

# A MULTI-RELATIONAL APPROACH TO SPATIAL CLASSIFICATION

by

Richard Frank  
Bachelors of Math, University of Waterloo, 2001

A THESIS  
SUBMITTED IN PARTIAL FULFILMENT OF  
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in the School of  
Computing Science

© Richard Frank 2010

SIMON FRASER UNIVERSITY

Spring 2010

All rights reserved. However, in accordance with the *Copyright Act of Canada*, this work may be reproduced, without authorization, under the conditions of *Fair Dealing*. Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

## **APPROVAL**

**Name:** **Richard Frank**  
**Degree:** **Doctor of Philosophy**  
**Title of Thesis:** **A Multi-Relational Approach To Spatial Classification**

**Examining Committee:**

**Chair:** **Dr. Oliver Schulte**  
Associate Professor of Computing Science

---

**Dr. Martin Ester**  
**Senior Supervisor**  
Professor of Computing Science

---

**Dr. Ke Wang**  
**Supervisor**  
Professor of Computing Science

---

**Dr. Patricia Brantingham**  
**Internal Examiner**  
Professor of Computational Criminology  
Simon Fraser University

---

**Dr. Stefan Wrobel**  
**External Examiner**  
Professor of Computer Science  
University of Bonn

**Date Defended/Approved:** November 12, 2009



SIMON FRASER UNIVERSITY  
LIBRARY

## Declaration of Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website <[www.lib.sfu.ca](http://www.lib.sfu.ca)> at: <<http://ir.lib.sfu.ca/handle/1892/112>>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library  
Burnaby, BC, Canada

## **ABSTRACT**

Spatial classification is the task of learning models to predict class labels for spatial entities based on their features as well as the spatial relationships to other entities and their features. One way to perform classification on spatial data is to use a multi-relational database, by transforming the spatial data into multi-relational data, and then applying Inductive Logic Programming (ILP) onto it. However this presents novel challenges not present in multi-relational data mining problems.

One such problem is that spatial relationships are embedded in space. When applying a multi-relational data mining algorithm, the algorithm needs to determine which relationships are important and what spatial features of the entity to consider. In order to determine when two entities are spatially related in an adaptive and non-parametric way, a Voronoi-based neighbourhood definition is introduced in this thesis upon which spatial literals can be built.

Compounding the complexity is the need to use aggregation since the effect of a single spatial entity is negligible when in the neighbourhood of hundreds or thousands of other such entities. Properties of these neighbourhoods also need to be described and used for classification purposes. Non-spatial aggregation literals already exist within the multi-relational framework of ILP, but are not sufficient for comprehensive spatial classification. This thesis adds a formal set of additions to the ILP framework, to be able to represent the aggregation of multiple features, spatial aggregations as well as spatial features and literals. These additions allow for capturing more complex interactions and spatial occurrences such as spatial trends.

In order to more efficiently perform the rule learning and exploit powerful multi-processor machines, a scalable parallelized method capable of reducing the runtime by several factors is presented. The method is compared against existing methods by experimental evaluation on several real world crime datasets.

**Keywords:** spatial data mining; multi-relational; spatial classification; aggregation; parallelization; Voronoi Diagrams

**Subject Terms:** Multi-Relational Data-Mining, Algorithms, Experimentation, Theory

*To my parents,  
Who pushed me to start.*

*To my wife,  
Who supported me throughout.*

*To my son,  
Who helped me finish  
by giving me joy only like a son can.*

*To my baby,  
Who reminds me with each kick  
of the adventures yet to come.*

## **ACKNOWLEDGEMENTS**

I initially enrolled in the degree program at SFU expecting to leave after 2 years. That was 7.5 years ago. This is due to the great opportunities, collaborations and support during these years. Without them, none of this would have been possible.

First, I would like to thank my supervisory committee. I am especially thankful for the guidance over the years provided by Dr. Martin Ester who involved me in his collaborations and encouraged my involvement, which eventually led me to ICURS, where I continue my academic career. The entire practical section of this thesis would not have been possible without the support of Drs. Patricia and Paul Brantingham and their ICURS lab. They made their crime dataset available to me, and allowed me to explore and experiment with it unrestricted (sometimes even when I should have been working on ICURS material). I also appreciate the time I was able to spend in discussions with Dr. Arno Knobbe, Dr. Bryan Kinney and Dr. Oliver Schulte. I thank Dr. Ke Wang for his criticism, which was always welcome and invaluable. It was also a pleasure to have Dr. Stefan Wrobel as my external examiner.

These years also allowed me to travel to various conferences, meet people, exchange ideas and get feedback on my own research. I am very grateful for this opportunity and everyone who made this possible via various grants and funds.

I would also like to thank my fellow students for their discussions, support, thoughts and feedback during all phases of my studies. Sharing their own knowledge allowed me to avoid mistakes. I am especially happy to have been able to work with Flavia Moser (now Dr.) on numerous projects, papers and research.

Finally, I would like to thank my wife, Viktoria, and my parents who gave their love and extensive support which made this adventure possible and worthwhile.

# TABLE OF CONTENTS

<b>Approval .....</b>	<b>ii</b>
<b>Abstract.....</b>	<b>iii</b>
<b>Acknowledgements .....</b>	<b>v</b>
<b>Table of Contents .....</b>	<b>vi</b>
<b>List of Figures.....</b>	<b>ix</b>
<b>List of Terminology Used .....</b>	<b>xii</b>
<b>List of Notation Used .....</b>	<b>xiii</b>
<b>Chapter One: Introduction .....</b>	<b>1</b>
1.1 Data Mining.....	2
1.2 Propositional Data Mining .....	3
1.3 Multi-Relational Data Mining.....	3
1.4 Propositionalization of Multi-Relational Data .....	5
1.5 Spatial Data Mining.....	6
1.6 Spatial Data to Multi-Relational Data .....	8
1.7 Crime Data-Warehouse: A Criminology Application.....	10
1.8 Outline of Thesis .....	11
<b>Chapter Two: Spatial Data Mining.....</b>	<b>12</b>
2.1 Spatial Clustering .....	12
2.2 Spatial Association Rules.....	13
2.3 Spatial Co-Location.....	15
2.4 Spatial Statistics.....	17
2.5 Spatial Classification .....	18
2.6 Spatial Trend .....	20
2.7 Spatial Outliers .....	21
<b>Chapter Three: Relationships in Spatial Data .....</b>	<b>22</b>
3.1 Existing Neighbourhood Definitions.....	23
3.1.1 Topological Relationships .....	23
3.1.2 Buffer-Zones.....	24
3.2 Neighbourhood Definition Based on Voronoi Diagrams.....	26
3.2.1 Voronoi Diagram .....	26
3.2.2 Asymmetric Voronoi-Based Neighbourhood Definition.....	28
3.2.3 Symmetric Voronoi-Based Neighbourhood Definition.....	30
3.3 Comparison of the Efficiency of Different Neighbourhoods.....	31
3.3.1 Determining Neighbourhood using Buffer-Zones.....	32
3.3.2 Analyzing Neighbourhood using Buffer-Zones .....	32

3.3.3	Determining Neighbourhood using Voronoi Neighbourhoods .....	33
3.3.4	Analyzing Neighbourhoods using Voronoi Neighbourhoods .....	33
3.4	Discussion.....	34
<b>Chapter Four: Extending ILP for Aggregation and Spatial Data .....</b>		<b>35</b>
4.1	Preliminaries.....	36
4.1.1	Types of Aggregation .....	37
4.1.2	Aggregation of Different Data Types .....	38
4.1.3	Virtual Features .....	38
4.1.4	Selection Graphs.....	40
4.2	ILP Rule Language.....	40
4.2.1	Basic ILP Rule Formalism .....	41
4.2.2	ILP Rule Language Involving Aggregation .....	44
4.3	Extensions of ILP for Spatial Classification .....	45
4.3.1	Multi-Feature Aggregation .....	46
4.3.2	Spatial Features.....	48
4.3.3	Spatial Literals .....	49
4.3.4	Spatial Aggregation Literals.....	51
4.4	Relationship between ILP and DB .....	52
<b>Chapter Five: Spatial Rule Learning.....</b>		<b>55</b>
5.1	Rule Learning Strategies .....	56
5.1.1	Top-Down Rule Construction .....	57
5.1.2	Bottom-Up Rule Construction.....	58
5.1.3	Rule Quality Metrics .....	58
5.1.4	Dealing with Local Optima .....	60
5.2	Rule Learning .....	60
5.3	Candidate Literal Search in Parallel .....	62
5.4	Complexity Analysis .....	63
5.4.1	Features from Pre-Processing.....	63
5.4.2	Features from Run-Time .....	64
5.4.3	Single Literal Search Complexity.....	64
5.4.4	Rule-Learning Complexity .....	64
5.5	Pruning .....	65
5.5.1	Pruning Threshold Values .....	65
5.5.2	Pruning within the Existential Operator .....	66
5.5.3	Pruning Aggregation Functions.....	67
5.5.4	Pruning Spatial Features and Literals.....	68
5.5.5	Monotonicity Properties of Aggregation.....	69
5.6	Measuring the Quality of Rules.....	71
5.6.1	Precision .....	71
5.6.2	Accuracy.....	71
5.6.3	Recall .....	72
<b>Chapter Six: The Crime Data-Warehouse and the Implementation of UnMASC.....</b>		<b>73</b>
6.1	Creating the Crime Data-Warehouse.....	74
6.1.1	Input Requirements.....	74



6.1.2	Data Sources .....	75
6.1.3	Data Cleaning .....	76
6.1.4	Address Geo-coding and Validation.....	78
6.1.5	Data Transformation.....	79
6.1.6	Dataset Descriptions after Pre-processing.....	84
6.2	UnMASC Implementation Details .....	85
6.2.1	UnMASC Details.....	85
6.2.2	Method: RuleLearner.....	89
6.2.3	Method: LiteralEvaluator .....	90
6.2.4	Evaluation Optimization.....	91
6.3	Comparison to CrossMine.....	92
<b>Chapter Seven: UnMASC Experiments on a Crime Data-Warehouse .....</b>		<b>94</b>
7.1	Experiments.....	94
7.1.1	Datasets.....	94
7.1.2	Experimental Results.....	97
7.1.3	Evaluating the Effectiveness and Efficiency of the Voronoi Neighbourhood Definition.....	97
7.1.4	Evaluating Effectiveness of Aggregations.....	99
7.1.5	Evaluating the Effectiveness of Parallelization .....	102
7.2	Analysis of Classification Rules by Experts in Criminology.....	103
<b>Chapter Eight: Conclusions.....</b>		<b>106</b>
<b>Appendices.....</b>		<b>108</b>
Appendix A	: Details of Literals.....	108
A.1	Spatial Features.....	108
A.2	Single-Feature Aggregation Literals.....	110
A.3	Multi-Feature Aggregation Literals.....	111
A.4	Spatial Literals.....	112
A.5	Spatial Aggregation Literals.....	114
Appendix B	: Summary of Datasets in the Crime Data-Warehouse .....	115
Appendix C	: Pseudo-code.....	117
C.1	Road-Distance Function .....	117
C.2	Angle Function .....	117
<b>References.....</b>		<b>119</b>

## LIST OF FIGURES

Figure 1 – Sample spatial database schema .....	1
Figure 2 – A typical propositional database .....	3
Figure 3 – Sample multi-relational table .....	4
Figure 4 – Universal relation .....	4
Figure 5 – Aggregated table.....	4
Figure 6 – Recursive multi-relational database .....	5
Figure 7 – Result of self-joining a recursive multi-relational table .....	5
Figure 8 – Converting a spatial database to multi-relational database .....	9
Figure 9 – Density reachability.....	12
Figure 10 – Density connectedness.....	12
Figure 11 – A spatial trend radiating out from the central mall.....	20
Figure 12 – A spatial trend following the path of a river.....	20
Figure 13 – Topological relationships .....	23
Figure 14 – House 1 meets House 4 but not House 2. House 5 overlaps House 3.....	24
Figure 15 – Buffer-zone neighbourhoods (grey areas denotes the buffer-zone) .....	24
Figure 16 – 3km buffer-zone around different entity-types.....	25
Figure 17 – Buffer-zone of entity with varying density .....	25
Figure 18 – Sample Voronoi neighbourhood relationships for houses and malls .....	27
Figure 19 – Voronoi cells for varying distribution .....	27
Figure 20 – Establishing neighbourhood relationships using Definition 6.....	28
Figure 21 – Simple unsymmetrical neighbourhood definition (shadings represent roads).....	29
Figure 22 – Establishing neighbourhood relationships using Definition 7.....	30
Figure 23 – Runtime analysis .....	32
Figure 24 – Horn Clause format .....	43
Figure 25 – Single-feature aggregations and the feature-types they are applicable to. The <i>conditions</i> need to be applicable to the entities in <i>input</i> .....	45
Figure 26 – Proposed multi-feature literals and the <i>conditions</i> need to be applicable to the entities in <i>input</i> .....	46

Figure 27 – Sample relationship between two variables .....	47
Figure 28 – Existing spatial features expressing spatial properties of an entity .....	48
Figure 29 – Novel spatial features expressing spatial properties of an entity .....	48
Figure 30 – Existing spatial literals. $\theta$ denotes a comparison operator and $T$ a threshold. ....	50
Figure 31 – Novel spatial literals. $\theta$ denotes comparison operator, $D$ the output, and $T$ a threshold. ....	50
Figure 32 – Spatial Aggregation Literals, the <i>conditions</i> need to be applicable to the entities in <i>input</i> . ....	51
Figure 33 – Comparison of DB and ILP terminologies (underlined text is DB, non-underlined text is ILP terminology).....	54
Figure 34 – Lattice showing a small <i>rule-space</i> being explored <i>top-down</i> .....	56
Figure 35 – Lattice showing a small <i>rule-space</i> being explored <i>bottom-up</i> .....	56
Figure 36 – Aggregated table (Sorted by <i>Average Value</i> ).....	66
Figure 37 – Class label count.....	66
Figure 38 – Pruning of spatial literals.....	68
Figure 39 – Binary classification possible outcomes.....	71
Figure 40 – Original database setup.....	76
Figure 41 – Address cleaning process .....	77
Figure 42 – Data pre-processor program .....	78
Figure 43 – Stage 1) Creating class labels.....	79
Figure 44 – Stage 2) Separating BCAA entity-types.....	80
Figure 45 – Stage 3) Extracting spatial features during the preparation of the CDW .....	81
Figure 46 – Stage 4) Extracting Neighbourhood Relationships .....	82
Figure 47 – UnMASC Algorithm to calculate neighbourhood relationships in a database .....	83
Figure 48 – Partial schema of the crime data-warehouse. Entities are shown in red. The full schema contains 29 entity tables, and 435 relationship tables .....	84
Figure 49 – Steps in candidate spatial literal generation. Components specific to spatial classification are highlighted. ....	86
Figure 50 – UnMASC Client-Server Model.....	87
Figure 51 – Search process Example .....	88
Figure 52 – UnMASC RuleLearner ( $FG = FOILGain$ value) .....	89

