HYPER-RECTANGLE-BASED DISCRIMINATIVE DATA GENERALIZATION AND APPLICATIONS IN DATA MINING

by

Byron Ju Gao

B.Sc., Simon Fraser University, Fall 2002

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Spring 2007

APPROVAL

Name: Byron Ju Gao

Degree: Doctor of Philosophy

Title of thesis: Hyper-rectangle-based Discriminative Data Generalization and Applications in Data Mining

Examining Committee: Dr. Anoop Sarkar
Chair

Dr. Martin Ester, Senior Supervisor

Dr. Binay Bhattacharya, Supervisor

Dr. Oliver Schulte, SFU Examiner

Dr. Hui Xiong, External Examiner,
Rutgers, the State University of New Jersey

Date Approved: December 8, 2006

Abstract

The ultimate goal of data mining is to extract knowledge from massive data. Knowledge is ideally represented as human-comprehensible patterns from which end-users can gain intuitions and insights. Axis-parallel hyper-rectangles provide interpretable generalizations for multi-dimensional data points with numerical attributes. In this dissertation, we study the fundamental problem of rectangle-based discriminative data generalization in the context of several useful data mining applications: cluster description, rule learning, and Nearest Rectangle classification.

Clustering is one of the most important data mining tasks. However, most clustering methods output sets of points as clusters and do not generalize them into interpretable patterns. We perform a systematic study of cluster description, where we propose novel description formats leading to enhanced expressive power and introduce novel description problems specifying different trade-offs between interpretability and accuracy. We also present efficient heuristic algorithms for the introduced problems in the proposed formats.

If-then rules are known to be the most expressive and human-comprehensible representation of knowledge. Rectangles are essentially a special type of rules with all the attributional conditions specified whereas normal rules appear more compact. Decision rules can be used for both data classification and data description depending on whether the focus is on future data or existing data. For either scenario, smaller rule sets are desirable. We propose a novel rectangle-based and graph-based rule learning approach that finds rule sets with small cardinality.

We also consider Nearest Rectangle learning to explore the data classification capacity of generalized rectangles. We show that by enforcing the so-called "right of inference", Nearest Rectangle learning can potentially become an interpretable hybrid inductive learning method with competitive accuracy.

Keywords: discriminative generalization; hyper-rectangle; cluster description; Minimum Rule Set; Minimum Consistent Subset Cover; Nearest Rectangle learning

To the Author of the author
The One who does not need this to add to His glory.

"Thinking is not the way leading to answers;
Reasoning is not the way leading to truth."

— B. G.

Acknowledgments

Upon completion of this dissertation, the good feeling of accomplishment rings a bell of the joy of my childhood, when I finished off a project of a skyscraper with toy bricks. Not quite comparable but close.

In such a moment, I feel I am indebted a bit to each and every one in the world. We are not just neighbors! We are indeed siblings, sometimes referred to as enemies.

My supervisor and mentor, Dr. Martin Ester, is a blessing to my life. I have been learning from him as a student, also as a human being. Although he tries to treat his students fairly with equal and ample patience, I have managed to consume more.

I am truly grateful to those who have served in my supervisory and examining committees for their time and effort in reading my depth report, proposal, and dissertation and attending the exam and defence seminars. In particular, Dr. Binay Bhattacharya continuously gave me so many inspiring and constructive suggestions. Dr. Anoop Sarkar kindly compromised his schedule for mine. Dr. Hui Xiong was more than cooperative and responsive in making my defence take place as planned. Dr. Oliver Schulte spent hour after hour with me going through his detailed comments on my dissertation literally page by page. You can imagine how much patience and effort it must have taken him to make those comments, as his only compliment on my writing went to the carefully drawn figures.

The research supporting this dissertation also greatly benefited from open discussions with the faculty and student members in the Database and Data Mining Lab of Simon Fraser University: Dr. Ke Wang, Dr. Jian Pei, Wen Jin, Rong Ge, Flavia Moser, Gabor Melli, Richard Frank, Christine Cheng, Xiang Zhang, Fei Chen, Arber Xu, Yuelong Jiang, Senqiang Zhou, Rong She, and Benjamin Fung. It is an honor to be affiliated with such a talented group. Each of these names, at some point of time, left me feeling inferior.

A special group of people have become a constant source of joy, without which my PhD

journey would have become much dryer: Dr. Tiko Kameda, Dr. Binay Bhattacharya, Zengjian Hu, Yi Sun, and Zhong Zhang. My memory has 640K but the ping-pong table will always be set there unfolded, with enough surrounding space allowing double plays.

I owe so much to so many, I wish I could return the favor but keep the gratitude forever.

No special thanks necessary to my mother, who always smiles others' smiles, grieves others' griefs. Mom, why do I not see you in your own world?

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ultimate goal of data mining is to extract knowledge from massive data. Knowledge is ideally represented as human-comprehensible patterns from which end-users can gain intuitions and insights. The type of knowledge that we are interested in should appear as generalizations, instead of collections of ground instances. The representation of knowledge that we consider comprehensible should appeal to instinctive and intuitive awareness, perception, reasoning, and judgement of the majority of human beings. In this dissertation, we focus on the interpretability perspective of data mining and study the fundamental problem of rectangle-based discriminative data generalization in the context of several useful data mining applications: cluster description, rule learning, and Nearest Rectangle classification.

1.1 Motivation

Rapid advances in data collection, storage, and processing have provided an unprecedented challenge and opportunity for automated data exploration. Enormous amounts of data are being collected daily from stock trading, hospital information systems, computerized sales records and scientific projects such as the Human Genome Project, the Hubble Space Telescope, and the Geographical Information Systems [Mur98]. In addition to quantity, the variety of data also increases. The massive data available for exploration are from scientific researchers of diverse disciplines and business practitioners of various fields. Therefore, there is a commensurate need for robust, efficient, and versatile data generalization mechanisms to automatically acquire compact and concise knowledge that speaks to the majority of people's intuitive cognition.

Axis-parallel hyper-rectangles provide such self-explanatory generalizations for numerical attributes. For example, "3.80 $\leq$ *GPA* $\leq$ 4.33 and 0.1 $\leq$ *visual acuity* $\leq$ 0.5 and 0 $\leq$ *minutes in gym per week* $\leq$ 30" intuitively describes a group of "nerds". The notion of rectangle can be extended to accommodate categorical attributes by allowing internal disjunction of attribute values. However, rectangles have no generalization power on categorical attributes. In general, numerical and mixed data are relevant to the task of rectangle-based data generalization.

The study of rectangle-based generalization is of special importance for data mining since numerical data have been the "secondary" data type in knowledge-based systems. An attribute is called numerical if it takes values from an ordered set, and categorical if it takes values from a set not having a natural ordering [BFOS84]. Numerical attributes can have either discrete or continuous values. There are numerous application domains that heavily involve numerical data analyses such as pattern recognition, remote sensing and spectral analysis. However, interpretable discrimination and classification models have a "categorical tradition". For example, most decision tree induction and decision rule learning methods were originally developed for categorical attributes and the problem of incorporating continuous attributes is considered subsequently. Some methods adapt to the problem along with the induction process as seen in the extensions of ID3 [CT91] and C4.5 [Qui90]. Other methods have to rely on data discretization as pre-processing, which has consequently triggered a field of study for meaningfully discretizing continuous attributes [FI92, Ker92, dM93, MW94]. This "secondary" treatment of (continuous) numerical data does not directly address their intrinsic characteristics and would often lead to inferior results.

Rectangle-based discriminative generalization can be used for the purposes of both data description and data classification. The former focuses on existing data whereas the latter focuses on future data. While classification involves building models based on labeled data for the purpose of inferring class labels of unseen data, description can be considered as the process of reducing the volume of data by transforming it into a more compact and interpretable form while preserving accuracy [Mur98].

In this dissertation, we address three useful data mining applications in studying rectangle-based discriminative data generalization: cluster description [GE06a, GE06c], rule learning, and Nearest Rectangle classification [GE06b]. In the following, we give a brief introduction to these applications.

1.1.1 Cluster description

Clustering is one of the most important data mining tasks, grouping objects together into clusters that exhibit internal cohesion and external isolation. Unfortunately, most clustering methods simply represent clusters as sets of data points and do not generalize them into interpretable patterns. So far, the database and data mining literature lacks systematic studies of cluster description that transforms clusters into human-understandable patterns.

We study cluster description formats, problems, and algorithms following the hyper-rectangle approach. Using rectangle-based generalization is a standard data description method in database systems [AGDP98, LNW+02, MP03]. Such descriptive patterns are models with generalization capacity. They are also templates that can be used to generate objects. For example, as a practical application, rectangle-based descriptions can be used as search conditions in SELECT query statements to retrieve (generate) cluster contents, supporting query-based iterative mining [IM96] and interactive exploration of clusters.

To be understandable, cluster descriptions should appear short in length and simple in format. Sum of Rectangles (SOR) has been the canonical format for cluster descriptions in the database literature. However, this relatively restricted format may produce unnecessarily lengthy descriptions. We introduce two novel description formats, SOR^{-} and $kSOR^{\pm}$, leading to more expressive power yet still simple enough to be intuitively understandable.

Meanwhile, cluster descriptions should cover cluster contents accurately, which conflicts with the goal of minimizing description length. We introduce the novel Maximum Description Accuracy (MDA) problem with the objective of maximizing description accuracy at a given length. The optimal solutions to the MDA problems with different length specifications constitute the Pareto front for the bicriteria problem of optimizing description length and description accuracy. Previous research has only considered the Minimum Description Length (MDL) problem, which aims at finding some shortest perfect description that covers a cluster completely and exclusively. However, perfect descriptions can become very lengthy and hard to interpret for arbitrary shape clusters. The MDA problem allows trading accuracy for interpretability so that users can zoom in and out to view the clusters.

The description problems are NP-hard. We propose heuristic algorithms *Learn2Cover* and *DesTree* to approximate the Pareto front. We also present *FindClans* to transform the resulting descriptions, which are in the format of SOR or SOR^{-}, into shorter $kSOR^{\pm}$ descriptions with at least the same accuracy.

1.1.2 Rectangle-based rule learning

If-then rules are known to be the most expressive and human-comprehensible representation of knowledge. Rectangles are essentially a special type of rules with all the attributional conditions specified. Decision rules can be used for both data classification and data description. For either scenario, the minimality of rule sets is desirable. Our study of rule learning focuses on the Minimum Rule Set (MRS) problem, which finds a complete and consistent set of rules with the minimum cardinality.

We propose a novel rule learning algorithm *RGB* for the MRS problem. The algorithm is rectangle-based and graph-based with many unique and/or desirable properties. It first calls *CAG* to learn a complete and consistent set of possibly overlapping rectangle rules, from which compact rules are extracted with redundant conditions removed.

As the core procedure of *RGB*, *CAG* works with consistency graphs and assignment graphs to solve instances of the Minimum Consistent Subset Cover (MCSC) problem with anti-monotonic constraints. In the MCSC problem, we are given a ground set X and a constraint t, and the goal is to find the minimum number of consistent subsets that cover X, where a subset of X is consistent if it satisfies t. The constraint t is anti-monotonic if any subset of a consistent subset is also consistent. Such MCSC instances allow *CAG* to work in a pruned search space. The MRS problem, the Minimum Clique Partition problem, a dual problem of graph coloring on the complementary graph, and the traditional set covering problem are all MCSC instances with anti-monotonic constraints that can be solved by the generic *CAG*.

In a *consistency graph*, two vertices share an edge if and only if they form a consistent subset. In a bipartite *assignment graph*, each edge connects an element vertex (uncovered element) u to a condensed vertex (consistent subset) v indicating that u can be assigned to v while preserving its consistency. *CAG* starts by constructing a maximal optimal partial solution, represented by a maximal independent set of the consistency graph, then performs an example-driven specific-to-general search on a dynamically maintained assignment graph to learn a set of consistent subsets with small cardinality.

Although the MRS problem is defined on complete and consistent models, it can be easily extended to admit inconsistency, which is important from both data classification and data description points of view. Accordingly, our proposed generic *CAG* search framework can be adapted to the extended MRS problem with little effort.

1.1.3 Nearest Rectangle learning

Classification has a wide range of applications including scientific experiments, medical diagnosis, fraud detection, credit approval, and target marketing. Interpretable classifiers are desirable in general and can be crucial in certain applications such as expert systems that could be used for prediction, diagnosis, simulation, and training purposes.

Decision trees and decision rules are known to be interpretable classifiers providing insights to end-users. These interpretable classifiers are essentially rectangle-based. If we consider a closed attribute space, say the bounding box of the training instances, each leaf node of a decision tree as well as each rule in a rule set correspond to a hyper-rectangle geometrically. A major difference is that the rectangles in decision trees are mutually exclusive while in a rule set they are allowed to overlap.

Nearest Rectangle (*NR*) learning also produces interpretable classification models. In *NR* learning, training instances are generalized into axis-parallel hyper-rectangles and a query is classified according to the class of its nearest rectangle. However, existing *NR* learners suffer from relatively inferior accuracy. As hybrid learners, they do not gain an accuracy advantage while sacrificing classification time comparing to some other interpretable eager learners such as decision trees.

We seek to improve the accuracy of *NR* learning through controlling the generation of rectangles, so that each of them has the so-called "right of inference". Rectangles having the right of inference are compact, conservative, and good for making local decisions. We show that by enforcing the right of inference of rectangles, *NR* learning can potentially become an interpretable hybrid inductive learning method with competitive accuracy.

1.2 Contributions of Dissertation

This dissertation makes the following contributions:

- We focus on the interpretability perspective of data mining and study the fundamental problem of rectangle-based discriminative data generalization in the context of several useful data mining applications: cluster description, rule learning, and Nearest Rectangle classification.

- For the application of cluster description, we propose and analyze novel description formats that lead to enhanced expressive power. We introduce and motivate novel

description problems that allow trading accuracy for interpretability. We also present and evaluate efficient heuristic algorithms approximating the Pareto front for the bi-criteria problem of optimizing interpretability and accuracy.

- For the application of rule learning, we study the Minimum Rule Set problem and generalize it into the so-called Minimum Consistent Subset Cover (MCSC) problem. We propose *RGB*, a rectangle-based and graph-based rule learner that has many unique and/or desirable properties, whose core procedure *CAG* provides a generic search framework for solving MCSC instances with anti-monotonic constraints.

- For the application of Nearest Rectangle (*NR*) classification, we improve the accuracy performance of *NR* learning by enforcing the so-called right of inference of rectangles, so that *NR* learning can potentially become an interpretable hybrid inductive learning approach with competitive prediction accuracy.

1.3 Organization of Dissertation

The main body of the dissertation contains three relatively independent chapters addressing the three applications related to the problem of rectangle-based discriminative data generalization. Chapter 2 addresses rectangle-based cluster description. Chapter 3 addresses rectangle-based rule learning. Chapter 4 addresses Nearest Rectangle learning. Each of these three chapters has its own overview, background, and summary sections wrapping the core technical sections. Chapter 5 concludes the dissertation with an overall review and a general discussion on future work.

Chapter 2

Rectangle-based Cluster Description

In this chapter, we perform a systematic study of cluster description that generates rectangle-based patterns from clusters. We introduce and analyze novel description formats leading to enhanced expressive power. We motivate and define novel description problems specifying different trade-offs between interpretability and accuracy. We also present effective heuristic algorithms together with their empirical evaluations.

2.1 Overview

The ultimate goal of data mining is to discover useful knowledge, ideally represented as human-comprehensible patterns, in large databases. Yet not all data mining methods produce such readily understandable knowledge. Clustering is one of the major data mining tasks, grouping objects together into clusters that exhibit internal cohesion and external isolation. Unfortunately, most clustering methods simply represent clusters as sets of points and do not generalize them into patterns that provide interpretability, intuitions, and insights.

So far, the database and data mining literature lacks systematic studies of cluster description that transforms clusters into human-understandable patterns. For numerical data, hyper-rectangles generalize multi-dimensional points, and a standard approach in database systems is to describe a set of points with a set of isothetic rectangles [AGDP98, LNW+02].

Patterns are models with generalization capacity, as well as templates that can be used to make or to generate objects. The rectangle-based expressions are interpretable models. As another practical application, they can also be used as search conditions in SELECT query statements to retrieve (generate) cluster contents, supporting query-based iterative mining [IM96] and interactive exploration of clusters.

To be understandable, cluster descriptions should appear short in length and simple in format. Sum of Rectangles (SOR), simply taking the union of a set of rectangles, has been the canonical format for cluster descriptions in the database literature. However, this relatively restricted format may produce unnecessarily lengthy descriptions. We introduce two novel description formats leading to more expressive power yet still simple enough to be intuitively understandable. The SOR^- format describes a cluster as the difference of its bounding box and an SOR description of the non-cluster points within the box. The $kSOR^{\pm}$ format allows describing different parts of a cluster separately, using either SOR or SOR^- descriptions. We prove that the $kSOR^{\pm}$-based description language is equivalently expressive to the (most general) propositional language [MP03].

To give some examples, let us consider a cluster C with bounding box B_C. Let R_1, R_2, R_3, R_4 and R_5 be rectangles describing C and R_6, R_7 and R_8 be rectangles describing points in B_C but not in C. Then, $R_1 + R_2 + R_3 + R_4 + R_5$ is an SOR description for C while $B_C - (R_6 + R_7 + R_8)$ is an SOR^- description for C. A $kSOR^{\pm}$ description for C can be $R_3 + (B_{C'} - R7)$, where the first term is an SOR description, the second term is an SOR^- description, and $B_{C'}$ is a rectangle describing a part (subset) of C.

Meanwhile, cluster descriptions should cover cluster contents accurately, which conflicts with the goal of minimizing description length. The Pareto front for the bicriteria problem of optimizing description accuracy and length, as illustrated in Figure 2.3 (page 25), offers the best trade-offs between accuracy and interpretability for a given format. To solve the bicriteria problem, we introduce the novel Maximum Description Accuracy (MDA) problem with the objective of maximizing description accuracy at a given description length. The optimal solutions to the MDA problems with different length specifications up to a maximal length constitute the Pareto front. The maximal length to specify (20 in Figure 2.3, page 25) is determined by the optimal solution to the Minimum Description Length (MDL) problem, which aims at finding some shortest perfectly accurate description that covers a cluster completely and exclusively. Previous research only considered the MDL problem; however, perfectly accurate descriptions can become very lengthy and hard to interpret for arbitrary

shape clusters. The MDA problem allows trading accuracy for interpretability so that users can zoom in and out to view the clusters.

The description problems are NP-hard. We present heuristic algorithms *Learn2Cover* for the MDL problem to approximate the maximal length, and starting from which *DesTree* for the MDA problems to iteratively build the so-called description trees approximating the Pareto front. The resulting descriptions, in the format of SOR or SOR^-, can be transformed into shorter $kSOR^{\pm}$ descriptions with at least the same accuracy by *FindClans*, taking advantage of the exceeding expressive power of $kSOR^{\pm}$ descriptions.

Contributions

- We introduce and analyze novel description formats, SOR^- and $kSOR^{\pm}$, providing enhanced expressive power.
- We define and investigate a novel description problem, MDA, allowing trading accuracy for interpretability.
- We present and experimentally evaluate efficient and effective description heuristics, *Learn2Cover*, *DesTree* and *FindClans*, approximating the Pareto front.

Organization of the chapter

In Section 2.2, we review existing work related to cluster description. In Section 2.3 we introduce and analyze the description formats. In Section 2.4 we formalize the description problems. We present heuristic algorithms in Section 2.5, report empirical evaluations in Section 2.6, and summarize the chapter in Section 2.7.

2.2 Background

In this section, we review existing research related to the cluster description problems, Minimum Description Length (MDL) and Maximum Description Accuracy (MDA). The MDL problem is to find an expression in a given format with the minimum length that completely and exclusively describes a given cluster. The MDA problem is to find an expression in a given format with the maximum accuracy that describes a given cluster within a given length. The two problems will be formally defined and explained in Section 2.4.

For the MDA problem, the accuracy measure can be the F-measure, the *recall at fixed precision* (*of 1*) measure, and the *precision at fixed recall* (*of 1*) measure. The study of the MDA problem has practical importance since it can be used to specify alternative trade-offs between interpretability (description length) and accuracy. Unfortunately, to our knowledge, there is no existing research directly addressing the MDA problem. However, the MDL problem has been explicitly studied in the database and data mining literature. For both problems, indirectly related research exists in the theory, computational geometry, and machine learning literatures.

2.2.1 Directly related research

Currently in the database and data mining literature, there are only a few algorithms that explicitly work on description problems, i.e., *Greedy Growth* [AGDP98] and *BP* [LNW+02]. Both of them have exclusively focused on the MDL problem with *SOR* as the description format. In the following, we review these two algorithms and then discuss a theoretical study on concise descriptions.

Greedy Growth

[AGDP98] introduces the novel concept of subspace clustering for grid data, where a dataset is discretized into a grid of multi-dimensional cells and a cluster is defined as a set of connected dense cells. After a subspace cluster is discovered, *Greedy Growth* is called to generate a description that covers the cluster exactly with a set of maximal isothetic rectangles, where the objective is to minimize the number of rectangles.

The *Greedy Growth* heuristic consists of two phases: a growth phase and a removal phase. The growth phase greedily covers the cluster with a number of maximal rectangles, and the removal phase discards the redundant rectangles to generate a minimal cover. In the growth phase, a yet-uncovered dense cell is arbitrarily chosen to grow as much as possible along an arbitrarily chosen dimension and continue with other dimensions until a hyper-rectangle is obtained. Note that a rectangle must have all the attributional conditions specified. This process continues until the cluster is covered. The worse-case time complexity for the growth phase is $O(n^2)$, where n is the number of dense cells in the cluster. In the removal phase, the smallest redundant maximal rectangle is removed from the cover and the process continues until no rectangle can be removed. A rectangle is considered redundant if each cell it covers

is also covered by some other rectangle. The worse-case time complexity for the removal phase is also $O(n^2)$.

This formulation of cluster description is related to the problem of covering rectilinear polygons with axis-parallel rectangles [GJ79], which is a special case of the general set cover problem. Computing the optimal cover is known to be NP-hard even in 2-dimensional case [Mas79, RC87]. For general cases, there is no polynomial time approximation scheme [BD97]. The best approximation algorithm known for the special case of finding a cover of a 2-dimensional rectilinear polygon with no holes achieves a 2-approximation [Fra89]. Most studies in the computational geometry community, e.g., [KR99, LG97, MS00], are in 2-dimensional space. They are not directly applicable for finding cluster covers in general cases.

The problem of finding the optimal cover is also similar to the problem of constructive solid geometry formulae in solid-modeling [ZRL96]. It is related to the problem of covering marked boxes in a grid with rectangles in logic minimization (e.g., [HCO87]) as well. Some clustering algorithms in image analysis (e.g., [BR91, SB95, Wha83]) also find rectangular dense regions. In these domains, datasets are in low-dimensional spaces and the techniques used are computationally expensive for large datasets of high dimensionality. In general, the heuristic algorithms from the database community such as *Greedy Growth* are more scalable but without performance guarantees, and the ones from the computational geometry community can be very accurate – the best known approximation bound on the heuristics of rectangle cover problem is $O(\sqrt{logn})$ [KR99] – but can be very expensive, e.g., $O(n^6)$ [GL96].

Algorithm BP

[LNW+02] also studies grid data but generalizes the cluster description problem studied in [AGDP98] by allowing covering some "don't care" cells to reduce the cardinality of the set of rectangles. They formulate the description problem as summarizing a group of blue cells (the cluster) without covering the red cells while only a certain number of white cells (don't care cells) can be used.

Their proposed Algorithm *BP* is based on *Greedy Growth* and can be considered as an extension of *Greedy Growth*. Starting from an initial set of maximal rectangles returned by *Greedy Growth*, an exact cover of all the blue cells, *BP* greedily considers possible pair-wise merges of rectangles without covering the red cells and without exceeding the white budget. In picking up the pair of rectangles to merge, either a best fit or a first fit strategy can be

used. The best fit strategy is to find the pair of rectangles that minimizes the additional consumption of white cells. The first fit strategy is to find the first pair of regions based on a certain enumeration ordering of the elements in the rectangle pool. To facilitate the testing of red cell violation, all the red cells are built into a lean-tree index, which is a variant of X-trees [BKK96].

Greedy Growth and *BP* explicitly work on cluster description problems. However, despite the grid data limitation, they address the MDL problem solely while our focus is on the more useful and practical MDA problem. In addition, they only use the SOR format while we study and apply novel formats with enhanced expressive power.

A theoretical study

Similar to [AGDP98] (*Greedy Growth*) and [LNW+02] (*BP*), [MP03] is also motivated by database applications but with a focus on the theoretical formulation and analysis of concise descriptions. [MP03] formally defines the general MDL problem[1] for some given language L, denoted by L-MDL, and proves its NP-hardness. In short, the L-MDL problem is to search the shortest description in a given language L. The most general language they consider is the so-called propositional language $\mathcal{L}$, which consists of all expressions composed of symbols from the alphabet and operators standing for the usual set operations of union, intersection and set difference. They prove that the $\mathcal{L}$-MDL decision problem is NP-complete. They further define the Disjunctive Product Language $\mathcal{L}_P^+$, which corresponds to the language for SOR expressions, and proves the NP-completeness of the $\mathcal{L}_P^+$-MDL decision problem.

2.2.2 Indirectly related research

In the following, we review some indirectly related work in the theory community. We also show how decision trees can be related to the description problems.

Red blue set cover and maximum box

[CDKM00] studies the red blue set cover problem. Given a set of red and blue elements and a family which is a subset of the power set of the element set, find a subfamily of given cardinality that covers all the blue elements and the minimum number of red elements.

[1] Readers should not confuse the MDL problem with the Minimum Description Length principle [Ris78], which is an operational formalization of Occam's Razor that can be applied in inductive inference.

[EHL+02] studies the maximum box problem. Given two finite sets of points, find a hyper-rectangle that covers the maximum number of points from one designated set and none from the other. Both problems are NP-hard and related to the MDA problem under certain assumptions.

The red blue set cover problem is closely related to several combinatorial optimization problems, such as the Group Steiner problem [GKR98], directed Steiner problem [CCC+98], minimum label path [KW98], minimum monotone satisfying assignment [ABMP98, GM97], and symmetric label cover [DK99]. The problem is reducible to the classical set cover problem [Joh74a] and NP-hard.

For the maximal box problem, the NP-hardness proof is given in [EHL+02]. [LN03] and [Seg04] study approximation algorithms on a plane. [AHP05] studies a similar problem with "maximum disk" instead of "maximum box". [DbSS+05] studies a similar problem of using two, instead of one, either boxes or disks for the same objective.

The red blue set cover problem is related to the MDA problem with *precision at fixed recall of 1* as the accuracy measure, except that it is given an alphabet (family) and restricted to the SOR format. The maximum box problem corresponds to the MDA problem with *recall at fixed precision of 1* as the accuracy measures, except that it is limited to use one rectangle only. These accuracy measures are explained in § 2.4.1.

Traditional set covering problems

The research discussed above more or less stems from the classical minimum set cover and maximum coverage problems. The former attempts to select as few as possible subsets from a given family such that each element in any subset of the family is covered. The latter, a close relative, attempts to select k subsets from the family such that their union has the maximum cardinality. The simple greedy algorithm, iteratively picking the subset that covers the maximum number of uncovered elements, approximates the two NP-hard problems within $(1 + ln\, n)$ [Joh74a] and $(1 - \frac{1}{e})$ [Hoc97] respectively. The ratios are optimal unless NP is contained in quasi-polynomial time [Fei98]. The minimum set cover and maximum coverage problems are related to the MDL and MDA (with *recall at fixed precision of 1* as the accuracy measure) problems respectively except that they are given an alphabet (family) and restricted to the SOR format.

Decision trees

Axis-parallel decision trees [BFOS84] can be related to cluster description technically as they provide feasible solutions to the MDL and MDA problems even if with different objectives. If we consider a closed rectangular attribute space, the leaf nodes of a decision tree correspond to a set of isothetic rectangles forming a partition of the training data. Stipulating rectangles to be disjoint, decision tree methods address a partitioning problem while cluster description is essentially a covering problem allowing rectangles to overlap. Partitioning problems are "easier" than covering problems in the sense that they have a smaller search space. Algorithms for a partitioning problem usually work for the associated covering problem as well but typically generate larger covers.

In decision tree induction, the preference for shorter trees coincides with the preference for shorter description length in the MDL problem. As in the MDA problem, decision tree pruning allows trading accuracy (on training data) for shorter trees, and the technique can be applied to generate feasible solutions for the MDA problems.

2.3 Alphabets, Formats, and Languages

In this section, we study alphabets, formats, and languages for cluster descriptions in depth. We start with an introduction on preliminaries, and then discuss description alphabets. After that, we propose and analyze novel description formats SOR^{-} and $kSOR^{\pm}$, which is the focus of the section.

In cluster description, alphabets are not explicitly given as in the set covering problems. By assuming some alphabets, together with the given formats, we can precisely specify languages for description problems so that they can be solved by searching the associated languages. This study allows us to gain insight into description problems by connecting them to the traditional set covering problems. It also allows existing heuristics (or exhaustive search) to be applied on description problems for small datasets.

2.3.1 Preliminaries

Given a finite set of multi-dimensional points U as the universe, and a set of isothetic hyper-rectangles Σ as the alphabet, each of which is a symbol and subset of U containing points covered by the corresponding rectangle. A description format F allows certain Boolean set

expressions over the alphabet. All such expressions constitute the description language L with each expression $E \in L$ being a possible description for a given subset $C \subseteq U$. The vocabulary for E, denoted by V_E, is the set of symbols in E.

In the following, we summarize the notations used and to be used for easy lookup.

D: data space; $D = D_1 \times D_2 \times ... \times D_d$
U: data set; $U \subseteq D$
R: rectangle or the set of points it covers; $R \subseteq U$
Σ: alphabet; a set of symbols with each denoting a rectangle
V_E: vocabulary of expression E; set of symbols used in E
$||E||$: length of expression E; $||E|| = |V_E|$
B_S: bounding box for S, where S is a set of points or rectangles
C: set of points in a given cluster; $C \subseteq U$
C^-: set of points in B_C but not in C; $C^- = B_C - C$
$E_{F,\Sigma}$: expression in format F with vocabulary in Σ
$L_{F,\Sigma}$: language comprising all $E_{F,\Sigma}$ expressions

Note that R (E) is overloaded to denote a rectangle (expression) or the set of points it covers (describes). The distinction should be made clear by the context. The alphabet Σ is often left unspecified in $E_{F,\Sigma}$ and $L_{F,\Sigma}$ when assuming some default alphabet (to be discussed shortly). "$+$", "$\cdot$", "$-$" and "$\neg$" are used to denote Boolean set operators union, intersection, difference and complement.

Two descriptions E_1 and E_2 are equivalent, denoted by $E_1 = E_2$, if they cover the same set of points. Logical equivalence implies equivalence but not vice versa.

$||E||$ indicates the interpretability of E for a given format. There are two simple ways of defining $||E||$, absolute length and relative length. Absolute length is the total number of occurrences of symbols in E. Relative length is the cardinality of V_E. Neither alone captures the interpretability of E perfectly. The former overestimates the repeated symbols, while the latter underestimates the repeated symbols. The two converge if the repeated symbols are few, which we expect to be the case for cluster descriptions. For example, for descriptions in the SOR or SOR^- format (to be discussed in §2.3.3), absolute length equals relative length since each rectangle appears only once. However, we define $||E||$ to be the relative length of E for the ease of analysis of $kSOR^{\pm}$ descriptions. The conclusion in Theorem 2.1, claiming the powerfulness of $kSOR^{\pm}$ descriptions, is conditioned that the relative length is used.

Description accuracy is another important measure for the goodness of E, which we will discuss in more details in Section 2.4. For the use of this section, we conservatively define the "more accurate than" relationship for descriptions. A description for C is more accurate than the other if it covers more points from C and less points from C^-.

DEFINITION 2.1. (*more accurate than*) *Given E_1 and E_2 as descriptions for a cluster C, we say E_1 is more accurate than E_2, denoted by $E_1 \geq_{accu} E_2$, if $|E_1 \cdot C| \geq |E_2 \cdot C|$ and $|E_1 \cdot C^-| \leq |E_2 \cdot C^-|$.*

Certainly, $(E_1 \cdot C \supseteq E_2 \cdot C) \wedge (E_1 \cdot C^- \subseteq E_2 \cdot C^-) \Rightarrow E_1 \geq_{accu} E_2$, which means, if the points in C covered by E_1 is a superset of the points in C covered by E_2 and the points not in C covered by E_1 is a subset of the points not in C covered by E_2, then $E_1 \geq_{accu} E_2$. It is interesting to note that as pointed out by [Sch99], the condition $(E_1 \cdot C \supseteq E_2 \cdot C) \wedge (E_1 \cdot C^- \subseteq E_2 \cdot C^-)$ can be rewritten as a much neater (but less intuitive) equivalent form, $E_1 \triangle C \subseteq E_2 \triangle C$, where $\triangle$ denotes the symmetric difference of two sets.

A description problem, viewed as searching, is to search good descriptions that optimize some objective function in a given description language. A more general description language implies more expressive power. Language L_1 is more general than language L_2 if $L_1 \supseteq L_2$. To characterize expressive power more precisely, we define the "more expressive than" relationship for languages. A language is more expressive than the other if there is a shorter and more accurate description in it for any description from the other.

DEFINITION 2.2. (*more expressive than*) *Given two description languages L_1, L_2 and a cluster C, we say L_1 is more expressive than L_2, denoted by $L_1 \geq_{exp} L_2$, if for any description $E_2 \in L_2$, there exists some description $E_1 \in L_1$ with $||E_1|| \leq ||E_2||$ and $E_1 \geq_{accu} E_2$.*

Also, $L_1 =_{exp} L_2$ if $(L_1 \geq_{exp} L_2) \wedge (L_2 \geq_{exp} L_1)$.

A more expressive language is guaranteed to contain "better" expressions with respect to length and accuracy. Certainly, $L_1 \supseteq L_2 \Rightarrow L_1 \geq_{exp} L_2$, meaning that L_1 is more expressive than L_2 if L_1 is a superset of L_2. However, to restrict the search space, we do not want languages to be unnecessarily general. A languages is uniquely specified by a given alphabet and a given format. In the following discussions on description alphabets and formats, we consider the expressiveness as well as the conciseness of languages. For example, the proposed default alphabets are defined such that the associated languages are as specific as possible while being sufficiently general.

2.3.2 Description alphabets

Unlike the set cover problem, alphabet Σ is not explicitly given in cluster description. Assuming a given Σ, a description problem, MDA or MDL, can be considered as a search problem through a given language L for the optimal expression. We call such problems L-problems; in particular, L-MDA and L-MDL for the MDA and MDL problems respectively. The L-problems, to be detailed shortly in §2.4.2, are variants of the set cover problem.

An alphabet Σ is potentially an infinite set since for a (finite) set of points, there are an infinite number of covering rectangles with each being a candidate symbol. A simple finite alphabet can be defined as the set of bounding boxes for the subsets of B_C, i.e., $\Sigma_{most} = \{B_S \mid S \subseteq B_C\}$. This alphabet is finite since there is a unique bounding box for a set of points. Σ_{most} is the most general alphabet relevant to the task of describing C; however, it can be unnecessarily general for some L-problem with a mismatched format, meaning that some symbols in Σ_{most} would never be used in any feasible description of the given format. It is desirable to have some *most specific sufficiently general* alphabets.

