark, it is important to observe that DRC and OC are nondeterministic in their behavior, since in both methods the choice of which pieces of the parents will be combined and transferred to their children is random. Actually, this is a property common to all crossover methods [19, 28, 39].

3.2.5.2 Mutation

Children resulting from crossover can be also subject to mutation. This operator is applied to one individual, transforming it into a slightly modified mutant. Figure 3.6 presents three examples of mutation methods for permutation problems: swap, insert and scramble [28, 72]. Swap mutation (Figure 3.6a) consists in randomly picking two positions in the individual and swap their values. The insert mutation (Figure 3.6b) randomly picks two positions and moves one next to the other, shifting the remainder ones. In the scramble mutation (Figure 3.6c), the entire chromosome or some randomly subset of positions within it, have their contents scrambled.

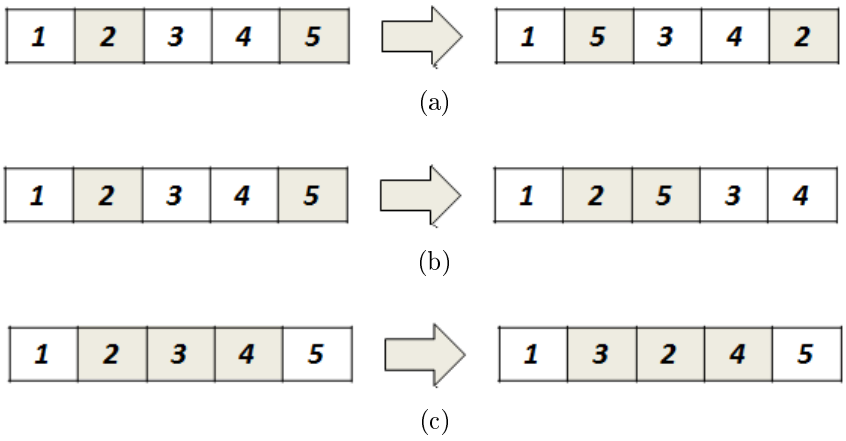


Figure 3.6: Three examples of mutation methods for permutation problems: (a) swap, (b) insert and (c) scramble.

Similar to crossover, a parameter named *mutation rate* (p_m) defines the probability that a chromosome undergoes mutation. Also similar to crossover, mutation is a nondeterministic operation, since the piece that will be mutated within a candidate solution is chosen randomly.

3.2.6 Population Replacement

After the application of crossover and mutation, two sets of individuals will be available: the old population of chromosomes and the brand new offspring set. The last step in the GA cycle is to produce a new generation of individuals taking into consideration both sets (*fitness based replacement*) or only the latter set (*age-based replacement*) [28].

In fitness based replacement, the sets of old individuals and offspring are joined in a unified multiset. This multiset is ranked according to the fitness of its members and the μ

fittest individuals are selected to compose the next generation, where μ corresponds to the population size. In other words, at the end of each cycle, individuals from the offspring set compete based on their fitness with the old ones for a place in the next generation.

On the other hand, the age-based replacement scheme works by simply replacing the entire population by the offspring on each generation. Thus, in this approach, each individual exists for just one generation. The problem is that the top best individuals of a generation may "die" without producing offspring (due to probabilistic selection). In order to solve this problem, an *elitist* strategy may be adopted, where a small set of best individuals in the old population set (named elite individuals) are copied unaltered to the next generation.

It is worth pointing out that, independently of the adopted strategy and differently from natural selection, the size of the population in GAs usually remains constant from one generation to the next [4]. Hence, there is no chance of the chromosome population become either overgrown or extinct during the many iterations of the GA cycle.

Chapter 4

Multi-Label Chain Classifiers

The BR approach, introduced in Chapter 2, represents a simple solution to the MLC problem, yet offering the advantages of being effective in many application domains [66], scalable to large datasets and algorithm independent [113]. However, it has the serious disadvantage of ignoring the possible relationships among labels. This chapter is devoted to the chaining classification model (a.k.a. classifier chains model – CC), a direct extension of the BR approach, which is capable of exploiting label relationships though.

The text is divided as follows. Section 4.1 introduces the basic approach for building multi-label chain classifiers as it was originally proposed in [82, 83]. The CC's training and classification mechanisms are first presented through the use of an illustrative example and then described in pseudocode. At the end of the section, the main advantages and disadvantages associated to the CC method are outlined. Section 4.2 reports and discusses an experiment that, for the first time, investigated in depth the influence of the label sequence in the predictive accuracy of CC models. The results confirm that the use of an optimized label sequence is actually a key factor in inducing effective chain classifiers. In Section 4.3, we propose and evaluate a few baseline heuristics for the determination of optimized label sequences. Subsequently, in Section 4.4, we examine different techniques proposed in the literature to improve the effectiveness of the basic CC method. Concluding remarks are given in Section 4.5.

4.1 An Introduction to the Classifier Chains Method

The CC method was originally conceived in [82, 83]. As with BR, CC is a problem transformation approach that: (i) decomposes the multi-label problem into q single-label binary problems; (ii) trains one binary classifier for each label; and (iii) determines the

labelset of new objects by combining the outputs produced by each classifier. However, differently from BR, the binary classifiers in CC are not isolated from each other. Instead, they are linked in a chain structure which allows each one to be able to communicate their predictions to the other binary classifiers ahead in the chain. Next, we discuss the steps involved in the CC's training and classification processes, once again making use of the hypothetical music categorization dataset introduced in Figure 2.1.

The first step of the CC's training procedure consists in generating a randomly-ordered chain that must contain all the q labels involved in the classification problem. For instance, considering the dataset of Figure 2.1, an example of valid chain would be $C = \{Metal \rightarrow Jazz \rightarrow Bossa \rightarrow Pop\}$. Once the chain has been defined, q binary classifiers are trained, one for each label, according to the chain sequence. The first binary classifier, y_1, is trained using solely the attributes that compose the feature set X as its input attributes. This classifier will be responsible for the prediction of the first label in the chain ("Metal", according to the example chain C). The second binary classifier, y_2, is trained using X augmented with the binary information of the first label in the sequence (in this example, the true values of label "Metal" in the training set) as its input attributes. This second binary classifier will be responsible for the prediction of the second label in the chain ("Jazz", considering the example chain). Each subsequent classifier y_j is trained using X augmented with the information of $j-1$ labels as its input attributes, i.e., the feature space of y_j is extended with the true label information of all previous labels in the chain.

The CC's classification process must also be executed according to the same chain sequence employed in the training phase. To predict the labelset of a new object, q binary classifications are performed, with the process beginning at the classifier associated to the first label in the sequence and going along the chain. For example, Figure 4.1 illustrates the classification of the song $t =$ 'The Girl from Ipanema" considering a hypothetical CC model trained with the sequence C. In this figure, y_1, y_2, y_3 and y_4 respectively represent the trained binary classifiers to predict the genres "Metal", "Jazz", "Bossa" and "Pop" and x represents the set of features describing t. The classification process begins at y_1 and goes along the chain, i.e., the classifier y_j predicts the relevance of label l_j, given the feature space augmented by the predictions carried out by the previous $j-1$ classifiers. Observe that, differently from the BR model, the binary classifications performed by a CC model are not independent of each other, because each classifier communicates its decision (binary prediction) to the subsequent classifiers in the chain. In the example of Figure 4.1, the binary classification of y_1 (which predicted that t is *not* a heavy metal song) is incorporated into the set of features x, becoming immediately available to be

considered as additional predictive information by y_2 and the remaining binary classifiers. Despite its simplicity, the example can immediately highlight the advantage of allowing communication among the binary models. Note that a classifier such as y_3, which is placed near the end of the chain, can clearly benefit from having been informed about the decisions of both y_1 and y_2. Thus, the CC model is able to generate predictions different from the ones generated by a BR model.

t = *"The Girl from Ipanema"* – Tom Jobim & Frank Sinatra

Classifiers	Classifications
y_1: x → {Metal, ~Metal}	~Metal
y_2: x ∪ ~Metal → {Jazz, ~Jazz}	Jazz
y_3: x ∪ ~Metal ∪ Jazz → {Bossa, ~Bossa}	Bossa
y_4: x ∪ ~Metal ∪ Jazz ∪ Bossa → {Pop, ~Pop}	Pop
Predicted Labelset	***{Jazz, Bossa, Pop}*** $\subseteq L$

Figure 4.1: CC classification of the song "The Girl from Ipanema"

4.1.1 Algorithm Specification

Algorithm 2 formalizes the CC's training procedure in pseudocode. This algorithm requires a training set D as the only input parameter and produces a multi-label chain classifier h as output. In the adopted notation, assume that there is a set $L = \{l_1, ..., l_q\}$ of q class labels involved in the target classification problem. Also assume that the training set D is composed of N instances, where each instance i has the form $(x_1^{(i)}, ..., x_d^{(i)}, \iota_1^{(i)}, ..., \iota_q^{(i)})$. Consider that $(x_1^{(i)}, ..., x_d^{(i)})$ is a vector that stores values for d predictive attributes describing i whereas $(\iota_1^{(i)}, ..., \iota_q^{(i)})$ is a label relevance vector, with $\iota_j^{(i)} \in \{0, 1\}$ being the jth label assignment (1 if label l_j is relevant to i; 0 otherwise)[1].

The algorithm works as follows. First, a randomly-ordered label sequence C is defined (line 1). This sequence must contain all the q labels involved in the classification problem. Then, the FOR loop that encompasses lines 3-11 is responsible for inducing a binary classifier for each label l_j, following the order specified in C. The process is divided into two phases. In the first (lines 4-8), a dataset D'_j, derived from D, is generated. This dataset is subsequently used in the second phase (lines 9-10) to train y_j, the binary classifier responsible for predicting the relevance of l_j (the label being processed in current iteration in the FOR loop). Observe that the set of predictive attributes in each D'_j

[1] In order to facilitate the description and explanation of the CC's training procedure, we decided to represent the subset of labels associated to each training instance as a binary vector of length q. This notation differs from that adopted in Chapter 1, where we used a subset of label identifiers.

(denoted as x') comprises the original feature set augmented with the binary information of the $j-1$ labels processed in the previous iterations. Once all labels have been processed, Algorithm 2 returns h (line 7), a multi-label chain classifier composed of q binary SLC classifiers, each one trained with a distinct and specific set of input attributes (defined according to C).

It is worth claryfing that each binary classifier $y_j \in h$ is induced considering the actual label values present in the original training set D. Nonetheless, as pointed out by [67, 91], it would also be possible to train a classifier using the estimations of l_j produced by other classifier (e.g., an external BR model). Actually, this approach has been evaluated in the above works, but it has demonstrated to be less effective.

Algorithm 2 CC's training procedure

Input : D (training set)
Output: h (multi-label chain classifier induced from D)

1: generate a random label sequence $C = \{l_1 \rightarrow l_2 \rightarrow ... \rightarrow l_q\}$
2: $h \leftarrow \emptyset$
3: **for all** labels l_j according to the order specified in C **do**
4: $D'_j \leftarrow \emptyset$
5: **for** $i = 1$ **to** N **do**
6: $x' \leftarrow [x_1^{(i)}, ..., x_d^{(i)}, \iota_1^{(i)}, ..., \iota_{j-1}^{(i)}]$
7: $D'_j \leftarrow D'_j \cup (x', \iota_j^{(i)})$
8: **end for**
9: train a binary classifier y_j to predict the relevance of l_j using D'_j.
10: $h \leftarrow h \cup y_j$
11: **end for**
12: **return** h

Algorithm 3 describes the classification procedure employed in the CC method. The following three input parameters are required: h – a trained CC model; C – the label sequence used to train h; t – the new instance to be classified. The algorithm produces as output Z – the predicted labelset for instance t. In Algorithm 3, the notation $\hat{l}_j$ is used to represent the $0/1$ relevance of label l_j predicted by the binary classifier $y_j \in h$.

4.1.2 Pros and Cons

The CC method offers a considerable number of advantages. The most important is that, in spite of its simplicity, a comprehensive recent study comparing several state-of-the-art methods for MLC [66] demonstrated that CC is among the top best performing

Algorithm 3 CC's classification procedure

Input : h (trained CC model), C (label sequence used to train h), t (instance to be classified)
Output: Z (the predicted labelset for instance t)

1: $Z \leftarrow \emptyset$
2: **for all** labels l_j according to the order specified in C **do**
3: $\quad t' \leftarrow t[x_1, ..., x_d, \hat{l}_1, ..., \hat{l}_{j-1}]$
4: $\quad \hat{l}_j \leftarrow y_j(t')$
5: $\quad Z \leftarrow Z \cup \hat{l}_j$
6: **end for**
7: **return** Z

algorithms in terms of predictive performance. Furthermore, CC still maintains most of the attractive characteristics of BR: it is algorithm independent, scales linearly with q and can be easily parallelizable for better time performance. Not surprisingly, a recent comprehensive survey on multi-label classification included CC in the family of the topmost representative methods for MLC [113]. Indeed, over the last few years, a considerable number of variations of the basic CC model have been proposed in the literature [16, 21, 48, 58, 63, 78, 79, 80, 91, 110].

However, there are two important drawbacks in the basic CC approach. First, the label ordering is decided at random instead of being selected using an "intelligent" method. Second, the CC method imposes that all labels must be present in the chain, even the ones that might carry irrelevant information with respect to the prediction of the other labels. The first drawback is the object of study of the next subsection whilst the latter is covered in Chapter 6.

4.2 The Label Sequence Issue

In the original CC method, the label sequence is decided at random. This has often been considered a major drawback, even noted by the authors of CC themselves, which deemed that if the first members of the chain have low accuracy (i.e., if they output many wrong predictions), error propagation will occur along the chain causing a significant decrease in predictive accuracy [82, 83]. In a similar vein, [58, 69, 78] argued that different label orderings can lead to different results in terms of predictive accuracy mainly due to finite sample effects. For example, if a label l_j is rare, then it may lead to the induction of an unreliable binary model, which should not be placed in the beginning of the chain. On

the other hand, the authors of [93] have a completely different belief. They consider that the effect of the chain order will be very small when the number of features in the dataset is much higher than the number of labels (which corresponds to the most typical situation in real-world applications, as seen in Section 2.2).

Nevertheless, [11] realized that "the effect of different orders on the predictive performance of the CC method has not yet been studied in depth". Motivated by this consideration and by the conflicting views of [93] and [58, 78, 82, 83], we decided to carry out an exhaustive experiment to examine the influence of the label sequence in the predictive accuracy of CC models. The experiment consisted in assessing the predictive accuracy of CC considering all $q!$ label permutations of three benchmark datasets using the following single-label base algorithms[2]: k-NN [106], C4.5 [75], Naïve Bayes [26], and SMO (an SVM algorithm) [71]. The main goal is to observe the differences in predictive accuracy between the best (most accurate) and the worst (less accurate) chain sequences for CC models built using each of the datasets and base classifiers. If most of the differences are large, then there is evidence that the use of an optimized label sequence is actually important for training a CC model. In the experiment, the predictive performance is determined in terms of the Accuracy measure, defined in Equation 2.3 (a brief note on results for other measures is mentioned at the end of the section).

The experiment was carried out using the implementation of the CC method available in the Mulan tool [99], an open source platform for the evaluation of multi-label algorithms developed in Java that works on top of the well-known Weka API for data mining [46]. The datasets "flags" ($q = 7, d = 19, N = 194$), "emotions" ($q = 6, d = 72, N = 593$), and "scene" ($q = 6, d = 294, N = 2407$), previously described in Section 2.2, were used in this experiment. Since they have a small number of labels, it became feasible to build and test CC models for all possible label permutations (a total of 5040 distinct chain orderings for "flags" and 720 for both "emotions" and "scene"). In our experiment, the CC models were evaluated by applying the holdout method using the training and test partitions supplied with the datasets[3].

Tables 4.1, 4.2 and 4.3 present the results for the datasets "flags", "emotions" and "scene", respectively. In these tables, the first column indicates the name of the base algorithm (the acronym "x-NN" is used to refer to the k-NN algorithm configured with $k = x$; "NB" is used to refer to the Naïve Bayes algorithm). The values in the second

[2]We selected these four SLC algorithms based on two criteria: (i) usage by the community, and (ii) the representation of different underlying principles for addressing the classification task.

[3]Datasets obtained from the Mulan repository: http://mulan.sourceforge.net/datasets-mlc.html.

Table 4.1: Results of the exhaustive experiment considering all possible label permutations in the chain for different base classifiers according to the Accuracy measure: FLAGS dataset

Base Algorithm	**Best**	**Worst**	**Avg**	**SD**	**Best-Worst**	**Best-Avg**
C4.5	0.622	0.496	0.569	0.023 (1)	0.126 (1)	0.053 (3)
1-NN	0.531	0.531	0.531	0.000 (8)	0.000 (8)	0.000 (8)
3-NN	0.622	0.516	0.563	0.016 (2)	0.106 (2)	0.059 (1)
5-NN	0.628	0.534	0.578	0.012 (6)	0.094 (4)	0.050 (4)
7-NN	0.614	0.510	0.558	0.013 (4)	0.104 (3)	0.056 (2)
9-NN	0.601	0.508	0.564	0.013 (4)	0.093 (5)	0.037 (6)
NB	0.576	0.487	0.537	0.015 (3)	0.089 (6)	0.039 (5)
SMO	0.607	0.522	0.589	0.007 (7)	0.085 (7)	0.018 (7)

and third columns ("Best" and "Worst") represent the Accuracy value obtained by the best and the worst classifier chains model, respectively. The fourth and fifth columns show, respectively, the mean Accuracy and the standard deviation. The sixth column ("Best-Worst") gives the difference of the best and the worst accuracies and the seventh ("Best-Avg") the difference between the best and the mean accuracies. In the three last columns, the values between parentheses are used to rank the standard deviation and the values of the computed differences (the smaller the rank value, the greater the standard deviation or the computed difference).

The exhaustive experiment revealed that, overall, the order of the chain indeed has a strong effect on predictive performance. Nonetheless, the obtained results also evidence that the different base algorithms, due to their own characteristics, are affected to different degrees by the use of distinct label orderings. In this sense, it has been observed that the effect tends to be large when the base algorithm is C.45. Observe that in the "flags" dataset the difference between the best and the worst Accuracy values for C4.5 is above 12%. Considering the other datasets, C4.5 also presented the largest difference between the best and the worst chains in "emotions" and the second largest in "scene". Furthermore, the standard deviation values were consistently large when compared with the majority of the other base algorithms as well as the values of the differences between the best and the mean accuracies.