Figure 53 – UnMASC LiteralEvaluator (to denote best solution, an ' is used, $FG =$ FOILGain value) .....	90
Figure 54 – <i>Malls</i> entities from the original spatial database .....	92
Figure 55 – <i>Malls</i> propagated to <i>Roads</i> entities.....	92
Figure 56 – <i>Malls</i> propagated to <i>Roads</i> , then to Houses .....	92
Figure 57 – UnMASC classifier and the Crime Data-Warehouse.....	94
Figure 58 – UnMASC program running with 6 worker threads, evaluating all possible literal types.....	95
Figure 59 – Statistics for database used for experiments.....	96
Figure 60 – Results for <i>Burglary</i> from Commercial properties in <i>Burnaby (BCB)</i> .....	98
Figure 61 – Results for <i>Burglary</i> from <i>High-density</i> properties in <i>Burnaby (BHB)</i> .....	98
Figure 62 – Results for <i>Auto-theft</i> from Commercial properties in <i>Surrey (ACS)</i> .....	98
Figure 63 – Comparison of rules found by different aggregation operators ( <i>Burglary</i> for Commercial properties in <i>Burnaby</i> - Rule #2).....	101
Figure 64 – Rules discovered by UnMASC .....	103
Figure 65 – Description of datasets used for experimentation. Dataset is restricted to <i>Burnaby</i> and <i>Surrey, BC</i> .....	116

## LIST OF TERMINOLOGY USED

**Entity:** A thing which is capable of an independent existence and can be uniquely identified.

**Entity-Type:** A set of entities of the same kind/nature. Sometimes written simply as *Type*.

**Feature:** A prominent aspect of an entity. All entities of the same type share the same features, but not necessarily the values of the features.

**ID:** A uniquely identifying value assigned to each entity. In the database, it is a (set of) column(s) containing the primary or composite key.

**Label:** An identifying marker that is attached to an entity. In this thesis each example is associated with a single label  $l$  from a set of disjoint labels  $L$ , where  $|L| > 1$ .

**Spatial Entity:** An entity which has attributed to it a spatial location and a point, line or polygonal representation.

**Spatial Feature:** A prominent aspect of an entity which is spatial in nature. These features can be derived from the entity's location and/or its point/line/polygonal representation.

**Vertex:** A point representing the intersection of the edges of a polygon.

## LIST OF NOTATION USED

$S$	Spatial database.
$T, t_i, t_\tau, /T/$	$T$ represents the set of entity-types, with $t_i$ representing a specific type, for example <i>house</i> , and $t_\tau$ represents the target-type used for classification. $T \in \{t_1, t_2, \dots, t_i, \dots, t_\tau, \dots, t_{ T }\}$ . With $/T/$ the count of entity-types.
$Q, Q_i, q$	$Q$ represents the (average) number of entities of each type, with $Q_i$ the count of entity-type $t_i$ . $q$ is an entity index, $1 \leq q \leq Q_i$ .
$e^i, e^{i,q}$	Set of entities $e^i \in \{e^{i,1}, e^{i,2}, \dots, e^{i,q}, \dots, e^{i,Q_i}\}$ of type $t_i$ , for example, $e^{house,2}$ represents <i>house 2</i> from the set of all houses. Thus $ e^i  = Q_i$ is the count of entities of type $i$ and $\sum_i  e^i  = Q$ is the total number of entity-types.
$P, p$	Number of vertices of an entity, $1 \leq p \leq P$ . If $P=1$ , then $e^{i,q}$ is a point. $p=1$ and $p=P$ are respectively defined to be the first and last vertex of the polygonal representation of $e^{i,q}$ .
$e_p^{i,q}$	Vertex $p$ of entity $e^{i,q}$ is a point in the Cartesian plane, i.e.: $e_p^{i,q} = (x, y)$ where $x, y \in \mathbb{R}$ .
$e_{p,x}^{i,q}, e_{p,y}^{i,q}$	The $x$ and $y$ coordinate of $e_p^{i,q}$ respectively
$f_d, f_n, f_c, f_s$	A feature for an entity is of one type from the set {date, numeric, categorical, spatial}. Let $ f $ represent the total number of features of all kinds.
$k$	Constant time required by the database to extract a spatial feature.
$g, h$	Features of an entity with both being exactly one type from $\{f_d, f_n, f_c, f_s\}$ , for example, <i>age</i> and <i>size</i> . $g \neq h$ .
$g(e^{i,q}, G)$	The value $G$ of feature $g$ for entity $e^{i,q}$ , for example $age(e^{i,q}, a)$ . There are two special features: <ul style="list-style-type: none"> <li>- <math>ID(e^{i,q}, I)</math> denotes the ID <math>I</math> of entity <math>e^{i,q}</math></li> <li>- <math>label(e^{i,q}, L)</math> denotes the (class) <i>label</i> <math>L</math> of entity <math>e^{i,q}</math></li> </ul>
$dist(e^{i,q}, e^{i,r}, D)$	Distance $D$ between two entities $e^{i,q}$ and $e^{i,r}$ .
$\beta(e^{i,q})$	Set of entities that are related (via some neighbourhood definition) to $e^{i,q}$ .
$\beta(e^{i,q}, e^{j,r})$	Denotes existence of a relationship between entities $e^{i,q}$ and $e^{j,r}$ ( $i$ could = $j$ ).
$\beta$	Average number of relationships for each entity.
$v$	Threshold value used for comparisons, for example $dist(e^{i,q}, e^{i,r}) < v$ denotes the set of entities that are within distance $v$ of $e^{i,q}$ .
$\Psi_{name}$	<i>Name</i> denotes the name of the parameter $\psi$ used for classification. For example, $\Psi_{CITY=Burnaby}$ denotes that the city used for classification is Burnaby.

# CHAPTER ONE: INTRODUCTION

Will a specific house be burgled, or a new mall yield profit? How do profitable malls differ from non-profitable malls? Are they located near businesses, or in neighbourhoods with high income levels? How can knowledge about existing profitable and non-profitable malls allow for the selection of locations which are likely to be profitable? These are inherently spatial problems which rely on the interaction of spatial events that must be found and accounted for. Non-spatial analysis is not capable of capturing these interactions, and hence non-spatial analysis methods must be adapted or new ones created in order to work in the spatial domain [67]. In non-spatial domains this type of analysis has been available, but with the large quantity of spatial data being captured, the need is there to discover these models in the spatial domain. These models would allow for describing the differences between instances of spatial data (ex: malls) belonging to different groups (ex: profitable/not-profitable) and then predict the group membership for those instances for which group membership is not known. In this thesis, a novel spatial classification method is introduced to answer such questions.

Spatial data mining is defined as the “nontrivial extraction of implicit, previously unknown, and potentially useful information from (spatial) data” [40]. Data mining is also viewed as a step in the process of Knowledge Discovery in Databases (KDD) of which the goal is to find *knowledge* in data [34]. In this process, once the data is prepared and pre-processed, an algorithm is applied to the data which analyzes it to discover interesting knowledge in the form of patterns. This algorithm is the data mining algorithm, an algorithm designed to find patterns in data.

In general, the database shown in Figure 1 is used as the running example in this thesis. The

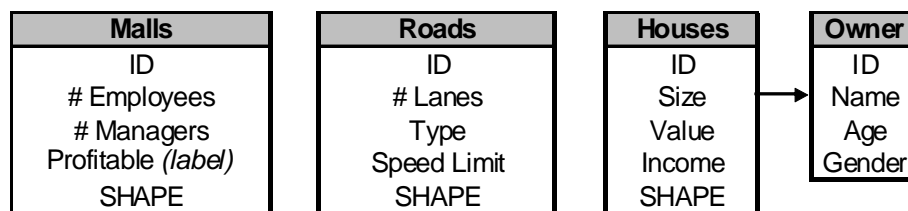


Figure 1 – Sample spatial database schema

database is spatial since it contains information about three spatial and one non-spatial entity-types. Note the lack of (non-spatial) relationships between the spatial entities. Each spatial entity has a shape feature that contains the polygonal representation of the entity, from which the area, perimeter, location and other spatial features can be derived.

## 1.1 Data Mining

The underlying foundation of many data mining problems is a database. A database is a collection of facts about some domain(s) in the real world structured according to a database schema. Its aim is to organize the data in a logical format that can be queried and searched in an efficient fashion. For example, it can contain information about customer purchases at a store, student grades or information about houses.

There are many interesting data mining tasks that can be performed on a database [5, 100, 30, 87, 107, 53, 55]. These tasks include the identification of groups that are similar to each other, called clusters, or identifying individual pieces of data, called outliers, which do not fit well with other pieces of data. This thesis is focused on classification, which identifies patterns within the data that may yield a better understanding of the domain(s) depicted by the database. The process involves the analysis of the datasets within the database, made of individuals called entities, by applying conditions on the data which match certain portions of the dataset, called a subgroup. For example, malls which *contain a bank* would be such a subgroup. Not all subgroups are interesting, but there are those that are different from the rest of the dataset in some aspect that will make that particular subgroup interesting. For example, if the malls subgroup that satisfy *contain a bank* also happen to have the property that they are *profitable*, it is now reasonable to conclude that there might be a dependency between the properties *contain a bank* and *profitable*. Due to the large amount of entities and features, it is impossible to identify such dependencies manually. The goal of a spatial data mining algorithm is to identify these dependencies automatically.

Individually such a dependency would not be so interesting, as it usually describes only a small portion of the entire dataset. Hence the goal of this task is not to discover individual distinct dependencies, but multiple subgroups each with multiple different dependencies. A collection of such subgroups can describe a significant portion of the dataset, and is known as a model. A model is useful because it describes the entire dataset, and not the individuals in it, and hence allows for reasoning about the entire dataset. Even better, if the dataset is large enough to represent a realistic sample of the real world, then the model can be used to predict additions to



the real world, such as a model describing the profitability of malls used to estimate the profitability of different locations for a potential new mall.

## 1.2 Propositional Data Mining

In propositional data mining [50, 89], a database  $S$  contains information about a single entity-type, represented as  $t_i$ . Each type  $t_i$  contains a set  $e^i \in \{e^{i,1}, e^{i,2}, \dots, e^{i,q}, \dots, e^{i,Q}\}$  of  $Q_i$  distinctly identifiable entities. For example, a database containing people would have a *Person* as a distinctly identifiable thing (Figure 2). An entity is uniquely identified by some feature(s) called the Primary Key, simply denoted using functional notation as  $ID(e^{i,q}, I)$  where  $e^{i,q}$  is the input entity, and  $I$  is the output variable representing the ID of  $e^{i,q}$ . All entities in this type of data are described by a fixed set of features, *age* for example, where each feature  $g$  is written in functional notation as  $g(e^{i,q}, G)$ , where the variable  $G$  has value corresponding to the result of function  $g$  applied onto  $e^{i,q}$ . For example, the age of person *P4* can be written as:

$$age(e^{People, P4}, A), A=17.$$

This type of database contains no relationships between any of the entities, or any other entity-types.

## 1.3 Multi-Relational Data Mining

Most existing data mining methods are propositional, and hence unfortunately do not work with many current databases, which contain multiple tables, such as those typically required for spatial data. Current databases need to store data that is simply too complex to be expressed as a single type of entity, or could contain relationships between entities. Propositional data mining in these cases is replaced by Multi-Relational Data Mining (MRDM) [5, 100]. MRDM works with data stored in a multi-relational database that contain  $|T|$  types of entities,  $T \in \{t_1, t_2, \dots, t_{|T|}\}$ , as well as

<i>(a) People</i>			
<b>P_ID</b>	<b>First Name</b>	<b>Last Name</b>	<b>Age</b>
P1	Mark	Doe	34
P2	John	Smith	45
P3	Betty	Smith	39
P4	Fred	Flint	54

**Figure 2 – A typical propositional database**

the relationships between these entities. In this type of database, all the relationships between the entities are explicitly given and are expressed through the use of Foreign Keys (FK) which reference the Primary Key (PK) of other entity-types. Figure 3(a) illustrates an entity-type in a multi-relational dataset which (b) references. In this example, the column ‘P\_ID’ of *House* is acting as the FK column since it references values from the PK feature of *People*.

It is not trivial to extend techniques that mine propositional data so that they work efficiently and accurately on multi-relational databases [26, 5, 100]. One alternative is to convert the multiple relationships and entity-types to a single relation, the so-called universal relation (Figure 4), that represents all of the data in the database. The result of this process can be huge, contain much duplicate information and still lose essential information [100, 101]. For example, if similar entities are grouped, a single entity might end up in multiple groups even though conceptually it is a single entity. Assume two groups are created:  $G_1 = \{H1, H3\}$  and  $G_2 = \{H2, H4\}$ , i.e.: people with a house worth  $< \$200,000$  and  $> \$200,000$  respectively (groupings shown in Figure 4). While this is a valuable grouping, Person *P1* would belong to multiple groups since he owns houses of both types. Aggregation would be able to resolve this predicament but if *House* was to be aggregated (resulting in Figure 5) then the fact that ‘John Smith’ owns multiple houses is lost.

<i>(a) People</i>			
P_ID	First Name	Last Name	Age
P1	Mark	Doe	34
P2	John	Smith	45
P3	Betty	Smith	39
P4	Fred	Flint	54

<i>(b) House</i>			
H_ID	P_ID	Value	Size
H1	P1	60,000	400
H2	P1	240,000	1500
H3	P2	120,000	500
H4	P3	232,000	800

Figure 3 – Sample multi-relational table

<i>People-House</i>					
First Name	Last Name	Age	Value	Size	
Mark	Doe	45	60,000	400	$G_2$
Mark	Doe	45	240,000	1500	$G_1$
John	Smith	34	70,000	500	$G_2$
Betty	Smith	39	232,000	800	$G_1$

Figure 4 – Universal relation

<i>People-House aggregated</i>					
First Name	Last Name	Age	Average Value	Total Value	Size
Mark	Doe	45	150,000	300,000	400
John	Smith	34	70,000	70,000	500
Betty	Smith	39	232,000	232,000	800

Figure 5 – Aggregated table

Even if the aggregation function *count of houses* is added, it would still not be possible to discover rules which involve the fact that ‘John Smith’ owns one expensive and one inexpensive house. In most cases, some form of information gets lost when aggregating.