Recall that an L-problem is a description problem with a given language L. Also recall that $L_{F,\Sigma}$ is a language comprising all expressions in format F with vocabulary in Σ. Let p denote a description problem and $f(L\text{-}p)$ denote the feasible region of an L-problem L-p. Certainly, $f(L\text{-}p) \subseteq L$. We say Σ is *sufficiently general* for $L_{F,\Sigma}$-p if and only if $L_{F,\Sigma} \geq_{exp} f(L_{F,\Sigma_{most}}\text{-}p)$, which intuitively means that Σ is not even inferior to the most general Σ_{most} if used in the L-p case. On top of being sufficiently general, Σ is *most specific* if removing any element from it, Σ would not be sufficiently general anymore. In the following, we define some alphabets with this desirable property, i.e., most specific sufficiently general with respect to the description formats we study in this dissertation.

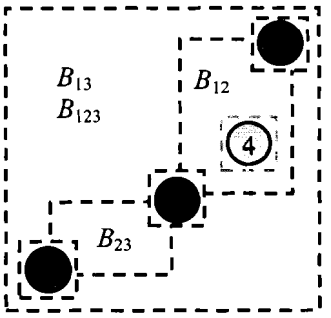
$$\Sigma_{pure} = \{B_S \mid B_S \cdot C^- = \emptyset \wedge S \subseteq C \wedge S \neq \emptyset\}$$

$$\Sigma^-_{pure} = \{B_S \mid B_S \cdot C = \emptyset \wedge S \subseteq C^- \wedge S \neq \emptyset\}$$

$$\Sigma_{mix} = \{B_S \mid S \subseteq C \wedge S \neq \emptyset\}$$

$$\Sigma^-_{mix} = \{B_S \mid S \subseteq C^- \wedge S \neq \emptyset\}$$

Σ_{pure} contains all the pure rectangles each covering some points in C and no points in C^-. Σ^-_{pure} contains all the pure rectangles each covering some points in C^- and no points in C. Symbols in Σ_{mix} and Σ^-_{mix}, however, can be mixed allowing points from C and C^-



$C = \{1, 2, 3\}$ $C^- = \{4\}$

$\Sigma_{pure} = \{B_1, B_2, B_3, B_{23}\}$
$\Sigma_{mix} = \{B_1, B_2, B_3, B_{12}, B_{13}, B_{23}\}$

Figure 2.1: Description alphabets.

to co-exist. Σ_{mix} contains all the rectangles each covering some points in C. Σ^-_{mix} contains all the rectangles each covering some points in C^-. Evidently, $\Sigma_{pure} \subseteq \Sigma_{mix} \subseteq \Sigma_{most}$ and $\Sigma^-_{pure} \subseteq \Sigma^-_{mix} \subseteq \Sigma_{most}$.

Note that multiple subsets may match to the same bounding box. Otherwise, the sizes of Σ_{mix} and Σ^-_{mix} may be precisely calculated, i.e., $|2^C|$ and $|2^{C^-}|$ respectively, where 2^C (2^{C^-}) denotes the power set of C (C^-). As shown in Figure 2.1, B_{13} and B_{123} are identical, where B_{13} is short for $B_{\{1,3\}}$ denoting the bounding box of $\{1, 3\}$. Alphabets Σ_{pure} and Σ_{mix} are illustrated in Figure 2.1. Note that B_{12} is not in Σ_{pure} since it covers point 4, which is in C^-. However, B_{12} is in Σ_{mix} as Σ_{mix} allows mixed rectangles as symbols.

Examples of some L-problems with matching alphabet and format are $L_{SOR,\Sigma_{pure}}$-MDL, $L_{SOR^-,\Sigma^-_{pure}}$-MDL, $L_{SOR,\Sigma_{mix}}$-MDA (with respect to the F-measure), $L_{SOR^-,\Sigma^-_{mix}}$-MDA (with respect to the F-measure), $L_{SOR,\Sigma_{pure}}$-MDA (with respect to the *recall at fixed precision of 1* accuracy measure), and $L_{SOR^-,\Sigma^-_{pure}}$-MDA (with respect to the *recall at fixed precision of 1* accuracy measure). In addition, $L_{kSOR^{\pm},\Sigma_k}$-MDL and $L_{kSOR^{\pm},\Sigma_k}$-MDA (with respect to the F-measure) are also such examples, where Σ_k is defined as follows:

$$\Sigma_k = \Sigma_{mix} + \Sigma^-_{pure}$$

We take $L_{SOR,\Sigma_{pure}}$-MDL as an example to explain why the alphabets and the formats are matching in the specified problems. The $L_{SOR,\Sigma_{pure}}$-MDL problem is to find the shortest perfect description in the SOR format for a given cluster, where the rectangles used in the descriptions can be chosen from a given alphabet Σ_{pure}. The alphabet Σ_{pure} is the most

specific sufficiently general alphabet with respect to the problem. It is sufficiently general since for any instance of the MDL description problem with SOR as the given format, the rectangles used in any feasible description are symbols contained in Σ_{pure}. It is also the most specific one among the sufficiently general alphabets since the removal of any symbol in it would result in the disappearance of the sufficiently general property. We omit explanations for other example problems, which can be interpreted in the same manner. The description formats and accuracy measures mentioned in the example problems will be discussed shortly.

Such a desirable most specific sufficiently general alphabet Σ is assumed given by default if it is left unspecified in $E_{F,\Sigma}$ or $L_{F,\Sigma}$. Although these default alphabets are made specific, they can still be prohibitively large (e.g., $O(|2^C|)$ and not scalable to real problems. Therefore, it is practically infeasible to generate Σ and apply existing set cover approximations on description problems in general.

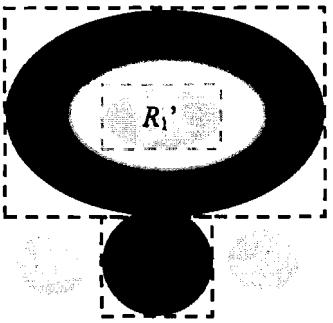
2.3.3 Description formats and languages

In the following, we introduce novel description formats that lead to enhanced expressive power yet remain intuitively comprehensible. We also measure the expressiveness of the resulting languages by comparing them to the (most general) propositional language.

We require descriptions to be interpretable. For descriptions to be interpretable, the description format has to have a simple and clean structure. Sum of Rectangles (SOR), denoting the union of a set of rectangles, serves this purpose well and has been the canonical format for cluster descriptions in the literature (e.g., [AGDP98, LNW+02]). For better interpretability, we also want descriptions to be as short as possible. Minimizing the description length of SOR descriptions has been the common description problem.

Nevertheless, there is a trade-off between our preferences for simpler formats and shorter description length. Simple formats such as SOR may restrict the search space too much leading to languages with low expressive power. On the other hand, if a description format allows arbitrary Boolean operations over a given alphabet, we certainly have the most general and expressive language containing the shortest descriptions, but such descriptions are likely hard to comprehend due to their complexity in format despite their succinctness in length. Moreover, not well-structured complex formats bring difficulties in manipulation of symbols and design of efficient and effective searching strategies.

Clearly, we require description languages with high expressive power yet in intuitively understandable formats. In the following, we explore several alternative description formats



SOR: 5 SOR^-: 4 $kSOR^{\pm}$: 3

$E_{kSOR^{\pm}}(C) = E_{SOR}(C_1) + E_{SOR^-}(C_2)$
$= B_{C_1} + (B_{C_2} - R_1')$

Figure 2.2: Description formats.

beyond SOR, in particular, SOR^- and $kSOR^{\pm}$. While SOR takes the form of $R_1 + R_2 + ... + R_l$, an SOR^- description for C specifies the set difference between B_C and an SOR description for C^-.

DEFINITION 2.3. *(SOR^- description) Given a cluster C, an SOR^- description for C, $E_{SOR^-}(C)$, is a Boolean expression in the form of $B_C - E_{SOR}(C^-)$, where $E_{SOR}(C^-)$ is an SOR description for C^-.*

In addition, an $SOR^{\pm}$ description for C, denoted by $E_{SOR^{\pm}}(C)$, is an expression in the form of either $E_{SOR}(C)$ or $E_{SOR^-}(C)$. Clearly, $L_{SOR^{\pm}} \supseteq L_{SOR}$ and $L_{SOR^{\pm}} \geq_{exp} L_{SOR}$. In describing a cluster C, SOR and SOR^- descriptions together nicely cover two situations where C is easier to describe or C^- is easier to describe. Different data distributions, which are usually not known in advance, favor different formats. In Figure 2.2, let us consider C_2 as a single cluster to be described with perfect accuracy. Certainly SOR^- descriptions are favored. We can see that the shortest $E_{SOR}(C_2)$ has length of 4 whereas the shortest $E_{SOR^-}(C_2)$ has length of 2.

SOR^- descriptions have a structure as simple and clean as SOR descriptions. In addition, the added B_C draws a big picture of cluster C and contributes to interpretability in a positive way. The two formats together also allow us to view C from two different angles. Note that the special rectangle B_C is required for the format, it is not included in the default alphabets neither counted in $||E||$ for simplicity.

$SOR^{\pm}$ descriptions generally serve well for the purpose of describing compact and distinctive clusters. Nevertheless, arbitrary shape clusters are not uncommon and for such applications, we may want to further increase the expressive power of languages by allowing less restrictive formats. For example, if some parts of cluster C favor SOR and some other parts favor SOR^{-}, then $SOR^{\pm}$ is too restrictive to consider different parts separately. Instead, it can only provide a global treatment for C. To overcome this disadvantage, we introduce $kSOR^{\pm}$ descriptions.

DEFINITION 2.4. ($kSOR^{\pm}$ *description*) *Given a cluster* C, *a* $kSOR^{\pm}$ *description for* C, $E_{kSOR^{\pm}}(C)$, *is a Boolean expression in the form of* $E_{SOR^{\pm}}(C_1)+E_{SOR^{\pm}}(C_2)+...+E_{SOR^{\pm}}(C_k)$, *where* $C_i \subseteq C\ \forall i \in \{1, 2, ..., k\}$.

Clearly, $kSOR^{\pm}$ descriptions generalize $SOR^{\pm}$ descriptions by allowing different parts of C to be described separately, and the latter one is a special case of the former one with $k = 1$. In Figure 2.2, C_1 favors SOR whereas C_2 favors SOR^{-}. The shortest $E_{SOR}(C)$ and $E_{SOR^{-}}(C)$ have length of 5 and 4 respectively. $kSOR^{\pm}$ is able to provide local treatments for C_1 and C_2 separately and the shortest $E_{kSOR^{\pm}}(C)$ has length of 3.

In many situations $kSOR^{\pm}$ is much more effective than other simpler formats. But how expressive is $L_{kSOR^{\pm}}$ precisely? In the following, we compare it with the *propositional language*, the most general description language we consider (as in [MP03]).

DEFINITION 2.5. (*propositional language*) *Given* Σ *as the alphabet,* $L_{P,\Sigma}$ *is the propositional language comprising expressions allowing the usual set operations of union, intersection and difference over* Σ.

THEOREM 2.1. $L_{kSOR^{\pm},\Sigma_k} =_{exp} L_{P,\Sigma_k}$

Proof. Since $L_{P,\Sigma_k} \supseteq L_{kSOR^{\pm},\Sigma_k} \Rightarrow L_{P,\Sigma_k} \geq_{exp} L_{kSOR^{\pm},\Sigma_k}$, we only need to prove that $L_{kSOR^{\pm},\Sigma_k} \geq_{exp} L_{P,\Sigma_k}$.

Consider any $E \in L_{P,\Sigma_k}$. Since E can be rewritten as E_{DNF} in disjunctive normal form with the same vocabulary, thus $||E|| = ||E_{DNF}||$ and $E = E_{DNF}$. Each disjunct in E_{DNF} is a conjunction of literals and each literal takes the form of R or $\neg R$ where $R \in \Sigma_k$. Consider any disjunct E_j in E_{DNF}. Since axis-parallel rectangles are intersection closed, E_j can be rewritten as E'_j, which takes one of the following three forms: (1) $E'_j = R_0$; (2) $E'_j = R_0 \cdot$

$\neg R_x \cdot \neg R_y \cdot ... \cdot \neg R_z$; (3) $E'_j = \neg R_x \cdot \neg R_y \cdot ... \cdot \neg R_z$. In all three cases $||E'_j|| \leq ||E_j||$ and $E'_j = E_j$.

For case 3, due to the generalized De Morgan's law, $E'_j = \neg R_x \cdot \neg R_y \cdot ... \cdot \neg R_z = \neg(R_x + R_y + ... + R_z) = B_C - (R_x + R_y + ... + R_z)$, which is an SOR^- description.

For case 1 and 2, we suppose $R_0 \in \Sigma_k$. Then for case 1, E'_j is an SOR description. For case 2, due to the generalized De Morgan's law, $E'_j = R_0 \cdot \neg(R_x + R_y + ... + R_z) = R_0 - (R_x + R_y + ... + R_z)$, which is an SOR^- description.

Then, E_j can be rewritten as an equivalent $SOR^\pm$ description with length $\leq ||E_j||$. We do this for every disjunct of E_{DNF}, then E can be rewritten as a $kSOR^\pm$ description $E' \in L_{kSOR^\pm,\Sigma_k}$ with $||E'|| \leq ||E||$ and $E' = E$, which implies $L_{kSOR^\pm,\Sigma_k} \geq_{exp} L_{P,\Sigma_k}$.

Note that for case 1 and 2, we have supposed $R_0 \in \Sigma_k$, which may not hold since Σ_k is not intersection closed. As a simple counter-example, the intersection of two bounding boxes may not be a bounding box. However, we note that for the two cases, the purpose of E'_j is to describe $R_0 \cdot C = C_0 \subseteq C$, thus $R'_0 = B_{C_0} \in \Sigma_k$. As expressions of length 1 describing C_0, $R'_0 \geq_{accu} R_0$ because $(R'_0 \cdot C_0 = R_0 \cdot C_0) \wedge (R'^-_0 \cdot C^-_0 \subseteq R_0 \cdot C^-_0)$. We replace R_0 with R'_0 in E'_j to get E''_j, then $E''_j \geq_{accu} E'_j$ and $||E''_j|| = ||E'_j||$. Then, E_j can be rewritten as a more accurate $SOR^\pm$ description with length $\leq ||E_j||$. We do this for every disjunct of E_{DNF}, then E can be rewritten as a $kSOR^\pm$ description $E'' \in L_{kSOR^\pm,\Sigma_k}$ with $||E''|| \leq ||E||$ and $E'' \geq_{accu} E$, which implies $L_{kSOR^\pm,\Sigma_k} \geq_{exp} L_{P,\Sigma_k}$.

Theorem 2.1 does not hold if description length $||E||$ is defined as the absolute length, in which case $E = (R_1 + R_2) - R_3$ in L_P has length of 3 but the equivalent $E' = (R_1 - R_3) + (R_2 - R_3)$ in $L_{kSOR^\pm}$ has length of 4. Nevertheless, the general conclusion persists, that is, $L_{kSOR^\pm}$ is a very expressive language close or equal to L_P.

Despite its exceptional expressive power, the $kSOR^\pm$ format is very simple and conceptually clear, allowing only one level of nesting as the SOR^- format. It is also well-structured to ease the design of searching strategies.

Assuming some given default alphabet, previous research studied cluster description as searching for the shortest expression in $L_{SOR,\Sigma_{pure}}$, the simplest and least expressive language we discussed in this section. We study the same problem but considering other more expressive languages $L_{SOR^\pm}$ and $L_{kSOR^\pm}$, and our main focus is on the problem of finding the best trade-offs between accuracy and interpretability, as to be introduced in the following section.

2.4 Description Problems

In this section, we formally define the cluster description problems, Maximum Description Accuracy (MDA) and Minimum Description Length (MDL). A description problem in general is to find a description for a cluster in a given format that optimizes some objective. Thus we start by discussing alternative objective measures that can be used for optimization in the description problems.

2.4.1 Objective measures

We want to describe a given cluster C with good interpretability and accuracy. Simple formats and shorter descriptions lead to improved interpretability. We have studied alternative description formats that are intuitively comprehensible. Within a given description format, description length is the proper objective measure for interpretability.

In addition to interpretability, the objective of minimizing description length can also be motivated from a "data compression" point of view. There are many situations when we need to retrieve the original cluster records, e.g., to send promotion brochures to a targeted class of customers, to perform statistical analysis, or in a query-based iterative mining environment as advocated by [IM96], to resume the mining process from stored temporary or partial results. Cluster descriptions provide a neat and standalone way of "storing and retrieving" cluster contents.

In DBMS systems, an isothetic rectangle can be specified by a Boolean search condition such as $1 \leq D_1 \leq 10 \wedge ... \wedge 5 \leq D_d \leq 50$. A cluster description is then a compound search condition for the points in the cluster, which can be used in the WHERE clause of a SELECT query statement to retrieve the cluster contents entirely. In this scenario, the cluster description process resembles encoding and the cluster retrieval process resembles decoding. The compression ratio for cluster description E can be roughly defined as $|E|$ / ($||E|| \times 2$), as each rectangle takes twice as much space as each point. The goal of maximizing compression ratio leads to the objective of minimizing description length. Meanwhile, shorter length also speeds up the retrieval process by saving condition checking time [MP03].

Accuracy is another important measure for the goodness of cluster descriptions. An accurate description should cover many points in the cluster and few points not in the cluster. To precisely characterize description accuracy, we borrow some notations from the information retrieval community [BYRN99] and define *recall* and *precision* for a description

E of cluster C. In the definitions, $|C|$ is the number of points in C, $|E|$ is the number of points covered by E, and $|E \cdot C|$ is the number of points in C that are covered by E.

$$recall = |E \cdot C| \ / \ |C|$$
$$precision = |E \cdot C| \ / \ |E|$$

If we only consider *recall*, the bounding box B_C could make a perfectly accurate description. If we only consider *precision*, any single point in C would do the same. The F-measure considers both *recall* and *precision* and is the harmonic mean of the two.

$$f = \frac{2 \times recall \times precision}{recall + precision}$$

A perfectly accurate description with $f = 1$ has $recall = 1$ and $precision = 1$. In a perfectly accurate SOR or SOR^- description, all rectangles are pure in the sense that they contain same-class points only.

The F-measure does not fit situations where users want to specify constraints on either *recall* or *precision*. We introduce two additional measures to provide this flexibility. The first is *recall at fixed precision*; often, we want to fix *precision* at 1. The second is *precision at fixed recall*; often, we want to fix *recall* at 1. The measures can be found useful in many situations. If we can afford to lose points in C much more than to include points in C^-, then we can choose *recall at fixed precision of 1* to sacrifice *recall* and protect *precision* as in the maximum box problem [EHL+02]. In the opposite situation, we can choose *precision at fixed recall of 1* as in the red blue set cover problem [CDKM00].

Recall the "more accurate than" relationship we defined in §2.3.1, which is rather conservative since if $E_1 \geq_{accu} E_2$, E_1 is more accurate than E_2 with respect to any accuracy measure we defined in this section.

2.4.2 MDA problem and MDL problem

Description length and accuracy are two conflicting objective measures that cannot be optimized simultaneously. The Pareto front for the bicriteria problem, as illustrated in Figure 2.3, offers the best trade-offs between accuracy and interpretability for a given format. To solve the bicriteria problem and obtain the best trade-offs, we introduce the novel Maximum Description Accuracy (MDA) problem.

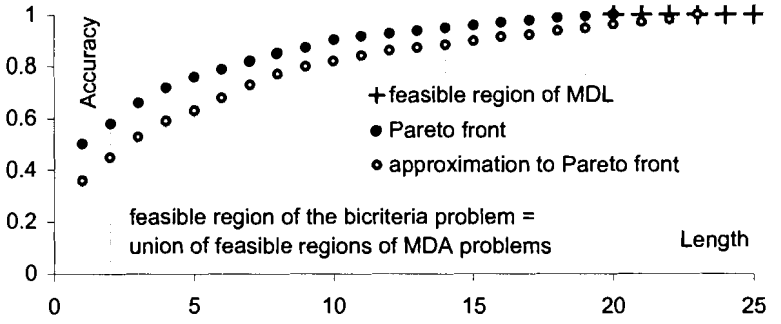


Figure 2.3: Accuracy vs. length.

DEFINITION 2.6. (*Maximum Description Accuracy problem*) *Given a cluster C, a description format F, an integer l, and an accuracy measure, find a Boolean expression E in format F with $||E|| \leq l$ such that the accuracy measure is maximized.*

The optimal solutions to the MDA problems with different length specifications up to a maximal length constitute the Pareto front. The vertical lines in Figure 2.3 illustrate the feasible regions of the MDA problems, whose union is the feasible region of the bicriteria problem. The maximal length is the length of some shortest description with perfect accuracy, which is 20 in Figure 2.3. It is pointless to specify larger lengths as the best accuracy has been achieved. To determine this maximal length, we define the Minimum Description Length (MDL) problem, whose objective is to find some shortest description that covers a given cluster completely ($E \cdot C = C$) and exclusively ($E \cdot C^- = \emptyset$), i.e., with $f = 1$.

DEFINITION 2.7. (*Minimum Description Length problem*) *Given a cluster C and a description format F, find a Boolean expression E in format F with minimum length such that* $(E \cdot C = C) \wedge (E \cdot C^- = \emptyset)$.

The optimal solution to the MDL problem gives the maximal length to specify in solving the MDA problems. The feasible region of the MDL problem is also illustrated in Figure 2.3. Previous research only considered the MDL problem with SOR as the description format. However, in practice, perfectly accurate descriptions can become lengthy and hard to interpret for arbitrary shape clusters. The MDA problem allows trading accuracy for interpretability so that users can zoom in and out to view the clusters. From a "data compression" point of view, description requiring perfect accuracy resembles lossless compression while description allowing lower accuracy resembles lossy compression.

Assuming some given default alphabets as previously discussed, the L-problems, which are easier than their counterparts, can be related to some variants of the set cover problem. In particular, $L_{SOR,\Sigma_{pure}}$-MDL corresponds to an instance of the minimum set cover problem, which is known to be NP-hard. Other L-MDL problems are harder in the sense that they have larger search spaces with more general languages.

Given the *recall at fixed precision of 1* accuracy measure, the $L_{SOR,\Sigma_{pure}}$-MDA problem corresponds to an instance of the maximum coverage problem, which is known to be NP-hard. Given the *precision at fixed recall of 1* accuracy measure, the $L_{SOR,\Sigma_{pure}}$-MDA problem corresponds to an instance of the red blue set cover problem, which is also NP-hard [CDKM00]. Given the F-measure, the decision version of the $L_{SOR,\Sigma_{pure}}$-MDA problem is reducible to the decision problem of either of the two. Other L-MDA problems are harder in the sense that they have larger search spaces.

As we have seen, the cluster description problems, with different format and accuracy measure specifications as discussed, are all NP-hard. We present efficient and effective heuristic algorithms for these problems in the next section.

2.5 Description Algorithms

In this section, we present three heuristic algorithms. *Learn2Cover* solves the MDL problem approximating the maximal length. Starting with the output of *Learn2Cover*, *DesTree* iteratively builds a so-called description tree for the MDA problems approximating the Pareto front. *FindClans* transforms the output descriptions from *DesTree* into shorter $kSOR^{\pm}$ descriptions without reducing the accuracy.

2.5.1 *Learn2Cover*

Given a cluster C, *Learn2Cover* generates a concise SOR description and a concise SOR^{-} description for C with the F-measure $f = 1$. For this purpose, it suffices to learn a set of pure rectangles $\Re$ covering C and a set of pure rectangles $\Re^{-}$ covering C^{-} completely. Recall that a pure rectangle contains points with the same class label. *Learn2Cover* is carefully designed such that $\Re$ and $\Re^{-}$ are learned simultaneously in a single run. In addition, the extra learning of $\Re^{-}$ does not come as a cost but rather a boost to the running time.

Sketch of *Learn2Cover*

To better explain the main ideas, we give the pseudocodes for the simplified *Learn2Cover* and its major procedure cover() in the following. We then explain how they work.

Algorithm: ***Learn2Cover***

Input: C: the cluster to be described.
Output: $\Re$ and $\Re^-$: $\Re$ is a set of rectangles covering the points in C completely and exclusively. $\Re^-$ is a set of rectangles covering the points in C^- completely and exclusively, where C^- denotes the set of points in the bounding box of C but not in C.

```
1 preprocessing(); // sort B_C along D_s, where B_C is the set of points in the bounding box of
  C and D_s is the chosen dimension
2 for each o_x ∈ B_C // processed in sorted order
3     if (o_x ∈ C)
4         cover(ℜ, o_x, ℜ⁻);
5     else
6         cover(ℜ⁻, o_x, ℜ);
7     end if
8 end for
```

In preprocessing(), the bounding box B_C is determined. The points in B_C are normalized against B_C and sorted along a selected dimension D_s. Ties are broken arbitrarily. At this moment we suppose there are no mixed ties involving points from different classes. In general, the choice of D_s does not have a significant impact if the data is not abnormally sparse, thus D_s can be arbitrarily chosen. Nevertheless, *Learn2Cover* offers an option to choose D_s with the maximum variance, which makes the algorithm more robust against some rare, tricky cases such as the presence of large mixed tie groups. *Learn2Cover* is deterministic once D_s is chosen.

Initially $\Re = \emptyset$ and $\Re^- = \emptyset$. Let o_x be the next point from B_C in the sorted order to be processed. Procedure cover($\Re$, o_x, $\Re^-$) or cover($\Re^-$, o_x, $\Re$) is called upon to process o_x depending on $o_x \in C$ or $o_x \in C^-$. The two situations are symmetric, thus we only provide the pseudocode for the former. The process iterates until all the points in B_C are processed. In each iteration, o_x is either covered by an existing rectangle or by a newly-created degenerate rectangle containing o_x itself.

Let us suppose $o_x \in C$, then procedure cover($\Re$, o_x, $\Re^-$) chooses a non-closed rectangle

Procedure: **cover**($\Re$, o_x, $\Re^-$)

Input: C, $\Re$, $\Re^-$ and o_x: C is the cluster to be described. $\Re$ and $\Re^-$ are the sets of rectangles covering the processed points in C and C^- respectively, where C^- is the set of points in the bounding box of C but not in C. o_x is the next point to be processed where $o_x \in C$.

Output: Updated $\Re$: $\Re$ is updated so that o_x is covered by some rectangle in $\Re$ which contains no points in C^-.

```
1  for each R ∈ ℜ⁻
2      if (cost(R, o_x) == 0)
3          close R;
4      end if
5  end for
6  for each R ∈ ℜ where R is not closed
7      if (cost(R, o_x) == 0)
8          extend R to cover o_x;
9          return;
10     end if
11 end for
12 for each (R ∈ ℜ) // processed in ascending order of cost
13     if (no violation against ℜ⁻)
14         expand R to cover o_x;
15         return;
16     end if
17 end for
18 insert(ℜ, R_new); // o_x not covered. R_new = o_x
```

$R \in \Re$ with the minimum *covering cost* with respect to o_x to expand and cover o_x such that the expanded rectangle does not cover any point covered by some rectangle in $\Re^-$, which causes a *covering violation*. Covering cost is used to measure the negative impact on the future covering process caused by the possible expansion of a rectangle in covering o_x. A covering violation occurs when a rectangle covers some point from the other class. A rectangle is *closed* if it cannot be expanded to cover any further point without causing a covering violation. We will further discuss covering cost and covering violation in more details right after this short overview. If there exists no such existing rectangle $R \in \Re$, a new rectangle R_{new} minimally covering o_x will be created and added to $\Re$ (line 18). In the beginning, the procedure checks if any rectangle $R \in \Re$ needs to be closed (lines 1, 2, 3, 4, 5). Then, for the actual covering of o_x, since violation checking can be expensive, the procedure always calculates $cost(R, o_x)$ first for each $R \in \Re$. $cost(R, o_x)$ is the covering

cost of rectangle R with respect to o_x. If there is a non-closed rectangle R with $cost(R, o_x) = 0$, we only need to extend R along D_s to cover o_x, in which case violation checking is unnecessary (lines 6, 7, 8, 9, 10, 11). Otherwise, rectangles are considered in ascending order of $cost(R, o_x)$ for violation checking. The first qualified rectangle will be used to cover o_x (lines 12, 13, 14, 15, 16, 17).

Covering cost and choice of rectangle

The behavior of *Learn2Cover* largely depends on how the covering cost $cost(R, o_x)$ is defined. Conceptually, $cost(R, o_x)$ is the cost of rectangle R in covering point o_x. This cost should estimate the negative impact on the future covering process caused by the expansion of R. In other words, it should estimate the reduction of the potential of existing rectangles in covering future points after o_x. Then for the choice of rectangle in covering o_x, we can simply choose the rectangle R that is available (non-closed), feasible (not incurring covering violations), and with the smallest $cost(R, o_x)$.

Intuitively and in a straightforward manner, the potential of the rectangle set in covering future points can be measured by their total volume. The bigger the total volume, the smaller the potential of the rectangles that can afford to expand and cover future points without incurring covering violations. Accordingly, $cost(R, o_x)$ can be defined as the increased volume of R in covering o_x. Nonetheless, there are more issues that need to be considered beyond this basic principle.

First, when calculating the increased volume for R, we should not consider D_s. Since points are sorted along D_s and processed in the sorted order, the extension of R along D_s is the distance it has to travel to cover further points after o_x. To keep R short on D_s does not help to keep its potential for future expansions. In Figure 2.4, if we considered D_s, R_1 would have the biggest increased volume and not be chosen to cover o_x. But this saved space would be part of the expanded R_1 in covering any point after o_x, and whether R_1 had been expanded already to cover o_x or not would not make a difference. This suggests that the increased volume of R_1 with respect to o_x should be 0, ignoring D_s in the calculation.

Second, if the expanded R has a length of 0 or close to 0 in any dimension, its volume and increased volume will be 0 or close to 0, which makes R a favorable choice. But R may have traveled far along some other dimensions to cover o_x and its expansion potential would thus be largely reduced. In Figure 2.4, both R_1 and R_3 require the same increased volume of 0 to cover o_x since R_1 has to be extended only along D_s and R_3 has a length of 0 in

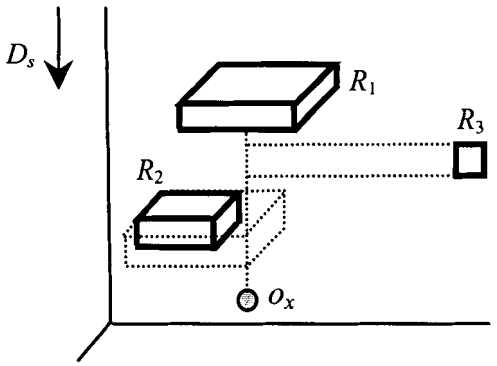


Figure 2.4: Choice of rectangle.

one of its dimensions. However, R_3 has to travel far along some dimensions other than D_s whereas R_1 does not. Moreover, R_2 does not have the increased volume of 0, but it seems not to be a worse choice than R_3 because o_x is closer to R_2 than R_3. Therefore, *cost* needs to fix the illusion of 0 increased volume and take into account the locality of rectangles.

In the following, we define $cost(R, o_x)$ with these issues in consideration. Let $l_j(R)$ denote the length of rectangle R along dimension D_j. Let R' denote the expanded R in covering o_x.

$$vol(R) = \prod_{j=1..d, j \neq s} l_j(R)$$

$$aveIncVol(R, o_x) = (vol(R') - vol(R))^{1/(d-1)}$$

$$dist(R, o_x) = (\sum_{j=1..d, j \neq s} |l_j(R') - l_j(R)|^2)^{1/2}$$

$$cost(R, o_x) = aveIncVol(R, o_x) + dist(R, o_x)$$

The dimension D_s is ignored if we project R and o_x onto the subspace $D \backslash D_s$. The term $vol(R)$ is the volume of the projected R. The term $aveIncVol$ can be viewed as the increased volume averaged on each dimension and *dist* is precisely the Euclidean distance from the projected o_x to the projected R. We define *cost* as the sum of $aveIncVol$ and *dist*, i.e., we assign equal weights to both components of *cost*. According to this definition of $cost(R, o_x)$, the choices in Figure 2.4 would be R_1, R_2, and R_3 in priority order.

Sometimes there are no good rectangles available, in which case forcing greedy expansions may deteriorate the overall performance. *Learn2Cover* provides an expansion control

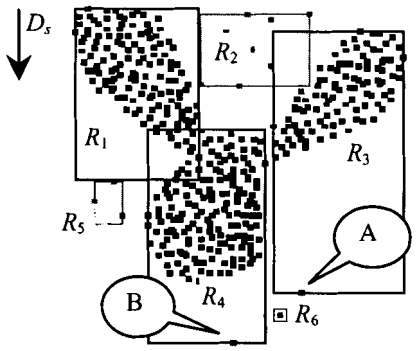


Figure 2.5: Demonstration run of *Learn2Cover*.

parameter to limit the maximum distance each dimension can travel at each expansion. Since data is normalized, the default choice of 0.5 means that each expansion cannot exceed half of the span of B_C in each dimension. The parameter is user-specified but not sensitive. Without expansion control, *Learn2Cover* works generally well, but it may help in cases of extremely sparse or tricky datasets.

Figure 2.5 is a real run of *Learn2Cover* on a toy dataset. Dark and light points denote points in C and C^- respectively. Rectangles are numbered in ascending order of their creation time. Note that on processing A, a better choice of R_3 was made while R_4 was also available. R_3 had *cost* of 0 with D_s ignored. If R_4 had been chosen to cover A, it would have been closed before covering B.

Covering violation and correctness

Learn2Cover is required to output pure rectangles only. That is, all the points contained in each rectangle should have the same class label. When a rectangle is expanded to cover the point being processed, a covering violation would occur if the expanded rectangle covers some point from the other class. Violation checking is necessary in processing each point.

The algorithm *BP* [LNW+02] considers all the points from the other class in violation checking. In *Learn2Cover*, since points are processed in the sorted order, the only points that could lead to violations in the expansion of $R_i \in \Re$ ($\Re^-$) are those currently processed points in $R_j \in \Re^-$ ($\Re$) that overlaps with the expanded R_i. Thus $\Re$ and $\Re^-$, the sets of rectangles to be learned, also exist to help each other in violation checking to dramatically reduce the number of points in consideration.

A simple and economic auxiliary data structure is maintained to avoid the possible performance deterioration in the presence of extremely dense and big rectangles. For each rectangle, we maintain a list of sub-rectangles with pre-defined capacity. The next sub-rectangle starts when the current one reaches the capacity. This way, violation checking will be always against some overlapping sub-rectangles of bounded size in an overlapping rectangle of the opposite class. The granularity of violation checking is thus reduced resulting in increased efficiency.

Learn2Cover outputs pure rectangle collections $\Re$ and $\Re^-$ covering each point in C and C^- respectively. The main procedure of *Learn2Cover* guarantees each point in B_C is processed. Now we examine procedure cover() to argue the correctness of *Learn2Cover*. From the definition of *cost*, $cost(R, o_x) = 0$ if and only if the projected o_x is in the projected R. In such case, if $R \in \Re$ ($\Re^-$) and $o_x \in C^-(C)$, R will not be able to cover any further point without covering o_x, which would cause a covering violation and R will thus be closed (line 3). If $R \in \Re$ ($\Re^-$) and $o_x \in C$ (C^-), R can be extended along D_s to cover o_x (line 8) without causing any covering violation. If there existed o' causing a violation, o' must have been processed before o_x with $cost(R, o') = 0$, which would have caused R to be closed. In line 14, R is expanded to cover o_x after violation checking. In line 18, R_{new} is a degenerate rectangle covering only one point o_x. Thus, in all the three ways to cover o_x, covering violations are avoided. Then upon completion of *Learn2Cover*, each $o_x \in B_C$ will be covered without incurring violations.

In the sketch of *Learn2Cover*, we assumed there were no mixed ties. A mixed tie group contains points of different classes that have the same coordinate value on the sorting dimension D_s. This case may happen and may happen rather frequently on grid data. Mixed tie points may cause covering violations to one another. *Learn2Cover* identifies mixed tie groups in the preprocessing() step, and then some extra checking is performed on processing o_x belonging to a mixed tie group.