The above behavior can be explained by the fact that C4.5 adopts a greedy recursive strategy, based on attribute importance, for selecting the attributes that will be placed closer to the root of the decision tree (i.e., the attributes that are initially evaluated to determine the class to which a new instance belongs). Thus, if two labels l_i and l_j are

Table 4.2: Results of the exhaustive experiment considering all possible label permutations in the chain for different base classifiers according to the Accuracy measure: EMOTIONS dataset

Base Algorithm	Best	Worst	Avg	SD	Best-Worst	Best-Avg
C4.5	0.538	0.406	0.467	0.025 (2)	0.132 (1)	0.071 (1)
1-NN	0.493	0.493	0.493	0.000 (8)	0.000 (8)	0.000 (8)
3-NN	0.584	0.498	0.541	0.016 (3)	0.086 (3)	0.043 (3)
5-NN	0.596	0.531	0.564	0.012 (4)	0.065 (5)	0.032 (4)
7-NN	0.602	0.531	0.571	0.012 (4)	0.071 (4)	0.031 (5)
9-NN	0.606	0.544	0.579	0.012 (4)	0.062 (5)	0.027 (6)
NB	0.544	0.518	0.531	0.004 (7)	0.026 (7)	0.013 (7)
SMO	0.617	0.486	0.550	0.031 (1)	0.131 (2)	0.067 (2)

Table 4.3: Results of the exhaustive experiment considering all possible label permutations in the chain for different base classifiers according to the Accuracy measure: SCENE dataset

Base Algorithm	Best	Worst	Avg	SD	Best-Worst	Best-Avg
C4.5	0.603	0.550	0.578	0.010 (2)	0.053 (2)	0.025 (2)
1-NN	0.637	0.637	0.637	0.000 (8)	0.000 (8)	0.000 (8)
3-NN	0.684	0.662	0.671	0.005 (6)	0.022 (6)	0.013 (6)
5-NN	0.697	0.665	0.681	0.006 (4)	0.032 (4)	0.016 (4)
7-NN	0.694	0.666	0.679	0.006 (4)	0.028 (5)	0.015 (5)
9-NN	0.708	0.663	0.683	0.010 (2)	0.045 (3)	0.025 (2)
NB	0.472	0.467	0.469	0.001 (7)	0.005 (7)	0.003 (7)
SMO	0.692	0.611	0.655	0.018 (1)	0.081 (1)	0.037 (1)

strongly correlated, either positively or negatively, it is likely that l_i will be identified by C4.5 as an important attribute to discriminate the 0/1 relevance of l_j (and vice-versa). Consequently, if l_i comes before l_j in a chain sequence, there is a high probability for C4.5 selecting l_i as one of the topmost nodes of the decision tree to classify l_j. Figure 4.2 gives an illustrative example considering the "flags" dataset. Recall from Section 2.2 that the classification task associated to this dataset is to predict the colors present in a national flag. Figure 4.2 shows two distinct decision trees to classify the label "yellow" that were built during our exhaustive experiment. Each one was induced from two different kinds of label permutations. Figure 4.2a shows the tree built utilizing any label sequence of the form $\{yellow \rightarrow white \rightarrow ...\}$ (i.e., in which the first element is "yellow" and the second "white"). On the other hand, Figure 4.2b presents the tree built with the use of any label sequence of the form form $\{white \rightarrow yellow \rightarrow ...\}$ (the first element is "white" and the

second “yellow”). Observe that the tree from Figure 4.2b is much smaller than the one from Figure 4.2a, also having a lower error rate (9.30% against 12.40%). Note also that, in the smaller tree, “white” was selected by C4.5 as the second most relevant attribute to predict the 0/1 relevance of “yellow”. We further performed a correlation analysis on the attributes of the “flags” training set and confirmed that “yellow” and “white” exhibit one of the strongest negative correlations taking into consideration all attributes and labels that compose this dataset (more details in Appendix A).

```
colours <= 4
|   icon = 0
|   |   landmass = 1
|   |   |   triangle = 0: 0 (10.0)
|   |   |   triangle = 1: 1 (4.0/1.0)
|   |   landmass = 2
|   |   |   bars <= 1: 1 (7.0/2.0)
|   |   |   bars > 1: 0 (2.0)
|   |   landmass = 3
|   |   |   sunstars <= 0
|   |   |   |   area <= 246: 0 (10.0)
|   |   |   |   area > 246: 1 (3.0/1.0)
|   |   |   sunstars > 0: 1 (2.0)
|   |   landmass = 4
|   |   |   language = 1: 0 (3.0)
|   |   |   language = 2: 0 (0.0)
|   |   |   language = 3: 1 (7.0/1.0)
|   |   |   language = 4: 0 (0.0)
|   |   |   language = 5: 0 (0.0)
|   |   |   language = 6: 1 (2.0)
|   |   |   language = 7: 0 (0.0)
|   |   |   language = 8: 0 (5.0/1.0)
|   |   |   language = 9: 0 (0.0)
|   |   |   language = 10
|   |   |   |   area <= 925: 0 (8.0/1.0)
|   |   |   |   area > 925: 1 (2.0)
|   |   landmass = 5: 0 (18.0/2.0)
|   |   landmass = 6
|   |   |   colours <= 3: 0 (6.0/1.0)
|   |   |   colours > 3: 1 (2.0)
|   icon = 1: 1 (17.0/6.0)
colours > 4: 1 (21.0)
```

(a)

```
colours <= 4
|   white = 0: 1 (33.0/6.0)
|   white = 1
|   |   colours <= 3: 0 (53.0/4.0)
|   |   colours > 3
|   |   |   landmass = 1: 0 (2.0/1.0)
|   |   |   landmass = 2: 1 (2.0)
|   |   |   landmass = 3: 1 (2.0)
|   |   |   landmass = 4: 0 (4.0/1.0)
|   |   |   landmass = 5: 0 (9.0)
|   |   |   landmass = 6: 1 (3.0)
colours > 4: 1 (21.0)
```

(b)

Figure 4.2: Two decision trees built from the FLAGS training set using the C4.5 algorithm to classify the label *yellow*: (a) tree built using label sequences of the form $\{yellow \rightarrow white \rightarrow ...\}$ and (b) tree built using label sequences of the form $\{white \rightarrow yellow \rightarrow ...\}$

In regard to the other base algorithms, the exhaustive experiment indicated that the effect of the chain ordering was rather small for Naïve Bayes in the datasets “emotions” and “scene”, where the number of attributes is much larger than the number of labels. This can be explained by the fact that, unlike C4.5, Naïve Bayes gives the same weight to all predictive attributes, whose probabilities (conditioned on the target class label) are multiplied in the numerator of the Bayes’ Formula, attenuating the effect of different label

permutations.

On the other hand, SMO presented the largest difference between the best and the worst accuracies in the "Scene" dataset and the second largest in "Emotions". It also presented the largest standard deviation in the same two datasets. This algorithm is used for training Support Vector Machine classifiers, whose main idea is to find maximal marginal hyperplane (the optimal decision boundary separating data points belonging to two distinct classes). The results obtained in our experiments suggest that, for the datasets "emotions" and "scene", the boundary and the support vectors frequently change as the label sequence is modified.

The algorithm k-NN presented moderate to large differences for most configurations of k in the three datasets; however, it was observed that the effect of the label sequence is null if $k = 1$. This behavior occurs because, when $k = 1$, the nearest neighbor of a test instance will always be the same for each of the binary classifications performed by the CC model, independently of the chain sequence. For example, consider the classification of a new instance t using the chain sequence $\{l_1 \rightarrow l_2 \rightarrow ... \rightarrow l_q\}$. In the first binary classification (prediction of l_1), suppose that instance I is the nearest neighbor of t. So, the value of l_1 in I must be assigned to t. As a consequence I will continue to be the nearest neighbor of t and the value of l_2 in I will be assigned to t. Thus, I keeps being the nearest neighbor of t and it will be until the classification of the last label in the chain.

The exhaustive experiment also allowed us to identify that, in general, an effective label sequence for a specific base algorithm (e.g.: C4.5) does not necessarily constitute a good sequence for other base algorithm (e.g.: SMO). For instance, the experiments over the "flags" dataset revealed that the label sequence that leads to the best Accuracy value for C4.5 is only ranked as the 248^{th} best sequence for SMO. Similarly, the best label sequence for SMO is ranked as the 114^{th} best sequence for C4.5. In Tables 4.4, 4.5, 4.6 we extend this comparison by performing an analysis involving the 20 best ranked sequences in terms of the Accuracy measure for the algorithms C4.5, 5-NN, Naïve Bayes and SMO in the datasets "flags", "emotions" and "scene", respectively. In these tables, each cell i, j (where i represents a line and j a column) denotes the number of top-20 label sequences for algorithm j that also belong to the set of top-20 label sequences for algorithm i. Observe that the majority of cells contain a zero value, indicating that none of the very best sequences for a specific algorithm also belong to the set of topmost effective sequences for the other algorithms.

In summary, the exhaustive experiment allowed us to derive the following conclusions

Table 4.4: Number of top-20 label sequences for algorithm j (column) that also belong to the set of top-20 label sequences for algorithm i (line): FLAGS dataset

Base Algorithm	**C4.5**	**5-NN**	**NB**	**SMO**
C4.5	-	2	0	0
5-NN	8	-	0	1
NB	0	0	-	0
SMO	4	3	0	-

Table 4.5: Number of top-20 label sequences for algorithm j (column) that also belong to the set of top-20 label sequences for algorithm i (line): EMOTIONS dataset

Base Algorithm	**C4.5**	**5-NN**	**NB**	**SMO**
C4.5	-	0	1	0
5-NN	0	-	0	4
NB	0	0	-	1
SMO	0	4	1	-

Table 4.6: Number of top-20 label sequences for algorithm j (column) that also belong to the set of top-20 label sequences for algorithm i (line): SCENE dataset

Base Algorithm	**C4.5**	**5-NN**	**NB**	**SMO**
C4.5	-	0	3	0
5-NN	0	-	1	1
NB	0	0	-	0
SMO	0	3	10	-

about the influence of the label sequence in the effectiveness of CC models: (i) the use of an optimized label sequence is often important to ensure the effective performance of multi-label chain classifiers; (ii) the different base algorithms, due to their own characteristics, are affected to different degrees by the label sequence (for instance, the effect tends to be very large when the base algorithm is C.45, but it can be rather small for Naïve Bayes); and (iii) a good chain sequence for a given base algorithm is not necessarily good for other base algorithms. It is important to mention that we also ran the same experiment using the measures of Exact Match, F-Measure and Hamming Loss, having obtained equivalent results.

4.3 Baseline Methods to Determine the Label Ordering

In the present section, we propose three baseline strategies designed to determine an optimized label sequence and examine their predictive performance:

1. **PredCC**: In this method, the chain sequence is arranged in descending order of predictive accuracy according to the results of a preliminary BR classification. More specifically, the method has two steps: first, it builds and evaluates an ordinary BR classifier (using only the training set) and next it creates a chain in which the most accurate binary classifiers are placed in the very beginning of the chain. The goal is to attenuate the effect of error propagation along the chain, which is considered by [82, 83] as the major potential drawback of using a randomly-ordered sequence.

2. **FreqCC**: In this method, the most-frequent labels are selected as the very first elements of the chain (the greater the frequency of a label, the closer to the beginning of the sequence it is). The rationale behind this baseline strategy is that, as observed by [58, 69, 78], some labels may have much fewer instances compared to the rest, leading to the induction of unreliable binary models. Thus, placing those underrepresented labels at the end of the chain can benefit the accuracy of the CC model as a whole.

3. **DepCC**: In this method, the goal is to place the labels determined as most dependent on other labels at the end of the sequence. The strategy work as follows: for each label, we determine a score based on the number of correlated labels using the chi-square test for dependence [9, 47]. The sequence is then ordered in ascending order according to the computed score. The rationale is that the higher the number of correlations, the more a label is influenced by other labels. Refer to Appendix A for an introduction to the chi-squared statistics.

In spite of being very simple, the above strategies have never been previously analyzed in the multi-label literature, reinforcing the importance of the present study. The remainder of this section is organized as follows. Section 4.3.1 describes the methodology employed during the empirical analysis which is itself presented in Section 4.3.2.

4.3.1 Experimental Methodology

The goal of the experiment performed in this section is to compare the three proposed baseline strategies against the original CC approach [82, 83]. The methods were evaluated

on 10 benchmark datasets from the collection presented in Section 2.2, excluding, however, the datasets "flags", "emotions" and "scene". These were not included in the experiments because they have a small number of labels, allowing the best chain to be determined by an exhaustive method (as seen in Section 4.2).

Most of the datasets employed in the experiment came divided into training and testing parts, being "cal500", "university" and "thyroid" the only exceptions. Following the approach adopted in the extensive comparison of multi-label methods presented in [66], a holdout evaluation [47, 51] was performed to assess the predictive performance of the methods, using the benchmark datasets with that predefined division, where the training part comprises about 2/3 of the complete dataset and the test part, the remaining 1/3. For "cal500", "university" and "thyroid" we generated the training and test parts.

The experiment was carried out using the Java implementation of CC available in the Mulan tool [99]. J48 [46] with default parameters was used as the base SLC algorithm for the evaluated methods. This corresponds to the Weka's implementation for the C4.5 decision tree technique [75], which has been identified as the single-label algorithm most sensitive to the label sequence in the exhaustive experiment reported in the previous section. The predictive performance of the methods was evaluated in terms of four example-based measures: Accuracy, F-Measure, Hamming Loss and Exact Match. We employed the two-tailed Wilcoxon signed-rank test [51, 103] to verify the statistical significance of the results with a confidence level of 95%. This is a non-parametric test appropriate for comparing pairs of classifiers in multiple domains (datasets). It does not assume normal distribution and works well for small sample sizes [103]. Details on how this test works can be found at Appendix B. Since the sequence used in CC is created randomly, the results reported for CC are averaged over 10 executions, with different random sequences. Our adopted approach differs from experiments involving the CC method in past papers, such as [16, 42, 48, 58, 78, 93, 110], where the chain sequence was simply defined as the "default order" (the order specified in the database).

As a final remark, it is important to mention that all the other experiments in the subsequent chapters of this thesis were carried out using the same benchmark datasets and the same experimental setup described above.

4.3.2 Results

Tables 4.7, 4.8, 4.9, 4.10 present the performance of each method in terms of Accuracy, F-Measure, Exact Match, and Hamming Loss, respectively. The best results for each

dataset are highlighted in bold type. The rank obtained by each method in each dataset is presented in parenthesis whereas the mean rank of the methods is presented in the last line of each table. Although the mean ranks are not used by the Wilcoxon test (as presented in the example of Appendix B), we consider this information provides deeper insight into the overall behavior of each method.

We compared each baseline strategy against the CC method considering the four evaluation measures of predictive performance. In all comparisons, the two-tailed Wilcoxon signed-rank test indicated that, with a confidence level of 95%, no statistically significant differences exist between the performance of the baseline methods and the performance of CC. In other words: overall, none of the baseline strategies to obtain an optimized label sequence is able to significantly outperform the original CC approach, where the label sequence is simply decided at random, according to the four evaluation measures of predictive performance. Furthermore, observe that the differences between the mean ranks of the methods in the four tables is very small (although it is worth highlighting that DepCC obtained the best rank for all metrics), reinforcing that the methods lead to quite similar predictive performances.

The results are enough to conclude that it is necessary to invest in more sophisticated algorithmic solutions to overcome the label sequence optimization problem (LSOP) in multi-label chain classifiers. In the next section, we carefully revise and categorize the different techniques proposed in the literature to address this issue.

Table 4.7: Performance of CC, FreqCC, DepCC and PredCC in terms of Accuracy.

Dataset	Accuracy			
	CC	FreqCC	DepCC	PredCC
university	0.310 (2.0)	0.307 (3.0)	**0.321** (1.0)	0.265 (4.0)
yeast	0.418 (3.0)	0.416 (4.0)	**0.452** (1.0)	0.448 (2.0)
ces-16	0.194 (2.0)	**0.196** (1.0)	0.188 (3.0)	0.180 (4.0)
birds	0.564 (2.0)	0.562 (4.0)	**0.568** (1.0)	0.563 (3.0)
thyroid	0.983 (3.0)	**0.984** (1.0)	0.983 (3.0)	0.983 (3.0)
genbase	**0.987** (2.5)	**0.987** (2.5)	**0.987** (2.5)	**0.987** (2.5)
medical	0.743 (4.0)	0.745 (3.0)	**0.757** (1.5)	**0.757** (1.5)
enron	0.402 (2.0)	0.391 (4.0)	**0.411** (1.0)	0.395 (3.0)
llog	0.239 (4.0)	0.241 (2.0)	0.240 (3.0)	**0.251** (1.0)
cal500	0.218 (2.0)	**0.221** (1.0)	0.200 (4.0)	0.215 (3.0)
average rank	2.65	2.65	**2.10**	2.60

Table 4.8: Performance of CC, FreqCC, DepCC and PredCC in terms of F-Measure.

Dataset	F-Measure			
	CC	FreqCC	DepCC	PredCC
university	0.335 (2.0)	0.333 (3.0)	**0.344** (1.0)	0.279 (4.0)
yeast	0.523 (4.0)	0.524 (3.0)	**0.563** (1.0)	0.553 (2.0)
ces-16	0.249 (2.0)	**0.251** (1.0)	0.242 (3.0)	0.234 (4.0)
birds	**0.592** (2.0)	**0.592** (2.0)	**0.592** (2.0)	0.589 (4.0)
thyroid	0.990 (3.0)	**0.991** (1.0)	0.990 (3.0)	0.990 (3.0)
genbase	**0.991** (2.5)	**0.991** (2.5)	**0.991** (2.5)	**0.991** (2.5)
medical	0.769 (4.0)	0.776 (3.0)	**0.780** (1.5)	**0.780** (1.5)
enron	0.506 (2.0)	0.495 (3.0)	**0.509** (1.0)	0.492 (4.0)
llog	0.255 (4.0)	0.259 (2.0)	0.257 (3.0)	**0.265** (1.0)
cal500	0.348 (2.0)	**0.350** (1.0)	0.325 (4.0)	0.345 (3.0)
average rank	2.65	2.25	**2.20**	2.90

Table 4.9: Performance of CC, FreqCC, DepCC and PredCC in terms of Exact Match.

Dataset	Exact Match			
	CC	FreqCC	DepCC	PredCC
university	0.248 (2.0)	0.238 (3.0)	**0.263** (1.0)	0.225 (4.0)
yeast	0.127 (3.0)	0.113 (4.0)	0.135 (2.0)	**0.147** (1.0)
ces-16	0.058 (2.0)	**0.060** (1.0)	0.057 (3.0)	0.054 (4.0)
birds	0.485 (3.0)	0.477 (4.0)	**0.495** (1.0)	0.486 (2.0)
thyroid	0.941 (2.0)	**0.944** (1.0)	0.938 (4.0)	0.940 (3.0)
genbase	**0.975** (2.5)	**0.975** (2.5)	**0.975** (2.5)	**0.975** (2.5)
medical	0.665 (3.0)	0.653 (4.0)	**0.688** (1.5)	**0.688** (1.5)
enron	0.125 (3.0)	0.109 (4.0)	**0.149** (1.0)	0.142 (2.0)
llog	0.201 (2.0)	0.197 (4.0)	0.199 (3.0)	**0.213** (1.0)
cal500	**0.000** (2.5)	**0.000** (2.5)	**0.000** (2.5)	**0.000** (2.5)
average rank	2.50	3.00	**2.15**	2.35

Table 4.10: Performance of CC, FreqCC, DepCC and PredCC in terms of Hamming Loss.

Dataset	Hamming Loss			
	CC	FreqCC	DepCC	PredCC
university	0.139 (3.0)	**0.134** (1.0)	0.136 (2.0)	0.144 (4.0)
yeast	0.274 (3.0)	0.275 (4.0)	**0.256** (1.0)	0.264 (2.0)
ces-16	**0.188** (2.0)	**0.188** (2.0)	**0.188** (2.0)	0.189 (4.0)
birds	0.051 (3.0)	0.053 (4.0)	0.050 (2.0)	**0.049** (1.0)
thyroid	0.006 (3.0)	**0.005** (1.0)	0.006 (3.0)	0.006 (3.0)
genbase	**0.001** (2.5)	**0.001** (2.5)	**0.001** (2.5)	**0.001** (2.5)
medical	0.011 (3.5)	0.011 (3.5)	**0.010** (1.5)	**0.010** (1.5)
enron	**0.054** (1.5)	**0.054** (1.5)	0.055 (3.0)	0.057 (4.0)
llog	**0.019** (2.5)	**0.019** (2.5)	**0.019** (2.5)	**0.019** (2.5)
cal500	0.176 (3.0)	0.184 (4.0)	0.160 (2.0)	**0.156** (1.0)
average rank	2.60	2.90	**2.05**	2.45

4.4 Extensions to the Classifier Chain Model

Over the last few years, the CC model has become one of the main methods for MLC. A considerable number of variations of the CC basic approach have recently been proposed in the literature [16, 21, 22, 48, 58, 63, 78, 80, 91, 110]. In this section we introduce, compare and discuss these techniques. The text is divided into two sections. Subsection 4.4.1 covers the proposals that improve CC by modifying its training step whereas Subsection 4.4.2 addresses the ones that modify the CC's inference step.

4.4.1 Approaches Based on Training Optimization

The majority of current variations of the basic CC approach try to overcome the LSOP by modifying the CC's training step. The proposals can be categorized into three basic groups: ensemble approach [82, 83], methods that explore candidate chain sequences [48, 63, 80, 91, 110], and methods that search for a single optimized chain sequence [58, 78]. These different families of methods are covered in the next subsections.

4.4.1.1 Ensemble of Classifier Chains

The authors of the original CC model suggest the use of an ensemble of classifier chains (ECC) [82, 83] in order to cope with the label sequence issue. In this approach the individual classifiers vote and the output labelset for a new instance is determined based on the collection of votes. The expectation is that the effect of poorly ordered chains in predictive accuracy will be mitigated. Indeed, the machine learning literature have evidenced that in diverse SLC problems, ensembles are likely to be more accurate than their individual member classifiers [47, 86, 92].

4.4.1.2 Exploring Candidate Chain Sequences

Methods in this category work by first running a preprocessing step, prior to model training, that aims at identifying pairs of unconditionally dependent labels. Further, this information is employed to determine a restricted set of candidate chain sequences that, basically, correspond to chains in which correlated labels are placed close to each other. Finally, one of these candidate sequences should be randomly chosen or, optionally, ensembles can be built by randomly selecting some of the candidates. There are four of such methods proposed in the literature: SCC, BCC, HBCC and CT.

The Sorted-Label Chain Classifiers (SCC) approach [63] employs association rule mining [1, 47, 92, 105] in order to identify pairs of dependent labels. This technique works in three steps. First, considering only the label attributes in the dataset, SCC mines all association rules of length two (i.e., rules with one label in the antecedent and one label in the consequent) with support and confidence above user-specified thresholds. From this set of rules, the SCC method generates a graph where each node corresponds to a specific label and each edge represents an association rule between a pair of labels. The next step consists in transforming this graph into a DAG, by employing a simple algorithm that prunes the "weakest" edges (edges representing rules with less significant values of support or confidence). From this final DAG, it is possible to obtain different full candidate chains using a topological sort algorithm [13]. Any of these chains can be chosen for training a CC model. Figure 4.3 illustrates the whole process.