As another example where propositional data mining methods fail on multi-relational data, consider the self-referencing recursive (but still multi-relational) database in Figure 6, denoting parent-child relationships between the different entities within the *People* entity-type. Creating a ‘universal entity’ for this database consists of doing a self-join, possibly multiple times, to create the hierarchy of parent-child relationships over multiple generations. Creating the self-join, Figure 7, introduces multiple features of the same name which would have to be kept track of during data mining. The number of self-joins could also get very large creating a ‘wide’ table with lots of features and duplicate information. In fact, all features outside of the first four features (of Figure 7) contain duplicate information since they exist in the original *People* (Figure 6). Aside from duplicating the information in ‘new’ features, a person with two parents would create almost-identical tuples in the database. In this case, the main problem would be how to modify the algorithm to handle the identification and duplication of information.

## 1.4 Propositionalization of Multi-Relational Data

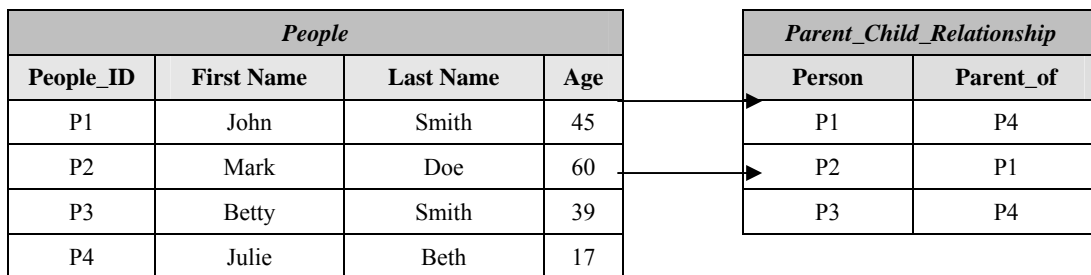


Figure 6 – Recursive multi-relational database

<i>People_Child_Joined</i>								
First Name	Last Name	Age	First Name	Last Name	Age	First Name	Last Name	Age
Mark	Doe	60	John	Smith	45	Julie	Beth	17
Betty	Smith	39	Julie	Beth	17			
...	...	...	...	...	...	...	...	...

Figure 7 – Result of self-joining a recursive multi-relational table

An approach to reduce the duplicate information contained in the *universal entity* is to transform the information from the multiple related entities into features of the entity used for classification. The transformation can make use of different aggregation functions, such as *sum* or *max*, for each feature. After this, conventional data mining techniques can be applied. This process is called *propositionalization* or *constructive induction*. Multiple algorithms, such as RSD (Relational Subgroup Discover) [62, 103], SINUS [60] and RELAGGS [58] take this approach of introducing aggregate features. According to the authors in [103], this approach is relatively successful in practice.

This technique however has a major weakness. Once the set of features are introduced, they are fixed. Data-mining algorithms might introduce conditions which invalidate the pre-calculated aggregate features. New techniques must be employed in order to effectively mine multi-relational (MR) databases by taking advantage of all the other entity-types represented in the database including the relationships between them. CrossMine [100] is one such MR data mining algorithm. The user selects the entity-type  $t_r$  that they are interested in, and in order to build subgroups, the algorithm attempts to add conditions onto this entity-type, and all other entity-types related to  $t_r$ . There is an important pre-requisite for these algorithms: the relationships between the entities must be known. This however does not hold for spatial data.

## 1.5 Spatial Data Mining

A spatial entity is an entity that also contains a spatial feature. A spatial feature describes the spatial properties of an entity such as area, location, perimeter, and contains coordinates (for example, longitude and latitude) that identify a specific location for the entity. A simple spatial entity could consist of a single coordinate defining a point in space, while a more complex spatial entity can contain multiple ordered coordinates that define the linear path that a river forms, or even the entire boundary of the river. Non-spatial features are explicitly stated within the database (as in MRDM), while spatial features are only implied by the spatial location of the entity or its embedded point/line/polygonal representation.

At the database level, in contrast to multi-relational databases, spatial data lacks the explicit relationships between the entities. All entities are embedded in a 2D or 3D space, a map for example, and even though the relationships are not specified in the database, a relationship could exist between entities  $e^{i,q}$  and  $e^{j,r}$ . This relationship, denoted as  $\beta(e^{i,q}, e^{j,r})$ , could take many forms, for example, *close*, *distance of 6km*, or *travel time of 5 minutes*. The relationship is implicit,

namely implied by the spatial location and possibly influenced by other entities near them (for example, a road influences travel time between two entities). In order to apply a MRDM algorithm, the relationships need to be made explicit. Due to the relational nature of spatial data, it is logical to approach the data mining tasks as if one were to perform data mining on a multi-relational database. Although a number of techniques employed in this thesis indeed stem from the multi-relational domain, it is going to be shown that the MR techniques are not expressive enough and novel methods are required to deal with the unique issues of the spatial domain.

One of these differences between MR and spatial data is in the way relationships between entities are defined. In an MR database, the relationships are explicitly given, but with spatial data, relationships are only implied through the spatial location of the entities themselves. If all possible spatial relationships are considered then the number of combinations between each entity becomes quadratic which is prohibitively expensive to compute for large datasets/maps [2]. The data mining algorithm must cut down this search-space and possibly only consider neighbourhood relationships, which presents the problem that generally it is not known a-priori what the suitable neighbourhood relationships are. Another factor to consider when trying to figure out which entity is a neighbour to which other entity: the influence between spatial entities differs based on how close they are. This is according to *Tobler's First Law Of Geography* which states that 'everything is related to everything else; but that near things are more related than those far apart' [95, 59, 32]. The question becomes "When do 'near things' become far enough apart that their influence over each other becomes negligible?". Several neighbourhood definitions exist in the literature, but these are mediocre: they require user input and do not take into account the number of entities or their distribution. A novel method is needed which overcomes the shortcomings of the current methods.

As an additional complication, relationships in spatial data are numerous: for a typical mall there are literally thousands of houses scattered nearby. Each individual relationship might be insignificant on its own, requiring the use of some form of (spatial) aggregation. At best, the current state-of-the-art classification methods use simple aggregation operators [52], those taking one parameter – *sum* or *count* for example. These operators however cannot describe the interaction between *distance from mall* and neighbouring *household income*, even though intuitively this might be a good way to distinguish profitable malls from failures. Aggregation operators capable of considering multiple features are required to incorporate this necessity into the data mining method. Spatial statistical independence cannot be assumed and hence many

statistical methods which are useful for single- or multi-relational data are not appropriate when analysing spatial data [54]. Spatial-specific aggregation operators are needed.

A spatial database also contains spatial entities, each with a spatial component, such as a geometric representation of the entity, which could be a point, 2D or 3D polygon, and could contain location and perimeter. This type of information cannot be used in a multi-relational database unless it is pre-materialized, but the information that is needed can vary depending on the dataset and application. It usually includes distance, direction, or amount of overlap. The authors of [87] suggested the following features to extract:

- *Location*, XY coordinates, longitude or latitude
- *Distance relationships*, such as Euclidian distance, Road-network distance, Manhattan distance. The measures could be discretized, such as: {close, far, very far}
- *Geometrical features*, such as perimeter or area
- *Directional features*, such as {North, East, South, West} or 35° East of North
- *Topological features*, such as intersect, coincide, overlap

This list is not exhaustive and contains redundancy. In order to not complicate or overwhelm the data mining task (and user!), the list of spatial features needs to be designed to be a concise selection of spatial features without redundancy.

Spatial data mining is also a computationally intensive process; there are many entities and possibly millions of neighbourhood relationships between them. For example, one of the real-world datasets used for this thesis contained over 60,000 entities, but the number of relationships was on the order of 3-4 million, depending on the neighbourhood definition used. In existing algorithms, the data mining is performed serially, with each neighbourhood relationship explored one after the other [100]. This type of algorithm is inefficient since it is unable to exploit the available processing power of the multi-processor machines available today.

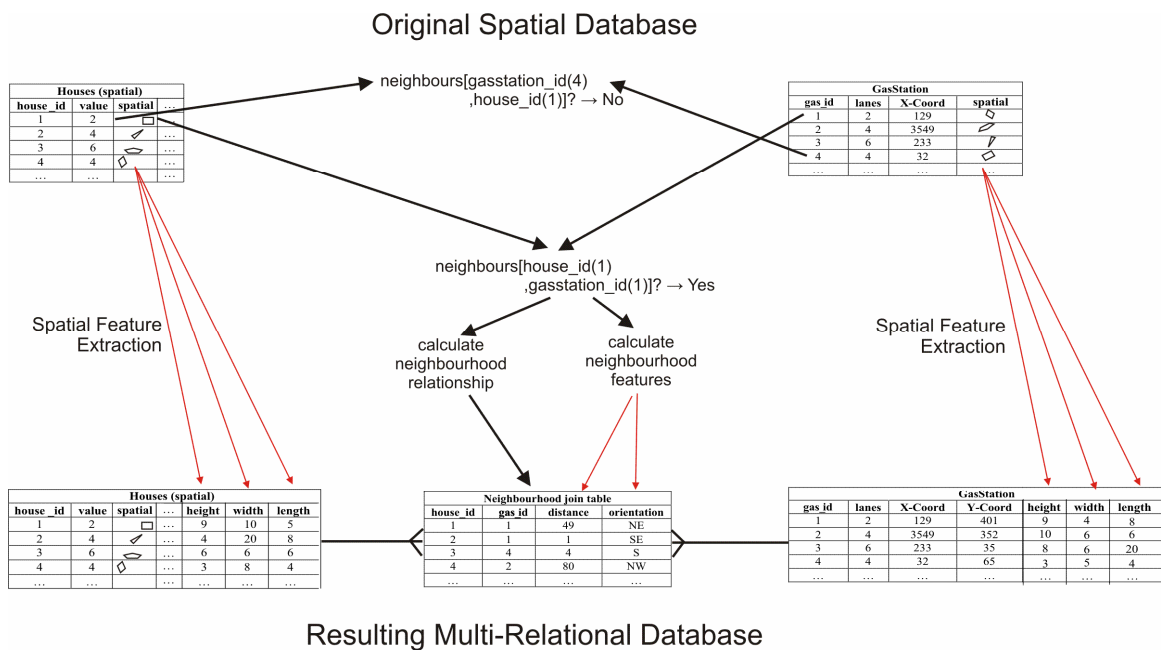
## **1.6 Spatial Data to Multi-Relational Data**

This thesis will present a way of formulating the spatial classification task as a multi-relational (MR) classification task. As a common way of representing multi-relational classification, the Inductive Logic Programming framework will be used to represent the results of the spatial data mining. Spatial data is different from MR data. In MR databases, the database contains all the information in explicit form, whereas with spatial databases, much of the information contained

in the database itself is implicit. This implicit information must be explicitly stated somewhere in the database in order for MR classifiers to be able to properly handle the data.

Probably the biggest reason MR classifiers cannot be applied to spatial data is because these algorithms were designed to take advantage of all explicitly stated features within the database. With spatial data, the relationships between the entities and features of both the relationships and the entities themselves are missing. A way must be found to extract the relationships and features, then store them in the database, and only then can a multi-relational classifier can be applied. The approach in this thesis extracts the relationships between the entities via Voronoi-based neighbourhood definitions, and then extracts a carefully selected set of features from the entities and from the relationship between them. This process (Figure 8) converts a spatial database into a multi-relational database, appropriate for a MR classifier to mine for classification rules.

Although the spatial database itself can be converted into a multi-relational database, the MR classifier also needs to be enhanced in order to deal with the peculiarities of spatial data. According to established laws (for example, Tobler's First Law of Geography [95]), and to statistical approaches (for example, those presented in Cressie [24]), entities in space influence each other to different degrees based on the distance between them. The classifier must be modified in order to handle these influences. Currently no data mining algorithm does this in an



**Figure 8 – Converting a spatial database to multi-relational database**

exhaustive, but generic, way. The approach in this thesis is capable of capturing these interactions either by performing multi-feature aggregations (for example, covariance between two features of a set of entities), or calculating spatial aggregations (such as spatial trends).

Thus, the spatial database is first transformed into a multi-relational database, and then a classifier, based on an established MR classification algorithm, is extended and applied. This allows the modified-MR classifier to take all the benefits and strengths of a multi-relational classifier and apply them to spatial data. In the end an aggregation-based multi-relational approach to spatial classification is created.

## **1.7 Crime Data-Warehouse: A Criminology Application**

In this thesis a spatial classification framework is described. The framework is called the *Unified Multi-relational Aggregation-based Spatial Classifier* (UnMASC), which contains two components. One part of it is the actual classification rule-learner, which is simply called UnMASC, and the other is the collection of databases that UnMASC learns the rules from. This collection of databases is called the Crime Data-Warehouse (CDW). The CDW was achieved in collaboration with the ICURS lab at the Criminology Department of SFU, where UnMASC was developed, implemented, and the experiments carried out.

As a result of an initiative between the ICURS lab and the Royal Canadian Mounted Police (RCMP), a secure computing facility was set up at ICURS to house, amongst other datasets, five years of real-world crime data for research purposes. This data was retrieved from the RCMP's Police Information Retrieval System (PIRS). PIRS, integrated with non-secure commercially available datasets such as the road-network, formed the core of the CDW. These source datasets first needed to be cleaned, the addresses contained in them verified before transforming into the final Crime Data-Warehouse. See Chapter 6 for full details.

Two cities were selected: Burnaby and Surrey. Datasets within the CDW were created about these two cities: Burnaby contained 66,260 entities with 2.9-4.1 million relationships between the entities, and Surrey contained 130,993 entities with 4.6-7.5 million relationships. Experiments for this thesis were done on these two cities to highlight both the power and benefit of the UnMASC framework and the Crime Data-Warehouse. See Chapter 7 for full details.