2.5.2 *DesTree*

The MDA problem is to find a description E with a user-specified length l for cluster C maximizing a given accuracy measure. Our algorithm *DesTree* takes $\Re$ or $\Re^-$, the output from *Learn2Cover* whose cardinality approximates the maximal length to specify, and iteratively builds a so-called *description tree* that approximates the Pareto front illustrated in Figure 2.3.

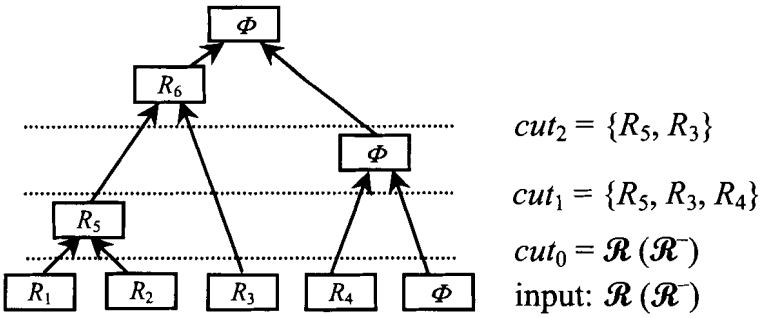


Figure 2.6: Description tree.

Description trees are tree structures resembling dendrograms to provide overviews on alternative trade-off descriptions of different lengths. Accordingly, *DesTree* resembles agglomerative hierarchical clustering to iteratively merge child nodes into parent nodes until a single node is left. Each node in a description tree represents a rectangle, and a normal merge operation produces a parent node that is the bounding box of its child nodes.

Each horizontal cut in the tree defines a set of rectangles. For the so-called C-description tree, a cut set constitutes the vocabulary for an SOR description of C. For the so-called C^--description tree, a cut set constitutes the vocabulary for an SOR description of C^- leading to an SOR^- description of C. The cardinality of a cut set, the description length, equals the number of links being cut. Each cut offers an alternative trade-off between description length and accuracy. The higher in the tree we cut, the shorter the length and the lower the accuracy.

Let us consider an SOR (SOR^-) description. Merging two rectangles into their bounding box may cause *precision* (*recall*) to decrease. Removing a rectangle may cause *recall* (*precision*) to decrease. Both operations trade the accuracy measure, say f, for shorter length and we want to consider both. To incorporate the removal operation in building description trees, we add a symbolic node, the empty set $\emptyset$, into the leaf nodes and define the *merge* operator as follows.

DEFINITION 2.8. (*merge*) R_i *merge* R_j =

(1) *bounding box of* R_i *and* R_j;	*if* $R_i \neq \emptyset$ *and* $R_j \neq \emptyset$
(2) $\emptyset$;	*otherwise*

DesTree is a greedy approach starting from the input leaf nodes, a set of pure rectangles $\Re$ or $\Re^-$ generated by *Learn2Cover*, building the tree in a bottom-up fashion. Pairwise *merge* operations are performed iteratively, and the merging criterion is the biggest resulting accuracy measure. The C-description tree and C^--description tree are built separately in the same fashion.

Figure 2.6 exemplifies a description tree. $R_1 \sim R_4$ are the input rectangles. R_1 and R_2 are chosen for the first merge to give R_5. The second merge of R_4 and $\emptyset$ results in the removal of R_4. R_6, the parent node of R_5 and R_3, merges with $\emptyset$ to give the symbolic root.

In Figure 2.6, the lowest cut cut_0 is $\Re$ or $\Re^-$. Each cut corresponds to an SOR or SOR^- description. Let us take cut_2 as an example. For a C-description tree, cut_2 corresponds to $E_{SOR}(C) = R_5 + R_3$. For a C^--description tree, it corresponds to $E_{SOR^-}(C) = B_C - (R_5 + R_3)$. Description trees are not necessarily binary. A merge could result in more rectangles fully contained in the parent rectangle. Nonetheless, the merging criterion discourages branchy trees and Figure 2.6 is a typical example.

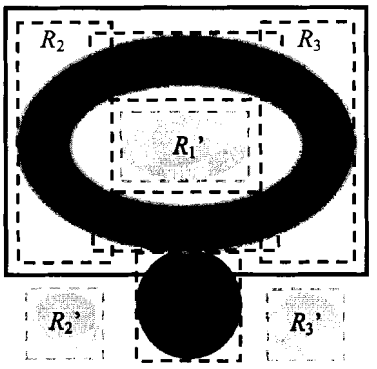
The merging process of *DesTree* can be simplified for some accuracy measures. Given *recall at fixed precision of 1*, for the C-description tree, only *merge* operation (2) (the removal operation) needs to be considered, and the root is always $\emptyset$. For the C^--description tree, only *merge* operation (1) (the normal merge operation) needs to be considered, and the root is always B_{C^-}. For both cases, *precision* is guaranteed to be 1 and *recall* reduces along the merging process.

Given *precision at fixed recall of 1*, for the C-description tree, only *merge* operation (1) needs to be considered, and the root is always B_C. For the C^--description tree, only *merge* operation (2) needs to be considered, and the root is always $\emptyset$. For both cases, *recall* is guaranteed to be 1 and *precision* reduces along the merging process.

It is easy to prove that the accuracy measure, *recall at fixed precision of 1* or *precision at fixed recall of 1*, reduces monotonically along the merging process in *DesTree*. With respect to the F-measure, though evident in experiments, it is non-trivial to construct a proof of the same property.

2.5.3 *FindClans*

FindClans takes as input a cut, essentially a set of rectangles, from a description tree representing an SOR or SOR^- description, outputs a $kSOR^{\pm}$ description with shorter length and equal or better accuracy.



$\mathcal{R} = \{R_1, R_2, R_3, R_4, R_5\}$
$\mathcal{R}^- = \{R_1', R_2', R_3'\}$

$T = \mathcal{R} = \{R_1, R_2, R_3, R_4, R_5\}$
$E_{SOR}(C) = R_1 + R_2 + R_3 + R_4 + R_5$
$N = \{R_2, R_3, R_4, R_5\} \subseteq T$
$N' = \{R_1'\}$
B_N = bounding box for N
$N.score = |N| - |N'| - 1 = 4 - 1 - 1 = 2$

$E_{kSOR^\pm}(C) = R_1 + (B_N - R_1')$
length reduction = $||E_{kSOR^\pm}(C)|| - ||E_{SOR}(C)|| = 2$

Figure 2.7: Clan helps in length reduction.

In the following, we use T to denote the input cut. We also use SOR_V to denote an SOR description with vocabulary V, e.g., $SOR_V = R_1 + R_2$ for $V = \{R_1, R_2\}$.

The algorithm *FindClans* is based on the concept of *clan*. Intuitively, a clan $N \subseteq T$ is a group of rectangles that dominate (densely populate) a local region, so that by replacing them as a whole, SOR_N can be rewritten as a shorter and more accurate SOR^- description for the set of points in the region that N tries to cover. We formally define clan and explain it in more details in the following.

DEFINITION 2.9. *(clan) Given T as a cut from a C (C^-) -description tree, $N \subseteq T$ is a clan if there exists N' such that $|N| - |N'| > 1$ and $B_N - SOR_{N'} \geq_{accu} SOR_N$ in describing $SOR_N \cdot C$ ($SOR_N \cdot C^-$), where B_N is the bounding box of N and N' a set of rectangles associated with N called the replacement of N. We also refer to $|N| - |N'| - 1$ as $N.score$.*

Note that the purpose of SOR_N is to describe $SOR_N \cdot C$ ($SOR_N \cdot C^-$) if T is from a C (C^-) -description tree. $N.score$ is the possible length reduction offered by a single clan N since SOR_N will be replaced by $B_N - SOR_{N'}$ and the newly introduced B_N will increase the length by 1. Two clans N_1 and N_2 are disjoint if $N_1 \cap N_2 = \emptyset$. For a set of mutually disjoint clans, *Clans*, the total length reduction is at least $\sum N_i.score$ where $N_i \in Clans$.

Figure 2.7 uses the example in Figure 2.2 and illustrates how a clan can help to rewrite an SOR description represented by T into a shorter $kSOR^\pm$ description.

Suppose we have found *Clans* for T and T is from a C-description tree. It is straightforward to rewrite the input SOR description $E_{SOR}(C) = SOR_T$ into a $kSOR^\pm$ description as illustrated in Figure 2.7. For each $N \in Clans$, we simply replace SOR_N in SOR_T by the shorter and more accurate $(B_N - SOR_{N'})$.

If T is from a C^--description tree, the input SOR^- description is $E_{SOR^-}(C) = B_C - SOR_T$. For each $N \in Clans$, we replace SOR_N in SOR_T by B_N and add back $SOR_{N'}$. As an example, let $E_{SOR^-}(C) = B_C - SOR_T = B_C - (R_1 + R_2 + R_3 + R_4 + R_5)$. Suppose we have a clan $N = \{R_2, R_3, R_4, R_5\}$ with replacement $N' = \{R'_1\}$, then $E_{kSOR^\pm}(C) = B_C - (R_1 + B_N) + R'_1$. The length reduction $= T.score = 4 - 1 - 1 = ||E_{SOR^-}(C)|| - ||E_{kSOR^\pm}(C)|| = 6 - 4 = 2$. It is easy to verify that after all such replacements, the resulting $E_{kSOR^\pm}(C)$ is shorter and more accurate than $E_{SOR}(C)$ or $E_{SOR^-}(C)$ with respect to any accuracy measure we discussed.

All we need now is to find *Clans*. To simplify the task, we define N', the replacement of N, to contain each rectangle $R \in \Re^-$ ($\Re$) overlapping with B_N if T is from a C (C^-) -description tree. Then, it is guaranteed that $B_N - SOR_{N'} \geq_{accu} SOR_N$ in describing $SOR_N \cdot C$ ($SOR_N \cdot C^-$) since N' contains pure rectangles completely covering $B_N \cdot C^-$ ($B_N \cdot C$). Thus given a candidate clan N, N' is uniquely determined and N is a clan if $|N| - |N'| > 1$.

Algorithm *FindClans*, as presented in the following, continues to call procedure findAClan() to find a clan candidate N in the updated T. If N is indeed a clan, it will be inserted it into *Clans* and T will be updated with the rectangles in N removed; otherwise, the algorithm terminates. Procedure findAClan() first checks each pair of rectangles in the input (updated) T and finds GN which is the pair with the highest score. N is used to keep track of the best stage of GN, which continues to grow greedily one more rectangle $R \in (T - GN)$ at a time resulting in the largest score increase, until no more rectangles are available. N is then returned as a clan candidate back to *FindClans*.

Algorithm: ***FindClans***

Input: T: a cut from a description tree denoting a set of rectangles.
Output: *Clans*: a set of clans.

```
1 Clans ← ∅;
2 N ← findAClan(T);
3 while (N.score ≥ 1)
4     insert(Clans, N);
5     T ← T − N;
6     N ← findAClan(T);
7 end while
```

Procedure: ***findAClan***(T)

Input: T: the updated T.
Output: N: a subset of T which is possibly a clan.

```
1 find GN; // consider each pair in T
2 N ← GN;
3 while (GN ⊆ T)
4     grow GN; // consider each R ∈ (T − GN)
5     if (GN.score > N.score)
6         N ← GN;
7     end if
8 end while
```

2.6 Empirical Results

In this section, we present our experimental results. We have evaluated and compared our methods against *CART* (Salford Systems, Version 5.0) and *BP* [LNW+02]. While decision tree classifiers, as argued in related work, can be applied to the MDA and MDL problems, *BP* addresses the MDL problem with respect to the *SOR* format only.

We implemented *Learn2Cover*, *DesTree*, *FindClans*, and *BP*. For *BP*, we also implemented *Greedy Growth* [AGDP98] and a synthetic grid data generator. To make our experiments reproducible, real datasets from the UCI repository [BM98] with numerical attributes and without missing values were used, where data records with the same class label were treated as a cluster. Note that in the broad sense, a cluster can be used to represent an arbitrary class of labeled data that require discriminative generalization. The notion of rectangle can be extended (but not implemented in this version) to tolerate categorical attributes, e.g., to use sets instead of intervals. Rectangles do not provide generalization on the categorical attributes.

2.6.1 Comparisons with *CART*

To approximate the Pareto front, our approach starts by applying *Learn2Cover* for the MDL problem, then *DesTree* for the MDA problems to build description trees. Decision tree classifiers provide feasible solutions to both the MDL and MDA problems. We compared *Learn2Cover* and *DesTree* with *CART* on UCI datasets. In each experiment we described one class of points C, considering all points from other classes within B_C, the bounding box for C, as C^-.

Learn2Cover vs. *CART*

In each experiment, B_C was fed to *Learn2Cover* and *CART*. The *CART* parameters were set such that a complete tree without misclassifications could be built. The entropy method was used for tree splits.

Table 2.1: *Learn2Cover* vs. *CART*

Dataset	*cls*	*dim*	$\|C\|$	$\|C^-\|$	*Learn2Cover*	*CART*	*FindClans*
wine	2	13	71	12	2 / **1**	5 / 5	–0
iono	g	34	225	11	3 / **2**	7 / 6	–0
iris	vir	4	50	18	**3** / 3	5 / 5	–0
blocks	4	10	88	1588	31 / **12**	35 / 20	–6
blocks	2	10	329	4974	26 / **19**	48 / 49	–7
yeast	cyt	8	463	783	**69** / 97	144 / 174	–13
yeast	nuc	8	429	939	**74** / 87	164 / 170	–16
abalone	I	8	1342	2549	**165** / 179	313 / 317	–37
abalone	F	8	1307	2693	**214** / 259	454 / 449	–62
abalone	M	8	1528	2626	**251** / 278	510 / 488	–54

For each dataset, Table 2.1 presents the class label, the dimensionality, the cardinalities of C and C^-, and the results. For *Learn2Cover* and *CART*, the term a/b denotes the cardinalities of the two sets of rectangles covering C and C^- respectively. For *CART*, a and b correspond to the numbers of leaf nodes of the two classes. For *Learn2Cover*, they correspond to $|\Re|$ and $|\Re^-|$. The smallest a or b is highlighted in bold font. We observe that, on average, *Learn2Cover* needs only half of the description length required by *CART*. Results from other UCI datasets as well as synthetic datasets are not presented. However, the above observation holds consistently through all the experiments.

The output of *Learn2Cover* was further fed into *FindClans* for additional length reduction. In Table 2.1, the term $-c$ denotes the additional reduction achieved by *FindClans* comparing to the shortest length (in bold font). The effectiveness of *FindClans* depends on the size and distribution of the input data. Bigger and more complex datasets are likely to exhibit more clans, leading to more length reduction. We observe that *FindClans* further reduces the shortest description length by about 20% on average and up to 50%.

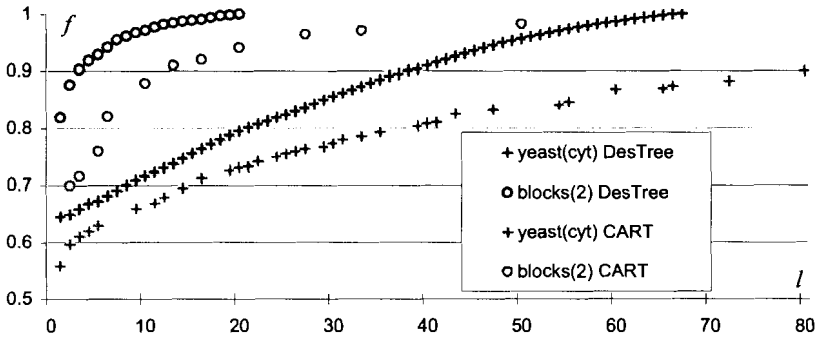


Figure 2.8: *DesTree* vs. *CART*.

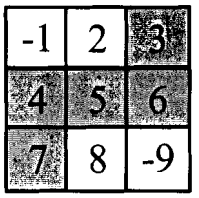
DesTree vs. *CART*

In each experiment, *DesTree* took as input $\Re$ or $\Re^-$ returned by *Learn2Cover*. For *CART*, the complete tree without misclassifications was pruned step by step. In each step, the misclassifications for both classes were counted to calculate the F-measure. The number of rectangles covering the target class C or C^- was also recorded as the description length.

Figure 2.8 demonstrates the results of *DesTree* (C-description tree) and *CART* (target class C), for clarity, on only two of the UCI datasets used in Table 2.1. The terms f and l denote the F-measure and description length respectively. As expected, for both methods, f decreases monotonically with decreasing l. However, the *DesTree* results clearly dominate the ones from *CART*. For each l, *DesTree* achieves a significantly higher f and for each f a significantly smaller l. Again, this observation holds consistently for all other experiments not presented, including ones on other UCI datasets and synthetic datasets, as well as C^--description tree experiments on both data types.

2.6.2 Comparisons with *BP*

While the focus of our study is on the MDA problem, *BP* addresses the MDL problem only. In this series of experiments, we compared *Learn2Cover* with *BP* on synthetic datasets as *BP* works on grid data only. Our data generator follows exactly what [LNW+02] does for *BP*. It takes as input the dimensionality, the number of intervals of each dimension, and the density. Dense cells are randomly generated in a grid with specified density, then one of them is randomly selected to grow a connected dense cell set as cluster C, the rest of the dense cells in B_C constitute C^-.



C: {3, 4, 5, 6, 7}; C^-: {-1, -9}; "don't care" cells: 2, 8

Greedy Growth: $R_{47} + R_{456} + R_{36}$

BP: the same as *Greedy Growth* since any pairwise merge of rectangles would cause a violation covering -1 or -9

Learn2Cover: $R_{2356} + R_{4578}$

Figure 2.9: A typical case in *BP*.

In our experiments with *BP*, we did not limit the number of "don't care" cells for use and allowed *BP* to find the best possible results. Since *BP* only generates one set of rectangles, we used $\Re$, a set of rectangles covering C generated by *Learn2Cover*, for the comparison. We studied the averaged percentage length reduction compared to *BP* for varying dataset size and dimensionality. We observed that in all cases *Learn2Cover* clearly outperformed *BP*, gaining 20% to 50% length reduction. As a general tendency, the reduction increased with increasing complexity of data. *FindClans* further improved the results, gaining an additional 25% length reduction on average.

One major reason *BP* does not perform well is that it starts with the maximal rectangles generated by *Greedy Growth*. Recall that *Greedy Growth* randomly picks a yet-uncovered dense cell to grow as much as possible along an arbitrarily chosen dimension and continue with other dimensions until a rectangle is obtained. The obtained rectangles tend to be long in some (different) dimension. As a consequence, even if with the help of "don't care" cells, it is difficult for *BP* to perform pairwise merges without incurring covering violations, as illustrated in Figure 2.9. In the figure, even if with the "don't care" cells 2 and 8, the rectangles generated by *Greedy Growth* cannot proceed with any pairwise merge without covering the "undesirable" cells -1 or -9. The "don't care" cells come too late to be helpful. On the contrary, *Learn2Cover* makes use of these "don't care" cells from the beginning and during the growing phase of rectangles. Consequently, dense cells in a local region are more likely to be covered by a single rectangle resulting in much improved results.

Runtime was not a major concern of this study. We did not integrate the X-tree index, on which *BP* relies to reduce the violation checking time. However, without indexing for both, we observed that *Learn2Cover* ran faster than *BP* by one to two orders of magnitude. Recall that *Learn2Cover* can significantly reduce the number of points in consideration for violation checking. If we assume constant number of rectangles, *Learn2Cover* has a worst case runtime of $O(|B_C|^2)$. If we further assume constant number of points in consideration for violation checking, *Learn2Cover* has a linear worse case runtime.

DesTree and *FindClans* have quadratic worst case runtime in the number of input rectangles, $O(|\Re|^2)$ or $O(|\Re^-|^2)$ for *DesTree* and $O(|T|^2)$ for *FindClans* respectively. In particular, for *DesTree*, the accuracy calculation results of all possible pairwise merges of rectangles can be reused in each iteration, recalculation is needed only for the resulting parent rectangle, which takes linear time. For *FindClans*, in each call of procedure findAClan(), the results from "find *theN*" (line 1) can be reused.

2.7 Summary

In this chapter, we systematically studied rectangle-based discriminative data generalization in the context of cluster description, which transforms clusters into patterns and provides the possibility of obtaining human-comprehensible knowledge from clusters. In particular, we introduced and analyzed novel description formats, SOR^- and $kSOR^\pm$, providing enhanced expressive power. We defined and studied description problem, Maximum Description Accuracy (MDA) and Minimum Description Length (MDL), allowing users to specify different trade-offs between interpretability and accuracy. We also presented heuristic algorithms, *Learn2Cover*, *DesTree*, and *FindClans*, for the MDL and MDA description problems, together with their experimental evaluations on real and synthetic datasets.

For future work, we consider further improvements of the description algorithms, in particular, *Learn2Cover* and *FindClans*. For *Learn2Cover*, by modeling the potential of rectangles in covering future points more properly, we may design better covering cost measures for the choice of rectangle. By incorporating indexing techniques in violation checking, we may significantly save the runtime for large datasets. By ordering the data points using more carefully designed strategies in pre-processing, the growing rectangles may remain more compact and the cardinalities of the rectangle sets may be consequently reduced. For *FindClans*, more sophisticated algorithms may be developed to fully materialize the

potential of the exceptional expressive power of $kSOR^{\pm}$ descriptions.

The concept of cluster in our study, in the narrow sense, is the output from clustering algorithms. Our study is motivated from and can find most applications in describing such clusters. In the broad sense, a cluster can be used to represent an arbitrary class of labeled data that require discriminative generalization.

Last but not least, cluster descriptions are patterns which can be stored as tuples in a relational table, so that a clustering and its associated clusters become queriable data mining objects. Therefore, this research can serve as a first step for integrating clustering into the framework of inductive databases [IM96], a paradigm for query-based "second-generation" database mining systems.

Chapter 3

Rectangle-based Rule Learning

In this chapter, we study the Minimum Rule Set (MRS) problem. Given a set of labeled examples, find a complete and consistent set of rules with the minimum cardinality. The completeness and consistency constraints require correct classifications of all the given examples. The MRS problem, with the goal of minimizing model complexity, can be motivated from both data classification and data description applications of decision rules.

For the MRS problem, we propose a novel rule learning algorithm *RGB* that is rectangle-based and graph-based with many unique and/or desirable properties. *RGB* first calls its core procedure *CAG* to simultaneously learn a set of possibly overlapping rectangle rules that cover a given set of multi-class examples completely and consistently. Then, compact rules are extracted from the learned rectangle rules with those "futile" conditions removed. *CAG* works with a consistency graph G_c and an assignment graph G_a, where G_c is constructed from the input data and G_a is a bipartite graph initially derived from G_c as an optimal partial solution and dynamically maintained during an example-driven specific-to-general search process.

In this chapter, we also establish that the MRS problem exhibits interesting connections to the well-known Minimum Clique Partition (MCP) problem, a dual problem of graph coloring on the complementary graph. The MRS and MCP problems, together with the traditional set covering problem, can be generalized into the so-called Minimum Consistent Subset Cover (MCSC) problem with consistent subsets differently defined. We show that our *CAG* search framework can serve as a generic approach for the purpose of solving MCSC instances with anti-monotonic constraints.

3.1 Overview

As the most human-comprehensible discrimination and classification tool, decision rules play a unique role in both research and application domains that cannot be replaced. With the increasing availability of data, it is more and more evident that the bottleneck in knowledge acquisition resides in obtaining rules directly from experts. Rule learning is thus well-motivated facilitating automated rule acquisition.

Decision rules can be used for the purposes of data classification and data description. The former focuses on unseen data whereas the latter focuses on existing data. For both applications, simpler models are preferred. By the widely applied principle of Occam's Razor [BEHW89, QR89, Ris89, MRA95], simpler models tend to have higher generalization accuracy. From the understandability point of view, simpler models provide more compact and concise descriptions that are easier for human beings to comprehend.

In this chapter, we study the Minimum Rule Set (MRS) problem. Given a set of labeled examples, find a *complete* and *consistent* set of rules with the minimum cardinality. The completeness and consistency constraints require that all the given examples are correctly classified. Such a rule set is also called *perfect*, analogous to perfect decision trees [Qui86].

By performing a graph transformation of the example set, we demonstrate some interesting connections between the MRS problem and the traditional Minimum Clique Partition (MCP) problem, a dual problem of graph coloring on the complementary graph. We show that the optimal solution to the MRS problem is lower-bounded by the optimal solution to the MCP problem, and the two problems are equivalent when the given examples are in 2-dimensional space. The MCP and MRS problems, together with the traditional set covering problem, can all be considered as instances of the generalized Minimum Consistent Subset Cover (MCSC) problem. The MCSC problem finds the minimum number of consistent subsets that cover a given ground set, where a subset is consistent if it satisfies a user-specified constraint. In addition to rule learning, the MCSC problem has many other data mining applications such as in converse k-clustering and frequent pattern summarization. A theoretical study of the MCSC problem provides insights to the design of *CAG*, a generic algorithm for solving MCSC instances. Our proposed rule learning algorithm *RGB* for the MRS problem has *CAG* as its core procedure.

RGB can seamlessly admit inconsistency during the induction process allowing efficient overfitting treatment without resorting to post-pruning. However, for a clearer presentation,

we do not generalize the MRS problem and discuss how *RGB* adapts to approximate models until the extension and discussion section at the end of the chapter.

Although most existing rule learners implicitly reduce the complexity of rule sets in the pursuit of good generalization accuracy, the minimality bias has not been "seriously enforced" [Hon97]. As evidence, in the experimental comparison with a popular rule learner *AQ21* [WMKP06], our algorithm *RGB* achieved about 40% reduction in both the total number of rules and total number of conditions.

RGB is rectangle-based and graph-based, which calls *CAG* to learn a set of rectangle rules first and then extracts compact rules with redundant conditions removed. The core procedure *CAG* starts by constructing a so-called *consistency graph* G_c from the input data. A bipartite *assignment graph* G_a is then derived from G_c as an optimal partial solution. *CAG* then performs an example-driven specific-to-general search on the dynamically maintained G_a for a perfect rectangle rule set of small cardinality. Rectangle rules are a special type of rules with all the attributes conditioned to specify the minimal ranges for the examples they cover. In other words, the antecedents of such rectangle rules are bounding boxes (rectangles) and least general generalizations for the vertices (examples) they cover. However, there can be many "futile" conditions whose removal will not negatively affect the consistency and coverage of the rectangle rules. Such conditions are dropped in *RGB* to obtain compact rules through a general-to-specific beam search. Our rectangle-based learning approach is rather conservative in the sense that it makes little steps in generalization, and thus the learning process can be finer-tuned and better-guided.

For the MRS problem, the vertices in G_c correspond to the input examples, and a pair of vertices share an edge in G_c if they have the same class label and their bounding box contains no vertices of other classes. For the generalized MCSC problem, a pair of vertices in G_c share an edge if the pair qualifies as a consistent subset according to a user-specified constraint. The vertex set of G_a consists of a set of *element vertices* each representing an uncovered element and a set of *condensed vertices* each representing a consistent subset containing the elements assigned to it. An edge in G_a connects an element vertex to a condensed vertex indicating that the element vertex can be assigned to the condensed vertex while maintaining its consistency.

G_a is initially a subgraph of G_c with a maximal independent set of G_c forming the condensed vertex set. It is dynamically maintained with the element vertices continuously assigned to the condensed vertices. New condensed vertices need to be created for isolated

element vertices with degree of 0. On completion, the element vertex set becomes empty and the condensed vertex set contains the learned consistent subsets (rectangle rules).

The information embedded in G_a allows the design of effective evaluation measures that can be utilized to guide the search. Our default *CAG* uses the *least degree first* strategy in choosing the next element vertex to be assigned since such vertices in G_a are most likely to become isolated. It also uses *weighted edge gain* for tie-breaking and selection of condensed vertex to take the chosen element vertex. The edges in G_a reflect the relational resources that the element vertices can make use of to join the condensed vertices. In general we want to keep as many edges as possible to avoid producing isolated element vertices. The initial G_a contains no such vertices since a maximal independent set is also a dominating set. However, each assignment would cause some edge loss (or gain in the creation of a new condensed vertex) and with the evolution of G_a, some element vertices would unfortunately become isolated. The gained or lost edges have different degrees of importance, thus weighted edge gain can be used to measure the influence of an assignment on G_a.

In addition to the rectangle-based and graph-based nature, *RGB* has some other unique and/or desirable properties. Unlike most existing methods, *RGB* does not follow a separate-and-conquer approach where rules are learned one after another. Instead, all rules are learned simultaneously in *RGB*. Unlike most existing bottom-up search methods that start the initial rule set with the example set, *RGB* starts with a subset containing a maximal number of examples such that no pair of them can co-exist in a single consistent rule, which constitute a maximal optimal partial solution. Unlike many existing methods that learn a set of rules for one class at a time, *RGB* naturally learns a set of rules for all classes simultaneously. Unlike many existing methods that can only learn either perfect models such as early members of the *AQ* family [Mic69, ML86], or approximate models such as *CN2* [CN89] and *RIPPER* [Coh95], *RGB* has the flexibility to learn both without resorting to post-pruning. Finally, although rectangle rules provide good generalization on numerical attributes, they can be adapted to categorical attributes as well. *RGB* can smoothly work on numerical, categorical or mixed datasets.

Our study has a focus on the minimality of models. Our experimental evaluations on *RGB* also used this criterion and demonstrated significant improvements over *AQ21*, the latest release of the famous *AQ* family, that can well represent most of the existing rule learners. The lower bound knowledge for the MRS problem helped to confirm the optimality of some results returned by *RGB*.

Contributions

- We study the Minimum Rule Set (MRS) problem emphasizing the minimality of model complexity. The study should benefit both data classification and data description applications of decision rules.
- We establish insightful connections between the MRS problem and the Minimum Clique Partition (MCP) problem. The two, together with the traditional set covering problem, are all instances of the generalized Minimum Consistent Subset Cover (MCSC) problem, which has many other practical data mining applications.
- We propose a novel rule learning algorithm *RGB* that is rectangle-based and graph-based with many unique and/or desirable properties. The core procedure *CAG* provides a generic search framework that can be applied to solve instances of the MCSC problem with anti-monotonic constraints.
- Our extensive experiments on UCI benchmark datasets demonstrated that *RGB* significantly outperformed a popular and representative rule learner *AQ21*, gaining about 40% reduction in both the number of rules and number of conditions.

Organization of the chapter

In Section 3.2, we define the Minimum Rule Set (MRS) problem and list some common notations used in the chapter. In Section 3.3, we motivate the MRS problem and discuss related topics. In Section 3.4, we establish insightful connections between the MRS problem and the traditional Minimum Clique Partition (MCP) problem. In Section 3.5, we generalize the MRS, MCP, and set covering problems into the Minimum Consistent Subset Cover (MCSC) problem and study its properties. In Section 3.6, we introduce the working principles of our *RGB* rule learning algorithm and explain its core procedure *CAG* in detail. In Section 3.7, we experimentally evaluate the performance of *RGB* on benchmark datasets. In Section 3.8, we discuss possible extensions and improvements of *RGB*. In Section 3.9, we summarize the chapter and introduce interesting directions for future work.

3.2 Problem Definition and Notations

In this section, we define and explain the Minimum Rule Set (MRS) problem that we study in this chapter. For simplicity, the problem is defined on perfect models, which can be easily generalized to admit approximate models as to be shown in later sections. We also introduce the common notations used in the chapter.

3.2.1 Minimum Rule Set problem

The problem we study in this chapter, Minimum Rule Set, is to find a disjunctive set of rules with the minimum cardinality that cover a given set of labeled examples completely and consistently. A feasible rule set must be *complete* and *consistent*. The completeness of a rule set requires that each example is covered by some rule in the rule set. The consistency of a rule set requires that each rule in the rule set is consistent. A rule is consistent if it classifies all examples it covers correctly. Certainly, two examples of different classes cannot be covered by the same rule that is consistent.

A rule covers an example if the antecedent of the rule covers the example, that is, the attribute values of the example satisfy the conditions specified in the antecedent of the rule. An example is correctly classified by a rule if the example is covered by the rule and the label of the example agrees with the consequent of the rule. In an if-then rule "if a then b", the if-part a is called the *antecedent* or *premise*, and the then-part b is called the *consequent* or *conclusion*. If-then rules are among the easiest representations of knowledge for human beings to understand intuitively. In the following, we formally define the Minimum Rule Set problem, we also define *rule set number* that is associated with the optimal solution to the problem.

DEFINITION 3.1. (*Minimum Rule Set*) *Given a set X of labeled examples of multiple classes, find a complete and consistent set R of rules for X, i.e., for each example $e \in X$, there exists some rule $r \in R$ that covers e and for each rule $r \in R$, all examples covered by r must have the same class label, such that $|R|$ is minimized.*

DEFINITION 3.2. (*rule set number*) *Given a set X of labeled examples of multiple classes, the rule set number for X, $\gamma(X)$, is the minimum $|R|$ where R is a complete and consistent set of rules for X.*

Our study is confined to propositional rules. The antecedent of each rule is a conjunction of one or more conditions that take the following form:

$$[< attr > \ < rel > \ < val >]$$

where $< attr >$ is an attribute, $< rel >$ is one of $\leq$, $\geq$, $=$, or $\neq$, and $< val >$ is an attribute value or an internal disjunction of attribute values.

For numerical attributes, $< rel >$ takes one of $\leq$ or $\geq$. $< val >$ is always an attribute value. An example of such a condition can be $[GPA \leq 1.5]$ where GPA is a numerical attribute. Such a numerical condition specifies a closed half-space with the corresponding dividing half-plane, $[GPA = 1.5]$, perpendicular to the axis defined by the attribute.

For categorical attributes, $< rel >$ takes one of $=$ or $\neq$. $< val >$ can be an attribute value or an internal disjunction of attribute values. An example of such a condition can be $[MOOD = good]$ or $[MOOD = good \vee bad]$ where $MOOD$ is a categorical attribute. By allowing $< val >$ to be an internal disjunction, the language we use is richer than in a typical rule learning program where $< val >$ is restricted to be an attribute value.

Note that in the literature, many rule learning methods focus on learning a set of rules, appearing concise, for one particular class at a time, and it is the sequentially learned rule sets that contain all the if-then rules. In our notion of rule set, those rule sets are flattened into one set of rules, and our algorithm learns all of them simultaneously. This difference in interpretation should not distort the nature of the problem we study due to the fact that under the completeness and consistency constraint, "our rule set" has its cardinality minimized if and only if each of "their rule sets" has its cardinality minimized. This can be proved easily by contradiction.

Also note that in our rule set, we do not allow a default rule established by applying negation as failure as seen in some methods, e.g., *RIPPER* [Coh95]. This casts no shadow on the significance of our study as the problem can be easily adjusted to that scenario. After removal of a particular class of rules, an optimal rule set remains optimal for the rest of the classes.

In a decision tree, each path from the root to a leaf node corresponds to a rule, which is a conjunction of conditions each being a test in the path. Thus decision trees can be used to extract a disjunctive set of rules [Qui93a]. However, as a divide and conquer approach, decision trees learn a set of rules that are mutually exclusive. Rule learners, in the conventional sense, allow rules to overlap. We follow the convention and view decision trees and

decision rules as different classification models. However, due to the many similarities they share, we compare the two models side by side on many occasions.

Both decision trees and decision rules are considered as interpretable classification models with axis-parallel decision boundaries.[1] If we consider a closed attribute space, the bounding box of the given labeled examples, each leaf node in a decision tree or each rule in a rule set corresponds to an axis-parallel rectangle rule. While such rectangle rules in a rule set are allowed to overlap, the ones in a decision tree have to be disjoint and organized to form a feasible structure resulting from a recursive partitioning process.

A decision tree that correctly classifies all the training examples is called a *perfect tree* [Qui86, Qui90]. Similarly, we can use *perfect rule set* to mean a rule set that is complete and consistent. The optimal decision tree problem has been studied to induct a perfect tree with some objective optimized (e.g., [MM73, KK76]). While various objectives have been investigated, a common one is to minimize the complexity of trees, which can be measured by the number of leaf nodes. The problem we study, MRS, can be accordingly referred to as an optimal rule set problem with an objective of the same kind.