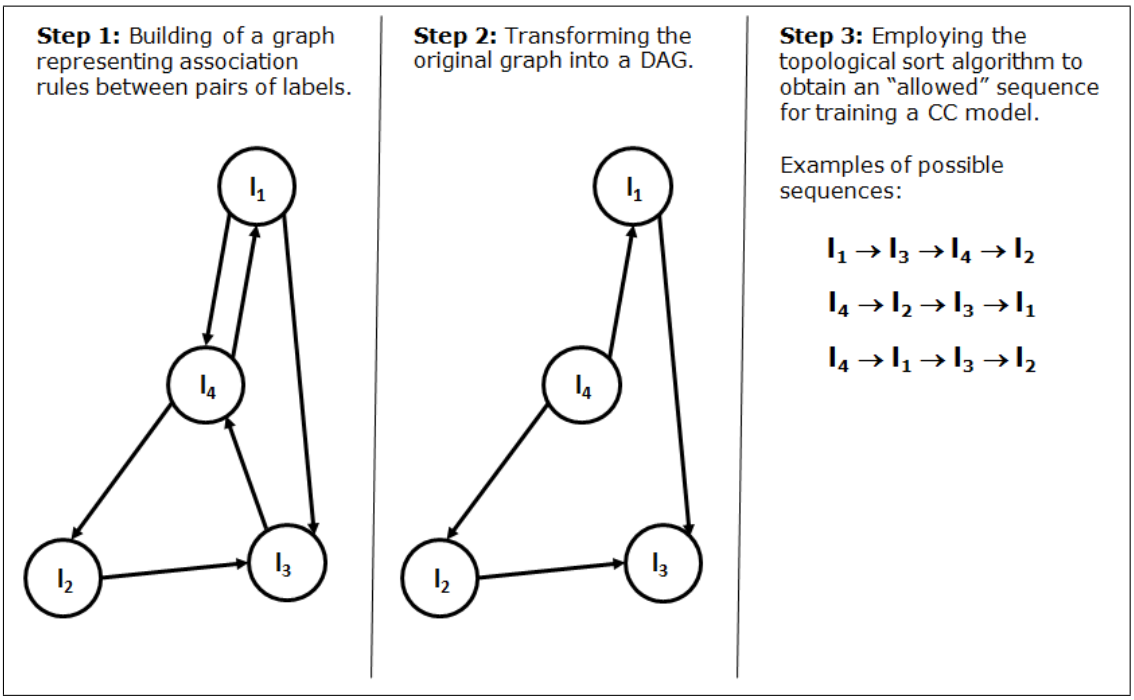


Figure 4.3: SCC method example

In [91, 110], the authors present the Bayesian Chain Classifier (BCC), a technique that relies on the use of Bayesian networks to identify and represent unconditionally dependent labels. In this approach, the first step is to induce a maximum weighted spanning tree [34] according to the mutual dependence measure between each pair of labels. Subsequently, one of the nodes is randomly selected as the root node and the tree is transformed into a directed tree by setting the direction of all edges outwards the root node. Each path in the directed tree will then form a different partial chain. The final CC model consists of the aggregation of all these partial chains. The procedure is illustrated in the example of

Figure 4.4, in which the final CC model comprises the chains defined by the two distinct paths in the tree: $\{l_1 \rightarrow l_4 \rightarrow l_2\}$ and $\{l_1 \rightarrow l_4 \rightarrow l_3\}$. In this example, the binary classifier for predicting l_1 will be trained using $\{X\}$ as its set of input attributes. The binary classifier for l_4 will be trained using $\{X \cup l_1\}$. Finally, the binary classifiers for l_2 and l_3 will both be trained using $\{X \cup l_1 \cup l_4\}$ as their set of input attributes.

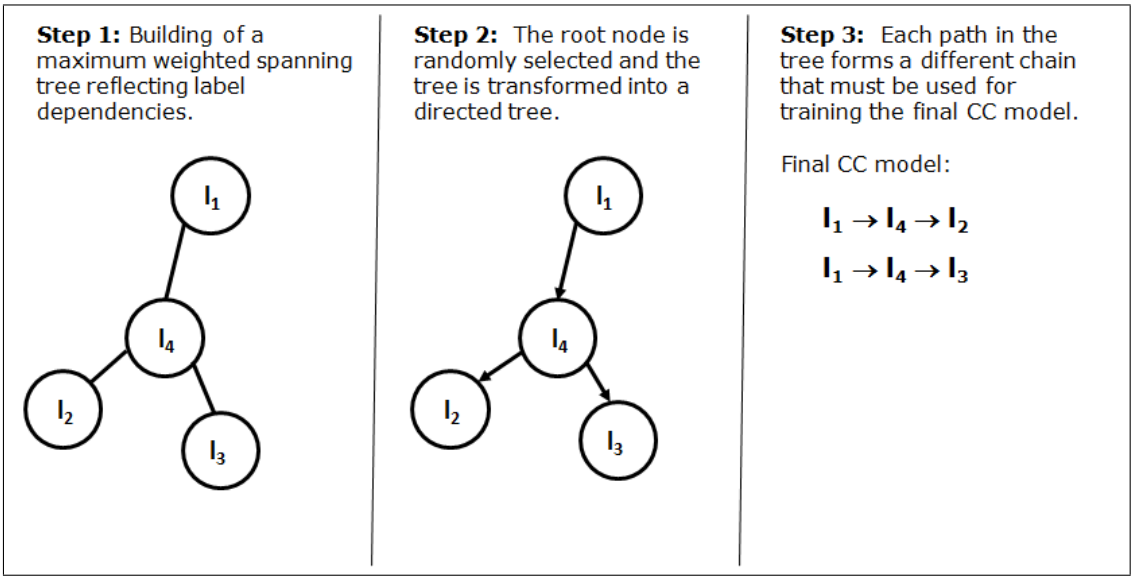


Figure 4.4: BCC method example

The Hybrid-Binary Chain Multi-Label Classifier (HBCC), proposed in [48], employs two steps to modify the CC's training phase. The first step consists in calculating the Pearson's linear correlation coefficient [47] between each pair of labels, generating a correlation matrix. According to the obtained results, different chains can be defined in the second step, each one composed of labels identified as strongly correlated (either negatively or positively). Suppose, for instance, that the pair of labels l_1 and l_4 have a strong correlation, but are not correlated with both l_2 and l_3. Consider also that l_2 and l_3 are correlated. In this situation, the HBCC algorithm would train two separate classifier chains, such as, for instance, $\{l_1 \rightarrow l_4\}$ and $\{l_2 \rightarrow l_3\}$ or $\{l_4 \rightarrow l_1\}$ and $\{l_3 \rightarrow l_2\}$. Note that although the HBCC method defines the set of labels that will form each of the different chains, these are placed in random order into their respective chains.

The CT (Classifier Trellis) method, recently proposed in [80], is even simpler than BCC and HBCC, also working in two steps. The first consists in computing the value of the mutual dependence between each pair of labels. However, instead of generating a maximum spanning tree considering these results (like BCC), in the second step the labels are placed into a fixed "trellis" structure such as the one presented in Figure 4.5. The top-left node of the trellis is randomly-chosen, being the remainder labels inserted

using a greedy (hill climbing) approach, which tries to maximize the mutual dependence between parents and children. In the prediction phase, the trellis structure is treated as a Bayesian network. To predict the relevance of each label l_j, the feature set is augmented with the binary information of the ancestors of l_j in the network.

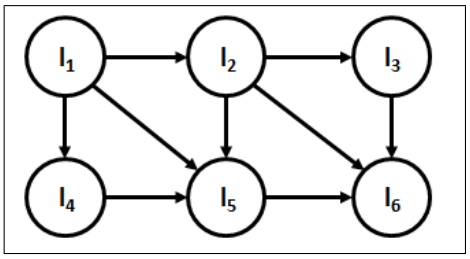


Figure 4.5: An example of CT structure with $q = 6$

The main advantage of the methods SCC, BCC, HBCC and CT is that they are simple and fast, consisting of the original CC model plus a preliminary step that is performed before the training step, in which the correlation between pairs of labels is measured. BCC, HBCC and CT have also the interesting characteristic of changing the chain structure to another kind of structure (a tree in BCC, a trellis in CT and a set of partial chains in HBCC). A potential advantage is that these structures allow uncorrelated labels to be put in different chains. Nevertheless, these methods have three significant drawbacks. The first and most important is that they are based on the measurement of the unconditional dependence among labels. Typically, this is not sufficient to truly represent dependencies. Second, they rely on the simplistic assumption that a good chain sequence is the one in which strongly correlated labels are connected or are placed close to each other, not paying attention to the way the sequence is ordered. Third, none of the methods is able to find out a single optimized chain. Instead, they produce a set of candidate structures, which can define different chain sequences. In order to train a CC model, a candidate must be randomly chosen or, alternatively, an ensemble of sequences defined by different candidate structures can be generated.

4.4.1.3 Searching for a Single Optimized Label Sequence

This family of CC variations are based on the use of heuristic search techniques that aim at finding a single optimized label sequence. I.e., they search for a unique and specific label sequence that leads to an improvement on the predictive accuracy of the CC model. There are two of such methods proposed in the literature: BS and M2CC.

The BS method, presented in [58], tackles the LSOP by performing a beam search over

a tree in which every distinct path represents a different label permutation. An example of such tree for $q = 3$ labels is illustrated in Figure 4.6. Since the construction of a tree with $q!$ paths is infeasible even for moderate sizes of q, the BS method employs an user adjustable input parameter called *beam width* (b) to reduce the number of paths. The tree is built in a level-wise fashion, starting from the root node. During the construction of each level, only the top-b vertices in terms of predictive accuracy must be maintained in the tree (which is computed using only the training set). Once the tree has been fully constructed, the final sequence is the one – among the b full chains represented by the b paths in the tree – that leads to the best value of a chosen performance measure in the training set.

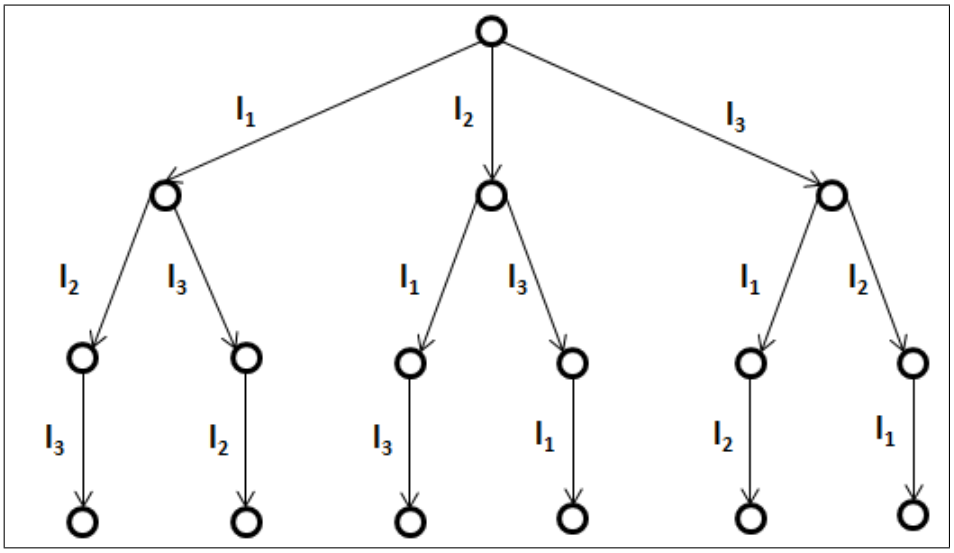


Figure 4.6: An example of BS tree with $q = 3$

The M2CC method, described in [78], employs a double-Monte Carlo optimization technique to efficiently generate and evaluate a small population of distinct label sequences. The best sequence is the one that maximizes some payoff function in the training set. The method works as follows. It starts with a randomly-defined sequence. During the execution of the algorithm, this sequence is modified with the aim of finding, at least, a local maximum of the payoff function. In [78], the authors adopted the Exact Match as the payoff function and a simple procedure in order to determine the candidate sequences. This procedure consists in choosing two positions of the label sequence and swapping the labels corresponding to them.

The methods BS and M2CC offer important advantages. First, they are capable of automatically taking into consideration the conditional dependencies during their search procedure. Second, unlike the ensemble approach and the approaches based on candidate

chain sequences, both BS and M2CC find a single optimized chain sequence reflecting the label dependencies. This is especially interesting for applications that require interpretable classifiers. The major drawback associated to these techniques is that their training step is more computationally expensive in comparison with the methods described in the previous subsections.

4.4.2 Approaches Based on Inference Optimization

This subsection examines the only two methods proposed in the literature that modify the inference step of the original CC method in order to improve the predictive performance of the classification model: probabilistic classifier chains [21] and the one-to-one classifier chains [16].

4.4.2.1 Probabilistic Classifier Chains

The first technique for improving CC by performing inference optimization instead of training optimization was introduced in [21], in the approach named as Probabilistic Classifier Chains (PCC). The basic idea of this method is to apply the chain rule of the probability theory at the inference step to obtain more accurate predictions.

The PCC's training step is identical to the CC's one: a label sequence is randomly chosen and used to train a CC model. However, its classification step works differently. According to the chain sequence used in the training step, the PCC classifier aims at maximizing the posterior probability of the predicted labelset for each test instance. In order to accomplish this task, it first generates a PCC tree composed of 2^q paths. An example of such tree corresponding to the label sequence $\{l_2 \rightarrow l_1 \rightarrow l_3\}$ is illustrated in Figure 4.7. To classify a new instance $t = (x, ?)$, the algorithm evaluates all possible paths in this tree, calculating the conditional probability for all possible outputs. For example, the probability estimation of the output $\Pr\big(l_2 = 0, l_1 = 1, l_3 = 0 \mid x\big)$ is obtained with the application of the chain rule:

$$\Pr\big(l_2 = 0 \mid x\big) \times \Pr\big(l_1 = 1 \mid x, l_2 = 0\big) \times \Pr\big(l_3 = 0 \mid x, l_2 = 0, l_1 = 1\big).$$

The returned labelset is the one with highest estimated conditional probability. It is important to state that, in theory, the chain rule should always return the same result regardless of the label sequence. However, since each $\Pr(y_j \mid x)$ corresponds to an estimated probability (instead of a real one), in practice, the results end up being different.

The PCC method is effective, simple, and based on a principled probabilistic approach.

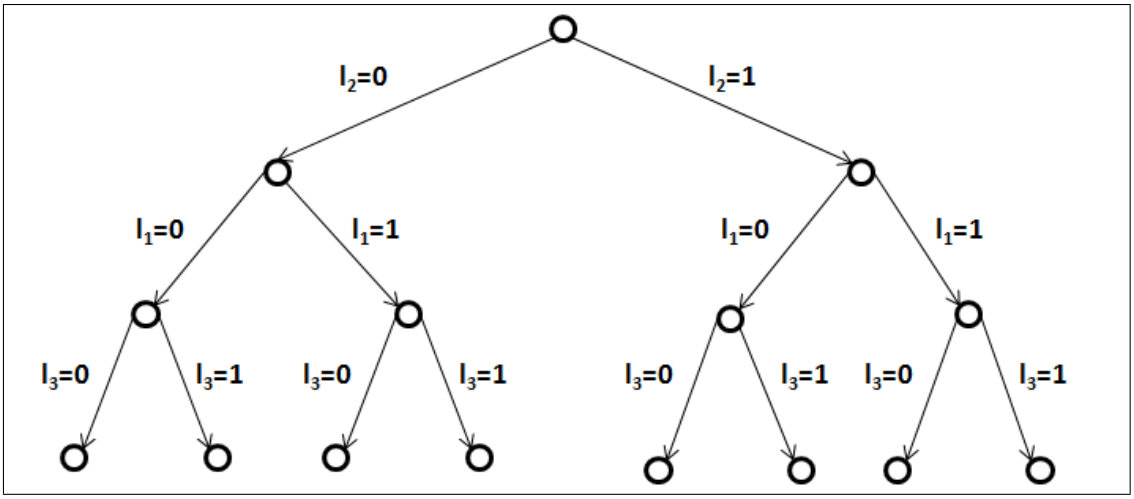


Figure 4.7: An example of PCC tree associated to the label sequence $\{l_2 \rightarrow l_1 \rightarrow l_3\}$

However, it has two drawbacks. First, it requires a probabilistic single-label base classifier – such as naïve Bayes or logistic regression – to estimate the posterior probabilities for each path in the PCC tree. Second, it employs an exhaustive search in the space of 2^q possible label combinations. Thus, its practical applications are restricted to problems where q is small. To cope with this problem [22, 58, 78] suggest the use of heuristic techniques to explore the PCC tree. The proposals of [22, 78] are equivalent, suggesting the use of a Monte Carlo search on the tree whereas [58] suggests performing a beam search over the PCC tree. It is also important to remark that the PCC method is susceptible to the "bad label ordering" problem, since a random label sequence is used in the training step. The authors of PCC simply recommend the use of an ensemble of PCC classifiers to attenuate this problem. However, any method for determining an optimized chain sequence (such as the ones presented in Subsection 4.4.1.3) can be used in conjunction with PCC.

4.4.2.2 One-To-One Classifier Chains

The One-To-One Classifier Chains method (OOCC), proposed in [15, 16], has emerged from a new modified execution of the exhaustive experiment described in Section 4.2. In this new experiment, we identified that the use of different optimized label sequences for distinct instances can lead to high gains in the predictive performance of CC.

Differently from PCC, the OOCC method modifies not only the inference step of CC, but also the training step and offers the advantage of not requiring the use of probabilistic single-label classifiers. The method works as follows. First, in the training step, one or more label sequences that perform well for each training instance are found (according to a given evaluation measure). In order to accomplish this task, the training dataset is

randomly partitioned into m distinct subsets, each representing a different data partition D_v. For each subset D_v, the method induces r CC models (each one using a distinct random label sequence), using a training set formed by the remainder $m-1$ data partitions (i.e., all data partitions except D_v). Once the models are built, it becomes possible to identify the best label sequences associated to each instance of the data partition D_v. Both parameters m and r are user-specified.

At the classification step, a k-NN (k-nearest neighbors) algorithm is employed to retrieve the k training instances that are most similar to the instance $t = (x, ?)$ being classified, and assign, to t, the label sequence that was found to perform best for the k training instances. Due to the similarity between the test instance t and its k nearest training instances, it is expected that an effective label sequence for instance t's nearest neighbors will also be an effective label sequence for instance t. Actually, experiments reported in [16], have shown that OOCC obtained, overall, better predictive performance than the BR, CC and ECC methods. However, a disadvantage of OOCC lies in the fact that, as a lazy learning approach, it is expensive at classification time.

4.5 Concluding Remarks

This chapter presented a review of the original CC method and their current extensions proposed in the literature, making two additional contributions in regard of improving the fundamental understanding of the CC model.

The first consisted in a study that, for the first time, investigated in depth the alleged key drawback associated to the original CC method: the fact that the label ordering is decided at random. An exhaustive experiment (presented in Section 4.2), confirmed that the order of the chain actually has a strong effect on the predictive performance. However, the different base (single-label) algorithms, by their own characteristics, may be more or less affected by the chain ordering. For instance, the effect tends to be very large when the base algorithm is C4.5, but it can be rather small for Naïve Bayes. The same experiment also revealed that a good chain sequence for a given base algorithm (e.g., C4.5) is not necessarily good for other base algorithm (e.g., SMO).

The second contribution consisted in the proposal of three novel baseline methods to determine an optimized label sequence for CC, named PredCC, FreqCC and DepCC. These baseline methods were compared against the original CC method (where the label ordering is decided at random) considering four distinct evaluation measures of predictive

performance. Nonetheless, the two-tailed Wilcoxon signed-rank test indicated that, with a confidence level of 95%, none of the methods is statistically superior to CC, evidencing that it is necessary to invest in more sophisticated solutions to overcome the LSOP.

Motivated by these empirical findings, in the next chapter we present one of the main contributions of this thesis: a genetic algorithm for optimizing the label ordering in multi-label classifier chains.

Chapter 5

The GACC Method

This chapter presents one of the main contributions of this thesis: the GACC method (Genetic Algorithm for Optimizing Classifier Chains) [42]. It represents the first strategy based on the evolutionary paradigm of GAs to overcome the LSOP. The text is structured as follows. Section 5.1 presents the main motivations for developing this method and then describe in detail the designed GA (indicating additional improvements to the original proposal published in [42]). The results of the experimental evaluation of GACC, CC and BR on the collection of benchmark datasets is presented in Section 5.2. Finally, we give concluding remarks in Section 5.3.

5.1 The GACC Method

In [42], we proposed the GACC method, a genetic algorithm for finding an optimized ordering for a chain of classifiers. More clearly, the goal of GACC is to search for a label ordering that leads to a significant improvement on the predictive accuracy of the CC model. Our main motivations for using GAs in the MLC context – especially under the classifier chains framework – are described below:

- GAs are a global search method capable of effectively exploring the extremely large search space of $q!$ possible solutions associated to the LSOP. As a global method, GAs tend to cope better with attribute interactions than greedy methods [30, 32]. Hence, intuitively, GAs are expected to discover correlations among labels that would be missed by greedy approaches.

- Over the last decades, GAs have been successfully applied to solve a myriad of optimization problems where a candidate solution is represented as a permutation,

like the TSP [59, 68, 72] and others [28, 36, 64, 73, 85, 89, 94]. Although TSP does not constitute a classification problem, it bears some resemblance to the LSOP (in essence, LSOP and TSP are permutation problems). Nonetheless, it is necessary to highlight that LSOP, as a classification problem, involves prediction and overfitting issues, unlike optimization problems (both issues are examined in this section).

- GAs have also been widely employed to solve a large number of classification problems in the most distinct contexts and application domains. For instance, the technique is widely employed to perform feature selection [32, 47, 60] (evolving the subset of features that leads to the best classification accuracy), to determine the best set of weights for training neural networks [4] and to discover classification rules [10, 30, 32].

- GAs are capable of delivering an interpretable result (a single optimized chain ordering that can be interpreted by users), which is important in many real-world classification problems [33], such as medical diagnosis, functional genomics, social research and direct marketing.

- As discussed in Section 2.3, the evaluation of MLC classifiers often involves the use of several distinct measures. GAs naturally allow the evaluation of a candidate solution by simultaneously considering different quality criteria in the fitness function [31].

- As the CC and BR approaches, GAs can be easily parallelizable for better time performance.

GACC is the first proposed strategy that makes use of Evolutionary Algorithms to address the LSOP, also constituting our first main contribution towards the improvement of CC classifiers. In the next subsection, a full description of the GACC method is presented. Comments on additional improvements to our original proposal published in [42] are made throughout the text.