## 1.8 Outline of Thesis

In this thesis a novel comprehensive spatial classification rule-learner is proposed, called *Unified Multi-relational Aggregation-based Spatial Classifier* (UnMASC). It creates neighbourhood relationships via a novel Voronoi-based approach, analyzes multiple relationships simultaneously, and learns classification rules incorporating a broad range of spatial, single- and multi-feature, aggregation literals that are created on-the-fly.

The contributions of this thesis are as follows:

- Formulating the spatial classification problem as a multi-relational classification problem. This gives the rule-learner a solid theoretical foundation and allows for new insights into the spatial classification problem and solution.
- Establishing explicit neighbourhood relationships between entities via a novel non-parametric Voronoi-diagram based approach which yields a unique neighbourhood structure, materializing only relevant relationships and filtering out irrelevant ones.
- Introducing novel extensions into the MRDM framework, to allow for representation of multi-feature and spatial aggregation using MR terms and literals. This allows UnMASC to evaluate dependencies between multiple (spatial) features of an entity. A concise list of varying aggregation operators and spatial features without redundancies are used.
- Presenting a parallel, scalable spatial classifier which is able increase the efficiency of the data mining task to utilize all the computing power available while not sacrificing the quality of the data mining results.
- Experimentally evaluating UnMASC on a Crime Data Warehouse built on real-world data with real crime data from the federal police.

Chapter 2 focuses in on specific tasks commonly found in spatial data mining. Chapter 3 discusses alternative methods of defining the neighbourhood while proposing a superior method. Chapter 4 lays the language foundations and explores the additions required to this language. Chapter 5 covers spatial rule-learning on a theoretical level, while Chapter 6 presents the actual implementation details of the UnMASC algorithm. Chapter 7 presents the results of experimentation on the Crime Data-Warehouse, and Chapter 8 concludes the thesis.

## CHAPTER TWO: SPATIAL DATA MINING

There are many sub-areas of spatial data mining. Amongst these are spatial clustering, spatial outlier detection and spatial co-location. This Chapter briefly introduces a selection of sub-areas within spatial data mining in order to provide an overview of the domain. All of the tasks described in this Chapter assume that the neighbourhood relationships already exist or are calculated somehow on-the-fly.

### 2.1 Spatial Clustering

Clustering is the process of grouping entities in a database into meaningful subgroups. DBSCAN (Density Based Spatial Clustering of Applications with Noise) [29] is a propositional clustering algorithm which relies on density-based clustering to discover clusters of arbitrary shapes in a spatial database disturbed by noise. DBSCAN makes use of three concepts, direct density-reachability, density-reachability and density-connected. Two points  $p$  and  $q$  are **direct density-reachable** if  $q$  is within distance  $\epsilon$  of  $p$  and  $p$  has enough surrounding points that  $p$  and  $q$  are both part of a cluster.  $p$  and  $q$  are **density-reachable** if they are not directly density reachable themselves but are directly density-reachable via a sequence of intermediary nodes (Figure 9).  $p$  and  $q$  are **density-connected** if a point  $r$  exists such that both  $p$ -and- $r$ , and  $q$ -and- $r$  are density-reachable (Figure 10). Thus, each cluster must have all points within the cluster mutually density-

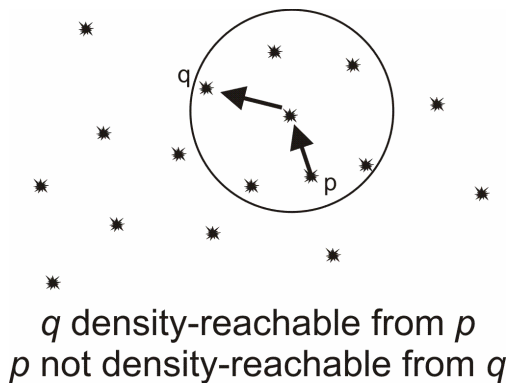


Figure 9 – Density reachability

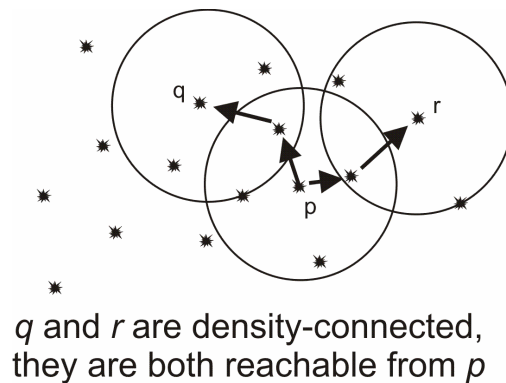


Figure 10 – Density connectedness

connected, and vice-versa, each point which is density-connected to any point in the cluster is part of the cluster [30].

CLARANS has also been extended to handle clustering of spatial polygons. Given a set of polygons  $P$ , where each polygon of the set has a boundary, a set of features, and a set of spatial events, the method presented in [70, 106] partitions  $P$  into clusters with respect to some similarity measure. This similarity measure can be a generalized distance, which combines three independent similarity measures calculated from the different features and spatial locations of each polygon. By taking into account the polygon associated to each entity, more accurate classification was achieved.

Another type of clustering is clustering of spatial networks. ELink, a propositional clustering approach, [66] performs in-network clustering of sensor nodes by building within each node an auto-regression model of readings from all neighbouring nodes. The distance measure between two nodes depends on the parameters of the auto-regression model of the nodes. ELink grows clusters as follows. The set of nodes to be clustered are recursively broken down into cells at different levels (as in a quad-tree), with level  $i$  having 4 times the number of nodes as level  $i-1$ , and the root-cell being defined as level 0. Each cell elects a leader. Since the cells are structured as a tree, each cell has a parent-cell. Starting from the leader of level 0, the cluster is grown until the cluster is  $\delta$ -compact. That is, all pairs of nodes within the cluster are within distance  $\delta$  of each other, where the distance is based on the auto-regression model, as discussed above. Once the clustering is complete at a specific level, the clustering is repeated from the leader nodes of the next level, until all nodes belong to a cluster.

## 2.2 Spatial Association Rules

Spatial association rules are a descriptive task intended on identifying frequent related entities in spatial data [87]. Unlike Spatial Co-Location (Chapter 2.3), the items in the association rule do not need to be located in the same neighbourhood and are expressed as explicitly stated transactions. Conceptually, spatial association rules are similar to non-spatial association rules. Non-spatial association rules show the dependency relationships between different features for a set of entities (items in a transaction, for example) [44] and are of the form  $Y \leftarrow X$ , i.e.: *antecedent*  $\leftarrow$  *precedent*, with some confidence and support measures assigned. Spatial association rules however reference at least one spatial entity in their precedent or antecedent. For example, the following spatial association rule states that if a house is close to a beach, it is expensive:

R1: expensive(H)  $\leftarrow$  house(H), beach(B), close\_to(H, B).

The support and confidence measures are calculated identically to their non-spatial counterparts [91]. A spatial association rule

$$P_j \leftarrow P_1, P_2, \dots, P_i \text{ (support: } S\%, \text{ confidence: } C\%)$$

states that if a  $P_1, P_2, \dots, P_i$  are true then so is  $P_j$  with probability  $C\%$ , and, of the entire dataset,  $S\%$  of the entries in the database contain all the components of the rule. The confidence value is given by

$$\text{Confidence} = \frac{\text{Support}(P_1, \dots, P_i, P_j)}{\text{Support}(P_1, \dots, P_i)} \quad (2.1)$$

This confidence value measures the correlation between the body and head of the rule, signifying the rule's strength. The support value is given by:

$$\text{Support} = \frac{|\text{tuples with items } \{P_1, \dots, P_i, P_j\}|}{|\text{all tuples}|} \quad (2.2)$$

This can be thought of as the percent of the data that contains all the items in the rule. The minimum support value desired by the end-user is controlled by a value they specify a-priori, called the *minimum threshold*.

Complex rules are discovered in a similar fashion to non-spatial association rules. Initially, the occurrences of  $P_1$  and  $P_j$  are counted, i.e.: the support of  $P_j \leftarrow P_1$  is tested, and, assuming that it is found to be infrequent (unsupported), then no specialization of  $P_j \leftarrow P_1$  (for example:  $P_j \leftarrow P_1, P_2$ ) is going to be tested. If, however, the number of occurrences is above the minimum threshold then more features will be added and these specializations will be recursively tested for support and confidence. This process of adding more terms is referred to as *refining the rule*, and is continued until the support of the rule falls below the threshold. At the end, for all frequent rules the confidence is calculated, and assuming it is greater than a given minimum confidence threshold, the rule is added to the collection that is returned to the user (or further analyzed for 'interestingness measures' [41]).

While the above is a propositional association rule, multi-relational ILP based spatial association rules [5] have also been used to create a set of association rules from spatial data. The support and confidence levels of a specific rule are defined as in non-spatial association rules. As opposed to propositional data, the entities in multi-relational data can form a hierarchy relationship with each other. Due to this, there is another approach to refining association rules that cannot be done with

propositional rules: entities in rules can be substituted for other entities at different granularity levels. For example, if a rule contains

$$house(H), neighbour(H,S), transportation\_stop(S)$$

it can either be extended by adding a restriction to the rule

$$house(H), neighbour(H,S), transportation\_stop(S), size(H, S), S > 100m^2$$

or an entity can be exchanged for something at a finer granularity level, such as

$$house(H), neighbour(H,S), metro(S).$$

The drawback of this type of restriction is that a different set of minimum support and confidence thresholds might need to be set for each granularity level. The approach presented in [5] pre-processes the relationships between the entities (as was done for the experiments presented in this thesis); however, no form of aggregation is performed when the association rules are calculated. Further, numerical data is discretized into ordinal data, which is not a limitation with the approach presented in this thesis.

## 2.3 Spatial Co-Location

As opposed to spatial association rules, where the entities do not need to be located in the same neighbourhood, the goal of spatial co-location is to detect entities that frequently share the same neighbourhood. In addition, spatial association rules have explicitly expressed transactions which involve the entities. This does not hold for spatial co-location, where the entities are embedded in space and contain a variety of spatial relationships [91]. More formally, given spatial entity-types  $J, K, \dots, L$  with each type containing multiple entities, a spatial co-location pattern  $P$  is the subset of types which frequently (more than some threshold value) co-exist in the same neighbourhood [107]. The spatial entities are embedded in space and hence only implicitly form spatial neighbourhood relationships. Spatial co-location however depends on an established neighbourhood graph which must be given as input: all neighbourhood relationships must be established a-priori somehow. The neighbourhood relationships must be pre-processed and provided as input to the algorithm.

Co-location rules are of the form “ $P: K \leftarrow J(S_{J,K}, CP_{J,K})$ ” [48], where:

- $P$  denotes the pattern
- $J, K$  are co-located entity-types

- $S_{J,K}$  is the support of the rule
- $CP_{J,K}$  is the confidence of the rule, i.e.: the conditional probability of an instance of  $K$  occurring given  $J$  occurs

Multiple such techniques for discovering such co-location patterns exist [99, 32, 20, 48, 91, 107]. Two of the common approaches to finding spatial co-locations include using Extended Spatial Entities or using Extended Event-Centric Models.

The Extended Spatial Entities approach can find co-location patterns with 2D (polygonal) spatial entities [99]. First, all entities are labelled (for example, entity 2 of type  $J$  would be labelled  $J_2$ ). Then an area of a user-specified diameter, called a *buffer-zone*, is created around each entity with intersecting buffers establishing neighbourhood relationships. The neighbouring entities are then listed and frequent neighbouring entities are recorded. Supersets of those frequent types are tested for frequency and non-frequent types are pruned from the search-space. The support is calculated by using the *coverage ratio* (the fraction of the total area covered by the set), given by

$$S_{P,K} = \frac{N(f_1, f_2, f_3, \dots, f_k)}{\text{Total\_Area\_of\_Plane}} \quad (2.3)$$

where  $N(f_1, f_2, f_3, \dots, f_k)$  represents the area covered by the buffer-zones of the set. The algorithm can be modified to be more efficient (but less accurate) by using the bounding-box of the buffer-zone as a coarse-level estimator of the neighbourhood relationship (since testing the bounding-box for intersections is less computationally expensive). After all the coarse-level estimators have yielded neighbourhood relationships, they are then tested again using the original buffer-zone.

An alternative approach to co-location detection uses an Extended Event-Centric Model [48]. The algorithm looks for instances of different entity-types that form a clique (fully connected sub-graph). Each entity-type  $K$  that is involved in a pattern  $P$  is assigned a participation ratio

$$PR_{P,K} = \frac{\# \text{ of instances of } K \text{ that participate in pattern } P}{\# \text{ of instances of feature } K} \quad (2.4)$$

The support of the entire pattern is calculated by the use of a measure called the *Participation Index (PI)*. The *PI* for a pattern  $P$ , denoted  $PI_P$ , is defined as the minimum of all the participation ratios across all entity-types involved in pattern  $P$ , i.e.  $PI_P = \min_{i=1}^{\text{all entity-types}} \{PR_{P,i}\}$ . All patterns with a high *PI* value are then evaluated for supersets with a high *PI* value. As the set grows, the value of the index cannot increase (since  $\min(PR_{P,K})$  cannot increase), and hence the *PI* value can be used to prune supersets of entity-types from further consideration.

The Extended Event-Centric Model finds cliques, but if the user wants to look for relationships that are not cliques then this is not an optimal approach, in which case the Extended Spatial Entities could find appropriate co-location patterns. Neither approach is able to find patterns that span to further entities, i.e.: where all entities in the rule are not neighbours of each other but could be neighbours-of-a-neighbour. Finding neighbourhood relationships with neighbours other than direct neighbours of the target entity seems to be a drastic shortcoming. The method could probably be modified to involve further neighbours by scanning the database to find an interesting pattern  $P_1$  involving entity  $K$  and then performing another scan to find an interesting pattern  $P_2$  that contains entities co-located with some entities from  $P_1$  (as opposed to **all** entities from  $P_1$ ).