Another popular measure for tree complexity is the total number of tests (internal nodes), which corresponds to the total number of conditions in a rule set. From the perspective of obtaining an optimal solution, the two complexity measures may lead to totally different problems for either case, optimal decision tree or optimal rule set, that requires totally different techniques to solve. From the perspective of obtaining a heuristic solution, the two complexity measures are more or less consistent. In the rule set case, fewer number of rules usually lead to fewer number of conditions, as shown in our experimental study.

The MRS problem thus defined is consistent with the literature. Most rule learners implicitly reduce the complexity of rule sets in order to achieve good generalization accuracy assuming the validity of Occam's Razor. However, as pointed out by [Hon97], the minimality of rule sets has not been a "seriously enforced bias". This claim will be further justified by our analyses and experiments in later sections of the chapter.

In addition to data classification, as an interpretable model, rules can also be used for the purpose of data description as decision trees [Mur98]. Data description seeks to

[1] In contrast to *axis-parallel decision trees* that test a single attribute at each internal node, there are studies on *oblique decision trees* that test a linear combination of the attributes. By default, decision trees are axis-parallel and they are such called because each test is equivalent to an axis-parallel hyperplane in the attribute space.

reduce the volume of data by transforming it into a more compact and interpretable form while preserving accuracy. To maximize interpretability, it is also necessary to emphasize minimizing the complexity of descriptive models.

Chapter 2 provides an example of such application where data clusters demand discriminative description. In Section 2.4, we defined the Minimum Description Length (MDL) problem, where we are given a cluster and a format and asked to find a shortest description in the given format covering the cluster consistently and completely. While related, there are several differences between the two problems. First, the MRS problem is a multi-class problem whereas the MDL problem is a two-class problem. Second, the MRS problem asks for a disjunctive set of rules, corresponding to the SOR format for cluster description, while the MDL problem is given a format that can be some other kind. Third, the MRS problem is defined on rules instead of rectangle rules as in the MDL problem. Although our rule learning algorithm RGB does make use of the observation that each rule corresponds to a rectangle, a normal rule appear more compact and concise without all the attributional conditions specified as in rectangle rules.

3.2.2 Notations

MRS: Minimum Rule Set problem
MCP: Minimum Clique Partition problem
MCSC: Minimum Consistent Subset Cover problem
X: example set for the MRS problem or ground set for the MCSC problem
(X, t): MCSC instance with X as the ground set and t as the given constraint
R: rule set
$G = (V, E)$: simple graph with vertex set V and edge set E
$\overline{G}$: complementary graph of G
$\chi(G)$: chromatic number of G
$\overline{\chi}(G)$: clique partition number of G
$\gamma(X)$: rule set number for the MRS problem with X as the example set
$\gamma(X, t)$: consistent subset cover number for the MCSC instance (X, t)
$G_c = (V_c, E_c)$: consistency graph with vertex set V_c and edge set E_c
$G_a = (U_a \cup V_a, E_a)$: bipartite assignment graph with vertex set $U_a \cup V_a$ and edge set E_a, where U_a is the element vertex set and V_a is the condensed vertex set
IS: (maximal) independent set

3.3 Background

In this section, we discuss the research topics centered around the Minimum Rule Set (MRS) problem. We motivate interpretable classifiers and introduce the rule extraction application in particular. We compare the two interpretable models of decision rules and decision trees. We also explain the benefits of allowing both perfect and approximate models, discuss the merits of minimality, and review research on optimal models and existing rule learning techniques. In addition, we compare the separate-and-conquer rule learning paradigm with the well-known greedy set covering algorithm and gain insight into the performance of existing rule learners.

3.3.1 Interpretable classifiers

Decision trees and decision rules are known as interpretable classification models that are able to provide insights and intuitions to end users. This interpretability is desirable in general for data mining tasks, and can be crucial in certain applications such as expert systems that could be used for prediction, diagnosis, simulation, and training purposes.

Other classification models, such as Support Vector Machines (SVMs), Artificial Neural Networks (ANNs) and k-Nearest Neighbor (kNN) classifiers, are possibly more accurate in certain occasions but known to have the drawback of working in a "black box" [HV06]. We feel that the accuracy advantage of these black box models are moderate especially in real applications instead of scientific research. Real datasets are noisy. The main source of error in classification resides in data instead of methods. As a common observation in benchmark experiments, good (bad) data generally lead to high (low) prediction accuracy for all the comparison participants. Lack of training examples and noise in the examples and features are predominately responsible for low accuracy performance regardless of the actual method used. Currently, there exists no classification method that has proved to be clearly advantageous in identifying relevant examples and relevant features from data to obtain significant accuracy gain.

Even when the accuracy advantage of black box classifiers is clear and needed, interpretable models can still be helpful whenever insights and intuitions are also desirable. They can be applied to "transparentize" the black box by using the artificial training examples coming out of it, an approach known as rule extraction [HV06]. This usage of interpretable models is for the purpose of data description, instead of classification.

In addition to data classification, interpretable models can be used to describe (labeled) data. As mentioned in [Mur98], "decision trees have proved to be valuable tools for both the description and classification of data." The objective of *data description* is to reduce the volume of data by transforming it into an interpretable and compact form that preserves the essential characteristics and provides an accurate summary. As also mentioned in [Han81], "decision trees have been used for discrimination as well as classification." Discrimination is the process of deriving classification rules from samples of classified objects, and classification is applying the rules to new objects of unknown class. The key difference between data description and data classification is that the former focuses on existing data whereas the latter focuses on unseen data. Certainly, decision rules can also provide such descriptive functionality.

Cluster description we studied in Chapter 2 is an example of data description application, where data clusters require interpretable discriminative generalization . As the entire dissertation has a focus on the interpretability perspective of data mining, it is worthwhile to introduce in more detail another data description application, rule extraction, as a supplement to cluster description.

3.3.2 Rule extraction application

Rule extraction has indeed formed a popular research community. The techniques are used to improve the comprehensibility of predictive models such as ANNs or SVMs. Given an opaque predictive model and the data on which it was trained, the objective of rule extraction is to produce a description of the predictive model's hypothesis that is understandable yet closely approximates the predictive model's behavior [HV06].

As pointed out in [HV06], the application of newer machine learning techniques such as ANNs or SVMs remains largely restricted (to research environments) due to their lack of explanation capability required in some application areas such as medical diagnosis or credit scoring. Rule extraction is used to provide meaningful descriptions for the underlying black box models, so rule extraction techniques should ideally mimic the behavior of the black boxes as closely as possible while maximizing the interpretability of the descriptions. These two conflicting goals are similar to what we have discussed in Chapter 2, i.e., higher description accuracy and shorter description length. In fact, rule extraction is a data description application focusing on existing data, where the data to describe are the artificial examples generated by the black boxes.

There are various types of rules that have been used for rule extraction. Propositional if-then rules, M-of-N rules, oblique rules, equation rules, fuzzy rules, and first order logic. Among them, propositional rules are the most widely used, and natural propositional rule learners can certainly be applied for the purpose of rule extraction, although specialized techniques exist [HV06].

In situations where performance is a crucial issue, the opaque models can be the preferred choice for implementing the decision process while rule extraction can be adopted to verify the knowledge encoded in these models. This knowledge verification is crucial in many applications and sometimes even legally required. [HV06] provides a convincing real life scenario in the practice of credit scoring [MBGV05] and interested readers can refer to the original paper for more details. As another example in the medical domain, users are reluctant to use black box computer-aided diagnosis (CAD) systems that can influence patient treatment [FSR05]. The ability to generate even limited explanations is essential for user acceptance of such systems.

Automatic knowledge acquisition, induction of scientific theories and study of the generalization behavior of an underlying model are several other reasons than the explanation capacity that underline the importance of rule extraction [ADT95].

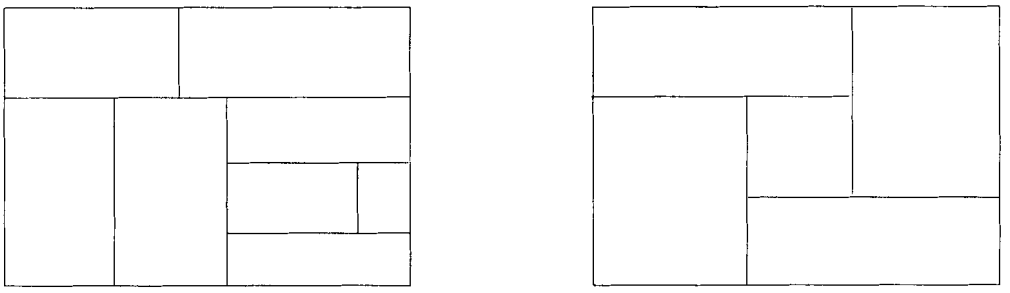
We note that the above arguments motivating rule extraction can also serve the purpose of motivating interpretable data mining in general, which is the center of our study in the dissertation.

Rule extraction techniques for ANN are studied in [TS93, CS94, Cra96, CS96, Thr95, SL97, SLZ02, RDP+04] and surveyed in [ADT95, Dar01]. Rule extraction techniques for SVM can be found in [NAC02, Che04, BD05, MBGV05, FSR05]. We do not further discuss the details of these techniques.

3.3.3 Rules vs. trees

Despite the popularity of decision tree learning partly due to its efficiency, many researchers consider decision rules more advantageous than decision trees. [Fur99] argues that decision trees are often quite complex and hard to understand. [Qui93a] has noted that even pruned decision trees may be too cumbersome, complex, and inscrutable to provide insight into the domain at hand and has consequently devised procedures for simplifying decision trees into pruned production rule sets [Qui87, Qui93a].

Rule sets obtained from trees can still be more complex than those directly learned rule



(a) allowed by decision trees (b) not allowed by decision trees

Figure 3.1: Partition feasibility of decision trees.

sets due to the restrictions imposed on decision trees. [Riv87] shows that decision lists (ordered rule sets) with at most k conditions per rule are strictly more expressive than decision trees of depth k. A similar result has been proven in [Bos95]. [Fur99] further argues that the disjointness restriction of decision trees imposes strong constraints on learnable rules. One resulting problem is the *replicated subtree problem* [PH90], where identical subtrees have to be learned at various places in a decision tree due to the fragmentation of the example space imposed by the restriction. Decision rules are less susceptible to this problem.

The disjointness constraint poses an obvious and costly restriction on the expressive power of decision trees. Rules, however, do not entail such a constraint and allow more flexibility in covering decision regions. If we only consider the difference of without or with the disjointness constraint, rule learning addresses a covering problem instead of a partitioning problem as in decision tree learning. It is known that solutions to a partitioning problem are also feasible solutions to the corresponding covering problem but usually with bigger sizes.

A less obvious restriction on decision trees comes from the common tree induction approach, which forces a recursive partitioning of the attribute space. This subtle point is illustrated in Figure 3.1, where both (a) and (b) partition the attribute space but only the partition in (a) is a feasible one allowed by decision trees.

As also pointed out in [WI93], decision rules are a theoretically stronger model than decision trees, and the restrictions imposed on decision trees cause the increase of tree complexity. As a tree grows in size, its explanatory capability diminishes. Therefore, in many situations decision rules are considered having a better effectiveness-efficiency tradeoff than decision trees.

3.3.4 Perfect and approximate models

A perfect decision tree [Qui86, Qui90] correctly classifies all the training examples. Accordingly, a perfect rule set must be complete and consistent.

Not all rule learners learn such perfect rule sets. Some rule learners, such as *CN2* [CN89], learn inconsistent rules as a means of tackling overfitting. Some other rule learners, such as *RIPPER* [Coh95], learn consistent rules but stop early before reaching completeness by utilizing multiple stopping criteria. *RIPPER* also assumes a closed world and skips the largest class for efficiency. The famous *AQ* family members [Mic69, ML86, MK01, WMKP06], *PRISM* [Cen87] and *RAMP* [Apt95] are some of the many methods that (can) learn perfect models.

To overcome overfitting of noise, many researchers advocate post-pruning over pre-pruning [BFOS84, WK91]. Post-pruning generally uses an overfit-and-simplify learning strategy, where a perfect model is learned first and pruned afterwards. Therefore, from the data classification point of view, it is beneficial for algorithms to retain the ability of learning perfect models. From the data description point of view, this ability is more than beneficial, it is a must. Perfect models are perfect and exact in data description, but imperfect and approximate in data classification whose focus is on generalization performance. Although the two objectives of higher description accuracy and shorter description length conflict and there can be many different trade-offs as different angles to view the data, perfect descriptions provide a special angle that is unique and irreplaceable. As discussed previously, cluster description and rule extraction are two example applications of data description.

While post-pruning is generally considered more effective, it can be less efficient in runtime than pre-pruning [Fur97, Fur99]. Data description applications also frequently require approximate descriptions as a compromise for interpretability priorities. Our *DesTree* algorithm used for cluster description performs post-processing in a way similar to post-pruning. It would be more efficient if such approximate descriptions can be obtained during description generation. Thus, it is desirable for learning methods to admit inconsistency during the induction process while keeping the ability to learn perfect models. *AQ18* [MK01] and *RISE* [Dom94] are methods of this type, as well as our *RGB*.

3.3.5 Merits of minimality

The merits of minimality of classification models have been well discussed [BEHW89, BEHW87, Ris89] and successfully utilized [QR89, MRA95]. The principal hypothesis is that simpler models lead to higher generalization accuracy. In other words, if two different models in the same representation equivalently describe a particular data set, the less complex of the two will have more prediction accuracy.

The notion that the accuracy of an explanation is associated with its simplicity dates back as early as the 14th century, when William of Occam first posited the medieval rule of parsimony, or principle of economy, that came to be known as *Occam's Razor*. The principle states that plurality should not be assumed without necessity, meaning that, one should not increase, beyond what is necessary, the number of entities required to explain anything.

More recently, machine learning researchers have followed the principle in biasing their algorithms toward finding hypotheses with simple representations [BS95]. It is (arguably) argued that the simplicity of nature and rarity of simple theories can be used to justify Occam's Razer. First, nature exhibits regularity and natural phenomena are more often simple than complex. At least, the phenomena humans choose to study tend to have simple explanations. Second, there are far fewer simple hypotheses than complex ones, so that there is only a small chance that any simple hypothesis that is wildly incorrect will be consistent with all observations.

The Minimum Description Length principle [Ris78] is an operational formalization of Occam's Razor, which triggered a large body of research in the communities of statistics, mathematics, machine learning, and philosophy since its introduction. By the Minimum Description Length principle, the best hypothesis for a given set of data is the one that leads to the largest compression of the data.

The design of decision trees has been largely influenced by the preference for simplicity. [Sal91] argues that regardless of which of the two types of trees (perfect or approximate) one may need, it is usually desirable to keep the size of the tree as small as possible. This is because smaller trees are most efficient both in terms of tree storage requirements and test time requirements. More importantly, smaller trees tend to generalize better for the unseen examples because they are less sensitive to the statistical irregularities and idiosyncrasies of the training data. [MP94, Mur95] has performed empirical investigations and concluded that the bias for smaller trees is generally beneficial in terms of accuracy.

The Minimum Description Length principle has been applied to prune trees for reducing error rates [QR89, MRA95].

While minimality of models is generally desirable, the complexity can be measured differently for different modeling techniques. For decision trees, it can be the total number of leaf nodes or the total number of tests (internal nodes). Correspondingly for decision rules, it can be the total number of rules or the total number of conditions. [FI90] provides an analysis for why the minimal total number of leaf nodes in a tree is perhaps the most important bias. In our MRS problem, the objective is to minimize the total number of rules. The two measures tend to agree with each other in general for both trees and rules, which is intuitive and also justified by our experiments.

We have to point out that in the machine learning literature, there have been disputes over the application, or the way of the application, of Occam's Razor [BS95]. In a general issue [Wol96, Wol94, Sch94, Sch93], many researchers claim that no inductive bias can be validly preferred over another without making certain *a priori* assumptions. Some researchers claim that the opposite of Occam's Razor is true [Dom99]. The *RISE* rule learning system, against Occam's Razor, is biased for complex models inducing substantially more complex rule sets than other rule learners.

In our study, we pay special attention to the data description application of interpretable models where the focus is on existing data instead of unseen data. Thus at a given consistency (accuracy) level, there is no dispute that simpler models lead to better interpretability, and the merits of minimality are much appreciated.

3.3.6 Optimality of models

Optimal decision trees have been studied [MM73, PM77, KK76, KK78, Cho92, GGRL99, AH05], where a perfect tree that correctly classifies all the training examples is learned with a specified objective measure optimized. Some of the common objectives are minimum number of leaf nodes, minimum number of nodes, minimum error rate, min-max path length, minimum expected path length, and maximum average mutual information gain [SL91].

Optimal decision trees provide the best solutions under certain conditions or in certain applications. However, the problem of inducting a truly optimal tree is very difficult, the NP-hardness results on different objectives are shown in [HR76, CQK89, MM91, Nau91, TC92]. Existing optimal tree induction methods use mathematical programming or branch and bound techniques. An inevitable problem associated with any of these methods is their

computational infeasibility. The required large amounts of computational time and space have placed a restriction on the type of tree and/or the number of features to be used at each node [SL91].

Our MRS problem corresponds to the optimal decision tree problem with the objective of minimum number of leaf nodes. Optimal rule learning has not been seriously considered in the literature. We have not seen any exact methods that solve an optimal rule set problem of any formulation. Although most rule learners implicitly reduce the complexity of rule sets in the pursuit of good generalization accuracy, the minimality bias has not been "seriously enforced" [Hon97]. This careless treatment often leads to solutions that are less satisfactory in terms of minimality. In our experimental comparisons with a popular rule learner *AQ21*, our algorithm *RGB* achieved about 40% reduction in both the number of rules and the number of conditions.

This lack of attention is partially due to the hard nature of the problem. As we have previously mentioned, optimal decision tree learning addresses a partitioning problem whereas optimal rule learning addresses a covering problem, which is generally harder than its corresponding partitioning problem in the sense that it has a much larger search space.

3.3.7 Existing rule learners

In the past few decades, new rule learners have never stopped emerging mainly from the machine learning literature. The continuous development of the famous *AQ* family algorithms may well reflect this endless effort. The first *AQ* member [Mic69] dates back to the late sixties of last century. Along the way, we have seen *AQ15* [ML86], *AQ17-DCI* [BM96] and *AQ18* [KM99, MK01], just to name a few. The latest *AQ21* [WMKP06] has just been released not long ago.

Most of the existing rule learners follow a separate-and-conquer approach, which originated from *AQ* and still enjoys popularity. The approach searches for a rule that covers a part of the given (positive) examples, removes them, and recursively conquers the remaining examples by learning more rules until no examples remain. The approach is also known as sequential covering, learning one rule at a time until all the positive examples are covered. Besides the *AQ* family, some other separate-and-conquer rule learners are: *PRISM* [Cen87], *CN2* [CN89, CB91], *SWAP-1* [WI93], *GROW* [Coh93], *IREP* [FW94], *RIPPER* [Coh95], *BEXA* [TC96], *PART* [FW98], *IREP++* [DB04], and *TRIPPER* [VH06]. [Fur99] provides a good survey on separate-and-conquer rule learning.

Among these algorithms, *AQ*, *CN2*, and *RIPPER* are the most popular ones. In the following, we discuss the working principles of the three algorithms in more detail.

The *AQ* type learning programs, e.g., [ML86], are originally oriented toward generating perfect (complete and consistent) rule sets. Approximate models are generated by post-processing to handle overfitting. From *AQ18* [MK01] onwards, they can admit inconsistency during the induction process while keeping the ability to generate perfect models. *AQ* follows an example-driven general-to-specific (top-down) search strategy. In learning one rule, it initializes the rule as *true* and randomly selects an uncovered positive example as seed. Then, it repeatedly specializes the rule until it still covers the selected seed but none of the negative examples. The specialization operator is to add a condition. An evaluation measure is used for ordering the conditions and determining which one to add. Although this evaluation measure can be set by an external entity such as an end-user or calling function, the one that is normally used is the ratio of positive examples covered by the candidate rule to the total examples covered by the candidate rule. The rule set learning process terminates until all the positive examples are covered. *AQ* employs a beam search, which can be viewed as several hill-climbing searches described above in parallel.

CN2 can be regarded as an extension of *AQ* with the ability to deal with noise without resorting to post-processing. It retains most characteristics of *AQ* but has made several significant changes. Unlike *AQ*, it is not example-driven, i.e., it does not depend on specific examples during search. It also uses different rule evaluation measures, e.g., entropy as in early versions of *CN2* [CN89], or the Laplace estimate [CB91] that penalizes rules with low coverage. In *CN2*, specializations of rules halt when no further specializations are statistically significant. While rules learned in *CN2* admit inconsistency during the induction process, there are no internal tuning mechanisms that guarantee the consistency of rules.

RIPPER improves on *IREP*, a separate-and-conquer rule learning algorithm that tightly integrates reduced error pruning to efficiently handle large noisy datasets. In *RIPPER*, a rule is learned via a top-down search and the specialization stops until no negative examples are covered by the rule. The specialization operator is to add a condition and the rule evaluation measure uses FOIL's information gain [QJ93]. Thus *RIPPER* does learn consistent rules. However, it stops adding rules early before the rule set reaches completeness. *RIPPER* uses the following stopping criterion: After each rule is added, the total description length of the rule set and the examples is computed. *RIPPER* stops adding rules when this description length is more than 64 bits larger than the smallest description length obtained so far,

or when there are no more positive examples. Completeness is not guaranteed. Besides, *RIPPER* skips learning the largest class of examples and uses negation as failure assuming a closed world.

While we explained the inability of *CN2* and *RIPPER* to learn perfect models, we fully realize that they choose to do so for a purpose.

There are a variety of other rule learners that do not separate and conquer, for example, *RISE* [Dom94]. [Dom94] argues that both divide-and-conquer and separate-and-conquer suffer from the splintering problem: as induction progresses, the size of the available examples dwindles, resulting in decisions being made with less and less statistical support. Statistical anomalies become harder to weed out, and noise sensitivity increases. As a remedy, *RISE* was proposed as a conquering-without-separating approach that learns a set of rules simultaneously. *RISE* performs a specific-to-general (bottom-up) search, which starts with a set of rules, each representing an example, and successively generalizes the entire rule set. For a categorical condition, the generalization operator is to drop it. For a numerical condition, the generalization operator is to extend its range to include the next higher (or lower) value outside it. These values are taken from an ordered list of mid-points between consecutive observed values of the attribute. Thus, *RISE* can accommodate both categorical and numerical attribute smoothly. The rule evaluation measure used in *RISE* can be tuned to learn perfect models or admit inconsistency during the induction process.

However, *RISE* is biased towards complex models instead of simple ones. It initializes the rule set with the example set and deletes a rule when it is identical to some existing one. Rules are unlikely to be identical even if they cover similar or even identical contents, thus *RISE* introduces a lot of redundancies in the rule set. Nevertheless, the generalization performance of *RISE* is claimed to be good [Dom94], which is used as empirical evidence to justify that the opposite of Occam's Razor is true [Dom99].

In most of the existing rule learning approaches, the minimality of rule sets has not been a "seriously enforced bias" [Hon97]. The *RAMP* [Apt95] rule generation system, however, generates minimal classification rules from categorical data. The primary goal of *RAMP* is to strive for a minimal rule set that is complete and consistent with the training data, utilizing a logic minimization methodology called *R-MINI* [Hon97]. This technique was first developed for programmable logic array circuit minimization (*MINI*) [HCO74, HCO87], and is considered as one of the best known 2-level logic minimization techniques. *RAMP* and some other popular rule learners are surveyed in [AW97].

The core heuristic used in *RAMP* to achieve minimality consists of iterating (for a reasonable number of rounds) over two key steps: generalization and specialization. The generalization step initializes the rule set as the example set. It takes each rule in the current set and opportunistically generalizes it while removing other rules that are subsumed. The specialization step takes each rule in the current set and specializes it to the most specific one necessary to continue covering only the unique positive examples it covers. Redundant rules will be removed during this step. A limit is used to control how long the iterative process would continue without observing an improvement (reduction). If no reduction takes place within this limit, the minimization process will be stopped.

While *RAMP* works only on categorical (or discretized continuous) data, *RGB* can work on both numerical and categorical data, primarily intended for numerical data.

3.3.8 Separate-and-conquer vs. greedy set covering

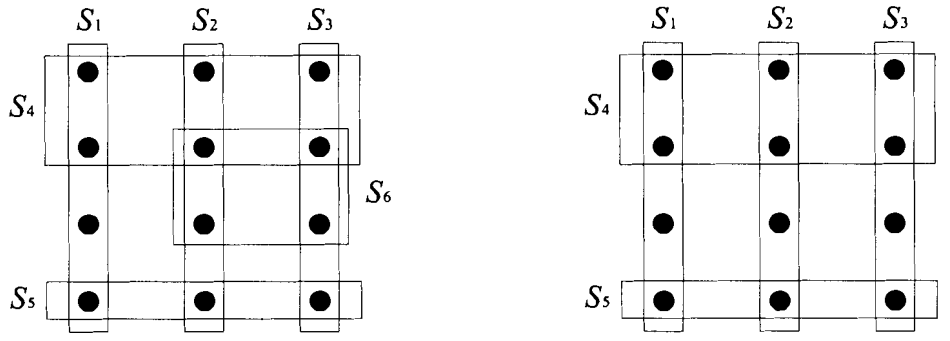
In the following, we introduce the greedy set covering algorithm and compare it with the separate-and-conquer (sequential covering) rule learning approach. Based on the comparison, we argue that the separate-and-conquer rule learners are generally inferior in rule learning when minimality is the objective.

The traditional set covering problem [Joh74a] seeks to select as few as possible subsets to cover all the elements in a given ground set. It is one of the most fundamental algorithmic problems that has many variants, settings, and applications. We give the formal definition in the following.

DEFINITION 3.3. *(Set Covering) Given $S_1, S_2, ..., S_m$ as subsets of a finite ground set X, find $I \subseteq \{1, 2, ..., m\}$ with $\bigcup_{i \in I} S_i = X$ such that $|I|$ is minimized.*

An instance of the MRS can be transformed into an instance of the set covering problem, where we consider the set of positive examples as the ground set X to cover, and assume given all the possible legal rectangles, $S_1, S_2, ..., S_m$, that are bounding boxes of some positive examples without covering any negative examples. We showed a similar insight in Section 2.3 when we studied alphabets for cluster description.

The set covering problem is NP-hard and approximable within $1 + \log |X|$ by a simple greedy algorithm [Joh74a]. The greedy set covering algorithm selects at each stage the subset that covers the largest number of uncovered elements. Let us look at how it works to solve the instance illustrated in example 1 of Figure 3.2.



(a) example 1 (b) example 2

Figure 3.2: Greedy set covering.

For example 1, the greedy algorithm selects S_4 first which covers 6 out of the 12 elements. Then it selects S_5 which covers 3 out of the 6 remaining elements. Then it selects S_6 which covers 2 out of the 3 remaining elements. Finally, it selects S_1 to cover the last remaining element. Thus, the greedy algorithm returns $\{S_4, S_5, S_6, S_1\}$, a cover of size 4.

This greedy set covering algorithm resembles sequential covering rule learners. It selects one subset (learning one rule) at a time, considers the uncovered elements (removing the covered examples), and iterates until all the elements (examples) are covered. In selecting one subset (learning one rule), the coverage of remaining elements (some statistical measure, e.g., Laplace estimate) is used for the greedy choice. In a way, the greedy set covering algorithm is better than most sequential covering rule learners in the sense that it knows exactly which subset covers the most uncovered elements and it selects it.

However, the greedy set covering algorithm does not have a superior performance in cover minimization. In example 1, the optimal set cover, $\{S_1, S_2, S_3\}$, has a size of 3. Note that this example is not designed purposely to make the greedy algorithm look bad. In example 2 of Figure 3.2, while the optimal solution $\{S_1, S_2, S_3\}$ still has a size of 3, the greedy algorithm returns $\{S_4, S_5, S_1, S_2, S_3\}$, a cover that requires all the 5 given subsets.

Due to the common greedy nature, the performance of separate-and-conquer rule learners on the MRS problem is quite limited. As evidence, our simultaneous learning algorithm *RGB* achieved about 40% improvement over *AQ21*, a separate-and-conquer representative, on benchmark datasets. The core procedure of *RGB*, *CAG*, is directly applicable to set covering instances and can easily obtain the optimal covers for both examples in Figure 3.2.

3.4 From Minimum Clique Partition to Minimum Rule Set

In this section, we establish interesting connections between the Minimum Rule Set (MRS) problem and the Minimum Clique Partition (MCP) problem, the dual problem of graph coloring, by a graph transformation of the example set. The optimal solution to the MRS problem is lower-bounded by the optimal solution to the MCP problem. The two problems are equivalent in 2-dimensional space. In this section, we also survey well-known constructive heuristics for the MCP problem.

3.4.1 Preliminaries

We start with some common terminology in graph theory. Then we discuss the well-known graph problems related to the MRS problem.

In graph theory, a graph is *complete* if all of its vertices are pairwise adjacent. A *clique* of a graph $G = (V, E)$ is a subset of V such that the induced subgraph is complete. An *independent set* of G is a subset of V such that no two vertices in the subset are connected in G. The *independence number* of G, usually denoted by $\alpha(G)$, is the cardinality of the largest independent set of G. Maximum independent sets should not be confused with maximal independent sets. An independent set is maximal if it is not a subset of any larger independent set. While the Maximum Independent Set problem is NP-hard, the problem of finding a maximal independent set can be solved in polynomial time by a trivial greedy algorithm [GJ79].

We say $V' \subseteq V$ is a *dominating set* if for all $u \in V - V'$, there is some $v \in V'$ such that $(u, v) \in E$. A maximal independent set, say V', is also a dominating set. If it is not, there must be some $u \in V - V'$ that is not adjacent to any $v \in V'$, then u can be added to V' to form a larger independent set, but then V' is not maximal.

A *vertex cover* of $G = (V, E)$ is a subset of V containing at least one of the two endpoints of each edge in E. Given a vertex cover V' of G, all vertices not in the cover, i.e., $V - V'$, define an independent set [Ski90]. This is rather straightforward. If $V - V'$ does not define an independent set, there must be at least one edge e connecting two vertices in it. Since both the two endpoints of e are not in V', V' cannot be a vertex cover. Therefore, the Maximum Independent Set problem is equivalent to the Minimum Vertex Cover problem.

Finding a maximum independent set is equivalent to finding a minimum vertex cover. Approximations to the two problems, however, differ widely. The Minimum Vertex Cover

problem has a 2-approximation algorithm (but has no polynomial-time approximation scheme unless P = NP), while the Maximum Independent Set problem has no constant-factor approximation unless P = NP.

In graph theory, the *complement* of a graph G is a graph $\overline{G}$ on the same vertices such that two vertices of $\overline{G}$ are adjacent if and only if they are not adjacent in G. Independent set and clique are opposite in the sense that every independent set in G corresponds to a clique in the complementary graph $\overline{G}$. The Maximum Independent Set problem and the Maximum Clique problem are equivalent, and most algorithms for solving one problem can be transformed into an algorithm which solves the other in the same time and space.

The Minimum Clique Partition (MCP) problem [PM81] is to find a partitioning of the vertices of a given graph G into the minimum number of disjoint vertex sets, each of which must be a clique in G.[2] The *clique partition number*, usually denoted by $\overline{\chi}(G)$, is the minimum number of cliques that form a partitioning of the vertices of G.

The well-known Minimum Graph Coloring problem, or graph coloring in short, is to use a minimum set of colors to color the vertices of a given graph $G = (V, E)$, such that no adjacent vertices receive the same color. The minimum number of colors needed is known as the *chromatic number* of G, usually denoted by $\chi(G)$. Since the vertices of an independent set can be safely colored with the same color, the graph coloring problem is precisely to minimize the number of disjoint independent sets of G that form a partitioning of V.

It is easy to see that the MCP problem and the graph coloring problem are dual problems. An instance of the former on G is an instance of the latter on $\overline{G}$. Also, we have $\overline{\chi}(G) = \chi(\overline{G})$, meaning that the clique partition number of G equals the chromatic number of $\overline{G}$.

OBSERVATION 3.1. $\alpha(G) \leq \overline{\chi}(G)$.

This is a rather straightforward observation, saying that the clique partition number of a graph is lower-bounded by the independence number of the graph. To see why, note that

[2]There have been some confusions in the literature about the definition of the MCP problem. Sometimes Minimum Clique Partition is referred to as the problem of finding a set of cliques that partitions the edges instead of the vertices as in our definition. Corresponding to the edge version of the MCP problem, the Minimum Clique Cover problem is also defined where overlapping of edges are allowed. We study the vertex version of the MCP problem. A corresponding vertex version of the Minimum Clique Cover problem is not defined. Any induced subgraph of a clique remains as a clique. By removing redundancies, a solution to the vertex version of the Minimum Clique Cover problem can be transformed into a solution to the vertex version of the MCP problem of the same size.

no two vertices in an independent set have a good relationship (no edge in between), they cannot share the same room (clique) and have to stay in their own.

The MCP problem belongs to the hardest category to approximate. While many combinatorial optimization problems have proved to be NP-hard, they behave different from the point of view of approximability [Tre05]. For some problems, researchers are able to devise polynomial time approximation schemes (abbreviated as PTAS), that is, to devise an r-approximation algorithm for every $r > 1$. For other problems, such as Minimum Vertex Cover, r-approximation algorithms for constant r are known, but no approximation scheme. Other problems, such as set covering, have no known constant factor approximation algorithms, but are known to admit an approximation algorithm with a slowly growing (logarithmic) performance ratio. The final group of algorithms, such as Maximum Independent Set and Minimum Graph Coloring, have no known slowly growing performance ratio to be achievable in polynomial time.

3.4.2 Graph coloring methods

In the following, we review existing techniques that solve the graph coloring problem. These techniques can be applied on the complementary graphs to solve the MCP problem. They also provide insight for designing heuristic solutions to the MRS problem.

Graph coloring techniques in general are surveyed in [CSD05] with a focus on metaheuristics. The interest in graph coloring originated from the question of how many different colors are needed to color a map such that no adjacent countries receive the same color. The problem was soon extended from planar graphs to more general graphs. Graph coloring has many applications in register allocation, job scheduling, air traffic flow management, wavelength assignment in optical networks, and timetabling. There are also many mathematical and statistical problems that can be solved through graph coloring, such as solving the algebraic structure of Quasi-groups, numerical estimation of large, sparse Jacobian matrices, and constructing Latin squares [Chi05]. As to be discussed shortly in §3.4.4, our MRS problem is related to the MCP problem and thus related to graph coloring as well.

The decision problem of graph coloring, the k-coloring number problem, was shown to be NP-complete for arbitrary k [Kar72]. The problem is solvable in polynomial time only for $k = 2$, and for arbitrary k on some special graphs such as comparability, chordal, circular arc, (3, 1), and interval graphs [GJ79]. Graph coloring approximations are surveyed in [Pas03].

Algorithms to solve the graph coloring problem fall into 3 categories: exact methods,

metaheuristics, and construction methods. Exact approaches include integer linear programming. Branch-and-bound, however, is a simpler approach of exactly coloring graphs without the need of dedicated software for solving linear programming. In the general branch-and-bound approach, each vertex is selected and colored with the lowest feasible color. A new color is introduced when no feasible color is available. In case the introduction of a new color makes the number of colors used more than the number of colors in the best legal coloring seen so far, the coloring is pruned and the search continues after backtracking to some previously colored vertex.

Metaheuristics start with some construction method to quickly get an initial solution, which is further improved with metaheuristic techniques such as stochastic local search, tabu search, simulated annealing [JAMS89], or genetic algorithms [EHH98]. These techniques perform differently on different types of graphs.