5.1.1 GACC Description

5.1.1.1 Individual Representation and Population Initialization

A candidate solution for the LSOP must specify a label sequence. In GACC, we followed the path representation encoding, which allows individuals of the population to be simply represented by q-dimensional vectors regarding different specific label orderings for CC,

where q represents the number of labels. For instance, the sequence $\{l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4\}$ is encoded as the vector $[1, 2, 3, 4]$. Note that, differently from TSP, pairs of permutations where one permutation is the inverse of the other (such as $[1, 2, 3, 4]$ and $[4, 3, 2, 1]$) do not represent equivalent solutions (i.e., the accuracy of a CC model built using the chain $\{l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4\}$ is probably different from the one obtained with the chain $\{l_4 \rightarrow l_3 \rightarrow l_2 \rightarrow l_1\}$).

The initial population is created randomly. In the adopted approach, each individual (candidate chain sequence) is created by randomly selecting integer numbers (representing the label indexes) according to a uniform distribution in the range $[1, q]$. Repeated integers are not allowed in a chromosome.

5.1.1.2 Fitness Computation

To assess the predictive accuracy of a chromosome, we use the Quality (fitness) function defined in Equation 5.1. This function simultaneously takes into account the measures of Exact Match (EM), Accuracy (ACC) and Hamming Loss (HL), respectively defined in Equations 2.2, 2.3 and 2.5. The Quality of CC_i – a CC model built using the label sequence encoded in a chromosome i – is computed as[1]:

$$Quality(CC_i) = \frac{(1 - HL) + ACC + EM}{3} \tag{5.1}$$

The evaluation method follows the wrapper approach [30] in which the quality of an individual (candidate CC model) is determined by using the target MLC method (i.e., the CC method). The fitness function is calculated using only the training set, according to a holdout method that works as follows. First, the training set is partitioned into two mutually-exclusive subsets: building (2/3 split) and validation (1/3 split). Next, for each chromosome we build a CC model using only the building set, and then that model is evaluated with the validation set (i.e., the model's fitness is computed using Equation 5.1).

5.1.1.3 Parent Selection

At each generation, parent selection is performed using the tournament procedure described in Subsection 3.2.4. First, the GA randomly chooses k individuals from the population, where k is a user-specified parameter called tournament size. These individuals

[1] It is worth mentioning that it is possible to change the fitness function so as to optimize any specific performance measure.

"play a tournament" which consists of a comparison of their fitness values. The winner is the individual with the best fitness among the k participants.

5.1.1.4 Genetic Operators

Each pair of individuals selected by tournament undergo crossover operation so as to create the offspring. In the preliminary experiments reported in [42], GACC employed the Donor-Receptor crossover method (DRC) (Figure 3.4). Nonetheless, in this thesis we decided to substitute DRC for the Order Crossover method (OC) (Figure 3.5), since the latter has demonstrated to be more effective. This can be explained in terms of the OC capacity for promoting a mutual exchange of genetic material (sub-chains) from the two selected parents in the produced children.

The implemented mutation operation is same used in [42], consisting in selecting a group of children from the offspring set (new individuals produced by the crossover operation) and swapping two class labels (labels in two different positions) at random in each selected child. We evaluated the use of other kinds of mutation operators, but these did not lead to an overall improvement on the accuracy of the GACC method.

5.1.1.5 Population Replacement

GACC adopts age-based replacement with elitism. As introduced in Subsection 3.2.5, in this scheme the entire population is replaced on each generation, but a percentage of elite individuals is preserved from the previous generation according to their fitness values.

5.1.2 The Overfitting Issue

Since GAs perform a large number of candidate solution evaluations (given by the number of individuals in the population times the number of generations), it is possible it will be prone to the problem of overfitting the training data – i.e., finding a classification model that achieves high accuracy on training data, but does not generalize well for unseen instances [3, 44]. In fact, in the preliminary experiments with GACC reported in [42], we observed the occurrence of overfitting when a moderate to large number of generations was set for the execution of the GA (e.g., 50 or 100 generations). To alleviate this problem and also to reduce the computational time required for the experiments, we followed the general suggestion of [25], which advises reducing the number of models to be considered. Hence, we limited the number of generations to a value between 15 and 20 (according to

the size of the target dataset) in the experiments of [42], obtaining an attenuation on the effects of overfitting.

Nonetheless, the above values are rather small when compared to the number of generations used in typical GA applications, which may be detrimental to the overall effectiveness of GACC. In order to cope with this issue, in this thesis we modified the method to mitigate overfitting, adopting the technique indicated in [32]. The approach consists in changing the building and validation sets at each generation of GACC by re-splitting the training set. More clearly: at each generation, a different subset of training instances is randomly selected to form the building and validation sets.

5.1.3 GACC Pseudocode

Algorithm 4 outlines the GACC method in pseudocode. It can be seen as an expanded version of the generic GA pseudocode (presented in Algorithm 1, Chapter 3) tailored for solving the specific LSOP. The GACC algorithm receives a training set D as input[2] and produces as output a single optimized label sequence *bestChain*, representing the fittest individual found after the evaluation of the GACC's last generation.

First, an initial population of candidate label sequences is randomly generated (line 1). This initial population will evolve over successive generations in the REPEAT loop that encompasses lines 2 to 12, until a maximum number of generations has been performed. This loop is divided into two phases: fitness evaluation of the current population (lines 3 to 7) and generation of a new population (lines 8 to 11).

The first phase of the GA cycle (lines 3 to 7) works as follows. Initially, in line 3, we construct the build and validation sets from the training set D. As stated in the previous subsection, in order to mitigate the effects of overfitting, varying subsets of the training data are used to form the build and validation sets at each generation of the GA. Next (lines 4 to 7), the fitness of the current population is computed following the wrapper approach. In this process, distinct CC models are built and evaluated, one for each chromosome (label sequence). It is worth reinforcing the fact that only the training set is used during the evaluation process. The CC models are inferred using the build set (line 5) and have their fitness calculated with the validation set (line 6).

[2]In this thesis, the genetic algorithm parameters (population size, number of generations, tournament size, number of elite individuals and probability of using genetic operators) were optimized by performing calibrating tests in a separate group of datasets (datasets other than the ones used in the experiments to evaluate the predictive accuracy of GACC). This is further addressed in Subsection 5.2.1.

The goal of the second phase (lines 8 to 11) is to expand the GACC's search space, by producing a new population of candidate solutions to be evaluated in the next cycle (generation). First, in line 8, a tournament selection procedure is employed to perform parent selection. Next, the selected parents undergo crossover (according to the OC approach), resulting in the generation of the offspring set (line 9). Members in the offspring may also be subjected to mutation (line 10). Finally, the old population is replaced by the offspring, with the use of an elitist strategy to preserve a small group of elite individuals (line 11).

After the individuals of the last generation have been evaluated, GACC returns to the user the fittest individual (the label sequence with best value for Quality metric) from the population in this generation (line 13).

Algorithm 4 GACC overall pseudocode

Input : D (training set)
Output: *bestChain* (a single optimized label sequence)

1: create an INITIAL POPULATION of individuals (candidate label sequences)
2: **repeat**
3: divide the training set D into $BuildSet$ and $ValidationSet$
4: **for all** candidate label sequences i in the current population **do**
5: build the CC_i model using $BuildSet$ and the label sequence i
6: $Fitness_i \leftarrow$ calculate Quality(CC_i) using $ValidationSet$
7: **end for**
8: SELECT parents using tournament selection
9: apply CROSSOVER to selected individuals, generating offspring
10: apply MUTATION to the offspring
11: UPDATE the current population using age-base replacement with elitism
12: **until** (maximum number of generations is reached)
13: **return** *bestChain* {individual with best Quality value in the last generation}

5.2 Experiments

In this section, we present the results of the experiments that compared our proposed genetic algorithm against the BR and CC methods.

5.2.1 Experimental Setup

We implemented GACC in Java and made use of the CC and BR implementations available at the Mulan platform. The empirical evaluation was performed using the same

datasets and following the same methodology described in Section 4.3, which can be summarized as follows: (i) the Weka's J48 implementation with default parameters was used as the base SLC algorithm for the three methods; (ii) a holdout evaluation was performed to assess the predictive performance of the multi-label methods by using the training and test parts that come with the benchmark datasets; (iii) the predictive performance was evaluated in terms of Accuracy, F-Measure, Exact Match and Hamming Loss; (iv) the Wilcoxon signed-rank test was employed to verify the statistical significance of the results with a confidence level of 95%.

As GACC is a probabilistic method, we averaged the results of ten executions with distinct random seeds (except for the larger datasets "enron" and "llog" where the results were averaged over five executions). The results reported for CC are also averaged over ten executions with distinct random seeds. This differs from [42] – where preliminary results regarding the GACC method were presented – in which the reported results considered a single execution of both GACC and CC.

GAs require the use of a set of parameters, namely: number of generations (g_{max}), population size (μ)), crossover rate (p_c), mutation rate (p_m), tournament size (k), and number of elite individuals (e). In practice, parameter tuning plays an important role in the effectiveness of any GA [28]. In this thesis, the values of the genetic algorithm parameters were set by performing calibrating tests with three benchmark datasets: "flags" ($q = 7, d = 19, N = 194$), "emotions" ($q = 6, d = 72, N = 593$), and "scene" ($q = 6, d = 294, N = 2407$). In total, 15 parameter setting combinations were considered (Table 5.1).The final settings are shown in Table 5.2. It is worth remarking that, intuitively, the performance of GACC could be improved by separately performing parameter optimization for each of the ten datasets involved in our experimental evaluation. However, we decided to optimize the GA parameters across three selected small datasets due to two advantages pointed out in [53]. The first is simply because using separate datasets for parameter optimization is less time consuming. The second and more important advantage lies in the fact that our adopted approach allows for the identification of GA parameters that are robust across different datasets. In this sense, the parameters presented in Table 5.2 can be understood as the recommended set of parameters (or default parameters) for GACC, i.e., the ones used when users do not have time to perform parameter tuning.

Table 5.1: Parameter setting combinations evaluated in the calibrating tests

g_{max}	μ	p_c	p_m	k	e
20	35	100%	25%	5	2
20	35	100%	15%	3	2
20	35	100%	25%	2	2
50	35	100%	10%	2	2
50	35	100%	15%	2	2
50	50	100%	25%	2	2
50	50	100%	25%	2	3
50	50	100%	15%	2	3
50	100	100%	25%	2	2
50	100	100%	25%	3	2
50	100	100%	25%	5	2
50	200	100%	25%	2	2
50	200	100%	15%	2	2
50	200	100%	25%	2	4
50	200	90%	25%	2	2

Table 5.2: GACC final recommended set of parameters (used in the experimental evaluation)

Number of Generations (g_{max})	50
Population Size (μ)	200
Crossover Rate (p_c)	100%
Mutation Rate (p_m)	25%
Tournament Size (k)	2
Number of elite individuals (e)	2

5.2.2 Results

The results for the measures of Accuracy, F-Measure, Exact Match, and Hamming Loss are respectively shown in Tables 5.3, 5.4, 5.5 and 5.6. The best results for each dataset are highlighted in bold type. The rank obtained by each method in each dataset is presented in parenthesis whilst the average rank for each method is show in the last row. In the rows below Tables 5.3 and 5.4 and 5.5, the symbol $\succ$ represents a statistically significant difference between one or more methods. For instance, $\{a\} \succ \{b, c\}$ shows that the method a is significantly better than b and c.

According to Tables 5.3 and 5.4, the best values for Accuracy and F-Measure were achieved by GACC in the majority of the datasets. The two-tailed Wilcoxon test indicated that, with confidence level of 95%, GACC is statistically superior to both BR and CC for

Table 5.3: Performance of BR, CC and GACC in terms of Accuracy.

Dataset	Accuracy		
	BR	CC	GACC
university	0.300 (3.0)	0.310 (2.0)	**0.319** (1.0)
yeast	0.423 (2.0)	0.418 (3.0)	**0.432** (1.0)
ces-16	0.196 (2.0)	0.194 (3.0)	**0.213** (1.0)
birds	**0.573** (1.0)	0.564 (3.0)	0.569 (2.0)
thyroid	**0.984** (1.0)	0.983 (2.5)	0.983 (2.5)
genbase	**0.987** (2.0)	**0.987** (2.0)	**0.987** (2.0)
medical	0.743 (2.5)	0.743 (2.5)	**0.744** (1.0)
enron	0.367 (3.0)	0.402 (2.0)	**0.409** (1.0)
llog	0.243 (2.0)	0.239 (3.0)	**0.249** (1.0)
cal500	0.212 (3.0)	0.218 (2.0)	**0.224** (1.0)
average rank	2.15	2.50	**1.35**
GACC $\succ$ {BR, CC}			

Table 5.4: Performance of BR, CC and GACC in terms of F-Measure.

Dataset	F-Measure		
	BR	CC	GACC
university	0.315 (3.0)	0.335 (2.0)	**0.343** (1.0)
yeast	0.547 (2.0)	0.523 (3.0)	**0.548** (1.0)
ces-16	0.251 (2.0)	0.249 (3.0)	**0.275** (1.0)
birds	**0.603** (1.0)	0.592 (3.0)	0.597 (2.0)
thyroid	**0.990** (2.0)	**0.990** (2.0)	**0.990** (2.0)
genbase	**0.991** (2.0)	**0.991** (2.0)	**0.991** (2.0)
medical	**0.773** (1.0)	0.769 (2.5)	0.769 (2.5)
enron	0.474 (3.0)	0.506 (2.0)	**0.507** (1.0)
llog	0.261 (2.0)	0.255 (3.0)	**0.264** (1.0)
cal500	0.344 (3.0)	0.348 (2.0)	**0.356** (1.0)
average rank	2.10	2.45	**1.45**
{GACC} $\succ$ {CC}			

Accuracy ($T_{wilcox}(9) = 4.5$ and $T_{wilcox}(8) = 0.0$, respectively). GACC is also statistically superior to CC for F-Measure ($T_{wilcox}(7) = 0.0$). There are no statistically significant difference between the BR and CC methods in both measures.

The GACC method also outperformed the other two methods in terms of Exact Match in the vast majority of the datasets, as presented in Table 5.5. Once again, the difference between the results is statistically significant in favor of GACC ($T_{wilcox}(8) = 3.0$ and $T_{wilcox}(7) = 0.0$ in regard to BR and CC models, respectively). On the other hand, although CC performed, on average, better than BR, the difference between the results of these methods is not statistically significant.

The results presented in Table 5.6 show that the BR model obtained the best results

Table 5.5: Performance of BR, CC and GACC in terms of Exact Match.

Dataset	Exact Match		
	BR	CC	GACC
university	**0.263** (1.0)	0.248 (3.0)	0.255 (2.0)
yeast	0.064 (3.0)	0.127 (2.0)	**0.134** (1.0)
ces-16	0.060 (2.0)	0.058 (3.0)	**0.063** (1.0)
birds	0.486 (2.0)	0.485 (3.0)	**0.492** (1.0)
thyroid	0.932 (3.0)	**0.941** (1.5)	**0.941** (1.5)
genbase	**0.975** (2.0)	**0.975** (2.0)	**0.975** (2.0)
medical	0.651 (3.0)	0.665 (2.0)	**0.670** (1.0)
enron	0.086 (3.0)	0.125 (2.0)	**0.135** (1.0)
llog	0.199 (3.0)	0.201 (2.0)	**0.209** (1.0)
cal500	**0.000** (2.0)	**0.000** (2.0)	**0.000** (2.0)
average rank	2.40	2.15	**1.35**
$\{GACC\} \succ \{BR, CC\}$			

Table 5.6: Performance of BR, CC and GACC in terms of Hamming Loss.

Dataset	Hamming Loss		
	BR	CC	GACC
university	**0.134** (1.0)	0.139 (3.0)	0.138 (2.0)
yeast	**0.259** (1.0)	0.274 (3.0)	0.266 (2.0)
ces-16	0.188 (2.5)	0.188 (2.5)	**0.187** (1.0)
birds	**0.051** (2.0)	**0.051** (2.0)	**0.051** (2.0)
thyroid	**0.005** (1.0)	0.006 (2.5)	0.006 (2.5)
genbase	**0.001** (2.0)	**0.001** (2.0)	**0.001** (2.0)
medical	**0.011** (2.0)	**0.011** (2.0)	**0.011** (2.0)
enron	**0.054** (1.5)	**0.054** (1.5)	0.055 (3.0)
llog	**0.019** (2.0)	**0.019** (2.0)	**0.019** (2.0)
cal500	**0.163** (1.0)	0.176 (3.0)	0.173 (2.0)
average rank	**1.60**	2.45	2.05
No statistical significance			

in terms of Hamming Loss in the majority of the datasets. This was expected, since BR is actually suitable for most loss functions that ignore label correlations, as demonstrated in [21]. However, the two-tailed Wilcoxon test suggested that no statistically significant differences exist between the Hamming Loss values achieved by BR and GACC and also between the values achieved by BR and CC. It is also noticeable that GACC obtained a better average rank than CC; however, no statistically significant differences were identified.

In summary, the results indicated that, in comparison with the original CC method, our proposed genetic algorithm is significantly superior in terms of Accuracy, F-Measure and Exact Match. In this sense, the behavior of GACC differs from the behavior of the

baseline methods PredCC, FreqCC and DepCC, which, overall, yield predictive accuracies statistically equivalent to the ones obtained by CC in experiments involving the same set of benchmark datasets and evaluation measures (Section 4.3).

With regard to the BR method, GACC obtained a significant gain on Accuracy and Exact Match, without significantly impacting the Hamming Loss measure. Furthermore, GACC obtained higher F-Measure values in most datasets. On the other hand, the original CC method performed poorer than the BR method, obtaining average ranks inferior to the ones obtained by BR method in three out of the four measures of predictive performance (though the two-tailed Wilcoxon test suggests that no statistically significant differences exist among the results achieved by these methods). Actually, our experimental results regarding CC and BR are analogous to the ones reported in [16, 66], reinforcing that the use of a single randomly-generated label sequence actually represents a fragile approach for multi-label chain classifiers, thus evidencing the importance of searching for an optimized label sequence.

5.3 Concluding Remarks

In this chapter, we introduced GACC, a novel global search method for optimizing the label ordering in CC that makes use of a genetic algorithm. In this strategy, each chromosome represents a different label sequence and the fitness function combines three evaluation measures: Exact Match, Accuracy, and Hamming Loss. The crossover operation works by transferring sub-chains of random length between pairs of individuals whilst mutation swaps pairs of labels of an individual. The proposed GA follows the wrapper approach, evaluating the quality of an individual (candidate label sequence) by using the target MLC algorithm (i.e., the CC algorithm). GACC is the first strategy that makes use of Evolutionary Algorithms to address the label ordering problem.

The GACC method was compared against the CC and BR methods. The obtained results show that GACC significantly outperformed both methods in terms of Accuracy and Exact Match. In terms of F-Measure, GACC significantly outperformed CC. There was no significant difference among the three methods in terms of the Hamming Loss measure. These results suggest that, as a global search method, GACC is more suitable to identify and model label dependencies in comparison with greedy and baseline methods, such as PredCC, FreqCC and DepCC (as reported in Subsection 4.3.2, overall, there is no statistically significant difference between the results of CC and the three baseline

methods). Moreover, the GACC approach offers other important advantage: it is suitable for applications that require the generation of comprehensible classifiers, since, at the end of the process, GACC returns a single optimized chain, reflecting the label dependencies (unlike other methods for improving CC, such as the approach based on the use of an ensemble of random chains, suggested by the authors of the CC in [82, 83]).

The main disadvantage of GACC is the fact that its training phase is computationally slow by comparison with BR and CC. Considering a dataset composed of d attributes and N instances, the BR's training complexity is equal to $O(q \times f(d, N))$, where $f(d, N)$ corresponds to the training complexity of the base single-label algorithm. For the same dataset and base SLC algorithm, the complexity of CC is equal to $O(q \times f(d+(q/2), N))$, i.e., a small penalty is incurred for having an average of $q/2$ attributes added to each instance. On the other hand, GACC's complexity can be roughly given by: $O(q \times g_{max} \times \mu \times f(d+(q/2), N))$, where g_{max} represents the number of generations and μ the size of the population. Nonetheless, as argued by [32], the importance of this drawback (GACC's slow training time) depends on the user requirements, such as the size of the target database and if the classification is performed as an off-line task. Anyway, it is possible to attenuate this problem by developing a parallel version of GACC to simultaneously process multiple candidate individuals.

In [42], we identified that a promising direction of future work would be to extend GACC by defining a chromosome structure capable of representing not only the chain ordering, but also the presence or the absence of any specific label, allowing the GA search to investigate the predictive performance of shorter classifier chains (chains with length inferior to q). Motivated by this issue, in the next chapter we propose the GA-PartCC method, the first multi-label classifier chain method that is able to search for a single optimized label sequence, while at the same time taking into consideration the utilization of partial chains.

Chapter 6

The GA-PartCC Method

This chapter presents the second main contribution of this thesis: the GA-PartCC method (Genetic Algorithm for Optimizing Partially-Chained Models) [43], a novel genetic algorithm that performs a global search for an optimized chain (i.e., a label sequence that leads to an improvement on the predictive accuracy of the CC model), by exploring partial chains with only a subset of labels. To the best of our knowledge, this is the first approach proposed with the explicit goal of finding an optimized partially chained model for MLC.

The text is organized as follows. Section 6.1 presents the primary motivations behind our proposal and formally defines the concept of partially chained (PartCC) model for MLC. Section 6.2 presents a new exhaustive experiment that investigated the influence of the label sequence length in the predictive performance of CC. In Section 6.3, we describe our proposed GA-PartCC method. Section 6.4 presents a comprehensive set of experiments comparing GA-PartCC against other well-known alternative multi-label classifier chain methods. Finally, we summarize our conclusions in Section 6.5.