## 2.4 Spatial Statistics

The study of spatial statistics contains a set of techniques for the purpose of studying spatial entities using their spatial representation, such as topological properties or relationships between the entities [24]. One example of this is the measure of spatial correlation between multiple spatial features in order to describe the relationships between them. Usually data is analyzed under the assumption that processes or relationships apply equally across the study area. The analysis can include location as a variable, but this does not necessarily improve the accuracy of the model [59]. More specifically, considering location could be irrelevant if the density is too low to sufficiently represent the actual spatial variation. Stationarity, as stated in [19], “is defined as a quality of a process in which the statistical parameters (mean and standard deviation) of the process do not change with time”. Spatial stationarity hence can be described as a ‘process not changing over space’, that is, the distribution of the values of the incidences do not change with location or distance from an entity. The incidences of sickness is an example in which the stationarity assumption would imply that the incidences are uniformly distributed across a spatial space, but in real life a single heavily-polluting factory would most likely cause more sickness to occur near the factory<sup>1</sup>, invalidating the stationarity assumption.

Assuming enough sample points are available, there are two basic approaches to measuring spatial statistics geographically: global and local. Geographically global approaches assume that

---

<sup>1</sup> For an example, see [http://www.ctv.ca/servlet/ArticleNews/story/CTVNews/1024895514366\\_20304714](http://www.ctv.ca/servlet/ArticleNews/story/CTVNews/1024895514366_20304714)

spatial non-stationarity does not have an effect (i.e.: the data is stationary spatially), which can cause decreased model accuracy since this assumption might not hold in real datasets. With this method, the spatial statistics are based on the entire dataset. The geographically local methods include in the analysis the contribution of each entity along with their location and are done using a moving window approach. The method only calculates the spatial correlation for the data points in the window and hence would not be able to capture spatial trends that do not ‘fit’ into the window [59]. Although the data (including spatial stationarity) is the same with both approaches, the global and local approaches measure the statistics differently with different results.

Although the majority of the spatial statistics apply only to propositional data, there are spatial statistics that take as input the spatial location of the entities and hence can be applied to a set of entity-types (represented as multi-relational data). The spatial statistics, such as spatial trend, that can be applied onto a set of entity-types are used in this thesis and discussed in Chapter 4.3.4.

## 2.5 Spatial Classification

Classification, covered in detail in the rest of this thesis, in general, involves the mapping of data into classes or categories where similar entities are mapped into the same class and entities significantly different are mapped into different classes. The number of classes is much smaller than the number of original entities. If the data mining task is to find classification rules, then the rules ideally should describe a single class of entities fully without describing entities of another class. There are two main approaches, data characterization and data discrimination [44]. Data characterization, or classification based on shared/similar characteristics, compresses the data into increasingly general relations while data discrimination generates statistical summaries in order to show the differences between the classes.

In spatial classification, a set of spatial entities (such as a set of malls) are given, each with a specific class label (such as *profitable*) on which rules are built. This can be done either by finding rules that partition the entities in the database into a set of classes [5] or through Inductive Logic Programming (Chapter 4) where a hypothesis is found that can predict class labels of the examples [101]. Ideally, the literals should take into account the distance, direction or connectivity relationships of the entities and learn the rules based on all spatial and non-spatial features. For example, the sample rule

$$\text{profit}(M, \text{'Yes'}) \leftarrow \text{mall}(M), \text{neighbour}(M, H), \text{house}(H)$$



expresses that a house is profitable if there are houses neighbouring the mall. A set of such rules are built in order to create the classifier.

Spatial association classification rules are built as described in Chapter 2.2, using spatial literals and relationships. The advantage is that both association rule mining and classification are done simultaneously. However, the disadvantage is that small changes to features could cause sudden classification changes because the rules are relatively independent. Association rules have been previously used for classification purposes but most of the existing methods only work for a single table and not multi-relational data [17].

SPADA [17], a spatial associative classifier, builds classification rules based on multiple entity-types taking into account the hierarchy information between the entities (for example, that a city is within a county). The approach presented in [17] is able to build these rules from a spatial database which has been pre-processed into a multi-relational database. Although no types of aggregations are performed, they do use Naïve Bayes Classifiers to fill in missing values.

As another approach, a few methods convert the spatial database into a multi-relational database by explicitly calculating the spatial relationships between the entities and storing them in a spatial join index [21]. The spatial join index could identify just the pairs of entities in a relationship [96], or alternatively it can contain an extra piece of information which describes the property of the spatial join [102]. While this thesis also uses a spatial join in order to be able to transform the spatial database into a multi-relational database, the current approaches do not materialize any significant features about the neighbourhood relationship, which the approach in this thesis does. This allows the spatial classifier to make much more informed decisions while building the classifier.

Another method, RELIEF [53], builds classifiers on a spatial database using the Inductive Logic Programming approach commonly used on multi-relational data. As opposed to pre-processing some or all of the literals, 'rough' literals are calculated and are refined if they turn out to be promising. Aggregate information can also be constructed based on the non-spatial information of other entities in the neighbourhood. Unlike the approach presented in this thesis, the aggregates are based on the features of all the neighbouring entities, not only those that match the conditions already in place in the rule.

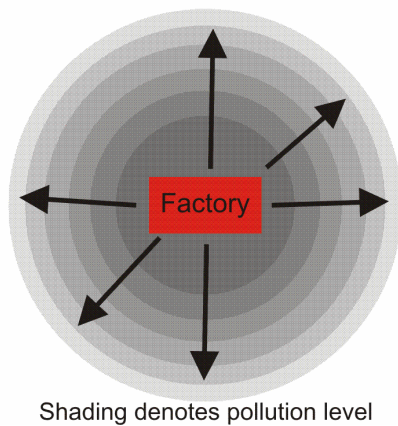
## 2.6 Spatial Trend

Spatial trends illustrate the correlation of one or more non-spatial features and the distance away from a central entity [87, 5]. This concept works equally well on propositional data, where the trends are calculated between all entities, or on multi-relational data, where the trend is calculated between a specific entity (factory, for example) and entities of another type. It allows for the detection of changes along the spatial dimension, such as *follows*, *coincides* or *parallels*. It is very similar to pattern-mining [77] since if the spatial relationship between a central entity and other entities is calculated and stored as a feature of the entity, then subsets of features should form patterns. This pattern, since it is spatial in nature, is the equivalent to a spatial trend.

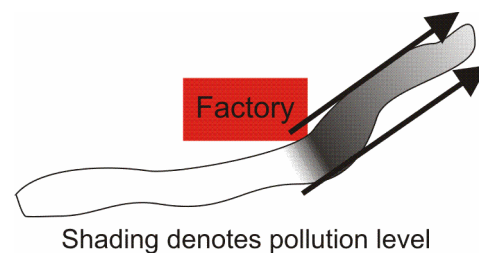
Spatial trends have been defined as a pattern of change of some non-spatial feature within the neighbourhood of some entity [31, 33]. For example, “*when moving away from an industrial area, the amount of pollutants in the air decreases*” is a spatial trend. There are two types of trends: global trends and local trends:

- Global trends show a pattern in all neighbourhood directions of the spatial entity (Figure 11)
- Local trends only show the pattern in a specific direction, along a river, for example (Figure 12)

Trends are detected through the use of regression analysis using feature values vs. distances of entities from an established center. The analysis starts with direct neighbours and is extended to neighbours-of-neighbours if a possible correlation is found.



**Figure 11 – A spatial trend radiating out from the central mall.**



**Figure 12 – A spatial trend following the path of a river.**

## 2.7 Spatial Outliers

Spatial outliers are entities which are inconsistent with their spatial neighbours on their non-spatial attributes even though the non-spatial values could be normal for the rest of the entities of the same type [92]. Outlier detection methods include *distribution-based* using standard statistical distributions, *depth-based* methods which map entities into an  $m$ -dimensional information space and *distance-based approaches* which calculate the proportion of entities that are a specified distance from a target entity [44]. For each entity, based on the feature values of neighbouring entities, the expected value of some feature of the entity is calculated then compared to the actual value, if the difference is above a threshold, then the entity is deemed an outlier. For example, a bank could be deemed an outlier in the middle of a residential area since it has very different feature values that are normal in its neighbourhood, even if the bank has typical feature values when compared to all banks.

The above-mentioned spatial outliers are applicable mainly to global outliers, where the outlier is determined with respect to the entire space of entities. Outliers can also be detected in local neighbourhoods where the outliers are different from the neighbourhood they are in [39]. This type of detection is different from detecting outliers in the entire space of entities since it also involves detecting the optimal neighbourhoods which produce the biggest outliers. The approach presented in [39] "grew" the neighbourhood around each entity, and at each growth cycle determined how much of an outlier the entity is with respect to the local neighbourhood. Although the neighbourhood was restricted to city-block size rectangles, the approach can be generalized to detect outliers in neighbourhoods of any shape or size.

## CHAPTER THREE: RELATIONSHIPS IN SPATIAL DATA

Spatial data contains multiple types of entities arranged in space, such as a map representing a city. When looking at a map, the reader cannot see explicit relationships between the houses, malls or other types of entities on the map. Similarly, spatial data also does not contain relationships between the entities. Simply because there are no relationships visible on a map does not mean that the entities live in isolation. They do in fact influence each other through different means, noise and pollution of a highway influences nearby property-values for example. To make an informed decision about a neighbourhood these influences or interactions must be captured and used.

The fundamental difference between spatial and multi-relational data is the implicit nature of the relationship between the two entities, instead of the links that exist as in multi-relational data. Spatial data only indirectly implies relationships via the spatial location of the entities, thus some work has to be done to explicitly materialize relevant relationships between the entities. This needs to take into account Tobler's First Law of Geography [95, 59, 32]:

**Definition 1 – Tobler's First Law of Geography** – “Everything is related to everything else; but that near things are more related than those far apart.”

This law implies that relationships are important to different degrees, and the importance is influenced by distance. Thus there is a need to prune the huge space of pair-wise relationships for two reasons. The first is to save analyzing unlikely dependencies between distant entities. The second is to avoid constructing misleading rules.

It is argued in this Chapter that current state-of-the-art methods, *topological relationships* [17, 75, 87] (Chapter 3.1.1) and *buffer-zones* [21, 5] (Chapter 3.1.2), produce relationships that are somewhat arbitrary. Topological relationships have been used in the literature but are very restrictive due to the fact that they cannot establish or remove neighbourhood relationships based on any form of distance criteria. The majority of approaches establish around each entity a buffer-zone using a fixed distance threshold, which is provided by the user, to find neighbourhoods. The disadvantage of this approach is that theoretically the distance threshold can take on a large number of possible values or might be inappropriate for different entities of the same type. Selecting the right distance threshold is crucial for identifying meaningful classification results,

as different thresholds yield different results [90]. Currently no work exists that provides a unique neighbourhood relationship with no user-definable parameters that also takes into account the number and distribution of entities. The notion that is used in this thesis is that of the *Voronoi diagram* (Chapter 3.2). Voronoi diagrams partition a plane into regions, called Voronoi cells, which contain all the points that are closest to the entity contained in the Voronoi cell, naturally representing relationships between entities [10]. The benefits of spatial classification using Voronoi diagrams are analyzed in Chapter 3.3.

### 3.1 Existing Neighbourhood Definitions

Two main approaches exist for defining neighbourhoods in the literature: *topological relationships* and *buffer-zones*.

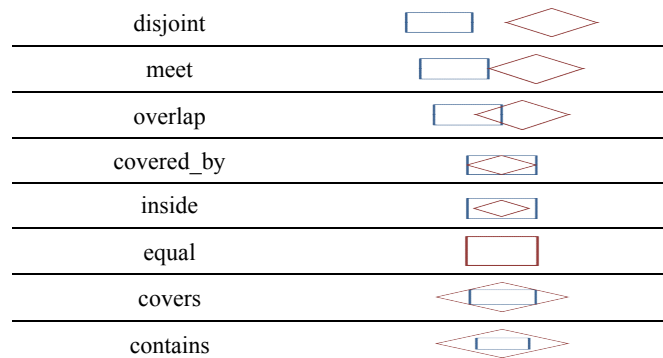
#### 3.1.1 Topological Relationships

Topological relationships, listed in Figure 13, can describe the relationship between two entities, and are defined as [28, 75]:

**Definition 2 – Topological Relationship** – The set of *topological relationships* is the set of eight types of interactions which are based on the intersections of the entity’s boundaries, interior and exterior. These relationships are preserved under translation, rotation or scaling.

These relationships are very common and easy to understand, but do not always capture the intuitive notion of a neighbourhood.

The neighbourhood definition can be defined two different ways: with or without the topological relationship *disjoint*. The *disjoint* relationship is based on a Boolean operator which returns true



**Figure 13 – Topological relationships**

in case the entities have no spatial area in common and false if they do (partially) share the same spatial area. For example, in Figure 14, House 1 *meets* House 4 but is *disjoint* from House 2. If the neighbourhood definition does include the *disjoint* relationship then this definition is going to explicitly state a relationship between *all* pairs of entities, with the vast majority of entities being simply *disjoint* from each other. This is because there are no limits to the pairs of entities onto which *disjoint* is applied, just the fact that two entities are disjoint is enough for it to explicitly establish the relationship. Thus one of the topological relationships always applies to any two entities, meaning no relationships can be pruned and the search space remains huge.

If the definition does not use the *disjoint* relationship, then the relationship between two entities can only be measured by the degree of the overlap of their boundaries. This however restricts the relationship to be between entities that do have some overlap of their boundaries. For entity pairs which do not have any overlap, no relationship could be defined. Many people would define themselves as living in the neighbourhood of a mall even if they lived in the close vicinity and not directly adjacent to it. Saying that ‘a mall is profitable if it *meets* 5 houses’, although possibly true, is not nearly as strong as stating that ‘a mall is profitable if more than 20,000 people live in its close vicinity’. In order to capture this information, *disjoint* is needed, but such a concept is missing from the definition and cannot be expressed otherwise.

Either way, basing a neighbourhood definition on topological relationships does not yield a viable set of relationships for classification purposes.

### 3.1.2 Buffer-Zones

Another common approach to defining neighbourhood relationships is through the use of *buffer-zones* [53] where a region of a user-specified size is constructed around the entity and any other



Figure 14 – House 1 meets House 4 but not House 2. House 5 overlaps House 3.