Construction methods are more practical in real applications involving large (more than 1000 vertices) and dense graphs due to their efficiency. Dense graphs have a number of edges roughly quadratic in the number of vertices. Sparse graphs exhibit a linear dependence. The complexity of a graph is usually measured by its edge density in addition to the number of vertices. Construction heuristics generally build feasible colorings in an incremental way. They start with an empty assignment and iteratively color the vertices until all vertices are colored. In the following, we introduce several representative construction heuristics.

DSATUR

In the *DSATUR* heuristic [JM82, Bre79], we are given a list of different colors indexed from 1 to $|V|$. The vertices are first sorted in decreasing order of degree and a vertex with the largest degree is assigned the color with the lowest index. Then at each construction step, the next vertex to be colored is chosen according to the *saturation degree*, that is, the number of different colors assigned to adjacent vertices. The vertex with the largest saturation degree is chosen and assigned the color with the lowest possible index such that the partial coloring remains feasible. Ties are broken favoring the vertex with the largest number of unassigned adjacent vertices. If ties still remain, they are broken randomly. *DSATUR* has a time complexity of $O(|V||E|)$.

Greedy-GC

Greedy-GC follows a straightforward greedy approach for graph coloring adopting an *excavation scheme* [Pas03]. The so-called excavation scheme performs separate-and-conquer, which iteratively finds an independent set IS and updates the graph with IS removed until the graph is empty. Note that vertices in an independent set can be colored with the same color, and the separate-and-conquer approach disallows overlapping of those discovered independent sets and thus the final coloring is a feasible one.

Different procedures for finding an independent set IS can be used leading to different excavation coloring algorithms. For example, [Joh74b] introduced *Greedy-IS*. Basically, it iteratively chooses a vertex v with the least degree from V and inserts it into IS that is initially empty. Then, V is updated with v and its adjacent vertices removed. The iteration terminates until V becomes empty.

The complexity of *Greedy-IS* is $O(|E|)$. It will be called at most $|V|$ times in the main loop of *Greedy-GC*, so the overall worst-case complexity of *Greedy-GC* is $O(|V||E|)$.

RLF

The *RLF* (Recursive Largest First) heuristic [Lei79] also follows the excavation scheme but uses a different procedure to find IS. The procedure first selects a vertex v with the largest degree from V and inserts it into IS. Then, v and all the vertices adjacent to v are removed from V and inserted into a set U, which is initially empty. While V is not empty, iteratively a vertex $v' \in V$ is chosen and inserted into IS that shares the largest number of edges with vertices in U, then v' and all the vertices adjacent to v' in V are removed from V and inserted into U. The iteration terminates until V is empty.

This IS-finding procedure also has an $O(|E|)$ running time as *Greedy-IS*. It will be called at most $|V|$ times, thus *RLF* has the same overall complexity of $O(|V||E|)$ as *Greedy-GC*.

Compared to *Greedy-IS*, the IS-finding procedure used in *RLF* keeps the set U of vertices that cannot be colored by the used colors to help guide the search. This information is not used in *Greedy-IS*.

The procedures to find independent sets certainly directly impact the overall performance of graph coloring. The procedures that can find larger independent sets are generally preferred. Research related to finding independent sets is surveyed in [Pas97] and [Pas03].

Among these algorithms, *DSATUR* and *RLF* are particularly popular heuristics with

the best performance reported in the literature. They both consistently achieve good results with reasonably low computational time. Theoretical analysis on these construction methods [dW90] shows that *DSATUR* and *RLF* are exact methods for bipartite graphs. Between the two, *RLF* performs better than *DSATUR* in many instance classes but is in practice more costly even if they have the same time complexity of $O(|V|^3)$.

Moreover, as pointed out by [Cul01], *DSATUR* and *RLF* are representatives of the two different strategies for coloring a graph: the heuristics that color a graph by iteratively selecting a vertex according to the number of colors already assigned to the adjacent vertices and the heuristics that color a graph by selecting large independent sets and using vertex degree information. Viewed from another angle, *RLF* can be considered as a separate-and-conquer approach, whereas *DSATUR* can be considered as a conquering-without-separating approach. The latter approach provides a richer search space and requires more sophisticated search techniques. Our algorithm *CAG*, the core procedure of *RGB*, is also a conquering-without-separating approach but with many fundamental differences from *DSATUR*. *CAG* starts with an optimal partial solution consisting of a subset of vertices instead of a single vertex. It works on a dynamically maintained bipartite assignment graph and uses different evaluation measures to guide the search. The details will be explained in later sections.

3.4.3 Minimum Clique Partition methods

As we have discussed, the MCP problem is the dual problem of graph coloring on the complementary graph. To solve an MCP instance on G, we can complement the graph G and apply the graph coloring techniques on $\overline{G}$. We can also "complement" the algorithms to achieve exactly the same results. In the following, we introduce these "complemented algorithms" that can be used for the MCP problem on a given $G = (V, E)$ directly.

$\overline{DSATUR}$

We are given a list of initially empty cliques indexed from 1 to $|V|$. The vertices are first sorted in increasing order of degree and a vertex with the smallest degree is assigned to the clique with the lowest index. Then at each construction step, the next vertex to be assigned is chosen according to the $\overline{\textit{saturation degree}}$, that is, the number of different cliques to which the non-adjacent vertices are assigned. The vertex with the largest $\overline{\text{saturation degree}}$ is chosen and assigned to the clique with the lowest possible index such that it remains as a

clique. Ties are broken favoring the vertex with the largest unassigned non-adjacent vertices. This tie breaking rule is the same for choosing the vertex with the smallest unassigned adjacent vertices since for any vertex, the total number of element vertices remain the same and they are partitioned into two groups: adjacent and non-adjacent. If ties still remain, they are broken randomly. $\overline{DSATUR}$ has a time complexity of $O(|V||E|)$.

$\overline{Greedy\text{-}GC}$

The excavation scheme used in graph coloring, complemented, also follows a separate-and-conquer strategy, which iteratively finds cliques instead of independent sets. Note that a clique in G is an independent set in $\overline{G}$. In the following, we show how $\overline{Greedy\text{-}IS}$ finds a clique $\overline{IS}$.

$\overline{Greedy\text{-}IS}$ iteratively chooses a vertex v with the largest degree from V and inserts it into $\overline{IS}$ that is initially empty. Then, V is updated with all the non-adjacent vertices of v removed. The iteration terminates until V becomes empty.

The overall time complexity of $\overline{Greedy\text{-}GC}$ is $O(|V||E|)$.

$\overline{RLF}$

$\overline{RLF}$ uses another procedure to find a clique $\overline{IS}$. It first selects a vertex v with the smallest degree from V and inserts it into $\overline{IS}$. Then, v and all the non-adjacent vertices of v are removed from V and inserted into a set U, which is initially empty. While V is not empty, iteratively a vertex $v' \in V$ is chosen that shares the smallest number of edges with vertices in U and inserted into $\overline{IS}$. Then v' and all the non-adjacent vertices of v' are removed from V and inserted into U. The iteration terminates until V becomes empty.

$\overline{RLF}$ also has an overall time complexity of $O(|V||E|)$.

3.4.4 Minimum Clique Partition vs. Minimum Rule Set

In the following, we relate the MRS problem to the traditional MCP problem by transforming the given example set into a consistency graph G_c. We also show that the optimal solution to the MRS problem is lower-bounded by the optimal solution to the MCP problem, and the two problems are equivalent in 2-dimensional space.

DEFINITION 3.4. *(MRS consistency graph) Given a multi-class example set X, a consistency graph for X, $G_c = (V_c, E_c)$, is a simple graph with $V_c = X$, and there is an edge*

between two vertices $u, v \in V_c$ if and only if they have the same class label and their bounding box contains no vertices of other classes.

A more general definition for consistency graph will be given shortly in Section 3.5 for the generalized Minimum Consistent Subset Cover problem, which has MRS as an instance. Given G_c, it is easy to obtain the following observation.

OBSERVATION 3.2. *Given X as an example set, the examples covered by a consistent rule form a clique in G_c, where G_c is the consistency graph of X.*

A consistent rule contains a set of examples of the same class. Then in G_c, these examples form a clique since there is an edge for any pair of them.

From the observation we can see that the two problems, MRS and MCP, are related. A feasible solution to the MRS problem is also a feasible solution to the MCP problem on G_c, thus the feasible region for the MRS problem is a subset of the feasible region of the MCP problem, which implies that an optimal solution to the MCP problem must be an optimal solution to the MRS problem. Therefore, the rule set number $\gamma(X)$ is lower-bounded by the clique partition number $\overline{\chi}(G_c)$, which is further lower-bounded by the independence number $\alpha(G_c)$ as we have discussed in the preliminaries.

THEOREM 3.1. *$\alpha(G_c) \leq \overline{\chi}(G_c) \leq \gamma(X)$, where G_c is the consistency graph for a given example set X.*

The proof is trivial and based on the fact that the examples covered by a consistent rule must form a clique in G_c. If under certain conditions that a clique must form a consistent rule too, the MRS problem and the MCP problem will be equivalent sharing the same feasible region and optimal solution. If the following, we show that it is the case in 2-dimensional space (and not in higher dimensional space). In the proof of Lemma 3.1, we show that (the bounding box of) a clique in 2-dimensional G_c also forms a consistent rectangle rule, thus, a consistent rule.

In our study, a *rectangle rule* is a special type of rule with all the attributes conditioned to specify the minimal ranges for the examples they cover. In other words, the antecedent of such a rectangle rule is a rectangle, which is geometrically a bounding box for the set of multi-dimensional examples it covers. While many rules may cover the same set of examples, a set of examples correspond to a unique rectangle rule since there is a unique bounding box

for a set of vertices. For this reason, our MRS problem can be translated into a problem of finding the smallest number of consistent rectangle rules that cover the given set of multi-class examples. Our algorithm *RGB*, as to be proposed in the next chapter, actually makes use of this observation. It first learns a set of rectangle rules, then removes redundant conditions for each rectangle rule, where the cardinality of the rule set is not changed.

LEMMA 3.1. *Given an example set X in d-dimensional space where $d \leq 2$, that is, each example in X has $d \leq 2$ attributes excluding the class attribute, a clique in the consistency graph G_c of X forms a consistent rule.*

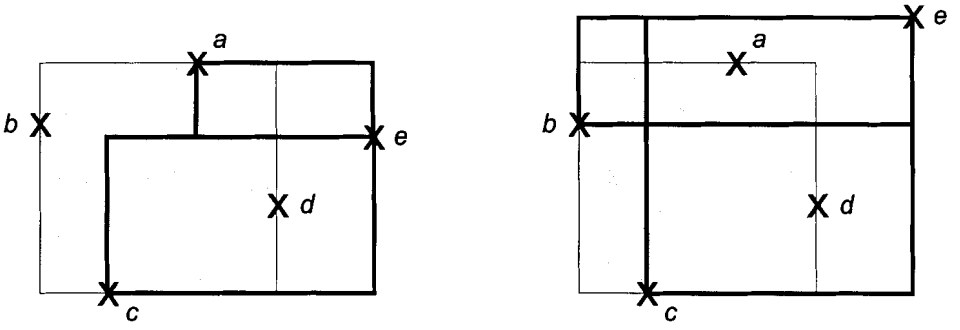


Figure 3.3: Proof of Lemma 3.1.

Proof. Let R be a rectangle containing a subset of vertices of V_c, let *Rects* be a set of rectangles containing the bounding boxes for all pairs of vertices in R. To prove the lemma, it is sufficient to prove the following *full coverage claim*: For any example $v \in V_c$, if v is covered by R, it must be covered by some rectangle in *Rects*. To see why, let us assume the full coverage claim is true, and the vertices in R form a clique but not a consistent rectangle rule. Since the vertices in R forms a clique, there must be an edge for each pair of vertices in R, i.e., all rectangles in *Rects* must be "pure" containing vertices of the same class. But since R does not form a consistent rectangle rule, there must be some vertex v in R with a negative class label. However due to the full coverage claim, v must be in some rectangle in *Rects*, causing that rectangle to be "impure", which is a contradiction.

The full coverage claim requires that the union of the bounding boxes of all pairs of vertices in R covers the bounding box of all the vertices in R. It is true for the d-dimensional space with $d \leq 2$. The proof is trivial for $d = 1$. Now we prove it for the case of $d = 2$ by induction on n, the number of vertices.

For the base case of $n = 2$, the claim is trivially true. Now we assume the claim is true for $n = k$ and prove it is true for $n = k + 1$. For a better explanation, we use Figure 3.3 to illustrate the inductive case.

In the figure, e is the newly added vertex and the shaded area is the bounding box for the k vertices. In a bounding box, each boundary must contain some vertex. In the figure, we use a, b, c, and d to denote these vertices. Note that we might not need as many as four vertices since two of them may merge into a single corner vertex. So a, b, c, and d can be considered as "symbolic vertices" where two (adjacent) symbols, say a and b, may map to a single real corner vertex.

Since the claim is assumed true for $k = n$, the bounding box for a, b, c, and d is shaded meaning that it is covered by the union of the rectangles in *Rects*. There are two cases in terms of where the newly added vertex e would appear: inside the shaded area or outside of it. If e is inside the shaded area, the claim is trivially true for $n = k + 1$.

There are two sub-cases for the case that e appears outside of the shaded area: e has one attribute or both attributes out of the range(s) of the shaded rectangle. Without loss of generality, we use the two graphs, left and right, in Figure 3.3 to illustrate these two sub-cases. In the left sub-case, the newly added area can be covered by two bounding boxes for the pairs of (a, e) and (c, e). In the right sub-case, the newly added area can be covered by two bounding boxes for the pairs of (b, e) and (c, e). Therefore, the claim is true for $n = k + 1$.

The following theorem immediately follows from Lemma 3.1.

THEOREM 3.2. *$\gamma(X) = \overline{\chi}(G_c)$, where G_c is the consistency graph for the example set X of an MRS problem that is in the d-dimensional space with $d \leq 2$.*

We have shown the MRS problem and the MCP problem are equivalent in 2-dimensional space. This equivalence does not hold for higher dimensional spaces as stated in the following observation.

OBSERVATION 3.3. *Lemma 3.1 does not hold for $d > 2$ and thus Theorem 3.2 does not hold either.*

It is easy to construct counter examples. In the following we give one for $d = 3$. Let $X = \{a, b, c, e\}$ where $a = (2, 4, 5)$, $b = (4, 3, 2)$ and $c = (7, 9, 4)$ are three positive examples

and $e = (3, 5, 3)$ is a negative example. Since e is not covered by any bounding box for a pair of examples in $R = \{a, b, c\}$, R is a clique in G_c. However, e is covered by the bounding box of R, thus R does not form a consistent rule.

One might be interested to know how big the gap between $\alpha(G)$ and $\overline{\chi}(G)$ is. The gap can be arbitrarily big. Let us consider a pentagon C_5, $\alpha(C_5) = 2$ and $\overline{\chi}(C_5) = 3$. We can have a graph with an arbitrary number of pentagons leading to an arbitrarily big gap between the independence number and the clique partition number.

3.5 Minimum Consistent Subset Cover Problem

The Minimum Rule Set (MRS) and the Minimum Clique Partition (MCP) problems share many similarities with the traditional set covering problem. In the following, we generalize them into the so-called Minimum Consistent Subset Cover (MCSC) problem. We also study the properties of the problem providing insights to the design of heuristic algorithms. The MCSC problem has many practical applications in data mining such as rule learning, clustering, and frequent pattern mining.

3.5.1 Problem definition

The MCSC problem finds the minimum number of consistent subsets that cover a given set of elements, where a subset is consistent if it satisfies a given constraint.

DEFINITION 3.5. (*Minimum Consistent Subset Cover*) *Given a finite ground set X and a constraint t, find a collection C of consistent subsets of X with $\bigcup_{S \in C} S = X$ such that $|C|$ is minimized, where a subset is consistent if it satisfies t.*

We also use (X, t) to denote an MCSC instance with ground set X and constraint t. The *consistent subset cover number* for (X, t), denoted by $\gamma(X, t)$, is the minimum number of consistent subsets with respect to t that cover X.

Given (X, t), we say the constraint t is *granular* if $\{x\}$ is consistent with respect to t for any $x \in X$. Apparently, with a granular constraint, (X, t) always has a non-empty feasible region. Most reasonable MCSC formulations feature granular constraints. For example, in the MCP problem, a single vertex also forms a clique. In the MRS problem, a single example forms a consistent rule.

The set covering problem can be considered as an MCSC instance where a subset is consistent if it is a subset of one of the given subsets. With the NP-hard set covering problem as an instance, the MCSC problem is NP-hard.

Consistent subsets are not given in typical MCSC instances. If we take a pre-processing step to generate all the consistent subsets, then an MCSC instance becomes a set covering instance and existing set covering heuristics are directly applicable. Unfortunately, as argued in 2.3, the generated collection of subsets would be prohibitively large and this is not a feasible approach to solve MCSC instances.

3.5.2 Properties of the problem

Covering problems have corresponding partitioning problems as special cases. Disallowing overlapping, partitioning problems are easier in the sense that they have smaller search spaces. Algorithms for partitioning problems usually work too for the associated covering problems but typically generate solutions of larger sizes. However, finding partitions can be an advantageous way of finding covers for MCSC instances under certain conditions.

DEFINITION 3.6. *(anti-monotonic constraint) Given (X, t), we say t is anti-monotonic if for any subset $S \subseteq X$ that is consistent, any $S' \subseteq S$ is also consistent.*

For example, the MCP problem has an anti-monotonic constraint since a subset of a clique is still a clique. The constraint in the MRS problem is also anti-monotonic since any subset of the set of points covered by a consistent rule forms a consistent rule.

THEOREM 3.3. *Given (X, t) where t is anti-monotonic, any solution to (X, t) can be transformed into a solution to the associated partitioning problem with the same cardinality.*

Proof. We give a constructive proof of the theorem. Suppose we have a solution to (X, t) at hand which is a set of overlapping consistent subsets. For each pair of consistent subsets that overlap, we can simply assign the overlap to any of the two and remove it from the other. Since t is anti-monotonic, the consistent subset with the overlap removed remains consistent. Then, we obtain a set of consistent subsets of the same cardinality that form a partition of X.

Theorem 3.3 implies that for (X, t) where t is anti-monotonic, optimal or good partitions are also optimal or good covers. By targeting on the simpler partitioning problem, we may design well-guided yet efficient search heuristics for (X, t).

In Definition 3.4 and Theorem 3.1, we introduced consistency graphs and lower bound results for the MRS problem. In the following, we generalize them for the MCSC problem.

DEFINITION 3.7. *(consistency graph) Given (X,t), a consistency graph for (X,t), $G_c = (V_c, E_c)$, is a simple graph where $V_c = X$, and there is an edge $(u,v) \in E_c$ for a pair of vertices $u, v \in V_c$ if and only if $\{u,v\}$ is consistent.*

OBSERVATION 3.4. *Given (X,t) where t is anti-monotonic, a consistent subset forms a clique in G_c, where G_c is the consistency graph for (X,t).*

The observation is rather straightforward. Since t is anti-monotonic, any pair of elements in a consistent subset must also constitute a consistent subset and the two corresponding vertices in G_c will share an edge connection.

The observation implies that a feasible solution to (X,t) is also a feasible solution to the MCP instance on G_c, thus the feasible region for (X,t) is a subset of the feasible region for the MCP instance, which implies that an optimal solution to the MCP instance must be an optimal solution to (X,t). Therefore, the consistent subset cover number $\gamma(X,t)$ is lower-bounded by the clique partition number $\overline{\chi}(G_c)$, which is further lower-bounded by the independence number $\alpha(G_c)$ as we have discussed in the preliminaries.

THEOREM 3.4. *$\alpha(G_c) \leq \overline{\chi}(G_c) \leq \gamma(X,t)$, where G_c is the consistency graph for (X,t) and t is anti-monotonic.*

Based on Theorem 3.4, since $\alpha(G_c)$ is the size of a maximum independent set in G_c, the size of any independent set must be less than or equal to any feasible solution to the MCSC (or MCP) problem. If the two are equal, then the solution is optimal for both. This implication has been used to confirm the optimality of some of our experimental results.

3.5.3 Extensions

In the following, we relax the conditions specified in the properties established above so that they can be extended to more applications. For example, converse k-center does not come with an anti-monotonic constraint. The removal of cluster centers may corrupt the consistency of clusters.

DEFINITION 3.8. (*pivot-anti-monotonic constraint*) *Given (X, t), we say t is pivot-anti-monotonic if for any subset $S \subseteq X$ that is consistent, there exists a* pivot *element $p \in S$ such that $S' \cup \{p\}$ is consistent for any $S' \subseteq S$.*

Example. Let us consider (X, t) for the converse k-center problem, where t requires each cluster to have a radius no larger than a given threshold. t is pivot-anti-monotonic with cluster centers as pivots. As long as the center remains, any sub-cluster of a consistent cluster remains consistent.

Obviously, if t is anti-monotonic, t must be pivot-anti-monotonic as well. A consistent subset could have multiple pivots. A pivot of a consistent subset may present in another as a non-pivot element. The concept of pivot can be extended from a single element to a set of elements, however, we keep the case simple in this study.

THEOREM 3.5. *Given (X, t) where t is pivot-anti-monotonic, and given that $S_1 \cup S_2$ is consistent if S_1 and S_2 are consistent subsets sharing the same pivot, any solution to (X, t) can be transformed into a solution to the associated partitioning problem with the same or smaller cardinality.*

Proof. We give a constructive proof of the theorem. Suppose we have a solution to (X, t) at hand which is a set of overlapping consistent subsets. We first merge all the consistent subsets sharing the same pivot resulting in a set of consistent subsets such that no two of them share the same pivot. Now, even if a pivot may still appear in an overlap of two consistent subsets, it is a pivot for one of them but not both. We just need to make sure the pivot is assigned to the one that needs it. We assign all the non-pivot elements in an overlap to either of the two overlapping consistent subsets and remove them from the other. Since t is pivot-anti-monotonic, the consistent subset with the non-pivot elements removed remains consistent. Then, we get a set of consistent subsets of (X, t) of the same or smaller cardinality that form a partition of X.

We also define the so-called *pseudo consistency graph* for (X, t), based on which a similar observation as in Observation 3.4 and a similar conclusion as in Theorem 3.4 can be obtained by following similar arguments.

DEFINITION 3.9. (*pseudo consistency graph*) *Given (X, t), a pseudo consistency graph for (X, t), $G'_c = (V'_c, E'_c)$, is a simple graph where $V'_c = X$, and there is an edge $(u, v) \in E'_c$ for a pair of vertices $u, v \in V'_c$ if and only if there exists $p \in X$ such that $\{u, v, p\}$ is consistent.*

OBSERVATION 3.5. *Given (X, t) where t is pivot-anti-monotonic, a consistent subset forms a clique in G'_c, where G'_c is the pseudo consistency graph for (X, t).*

THEOREM 3.6. *$\alpha(G'_c) \leq \overline{\chi}(G'_c) \leq \gamma(X, t)$, where G'_c is the pseudo consistency graph for (X, t) and t is pivot-anti-monotonic.*

In this study we focus on MCSC instances with anti-monotonic (or pivot-anti-monotonic) constraints. The theoretical results established above as well as our proposed algorithm *CAG* can be applied to such instances. In the following, we introduce several practical data mining problems that have MCSC formulations of this kind.

3.5.4 Applications

In this section, we discuss several practical data mining applications that can be formulated as MCSC instances with anti-monotonic constraints, in particular, the MRS problem, converse k-clustering, and frequent pattern summarization.

The MRS problem

The MRS problem we study in this chapter can be formulated as an MCSC instance (X, t), where X is the given example set and t requires a subset $S \subseteq X$ to form a consistent rule. In particular, S is consistent if its bounding box contains no examples of other classes. Since any single example forms a consistent rule, t is granular and (X, t) has a non-empty feasible region. In addition, any rule that covers a subset of the examples covered by a consistent rule is also consistent, thus t is also anti-monotonic, and the results in Theorem 3.3 and Theorem 3.4 are directly applicable. In fact, Theorem 3.1 is an MRS instantiation of the generalized Theorem 3.4.

The Minimum Description Length (MDL) problem we study in Chapter 2 is similar to the MRS problem, where we are given a cluster and a description format and asked to find a shortest description in the given format covering the cluster consistently and completely. If *SOR* or SOR^- is the given format, then the MDL problem is also an MCSC instance with a granular and anti-monotonic constraint.

Converse k-clustering

k-clustering methods generate k clusters that optimize a certain compactness measure, typically distance-based, that varies among different clustering models. While some measures use the sum or average of (squared) distances as in k-means and k-medoid [KR90], some measures use a single distance value, radius or diameter, as in k-center [TSRB71] and pairwise clustering [CCFM97]. The *radius* of a cluster is the maximum distance between a fixed point (center) and any point in the cluster, and the *diameter* is the maximum distance between any two points in the cluster.

A known limitation of k-clustering is that the appropriate number of clusters k is often hard to specify in advance, and methods that try to automatically determine this number have been investigated [PM00]. As another approach to address this limitation, alternative clustering models have been explored. In *converse k-clustering*, a compactness measure is given as threshold, and the task is to minimize the number of clusters k.

Converse k-clustering models have not received much attention in the data mining community. However, in many applications a distance-based constraint is easier to provide, based on domain knowledge, than the number of clusters. For example, molecular biologists have the knowledge that how similar a pair of sequences should be so that the two proteins can be assumed sharing the same functionality with high probability, or businessmen have the knowledge that how similar two customers should be so that they would have similar purchasing behaviors. The facility location problem [KP83], extensively studied in the operations research community, has the general goal of minimizing the total cost of serving all the customers by facilities. One of the problem formulations minimizes the number of located facilities based on the pairwise distance knowledge of customers, which is precisely a k-clustering model.

The converse k-center and converse pairwise clustering models can both be formulated as MCSC instances. In such an instance (X, t), X is the input data points for clustering and t requires a cluster to satisfy the maximum radius or diameter threshold constraint. As single distance measures, radius and diameter are more intuitive for domain experts to specify constraints than the more complex ones.

Both MCSC instances have granular constraints since singleton clusters satisfy any distance threshold. As explained in §3.5.3, converse k-center has a pivot-anti-monotonic constraint with cluster centers as pivots. For converse pairwise clustering, the constraint is

anti-monotonic since any sub-cluster can only have a smaller diameter than a cluster. The results established in §3.5.2 can apply to the two MCSC instances. Note that in converse k-center, a combination of two consistent clusters sharing the same center is also consistent, thus the additional condition required by Theorem 3.5 is satisfied and the theorem applies. In addition, we have the following observation.

OBSERVATION 3.6. *Given (X, t) as an MCSC instance for the converse pairwise clustering problem, we have $\overline{\chi}(G_c) = \gamma(X, t)$.*

Based on Observation 3.4, a consistent subset of (X, t) forms a clique in the consistency graph G_c. On the other hand, a clique in G_c must form a consistent subset, thus converse pairwise clustering is equivalent to the MCP problem on G_c. The same observation does not hold for the converse k-center problem.

The k-center problem has a variant, called continuous k-center where cluster centers are not required to be cluster members. The continuous converse k-center problem is in fact an MCSC instance with an anti-monotonic constraint.

Also in [EGG+06], the Connected k-Center (CkC) problem is studied advocating joint cluster analysis of attribute data and relationship data, where each cluster is required to be internally connected on a given graph. The converse CkC problem is as well an MCSC instance with a pivot-anti-monotonic constraint, or anti-monotonic for its continuous version.

Frequent pattern summarization

Frequent pattern mining has been one of the trademarks of data mining research and studied extensively for various kinds of patterns including itemsets, sequences, and graphs. While the mining efficiency has been greatly improved, interpretability instead became a bottleneck in the successful application of pattern mining. As a known problem, the overwhelmingly large number of generated frequent patterns containing redundant information are in fact "inaccessible knowledge" that need to be further mined and discovered. Interpretable summarization of large collections of patterns has emerged as an important research direction.

As a first approach, maximal frequent patterns [Bay98] and closed frequent patterns [PBTL99] have been introduced. These subsets of frequent patterns are more concise from which all the frequent patterns can be derived. However, the number of patterns generated in these two approaches is still too large to handle. As an alternative, methods have been developed that select a fixed number of frequent patterns to summarize the complete set of

patters. [HWLT02], for example, presents a method for mining the top-k frequent closed patterns of length no less than a given threshold. While these methods effectively limit the output size, they fail to take into account the redundancy among frequent patterns.

Some recent work aims at finding a fixed number of patterns representing the whole set of frequent patterns as well as possible. In [AGM04], the objective is to maximize the size of the part of the input collection covered by the selected k sets. In order to avoid trivial solutions, an additional constraint specifies a bound on the number of extra sets that are allowed to be covered. [YCHX05] presents an approach that takes into account not only the similarity between frequent itemsets, but also between their supports. Using a similarity measure based on Kullback-Leibler divergence, they group highly correlated patterns together into k groups.

Similar to the scenario for k-clustering, the appropriate number k of patterns to summarize the set of frequent patterns is often hard to specify in advance. However, the users may have the domain knowledge that how similar a group of patterns should be so that they can be represented as a whole without losing too much information. In light of this, some converse k-clustering models that can be formulated as MCSC instances, converse k-center or converse pairwise clustering, appear to be very reasonable and promising to provide concise and interpretable pattern summarizations. Such clustering models generate clusters, i.e., groups of patterns, with certain quality guarantee. With such an formulation, we have the objective of minimizing the number of pattern groups necessary to cover the entire collection of frequent patterns, an objective that is natural in the context of summarization since it maximizes the interpretability of the result and at the same time reduces redundancy. For the radius or diameter threshold, standard distance functions can be employed, such as the Jaccard's coefficient for itemsets or edit distance for sequential patterns.

These formulated MCSC instances with anti-monotonic constraints allow the design of efficient and effective heuristic algorithms for the NP-hard MCSC problem. Stimulated by the theoretical study in this section, we propose a graph-based generic algorithm *CAG* that serves as the core procedure for our rule learning algorithm *RGB* for the MRS problem. The good performance of *RGB*, as demonstrated in our experiments, is largely due to several novel design ideas embedded in *CAG*. We introduce these algorithms and ideas in detail in the following section.

3.6 *RGB* Rule Learning Algorithm

In this section, we introduce our rectangle-based and graph-based rule learner *RGB*. We start by explaining the working principles of *RGB*, then we summarize the properties of *RGB* by comparing it to existing rule learners. We introduce bipartite assignment graph G_a, based on which alternative evaluation measures can be designed. These measures are used in *CAG*, the core procedure of *RGB* to guide the search in solving instances of the Minimum Consistent Subset Cover (MCSC) problem. The generic *CAG* are presented and discussed in detail. Two running examples, one for the MRS problem and the other for the set covering problem, are used to demonstrate how *RGB* and *CAG* work.

3.6.1 Working principles

In *RGB*, we first focus on rectangle rules and formulate the MRS problem as an MCSC instance (X, t). Rectangle rules are a special type of rules with all the attributes conditioned to specify the minimal ranges for the examples they cover. In other words, the antecedents of such rectangle rules are bounding boxes (rectangles) and least general generalizations for the vertices (examples) they cover. However, there can be many redundant conditions whose removal will not negatively affect the consistency and coverage of a rectangle rule. In particular, a condition is considered redundant for a rule if its removal will not cause the inclusion of any new examples of different classes or the exclusion of any examples of the same class previously covered by the rule. Such conditions are dropped in *RGB* to obtain a compact rule through a general-to-specific beam search. Each rectangle rule will be processed in the same manner to obtain a corresponding compact rule and the resulting compact rule set will have the same cardinality as the rectangle rule set.

As shown in Algorithm 1, *RGB* first calls *CAG* for (X, t) and stores the set of learned rectangle rules in R. Then for each $r \in R$, a general-to-specific beam search is performed to remove redundant conditions. To explain the idea, we consider a beam search of width 1. We initialize and maintain a set of examples of other classes, the *elimination set* for r, that need to be eliminated to achieve consistency. The search starts with an empty rule, i.e., *true*, the most general rule, and the conditions to add are chosen from the original conditions in r. For the choice of condition to add, a greedy approach is adopted favoring the condition excluding the largest number of examples from the elimination set of r. The search stops when the elimination set is empty.

Algorithm 1 *RGB*

Input: X, t, and b: X is the example set, a set of multi-class examples. t is a constraint requiring rules to be consistent. b is a user-specified beam search width.
Output: R: a compact rule set with redundancy removed.

```
1: R ← CAG(X, t); //R stores a set of rectangle rules at this moment
2: for each r ∈ R
3:    initialize b empty rules r'_1, r'_2, ..., r'_b; //each r'_i contains no conditions initially
4:    initialize b sets of examples E_1, E_2, ..., E_b; //each E_i contains a set of examples for r'_i
      to eliminate, initialized as all the examples in X with different class labels from r
5:    while (E_1 ≠ ∅ ∧ E_2 ≠ ∅ ... ∧ E_b ≠ ∅)
6:       for each r'_i, choose the top b conditions from r, each being added to r'_i, to form b
         candidates that eliminate the most examples from E_i;
7:       choose the top b rules from the b × b candidates as r'_1, r'_2, ..., r'_b and update
         E_1, E_2, ..., E_b accordingly;
8:    end while
9:    r ← r'_j suppose E_j = ∅ caused the loop to terminate;
10: end for
```

Since the conditions to add are chosen from the original conditions in the input rules, the extracted rules are more general than their original ones. Since the consistency of each rule is also guaranteed, the resulting compact rule set R is complete and consistent.

Instead of the top-down approach, a bottom-up specific-to-general search could also be considered. However, for this particular task, the search may not be well guided. It is not straightforward to make an effective decision on which condition to drop for each generalization.

RGB is rectangle-based since it starts by learning a set of rectangle rules. *RGB* is also graph-based since its core procedure *CAG* works on consistency graphs and assignment graphs. The runtime of *RGB* is dominated by *CAG*. To be discussed in detail shortly, in solving an MCSC instance (X, t), *CAG* starts by constructing a maximal optimal partial solution from the consistency graph of (X, t), then performs an example-driven specific-to-general search on a dynamically maintained bipartite assignment graph to learn a set of consistent subsets with small cardinality. The characteristics of *CAG* are predominantly responsible for the many unique properties of *RGB*.

3.6.2 Properties of the algorithm

Compared to other rule learners, *RGB* has some unique and/or desirable properties. We list these properties and comment on them in the following.

1. Unlike most existing methods, *RGB* does not follow a separate-and-conquer approach where rules are learned one after another. Instead, all rules are learned simultaneously.

2. Unlike most existing methods, *RGB* is rectangle-based. It first generates a set of rectangle rules following an example-driven specific-to-general (bottom-up) search, and then extracts compact rules from the rectangle rules. For most bottom-up heuristics, the generalization operator is to drop conditions. In *RGB*, redundant conditions are dropped only after all the examples are covered by a set of learned rectangle rules.

3. Unlike most existing bottom-up methods that initialize the rule set as the example set, *RGB* starts with a subset of the example set, containing a maximal number of examples such that no two of them can be covered by any single rule that is consistent. This starting configuration corresponds to a maximal optimal partial solution.

4. Unlike most existing methods, *RGB* is graph-based. It transforms the input example set into a consistency graph G_c and works on a dynamically maintained bipartite assignment graph G_a. Rule evaluation measures derived from G_a are used to guide the search in *RGB* and make important decisions not only on how to generalize a rule, but also on which rule to be generalized. This enables the search to move forward towards a good set of rules, instead of a good single rule as in most other algorithms.

5. Unlike most existing methods that learn a set of rules for one class at a time, *RGB* naturally learns a set of rules for all classes simultaneously.

6. Unlike most existing methods that can only learn either consistent rule sets such as early members of the *AQ* family, or inconsistent rule sets such as *CN2* and *RIPPER*, *RGB* has the flexibility to admit inconsistency or not during the induction process, which makes it efficiently applicable to both noisy and noise-free environments.