6.1 The Partially Chained Multi-Label Model

GACC and most of the other variations of the basic CC method aim at improving the model's effectiveness by handling the LSOP. However, there is another important drawback in the original CC method that has been often neglected in the literature. It corresponds to the fact that CC forces all labels to be present in the chain. None of the extensions have yet explored the idea of generating models defined by optimized partial chains. In this context, the aim is to build a CC model in which one or more labels may be absent from the chain because their presence would lead to a decrease in the predictive

performance. This is because some classifiers may pass redundant and irrelevant information, or wrongly predicted labels, along the chain, which might confuse the subsequent classifiers in the chain. Therefore, it might be interesting to remove these irrelevant or redundant labels from the chain structure (using independent binary classifiers for predicting each of them) and to create a partial chain with an optimized sequence using only the remaining labels. For instance, an example of partially chained model for a MLC that could be induced from the music categorization database presented in Figure 2.1 (Chapter 2) is given by:

$$[\,\{y_{Bossa} \rightarrow y_{Jazz} \rightarrow y_{Pop}\},\, \{y_{Metal}\}\,]$$

In the above example, the binary classifiers associated to labels "Bossa", "Jazz" and "Pop" are linked together in a classifier chain model in the sequence $\{y_{Bossa} \rightarrow y_{Jazz} \rightarrow y_{Pop}\}$. The label "Metal", however, is not in the chain. The intuition behind this representation is to indicate that "Metal" will be treated by an independent binary classifier, because its presence in the chain would decrease the predictive performance of the multi-label chain classifier as a whole.

A formal definition of partially chained model is given below:

Definition 6.1 *(Partially Chained Model). Let $X = \{X_1, ..., X_d\}$ be a set of d predictive attributes and $L = \{l_1, ..., l_q\}$ be a set of q class labels, where $q \geq 2$. Consider a training dataset D composed of N instances of the form $\{(x_1, Y_1), (x_2, Y_2), ..., (x_N, Y_N)\}$. In this dataset, each x_i corresponds to a vector $(x_1, ..., x_d)$ that stores values for the d predictive attributes in X and each $Y_i \subseteq L$ corresponds to a subset of labels. A partially chained model is a multi-label classification model inferred from D with the following structure:*

$$[\,\{y_{\alpha_1} \rightarrow y_{\alpha_2} \rightarrow \ ... \ \rightarrow y_{\alpha_r}\},\, \{y_{\beta_1},\, y_{\beta_2},\, ...,\, y_{\beta_s}\}\,]$$

The model is divided into two components. The left component, denoted by $\{y_{\alpha_1} \rightarrow y_{\alpha_2} \rightarrow ... \rightarrow y_{\alpha_r}\}$, consists in a single classifier chain model composed of r members, where $0 \leq r \leq q$ (r is referred to as the length of the model). The right component, $\{y_{\beta_1},\, y_{\beta_2},\, ...,\, y_{\beta_s}\}$, consists of a group of $s = q - r$ independent binary classifiers.

The PartCC model can be seen as a hybrid strategy between a BR model (when $r = 0$) and a CC model (when $r = q$) using an optimized label sequence. Thus, it is expected to be effective in both situations: when most labels are independent and also in the converse

case, when the majority the majority of the labels are dependent on each other. More interestingly, since it deals with partial chains, it is capable of producing simpler, more compact multi-label chain classifiers, offering gains in terms of efficiency. Furthermore, a PartCC model is particularly suitable for problems that require the construction of comprehensible predictive models [33], as the shorter sequence is more realistic for the representation of true label dependencies in comparison with a full length sequence.

Nonetheless, the problem of finding an optimized PartCC model is much more challenging than the traditional LSOP as the search space is composed of an ordinary BR model (where all labels are "disconnected") plus the models formed by all possible chain permutations of length 2 to q. Consequently, while the search space in the LSOP has size equal to $q!$, the size of the search space in the problem of optimizing PartCC models is much higher, being given by Equation 6.1.

$$1 + \sum_{r=2}^{q} \frac{q!}{(q-r)!} \tag{6.1}$$

Additionally, it is important to highlight the fact that the LSOP involves the optimization of a single objective (model accuracy) whereas the PartCC optimization problem takes into account two distinct objectives (model accuracy and model size). This characteristic also contributes to make the later problem more difficult than the first.

6.2 Exhaustive Experiment

In this section, we report the results of a new exhaustive experiment similar to the one described in Section 4.2. However, the new experiment was carried out to investigate the influence of the label sequence length in the CC's effectiveness. The experiment consisted in assessing the predictive accuracy of all possible PartCC models that can be built considering every label permutations of *any length* of three datasets: "flags" ($q = 7, d = 19, N = 194$), "emotions" ($q = 6, d = 72, N = 593$), and "scene" ($q = 6, d = 294, N = 2407$). As with the first exhaustive experiment reported in Chapter 4, the PartCC models were evaluated by applying the holdout method using the training and test parts supplied with the datasets. Table 6.1 presents the number of evaluated models per chain length (r) considering the three datasets involved in the experiment.

Figures 6.1, 6.2 and 6.3 present the results for the datasets "flags", "emotions" and

Table 6.1: Total number of evaluated models per chain length (r) for the datasets EMOTIONS, SCENE and FLAGS

	$r = 0$	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = 7$	**Total**
emotions, scene ($q = 6$)	1	30	120	360	720	720	-	1951
flags ($q = 7$)	1	42	210	840	2520	5040	5040	13693

"scene", respectively. In the three figures, each point in the horizontal axis represents a distinct PartCC model whilst the vertical axis shows the Quality value (Equation 5.1) obtained after the holdout evaluation of each model. The models are ordered along the x-axis first by their length (from 0 to q) and then, within each length group, in lexicographic order according to the chain sequence used to train the model. Vertical dotted lines are used to separate the groups of models with distinct lengths. In each figure, we show the results obtained using two single-label classifiers: C4.5 and and Naïve Bayes (NB).

The results of the experiment show that, similar to the first exhaustive experiment, the effect of the chain ordering and chain length is larger for C4.5 in comparison with Naïve Bayes. Observe that in the "emotions" dataset, where $q = 6$, the best sequence for C4.5 has size 5 (i.e., it is composed of $q - 1$ labels). For the other two datasets, is has been observed that the mean Quality value increases with the chain length. In regard to the Naïve Bayes algorithm, it is interesting to observe that the best sequence has size 0 in the "scene" dataset (i.e., BR performed better than any CC model). In the same dataset, some of the sequences with size 2 performed better than the sequences with larger values of r.

In summary, the results of this new exhaustive experiment reveal that in some situations, models composed of shorter chains can achieve higher accuracy than models composed of full chains. It is important to remark that we also investigated the base algorithms SMO and k-NN, observing similar behaviors (however, the results were omitted from Figures 6.1, 6.2 and 6.3 to simplify the visualization).

6.3 The GA-PartCC Method

In this section, a genetic algorithm for learning optimized PartCC models is proposed: GA for Optimizing Partially-Chained Models (GA-PartCC). To the best of our knowledge, this is the first classifier chain method proposed with the explicit goal of finding an optimized partially chained model for MLC. The GA-PartCC method can be seen as a natural

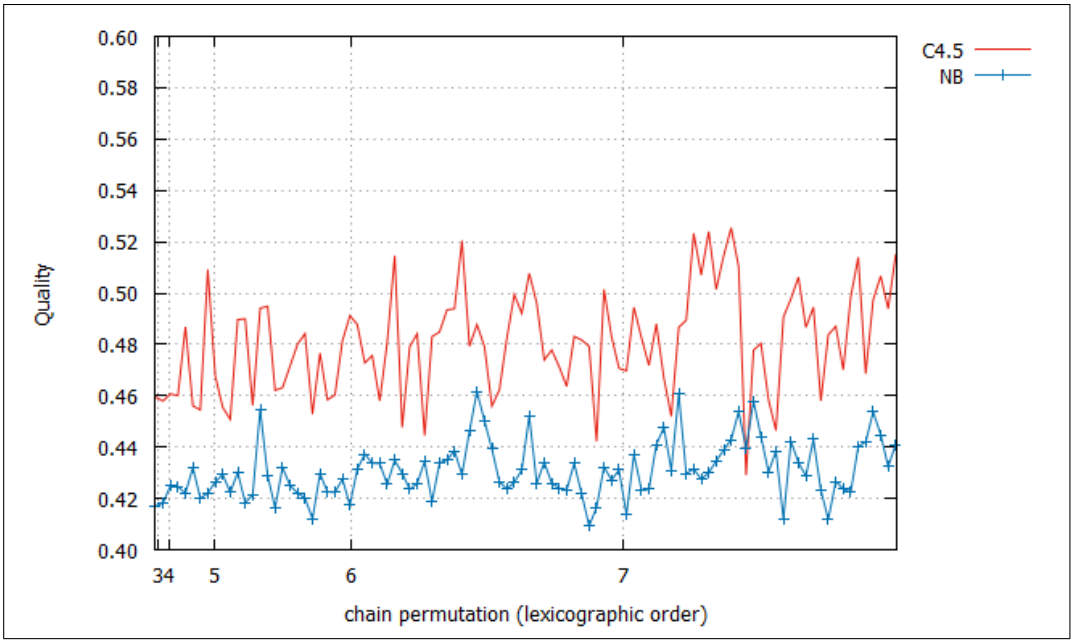


Figure 6.1: Results of the exhaustive experiment considering all possible label permutations of any length in the chain for the base classifiers C4.5 and NB according to the Quality measure: FLAGS dataset

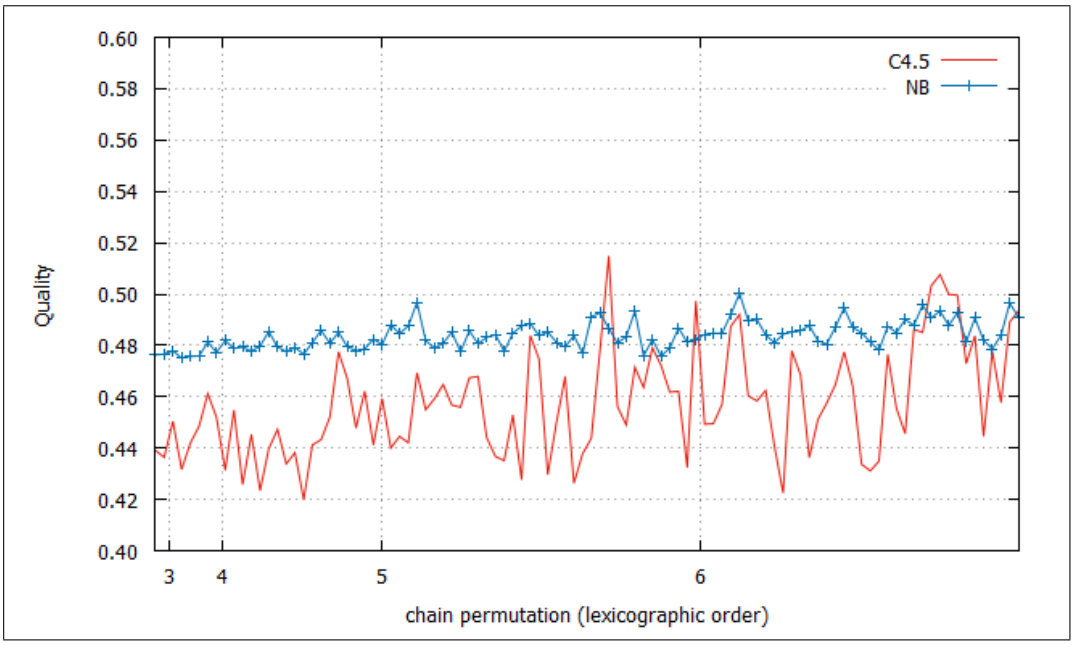


Figure 6.2: Results of the exhaustive experiment considering all possible label permutations of any length in the chain for the base classifiers C4.5 and NB according to the Quality measure: EMOTIONS dataset

extension to the GACC method. Actually, it was necessary to modify four constituent parts of GACC: the individual representation, the initial population, the fitness function and the genetic operators. These modifications are fully described below.

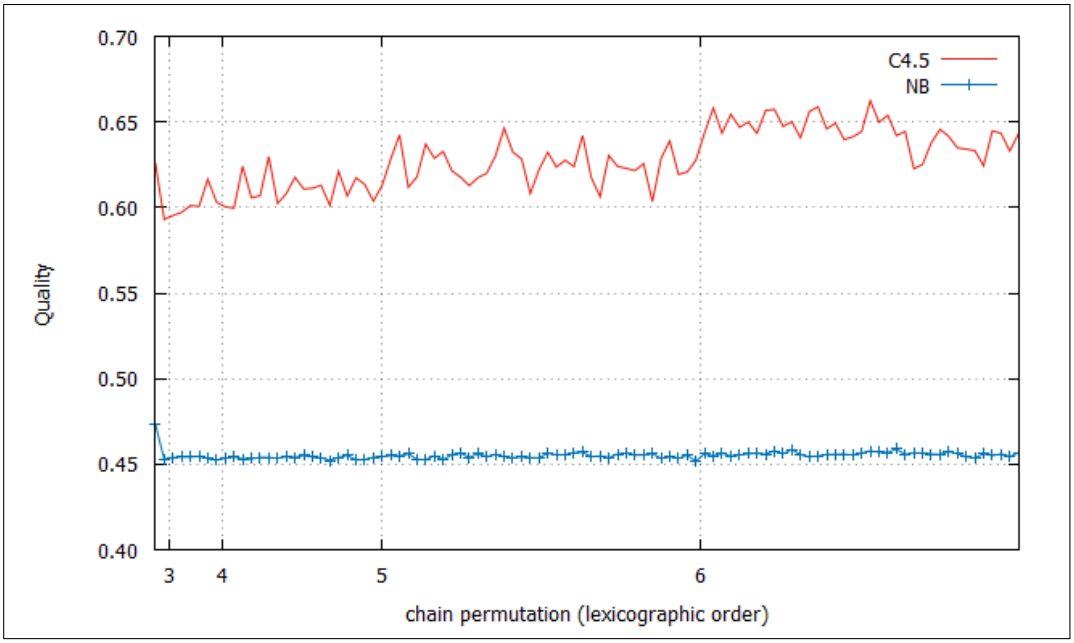


Figure 6.3: Results of the exhaustive experiment considering all possible label permutations of any length in the chain for the base classifiers C4.5 and NB according to the Quality measure: SCENE dataset

6.3.1 Individual Representation and Population Initialization

In GA-PartCC, individuals are represented by variable-length lists. Each candidate solution specifies both a subset of labels and the order they are placed into the chain. For instance, considering a problem where $q = 4$, the partial chain $l_3 \rightarrow l_1 \rightarrow l_4$ (in which l_2 is absent from the chain) is encoded as the list $[l_3, l_1, l_4]$. Similarly, a BR model is encoded as [ ] (empty list) and a full sequence (with no isolated labels) such as $l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4$ is represented as $[l_1, l_2, l_3, l_4]$.

We adopted a simplified controlled approach for generating the initial population. Let μ be the population size. The first individual is an empty chain (i.e., a BR model) and the second individual constitutes a complete CC model with the "default order" (the order specified in the database[1]). The remainder individuals are generated as follows. First, individual lengths are randomly drawn following a uniform distribution in the range $[2, q]$. As a consequence, $\mu - 2$ chromosomes with different lengths will be generated. For each of these chromosomes, starting from the leftmost position, the chain sequence is defined by randomly selecting integer numbers (representing the label indexes) according to a uniform distribution in the range $[1, q]$. Repeated integers (labels) are not allowed in a chromosome.

[1]See Subsection 4.3.1

6.3.2 Lexicographic Fitness Function

In GA-PartCC, we adopted a multi-objective lexicographic approach [31] to determine the fitness of the candidate solutions. In this approach, two or more objectives with distinct predetermined priorities are taken into consideration to define the quality of each chromosome. Consider the following example. Let c_i and c_j be two candidate solutions. In the lexicographic approach used in GA-PartCC, when comparing two chromosomes, the GA first tries to determine which one is better considering the highest priority objective. If c_i is not better than c_j, and vice-versa, then both are compared considering the second objective. The process is repeated until either a winner is found or all the criteria have been tested (in the later case, if no winner was found, one of the chromosomes is randomly chosen as winner).

In GA-PartCC, we only consider two objectives: predictive accuracy (first priority) and model simplicity (second priority). To assess the predictive accuracy of a chromosome, we use the Quality (fitness) function defined in Equation 5.1 (the same used by GACC). If two chromosomes have the same value for the Quality function, they must be compared with regard to model's simplicity (second objective). We consider a solution c_i simpler than c_j if c_i encodes a chain sequence shorter than the one encoded in c_j. The rationale lies primarily in the Occam's Razor principle [25] which states that "given two models with the same generalization error, the simplest one should be preferred because simplicity is desirable in itself". Indeed, apart from being simpler, a shorter model offers two other important advantages (as previously discussed in Section 6.1). First, it is more efficient, as it involves a smaller number of attributes. Second, it gives higher fidelity for representing label dependencies, since only the most relevant labels with regard to the classification problem are present in the chain.

By comparison with the simple use of a weighted formula to combine the two objective values, the lexicographic approach has the advantage of avoiding the specification of ad-hoc numeric weights. It requires only the specification of the relative priority of the objectives, which is well-defined in the context of classification (accuracy clearly has priority over the chain size). By comparison with the Pareto dominance-based approach [20] for multi-objective optimization, the lexicographic approach has the advantages of being simpler and avoiding the issue of how to choose one single solution to be used in practice (out of all non-dominated solutions). Also, the usual Pareto approach would not allow us to specify that maximizing accuracy is more important than reducing the chain size, an important application-specific piece of knowledge that is naturally specified using the

lexicographic approach.

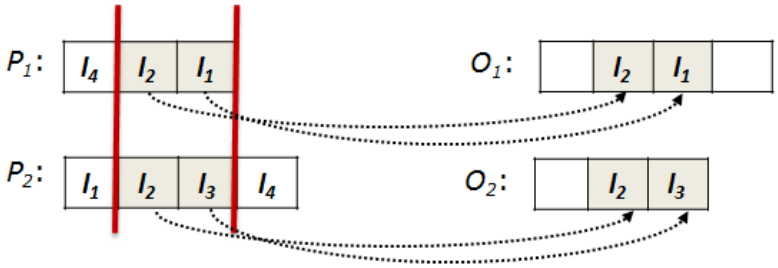
Like GACC, the GA-PartCC follows the wrapper approach [30] in which the quality of an individual (candidate PartCC model) is determined by using the target MLC algorithm (i.e., the CC method). The fitness function is calculated using only the training set, according to a holdout method that works as follows. First, the training set is partitioned into two mutually-exclusive subsets: building (2/3 split) and validation (1/3 split). Next, for each chromosome we build a PartCC model using only the building set, and then that model is evaluated with the validation set.

6.3.3 Genetic Operators

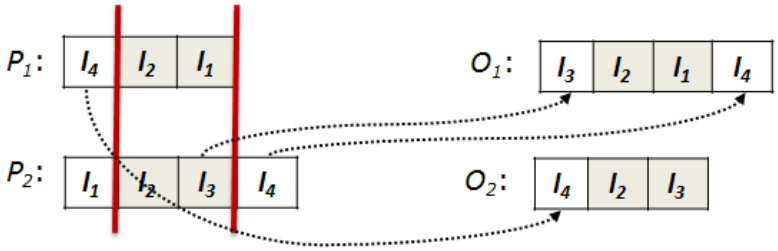
GA-PartCC implements crossover and mutation to deal with the two levels of representation encoded in the chromosomes (chain sequence and chain length).

The crossover operation consists in a modified version of the order crossover [28] method, the same used in GACC and in several other GAs for permutation problems. The adaptation was necessary because the original method can only deal with chromosomes of the same length. In order to facilitate the explanation, consider the example shown in Figure 6.4. Like the original method, our version of order crossover generates two children (represented by O_1 and O_2) from two parents (represented by P_1 and P_2). As shown in the figure, child O_1 must have the same length as P_2 and O_2, the same length as P_1. The crossover operates as follows. Two crossover points (represented by the two vertical thick lines in Figure 6.4) are chosen at random. The first step to generate O_1 is to copy the segment between the crossover points from P_1 into O_1 (Figure 6.4a). The second step consists in filling the remainder empty positions in O_1 with genetic material from P_2 (Figure 6.4b). The procedure works as follows. Starting from the position next to the second crossover point (the fourth position in our example), the values that are present in P_2 but are not contained in O_1 are transferred to the empty positions in O_1, wrapping around when the last position of both chromosomes is reached. O_2 is analogously generated.

The main advantages of the order crossover method used by GA-PartCC are the facts that the order in the parents is preserved in their children and that the procedure always generates valid solutions. However, it does not create children with lengths different from their parents' lengths, since the first child has the same length of the second parent and vice-versa. In order to increase the population's variability of length, in GA-PartCC, children resulting from crossover can be also subject to mutation, which consists in either



(a) Step 1



(b) Step 2

Figure 6.4: Adapted order crossover operation

inserting or removing a sub-chain. In the insertion procedure, a single insertion point is chosen at random in the current child. Next, a sub-chain with maximum length of 5% of q is randomly generated and inserted at that insertion point. The inserted sub-chain must contain only labels that do not occur in the current child. The deletion procedure removes a segment between two randomly-chosen points (also limited to a length of 5% of q).