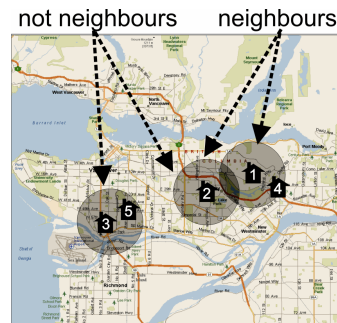


Figure 15 – Buffer-zone neighbourhoods (grey areas denotes the buffer-zone)

entity inside is considered to be a neighbour, as in Figure 15. More formally:

**Definition 3 – Buffer-Zone** – A buffer-zone with distance threshold  $d$  around an entity  $i$  of type  $t_j$ , denoted  $e^{i,j}$ , is the area that is within distance  $d$  to  $e^{i,j}$ . An entity  $e^{i,r}$ , if it intersects the buffer-zone of  $e^{i,j}$ , is defined to be a neighbour of  $e^{i,j}$ , that is two entities are neighbours if  $\text{dist}(e^{i,j}, e^{i,r}, D), D < d$ .

For example, using Figure 15 (the houses are numbered) the buffer-zone of  $e^{\text{house},1}$  encompasses  $e^{\text{house},2}$  but not  $e^{\text{house},3}$ , hence  $e^{\text{house},1}$  and  $e^{\text{house},2}$  are neighbours, but  $e^{\text{house},1}$  is not a neighbour of  $e^{\text{house},3}$ .

Although distance-based approaches, such as buffer-zones, are very common ways of determining the neighbourhood of an entity [21, 5, 53], they do have major drawbacks. First, they assume that the same threshold applies for every entity-type. For entity-types of greatly varying size and range of influence, a constant sized buffer-zone either becomes too large or becomes irrelevant. For example, using Figure 16, while a buffer-zone of 3km is realistic for cities (a), it is too great for houses (b) where the neighbourhoods are measured in 10s of meters, and too small for provinces (c) where ‘neighbourhoods’ might be measured in 100s of kilometres. Second, in principle, an infinite number of buffer-zone thresholds could be selected, but each can possibly change the result of the analysis [90], so how can one result be trusted over another? Third, for entities of the same type, the distribution of the entities across space can change significantly across the entire dataset (Figure 17), so while the size of a buffer-zone might be correct in one region, it could be inappropriate in another.

A solution to the above problems is to have multiple buffer-zone thresholds, perhaps one per entity-type, or even multiple thresholds for entities of the same type. If the result of classification can vary so greatly based on the choice of the buffer-zone threshold [90], then how can the user be trusted to select multiple such sizes? Therefore, it is desirable to have an automatic way to select the thresholds, both for entities of different types and for entities within the same type.

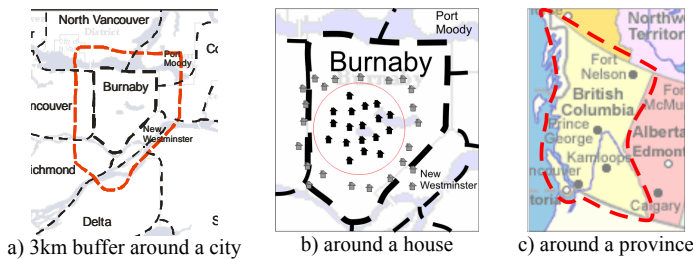


Figure 16 – 3km buffer-zone around different entity-types



Figure 17 – Buffer-zone of entity with varying density

## 3.2 Neighbourhood Definition Based on Voronoi Diagrams

Without a neighbourhood definition all spatial relationships need to be evaluated for all pairs of entities as they can potentially be constructed. In this thesis, the **Voronoi Neighbourhood** is introduced as a way of explicitly creating only those relationships which involve entities that have an influence on each other. The classification algorithm now cannot learn a classification model using any arbitrary pair of entities. This could be thought of as undesirable, but following Tobler's First Law of Geography [95, 59, 32] it is actually a necessity in order to avoid constructing a potentially misleading classification model.

When working with maps and spatial information, Voronoi diagrams, named after their creator Georgy Feodosevich Voronoy, are prevalent as they naturally represent relationships between entities [8, 47, 65]. Voronoi diagrams have been used in the domain of computational geometry [47] as well as data mining. In clustering, Voronoi polygons have been applied in order to cluster entities of a *single* type [34]. Voronoi polygons have also been used to establish neighbourhood relationships between entities of a *single* type (sensors) for the purposes of outlier detection [1]. As opposed to analyzing only a single entity-type, Voronoi polygons have been used for finding spatial association rules for *multiple* entity-types [10]; however, though the dataset contained multiple entity-types, the Voronoi polygon itself was created as if all entity-types were the same. This strategy does not take advantage of the fact that the entity-types are of different characteristics, and creates a neighbourhood structure that connects adjacent entities only.

A definition is required that creates meaningful neighbourhood relationships between multiple types of entities in spatial data while preserving the fact that they are of different types. People tend to go to the closest mall, grocery-store, hospital or airport, and the definition needs to take advantage of this. Voronoi diagrams accomplish this by partitioning a plane into regions, called Voronoi cells, which contain all the points that are closest to the entity contained in the Voronoi cell. Based on this idea, as the basis for the neighbourhood definition in this thesis, for each entity-type a different neighbourhood structure is created using Voronoi diagrams.

### 3.2.1 Voronoi Diagram

The definition of a Voronoi diagram is as follows [81]:

**Definition 4 – Voronoi Diagram** – *Given a set  $S$  of distinct entities  $e$  in  $d$ -dimensional Euclidean space  $\mathbb{R}^d$ , the Voronoi diagram is a partitioning of the space of  $S$  into  $e$  polyhedral regions. The region associated with entity  $e$  is called the Voronoi cell of  $e$  and is the space in  $\mathbb{R}^d$  that is closer to  $e$  than to any other entity in  $S$ .*



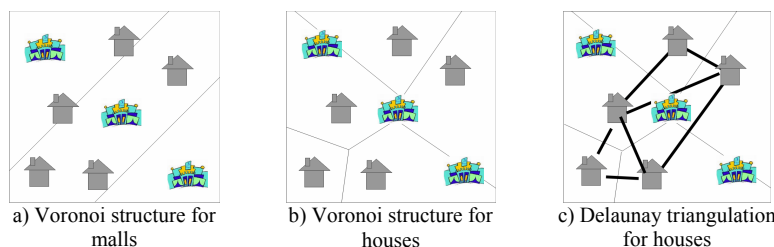
Sample Voronoi diagrams for two different entity-types are shown in Figure 18(a) and (b). In general, the size of Voronoi cells gives an indication of how frequent entities of the same type are in an area and thus is a good aid when determining the importance of the entity. The sparser the entities in the dataset, the larger the Voronoi region, and typically the more important the entity-type (ex: museums, airports, malls), as seen by contrasting malls and houses in Figure 18(a) and (b).

Voronoi diagrams can also adjust for changes in distribution for a single entity-type. If entity  $e^{i,q}$  is in a region where many entities of the same type are also located, then the *individual* entity  $e^{i,q}$  is less important than if  $e^{i,q}$  was far from any others of the same type. For example, a specific grocery store is not too important if it is surrounded by five other grocery stores since people have choices as to where to go. If that same store is far from other grocery stores, then people will have less choice and are going to depend on that grocery store. Voronoi diagrams are able to reflect this information: the Voronoi cell of  $e^{i,q}$  is going to be small if it is surrounded by other entities of the same type, and if entities of the same type are far, then the Voronoi cell is going to be large. This change in density is seen in Figure 19.

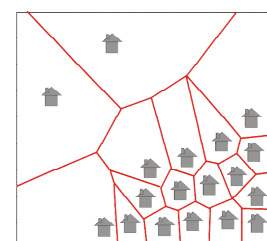
The Voronoi diagrams can also help with determining neighbours of entities by using the dual of Voronoi diagrams – the Delaunay triangulation (Figure 18c), which is defined as [81]:

**Definition 5 – Delaunay Triangulation** – *For a set of entities  $e$  in d-dimensional Euclidean space, the Delaunay triangulation is a triangulation  $DT(e)$  such that no point in  $e$  is inside the circum-hypersphere (circumcircle if  $d=2$ ) of any simplex (triangle if  $d=2$ ) in  $DT(e)$ .*

In essence, the Delaunay triangulation connects neighbouring Voronoi cells together, examples are shown in Figure 20(c) and (f). If Voronoi calculations are to be used for establishing neighbourhoods, then the structure yields a consistent and rigid neighbourhood which is not going to change, but most importantly, this is done without user parameters.



**Figure 18 – Sample Voronoi neighbourhood relationships for houses and malls**



**Figure 19 – Voronoi cells for varying distribution**

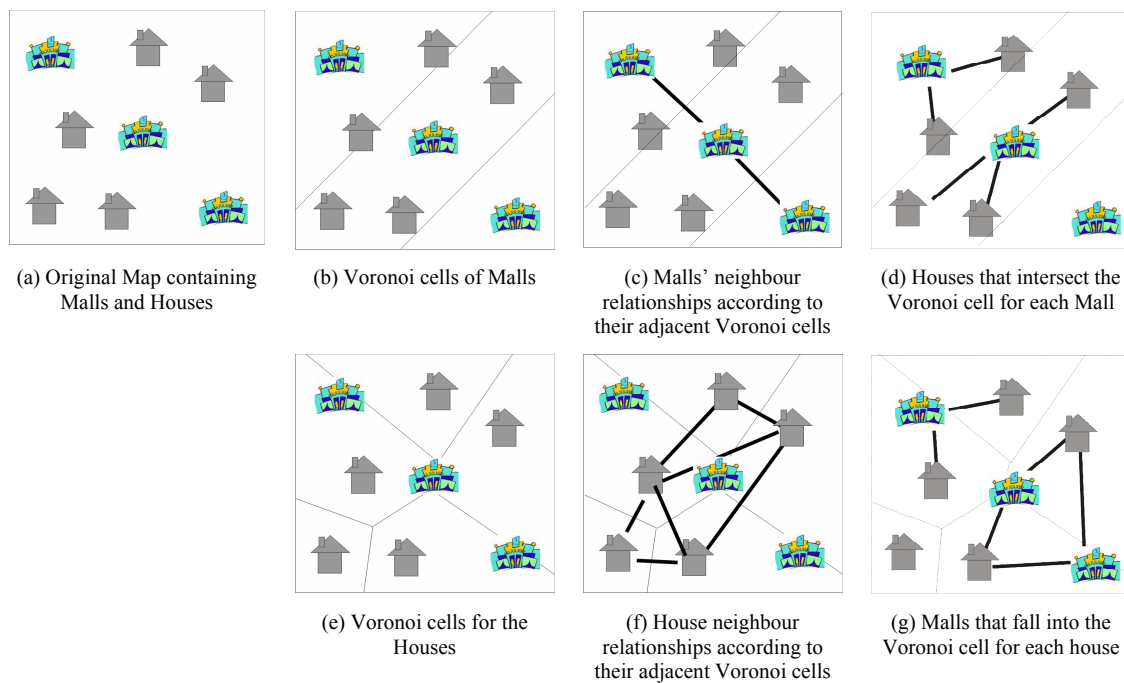
### 3.2.2 Asymmetric Voronoi-Based Neighbourhood Definition

By combining all entities on a map and using that as the input to the Voronoi algorithm, the resulting Voronoi cell structure is not desirable since each cell is going to be very small and the Delaunay triangulation is going to yield (by definition) adjacent relationships only. An alternative approach is to calculate the Voronoi diagram for each entity-type  $t_i$  and assign all other entities of different types into the Voronoi cell of each entity of type  $t_i$ . The following definition does this by establishing neighbourhood relationships between entities of the same and different types:

**Definition 6 – Asymmetric Voronoi Neighbourhood** – *The Asymmetrical Voronoi Neighbourhood defines two entities,  $e^{i,q}$  and  $e^{j,r}$ , as neighbours iff:*

- $e^{i,q}$  intersects the Voronoi cell of  $e^{j,r}$ , where  $e^{i,q}$  and  $e^{j,r}$  are different entity-types, i.e.:  $i \neq j$ , or,
- the Voronoi cells of  $e^{i,q}$  and  $e^{j,r}$  are adjacent, and  $e^{i,q}$  and  $e^{j,r}$  are of the same type, i.e.:  $i = j$ .

This definition is applied to part (a) in Figure 20 which illustrates a map containing 5 houses and 3 malls. The Voronoi diagram from the perspective of the malls is calculated (b), the adjacent Voronoi cells for malls are connected as neighbours (c), and finally the houses that belong to each mall are also connected to the corresponding mall (d). Similarly, the same is done from the

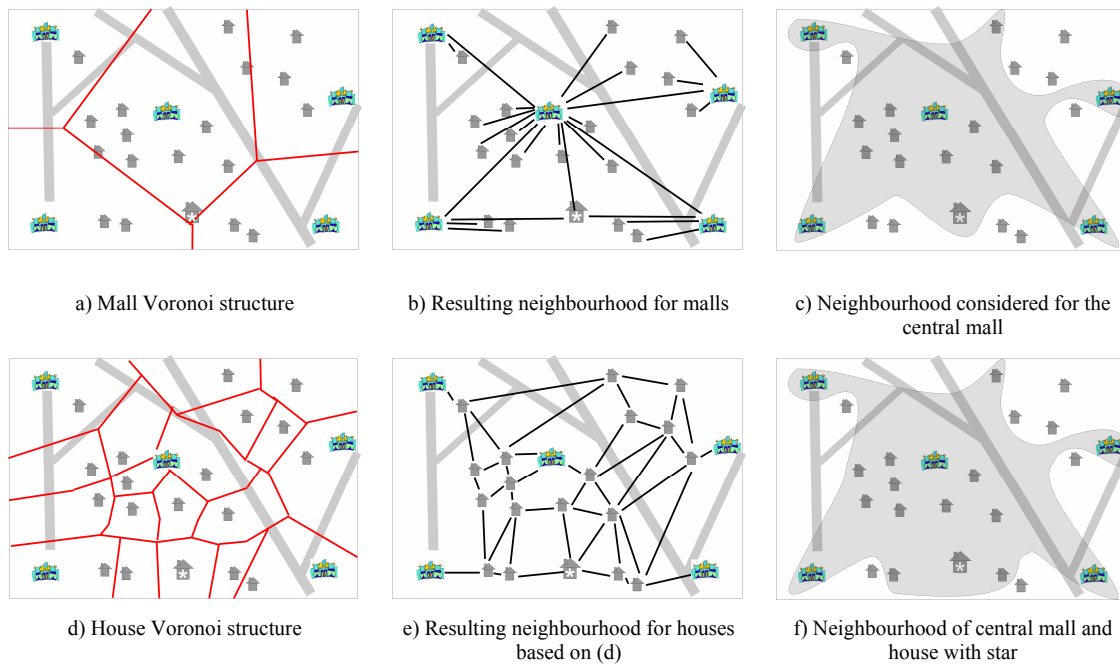


**Figure 20 – Establishing neighbourhood relationships using Definition 6.**

perspective of the houses ( $e, f$  and  $g$ ) giving all the neighbourhood relationships needed.