7. *RGB* can smoothly work on both numerical and categorical attributes.

Most rule learners follow a separate-and-conquer approach such as the *AQ* family [Mic69, ML86, MK01] and *CN2* [CN89]. The strategy is also known as *sequential covering* [Mit97],

as rules are learned one after another and the already-covered positive examples are removed from the current working set. [Mit97] also refers to decision tree methods such as *ID3* [Qui86] and *C4.5* [Qui93a] as performing *simultaneous covering*. If we consider the difference between partitioning and covering, a more appropriate name for the decision tree strategy might be simultaneous partitioning, while simultaneous covering can be used to describe a category of algorithms, such as *RGB* and *RISE* [Dom94], that learn a set of rules simultaneously. [Dom94] used *conquering without separating* to describe this non-mainstream strategy. But unlike *RGB*, *RISE* is biased towards complex models. The search techniques of the two are thus quite different following different biases.

Our rectangle-based approach provides a conservative way of learning. Least general generalizations are performed during the specific-to-general search while the rectangles become larger and larger. Each rectangle is a bounding box of the examples it covers and a special type of rule with all the attributes conditioned. No conditions are dropped in the generalization operations. Compact rules are extracted in *RGB* via a general-to-specific beam search to remove those redundant conditions while maintaining the coverage and consistency of rules. It actually performs a further generalization of the rectangle rules. For most other algorithms, the generalization operator is to drop conditions. Our example-driven rectangle-based learning process is conservative in the sense that it makes small steps in generalization, and thus the learning process can be finer-tuned and better-guided.

Most separate-and-conquer methods employ a top-down strategy to search the hypothesis space, such as *AQ* and *CN2*. The few bottom-up methods usually initialize the rule set with the example set where each example is an initial rule. *RGB* initializes the rule set with a maximal subset of examples such that no two of them can be covered by any single rule that is consistent. This is implemented by finding a maximal independent set of the consistency graph G_c transformed from the example set X. Each independent set of G_c is an optimal partial solution. The bigger the optimal partial solution, the closer it is to the optimal solution. By the discovery of a maximal independent set, we start with a maximal optimal partial solution.

In the development of a rule set in *RGB*, new rectangle rules are created on demand for examples that cannot be covered by any existing rule. Redundancy of rules is thus discouraged. A rule is redundant if all the positive examples it covers are covered by other learned rules without violating consistency. *RISE*, also a bottom-up approach, starts with the example set and deletes a rule when it is identical to some existing one. Rules are

unlikely to be identical even if they may cover similar or even identical contents, thus their approach introduces a lot of redundancies, resulting in a rule set that is complex and tedious. However, this is claimed to be their intention, i.e., they believe complex models lead to better generalization performance [Dom99]. Another bottom-up algorithm that works on categorical data only, *R-MINI*, strives to find minimized rule sets as *RGB*; however, it requires an additional specialization step to remove redundancies.

The MRS problem is an MCSC instance. *CAG*, the core procedure of *RGB*, performs an example-driven specific-to-general search on a bipartite assignment graph G_a to solve MCSC instances. We derive evaluation measures, such as *weighted edge gain*, from G_a to guide the search and make important decisions on which member vertex is the next to be assigned to which condensed vertex. In the context of rule learning, these decisions are about how to generalize (or specialize) a rule and which rule should be generalized (or specialized). Most existing rule learners do not address the second decision. Separate-and-conquer methods do not have to worry about the choice of rule since they follow a very greedy approach learning one rule at a time. Although various measures have been proposed, from as easy as accuracy and purity to more complex ones such as cross entropy, Laplace estimate, m-measure and ϕ-coefficient, they are all used to guide the specification (or generalization) operations within learning a single rule. For the conquering-without-separating method *RISE*, rules are generalized one after another following a pre-determined order. In summary, our method uses evaluation measures derived from the dynamically maintained G_a to guide the search, making effort towards a good set of rules, instead of a good single rule.

It is certainly an advantage to have the flexibility of admitting inconsistency or not during the induction process, which provides an efficient way of handling noise while keeping the ability to learn perfect rule sets. *CN2* and *RIPPER* are efficient in handling noise, but they cannot be used in situations where a perfect rule set is desirable. Early members of the *AQ* family only learn perfect rule sets, overfitting treatment has to be done during post-pruning, which can be less efficient. *AQ18* (onwards) and *RISE* are some of the few algorithms that also provide this flexibility.

RGB can work on both numerical and categorical attributes. To generalize categorical conditions, the attribute values are simply ORed since there are no ranges to expand as for numerical conditions. Thus rectangles do not provide generalization or compression on categorical attributes. However, during the further generalization process of compact rule

extraction, it is likely that categorical conditions can be dropped (generalized). Many rule learners are designed for categorical data. Similar to *RGB*, *RISE* can accommodate both numerical and categorical attributes using a different, non-example-driven, generalization operator. In *RISE*, a numerical condition is generalized by extending its range to include the next higher (lower) value outside it. These values are taken from an ordered list of the midpoints between consecutive observed values of the attribute. For categorical conditions, they follow the convention to drop conditions.

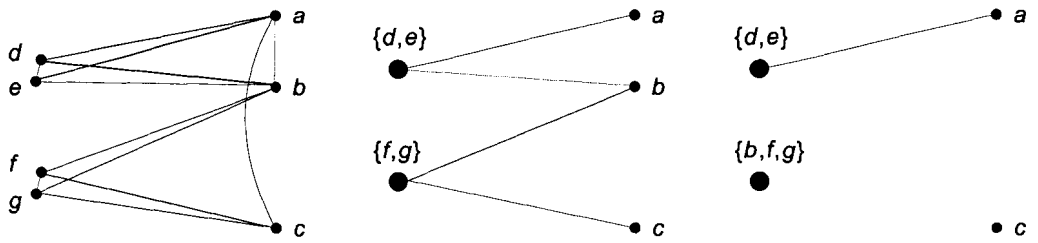
3.6.3 Assignment graph

A very unique characteristic of *CAG* (and thus *RGB*) is the use of bipartite assignment graphs, which provide the necessary information for effective evaluation measures to be designed to guide the search in solving MCSC instances.

DEFINITION 3.10. (*Assignment graph*) *In a bipartite assignment graph* $G_a = (U_a \cup V_a, E_a)$, U_a *contains a set of element vertices each representing an uncovered vertex, and* V_a *contains a set of condensed vertices each representing a consistent subset.* $(u, v) \in E_a$ *if and only if* $u \in U_a$ *and* $v \in V_a$ *and* $v \cup \{u\}$ *is consistent.*

In *CAG*, G_a is dynamically maintained showing all the feasible choices for the next assignment. Each condensed vertex is essentially a set of vertices condensed together. In order to maintain the consistency of subsets, an element vertex u can be assigned to some condensed vertex v only via an edge connection between them. However, each assignment may cause some edges to disappear in G_a. For those isolated element vertices (with degree of 0), new condensed vertices have to be created. Each creation may introduce some new edges to G_a.

In Figure 3.4, (a) shows the consistency graph G_c for an MCSC instance of the MCP problem. (b) shows a possible stage of G_a, where $\{d, e\}$ and $\{f, g\}$ are two condensed vertices and a, b, and c are three element vertices. The vertex a may be assigned to $\{d, e\}$, causing no extra edge losses in G_a since b is the only vertex that might be affected but it shares an edge with a in G_c. The vertex b may be assigned to $\{d, e\}$ or $\{f, g\}$. If b is assigned to $\{f, g\}$, the vertex c will lose the edge connection to $\{f, g\}$ since it is not connected to b in G_c. Thus when an element vertex has multiple choices, it is important to make a decision on which condensed vertex to assign to. (c) shows another possible stage of G_a. The vertex c cannot be assigned to any condensed vertices and a new condensed vertex has to be created for it.



(a) original G_c (b) a stage of G_a (c) another stage of G_a

Figure 3.4: Vertex assignment.

This creation, however, will cause some edge gain, i.e., a will have an edge connecting to the newly-created condensed vertex since it has an edge connecting to c in G_c.

OBSERVATION 3.7. *While a vertex assignment may cause multiple edge gains or losses in G_a, it can cause at most one edge gain or loss for any single element vertex.*

The observation holds because all the gained or lost edges have to be incident to the condensed vertex involved in an assignment, and any element vertex shares at most one edge with a condensed vertex. The implication of the observation is that if an element vertex has a degree of more than two, it will not become isolated after the next assignment.

Evaluation measures

In the following, we show how the information embedded in G_a can help to guide the search by answering important questions in the assignment process: Which element vertex is the next to be assigned? Which condensed vertex should it be assigned to?

In principle, the element vertex with the least degree should be considered first since it is most likely to get isolated. Also based on Observation 3.7, element vertices with degree more than two will stay "safe" after such an assignment. Thus, a *least degree first* criterion can be used to choose the next element vertex to be assigned.

Notice that $\overline{DSATUR}$ uses the largest $\overline{\text{saturation degree}}$ first criterion. This is in fact an MCP instantiation of the least degree first criterion. $\overline{\text{saturation degree}}$ is the number of different cliques to which the non-adjacent vertices are assigned. To maintain the status of being a clique, two non-adjacent vertices cannot be assigned to the same clique. Thus the

$\overline{DSATUR}$ criterion favors a vertex u with the largest number of unavailable cliques, i.e., u with the least number of available cliques, which is precisely the least degree first criterion.

The largest $\overline{\text{saturation degree}}$ first criterion can only be used for the MCP problem. It cannot be used for other MCSC instances such as MRS. In the MRS problem, a condensed vertex (consistent rectangle rule) v may not be available to u even if u is adjacent to all vertices in v. Recall that a consistent rectangle rule must be a clique but not vice versa. The least degree first criterion and the use of G_a, however, are general mechanisms that can be used for any MCSC instances.

An assignment of u would cause extra changes in E_a. If $d^\circ(u) = 0$, we may gain some edges; otherwise, we may lose some edges. The edges in G_a are the relational resources for the assignment of element vertices and we want to keep as many edges as possible. Edges have different degrees of importance. In general, a gained or lost edge is more important if it is incident to an element vertex with a smaller degree in G_a. We design a measure, *weighted edge gain*, that reflects the influence of an assignment transaction on G_a.

In the following, we define $weg(uu)$ and $weg(uv)$. $weg(uu)$ is the weighted number of edges that can be added to E_a if u is chosen to become a new condensed vertex. In such cases, u is assigned to itself, which happens only when $d^\circ(u) = 0$, where $d^\circ(u)$ denotes the degree of u in G_a. $weg(uv)$ is the weighted number of edges that would get lost (an edge loss is just a negative gain) if u is assigned to v via edge $(u, v) \in E_a$. For $weg(uu)$, the weight for each possibly added edge is $\frac{1}{d^\circ(u')+1}$ where we use $d^\circ(u') + 1$ because $d^\circ(u')$ can be 0 and for that case, the weight should be the largest, i.e., 1. For both measures, different weighting schemes can be used.

$$weg(uu) = \sum_{\forall u' \in U_a \, s.t. \, \{u,u'\} \, is \, consistent} \frac{1}{d^\circ(u') + 1}$$

$$weg(uv) = \sum_{\forall u' \, s.t. \, (u',v) \in E_a \wedge \, v \cup \{u,u'\} \, is \, inconsistent} \frac{-1}{d^\circ(u')}$$

Once u, the next element vertex to be assigned, is chosen, a *largest* $weg(uv)$ *first* criterion can be used to decide which condensed vertex v should take u so as to keep as many weighted edges as possible.

As we have explained and shown in Figure 3.4, while b can be assigned to either $\{d, e\}$ or $\{f, g\}$, its assignment to $\{d, e\}$ would cause no extra edge losses and result in a clique

cover of size 2. On the other hand, its assignment to $\{f, g\}$ would cause c to lose its only edge and result in a clique cover of size 3. Thus the choice of condensed vertex is crucial in solving MCSC instances. However, many MCP methods do not have this concern, e.g., $\overline{DSATUR}$. As in many other methods, $\overline{DSATUR}$ chooses the clique with the lowest index, which is pre-determined.

We also define the following measure $weg(u)$ that measures the influence on G_a if u is chosen as the next element vertex to be assigned. The associated *largest* $weg(u)$ *first* criterion can be used alone for the choice of u. It can also be used as a tie-breaking mechanism for the least degree first criterion in choosing u.

$$weg(u) = \begin{cases} weg(uu) & \text{if } d^{\circ}(u) = 0 \\ weg(uv) \text{ where } v = argmax_{\forall v' \text{ s.t. } (u,v') \in E_a}\{weg(uv')\} & \text{otherwise} \end{cases}$$

In the formula, for $d^{\circ}(u) = 0$, $weg(u) = weg(uu)$. For other cases, it is the largest $weg(uv)$ value considering all the condensed vertices adjacent to u in G_a, indicating the best assignment choice for u.

MCP and MRS implementations of the evaluation measures

The above defined weg measures appear in their generic forms. They need to be implemented for different MCSC instances. As an example, in the following, we show how they can be adapted to the MCP problem. Note that for an MCP instance, the consistency graph $G_c = (V_c, E_c)$ is the same as its original input graph.

$$weg(uu)_{MCP} = \sum_{\forall u' \in U_a \text{ s.t. } (u,u') \in E_c} \frac{1}{d^{\circ}(u') + 1}$$

$$weg(uv)_{MCP} = \sum_{\forall u' \text{ s.t. } (u',v) \in E_a \wedge (u,u') \notin E_c} \frac{-1}{d^{\circ}(u')}$$

$weg(uu)_{MCP}$ is implemented as such because if u becomes a new condensed vertex and an element vertex u' shares an edge with u in E_c, $\{u, u'\}$ will be consistent, i.e., forming a clique.

$weg(uv)_{MCP}$ is implemented as such because if u is assigned to v, even if u' has an edge with v in E_a, but if it does not have an edge with u in E_c, $v \cup \{u, u'\}$ will not be consistent

since it will not form a clique. Thus the edge between u' and v will get lost. For the MCP problem, this condition includes all the edges (not incident to u itself) that possibly get lost, since if $(u', v) \in E_a$ and $(u, u') \in E_c$, $v \cup \{u, u'\}$ is guaranteed to be a clique. Note that $(u', v) \in E_a$ infers that $u' \in U_a$, i.e., u' is unassigned.

For other MCSC instances, the implementation of $weg(uu)$ is as simple as $weg(uu)_{MCP}$. However, the implementation for $weg(uv)$ may incur further consistency checking. For example, for the MRS problem or any other MCSC instance with an anti-monotonic constraint, even if $(u', v) \in E_a$ and $(u, u') \in E_c$, $v \cup \{u, u'\}$ might not qualify as a consistent subset. To implement $weg(uv)$ precisely, we may need to perform consistency checking for all $v \cup \{u, u'\}$ where u' satisfies $(u', v) \in E_a$ and $(u, u') \in E_c$ to see if there are additional edge losses.

3.6.4 Generic *CAG*: the core procedure

In this section, we introduce a generic algorithm *CAG* that works with consistency graphs and assignment graphs to solve MCSC instances featuring anti-monotonic constraints. It is the core procedure of *RGB* as the MRS problem can be formulated as an MCSC instance.

Overview

Given an MCSC instance (X, t) where the constraint t is anti-monotonic, *CAG* starts by constructing a maximal optimal partial solution from the consistency graph G_c, then performs an example-driven specific-to-general search on a dynamically maintained bipartite assignment graph G_a to learn a set of consistent subsets with small cardinality.

The design of *CAG* has closely followed the insights provided in Section 3.5. From the definition of G_c and Theorem 3.4, we know that any pair of vertices in an independent set of G_c cannot appear in the same consistent subset. Thus a maximal independent set of G_c, denoted by IS, constitutes a maximal optimal partial solution to (X, t).

Each vertex in IS forms a singleton consistent subset, represented by a so-called *condensed vertex* in an assignment graph G_a, the rest of the vertices in G_a are called *element vertices*. G_a is defined such that there is an edge between an element vertex u and a condensed vertex v if and only if u can be assigned to v while maintaining the consistency of v. Since a maximal independent set is also a dominating set as discussed in the preliminaries, there is an edge for each u connecting to some v in the initial G_a.

Element vertices are processed in a sequential manner and assigned to condensed vertices. With the growth of condensed vertices, some element vertices would get isolated and new condensed vertices have to be created for them. Upon completion, G_a becomes edge-free with all vertices assigned, and the set of condensed vertices, each representing a consistent subset, constitute a solution to (X, t).

The dynamically maintained G_a provides the necessary information based on which effective evaluation measures can be designed to guide the search by deciding which element vertex is the next to be assigned to which condensed vertex. For example, with the *least degree first* criterion, we can process first the element vertex with the least degree since it is most likely to get isolated.

Note that CAG actually returns a partition of X. As argued in Section 3.5, a solution to a partitioning problem is also a feasible solution to its corresponding covering problem. Also due to Theorem 3.3 and Theorem 3.5, partitioning does not cause the increase of solution size compared to covering under certain conditions. In addition, partitioning problems have much smaller search spaces and the search can be better guided.

Also note that under anti-monotonic constraints, if a subset $S \subseteq X$ is not consistent, $S' \supseteq S$ cannot be consistent. Thus such inconsistent S does not need to be considered and the search space can be significantly pruned. Our assignment graph G_a maintains consistency of all condensed vertices after each assignment transaction allowing CAG to work in a pruned search space.

In contrast to the most-seen separate-and-conquer approach, e.g., the greedy algorithm [Joh74a] for set covering and most existing rule learners, CAG adopts a less greedy example-driven strategy to learn consistent subsets simultaneously. In summary, the construction of initial optimal partial solution, the use of assignment graph, and the example-driven simultaneous learning strategy are the three novel design ideas that account for the good performance of CAG.

Details of *CAG*

We have explained the working principles of CAG in the overview. In the following we present the algorithm and explain it in more details.

CAG starts by initializing the assignment graph G_a using a maximal independent set of G_c (line 1). Then, the element vertices are processed in a sequential manner (line 2). For each assignment, a decision is made on which $u \in U_a$ is the next element vertex to be

Algorithm 2 generic *CAG*

Input: (X, t): an MCSC instance.
Output: V_a: a set of condensed vertices representing consistent subsets that cover X.

1: initialize $G_a = (U_a \cup V_a, E_a)$;
2: **while** $(U_a \neq \emptyset)$
3: choose $u \in U_a$; *//choose the next vertex to be assigned*
4: **if** $(d^\circ(u) = 0)$ **then**
5: $V_a \leftarrow V_a \cup \{u\}$;
6: $U_a \leftarrow U_a \backslash \{u\}$;
7: $E_a \leftarrow E_a \cup \{(u, u')\}$ for each u' with $u' \in U_a$ and $\{u, u'\}$ is consistent; *//update E_a, possibly add some edges*
8: **else**
9: choose $v \in V_a$; *//choose the condensed vertex to take the chosen u*
10: $v \leftarrow v \cup \{u\}$; *//assign u to v*
11: $U_a \leftarrow U_a \backslash \{u\}$;
12: $E_a \leftarrow E_a \backslash \{(u, u')\}$ for each u' with $(u', v) \in E_a$ and $v \cup \{u, u'\}$ is inconsistent; *//update E_a, possibly remove some edges*
13: **end if**
14: **end while**

assigned (line 3). After u is chosen, if it has a degree of 0 (line 4), a new condensed vertex has to be created for u (line 5) and u will be removed from U_a (line 6). At this moment, E_a needs to be updated (line 7). Some element vertices (that are connected to u in E_c) may be able to connect to the newly added condensed vertex u resulting in creation of some new edges. If u is not an isolated vertex (line 8), another decision is to be made on which $v \in V_a$ should take u (line 9). After the assignment of u to v (lines 10, 11), E_a also needs to be updated (line 12). Because of the assignment of u to v, some element vertices, say u', that previously connected to v would lose the connection if $v \cup \{u, u'\}$ is inconsistent. Upon completion, $U_a = \emptyset$ and $E_a = \emptyset$. The set of condensed vertices V_a corresponds to a set of consistent subsets together covering X.

Initializing G_a. One way of initializing G_a is to derive the consistency graph G_c for (X, t) first then find a maximal independent set IS of G_c using some known heuristic, e.g., *Greedy-IS* introduced in [Joh74b]. With IS obtained, we let $V_a = IS$, $U_a = V_c \backslash IS$, and $(u, v) \in E_a$ if and only if $u \in U_a$ and $v \in V_a$ and $(u, v) \in E_c$.

In some cases, e.g, for the MCP problem, G_c is identical to the input graph. In other cases, G_c needs to be derived and this would take $O(|X|^2 t_c)$ time where t_c denotes the

consistency checking time for some subset of X. This is because each pair in X need to be checked for consistency. To improve efficiency, we provide another way of initializing G_a without actually deriving G_c. Initially, V_a and E_a are empty and $U_a = X$. For each $u \in U_a$, we check each $v \in V_a$ and if $v \cup \{u\}$ is consistent, we add edge (u, v) into E_a. If no edge can be added, we add u into V_a. Upon completion, V_a also contains a maximal independent set IS of G_c. This procedure would take $O(|IS||X|t_c)$. Note that usually $|IS| \ll |X|$.

Choosing u and v. By varying the decision making schemes, i.e., how to choose u (line 3) and v (line 9), the generic CAG can have different instantiations that are suitable for different applications. By default, we use the least degree first criterion to choose u with the largest $weg(u)$ first criterion for tie-breaking. Also, we use the largest $weg(uv)$ first criterion to choose v.

With the default scheme, the time spent in choosing u would be on calculating $weg(u)$ for tie-breaking. As an approximation, we adopt a sampling technique. We only consider a constant number of element vertices with the tied least degree and for each of them we only consider a constant number of vertices for consistency checking. This sampling technique works well because the least degree first criterion greatly reduces the number of vertices in consideration in calculating $weg(u)$. Recall that only the condensed vertices adjacent to u in G_a need to be considered. Therefore, the runtime for the choice of u is $O(t_c)$.

For the choice of v, the calculation of $weg(uv)$ is part of the calculation of $weg(u)$ and thus it takes $O(t_c)$ as well.

Updating G_a. By varying the G_a updating mechanisms (lines 7 and 12), the generic CAG can be adapted to different applications. These applications have different constraints requiring different consistency checking mechanisms for the purpose of updating G_a.

In line 7, the edge set E_a of G_a is updated after an isolated element vertex u becomes a new condensed vertex. This may introduce some new edges to E_a. Note that line 7 will be executed no more than $|V_a| - |IS|$ times where V_a here represents its final stage after CAG terminates, thus the total time spend on line 7 would be at most $O((|V_a| - |IS|)|X|t_c)$.

In line 12, the edge set E_a of G_a is updated after an element vertex u is assigned to a condensed vertex v. This may cause the loss of some edges incident to v. Each such update could take a worst case runtime of $O(|X|t_c)$. To reduce the runtime, we adopt a *late update* approach that significantly improves efficiency without sacrificing performance. This is based on the observation that the update attempts for many edges in G_a do not have

an impact on the assignment process if the corresponding vertices are not considered in the next assignment. Based on the least degree first criterion, we only need to check the edges connecting to element vertices with the least (or small) degrees since they will be considered in the next assignment. This late update strategy would leave some element vertices in G_a having several extra edges that should have been removed. However, the vertices will be checked for consistency sooner or later when their degrees decrease along the assignment process. Therefore, by checking consistency for a constant number of vertices connected to v, we only spend $O(t_c)$ time for this update without sacrificing performance.

The consistency checking time t_c is different for different mechanisms and different applications. For the MCP problem, t_c is constant since we only need to check if an edge is present or not in the input graph. For the MRS and converse k-clustering problems we discussed previously, a brute-force consistency checking would take $O(|X|)$ time. With the help of some multi-dimensional indexing technique, it would take log time on average.

Time complexity. As we have explained for each step, initializing G_a (line 1) takes $O(|IS||X|t_c)$. The update of G_a for the $d^\circ(u) = 0$ case (line 7) takes in total $O((|V_a| - |IS|)|X|t_c)$ time. Together they take $O(|V_a||X|t_c)$ time. Recall that IS is a maximal independent size of G_c, V_a is the final condensed vertex set, and t_c denotes the consistency checking time for a subset of the ground set X.

In each iteration, the choice of u (line 3), the choice of v (line 9) and the update of G_a for the $d^\circ(u) \neq 0$ case each takes t_c time. There are $O(|X|)$ iterations and thus together they take $O(|X|t_c)$ time.

Overall, the worst case runtime is $O(|V_a||X|t_c)$, where t_c is constant for the MCP and converse pairwise clustering problems. For the MRS problem, t_c is $O(|X|)$ for the brute-force consistency checking mechanism, which can be improved to log time on average by indexing techniques. In general, $|V_a| \ll |X|$ as can be observed from our experimental results in Table 3.1.

3.6.5 Running examples

In the following, we go through two running examples, one for the MRS problem and the other for the set covering problem, to demonstrate how *RGB* and *CAG* work.

MRS example

In Figure 3.5, (a) shows an example set X. (b) shows a typical result from a separate-and-conquer approach such as AQ on X. The greedy approach in learning one rule favors rules with large coverage, thus a rule covering $\{a, b, c, d, e\}$ is often learned resulting in four rules in total. Since we focus on the number of rules, for clarity, we use rectangles to indicate the corresponding rules that are possibly more compact yet covering the same examples. The rectangle rules returned by CAG are shown in (c).

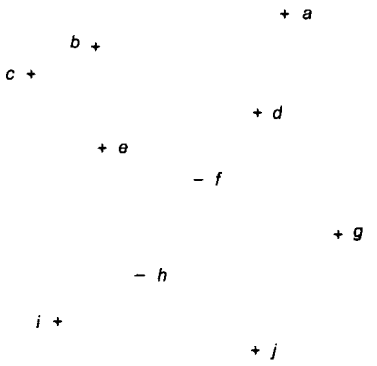
Figure 3.6 shows how the rectangle rules are learned in RGB for the MRS running example. (a) shows the consistency graph $G_c = (V_c, E_c)$ transformed from X. We can see that there is an edge between a pair of vertices if the two vertices have the same class label and their bounding box contains no vertices of other classes. (a) also shows a maximal independent set $IS = \{c, f, g\}$ of G_c.

Figure 3.6 (b) shows the initial assignment graph $G_a = (U_a \cup V_a, E_a)$ derived from G_c. In the initial G_a, we have $V_a = IS = \{c, f, g\}$ and $U_a = V_c - IS = \{a, b, d, e, h, i, j\}$. There is an edge $(u, v) \in E_a$ if and only if $u \in U_a \wedge v \in V_a \wedge (u, v) \in E_c$. Note that initially, there are no isolated vertices in G_a since IS is a maximal independent set and a dominating set.

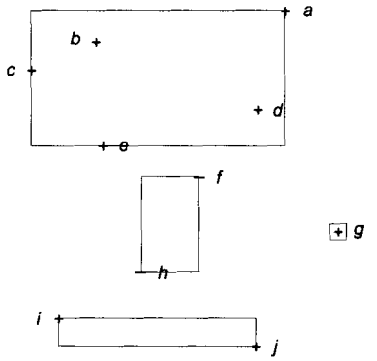
Figure 3.6 (c) to (i) show the iterative process that CAG works on G_a to assign all the vertices in U_a until it is empty. We use u to denote the next vertex to be assigned and v the condensed vertex for u to be assigned to. From the initial G_a shown in (b), we try to choose u according to the least degree first criterion. There are many vertices having the least degree of 1, after tie-breaking, h is chosen. Since $d^\circ(h) = 1$, there is only one choice for the condensed vertex, thus $v = \{f\}$. (c) shows the updated G_a after the first assignment of h to $\{f\}$. In the same fashion, j is assigned to g and (d) shows the updated G_a. b is assigned to c and (e) shows the updated G_a. e is assigned to $\{c, b\}$ and (f) shows the updated G_a.

In Figure 3.6 (g), the updated G_a is shown after i is assigned to $\{c, b, e\}$, which causes the loss of two edges incident to a and d respectively. Recall that though an assignment may cause multiple edge losses, it can cause at most one edge loss for any element vertex.

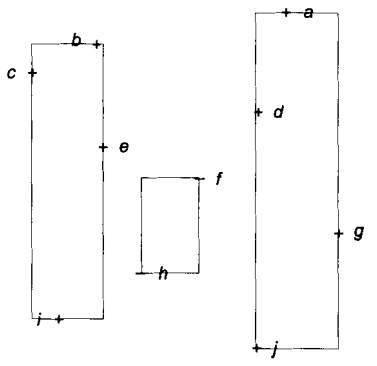
Then, a is assigned to $\{g, j\}$ and Figure 3.6 (h) shows the updated G_a. Finally, d is assigned to $\{g, j, a\}$ and (i) shows the updated G_a. This is also the last stage of G_a where $U_a = \emptyset$ and $E_a = \emptyset$. V_a contains the set of three condensed vertices $\{c, b, e, i\}$, $\{f, h\}$, and $\{g, j, a, d\}$ that cover V_c. Each of the condensed vertices represents a consistent rectangle rule. This is actually an optimal solution since it has the same size as IS.



(a) example set X

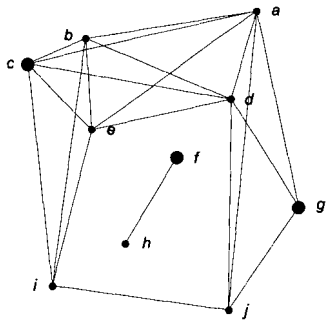


(b) separate-and-conquer result

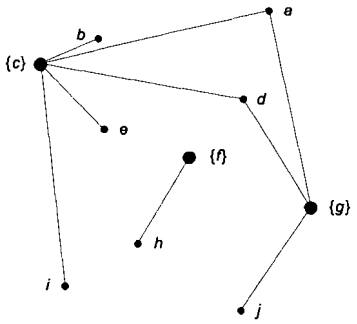


(c) *RGB* result

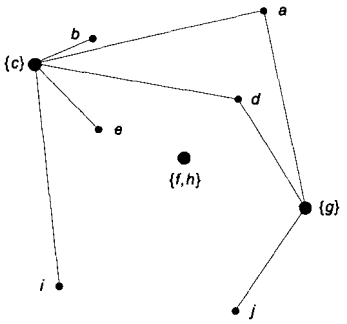
Figure 3.5: MRS running example: rectangle rules.



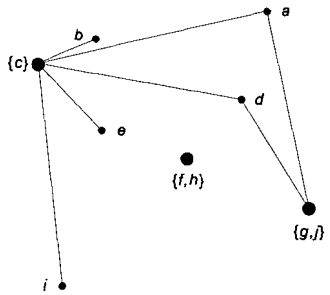
(a) $G_c = (V_c, E_c)$ and IS



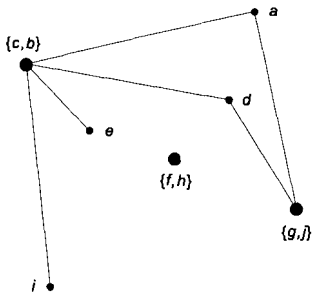
(b) initial G_a



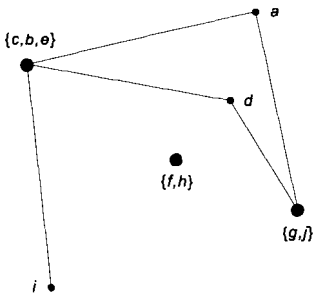
(c) assignment of h



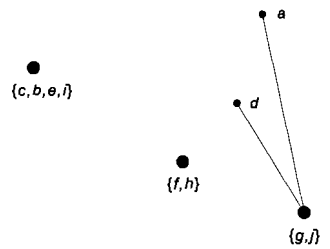
(d) assignment of j



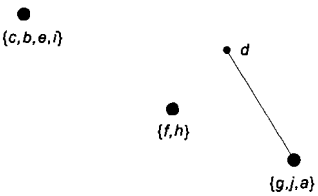
(e) assignment of b



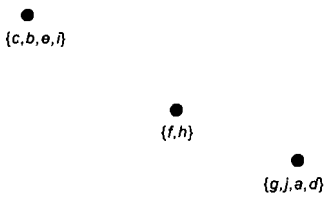
(f) assignment of e



(g) assignment of i



(h) assignment of a



(i) assignment of d

Figure 3.6: MRS running example: learning rectangle rules.

Set covering example

In Figure 3.7, (a) shows a set covering instance. We can see that the greedy algorithm will select all the three given subsets. (b) shows the consistency graph G_c for the MCSC formulation of the instance, from which we can derive a maximal independent set IS containing c and e. (c) shows the initial G_a with condensed vertices c and e.

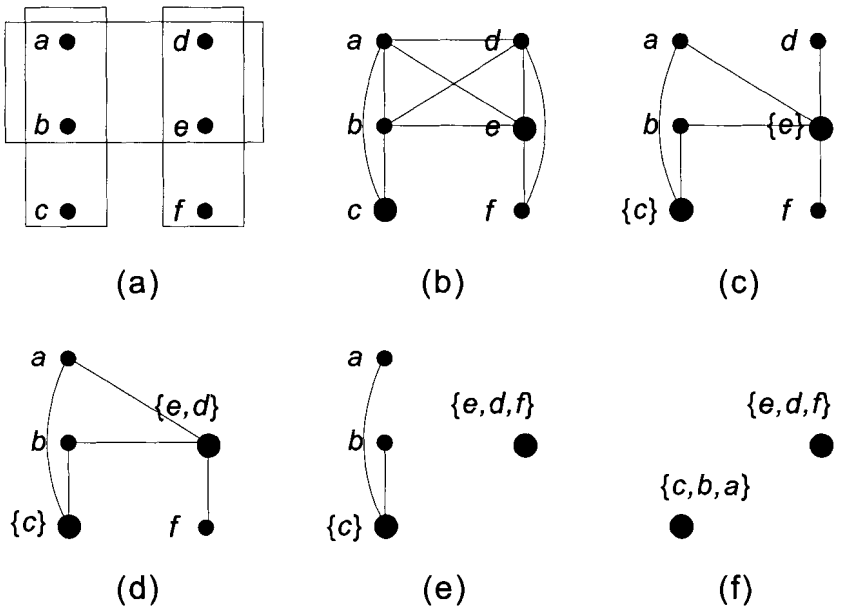


Figure 3.7: Set covering running example.

Now we start the assignment process. Both d and f have the least degree of 1, but assigning d to $\{e\}$ would not cause any edge loss, thus d is chosen and assigned to $\{e\}$, and (d) shows the updated G_a. Next, f is chosen and assigned to $\{e,d\}$, and (e) shows the updated G_a. We can see that G_a lost two edges since a or b cannot join $\{e,d,f\}$ while maintaining its consistency, i.e., $\{e,d,f,a\}$ or $\{e,d,f,b\}$ is not a subset of a given subset. Afterwards, a and b are assigned to $\{c\}$ and (f) shows the final G_a, where both U_a and E_a are empty and the condensed vertex set V_a represents a consistent subset cover.

The above standard CAG process can be simplified tailoring to the set covering problem. We check the involved condensed vertex after each assignment. If it uniquely identifies a given subset, all the vertices in the subset will be assigned to the condensed vertex.

Interested readers can verify that CAG easily obtains optimal solutions for both the examples illustrated in Figure 3.2.

3.7 Empirical Results

We have performed extensive experimental evaluations on our rule learning algorithm *RGB* using UCI benchmarks [BM98]. We used *AQ21* [WMKP06] as our comparison partner, which is a popular rule learner generating complete and consistent rule sets. Experiments show that *RGB* achieved about 40% reduction in the number of rules and 30% reduction in the number of conditions.

Our comparison partner, *AQ21*, is a popular separate-and-conquer rule learner also generating perfect rule sets. Other popular rule learners such as *CN2* and *RIPPER* do not return perfect rule sets. Another conquering-without-separating method *RISE* has a bias towards complex models. Currently, most rule learners follow a separate-and-conquer approach originated from *AQ*. Thus the performance of the latest release of the *AQ* family, *AQ21*, can well represent the performance of most rule learners in the same category.