6.4 Experiments

In this section, we report and discuss the results of four different experimental evaluations involving the GA-PartCC method. Initially, in Section 6.4.1, the experiments were carried out to compare GA-PartCC against the standard BR and CC methods. Section 6.4.2 is devoted to the experimental comparisons encompassing the HBCC and BCC methods (presented in Subsection 4.4.1.2). These methods are based on the exploration of candidate chain sequences and, along with GA-PartCC, are the only two approaches proposed in the literature which are able to produce models composed of partial chains[2]. In Section 6.4.3 we compare GA-PartCC against GACC. Finally, in Section 6.4.4, we address experiments comparing GA-PartCC and the OOCC lazy technique, recently published in our collaborative paper [16]. This is the empirical evaluation involving the largest number of multi-label classifier chains methods so far (six classifier chain methods plus the BR

[2]Although they are able to work with partial chains, these methods differ from GA-PartCC in important aspects, which are discussed throughout Subsection 6.4.2

method).

All experiments reported in this section were performed using the same experimental settings of the experiments presented in the previous chapters: (i) the Weka's J48 decision tree algorithm with default parameters was used as the base binary classification algorithm for all evaluated methods; (ii) a holdout evaluation was performed, where 2/3 of each dataset was used for learning the classifier and 1/3 for testing (using the same training and test splits that come with the datasets); (iii) the predictive performance of the methods was evaluated in terms of Accuracy, F-Measure, Hamming Loss and Exact Match; (iv) we used the Wilcoxon signed-rank test to verify the statistical significance of the results at a confidence level of 95%; (v) the parameters values used in GA-PartCC are the same as those used by GACC (these were presented in Table 5.2); (vi) the results reported for GA-PartCC, GACC, CC, HBCC, BCC and OOCC are averaged over 10 executions with distinct random seeds, except for the larger datasets "enron" and "llog" which were averaged over 5 executions (additional information on how the seeds were employed for each method is provided within each subsection).

6.4.1 GA-PartCC versus BR and CC

In this subsection, the predictive performance of GA-PartCC is compared against the BR and CC methods. Tables 6.2, 6.3, 6.4 and 6.5 present the performance of each method in terms of Accuracy, F-Measure, Exact Match and Hamming loss, respectively.

The results presented in Tables 6.2 and 6.3 show that the GA-PartCC obtained the smallest average rank (i.e., the best overall result) for both Accuracy and F-Measure. The Wilcoxon signed-rank test indicated that GA-PartCC is significantly better than CC for Accuracy ($T_{wilcox}(9) = 0.0$) and that GA-PartCC is significantly better than BR and CC for F-Measure ($T_{wilcox}(9) = 4.0$ in both cases). According to Table 6.4, our genetic algorithm also performed significantly better than both standard methods in terms of Exact Match ($T_{wilcox}(8) = 1.0$ and $T_{wilcox}(7) = 1.5$ in regard to BR and CC, respectively). Finally, no statistically significant differences existed between the Hamming Loss values obtained by the three methods (Table 6.5).

In short, our experiments revealed that, like GACC, GA-PartCC method obtained results significantly superior to CC according to Accuracy, F-Measure and Exact Match. GA-PartCC also obtained results significantly superior to the BR method according to F-Measure and Exact Match.

Table 6.2: Performance of BR, CC and GA-PartCC in terms of Accuracy.

Dataset	Accuracy		
	BR	CC	GA-PartCC
university	0.300 (3.0)	0.310 (2.0)	**0.312** (1.0)
yeast	0.423 (2.0)	0.418 (3.0)	**0.440** (1.0)
ces-16	0.196 (2.0)	0.194 (3.0)	**0.218** (1.0)
birds	**0.573** (1.0)	0.564 (3.0)	0.567 (2.0)
thyroid	**0.984** (1.5)	0.983 (3.0)	**0.984** (1.5)
genbase	**0.987** (2.0)	**0.987** (2.0)	**0.987** (2.0)
medical	0.743 (2.5)	0.743 (2.5)	**0.748** (1.0)
enron	0.367 (3.0)	0.402 (2.0)	**0.405** (1.0)
llog	**0.243** (1.5)	0.239 (3.0)	**0.243** (1.5)
cal500	0.212 (3.0)	0.218 (2.0)	**0.222** (1.0)
average rank	2.15	2.55	**1.30**

{GA-PartCC} $\succ$ {CC}

Table 6.3: Performance of BR, CC and GA-PartCC in terms of F-Measure.

Dataset	F-Measure		
	BR	CC	GA-PartCC
university	0.315 (3.0)	**0.335** (1.0)	0.329 (2.0)
yeast	0.547 (2.0)	0.523 (3.0)	**0.550** (1.0)
ces-16	0.251 (2.0)	0.249 (3.0)	**0.280** (1.0)
birds	**0.603** (1.0)	0.592 (3.0)	0.595 (2.0)
thyroid	0.990 (2.5)	0.990 (2.5)	**0.991** (1.0)
genbase	**0.991** (2.0)	**0.991** (2.0)	**0.991** (2.0)
medical	**0.773** (1.5)	0.769 (2.5)	**0.773** (1.5)
enron	0.474 (3.0)	0.506 (2.0)	**0.507** (1.0)
llog	0.261 (2.0)	0.255 (3.0)	**0.264** (1.0)
cal500	0.344 (3.0)	0.348 (2.0)	**0.355** (1.0)
average rank	2.10	2.40	**1.35**

{GA-PartCC} $\succ$ {BR, CC}

6.4.2 GA-PartCC versus HBCC and BCC

In this subsection, we present the results of an experiment that compared GA-PartCC against two methods that can deal with shorter chains: HBCC [48] and BCC [91, 110]. As with GA-PartCC, we implemented HBCC and BCC in Java.

Both methods were presented in Section 4.4.1.2. In the following, they are briefly summarized. The HBCC method works in two steps. In the first, the Pearson's linear correlation coefficient between each pair of labels in the training set is calculated. According to the results, in the second step, different partial chains are created, each one composed of labels identified as strongly positively correlated. The BCC technique

Table 6.4: Performance of BR, CC and GA-PartCC in terms of Exact Match.

Dataset	Exact Match		
	BR	CC	GA-PartCC
university	0.263 (2.0)	0.248 (3.0)	**0.266** (1.0)
yeast	0.064 (3.0)	0.127 (2.0)	**0.134** (1.0)
ces-16	0.060 (2.0)	0.058 (3.0)	**0.066** (1.0)
birds	0.486 (2.0)	0.485 (3.0)	**0.490** (1.0)
thyroid	0.932 (3.0)	0.941 (1.5)	**0.941** (1.5)
genbase	**0.975** (2.0)	**0.975** (2.0)	**0.975** (2.0)
medical	0.651 (3.0)	0.665 (2.0)	**0.669** (1.0)
enron	0.086 (3.0)	0.125 (2.0)	**0.128** (1.0)
llog	0.199 (2.0)	**0.201** (1.0)	0.198 (3.0)
cal500	**0.000** (2.0)	**0.000** (2.0)	**0.000** (2.0)
average rank	2.40	2.15	**1.55**
{GA-PartCC} $\succ$ {BR, CC}			

Table 6.5: Performance of BR, CC and GA-PartCC in terms of Hamming Loss.

Dataset	Hamming Loss		
	BR	CC	GA-PartCC
university	**0.134** (1.0)	0.139 (3.0)	0.137 (2.0)
yeast	**0.259** (1.0)	0.274 (3.0)	0.263 (2.0)
ces-16	0.188 (2.5)	0.188 (2.5)	**0.187** (1.0)
birds	**0.051** (1.5)	**0.051** (1.5)	0.052 (3.0)
thyroid	**0.005** (1.5)	0.006 (3.0)	**0.005** (1.5)
genbase	**0.001** (2.0)	**0.001** (2.0)	**0.001** (2.0)
medical	**0.011** (2.0)	**0.011** (2.0)	**0.011** (2.0)
enron	**0.054** (1.5)	**0.054** (1.5)	0.055 (3.0)
llog	**0.019** (2.0)	**0.019** (2.0)	**0.019** (2.0)
cal500	**0.163** (1.0)	0.176 (3.0)	0.170 (2.0)
average rank	**1.60**	2.35	2.05
No statistical significance			

combines Bayesian networks with classifier chains and also works in two steps. The first consists in constructing a maximum weighted spanning tree according to the mutual dependence measure between each pair of labels. In the second step, an arbitrary node from the tree must be randomly chosen as the root node. From this root node, the different paths that compose the tree are used to form different partial chains.

Despite the fact that HBCC and BCC are able to work with shorter chains, our GA-PartCC method differs from these techniques in two key aspects. First, the GA-PartCC method delivers to the user a single chain sequence at the end of the GA evolution. On the other hand, neither HBCC nor BCC generate a single specific chain. Instead, both algorithms first identify correlated labels and further employ this information to restrict

the set of possible valid chain orderings. One of the valid chain orderings must then be randomly chosen in order to train the CC model. Second and more importantly, GA-PartCC is actually concerned with the determination of an *optimized chain sequence* (i.e., the label ordering is important to GA-PartCC). Differently, HBCC and BCC rely on the assumption that a good chain sequence is the one in which strongly correlated labels are connected or are placed close to each other, not paying attention to the way the sequence is ordered.

Tables 6.6, 6.7, 6.8 and 6.9 present the results of HBCC, BCC and GA-PartCC concerning, respectively, the measures of Accuracy, F-Measure, Exact Match and Hamming Loss. The results reported for HBCC are averaged over 10 executions with distinct random seeds to define the order of each shorter sequence in the model. The HBCC method requires the specification of a user-input parameter: the threshold value for the Pearson's linear correlation coefficient (denoted as λ), used to define which pairs of labels are correlated in the training set (basically, two labels are regarded as correlated if the computed value of the Pearson's linear correlation coefficient between them is higher than λ). As recommended in [48], we separately optimized the value of this parameter for each dataset involved in our experiments, using only the training part. In this process, we evaluted different values of λ, ranging from 0.0 to 1.0, with steps of 0.05. The results reported for the BCC method are also averaged over 10 executions with distinct random seeds to define the root node of the maximum weighted spanning tree. Differently from HBCC, this method does not require the specification of any user-input parameter.

Table 6.6: Performance of HBCC, BCC and GA-PartCC in terms of Accuracy.

Dataset	Accuracy		
	HBCC	BCC	GA-PartCC
university	0.300 (2.5)	0.300 (2.5)	**0.312** (1.0)
yeast	0.430 (2.0)	0.423 (3.0)	**0.440** (1.0)
ces-16	0.202 (2.0)	0.198 (3.0)	**0.218** (1.0)
birds	0.566 (2.0)	0.560 (3.0)	**0.567** (1.0)
thyroid	**0.984** (2.0)	**0.984** (2.0)	**0.984** (2.0)
genbase	**0.987** (2.0)	**0.987** (2.0)	**0.987** (2.0)
medical	0.749 (1.0)	0.743 (3.0)	**0.748** (2.0)
enron	0.396 (3.0)	**0.405** (1.5)	**0.405** (1.5)
llog	0.241 (2.5)	0.241 (2.5)	**0.243** (1.0)
cal500	0.215 (2.0)	0.209 (3.0)	**0.222** (1.0)
average rank	2.10	2.55	**1.35**
{GA-PartCC} $\succ$ {HBCC, BCC}			

The genetic algorithm approach proved to be very effective, as GA-PartCC signifi-

Table 6.7: Performance of HBCC, BCC and GA-PartCC in terms of F-Measure.

Dataset	F-Measure		
	HBCC	BCC	GA-PartCC
university	0.315 (2.5)	0.315 (2.5)	**0.329** (1.0)
yeast	0.539 (2.0)	0.529 (3.0)	**0.550** (1.0)
ces-16	0.259 (2.0)	0.254 (3.0)	**0.280** (1.0)
birds	**0.595** (1.5)	0.588 (3.0)	**0.595** (1.5)
thyroid	0.990 (3.0)	**0.991** (1.5)	**0.991** (1.5)
genbase	**0.991** (2.0)	**0.991** (2.0)	**0.991** (2.0)
medical	**0.775** (1.0)	0.772 (3.0)	0.773 (2.0)
enron	0.496 (3.0)	**0.509** (1.0)	0.507 (2.0)
llog	0.256 (3.0)	0.258 (2.0)	**0.264** (1.0)
cal500	0.346 (2.0)	0.336 (3.0)	**0.355** (1.0)
average rank	2.20	2.40	**1.40**
{GA-PartCC} $\succ$ {HBCC, BCC}			

Table 6.8: Performance of HBCC, BCC and GA-PartCC in terms of Exact Match.

Dataset	Exact Match		
	HBCC	BCC	GA-PartCC
university	0.263 (2.5)	0.263 (2.5)	**0.266** (1.0)
yeast	0.111 (3.0)	0.118 (2.0)	**0.134** (1.0)
ces-16	0.064 (2.0)	0.061 (3.0)	**0.066** (1.0)
birds	0.483 (2.0)	0.478 (3.0)	**0.490** (1.0)
thyroid	0.941 (2.5)	**0.945** (1.0)	0.941 (2.5)
genbase	**0.975** (2.0)	**0.975** (2.0)	**0.975** (2.0)
medical	**0.671** (1.0)	0.655 (3.0)	0.669 (2.0)
enron	0.125 (2.0)	0.122 (3.0)	**0.128** (1.0)
llog	**0.201** (1.0)	0.197 (3.0)	0.198 (2.0)
cal500	**0.000** (2.0)	**0.000** (2.0)	**0.000** (2.0)
average rank	2.00	2.45	**1.65**
{GA-PartCC} $\succ$ {BCC}			

cantly outperformed both HBCC and BCC in two of the four evaluation measures: Accuracy ($T_{wilcox}(8) = 1.5$ and $T_{wilcox}(7) = 0.0$, respectively) and F-Measure ($T_{wilcox}(8) = 2.0$ in both cases). GACC is also statistically superior to BCC for Exact Match ($T_{wilcox}(8) = 3.0$). None of the methods was significantly superior to any other in terms of Hamming Loss.

The superior performance of GA-PartCC indicates that both issues are important: taking partial chains into consideration and determining an optimized label sequence for training the CC model. The later issue is not considered by the HBCC and BCC methods. It is also important to state that HBCC and BCC are based on the measurement of the unconditional dependence among labels, which is often not sufficient to truly identify

Table 6.9: Performance of HBCC, BCC and GA-PartCC in terms of Hamming Loss.

Dataset	Hamming Loss		
	HBCC	BCC	GA-PartCC
university	**0.134** (1.5)	**0.134** (1.5)	0.137 (3.0)
yeast	0.267 (3.0)	0.266 (2.0)	**0.263** (1.0)
ces-16	0.188 (2.0)	0.189 (3.0)	**0.187** (1.0)
birds	**0.052** (2.0)	**0.052** (2.0)	**0.052** (2.0)
thyroid	**0.005** (2.0)	**0.005** (2.0)	**0.005** (2.0)
genbase	**0.001** (2.0)	**0.001** (2.0)	**0.001** (2.0)
medical	**0.011** (2.0)	**0.011** (2.0)	**0.011** (2.0)
enron	**0.054** (1.5)	**0.054** (1.5)	0.055 (3.0)
llog	**0.019** (2.0)	**0.019** (2.0)	**0.019** (2.0)
cal500	**0.170** (1.5)	0.190 (3.0)	**0.170** (1.5)
average rank	**1.95**	2.10	**1.95**
No statistical significance			

dependencies. On the other hand, GA-PartCC can indirectly cope with this issue by employing a global search method based on a GA that follows the wrapper approach. Furthermore, GA-PartCC offers the advantage of delivering an interpretable result, i.e., at the end of the search the method returns a single optimized partial chain, reflecting the label dependencies, whilst HBCC and BCC return a set of candidate chains.

6.4.3 GA-PartCC versus GACC

In this subsection, we compare the performance of our two proposed chain methods for multi-label classification based on genetic algorithms, GACC and GA-PartCC. Tables 6.10, 6.11, 6.12 and 6.13 show, respectively, the results for the measures of Accuracy, F-Measure, Exact Match, and Hamming Loss.

In this experiment, GACC presented the best average rank for Accuracy and Exact Match and GA-PartCC, the best ones for F-Measure and Hamming Loss. However, the two-tailed Wilcoxon signed-rank test indicated that, with a confidence level of 95%, no statistically significant differences exist between the performance of the two methods considering the four evaluation measures of predictive performance.

The GA-PartCC method was also evaluated with respect to the length of the chains associated to the best solutions. Table 6.14 shows the results averaged over the 10 executions. It is possible to observe that, for the datasets "llog" and "medical", the best models are, on average, composed of nearly half of the labels. For the remaining datasets, with the exception of "genbase", the best found models are, on average, composed of a

Table 6.10: Performance of GACC and GA-PartCC in terms of Accuracy.

Dataset	Accuracy	
	GACC	GA-PartCC
university	**0.319** (1.0)	0.312 (2.0)
yeast	0.432 (2.0)	**0.440** (1.0)
ces-16	0.213 (2.0)	**0.218** (1.0)
birds	**0.569** (1.0)	0.567 (2.0)
thyroid	0.983 (2.0)	**0.984** (1.0)
genbase	**0.987** (1.5)	**0.987** (1.5)
medical	0.744 (2.0)	**0.748** (1.0)
enron	**0.409** (1.0)	0.405 (2.0)
llog	**0.249** (1.0)	0.243 (2.0)
cal500	**0.224** (1.0)	0.222 (2.0)
average rank	**1.45**	1.55
No statistical significance		

Table 6.11: Performance of GACC and GA-PartCC in terms of F-Measure.

Dataset	F-Measure	
	GACC	GA-PartCC
university	**0.343** (1.0)	0.329 (2.0)
yeast	0.548 (2.0)	**0.550** (1.0)
ces-16	0.275 (2.0)	**0.280** (1.0)
birds	**0.597** (1.0)	0.595 (2.0)
thyroid	0.990 (2.0)	**0.991** (1.0)
genbase	**0.991** (1.5)	**0.991** (1.5)
medical	0.769 (2.0)	**0.773** (1.0)
enron	**0.507** (1.5)	**0.507** (1.5)
llog	**0.264** (1.5)	**0.264** (1.5)
cal500	**0.356** (1.0)	0.355 (2.0)
average rank	1.55	**1.45**
No statistical significance		

large number of labels (however, not all labels). This conforms with the results of the exhaustive experiment presented in Section 6.2, where most of the best performing sequences were composed of q or $q-1$ labels. It can also be explained by the fact that in these datasets, a large number of labels exhibit dependence with each other. A notable exception is the dataset "genbase", in which the performance of BR method was equivalent to the performance of the six classifier chain methods evaluated in our experiments for all evaluation measures. In this case, the GA-PartCC method was consistent with the Occam's Razor principle, being able to determine the simplest model (an empty chain) as the best solution. Hence, the results presented in this subsection demonstrate that the proposed GA-PartCC has a predictive performance equivalent to GACC, offering the the

Table 6.12: Performance of GACC and GA-PartCC in terms of Exact Match.

Dataset	Exact Match	
	GACC	GA-PartCC
university	0.255 (2.0)	**0.266** (1.0)
yeast	**0.134** (1.5)	**0.134** (1.5)
ces-16	0.063 (2.0)	**0.066** (1.0)
birds	**0.492** (1.0)	0.490 (2.0)
thyroid	**0.941** (1.5)	**0.941** (1.5)
genbase	**0.975** (1.5)	**0.975** (1.5)
medical	**0.670** (1.0)	0.669 (2.0)
enron	**0.135** (1.0)	0.128 (2.0)
llog	**0.209** (1.0)	0.198 (2.0)
cal500	**0.000** (1.5)	**0.000** (1.5)
average rank	**1.40**	1.60
No statistical significance		

Table 6.13: Performance of GACC and GA-PartCC in terms of Hamming Loss.

Dataset	Hamming Loss	
	GACC	GA-PartCC
university	0.138 (2.0)	**0.137** (1.0)
yeast	0.266 (2.0)	**0.263** (1.0)
ces-16	**0.187** (1.5)	**0.187** (1.5)
birds	**0.051** (1.0)	0.052 (2.0)
thyroid	0.006 (2.0)	**0.005** (1.0)
genbase	**0.001** (1.5)	**0.001** (1.5)
medical	**0.011** (1.5)	**0.011** (1.5)
enron	**0.055** (1.5)	**0.055** (1.5)
llog	**0.019** (1.5)	**0.019** (1.5)
cal500	0.173 (2.0)	**0.170** (1.0)
average rank	1.65	**1.35**
No statistical significance		

advantage of generating simpler models though.

6.4.4 GA-PartCC versus OOCC

In this subsection, we compare the performance of GA-PartCC against the state-of-the-art OOCC method for MLC [15, 16], presented in Section 4.4.2.2.

The OOCC technique works by employing a lazy approach that searches for a distinct and more effective label sequence to each new instance t. In this strategy, for each instance in the training dataset, an effective label sequence is previously identified in the training step (according to the Quality metric). At classification time, the label sequence for the

Table 6.14: Average length of the best chain determined by GA-PartCC

Dataset	q	avg length
university	10	7.2
yeast	14	13.8
ces-16	16	14.0
birds	19	15.0
thyroid	25	24.4
genbase	27	0.0
medical	45	20.2
enron	53	52.0
llog	74	37.4
cal500	174	124.8

new instance t is chosen based on the sequences associated with the instances in the training dataset that are more similar to t to be classified.

The parameter values used in OOCC were the same used in [16]: $k = 5$ (number of neighbours), $m = 5$ (number of data partitions) and $r = 15$ (number of sequences trained for each data partition). The results reported are averaged over 10 executions with distinct random seeds used to create the label sequences associated to each data partition. Tables 6.15, 6.16, 6.17 and 6.18 show, respectively, the results for the measures of Accuracy, F-Measure, Exact Match, and Hamming Loss.

Table 6.15: Performance of OOCC and GA-PartCC in terms of Accuracy.