There is a very significant drawback to the above definition however: When exploring the neighbouring entities, the neighbourhood relationships are not symmetric (an entity  $e^{i,q}$  can be a neighbour to  $e^{j,r}$  but  $e^{j,r}$  might not be a neighbour to  $e^{i,q}$ ). This could have the consequence that, in a more complex example (Figure 21) the mall in the center is going to have many houses as neighbours, but only a few houses is going to have the mall as neighbours. Assume the neighbourhoods are being explored by the data mining algorithm and the central mall and its neighbourhood is being analyzed (c). Further assume that the best condition the classification algorithm finds is satisfied by a house in the neighbourhood of the central mall (this house is indicated with a star in Figure 21). Continuing the analysis, the set of entities considered in the next step is going to almost be identical to the set that was just previously considered since the new addition only has 2 new neighbours; hence the classification algorithm would be considering almost the same situation as it did previously. In this fashion, the classification algorithm would be ‘stuck’ considering almost the same set of entities over and over unless by chance it encountered a house that was adjacent to an entity of a different type.

The *house* entity-type is ‘dense’, meaning that the houses are close together and thus their neighbourhoods are small. Adding a new house to the analysis does not provide more than a few



**Figure 21 – Simple unsymmetrical neighbourhood definition (shadings represent roads)**

extra neighbours to base classification on, this can be seen by comparing (c) to (f). Hence further analysis is based on almost the same entities and the classification progress would proceed slowly. For example, using (c), in order to reach a non-*house* entity-type which could add something new to the classification process, the algorithm would require the addition of 3 more houses which could require multiple iterations of the classification algorithm. This leads to very slow classification progress.

In the context of the illustration in Figure 21, a *mall* could be profitable if there is a *house* which is really expensive within the neighbourhood of *mall*, but a *house* itself could be really expensive because it is in the neighbourhood of 3 malls (Figure 21b). The second component, *house* being in the neighbourhood of 3 malls, cannot be found with this definition. The reason for this is that when the algorithm looks at the rationale for the expensive house there are only other houses as neighbours and no *mall* is reachable (Figure 21e).

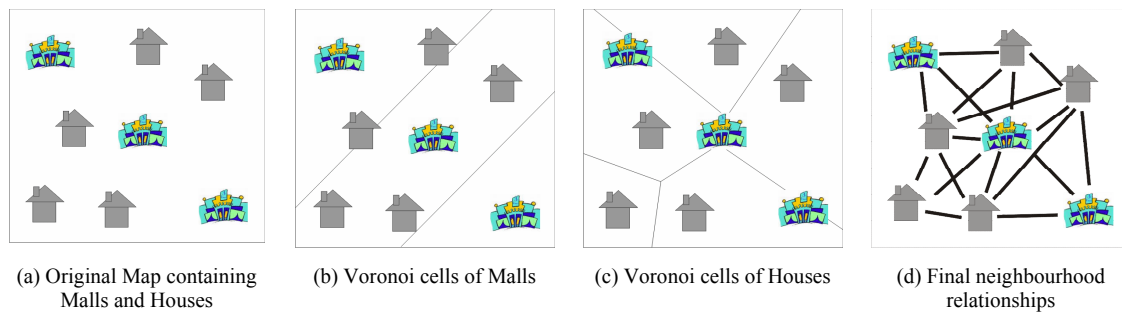
### 3.2.3 Symmetric Voronoi-Based Neighbourhood Definition

The drawback of the asymmetrical definition (Chapter 3.2.2) can be eliminated by modifying a portion of the definition in order to enforce symmetry. The symmetric version of the Voronoi Neighbourhood becomes:

**Definition 7 – Symmetric Voronoi Neighbourhood** – *The Symmetric Voronoi Neighbourhood defines two entities,  $e^{i,q}$  and  $e^{j,r}$ , as neighbours iff:*

- $e^{i,q}$  intersects the Voronoi cell of  $e^{j,r}$  or  $e^{j,r}$  intersects the Voronoi cell of  $e^{i,q}$ , where  $e^{i,q}$  and  $e^{j,r}$  are different entity-types, i.e.:  $i \neq j$ , or,
- the Voronoi cells of  $e^{i,q}$  and  $e^{j,r}$  are adjacent, and  $e^{i,q}$  and  $e^{j,r}$  are of the same type, i.e.:  $i = j$ .

For example, the neighbourhood structure for each house  $e^{House,q}$  is the union of the entities which neighbour  $e^{House,q}$  and the entities to which  $e^{House,q}$  is a neighbour. Figure 22 illustrates the Voronoi



**Figure 22 – Establishing neighbourhood relationships using Definition 7**

structure for both malls and houses, and the resulting neighbourhood relationships using this definition. This definition has a unique property: each entity-type will have at least one neighbourhood relationship with an entity from all entity-types.

**Proposition 1** – *Let  $t_i$  and  $t_j$  represent two entity-types. Every entity of type  $t_i$  has at least one neighbouring entity of type  $t_j$ .*

**Proof** – *The union of Voronoi cells of entities of type  $t_i$  covers the entire spatial domain with every entity of type  $t_i$  intersecting the cell of at least one entity of type  $t_j$ . Since each entity of type  $t_i$  intersects a cell of type  $t_j$ , this yields at least one neighbour relationship between entities of type  $t_i$  and some entities of type  $t_j$ , thus in total each entity of type  $t_i$  is related to  $1 \leq n \leq |j|$  entities of  $t_j$ . ■*

This symmetrical definition has the advantage that if the goal is to classify types containing a lot of entities (ex: houses) then the algorithm can always find a neighbouring entity of every other type. It never gets ‘stuck’ considering almost the same set of entities. For example, each house has at least one mall (along with at least one entity of every other type) as neighbour. In effect, the neighbourhood structure for each house  $e^{House,q}$  becomes the union of the entities which neighbour  $e^{House,q}$  and the entities to which  $e^{House,q}$  is a neighbour. The result from the previous example can be found in Figure 22(d). With  $|T|$  types and  $Q$  entities per type, this would imply that there are at least  $O(|T|^2Q)$  neighbourhood relationships with entities of different types, at least  $|T|-1$  **per entity**.

The advantages of this definition of a Symmetrical Voronoi Neighbourhood are that:

- it creates a more complex neighbourhood structure than direct neighbours,
- it is **uniform** across all of the map,
- it is intuitive,
- it does not require any user parameters.

The Symmetrical Voronoi-Based Neighbourhood cannot, and purposely does not, find relationships at arbitrarily fixed proximities (such as the buffer-zone). It also does not capture neighbour-of-neighbour relationships (this is left up to the rule-learner to explore). The reason for that is based on Tobler’s First Law of Geography in that patterns which do not appear in close proximity, based on the distribution of entities, are not interesting.

### 3.3 Comparison of the Efficiency of Different Neighbourhoods

This section analyzes the runtime complexities for computing the buffer-zone and the Voronoi-diagram based neighbourhood definitions. The classification model is constructed around a set of

entities  $e^r$  of entity-type  $t_\tau$ , called the target entity type. Assume that there are  $|T|$  entity-types, where  $T \in \{t_1, t_2, \dots, t_i, \dots, t_\tau, \dots, t_{|T|}\}$ . Let  $Q_i$  represent the number of entities of type  $t_i$  and let  $Q$  represent the total number of entities. The target entity-type has  $Q_\tau$  entities, and the rest of the dataset has  $\sum_{i=1, i \neq \tau}^{|T|} Q_i = Q - Q_\tau$  entities. The Buffer-Zone and the Symmetric Voronoi Neighbourhood are compared below and summarized in Figure 23.

### 3.3.1 Determining Neighbourhood using Buffer-Zones

The buffer-zone definition relies on the distance between two entities being smaller than a threshold distance. Thus, each entity of type  $t_i$  will need to be compared against all entities of other types, and against all entities of the same type:

$$= \sum_{i=1}^{|T|} Q_i (\text{comparing against entities not of } t_i + \text{comparing against entities of } t_i)$$

$$= \sum_{i=1}^{|T|} Q_i ((Q - Q_i) + Q_i) = O(Q^2).$$

### 3.3.2 Analyzing Neighbourhood using Buffer-Zones

Assume that the buffer-zone for each entity covers  $p\%$  of all the area covered by the dataset. Thus, on average,  $Qp\%$  entities are going to be neighbours of each entity. To analyze this neighbourhood structure, as the classifier will do, for each entity-type  $t_i$  that needs to be analyzed, the  $Q_i$  entities of that type are going to be compared to their  $Qp\%$  neighbours, leading to a complexity of  $O(Q_i \times Qp\%) = O(Q_i Q)$ .

	<i>Buffer-Zone</i>	<i>Voronoi-based neighbourhood relationship</i>
<i>Cost of determining neighbourhood relationships</i>	$=O(Q^2)$	$=O(Q^2)$
<i>Analysis (Run-Time) Cost</i>	$=O(Q_i Q)$	$=O(Q)$

$Q$  = entities in the database  
 $Q_i$  = entities of type  $t_i$

**Figure 23 – Runtime analysis**

### 3.3.3 Determining Neighbourhood using Voronoi Neighbourhoods

With a Voronoi neighbourhood relationship enforced, the runtime complexity is different. For each entity of type  $t_i$ , to determine relationships between entities of different types, the Voronoi cell is explicitly computed, which has runtime  $O(Q_i \log Q_i)$  [11], then all  $Q-Q_i$  entities not of that type are assigned to the Voronoi cell they fall into. For relationships between entities of the same type, the Delaunay Triangulation must be calculated which can be done during the calculation of the Voronoi cells and hence presents no extra runtime requirements. The total cost of this pre-processing is:

$$\begin{aligned}
&= \sum_{i=1}^{|T|} (\text{Voronoi diagram for } Q_i \text{ entities of type } t_i + \text{assigning } Q-Q_i \text{ entities to } Q_i \text{ cells}) \\
&= \sum_{i=1}^{|T|} (Q_i \log Q_i + Q_i(Q-Q_i)) \\
&= \sum_{i=1}^{|T|} (Q_i(\log Q_i + Q - Q_i)) \\
&\leq \sum_{i=1}^{|T|} (Q_i Q) && \text{since } \log(Q_i) < Q_i \ll Q \\
&= Q \sum_{i=1}^{|T|} Q_i = O(Q^2) && \text{since } \sum_{i=1}^{|T|} (Q_i) = Q \quad \text{i.e. the sum of entities per class} \\
&&& \text{over all classes equals the number of entities.}
\end{aligned}$$

### 3.3.4 Analyzing Neighbourhoods using Voronoi Neighbourhoods

At this point, since the Voronoi-diagram partitions the dataset, each of the  $Q_i$  entities of that type are going to have, on average,  $Q/Q_i$  entities assigned to it as neighbours. To analyze this neighbourhood structure, as the classifier will do, for each entity-type  $t_i$  that needs to be analyzed, each entity in  $e^i$  is compared only to its neighbours, leading to  $O(Q/Q_i) = O(Q)$  comparisons.

Hence introducing the Voronoi neighbourhood has a similar pre-processing cost but only linear cost when analyzing this neighbourhood.

### **3.4 Discussion**

The Voronoi-diagram based neighbourhood definition makes intuitive sense, based on the examples and other research findings shown above. It also is able to address the shortcomings of the buffer-zone especially the need for a user specified parameter. Theoretical runtime comparisons are also similar to the traditional buffer-zones, and hence complexity is not increased significantly.

However, there might be times when, due to domain requirements, neither neighbourhood definition works intuitively. The police might define a city block as a neighbourhood, for example. In these instances, the neighbourhood definition can be modified as required, as the exact definition does not impact the rule-learning process that is presented next. The rule-learner simply treats the neighbourhood definition as a black box.



## CHAPTER FOUR: EXTENDING ILP FOR AGGREGATION AND SPATIAL DATA

Although there are approaches to spatial classification other than ILP, such as kernel methods, as customary in Multi-Relational Data Mining, this thesis is going to also use Inductive Logic Programming (ILP) [18, 60] as its underlying formalism. One of the advantages of ILP is its ability to handle multi-relational data, as opposed to limited to propositional data, which would have presented problems associated with propositionalization. As opposed to kernel methods, for example, ILP is also able to produce rules that are more understandable.

The word *induction* refers to the process of deriving general principles from particular facts or instances, which is what ILP accomplishes when applied to data as part of a data mining task. In ILP, the following components are given:

- a set of known **positive and negative instances** of data (or only the positive instances are given with the *closed world assumption* that everything not given is negative),
- some **background knowledge** used to allow for more concise conditions (for example: *A is a parent of B, and A is male, then A is the father of B*),
- a **language** specifying syntactic restrictions (for example, someone cannot have 3 parents).

The task of ILP is to learn a model which entails as many positive examples and as few negative examples as possible. If the model is used for classification, one of the entity-types (called the target entity) has to be selected, around which the classification model is built. As in single-table classification, one of the (categorical) descriptive features of this entity-type is selected as the class label. The goal of the classification task is to predict the class label for new entities. For example, the class label of the entity *mall* is the value ‘Yes’ or ‘No’ of feature *Profitable*. Although the method detailed in this Chapter can be expressed for non-Boolean class labels, this thesis deals with only Boolean class labels.

Existing non-spatial ILP approaches, such as CrossMine [100, 101] and MDRTL-2 [7], have been used to mine multi-relational data, by learning classification models from a relational dataset.

These models however incorporated only the existential aggregation operator. The drawback to using the existential operator is that it is a very simplistic aggregation, testing whether a relationship between two entities exists. Spatial data is dense (spatially) with many entities being the same type (houses, for example) hence the effect of a single entity becomes relatively negligible. A shopping center for example is successful not because there is a single wealthy household in its neighbourhood, but rather because the average purchasing power of the neighbourhood that it is in is above a certain value. The crime-rate in a certain area is typically not considered high because a certain building got robbed many times, but because the aggregate number of robberies over all buildings in the area is above a certain value. Dealing with aggregation in spatial data is much more of a necessity since spatial data contains information much richer than what the existential operator can describe. In order to capture that information, aggregation other than the existential operator needs to be used. For example, using solely the existential quantifier the relationship “*an entity is related to 5 entities whose average value is 20*” cannot be expressed. To fully exploit the potential of multi-relational classification, aggregate information needs to be used during the process. Aggregation operators are used to summarize a set of values into a single value or a vector of values which describe the original set. The original set often comes from the ‘many’ part of a one- or many-to-many relationship for each entity. Unlike global features, which describe the entire dataset, aggregation operators act similar to local features, describing the area around a specific entity [59].