Our study has a focus on numerical attributes, thus we have chosen numerical datasets from the UCI repository for our experiments. Also, in handling categorical attributes, *RGB* uses a richer language than typical rule learners by allowing $< val >$ to be an internal disjunction of attribute values, instead of a single attribute value. Thus the choice of numerical datasets is also for fairness.

The experimental results are presented in Table 3.1. We explain the columns in the following:

- *Dataset*: the dataset
- *ins*: number of instances
- *dim*: dimensionality
- *cla*: number of classes
- *AQ21*: *AQ21* results on the dataset; two results are presented, *rul* indicates the number of rules and *con* indicates the total number of conditions
- *LearnCovers*: *LearnCovers* results. *LearnCovers* is an extension of *Learn2Cover* from 2-class to multi-class, where compact rules are further extracted using the same method as in *RGB*.
- $|E_c|$: cardinality of the edge set E_c of the consistency graph G_c

- $|IS|$: cardinality of a maximal independent set IS derived from G_c
- *cliques*: CAG result on the dataset for the Minimum Clique Partition problem
- RGB: RGB results on the dataset
- *reduction*: reduction achieved by RGB compared to $AQ21$ where

$$reduction = \frac{AQ21\ result\ -\ RGB\ result}{AQ21\ result}$$

Table 3.1: *RGB* results

Dataset	*ins*	*dim*	*cla*	*AQ21*		*LearnCovers*		$\|E_c\|$	$\|IS\|$	*cliques*	*RGB*		*reduction*	
				rul	*con*	*rul*	*con*				*rul*	*con*	*rul*	*con*
balance-scale	625	4	3	154	577	153	676	31578	153	153	153	676	0.006	-0.172
bupa	345	6	2	66	344	49	255	22149	12	19	38	224	0.424	0.349
car	1728	6	4	66	343	57	342	386152	55	57	57	340	0.136	0.009
diabetes	768	8	2	112	744	75	428	142994	10	18	57	420	0.491	0.435
ecoli	336	7	8	36	128	34	138	13569	22	24	28	131	0.222	-0.023
glass	214	10	6	6	7	6	9	5921	6	6	6	9	0.000	-0.286
haberman	306	3	2	73	198	66	260	6425	48	52	55	214	0.247	-0.081
ionosphere	351	34	2	25	172	13	60	33058	4	4	11	57	0.560	0.669
iris	150	4	3	10	26	8	20	3562	7	7	7	19	0.300	0.269
letter	15000	16	26	1160	15701	755	8377	3751964	137	221	552	6684	0.524	0.574
new-thyroid	215	5	3	10	33	9	22	12196	6	6	7	31	0.300	0.061
page-block	5473	10	5	112	687	75	450	11754498	28	37	60	369	0.464	0.463
satimage	4435	36	6	212	4379	145	1316	1784464	17	20	99	1316	0.533	0.669
segment	2310	19	7	49	382	34	195	374323	13	15	27	183	0.449	0.521
sonar	208	60	2	11	186	6	101	10761	2	2	5	79	0.545	0.575
spambase	4601	57	2	118	2292	87	1789	4686024	33	38	64	1286	0.458	0.439
vehicle	846	18	4	109	1133	66	479	80643	13	15	55	478	0.495	0.578
vowel	990	10	11	76	583	56	429	43164	20	20	47	367	0.382	0.370
waveform	5000	21	3	266	4989	192	3610	4164798	6	6	107	2347	0.598	0.530
wine	356	13	3	7	29	4	17	5324	3	3	4	17	0.429	0.414
yeast	2968	8	10	250	1740	255	1632	136416	66	117	194	1606	0.224	0.077
average				139.4	1651.1	102.1	981.2	1307142.0	31.5	40	77.8	802.5	**0.371**	**0.308**
reduction				0	0	0.267	0.401				**0.442**	**0.514**		

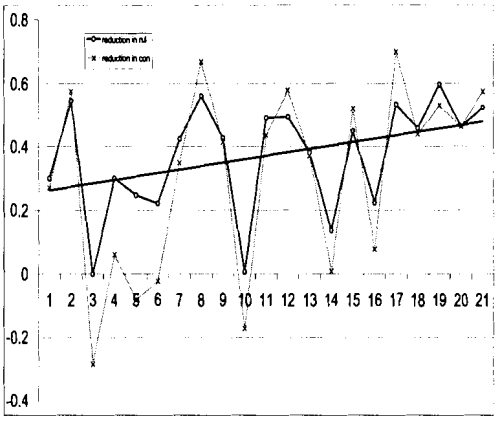
From Table 3.1 we can see that *RGB* achieved 37.1% and 30.8% averaged reduction over *AQ21* in terms of the number of rules and the number of conditions respectively. In the last row of the table, the reduction is calculated for the averaged number of rules and conditions, where we can see that *RGB* achieved 44.2% and 51.4% reduction over *AQ21* in terms of the average number of rules and average number of conditions respectively.

Recall that in Theorem 3.1, we have $\alpha(G_c) \leq \overline{\chi}(G_c) \leq \gamma(X)$, meaning that the rule set number is lower-bounded by the clique partition number, which is further lower-bounded by the independence number. Although many results presented in Table 3.1 are not optimal, the same trend exhibits in all datasets. In the table, we have observed that for each dataset, $|IS| \leq$ *cliques* $\leq$ *RGB rul* $\leq$ *LearnCovers rul* $\leq$ *AQ21 rul.*

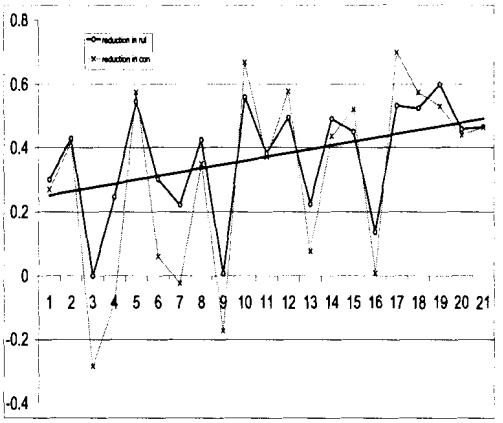
We also previously discussed the implications of Theorem 3.1. Any feasible solution to the Maximum Independent Set problem must be less than or equal to any feasible solution to the MCP (or MRS) problem. If the two are equal, then the solution is optimal for both. From Table 3.1 we can see that for datasets balance-scale, glass, and iris, *RGB* found the maximum independent set, the minimum clique partition, and the minimum rule set. In addition, for datasets ionosphere, new-thyroid, sonar, vowel, waveform, and wine, *RGB* found the maximum independent set and the minimum clique partition.

Our previous algorithm *LearnCovers* also achieves good reductions over *AQ21*, but not as good as *RGB*. However, *LearnCovers* is a more efficient algorithm than *RGB*. It also performs example-driven search, but the data are sorted before the learning process and the results would vary depending on the choice of dimension for sorting.

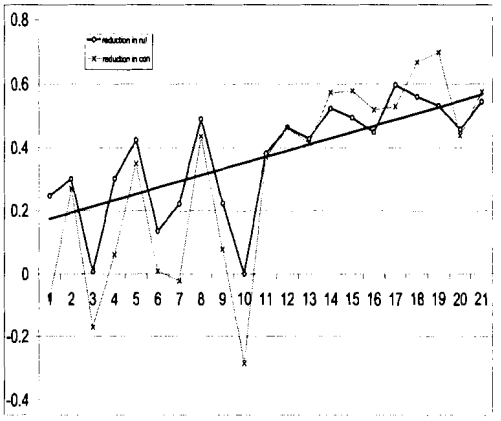
In Figure 3.8, we show the trend of the reductions achieved by *RGB* over *AQ21* by varying different parameters. In (a), (b), and (c), the 21 datasets are sorted on ins (number of instances), $|E_c|$ (cardinality of E_c), and dim (dimensionality) respectively. For all of them, we see both the "reductions in *rul*" (number of rules) and "reduction in *con*" (number of conditions) exhibit an increasing trend. The linear regression for the "reduction in *rul*" is also shown for each of them. In (d), the datasets are sorted on "reduction in *rul*", and the clear increasing trend of "reduction on *con*" confirms that the two model complexity measures, number of rules and number of conditions, are consistent with each other.



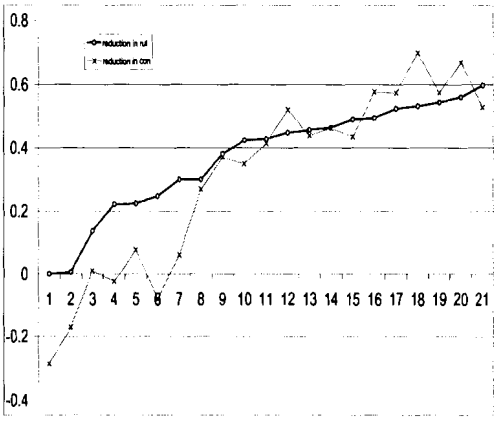
(a) sorted on ins



(b) sorted on $|E_c|$



(c) sorted on dim



(d) sorted on "reduction in *rul*"

Figure 3.8: *RGB* reductions over *AQ21*.

3.8 Extensions and Discussions

In this section, we discuss possible directions of improving *CAG* and *RGB*. We also generalize the Minimum Rule Set (MRS) problem to admit approximate models and explain how to adapt *RGB* to the generalized MRS problem. In addition, we discuss the overfitting issue and propose promising approaches for rule pruning.

3.8.1 Improving the algorithms

There are multiple directions that we can extend our work on the generic *CAG* as well as *RGB* for their improvement.

The excavation scheme is widely used for graph coloring. Such heuristics repeatedly find an independent set and removes it until the vertex set is empty. It is generally agreed that procedures that find larger independent sets are favored leading to better coloring algorithms and better approximation ratios. We feel that *CAG* can benefit from larger independent sets too, since the larger the starting independent set, the closer the optimal partial solution is to the optimal solution. Thus when running time is not a critical issue, we can derive the consistency graph from the input and apply a better heuristic than *Greedy-IS* that finds larger independent sets. For example, *QUALEX* [BBP02] shows the ability to find exact solutions for many hard instances in $O(n^3)$ time using a quadratic programming formulation for the maximum independent set problem.

To improve *CAG*, we may also vary the weighting scheme used in the weighted edge gain measures, or we may design new evaluation measures to guide the search. Different weighting schemes/measures favor different types of graphs, so it is possible to tailor the design to different situations.

In particular, the default *CAG* uses $weg(u)$ for tie-breaking. It can actually be used alone for the choice of u following the so-called largest $weg(u)$ first criterion. In this scenario, the choices of u and v are unified to follow the same general preference considering the influence of assignments on G_a. Note that this mechanism does not conflict with the least degree first intuition. Edge losses on the vertices with small degrees have more weights and they contribute more to $weg(u)$, thus a choice of u with high degree will be discouraged. When used alone, the largest $weg(u)$ first criterion is more sensitive to the weighting scheme.

Besides the discussed measures, there are many other evaluation measures that can be investigated, for example, the cardinality of condensed vertices. We may want to evenly (or

unevenly) distribute the vertices, then condensed vertices with small (large) cardinalities would be favored.

Due to the creation-on-demand (of condensed vertices) strategy, redundancy is not a serious issue in *CAG*. However, it is worthwhile to evaluate how *CAG* would benefit from a generalize-then-specialize redundancy removing procedure as used in *R-MINI*.

RGB is graph-based. The evaluation measures have been designed based on graph properties solely. The distance information in the original example set X has been somehow ignored. This in general would not cause serious problems since a pair of examples usually do not share an edge if they are far away and having many examples of different classes in between. However, there can be rare cases where pairs of far-away examples are connected in the consistency graph G_c, which may mislead the search in *CAG*. Thus we believe that by incorporating the distance information, either in the graph transformation procedure or in the search procedure, *RGB* can be improved to be more robust and effective.

3.8.2 Admitting approximate models

The study of approximate models is necessary from both the data classification and data description points of view. For data classification, perfect models are desirable for noise-free data. However, we live in a noisy world and perfect models are likely to overfit the data, thus approximate models are considered advantageous in achieving better generalization accuracy. For data description, perfect models may appear lengthy and hard to interpret, approximate models can provide better interpretability-accuracy trade-offs allowing the users to zoom out to view the data.

Post-processing (post-pruning) is always one way of admitting approximate models, which was used in the early *AQ* members [ML86]. However, post-pruning is less efficient than pre-pruning. It is desirable for a rule learner to generate approximate models during the induction process, while keeping its ability of generating perfect models. *AQ18* implemented a so-called $Q(w)$ measure for this purpose.

Approximate models can be obtained by introducing inconsistency and/or incompleteness. The MRS problem can be easily extended to admit both. Recall the MRS problem is to find a complete and consistent set of rules. We only need to redefine *complete* and *consistent*. A rule set is considered complete or consistent if it passes a user-defined completeness threshold or consistency threshold. These thresholds can be defined as global constraints on the rule set level, or local constraints on the rule level.

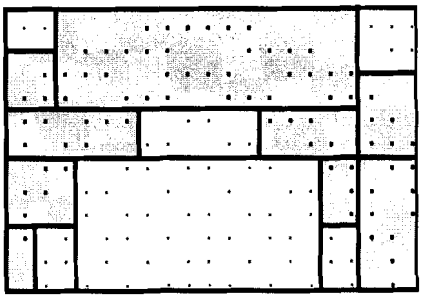
Currently, most studies consider rule level constraints only, and mostly on the consistency of rules. In particular, a rule maybe allowed to cover a certain number of negative examples. This constraint is anti-monotonic and our algorithm *RGB* can adapt to this scenario with no effort. We can simply use the user-defined consistency threshold when checking consistency in updating the assignment graph.

It is probably not worth the effort or not possible to effectively adapt *RGB* to other types of constraints, e.g., rule set level constraints or completeness constraints. For such constraints, it is not a bad idea to resort to post-processing. In fact, our algorithm *DesTree* proposed in Chapter 2 can be directly used for this purpose. We can simply send the output from *CAG* to *DesTree*, and after the constraints are satisfied, send the result from *DesTree* back to *RGB* to extract compact rules. The rule extraction part of *RGB* can be kept intact since it would not "worsen" a rule for the consistency and completeness measures that are reasonably defined.

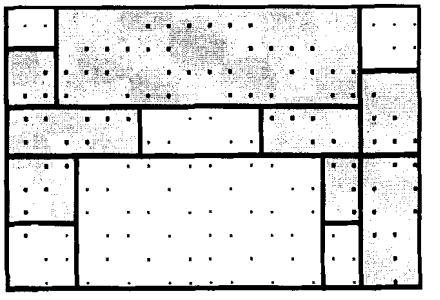
3.8.3 Overfitting issue

In this study, we have focused on the minimality of rule sets. However, it is interesting to evaluate the data classification performance of *RGB*. For this purpose, some issues have to be considered in extending the algorithm, for example, overfitting. Overfitting is a central issue in classification due to the fact that real datasets are noisy. General rule pruning algorithms are surveyed in [Fur97]. Although we do not elaborate on this topic, we provide some promising ideas in the following that can serve as the basis for future research. We feel that in addition to the pre-pruning technique discussed above, our rectangle-based rule learning approach may lead to effective post-pruning.

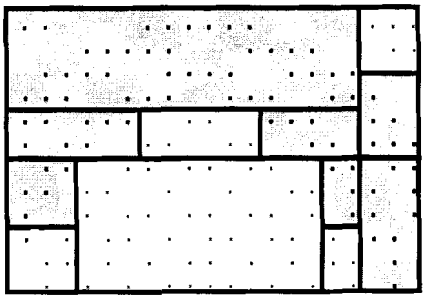
Despite the relative inefficiency, many researchers consider post-pruning more effective than pre-pruning, e.g., Breiman [BFOS84] suggested to build a large complete and consistent decision tree, then selectively prune the branches backwards. Recall that tree induction performs a recursive partitioning on the attribute space, pruning backwards corresponds to merging rectangles, as illustrated in Figure 3.9. The figure demonstrates a typical tree pruning process conducted by *CART* on a toy dataset. In the figure, (a) shows the complete and consistent tree built on the dataset, where dark points indicate positive examples and light-color points indicate negative examples. Along the pruning process step by step from (b) to (f), we can see how rectangles are merged progressively.



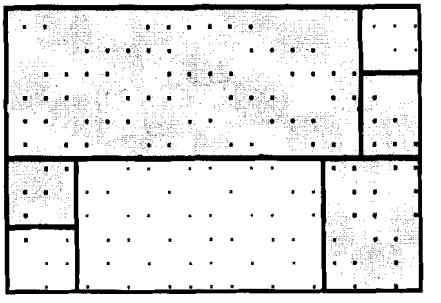
(a) complete and consistent tree



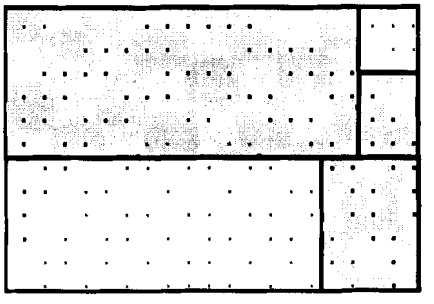
(b) after pruning step 1



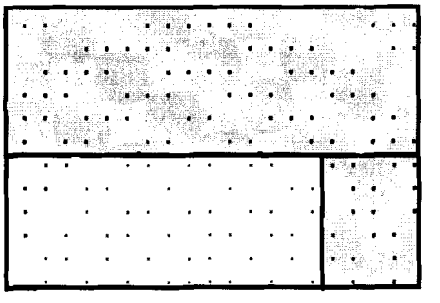
(c) after pruning step 2



(d) after pruning step3



(e) after pruning step 4



(f) after pruning step 5

Figure 3.9: Tree pruning in *CART*.

Our rectangle-based rule learning approach allows rules to be pruned in a similar fashion. As we mentioned in the discussion of admitting approximate models, we can merge the rectangle rules returned by *CAG* using a *DesTree* like algorithm. We can use a global measure, say the F-measure, to control the merging process. The pair of rectangles resulting in the largest F-measure are chosen for the next merge. Compared to tree pruning, rectangles from *CAG* are allowed to overlap and the merging process can be finer-tuned.

One way to overcome overfitting is to utilize multiple solutions. For example, *RAMP* [Apt95] generates multiple solutions that model the decision surface using different combinations of rules. The rationale is that each individual solution models some regions correctly and some incorrectly due to overfitting, then the union of multiple solutions enhances the correctness and smooths out the overfitting. *RGB* can easily generate such multiple rule sets for the purpose of tackling overfitting. We can simply find multiple maximal independent sets, which in general lead to different (rectangle) rule sets.

3.9 Summary

In this chapter, we studied the Minimum Rule Set (MRS) problem, which finds a complete and consistent set of rules with minimized cardinality. We revealed that by transforming the example set into a consistency graph G_c, the MRS problem can be related to the traditional Minimum Clique Partition (MCP) problem, a dual problem of graph coloring on the complementary graph. The optimal solution to the MRS problem is lower-bounded by the optimal solution to the MCP problem, which is further lower-bounded by the independence number of G_c. The MRS and MCP problems, together with the traditional set covering problem, can be considered as instances of the generalized Minimum Consistent Subset Cover (MCSC) problem, with each consistent subset being a consistent rectangle rule, a clique, and a given subset respectively.

We also proposed a novel rule learning algorithm, *RGB*, that is rectangle-based and graph-based with many unique and/or desirable properties. Unlike most rule learners, *RGB* does not follow the separate-and-conquer framework. Instead, it simultaneously learns a complete and consistent set of rectangle rules, and then compact rules are extracted through an efficient general-to-specific beam search. The core procedure of *RGB*, generic *CAG*, starts with a maximal independent set of G_c as a maximal optimal partial solution, and performs an example-driven specific-to-general search on a dynamically maintained bipartite assignment

graph G_a to solve instances of the MCSC problem. G_a contains the necessary information allowing the design of effective evaluation measures, such as weighted edge gain, to guide the search towards a consistent subset cover with small cardinality. Our extensive experiments on UCI benchmarks demonstrated the good performance of *RGB* in the comparison with *AQ21*, a popular representative of separate-and-conquer rule learners, achieving significant reductions in both the number of rules and number of conditions.

As an extension, we discussed how to generalize the MRS problem to admit approximate models, which is important from both data classification and data description points of view. Our proposed generic *CAG* algorithm can be easily adapted to the extended MRS problem.

There are many interesting directions for future work. In Section 3.8, we have discussed possible improvements on *CAG* and *RGB*. It is worthwhile to implement those proposals and experimentally evaluate their effectiveness. Also, we have discussed several MCP construction heuristics $\overline{Greedy\text{-}GC}$, $\overline{DSATUR}$ and $\overline{RLF}$. It is interesting to have them in the *RGB* framework replacing *CAG*, and experimentally compare with the current *RGB*.

Last but not least, we have focused on the minimality of rule sets. It is interesting to see how *RGB* would perform in the task of data classification. Some issues need to be taken care of before this attempt. For example, the ordering of rules, and more importantly, overfitting. In Section 3.8, we discussed the issue of admitting inconsistency, we also proposed promising approaches to tackle overfitting. These proposals need to be implemented and experimentally tested for justification.

Chapter 4

Nearest Rectangle Learning

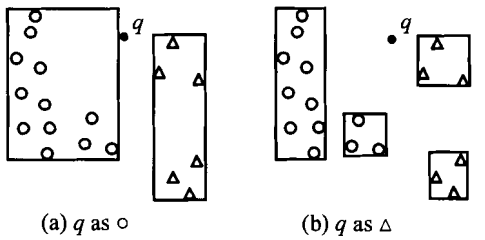
In this chapter, we consider Nearest Rectangle (*NR*) learning and seek for its accuracy improvement. We quantitatively define the concept of *right of inference* that measures the compactness of rectangles and their appropriateness in making inferences. We propose a rectangle learning algorithm with the right of inference of rectangles enforced. We experimentally evaluate the effectiveness of the proposed algorithm on benchmarks. The improved *NR* approach can serve as an interpretable hybrid inductive learning method with competitive accuracy.

4.1 Overview

Nearest Rectangle (*NR*) learning [Sal91] is a hybrid inductive learning framework, where training instances are generalized into axis-parallel hyper-rectangles, and a query (testing instance) is classified according to its nearest rectangle.

NR learners belongs to the class of hybrid lazy-eager learning algorithms. Lazy algorithms such as k-Nearest Neighbor (*kNN*) classifiers [Das91] are instance-based and non-parametric, whereas eager algorithms such as decision trees [BFOS84, Mur98] are model-based and parametric. Lazy algorithms are generally more advantageous in terms of prediction accuracy but less efficient in answering queries. Eager methods try to make predictions that are good on average using a single global model, thus they are generally less accurate than lazy methods but more efficient in answering queries.

Pure lazy learners do not generalize data, thus they are not considered interpretable. Eager learners, however, provide certain degree of comprehensibility. In particular some



In (a), the rectangles make "wild", inappropriate inferences and the query q is classified as ○;

In (b), the rectangles generalize the same training instances in a compact and conservative fashion, having the right of inference, and q is more appropriately classified as △.

Figure 4.1: Right of inference.

eager methods, such as decision trees, generate the most interpretable classification models with axis-parallel decision boundaries. This ease of comprehension is very appealing in decision support related data mining activities, where insight and explanations are of critical importance [AW97].

NR learning, as a hybrid approach, should provide a good trade-off between lazy and eager methods. However, previous *NR* learners were criticized having inferior prediction performance [Wet94, WD95]. Then compared to decision trees, they sacrifice classification efficiency with no gain in prediction accuracy, and this line of research discontinued soon after its introduction.

We study *NR* learning, and propose that the major reason accounting for its loss of accuracy in previous endeavors was that the generalized rectangles were not given the *right of inference* that guarantees the appropriateness of rectangles in making inferences. In general, rectangles having the right of inference should be compact, conservative, and good for making local decisions, as illustrated in Figure 4.1. By imposing the right of inference on rectangles, *NR* classifiers can potentially become an interpretable hybrid inductive learning method with competitive accuracy and many combined appealing properties from decision trees and *kNN* classifiers.

Contributions

- We consider Nearest Rectangle learning and seek for its accuracy improvement through imposing the right of inference of rectangles.
- We define the concept of right of inference in a quantitative manner so as to measure the compactness of rectangles and their appropriateness in making inferences.
- We propose a rectangle generation algorithm with the right of inference enforced. We also experimentally evaluate its effectiveness on benchmarks.

Organization of the chapter

In Section 4.2, we review existing research related to Nearest Rectangle learning. In Section 4.3, we discuss the concept of right of inference and its enforcement. In Section 4.4, we propose *LearnRight*, an *NR* learning heuristic with the right of inference imposed. We report empirical evaluations in Section 4.5 and summarize the chapter in Section 4.6.

4.2 Background

In this section, we introduce Nearest Rectangle (*NR*) learning as a hybrid learning algorithm. We also review the short history of *NR* learning.

In Nearest Rectangle (*NR*) learning [Sal91], training instances are generalized into axis-parallel hyper-rectangles, and a query is classified according to its nearest rectangle. If a query falls inside a rectangle, its distance to that rectangle is zero. If the query lies outside a rectangle, the distance is the (weighted) Euclidean distance from the query to that rectangle. If the query is equidistant to several rectangles, the smallest of which is chosen.

NR learners belongs to the class of hybrid lazy-eager learning algorithms. Lazy algorithms are instance-based and non-parametric, where the training data are simply stored in memory and the inductive process is deferred until a query is given. *kNN* classifiers exemplify the simplest form of lazy learners. In contrast, eager algorithms such as decision trees, neural networks, and naive Bayes classifiers are model-based and parametric, where the training data are greedily compiled into a concise hypothesis (model) and then completely discarded. Obviously, lazy algorithms incur lower computational costs during training but much higher costs in answering queries as well as greater storage requirements, thus they do not scale well to large datasets. In addition, lazy methods do not generate interpretable models as some eager methods, such as decision trees, that can be directly inspected to understand the decision surfaces embedded in data even for non-technical end-users. This ease of comprehension is very appealing in decision support related data mining activities, where insight and explanations are of critical importance [AW97].

However, in terms of accuracy, lazy methods can be more advantageous. They do not lose information since all the training data are retained. They have additional information to utilize, the query instances, so that local and adaptive decisions can be made for predictions. On the other hand, eager methods try to make predictions that are good on average using a single global model.

The characteristics of eager and lazy learners were identified in [Aha97]. Both types of learners have their own desirable properties. To compromise on some of the distinguishing characteristics of purely lazy or eager methods, varied hybrid lazy-eager algorithms are studied. [Qui93b] introduced a method combining instance-based learning and model-based learning. [ZB99] proposed a hybrid approach to overcome the drawback of high prediction time and storage requirements of locally weighted regression. [DY03] utilized an eager learner to build multiple classification models for cluster-based segments of the instance space.

NR learning is also a typical hybrid approach. It partially processes the training instances and generalizes them into hyper-rectangles. These intermediate results are retained and used to answer queries. Nonetheless, the *NR* method has not received much attention mainly because it is not considered accurate enough. The original *NR* learning algorithm as well as several improved versions were experimentally compared with *kNN* [Wet94, WD95], and it was concluded that the *NR* approach performed well in domains with axis-parallel decision boundaries, while in other occasions it was significantly inferior to *kNN* in terms of accuracy. Compared to axis-parallel decision trees, which are essentially rectangle-based, the rectangles induced by *NR* learners also offer a level of intuitive comprehensibility. However, as a hybrid approach, *NR* is slower in answering queries. Then with similar accuracy, it has no advantage over decision trees.

As a hybrid approach, Nearest Rectangle learning was first introduced in [Sal91] under the name of Nested Generalized Exemplar (*NGE*) theory. In *NGE*, an exemplar can be a generalized axis-parallel hyper-rectangle or a single training instance, which is a degenerate (trivial) rectangle. Arbitrary overlapping and nesting of rectangles of different classes are allowed. [Wet94, WD95] challenged the accuracy performance of *NGE* and made several attempts of improvement such as disallowing nesting and/or overlapping, modifying the rectangle construction heuristic, and weighting features (attributes) by mutual information. It was concluded that the major reason leading to the loss of accuracy of *NGE* was the overlapping of rectangles of different classes. Nevertheless, their best improved version was still significantly inferior to *kNN* in most of the tested datasets. We notice that all the above attempts did not pay much attention to the quality of the generated rectangles. They allowed rectangles to make wild and inappropriate inferences, which significantly affects the accuracy as illustrated in Figure 4.1.

4.3 Right of Inference and its Enforcement

In this section, we introduce the concept of right of inference that measures the compactness of rectangles and their appropriateness in making inferences. We also discuss the corresponding enforcement mechanisms.

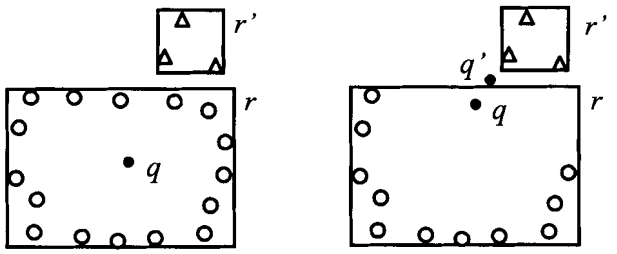
4.3.1 Right of inference

Rectangle-based classifiers can provide a certain degree of insight and understanding into data and the instance space. In fact, if we consider a closed rectangular instance space, the leaf nodes of an axis-parallel decision tree correspond to a set of isothetic rectangles (not bounding boxes) that form a partition of the instance space. The induction of decision trees generalizes the training data and makes inferences to the entire instance space simultaneously with the disjointness constraint, striving to achieve good-on-average predictions. Intuitively, if we separate generalization and inference into two serial phases and allow same-class rectangles to overlap, we should be able to build classifiers that are more flexible, adaptive, and accurate, with the capacity to make local decisions.

Potentially, *NR* learning can induce such explanatory, adaptive, and accurate classifiers. However, if in the generalization phase, the rectangles are not constructed in a conservative and compact fashion, they would make wild and improper inferences as demonstrated in Figure 4.1 (a). In fact, decision trees would suffer from the same problem. A typical decision tree constructed from the training data shown in Figure 4.1 (a) would make the same improper decision for the query q. On the contrary, Figure 4.1 (b) illustrates some compact rectangles for the same training data. These rectangles are good for making local decisions and considered having the so-called right of inference.

The *right of inference* of a rectangle can be conceptually considered as the appropriateness for the rectangle to make sound and local inferences. As we have seen, rectangles having the right of inference should appear compact and saturated. So, how should we define right of inference in a quantitative manner?

DEFINITION 4.1. (*right of inference*) *A rectangle r has the right of inference if and only if for any query q outside of r, $dist(q, q^k) - dist(q, r) \leq \delta$, where $dist(q, q^k)$ is the Euclidean distance from q to its kth nearest instance in r, $dist(q, r)$ is the Euclidean distance from q to r, and δ is the distance threshold.*



(a) false invalidation (b) true invalidation

Figure 4.2: False and true invalidation.

The distance from a query q to a rectangle r is equivalent to the length of the line dropped (perpendicularly) from q to the nearest face, edge, or vertex of r. In the following, we formally define this point-to-rectangle distance, where rectangle weighting and feature weighting are not considered. Let q_{f_i} denote the value of q on the ith feature, where $1 \leq i \leq m$ and m is the dimensionality. Let r_{lower,f_i} and r_{upper,f_i} denote the lower and upper end values of r on the ith feature. We define

$$dist(q,r) = \sqrt{\sum_{i=1}^{m} dif_i^2} \qquad \text{where}$$

$$dif_i = \begin{cases} q_{f_i} - r_{\text{upper},f_i} & \text{when } q_{f_i} > r_{\text{upper},f_i} \\ r_{\text{lower},f_i} - q_{f_i} & \text{when } q_{f_i} < r_{\text{lower},f_i} \\ 0 & \text{otherwise} \end{cases}$$

What is the rationale behind the right of inference thus-defined? Note that we always have $dist(q,r) \leq dist(q,q^k)$. If $dist(q,q^k) - dist(q,r)$ is too large, the inference on the class of q from r might be inappropriate. This is because $dist(q,r)$ would alter (bring closer) the locality of the instances in r with respect to q in an intolerable manner. For example, in Figure 4.1 (a), q is very close to the left rectangle but far away from the instances in it. In contrast, if $dist(q,q^k) - dist(q,r)$ is reasonably small, *NR* classifiers would behave similarly to *kNN*, as shown in Figure 4.1 (b).

In Definition 4.1, we only consider queries lying outside of the rectangle r. This is because some inside query may falsely invalidate a "good" r. In Figure 4.2 (a), even if $dist(q,q^k) - dist(q,r)$ is rather large, r is "good" because q would not be closer to other

instances/rectangles of different classes, say r'. Recall that overlapping of rectangles is allowed only if they have the same class label. On the other hand, for a "bad" r as shown in Figure 4.2 (b), not considering q or other queries in r would not falsely validate r. This is because if q should invalidate r due to the fact that it is closer to r', a rectangle of different class, there must be another q' outside of r that is even closer to r' and can serve for the purpose of invalidating r. Rectangles r and r' do not overlap since they have different class labels; then, we can easily find such a q', say, outside of r and on the line joining q and r', as demonstrated in Figure 4.2 (b).

In Definition 4.1, it is also reasonable to use the average of distances from q to its k nearest instances in r for $dist(q, q^k)$. Note that k is limited by the total number of instances in r, and the choice of k can be a legitimate research issue just as in the case of *kNN* classification. The distance threshold δ has a direct impact on how similar *NR* classifiers would behave to *kNN*. If δ is too large, the allowed rectangles tend to be very large, making wild and improper inferences. If δ is too small, *NR* learning would induce many small rectangles and become "lazy", losing the desirable properties as a hybrid learner. In the extreme case of $\delta = 0$, *NR* learning would lose the generalization capacity completely and essentially become 1-*NN*.

4.3.2 Enforcing right of inference

It is not straightforward to enforce the right of inference defined in Definition 4.1 since there are potentially an infinite number of queries to examine. In the following, we discuss some helpful observations and their practical implications.

THEOREM 4.1. *If for any query q that is on the surface of a rectangle r, $dist(q, q^k) \leq \delta$, where q^k is the kth nearest instance of q in r, then r has the right of inference defined in Definition 4.1 with respect to δ.*

Proof. Let p be any query outside of r. Let p^k be the *kth* nearest instance of p in r and q the projected p on the nearest face of r, as depicted in Figure 4.3. According to the definition of point-to-rectangle distance, $dist(p, q) = dist(p, r)$. We use arc_p to denote the intersection of r and the sphere with radius $dist(p, p^k)$ centered at p. We use arc_q to denote the intersection of r and the sphere with radius $dist(p, p^k) - dist(p, q)$ centered at q. Clearly, $arc_q \subseteq arc_p$.

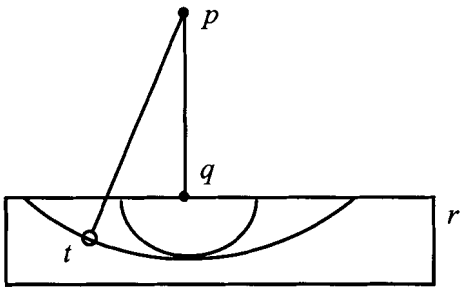


Figure 4.3: Proof of Theorem 4.1.

Since p^k is the *kth* nearest instance of p in r, the number of instances in arc_p is less or equal to k if not considering ties. Since arc_p and arc_q intersect on only one point, the number of instances in arc_q is less or equal to k even considering ties. That is to say, q^k, the *kth* nearest instance of q in r, lies outside or on the surface of arc_q. In other words, $dist(q, q^k) \geq dist(p, p^k) - dist(p, q) = dist(p, p^k) - dist(p, r)$. Therefore, $\delta \geq dist(q, q^k) \Rightarrow \delta \geq dist(p, p^k) - dist(p, r)$, and the conclusion of Theorem 4.1 follows.