Dataset	Accuracy	
	OOCC	GA-PartCC
university	0.305 (2.0)	**0.312** (1.0)
yeast	0.422 (2.0)	**0.440** (1.0)
ces-16	0.207 (2.0)	**0.218** (1.0)
birds	**0.567** (1.5)	**0.567** (1.5)
thyroid	**0.984** (1.5)	**0.984** (1.5)
genbase	**0.987** (1.5)	**0.987** (1.5)
medical	**0.751** (1.0)	0.748 (2.0)
enron	0.401 (2.0)	**0.405** (1.0)
llog	**0.245** (1.0)	0.243 (2.0)
cal500	0.220 (2.0)	**0.222** (1.0)
average rank	1.65	**1.35**
No statistical significance		

In this experiment, GA-PartCC presented the best average rank in three out of the four evaluation metrics (Accuracy, F-Measure and Hamming Loss). According to the Wilcoxon test, the F-Measure values obtained by GA-PartCC were significantly superior to the ones obtained by OOCC ($T_{wilcox}(7) = 2.0$). The results demonstrate that the

Table 6.16: Performance of OOCC and GA-PartCC in terms of F-Measure.

Dataset	F-Measure	
	OOCC	GA-PartCC
university	0.325 (2.0)	**0.329** (1.0)
yeast	0.526 (2.0)	**0.550** (1.0)
ces-16	0.265 (2.0)	**0.280** (1.0)
birds	0.593 (2.0)	**0.595** (1.0)
thyroid	0.990 (2.0)	**0.991** (1.0)
genbase	**0.991** (1.5)	**0.991** (1.5)
medical	**0.775** (1.0)	0.773 (2.0)
enron	0.503 (2.0)	**0.507** (1.0)
llog	0.261 (2.0)	**0.264** (1.0)
cal500	0.350 (2.0)	**0.355** (1.0)
average rank	1.85	**1.15**
{GA-PartCC} $\succ$ {OOCC}		

Table 6.17: Performance of OOCC and GA-PartCC in terms of Exact Match.

Dataset	Exact Match	
	OOCC	GA-PartCC
university	0.249 (2.0)	**0.266** (1.0)
yeast	**0.135** (1.0)	0.134 (2.0)
ces-16	0.065 (2.0)	**0.066** (1.0)
birds	**0.494** (1.0)	0.490 (2.0)
thyroid	**0.943** (1.0)	0.941 (2.0)
genbase	**0.975** (1.5)	**0.975** (1.5)
medical	**0.677** (1.0)	0.669 (2.0)
enron	0.126 (2.0)	**0.128** (1.0)
llog	**0.206** (1.0)	0.198 (2.0)
cal500	**0.000** (1.5)	**0.000** (1.5)
average rank	**1.30**	1.70
No statistical significance		

proposed GA-PartCC method exhibits a very competitive performance, offering the advantage of delivering much faster classification times; since OOCC, being a lazy method, has a very long classification time.

6.5 Concluding Remarks

This chapter presented a novel method for multi-label classifier chains based on a genetic algorithm. This method, named GA-PartCC, differs from the majority of current extensions to the original CC model because it is capable of evaluating chain sequences that vary not only in the ordering but also in the length. In order to accomplish this task,

Table 6.18: Performance of OOCC, GACC and GA-PartCC in terms of Hamming Loss.

Dataset	Hamming Loss	
	OOCC	GA-PartCC
university	0.138 (2.0)	**0.137** (1.0)
yeast	0.272 (2.0)	**0.263** (1.0)
ces-16	0.188 (2.0)	**0.187** (1.0)
birds	0.055 (2.0)	**0.052** (1.0)
thyroid	**0.005** (1.5)	**0.005** (1.5)
genbase	**0.001** (1.5)	**0.001** (1.5)
medical	**0.010** (1.0)	0.011 (2.0)
enron	**0.055** (1.5)	**0.055** (1.5)
llog	**0.019** (1.5)	**0.019** (1.5)
cal500	0.175 (2.0)	**0.170** (1.0)
average rank	1.70	**1.30**
No statistical significance		

GA-PartCC uses a variable-length list representation and a multi-objective lexicographic fitness function, taking into account two objectives: the model's accuracy and the model's size.

We reported the results of a comprehensive set of experiments that compared the effectiveness of GA-PartCC against BR, CC and four alternative multi-label classifier chain methods (HBCC, BCC, GACC and OOCC). In our experiments: GA-PartCC achieved statistically significantly better results than BR, CC, BCC, HBCC and OOCC in the F-Measure performance measure; GA-PartCC was significantly superior to CC, BCC and HBCC in the Accuracy measure; And the performance of GA-PartCC was also significantly superior to BR, CC and BCC considering the Exact Match measure. None of the evaluated methods was significantly superior to GA-PartCC in any of the four performance measures.

The experiments indicated that the predictive performance of GA-PartCC is similar to the one of GACC. Nonetheless, the simpler, more compact model produced by GA-PartCC offers an important advantage: it gives higher fidelity for representing label dependencies, since only the most relevant labels with regard to the classification problem are present in the chain.

Chapter 7

Conclusions

In the last few years, multi-label classification (MLC) has been a highly active topic of research within the data mining and machine learning communities. In this problem, each object of a dataset may belong to multiple class labels and the goal is to learn a model that can infer the correct labels of new, previously unseen, objects. The general objective of this thesis is to contribute to the development of the MLC area. More specifically, the focus of this research was on the proposal of novel strategies for improving the effectiveness of the classifier chains method (CC), one of the most widely known techniques for mining multi-label classifiers.

In this chapter, we conclude this thesis by giving a detailed summary of our findings and contributions (Section 7.1) and pointing towards potential future work (Section 7.2).

7.1 Thesis Contributions

First proposed in 2009, CC has become one of the most influential methods for multi-label classification. It is distinguished from other methods by its simple and effective approach to exploit label dependencies. The CC method involves the training of q single-label binary classifiers, where q is the number of labels and each classifier is solely responsible for classifying a specific label. These q classifiers are linked in a chain, such that each binary classifier is able to incorporate the predictions of the previous ones as additional information at classification time. Thus, possible correlations among labels – inherent in several MLC problems – can be straightforwardly exploited. However, the basic CC model suffers from two major drawbacks:

1. It decides the label ordering at random, although this ordering has a strong effect

on predictive accuracy.

2. It forces all labels to be present in the chain, despite the fact that some of them might cause a decrease in the predictive accuracy of the model.

The main contribution of this thesis was the proposal of two novel techniques for improving the effectiveness of multi-label chain classifiers: GACC (Chapter 5) and GA-PartCC (Chapter 6). Both of them make use of Genetic Algorithms (GAs) to perform a global search for a single optimized label sequence (i.e., a label sequence that leads to an improvement on the predictive accuracy of the CC model). One of the proposed strategies (GA-PartCC) is capable of taking into consideration chain sequences that vary not only in the ordering but also in the length, thus tackling both problems of the original CC method at once. Experiments on diverse benchmark datasets, followed by the Wilcoxon test for assessing statistical significance, indicate that our proposed GAs produce more accurate chain classifiers.

Secondary contributions of the thesis mainly aimed at obtaining a greater understanding of underlying principles of the CC model. In this regard, in Chapter 4, we reported a study that demonstrated the importance of using a good label sequence for training a CC model independently of the base single-label algorithm. Subsequently, we proposed and evaluated a group of baseline heuristics to find a good order for training CC models (PredCC, FreqCC and DepCC). In Chapter 6, we carried out a study to investigate the influence of the label sequence length in the CC's effectiveness. In the same chapter, we introduce a novel conceptual model for multi-label classification, named PartCC.

In the remainder of this section, we summarize the major findings of Chapters 4, 5 and 6 in order to provide details about the above contributions.

7.1.1 Multi-Label Chain Classifiers – Chapter 4

In Chapter 4, two secondary contributions of this thesis were presented. The first consisted in an exhaustive experiment that, for the first time, investigated in depth the influence of the label sequence in the predictive performance of CC models. The experiment consisted in assessing the predictive performance of CC considering all $q!$ label permutations of three benchmark datasets using four distinct single-label base algorithms: k-NN, C4.5, Naïve Bayes, and SMO. Our main motivation for carrying out this empirical analysis was the fact that, in the MLC literature, we found divergent views on the importance of the label sequence issue. For instance, while the authors of CC in [82, 83] argued that, due to

the effect of error propagation, it is likely that different label orderings might lead to the induction of CC models that have very different predictive power, the authors of [93] do not share the same belief, and argued that the effect of the chain order would probably not be significant for datasets in which the number of features is much higher than the number of labels (the most typical case in real-world applications).

The results of our exhaustive experiment indicated that, overall, the order of the chain indeed has a strong effect on predictive performance. However, additional findings evidenced that the different single-label base algorithms, due to their own characteristics, are affected to different degrees by the label ordering. For instance, the effect tends to be very large when the base algorithm is C.45, but it can be rather small for Naïve Bayes. Furthermore, the exhaustive experiment also allowed us to identify that, in general, an effective label sequence for a specific base algorithm (e.g.: C4.5) does not necessarily constitute a good sequence for other base algorithm (e.g.: SMO). Part of the results of this experiment were published in [16].

The second contribution in Chapter 4 consisted of the proposal of three simple baseline heuristics for the determination of optimized label sequences, namely: PredCC (most confident labels first), FreqCC (most frequent labels first), and DepCC ("most-dependent" labels last). In spite of being very simple, none of the three heuristics had been previously examined in the multi-label literature.

We conducted an experiment on a set of benchmark datasets with the aim of comparing PredCC, FreqCC and DepCC against the original CC method. Nonetheless, the obtained results indicated that, according to four distinct measures of predictive performance, there is no statistically significant difference between CC (where the label sequence is simply decided at random) and the presented baselines. Consequently, we concluded that it is necessary to adopt more sophisticated heuristics to overcome the label sequence optimization problem (LSOP). Motivated by this issue, in Chapter 5 we proposed the GACC method.

7.1.2 The GACC Method – Chapter 5

In Chapter 5, we presented the first main contribution of this thesis: the GACC method, published in [42]. This corresponds to the first proposed strategy based on the evolutionary paradigm of GAs to optimize multi-label chain classifiers. We opted for a solution based on GAs mainly due to the following reasons: (i) GAs perform a global search capable of effectively exploring the extremely large search space of $q!$ associated with the

LSOP; (ii) GAs have been traditionally applied to solve a large number of classification and permutation problems.

In GACC, each chromosome represents a different label sequence and the fitness function combines three performance evaluation measures: Exact Match, Accuracy, and Hamming Loss. The crossover operation works by transferring sub-chains of random length between pairs of individuals whilst mutation swaps pairs of labels of an individual. The proposed GA follows the wrapper approach, evaluating the quality of an individual (candidate label sequence) by using the target MLC method (i.e., the CC method).

Our proposed solution based on GAs was compared against the CC method and also against the BR method (which trains q independent binary classifiers). As opposed to the baseline heuristics of Chapter 4, GACC outperformed the original CC method with statistical significance in three out of the four evaluated measures of predictive performance (Accuracy, F-Measure and Exact Match). Additionally, GACC significantly outperformed BR according to the Accuracy and Exact Match measures. In summary, GACC yield competitive results, yet offering the advantage of delivering an interpretable result to the user (a single optimized chain ordering, reflecting the label dependencies).

7.1.3 The GA-PartCC Method – Chapter 6

In Chapter 6, we presented the second main contribution of this thesis: the GA-PartCC method, published in [43]. It represents a natural extension of GACC proposed with the goal of filling a gap in the range of available classifier chain methods in the literature: the fact that none of these methods have yet explored the idea of generating models defined by optimized partial chains[1].

Before describing the GA-PartCC method, we introduced and formally defined the concept of partially chained multi-label model (PartCC). This kind of MLC model is formed by two sets of labels: isolated and chained. The labels in the first set are predicted by independent binary classifiers whilst the ones in the later set are linked in a partial chain so as to be predicted according to the classifier chains framework. Hence, a PartCC model represents a hybrid model between BR and CC.

Subsequently, we carried out an exhaustive experiment (similar to the one reported in Chapter 4) which demonstrated the potential of inducing PartCC models from MLC datasets. Then we introduced the GA-PartCC method, a genetic algorithm for mining

[1]Although the methods HBCC and BCC can work with shorter chains, they do not pay attention to the way the chains are ordered, as discussed in Subsection 6.4.2

optimized partially chained models. This method adopts a variable-length list representation and makes use of a multi-objective lexicographic fitness function, which takes into account two objectives when evaluating a candidate solution: the model's accuracy and the model's size.

We reported the results of a comprehensive set of experiments to evaluate the predictive performance of GA-PartCC. In the first experiment, we compared GA-PartCC against CC and BR, where our genetic algorithm outperformed CC with statistical significance in three out of the four measures (Accuracy, F-Measure and Exact Match) and BR in two measures (F-Measure and Exact Match). In the second experiment, we compared GA-PartCC against HBCC and BCC, two alternative multi-label classifier chain methods based on building models composed of a set of partial chains. Although GA-PartCC works with simpler models (composed of only two sets – isolated labels and chained labels), our experiments revealed that, overall, it obtained superior results in comparison with both HBCC and BCC. In the third experiment, we compared GA-PartCC against GACC. Despite the fact that there was no statistically significant different between the results of GA-PartCC and GACC, the first method offers the advantage of generating simpler, more compact models, which gives higher fidelity for representing label dependencies. In the last experiment, we compared GA-PartCC against the state-of-the-art OOCC lazy approach. In this experiment, GA-PartCC achieved statistically significantly better results according to the F-Measure performance measure. As a final remark, it is important to state that none of the evaluated methods (CC, BR, HBCC, BCC, GACC and OOCC) was significantly superior to GA-PartCC in any of the four performance measures.

7.2 Future Work

In this thesis, we proposed two novel classifier chain methods based on genetic algorithms, GACC and GA-PartCC. We consider that both methods occupy an important niche in the multi-label classification field: they are competitive on diverse multi-label problems, yet being suitable for use with interpretable classifiers.

Some possibilities for future research are as follows. First, extend the GAs with local search methods to add some intelligence across the successive generations. For instance, a possible approach would be to identify, during the evolutionary process, groups of labels that tend to cause a decrease in the accuracy of most other labels when they take part in the chain. These labels could then be either removed from children resulting from

crossover or placed in the last positions of these children. Similarly, groups of labels that tend to cause an increase in the predictive accuracy of other labels could be identified by the local search procedure and placed in the first positions of new children.

A second possibility for future research would be to develop a chromosome representation capable of encoding not only the chain sequence, but also the base single-label algorithm (along with its parameter settings) that will be used to induce the classifier associated to each label. Thus, it would be possible to evaluate a PartCC model in which, for instance, the first label in the chain would be associated with a C4.5 classifier, the second with an SMO classifier, and so on.

Finally, we also consider that it would be interesting to search for optimized PartCC models with the use of other heuristics different from genetic algorithms.

References

[1] Agrawal, R.; Imieliński, T.; Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1993), SIGMOD '93, ACM, pp. 207–216.

[2] Back, T.; Fogel, D. B.; Michalewicz, Z., Eds. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1997.

[3] Becker, L. A.; Seshadri, M. Comprehensibility and overfitting avoidance in genetic programming for technical trading rules. Tech. rep., Worcester Polytechnic Institute, May 2003.

[4] Berry, M. L. A.; Linoff, G., Eds. *Data Mining Techniques: for Marketing, Sales and Customer Support*. J. Wiley Computer Publishing, 1997.

[5] Beyer, H.-G.; Schwefel, H.-P. Evolution strategies: A comprehensive introduction. *Natural Computing 1*, 1 (2002), 3–52.

[6] Bielza, C.; Li, G.; Larrañaga, P. Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning 52*, 6 (Sept. 2011), 705–727.

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

[8] Briggs, F.; Huang, Y.; Raich, R.; Eftaxias, K.; Lei, Z.; Cukierski, W.; Hadley, S. F.; Hadley, A.; Betts, M.; Fern, X. Z.; Irvine, J.; Neal, L.; Thomas, A.; Fodor, G.; Tsoumakas, G.; Ng, H. W.; Nguyen, T. N. T.; Huttunen, H.; Ruusuvuori, P.; Manninen, T.; Diment, A.; Virtanen, T.; Marzat, J.; Defretin, J.; Callender, D.; Hurlburt, C.; Larrey, K.; Milakov, M. The 9th annual MLSP competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2013, Southampton, United Kingdom, September 22-25, 2013* (2013), pp. 1–8.

[9] Brin, S.; Motwani, R.; Silverstein, C. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1997), SIGMOD '97, ACM, pp. 265–276.

[10] Cerri, R.; Barros, R. C.; de Carvalho, A. C. P. L. F. A genetic algorithm for hierarchical multi-label classification. In *Proceedings of the 27th Annual ACM*

Symposium on Applied Computing (New York, NY, USA, 2012), SAC '12, ACM, pp. 250–255.

[11] Cherman, E. A.; Metz, J.; Monard, M. C. Incorporating label dependency into the binary relevance framework for multi-label classification. *Expert Systems with Applications 39*, 2 (2012), 1647–1655.

[12] Clare, A.; King, R. D. Knowledge discovery in multi-label phenotype data. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery* (London, UK, 2001), PKDD '01, Springer-Verlag, pp. 42–53.

[13] Cormen, T.; Leiserson, C.; Rivest, R.; Stein, C. *Introduction To Algorithms*, third ed. MIT Press, 2009.

[14] Crammer, K.; Singer, Y. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research 3* (March 2003), 1025–1058.

[15] da Silva, P. N. *Classificação Multirrótulo em Cadeia: Novas Abordagens*. PhD thesis, Institute of Computing, Universidade Federal Fluminense (IC-UFF), 2014.

[16] da Silva, P. N.; Gonçalves, E. C.; Plastino, A.; Freitas, A. A. Distinct chains for different instances: An effective strategy for multi-label classifier chains. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II* (2014), vol. 8725 of *Lecture Notes in Computer Science*, Springer, pp. 453–468.

[17] Davis, L. Applying adaptive algorithms to epistatic domains. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1* (San Francisco, CA, USA, 1985), IJCAI'85, Morgan Kaufmann Publishers Inc., pp. 162–164.

[18] de Carvalho, A. C. P. L. F.; Freitas, A. A. A tutorial on multi-label classification techniques. In *Foundations of Computational Intelligence Volume 5*, A. Abraham, A.-E. Hassanien, and V. Snášel, Eds., vol. 205 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2009, pp. 177–195.

[19] De Jong, K. A. *Evolutionary Computation - a Unified Approach*. MIT Press, 2006.

[20] Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[21] Dembczynski, K.; Cheng, W.; Hüllermeier, E. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel* (2010), Omnipress, pp. 279–286.

[22] Dembczynski, K.; Waegeman, W.; Hüllermeier, E. An analysis of chaining in multi-label classification. In *Proceedings of the 20th European Conference on Artificial Intelligence* (Montpellier, France, August 2012), L. De Raedt, C. Bessiere, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. Lucas, Eds., ECAI'12, IOS Press, pp. 294–299.

[23] Dietterich, T. G. Machine learning. In *Encyclopedia of Cognitive Science*, L. Nadel, Ed., vol. 2. London: Nature Publishing Group, 2003, pp. 971–981.

[24] Diplaris, S.; Tsoumakas, G.; Mitkas, P. A.; Vlahavas, I. Protein classification with multiple algorithms. In *Advances in Informatics*, P. Bozanis and E. Houstis, Eds., vol. 3746 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 448–456.

[25] Domingos, P. The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery 3*, 4 (1999), 409–425.

[26] Duda, R. O.; Hart, P. E. *Pattern classification and scene analysis*. John Wiley and Sons, 1973.

[27] Duman, E.; Ozcelik, M. H. Detecting credit card fraud by genetic algorithm and scatter search. *Expert Systems with Applications 38*, 10 (September 2011), 13057–13063.

[28] Eiben, A. E.; Smith, J. E. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[29] Elisseeff, A.; Weston, J. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]* (2001), pp. 681–687.

[30] Freitas, A. A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Natural Computing Series. Springer, 2002.

[31] Freitas, A. A. A critical review of multi-objective optimization in data mining: A position paper. *SIGKDD Explorations Newsletters 6*, 2 (December 2004), 77–86.

[32] Freitas, A. A. A review of evolutionary algorithms for data mining. In *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer US, 2010, pp. 371–400.

[33] Freitas, A. A. Comprehensible classification models: A position paper. *SIGKDD Explorations Newsletter 15*, 1 (June 2013), 1–10.

[34] Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian network classifiers. *Machine Learning 29*, 2-3 (Nov. 1997), 131–163.

[35] Fürnkranz, J.; Hüllermeier, E.; Mencía, E. L.; Brinker, K. Multilabel classification via calibrated label ranking. *Machine Learning 73*, 2 (November 2008), 133–153.

[36] Ghaffarizadeh, A.; Ahmadi, K.; Flann, N. S. Sorting unsigned permutations by reversals using multi-objective evolutionary algorithms with variable size individuals. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June, 2011* (2011), pp. 292–295.

[37] Gibaja, E.; Ventura, S. Multi-label learning: A review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 4*, 6 (2014), 411–444.

[38] Gibaja, E.; Ventura, S. A tutorial on multilabel learning. *ACM Computing Surveys (CSUR) 47*, 3 (April 2015), 52:1–52:38.

[39] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[40] Gonçalves, E. C. Regras de associação e suas medidas de interesse objetivas e subjetivas. *INFOCOMP 4* (2005), 26–35.

[41] Gonçalves, E. C. A human-centered approach for mining hybrid-dimensional association rules. In *Proceedings of the 2014 Conference on Information Fusion* (Salamanca, Spain, July 2014), FUSION'14, pp. 1–8.

[42] Gonçalves, E. C.; Plastino, A.; Freitas, A. A. A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In *Proceedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence* (Washington, DC, USA, November 2013), ICTAI'13, IEEE Computer Society, pp. 469–476.