The implication of Theorem 4.1 is that we only need to consider queries on the surface of r to test its right of inference. In the following, we discuss our right of inference enforcement heuristic. We also discuss the ideas on how to choose the distance threshold δ.

Sketch of enforcement heuristic

In our prototype *NR* learner *LearnRight*, to be proposed shortly, a simple recursive testing and bisecting enforcement heuristic is embedded. For testing the right of inference of a rectangle r, a query pool is constructed consisting of a constant number of queries generated according to a ranking scheme that gives high ranks to queries with high probability of invalidating r. Generally, highly ranked queries include vertices that are far away from the mean of the instances in r. Some certain positions (say, centers) on long edges or large faces have the next priority to be inserted into the query pool, and some uncertain (random) positions on the surface of r are the next to be considered. The rectangle r is validated (passes the test) if it is not invalidated by any query in the query pool.

If r is invalidated, the k-means clustering algorithm with $k = 2$ is applied to bisect the instances in r, and the newly generated rectangles (bounding boxes of the two sections) are to be tested separately. This recursive testing and bisecting process terminates until all the rectangles pass the test.

Choice of distance threshold

For the choice of the distance threshold δ, we randomly sample a series of queries. For each query q, we record $dif_q = dist(q, t_q) - dist(q, \overline{t_q})$, where t_q denotes the nearest training instance of q and $\overline{t_q}$ denotes the nearest training instance of q that has a different class label from t_q. If we set $\delta = dif_q$, the resulting *NR* classifier behaves the same as 1-*NN* on q and q will not be assigned a class label other than the one of t_q. To see why, let r_{t_q} and $r_{\overline{t_q}}$ be any two rectangles covering t_q and $\overline{t_q}$ respectively, then $dist(q, r_{t_q}) \leq dist(q, t_q)$ and $dist(q, \overline{t_q}) - dist(q, r_{\overline{t_q}}) \leq \delta$ if $r_{\overline{t_q}}$ is enforced the right of inference, from which $dist(q, r_{t_q}) \leq dist(q, r_{\overline{t_q}})$ follows. Intuitively, since the right of inference is enforced on $r_{\overline{t_q}}$, $\overline{t_q}$ will not be brought close enough by the rectangular generalization to challenge the locality of t_q with respect to q. Note that t_q is also brought closer to q by r_{t_q}. After obtaining a series of dif_q's, we take their average and use the average value for δ.

While how to decide δ deserves further investigations, a practical situation would be we are given a upper bound constraint on the complexity of the classification models to generate, where the complexity is measured by the total number of rectangles. Since the number of generated rectangles increases roughly monotonically with the decrease of δ, the upper bound constraint on model complexity can be transformed as a lower bound constraint on δ, which can be used for the choice of δ. Note that this constraint transformation needs to be done by experiments.

4.4 *LearnRight*: Learning Right Rectangles

LearnRight heuristically constructs a set of rectangles with small cardinality and enforced right of inference. The rectangles generalize the given training instances with 100% accuracy, and only same-class rectangles are allowed to overlap.

Note that without considering the right of inference enforcement, *LearnRight* is an extension of *Learn2Cover*, presented in Section 2.5 of Chapter 2, from 2 classes to multiple classes. Such an extension was also referred to as *LearnCovers* in Section 3.7 of Chapter 3. The main ideas and techniques used in *Learn2Cover* remain in *LearnCovers* and *LearnRight*, such as how to calculate covering cost for the choice of rectangles and how to perform violation checking. While in we present the pseudocode for *LearnRight* and explain how it works in the following, we do not repeat the explanations for those ideas and techniques used in *Learn2Cover*. Interested readers may check Section 2.5 for details.

Algorithm: ***LearnRight***

Input: T and δ: T is the set of training instances. δ is the distance threshold.
Output: R: a set of rectangles with the right of inference enforced. Each rectangle covers some instances of the same class only. Rectangles are allowed to overlap only if they have the same class label.

```
1  R = ∅; // initialize R to be empty
2  sort T; // training instances in T will be processed in the sorted order
3  for each t ∈ T
4      for each r ∈ R
5          calculate cost(r, t); // r with smaller cost(r, t) is favored in covering t
6          if (r.class != t.class && cost(r, t) == 0)
7              validateToclose(r); // r can be closed only after validation with respect to δ
8          end if
9          if (r.class == t.class && r is not closed && cost(r, t) == 0)
10             extend r to cover t and continue to process the next t; // back to line 3
11         end if
12     end for
13     for each r ∈ R // t was not covered; fetch r in ascending order of cost(r, t)
14         if (r.class == t.class && r is not closed && violationCheck(r, R) == no)
15             expand r to cover t and continue to process the next t; // back to line 3
16         end if
17     end for
18     insert(R, r_new); // t cannot be covered; insert a degenerate rectangle r_new into R
19 end for
20 enforce(R); // with respect to δ
```

The rectangle set R is initialized to be empty (line 1). Instances in the given training set T are sorted along a selected feature (line 2) and processed in the sorted order. For each training instance t (line 3), we search through R (line 4) for the best rectangle to accommodate it. Expanding rectangles would incur *covering violations*, i.e., overlapping of rectangles of different classes, which are not allowed. The best rectangle to cover t is the one with the smallest *covering cost* with respect to t, which is defined such that the rectangles keep the maximum potential in covering future instances without incurring covering violations.

In line 5, the covering cost of rectangle r in covering instance t, $cost(r, t)$, is calculated. $cost(r, t) = 0$ only if t lies straight under r, i.e., by simply extending r along the selected sorting feature, t will be covered by r. If $cost(r, t) = 0$ and r and t are of different classes (line 6), r cannot be used to cover further instances after t without causing a covering

violation. Thus r will be marked as "closed" on condition that it can be validated, i.e., passing the right of inference test. If the validation fails, r will be bisected and the two propagated rectangles will be inserted into R (line 7). Closed rectangles in R will not be considered in the remaining procedures.

If a non-closed rectangle r has the same class label as instance t with $cost(r,t) = 0$ (line 9), r is an optimal rectangle for covering t. We can simply stop searching and continue to process the next instance (line 10). Note that in this case, violation checking is unnecessary since instances in T are sorted and we only need to extend r along the sorting feature to cover t.

If instance t has not been covered by such an optimal rectangle (line 13), we need to search through R for the best r with the smallest $cost(r,t)$. The rectangles in R will be considered in the ascending order of $cost(r,t)$, the first available one (line 14) will be used to cover t and we can continue to process the next instance (line 15) afterwards.

If there is no such rectangle $r \in R$ that can cover instance t without incurring a covering violation, a degenerate rectangle r_{new} minimally covering t will be constructed and inserted into R (line 18).

Upon reaching line 20, all the training instances in T have been processed and generalized. The enforcement heuristic discussed previously is applied to all non-closed rectangles in R (closed ones must have been validated), and all the recursively propagated rectangles will be inserted into R after validation. In the actual implementation, we have chosen $k = 1$ for testing the right of inference, that is, any query q on the surface of r with $dist(q, q^1) > \delta$ will invalidate r.

4.5 Empirical Results

A series of experiments were conducted to evaluate the accuracy performance of the proposed *NR* learner *LearnRight*. The notion of rectangle can be extended to tolerate categorical features but not in this prototype version, thus 20 numerical benchmark datasets without missing values from the UCI repository [BM98] were used to run *C4.5* [Qui93a], 1-*NN*, *kNN* and *LearnRight*. For *kNN*, the highest accuracy values were recorded. The datasets were normalized on each feature. For each of the datasets where cross-validation was needed, the averaged result over 3 runs of stratified 10-fold cross-validation was taken. The experimental results are presented in Table 4.1.

Table 4.1: *LearnRight* results

Dataset	*ins*	*dim*	*cla*	*C4.5*	1-*NN*	*kNN*	*LearnRight*
balance	625	4	3	0.758	0.790	0.900	0.828
bupa	345	6	2	0.655	0.632	0.652	0.672
car	1728	6	4	0.917	0.917	0.951	0.938
ecoli	336	7	8	0.841	0.806	0.871	0.869
glass	214	10	6	0.687	0.701	0.712	0.739
iono	351	34	2	0.900	0.869	0.869	0.937
iris	150	4	3	0.953	0.953	0.967	0.973
letter	20000	16	26	0.868	0.955	0.955	0.925
new-thyr	215	5	3	0.916	0.968	0.968	0.953
page-blo	5473	10	5	0.965	0.957	0.959	0.971
pima	768	8	2	0.737	0.701	0.738	0.752
satimage	6435	36	6	0.850	0.894	0.906	0.878
segment	2310	19	7	0.960	0.974	0.974	0.966
sonar	208	60	2	0.702	0.865	0.865	0.794
spambase	4601	57	2	0.895	0.908	0.908	0.897
vehicle	846	18	4	0.734	0.696	0.725	0.709
vowel	990	10	11	0.788	0.989	0.989	0.973
waveform	5000	21	3	0.781	0.809	0.833	0.808
wine	178	13	3	0.936	0.949	0.972	0.977
yeast	1484	8	10	0.494	0.526	0.574	0.581
average				0.817	0.843	0.864	0.857

In Table 4.1, *ins*, *dim*, and *cla* indicate the numbers of instances, the number of features, and the number of classes respectively for the datasets. As shown by the results, *LearnRight* outperforms *C4.5* in 19, 1-*NN* in 12, and *kNN* in 8 of the 20 datasets. *LearnRight* has the averaged accuracy of 0.857, significantly higher than *C4.5* (0.817), better than 1-*NN* (0.843) and comparable to *kNN* (0.864).

Recall that without the consideration of right of inference but with the assistance of some other sophisticated techniques such as rectangle weighting and feature weighting using mutual information, the remedies proposed in [Wet94, WD95] for the original *NR* learner only achieved moderate accuracy improvement and remained "significantly inferior to *kNN*". In contrast, with the consideration of right of inference but without the assistance of those sophisticated techniques, *LearnRight* achieved comparable results to *KNN*.

4.6 Summary

In this chapter, we considered Nearest Rectangle (*NR*) learning and sought for its accuracy improvement through imposing the right of reference on rectangles. Compared to decision trees, previous *NR* learning algorithms do not have the accuracy advantage even if as a hybrid learner, they have sacrificed classification time. In particular, we quantitatively defined the concept of right of reference and discussed its enforcement mechanisms. We also presented a rectangle generation algorithm, *LearnRight*, with the enforcement heuristic embedded. Our experimental evaluation shows that by imposing the right of inference on rectangles, the *NR* approach can potentially become an interpretable hybrid inductive learning method with competitive accuracy and many combined appealing properties from decision trees and *kNN* classifiers.

For future work, more effective and efficient testing and enforcement mechanisms should be investigated. The choice of distance threshold also remains as an open issue that requires further exploration. In addition, grounded on the right of inference of rectangles, there are several interesting directions to further extend *NR* learning. One is to consider k nearest (weighted) rectangles. Another is to consider multiple rectangle sets (e.g., obtained from *LearnRight* with different choices of sorting feature), as analogous to decision forests.

Chapter 5

Conclusion

In this chapter, we conclude the dissertation with an overall review and a general discussion on future work.

5.1 Review of Dissertation

In this dissertation, we focused on the interpretability perspective of data mining and studied the fundamental problem of rectangle-based discriminative data generalization in the context of several useful data mining applications: cluster description, rule learning, and Nearest Rectangle classification.

For the application of cluster description, we proposed and analyzed novel description formats, SOR^{-} and $kSOR^{\pm}$, providing enhanced expressive power. We introduced and defined novel description problems, Maximum Description Accuracy and Minimum Description Length, allowing users to specify alternative trade-offs between interpretability and accuracy. We also presented efficient heuristic algorithms for the introduced problems in the proposed formats, together with their experimental evaluations.

Cluster descriptions are patterns which can be stored as tuples in a relational table, so that a clustering and its associated clusters become queriable data mining objects. Therefore, this research can serve as a first step for integrating clustering into the framework of inductive databases [IM96], a paradigm for query-based "second-generation" database mining systems.

For the application of rule learning, we studied the Minimum Rule Set (MRS) problem, finding a complete and consistent set of rules with the minimum cardinality. We established

that by transforming the example set into a consistency graph, the MRS problem can be related to the traditional Minimum Clique Partition (MCP) problem, a dual problem of graph coloring on the complementary graph. The MRS and MCP problems, together with the traditional set covering problem, can be generalized into the so-called Minimum Consistent Subset Cover (MCSC) problem, finding the minimum number of consistent subsets that cover a given ground set, where a subset is consistent if it satisfies a given constraint.

We proposed *RGB*, a novel rule learning algorithm that is rectangle-based and graph-based with many unique and/or desirable properties. The core procedure of *RGB*, generic *CAG*, starts with a maximal independent set of the consistency graph as an optimal partial solution, then performs an example-driven specific-to-general search on a dynamically maintained bipartite assignment graph to solve MCSC instances with anti-monotonic constraints. Extensive experiments on benchmark datasets justified the performance of *RGB*.

For the application of Nearest Rectangle (*NR*) classification, we investigated the source of error associated with existing *NR* learners and proposed that the lack of right of reference of rectangles may lead to inappropriate inferences and inaccurate classifications. We experimentally showed that by enforcing the right of inference, *NR* learning can potentially become an interpretable hybrid inductive learning method with competitive accuracy as well as many combined appealing properties from decision trees and k-Nearest Neighbor classifiers.

5.2 Future Work

In the three applications of rectangle-based discriminative data generalization, we have proposed many different algorithms. A common operation involved in these algorithms is consistency-checking. Although we have used rectangles to effectively reduce the number of instances to be considered, the efficiency can be further improved if some auxiliary indexing structure is introduced, such as the lean tree [BKK96] implementation used in algorithm *BP* [LNW+02].

Interesting directions for future work have been discussed individually for each of the three applications. Among them, we are particularly interested in continuous research on rule learning, seeking improvements and extensions of the *CAG* and *RGB* algorithms. While many approaches have been proposed and discussed in Section 3.8 of Chapter 3, these proposals need to be further investigated, implemented, and experimentally evaluated.

As demonstrated in our experiments, *RGB* can significantly reduce the model complexity of rule sets, no matter the complexity is measured by the number of rules or the number of conditions. By the principle of Occam's Razor, simpler models should lead to an accuracy gain in classifying unseen data. To extend *RGB* into a competitive classifier, there are many design issues that need to be seriously considered. In Section 3.8 of Chapter 3, we have proposed promising pre-pruning and post-pruning techniques for the purpose of tackling overfitting. The boundaries of rules also need to be adjusted since currently rules are derived from bounding boxes. Another important design issue is the decision making strategy for instances that are uncovered or covered by multiple rules. We leave this interesting research direction of exploring the classification potential of *RGB* for future work.

In this dissertation, we have focused exclusively on the rectangle-based discriminative data generalization. In certain applications, some other geometric structures, such as spheres, may be favorable. For such applications, the specification of each dimension is not required. Many algorithms proposed in this dissertation may only need to be slightly modified to perform sphere-based discriminative data generalization. We also leave this interesting research direction for future work.

Bibliography

[ABMP98] M. Alekhnovich, S. Buss, S. Moran, and T. Pitassi. Minimum propositional proof length is NP-hard to linearly approximate. *Manuscript*, 1998.

[ADT95] R. Andrews, J. Diederich, and A. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, pages 373–389, 1995.

[AGDP98] R. Agrawal, J. Gehrke, D.Gunopulos, and P.Paghavan. Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD International Conference on Management of Data*, 1998.

[AGM04] F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. *ACM International Conference on Knowledge Discovery in Databases*, 2004.

[AH05] M. Adler and B. Heeringa. Approximating optimal decision trees. Technical Report 05-25, University of Massachusetts Amherst, 2005.

[Aha97] D. Aha. Lazy learning. *Editorial, Artificial Intelligence Review*, 11:7–10, 1997.

[AHP05] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[Apt95] S. Hong S. Prasad B. Rosen Apte, C. RAMP: Rules abstraction for modeling and prediction. Technical report, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598, 1995.

[AW97] C. Apte and S.M Weiss. Data mining with decision trees and decision rules. *Future Generation Computer Systems*, 13:197–210, 1997.

[Bay98] R. Bayardo. Efficiently mining long patterns from databases. *SIGMOD*, 1998.

[BBP02] S. Busygin, S. Butenko, and P. Pardalos. A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. *Journal of Combinatorial Optimization*, 6(3):287–297, 2002.

[BD97] P. Berman and B. Dasgupta. Approximating rectilinear polygon cover problems. *Algorithmica*, 17:331–356, 1997.

[BD05] N. Barakat and J. Diederich. Eclectic rule-extraction from support vector machines. *International Journal of Computational Intelligence*, 2(1):59–62, 2005.

[BEHW87] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36:929–965, 1989.

[BFOS84] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[BKK96] S. Berchtold, D. Kiem, and H.P. Kriegel. The X-tree: an index structure for high-dimensional data. *International Conference on Very Large Data Bases*, 1996.

[BM96] E Bloedorn and R. S. Michalski. The AQ17-DCI system for data-driven constructive induction and its application to the analysis of world economics. *International Syposium on Methodologies for Intelligent Systems*, pages 108–117, 1996.

[BM98] C. Blake and C. Merz. UCI repository of machine learning databases. 1998.

[Bos95] H. Bostrom. Covering vs. divide-and-conquer for top-down induction of logic programs. *International Joint Conference on Artificial Intelligence*, pages 1194–1200, 1995.

[BR91] M. Berger and I. Regoutsos. An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man and Cybernetics*, 21:5:1278–86, 1991.

[Bre79] D. Brelaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[BS95] N.C. Berkman and T.W. Sandholm. What should be minimized in a decision tree: a re-examination. Technical Report 95-20, Computer Science Department, University of Massachusetts at Amherst, 1995.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley-Longman: Harlow, UK, 1999.

[CB91] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. *European Working Session on Learning*, pages 151–163, 1991.

[CCC+98] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed steiner problems. *ACM-SIAM Symposium on Discrete Algorithms*, pages 192–200, 1998.

[CCFM97] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *STOC*, 1997.

[CDKM00] R.D. Carr, S. Doddi, G. Konjevod, and M. Marathe. On the red-blue set cover problem. *ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[Cen87] J. Cendrowska. Prism: analgorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27:349–370, 1987.

[Che04] F. Chen. Learning accurate and understandable rules from SVM classifiers. *Master's Thesis. School of Computing Science, Simon Fraser University*, 2004.

[Chi05] M. Chiarandini. Stochastic local search methods for highly constrained combinatorial optimization problems. *Ph.D. Thesis. FG Intellektik, FB Informatik, TU Darmstadt*, 2005.

[Cho92] P. Chou. Optimal partitioning for classification and regression trees. *IEEE Transactions on Pattern Analysis and Machine Learning*, 13(4):340–354, 1992.

[CN89] P. Clark and R. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–284, 1989.

[Coh93] W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. *International Joint Conference on Artificial Intelligence*, 1993.

[Coh95] William W. Cohen. Fast effective rule induction. *International Conference on Machine Learning*, 1995.

[CQK89] L. Cox, Y. Qiu, and W. Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations Research*, 21(1):1–30, 1989.

[Cra96] M.W. Craven. Extracting comprehensible models from trained neural networks. *Ph.D. Thesis. Department of Computer Science, University of Wisconsin-Madison*, 1996.

[CS94] M.W. Craven and J.W. Shavlik. Using sampling and queries to extract rules from trained neural networks. *International Conference on Machine Learning*, pages 37–45, 1994.

[CS96] M.W. Craven and J.W. Shavlik. Extracting tree-structured representations of trained networks. *Advances in Neural Information Processing Systems*, pages 24–30, 1996.

[CSD05] M. Chiarandini, T. Stutzle, and I. Dumitrescu. Stochastic local search algorithms for the graph colouring problem. *Approximation Algorithms and Metaheuristics; Computer and Information Science Series*, 2005.

[CT91] I. Cleote and H. Theron. CID3: an extension of ID3 for attributes with ordered domains. *South African Computer Journal*, 4:10–16, 1991.

[Cul01] J. Culberson. Hidden solutions, tell-tales, heuristics and anti-heuristics. *IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence*, pages 9–14, 2001.

[Dar01] A. Darbari. Rule extraction from treained ANN: a survey. Technical report, TU Dresden, Institute of Artificial Intelligence, Department of Computer Science, 2001.

[Das91] B.V. Dasarathy. Nearest Neighbor (NN) norms: NN pattern classification techniques. *IEEE Computer Society Press*, 1991.

[DB04] R. Cunningham Dain, O. and S. Boyer. IREP++, a faster rule learning algorithm. *SIAM International Conference on Data Mining*, 2004.

[DbSS+05] J.M. Diaz-banez, C. Seara, J.A. Sellares, J. Urrutia, and I. Ventura. Covering point sets with two convex objects. *European Workshop on Computational Geometry*, 2005.

[DK99] Y. Dodis and S. Khanna. Design networks with bounded pairwise distance. *ACM Symposium on Theory of Computing*, pages 750–759, 1999.

[dM93] T. de Merckt. Decision trees in numerical attribute spaces. *International Joint Conferences on Artificial Intelligence*, pages 1016–1021, 1993.

[Dom94] P. Domingos. The RISE system: Conquering without separating. *IEEE International Conference on Tools with Artificial Intelligence*, 1994.

[Dom99] Pedro Domingos. The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

[dW90] D. de Werra. Heuristics for graph coloring. *Computing Supplement*, 7:191–208, 1990.

[DY03] I. Davidson and K. Yin. Semi-lazy learning: combining clustering and classifiers to build more accurate models. *International Conference on Machine Learning; Models, Technologies and Applications*, 2003.

[EGG+06] Martin Ester, Rong Ge, Byron J. Gao, Zengjian Hu, and Boaz Ben-Moshe. Joint cluster analysis of attribute data and relationship data: the connected k-center problem. *SIAM International Conference on Data Mining*, 2006.

[EHH98] A. E. Eiben, J. K. Van Der Hauw, and J. I. Van Hemert. Graph coloring with adaptive evolutionary algorithms. *Journal of Heuristics*, 4(1):25–46, 1998.

[EHL+02] J. Eckstein, P.L. Hammer, Y. Liu, M. Nediak, and B. Simeone. The maximum box problem and its application to data analysis. *Computational Optimization and Applications*, 23:285–298, 2002.

[Fei98] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[FI90] U Fayyad and K. Irani. What should be minimized in a decision tree. *National Conference on Artificial Intelligence*, pages 749–754, 1990.

[FI92] U Fayyad and K. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[Fra89] D. Franzblau. Performance guarantees on a sweep-line heuristic for covering rectilinear polygons with rectangles. *SIAM Journal on Discrete Mathematics*, 2:307–321, 1989.

[FSR05] G. Fung, S. Sandilya, and R.B. Rao. Rule extraction from linear support vector machines. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 32–40, 2005.

[Fur97] Johannes Furnkranz. Pruning algorithms for rule learning. *Machine Learning*, 27(2):139–171, 1997.

[Fur99] Johannes Furnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, February 1999.

[FW94] J. Fürnkranz and G. Widmer. Incremental reduced error pruning. *International Conference on Machine Learning*, 1994.

[FW98] Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. *International Conference on Machine Learning*, 1998.

[GE06a] Byron J. Gao and Martin Ester. Cluster description formats, problems, and algorithms. *SIAM International Conference on Data Mining*, 2006.

[GE06b] Byron J. Gao and Martin Ester. Right of inference: Nearest rectangle learning revisited. *The 17th European Conference on Machine Learning*, 2006.

[GE06c] Byron J. Gao and Martin Ester. Turning clusters into patterns: Rectangle-based discriminative data description. *IEEE International Conference on Data Mining*, 2006.

[GGRL99] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Loh. BOAT–optimistic decision tree construction. *ACM SIGMOD International Conference on Management of Data*, 1999.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the Theory of NP-completeness.* W.H. Freeman: New York, 1979.

[GKR98] N Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the steiner group tree problem. *ACM-SIAM Symposium on Discrete Algorithms*, pages 253–259, 1998.

[GL96] J. Gudmandsson and C. Levcopoulos. Close approximations of minimum rectangular covering. *FST & TCS*, 1180:135–146, 1996.

[GM97] M. Goldwasser and R. Motwani. Intractability of assembly sequencing: Unit disks in the plane. *International Workshop on Algorithms and Data Structures*, pages 307–320, 1997.

[Han81] D.J. Hand. *Discrimination and Classification.* Wiley, Chichester, UK, 1981.

[HCO74] S.J. Hong, R.G. Cain, and D.L. Ostapko. MINI: A heuristic approach for logic minimization. *IBM journal of Research and Development*, 18:443–458, 1974.

[HCO87] S.J. Hong, R.G. Cain, and D.L. Ostapko. MINI: A heuristic algorithm for two-level logic minimization. *Selected Papers on Logic Synthesis for Integrated Circuit Design*, 1987.

[Hoc97] D. S. Hochbaum. *Approximation algorithms for NP-hard problems.* 94-143. PWS Publishing Company: Boston, 1997.

[Hon97] Se June Hong. R-MINI: An iterative approach for generating minimal rules from examples. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):709–717, 1997.

[HR76] L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[HV06] B. Baesens Huysmans, J. and J. Vanthienen. Using rule extraction to improve the comprehensibility of predictive models. Technical report, FETEW Research Report KBI-0612, K.U.Leuven, 2006.

[HWLT02] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. *ICDM*, 2002.

[IM96] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.

[JAMS89] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; Part I, graph partitioning. *Operations Research*, 37(6):865–892, 1989.

[JM82] A. Johri and D. Matula. Probabilistic bounds and heuristic algorithms for coloring large random graphs. Technical report, South Methodist University, Dallas, TX, USA, 1982.

[Joh74a] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[Joh74b] D.S. Johnson. Worst-case behavior of graph-coloring algorithms. *South-Eastern Conference on Combinatories, Graph Theory and Computing*, pages 513–528, 1974.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[Ker92] R. Kerber. Chimerge: Discretization of numeric attributes. *National Conference on Artificial Intelligence*, pages 123–128, 1992.

[KK76] A.V. Kulkarni and L.N. Kanal. An optimization approach to hierarchical classifier design. *International Joint Conference on Pattern Recognition*, 1976.

[KK78] A.V. Kulkarni and L.N. Kanal. Admissible search strategies for parametric and non-parametric hierarchical classifiers. *International Joint Conference on Pattern Recognition*, 1978.

[KM99] Kenneth A. Kaufman and Ryszard S. Michalski. Learning from inconsistent and noisy data: The AQ18 approach. *International Syposium on Methodologies for Intelligent Systems*, 1999.

[KP83] J. Krarup and P. Pruzan. The simple plant location problem: survey and synthesis. *European Journal of Operations Research*, 12:36–81, 1983.

[KR90] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons: New York, 1990.

[KR99] V.S. Anil Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. *ACM Symposium on Theory of Computing*, 1999.

[KW98] S.O. Krumke and H.C. Wirth. On the minimum label spanning tree problem. *Information Processing Letters*, 66:2:81–85, 1998.

[Lei79] F.T. Leighton. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, 84(6):489–506, 1979.

[LG97] C. Levcopoulos and J. Gudmundsson. Approximation algorithms for covering polygons with squares and similar problems. *RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science*, 1997.

[LN03] Y. Liu and M. Nediak. Planar case of the maximum box and related problems. *Canadian Conference on Computational Geometry*, pages 11–13, 2003.

[LNW+02] L.V.S. Lakshmanan, R.T. Ng, C.X. Wang, X. Zhou, and T.J. Johnson. The generalized MDL approach for summarization. *International Conference on Very Large Data Bases*, 2002.

[Mas79] W.J. Masek. Some NP-complete set covering problems. 1979.

[MBGV05] D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen. Adding comprehensibility to support vector machines using rule extraction techniques. *Credit Scoring and Credit Control IX*, 2005.

[Mic69] R.S. Michalski. On the quasi-minimal solution of the general covering problem. *International Symposium on Information Processing*, pages 125–128, 1969.

[Mit97] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[MK01] Ryszard S. Michalski and Kenneth A. Kaufman. Learning patterns in noisy data: The AQ approach. *Machine Learning and Its Applications, Advanced Lectures*, pages 22–38, 2001.

[ML86] I. Mozetic J. Hong Michalski, R.S. and N. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *National Conference on Artificial Intelligence*, 1986.

[MM73] W.S. Meisel and D.A. Michalopoulos. A partitioning algorithm with application in pattern classification and optimization of decision trees. *IEEE Transactions on Computers*, C-22:93–103, 1973.

[MM91] O.J. Murphy and R.L. McCraw. Designing storage efficient decision trees. *IEEE Transactions on Computers*, 40(3):315–319, 1991.

[MP94] P.M. Murphy and M.J. Pazzani. Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction. *Journal of Artificial Intelligence Research*, 1:257–275, 1994.

[MP03] A.Q. Mendelzon and K.Q. Pu. Concise descriptions of subsets of structured sets. *ACM Symposium on Principles of Database Systems*, 2003.

[MRA95] M. Mehta, J. Rissanen, and R. Agrawal. MDL-based decision tree pruning. *International Conference on Knowledge Discovery and Data Mining*, 1995.

[MS00] C.N. De Meneses and C.C. De Souza. Exact solutions of rectangular partitions via integer programming. *International Journal of Computational Geometry and Applications*, 10(5):477–522, 2000.

[Mur95] P.M. Murphy. An empirical analysis of the benefit of decision tree size biases as a function of concept distribution. Technical Report 95-29, Department of Information and Computer Science, University of California, Irvine, 1995.

[Mur98] S.K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2:345–389, 1998.

[MW94] W. Muller and F. Wysotzki. Automatic construction of decision trees for classification. *Annals of Operations Research*, 52:231–247, 1994.

[NAC02] H. Nunez, C. Angulo, and A. Catala. Rule extraction from support vector machines. *European Symposium on Artificial Neural Networks*, pages 107–112, 2002.

[Nau91] G.E. Naumov. Np-completeness of problems of construction of optimal decision trees. *Soviet Physics, Doklady*, 36(4):270–271, 1991.

[Pas97] V.T. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys*, 29:171–209, 1997.

[Pas03] V.T. Paschos. Polynomial approximation and graph-coloring. *Computing*, 70(1):41–86, 2003.

[PBTL99] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. *ICDT*, 1999.

[PH90] G Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, 1990.

[PM77] H. Payne and W.S. Meisel. An algorithm for constructing optimal binary decision trees. *IEEE Transactions on Computers*, C-26:905–916, 1977.

[PM81] A. Paz and S. Moran. Non-deterministic polynomial optimization problems and their approximations. *Theoretical Computing Science*, pages 251–277, 1981.

[PM00] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. *ICML*, 2000.

[QJ93] J.R. Quinlan and C. Jones. FOIL: a midterm report. *European Conference on Machine Learning*, 1993.

[QR89] J.R. Quinlan and R. Rivest. Inferring decision trees using the minimum descripton length principle. *Information and Computation*, 80:227–248, 1989.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui87] J.R. Quinlan. Generating production rules from decision trees. *International Joint Conference on Artificial Intelligence*, pages 304–307, 1987.

[Qui90] J.R. Quinlan. Decision trees and decision-making. *IEEE Transactions on Systems, Man and Cybernetics*, 20:2:339–346, 1990.

[Qui93a] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[Qui93b] J.R. Quinlan. Combining instance-based and model-based learning. *International Conference on Machine Learning*, pages 236–243, 1993.

[RC87] R.A. Reckhow and J. Culberson. Covering simple orthogonal polygon with a minimum number of orthogonally convex polygons. *ACM Computational Geometry Conference*, pages 268–277, 1987.

[RDP+04] J.R. Rabunal, J. Dorado, A. Pazos, J. Pereira, and D. Rivero. A new approach to the extraction of ann rules and to their generalization capacity through gp. *Neural Computation*, 16(47):1483–1523, 2004.

[Ris78] J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[Ris89] J. Rissanen. Stochastic complexity in statistical inquiry. *World Scientific Series in Computer Science*, 15, 1989.

[Riv87] R.L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[Sal91] S. Salzberg. A nearest rectangle learning method. *Machine Learning*, 6:251–276, 1991.

[SB95] P. Schroeter and J. Bigun. Hierarchical image segmentation by multi-dimensional clusering and orientation-adaptive boundary refinement. *Pattern Recognition*, 25:5:695–709, 1995.

[Sch93] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10:153–178, 1993.

[Sch94] C. Schaffer. A conservation law for generalization performance. *International Conference on Machine Learning*, pages 259–265, 1994.

[Sch99] Oliver Schulte. Minimal belief change and pareto-optimality. pages 144–155, 1999.

[Seg04] M Segal. Planar maximum box problem. *Journal of mathematical modelling and algorithms*, pages 31–38, 2004.

[Ski90] S. Skiena. *"Maximum Independent Set" in Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica.* Addison-Wesley, 1990.

[SL91] S. R. Safavian and D. Langrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21:660–674, 1991.

[SL97] R. Setiono and H. Liu. Neurolinear: from neural networks to oblique decision rules. *Neural Computing*, 17(1):1–24, 1997.

[SLZ02] R. Setiono, W.K. Leow, and J.M. Zurada. Extraction of rules from artificial neural networks for nonlinear regression. *IEEE Transactions on Neural Networks*, 13(3):564–577, 2002.

[TC92] P. Tu and J. Chung. A new decision tree classification algorithm for machine learning. *IEEE International Conference on Tools with Artificial Intelligence*, pages 370–377, 1992.

[TC96] Hendrik Theron and Ian Cloete. BEXA: a covering algorithm for learning propositional concept descriptions. *Machine Learning*, 24(1):5–40, 1996.

[Thr95] S. Thrun. Extracting rules from artificial neural networks with distributed representatoins. *Advances in Neural Information Processing Systems*, 1995.

[Tre05] L. Trevisan. Inapproximability of combinatorial optimization problems. *Optimisation Combinatiore 2*, 2005.

[TS93] G. Towell and J. Shavlik. The extraction of refined rules from knowledge-based neural networks. *Machine Learning*, 13(1), 1993.

[TSRB71] C. Toregas, R. SWain, C. Revelle, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19:1363–1373, 1971.

[VH06] A. Silvescu D. Kang Vasile, F. and V. Honavar. TRIPPER: Rule learning using taxonomies. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2006.

[WD95] D. Wettschereck and T.G. Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19:5–27, 1995.

[Wet94] D. Wettschereck. A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. *European Conference on Machine Learning*, pages 323–335, 1994.

[Wha83] S. Wharton. A generalized histogram clustering for multi-dimensional image data. *Pattern Recognition*, 16:2:193–199, 1983.

[WI93] S. M. Weiss and N. Indurkhya. Optimized rule induction. *IEEE Expert: Intelligent Systems and Their Applications*, 8:6:61–69, 1993.

[WK91] S. Weiss and C. Kulikowski. *Computer systems that learn.* Morgan Kaufmann, 1991.

[WMKP06] J. Wojtusiak, R.S. Michalski, K. Kaufman, and J. Pietrzykowski. Multitype pattern discovery via AQ21: A brief description of the method and its novel features. *Reports of the Machine Learning and Inference Laboratory, MLI 06-2, George Mason University, Fairfax, VA*, 2006.

[Wol94] D.H. Wolpert. The relationship between PAC, the statistical physics framework, the bayesian framework, and the VC framework. *The Mathematics of Generalization Proceedings*, pages 117–214, 1994.

[Wol96] D.H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[YCHX05] X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: A profile-based approach. *SIGKDD*, 2005.

[ZB99] Y. Zhou and C. Brodley. A hybrid lazy-eager approach to reducing the computation and memory requirements of local parametric learning algorithms. *International Conference on Machine Learning*, 1999.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. *ACM SIGMOD International Conference on Management of Data*, 1996.