[43] Gonçalves, E. C.; Plastino, A.; Freitas, A. A. Simpler is better: A novel genetic algorithm to induce compact multi-label chain classifiers. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference* (Madrid, Spain, July 2015), GECCO'15, ACM, pp. 559–566.

[44] Gonçalves, I.; Silva, S. Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In *Genetic Programming*, K. Krawiec, A. Moraglio, T. Hu, A. Etaner-Uyar, and B. Hu, Eds., vol. 7831 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 73–84.

[45] Guo, Y.; Gu, S. Multi-label classification using conditional dependency networks. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Two* (Barcelona, Catalonia, Spain, July 2011), IJCAI'11, AAAI Press, pp. 1300–1305.

[46] Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H. The weka data mining software: an update. *ACM SIGKDD Exploration Newsletter 11*, 1 (November 2009), 10–18.

[47] Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*, 3rd ed. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, San Francisco, CA, USA, 2011.

[48] Hernandez-Leal, P.; Orihuela-Espina, F.; Sucar, L. E.; Morales, E. F. Hybrid binary-chain multi-label classifiers. In *Proceedings of the 6th European Workshop on Probabilistic Graphical Models* (Granada, Spain, September 2012), PGM'12, pp. 139–146.

[49] Hüllermeier, E.; Fürnkranz, J.; Cheng, W.; Brinker, K. Label ranking by learning pairwise preferences. *Artificial Intelligence 172*, 16-17 (November 2008), 1897–1916.

[50] Husbands, P.; Copley, P.; Eldridge, A.; Mandelis, J. An introduction to evolutionary computing for musicians. In *Evolutionary Computer Music*, E. Miranda and J. Biles, Eds. Springer London, 2007, pp. 1–27.

[51] Japkowicz, N.; Shah, M. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, New York, 2011.

[52] Joachims, T. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning* (London, UK, UK, 1998), ECML '98, Springer-Verlag, pp. 137–142.

[53] Jungjit, S.; Freitas, A. A. A lexicographic multi-objective genetic algorithm for multi-label correlation based feature selection. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference* (Madrid, Spain, 2015), GECCO Companion '15, ACM, pp. 989–996.

[54] Karakatic, S.; Podgorelec, V. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing 27*, 0 (2015), 519–532.

[55] Karalič, A.; Pirnat, V. Significance level based multiple tree classification. *Informatica 15*, 5 (1991).

[56] Klimt, B.; Yang, Y. The enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings* (2004), pp. 217–226.

[57] Koza, J. R. *Genetic Programming: On the Programming of Computing by Means of Natural Selection*. MIT Press, 1992.

[58] Kumar, A.; Vembu, S.; Menon, A. K.; Elkan, C. Beam search algorithms for multilabel learning. *Machine Learning 92*, 1 (July 2013), 65–89.

[59] Larrañaga, P.; Kuijpers, C. M. H.; Murga, R. H.; Inza, I.; Dizdarevic, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review 13*, 2 (April 1999), 129–170.

[60] Levy, E.; David, O. E.; Netanyahu, N. S. Genetic algorithms and deep learning for automatic painter classification. In *Proc. of the 2014 Genetic and Evolutionary Computation Conference* (July 2014), GECCO'14, pp. 1143–1150.

[61] Lichman, M. UCI machine learning repository, 2013. University of California, Irvine, School of Information and Computer Sciences, `http://archive.ics.uci.edu/ml`.

[62] Liu, B.; Hsu, W.; Ma, Y. Integrating classification and association rule mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (1998), AAAI Press, pp. 80–86.

[63] Liu, X.; Shi, Z.; Li, Z.; Wang, X.; Shi, Z. Sorted label classifier chains for learning images with multi-label. In *Proceedings of the ACM 2010 International Conference on Multimedia* (Florence, Italy, October 2010), MM'10, pp. 951–954.

[64] Machado, P.; Tavares, J.; Pereira, F. B.; Costa, E. Vehicle routing problem: Doing it the evolutionary way. In *Proceedings of the 2002 Genetic and Evolutionary Computation Conference* (July 2002), GECCO'02, pp. 690–696.

[65] Madjarov, G.; Dimitrovski, I.; Gjorgjevikj, D.; ; Džeroski, S. Evaluation of different data-derived label hierarchies in multi-label classification. In *New Frontiers in Mining Complex Patterns - Third International Workshop, NFMCP 2014, Held in Conjunction with ECML-PKDD 2014, Nancy, France, September 19, 2014, Revised Selected Papers* (2014), pp. 19–37.

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

[67] Montañes, E.; Senge, R.; Barranquero, J.; Ramón Quevedo, J.; José Del Coz, J.; Hüllermeier, E. Dependent binary relevance models for multi-label classification. *Pattern Recognition 47*, 3 (2014), 1494–1508.

[68] Moon, C.; Kim, J.; Choi, G.; Seo, Y. An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research 140*, 3 (2002), 606–617.

[69] Newby, D.; Freitas, A. A.; Ghafourian, T. Comparing multilabel classification methods for provisional biopharmaceutics class prediction. *Molecular Pharmaceutics 12*, 1 (2015), 87–102.

[70] Pestian, J. P.; Brew, C.; Matykiewicz, P.; Hovermale, D. J.; Johnson, N.; Cohen, K. B.; Duch, W. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing* (Stroudsburg, PA, USA, 2007), BioNLP '07, pp. 97–104.

[71] Platt, J. C. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. MIT Press, Cambridge, MA, USA, 1999, pp. 185–208.

[72] Potvin, J.-Y. Genetic algorithm for traveling salesman problem. *Annals of Operation Research 63* (1996), 339–370.

[73] Potvin, J.-Y. State-of-the art review - evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing 21*, 4 (2009), 518–548.

[74] Quinlan, J. R. Induction of decision trees. *Machine Learning 1*, 1 (1986), 81–106.

[75] Quinlan, J. R. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, 1993.

[76] Quinlan, J. R.; Compton, P. J.; Horn, K. A.; Lazurus, L. Inductive knowledge acquisition: A case study. In *Proc. of the 2nd Australian Conference on Applications of Expert Systems* (1986), pp. 183–204.

[77] Read, J. *Scalable multi-label classification*. PhD thesis, Department of Computer Science, University of Waikato, 2010.

[78] Read, J.; Martino, L.; Luengo, D. Efficient monte carlo optimization for multi-label classifier chains. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013* (2013), IEEE, pp. 3457–3461.

[79] Read, J.; Martino, L.; Luengo, D. Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition 47*, 3 (2014), 1535–1546.

[80] Read, J.; Martino, L.; Olmos, P. M.; Luengo, D. Scalable multi-output label prediction: From classifier chains to classifier trellises. *Pattern Recognition 48*, 6 (2015), 2096–2109.

[81] Read, J.; Pfahringer, B.; Holmes, G. Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining* (Pisa, Italy, December 2008), ICDM'08, IEEE Computer Society, pp. 995–1000.

[82] Read, J.; Pfahringer, B.; Holmes, G.; Eibe, F. Classifier chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II* (Bled, Slovenia, September 2009), ECML PKDD '09, Springer-Verlag, pp. 254–269.

[83] Read, J.; Pfahringer, B.; Holmes, G.; Eibe, F. Classifier chains for multi-label classification. *Machine Learning 85*, 3 (December 2011), 333–359.

[84] Read, J.; Reutemann, P. Meka: A multi-label extension to weka, 2015. Retrieved May 13, 2015 from `http://meka.sourceforge.net/`.

[85] Reeves, C. R. Genetic algorithms. In *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds., 2nd ed. Springer Publishing Company, Incorporated, 2010, pp. 109–139.

[86] Rokach, L. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis 53*, 12 (2009), 4046 – 4072.

[87] Romero, A. E.; de Campos, L. M. A probabilistic methodology for multilabel classification. *Intelligent Data Analysis 18*, 5 (September 2014), 911–926.

[88] Ruepp, A.; Zollner, A.; Maier, D.; Albermann, K.; Hani, J.; Mokrejs, M.; Tetko, I.; Güldener, U.; Mannhaupt, G.; Münsterkötter, M.; Mewes, H. W. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research 32*, 18 (January 2004), 5539–5545.

[89] Sait, S. M.; Youssef, H. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*, 1st ed. IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

[90] Spyromitros, E.; Tsoumakas, G.; Vlahavas, I. An empirical study of lazy multilabel classification algorithms. In *Proceedings of the 5th Hellenic Conference on Artificial Intelligence: Theories, Models and Applications* (Berlin, Heidelberg, 2008), SETN '08, Springer-Verlag, pp. 401–406.

[91] Sucar, L. E.; Bielza, C.; Morales, E. F.; Hernandez-Leal, P.; Zaragoza, J. H.; Larrañaga, P. Multi-label classification with bayesian network-based chain classifiers. *Pattern Recognition Letters 41* (May 2014), 14–22.

[92] Tan, P.-N.; Steinbach, M.; Kumar, V. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[93] Tenenboim-Chekina, L.; Rokach, L.; Shapira, B. Identification of label dependencies for multi-label classification. In *Proceedings of the 2nd International Workshop on Learning from Multi-Label Data (MLD'10) in Conjunction with ICML 2010* (Haifa, Israel, June 2010), MLD'10, pp. 53–60.

[94] Toroslu, I. H.; Arslanoglu, Y. Genetic algorithm for the personnel assignment problem with multiple objectives. *Inf. Sci. 177*, 3 (February 2007), 787–803.

[95] Trohidis, K.; Tsoumakas, G.; Kalliris, G.; Vlahavas, I. Multilabel classification of music into emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)* (Philadelphia, PA, USA, June 2008), ISMIR'08, pp. 325–330.

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

[97] Tsoumakas, G.; Katakis, I.; Vlahavas, I. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer US, 2010, pp. 667–685.

[98] Tsoumakas, G.; Vlahavas, I. Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 18th European Conference on Machine Learning* (Warsaw, Poland, September 2007), ECML'07, Springer-Verlag, pp. 406–417.

[99] Tsoumakas, G.; Xioufis, E. S.; Vilcek, J.; Vlahavas, I. P. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research 12* (2011), 2411–2414.

[100] Tsoumakas, G.; Zhang, M.-L.; Zhou, Z.-H. Tutorial on learning from multi-label data, 2009. Retrieved October 14, 2014 from `http://www.ecmlpkdd2009.net/wp-content/uploads/2009/08/learningfrom-multi-label-data.pdf`.

[101] Turnbull, D.; Barrington, L.; Torres, D.; Lanckriet, G. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing 16*, 2 (February 2008), 467–476.

[102] van der Gaag, L. C.; de Waal, P. R. Multi-dimensional bayesian network classifiers. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models, Electronic Proceedings* (Prague, Czech Republic, 2006), PGM'06, pp. 107–114.

[103] Whitley, E.; Ball, J. Statistics review 6: Nonparametric methods. *Critical Care 6*, 6 (2002), 509–513.

[104] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics 1* (1945), 80–83.

[105] Witten, I. H.; Frank, E.; Hall, M. A. *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, San Francisco, CA, USA, 2011.

[106] Wu, X.; Kumar, V.; Ross Quinlan, J.; Ghosh, J.; Yang, Q.; Motoda, H.; McLachlan, G. J.; Ng, A.; Liu, B.; Yu, P. S.; Zhou, Z.-H.; Steinbach, M.; Hand, D. J.; Steinberg, D. Top 10 algorithms in data mining. *Knowledge Information Systems 14*, 1 (December 2007), 1–37.

[107] Xu, J. An efficient multi-label support vector machine with a zero label. *Expert Systems with Applications 39*, 5 (2012), 4796–4804.

[108] Younes, Z.; Abdallah, F.; Denoeux, T.; Snoussi, H. A dependent multilabel classification method derived from the k-nearest neighbor rule. *EURASIP Journal on Advances in Signal Processing 2011*, Article ID 645964 (2011).

[109] Zaki, M. J.; Meira Jr, W. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[110] Zaragoza, J. H.; Sucar, L. E.; Morales, E. F.; Bielza, C.; Larrañaga, P. Bayesian chain classifiers for multidimensional classification. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence - Volume Three* (Barcelona, Catalonia, Spain, 2011), IJCAI'11, AAAI Press, pp. 2192–2197.

[111] Zhang, M.-L.; Zhou, Z.-H. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering 18* (2006), 1338–1351.

[112] Zhang, M.-L.; Zhou, Z.-H. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition 40*, 7 (July 2007), 2038–2048.

[113] Zhang, M.-L.; Zhou, Z.-H. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering 26*, 8 (August 2014), 1819–1837.

APPENDIX A – The Chi-Squared Test for Independence

As its name suggest, the chi-squared test for independence (χ^2) is a statistical test that can be employed to evaluate the hypothesis that two or more variables are independent [9, 47]. The test works as follows. First, the value of the chi-squared statistic between the variables under investigation is computed. Next, it is compared against a cutoff value, obtained from a chi-square table. If the computed value is higher than the cutoff value, the independence assumption is rejected at some significance level.

For binary variables (such as the label variables in a multi-label dataset), the value of the chi-squared statistics can be can be straightforwardly obtained with the use of a 2×2 contingency table, such as the one presented in Table A.1. In this table, consider that A and B are two label variables from a training dataset composed of N instances. Columns A and $\neg A$ correspond to instances that do and do not, respectively, contain label A. Similarly, rows B and $\neg B$ correspond to instances that do and do not contain label B. Thus, the cell labeled as $f_{A,B}$ indicates the number of instances that contain both A and B (a similar notation was used to represent the contents of the remainder cells).

Table A.1: Contingency table for two binary variables A and B

	$\boldsymbol{A}$	$\boldsymbol{\neg A}$	$\boldsymbol{\Sigma_{row}}$
$\boldsymbol{B}$	$f_{A,B}$	$f_{\neg A,B}$	f_B
$\boldsymbol{\neg B}$	$f_{A,\neg B}$	$f_{\neg A,\neg B}$	$f_{\neg B}$
$\boldsymbol{\Sigma_{col}}$	f_A	$f_{\neg A}$	N

The value of the chi-squared statistic can be computed with the formula presented in Equation A.1. For two binary variables, the cutoff value to reject the hypothesis at the 95% confidence level is equal to 3.84 [9].

$$\chi^2 = \frac{N \times ((f_{A,B} \times f_{\neg A,\neg B}) - (f_{\neg A,B} \times f_{A,\neg B}))^2}{(f_A \times f_B \times f_{\neg A} \times f_{\neg B})} \tag{A.1}$$

In this thesis, the χ^2 test was used in our proposed DepCC method (Chapter 4) to determine all pairs of independent labels in a given training set. Below, we present an example of the use of the χ^2 test considering the "flags" training dataset, described in Section 2.2. The example focus on the relationship between labels "yellow" and "white" (refer to the decision tree illustrated in Figure 4.2b, Section 4.2 for additional details). The contingency table for these two labels is presented in Table A.2.

Table A.2: Contingency table for the pair of labels *yellow* and *white* (FLAGS dataset)

	$\boldsymbol{yellow}$	$\boldsymbol{\neg yellow}$	$\boldsymbol{\Sigma_{row}}$
$\boldsymbol{white}$	33	62	95
$\boldsymbol{\neg white}$	28	6	34
$\boldsymbol{\Sigma_{col}}$	61	68	129

According to Equation A.1, the chi-squared statistic between "yellow" and "white" is given by:

$$\chi^2 = \frac{129 \times (33 \times 6 - 62 \times 28)^2}{(61 \times 95 \times 68 \times 34)} = 22.77.$$

Since 22.77 is well above the cutoff value of 3.95, we reject the independence assumption at the 95% confidence level. In other words, we can conclude that labels "yellow" and "white" are dependent.

APPENDIX B – The Wilcoxon Signed-Rank Test

Originally proposed in [104], the Wilcoxon signed-rank test is a non-parametric statistical hypothesis test that has been often used for comparing the performance of MLC methods [38]. The test does not assume a normal distribution and can be used to perform the comparison of two classifiers on multiple domains. In the case of the experiments in this thesis, each benchmark dataset corresponds to a data sample for the test (i.e., we have 10 data samples for the test), where the values being compared are the predictive performance measures of the classifiers (Accuracy, F-Measure, Exact Match and Hamming Loss). In all experiments, the significance of the results were verified with a confidence level of 95%.

Next, we describe the steps involved in the Wilcoxon signed-rank test with the aid of a toy example based on the hypothetical data presented in Table B.1. Consider that this table presents the Accuracy values obtained after the evaluation of two MLC methods M_1 and M_2 on 10 different datasets (D_1 to D_{10}).

Table B.1: Results of two methods M_1 and M_2 according to the Accuracy measure considering $n = 10$ datasets (Wilcoxon signed-rank test example)

Dataset	$\boldsymbol{M_1}$ **Accuracy**	$\boldsymbol{M_2}$ **Accuracy**
D_1	0.900	0.910
D_2	0.472	0.467
D_3	0.865	0.902
D_4	0.664	0.675
D_5	0.818	0.812
D_6	0.239	0.244
D_7	0.994	0.996
D_8	0.870	0.920
D_9	0.501	0.526
D_{10}	0.697	0.718

Assuming that the null hypothesis states that the two methods are not significantly different, the first step in the Wilcoxon test consists of computing the differences between each pair of scores (in our example, Accuracy values) and ranking the obtained values in increasing order of magnitude, ignoring the sign of result (i.e., if it is either negative or positive). In Table B.2, columns 4 and 5, we respectively present the obtained differences and rankings for our given toy example.

Table B.2: Results, differences and ranking values of two methods M_1 and M_2 according to the Accuracy measure considering $n = 10$ datasets (Wilcoxon signed-rank test example)

Dataset	$\boldsymbol{M_1}$ **Accuracy**	$\boldsymbol{M_2}$ **Accuracy**	**Difference** $(\boldsymbol{M_2 - M_1})$	**Rank**
D_1	0.900	0.910	0.010	5.0
D_2	0.472	0.467	-0.005	2.5
D_3	0.865	0.902	0.037	9.0
D_4	0.664	0.675	0.011	6.0
D_5	0.818	0.812	-0.006	4.0
D_6	0.239	0.244	0.005	2.5
D_7	0.994	0.996	0.002	1.0
D_8	0.870	0.920	0.050	10.0
D_9	0.501	0.526	0.025	8.0
D_{10}	0.697	0.718	0.021	7.0

Note that in the above example, the dataset with the highest absolute value of difference is D_8; thus it was assigned the highest rank value of 10. On the other hand, the lowest rank value of 1 was given to D_7, the dataset with the smallest absolute value of difference. It is also important to observe that, in the case of ties, as it occurs with the difference values of D_2 and D_6, average ranks must be assigned to each tied observation. Zero-valued differences (i.e., a situation in which the two methods have the same value for the performance metric under evaluation) can also be treated as a tie [51] or, alternatively, can be simply ignored [103].

Once the rankings have been computed, the subsequent step in the Wilcoxon test is to determine R_+ and R_-, which correspond to the sum of the rankings associated to the differences with a positive and a negative sign, respectively. Then, a T_{wilcox} statistic can be calculated as $min(R_+, R_-)$ (the smaller value of R_+ and R_-). In the example of Table B.2, we have:

$$R_+ = 5.0 + 9.0 + 6.0 + 2.5 + 1.0 + 10.0 + 8.0 + 7.0 = 48.5.$$

$$R_- = 2.5 + 4.0 = 6.5.$$

$T_{wilcox} = min(48.5, 6.5) = 6.5$.

When the number of datasets involved in the comparison is small ($n \leq 25$), the final step in the Wilcoxon test corresponds to directly use the table of critical values to check if the null hypothesis stating that the two classifiers perform equally well can be rejected. In Table B.3 (data partially reproduced from [51]), we present the critical values for the two-tailed test at a significance level of 5%, considering $n \leq 10$.

Table B.3: Critical values for the two-tailed Wilcoxon signed-rank test at a significance level of 5%. In this table, n corresponds to the number of datasets and $\alpha_{0.05}$ to the critical value.

n	$\boldsymbol{\alpha_{0.05}}$
6	0
7	2
8	3
9	5
10	8

According to [51], the null hypothesis should be rejected with a confidence level of 95% if T_{wilcox} is smaller than the critical value associated to n in Table B.3. In the given example, we have $n = 10$ and $T_{wilcox} = 6.5$. In this case, Table B.3 indicates the critical value of $\alpha_{0.05} = 8$. Since $T_{wilcox} = 6.5$ is below this critical value, we conclude that the null hyphotesis can actually be rejected. In other words, we can say that the two-tailed Wilcoxon signed-rank test indicates that M_2 is statistically superior to M_1 with a confidence level of 95%. This result can be presented using the following notation: $T_{wilcox}(10) = 6.5$. In this notation, the value between parenthesis (in this case, 10) specifies the number of datasets where the two methods M_1 and M_2 do *not* have the same value for the performance metric under evaluation.

In short, the logic behind the Wilcoxon signed-rank test can be summarized as follows [51]. Given a set of results obtained by two methods M_1 and M_2 on the same population (i.e., a collection of datasets), M_2 is better than M_1 if: (i) most of the results obtained by M_2 are greater than those obtained by M_1; and (ii) those results that are not greater, are smaller by only a small amount.

As a final remark, it is worth mentioning that our main motivations for using the Wilcoxon test in this work were twofold. The first consists in the fact that, with a few exceptions, most of the reported experiments were designed with the goal of performing paired comparisons between one of our proposed methods (PredCC, FreqCC, DepCC,

GACC or GA-PartCC) against the original CC method (or a CC-like method). The second is because the test is non-parametric and works well for small sample sizes [103], being thus appropriate for our experimental setup.