This Chapter first provides an overview of aggregation (Chapter 4.1). Since this thesis uses Inductive Logic Programming (ILP) to express the results of classification, this process is going to need to involve aggregation. First, the standard state-of-the-art ILP terminology is discussed, including existing aggregation techniques (Chapter 4.2), followed by novel multi-feature and spatial extensions introduced in this thesis (Chapter 4.3). Although this thesis uses ILP for classification, the data itself is stored in a database. The connection between ILP and databases is discussed last (Chapter 4.4).

## 4.1 Preliminaries

Aggregation can be performed in many different ways, such as aggregating over a single argument of a  $P/\alpha$  predicate, also called a feature, or aggregating features over multiple neighbouring entity-types (Chapter 4.1.1). Aggregation can also be applied to different types of data (Chapter 4.1.2), each with specific restrictions. There are a few approaches to aggregation

that are considered state-of-the-art, such as Virtual Features (Chapter 4.1.3) and Selection Graphs (Chapter 4.1.4), but neither is appropriate for spatial data.

#### 4.1.1 Types of Aggregation

The aggregations possible involving a single entity-type  $t_i$  include [78]:

- **Simple aggregation.** A mapping of zero or more independent single values to a single categorical/numerical value, for example: sum, mean, max. This type of aggregation can be used to summarize a single feature occurring in multiple entities of type  $t_i$ .
- **Multi-dimensional aggregation.** Mapping of zero or more entities of type  $t_i$ , each with  $n$  features (i.e.: a feature vector) to a single or multiple values. This type of aggregation can be used to capture relationships between two or more features, by using the covariance for example.
- **Multi-type aggregation.** Mapping zero or more entities of type  $t_i$  (possibly feature vectors with different numbers of features) to a single or multiple values. For example, the “total value of products that a customer has returned” would require the aggregation of a number of numerical and categorical data pieces.

Aside from doing multiple types of aggregations within a *single* entity-type, aggregation can also be done involving *multiple* entity-types. The difference is that in the latter case entities of type  $t_j$  which are related to  $t_i$  can be aggregated even though the number and types of features can be different. The types of aggregations which involve relationships between entities are [78]:

- **Propositional.** This aggregation involves a relationship between entities of type  $t_i$  and  $t_j$  where there is a 1:1 correspondence between the entities. Thus, for each entity of type  $t_i$  there is a single entity of type  $t_j$ . In this situation, no aggregation needs to be performed since each entity is related to only a single entity of another type.
- **Independent features.** This type of aggregation summarizes a feature of entities of type  $t_j$  related to entities of type  $t_i$ . The aggregations that are possible include *max*, *count*, etc. For example, calculate the maximum size over all neighbouring houses to each mall.
- **Dependent features.** This type of aggregation summarizes relationships between multiple features of the entities of  $t_j$  related to each entity of type  $t_i$ . The aggregations that are possible include aggregations such as *covariance* or *correlation*. For example, calculate the relationship between house-hold income and house-size for each house.

This thesis introduces a framework capable of dealing with the above aggregation types.

#### 4.1.2 Aggregation of Different Data Types

Aside from operations on unstructured text features, such as names and descriptions, there are four basic types of data each requiring different types of aggregations: date, numeric, categorical and ordinal values. The corresponding aggregation operators for these types are discussed below:

- **Date data-type.** Adding dates, although possible mathematically is non-sensical. Average, mean and median can be applied to dates using the offset of a given set of dates from a fixed date.
- **Numerical data-type.** This is the most flexible type of data to which generally any kind of mathematical operation could be applied. There could be domain restrictions to the numeric being an integer-type for example.
- **Categorical data-type.** This type of data contains a fixed set of possible values. There does not need to be a concept of one category being ‘better’ or ‘greater’ than another category.
- **Ordinal data-type.** This is categorical data, with an ordering on the distinct values which imply that the ordinal values can be compared using the  $>$  and  $<$  operators. Aggregation operators applicable to categorical data are also applicable to ordinal data.
- **Spatial data-type.** Spatial entities have features, such as location, which might not be explicitly given, and hence must be ‘extracted’ from the spatial entity itself. Once this extraction process is done, the resulting feature-value is going to be of one of the above data-types.

This thesis will introduce a framework capable of aggregating all of the above data types.

#### 4.1.3 Virtual Features

Aggregation within ILP can be achieved by the use of virtual features [78]. A virtual feature is a feature which is ‘invented’ and then added to entities in  $e^{\tau}$  where the new feature-value of  $e^{\tau,q}$  (entity  $q$  of type  $t_{\tau}$ ) is derived from the aggregated values of entities related to  $e^{\tau,q}$ . These virtual features can aggregate a single feature, multiple features, or even categorical features. Once the new features are added to  $e^{\tau}$ , they can be used as consideration during the classification process.

The values of non-categorical features can be calculated by an aggregation function such as *sum* or *max*, but can be much more complex than those.

One approach for aggregation is to create a ‘case vector’ (CV) describing the values related to each labelled target entity  $e^{t,q}$  [78]. In a simple case, a CV can describe the most frequent values of a categorical feature related to  $e^{t,q}$ . Thus there is one CV per entity in  $e^t$ . For each class, from all the CVs of entities in the training data which belong to that class, a reference vector (RV) and variance vector (VV) is constructed containing the sum and variance of all the CVs respectively. During classification, when trying to assign a label to a specific unlabelled instance of  $e^{t,i}$ , a case vector (CV) is constructed from the features of  $e^{t,i}$ , and a class label is determined by calculating the distances between the specific CV and all RVs. The CV inherits the class-label of the RV to which it is closest. Distance can be calculated via any distance measure such as edit distance, cosine distance or the Mahalanobis distance  $MD(x, y) = \sqrt{(x - y)'C^{-1}(x - y)}$ .

Virtual features have also been used to summarize high-dimensionality features, such as product IDs [79]. In this instance, using training data, an RV is constructed for both classes along with an RV for the global data-set. The RVs are constructed from all the feature-values from the entities linked to each target entity. Distribution Vectors (DV), essentially normalized RVs, are also constructed to approximate the distributions within each class and globally. The classification model will use these DVs to find differences in the distributions with the assumption that all entities related to the target entities have been created from the same distribution. During classification, the classifier labels each entity by determining which distribution generated the CV, by calculating the distance between the CV and RV. For numerical attributes, RVs can be constructed from standard aggregates, such as MIN or SUM.

Another approach is to simply add a series of features to the target entity, created via aggregation from the information contained in features of related entities. Multiple algorithms, such as RSD (Relational Subgroup Discover) [62], SINUS [60] and RELAGGS [58] take this approach. RSD and SINUS are both logic-oriented approaches (whereas RELAGGS is database-oriented) and perform virtual feature creation in a similar fashion, with minor differences, such as SINUS allowing the user to constrain the number of literals while RSD can constrain the variable depth and occurrences of specified predicates [57].

The technique of using virtual features has a major drawback. Once the virtual features are created and added to the entity as a new feature, they are fixed. Conditions applied to any of the neighbours of a specific entity can change the set of neighbours, but this change is not going to be

reflected in the values of the virtual features, since those were pre-computed. Classification rules based on these virtual features may not yield accurate results.

#### 4.1.4 Selection Graphs

Selection graphs, defined below, have previously been used to represent the results of the classification process [7].

**Definition 8 – Selection Graph** – A selection graph is a directed graph  $S$ , with nodes  $N$  representing a set of entities, and edges  $E$  representing some conditions  $C$  on  $N$ . Each node  $N$  contains the set of entities which satisfy all the conditions between  $N$  and the root of  $S$ .

Each path in  $S$  can be represented by an SQL query. Each node along the path is equivalent to an additional condition in an SQL query's WHERE clause. The selection graph is refined by adding conditions as branches to an existing tree. This technique of representing rules as selection graphs was extended by including aggregations, where the edges, which originally could represent only the existential quantifier, now are complimented with aggregation functions (*sum*, *max*, etc) [52]. The authors use a measure called Novelty [61], the difference between the joint probability and the individual probabilities of features  $K$  and  $L$ :

$$\text{Nov}(K \leftarrow L) = p(KL) - p(K) - p(L).$$

Selection graphs are a way of representing SQL queries or inductive logic constructs since simple mappings exist between them [52]. In order to incorporate aggregation that is more complex than simple aggregation, the framework of selection graphs needs to be extended by introducing conditions on the edges capable of dealing with spatial aggregation and aggregation of multiple features. These extensions are similar in complexity to the extensions that inductive logic programming requires.

## 4.2 ILP Rule Language

One method of dealing with information contained in multi-relational databases is to transcribe the information contained in them into logic-rules against which hypotheses can be evaluated [5]. This is done by converting all entities, their features and spatial relations, into the language of Inductive Logic Programming (ILP). This Chapter introduces the fundamentals of ILP (Chapter 4.2.1), before moving on to expressing aggregation with ILP (Chapter 4.2.2).

### 4.2.1 Basic ILP Rule Formalism

Rules are built using the format of first-order logic (FOL). The formal syntax which forms the ILP structures are defined below [82, 26, 56, 71, 72, 18, 27]. ILP can be thought of as a language, with a distinct alphabet, words and structure. With each language, in order to construct proper sentences, first the alphabet must be established.

**Definition 9 – Alphabet** – *An ILP alphabet consists of the following:*

- A set of **constants**, denoted by lowercase letters  $a, b, c, \dots$ .
- A set of **variables**, denoted by uppercase letters  $X, Y, Z, \dots$ .
- A set of **functions**, each having an arity  $\alpha$  (a natural number) assigned to it. Functions are denoted by uppercase letters  $f, g, h, \dots$ , and written as  $f/\alpha$ . A function of arity 0 is a constant.
- A non-empty set of **predicates**, each having an arity  $\alpha$  (a natural number) assigned to it. Predicates are denoted by uppercase letters  $P, Q, R, \dots$ , and written as  $P/\alpha$ . A predicate is a mapping from a domain  $D^n$ ,  $n > 0$ , to a Boolean value.
- The set of five **logical connectives**  $\{\neg, \vee, \wedge, \leftrightarrow, \leftarrow\}$  respectively denoting logical negation, disjunction (logical OR), conjunction (logical AND), equivalence and implication.
- Two **quantifiers**: the universal quantifier  $\forall$  and the existential quantifier  $\exists$ .
- Three **punctuation symbols**:  $'(, ')$  and  $'\text{'}$ .

Similarly to natural languages, where entities of the described world are represented by nouns, entities in predicate logic are represented by *terms*:

**Definition 10 – Terms** – *The set of terms of a given alphabet are recursively defined as follows:*

- constants and variables are terms;
- $f/\alpha$  is a function with arity  $\alpha$  where each of the  $n$  arguments is a term.

Just as in natural languages only certain combinations of words form meaningful sentences; the sentences in predicate logic are formed via well-formed formulas (or simply *formulas*) and are defined as:

**Definition 11 – Formulas** – *A formula is valid if:*

- $P/\alpha$  is a predicate and  $t_1, t_2, \dots, t_\alpha$  are terms, then  $P(t_1, t_2, \dots, t_\alpha)$  is a formula;
- $F$  and  $G$  are formulas, then so are  $\neg F$ ,  $F \wedge G$ ,  $F \vee G$ ,  $F \leftarrow G$ ,  $F \leftrightarrow G$ ;
- $F$  is a formula,  $x$  a variable, then  $\forall xF$  and  $\exists xF$  are also formulas.

Those *formulas* which are not atomic, for example  $P(a) \vee Q(b)$ , are called *composite formulas*.

Intuitively a *term* refers to an entity while a *formula* represents an assertion about entities.

**Definition 12 – Atom** – *The smallest possible formula is called an atom and is constructed by assigning  $\alpha$  terms to a predicate of arity  $P/\alpha$ . If all  $\alpha$  terms are constants then the atom is called a ground atom.*

For example, a predicate *neighbour* can be applied to the pair of variables *M* and *R*, representing atoms *mall(M)* and *road(R)*, producing an *atom* of arity 2: *neighbour(M, R)*. If the constants ‘Loughheed Mall’ and ‘Highway 1’ are substituted for *M* and *R* respectively, then there is a *ground atom*: *neighbour(‘Loughheed Mall’, ‘Highway 1’)*.

**Definition 13 – Literals** – A literal is either an atom, called a positive literal, or its negation, called a negative literal, or a comparison  $\theta$  of a term to a constant value *T*.

Intuitively, the distinction between the three definitions of *predicate*, *atom* and *literal* can be seen as follows. A *predicate* is a mapping from some domain of some values to a Boolean value, similar to the mathematical mapping *squared* which maps real numbers to the domain of non-negative real numbers. An *atom* however actually applies some *terms* to the *predicate*, like applying the variable *x* to the mathematical mapping *squared* to get  $f(x) = x^2$ . Although in this thesis all *literals* are going to be positive, the meanings of *atoms* and *literals* are still distinct since *literals* can also express a comparison of a *term* to a *constant*. This would be similar to specifying that the *literal* maps to *true* only if the mapped value is less than 100, for example:  $x^2 < 100$ . This cannot be expressed with an *atom*.

**Definition 14 – Full Clause** – A full clause is a set of literals of the form  $\text{head} \leftarrow \text{body}$ . Both the head and body are conjunctions of literals  $L_i$ , denoted as  $L_1 \wedge L_2 \wedge \dots \wedge L_n$ , or simply as  $L_1, L_2, \dots, L_n$ .

**Definition 15 – Horn Clause** – A Horn clause is a clause of the form  $\text{head} \leftarrow \text{body}$  where the head is made up of at most a single positive literal  $L_0$ , while the body is a conjunction of literals  $L_i$ , denoted as  $L_1 \wedge L_2 \wedge \dots \wedge L_n$ , or simply represented as  $L_1, L_2, \dots, L_n$ . Hence a Horn Clause is of the form  $L_0 \leftarrow L_1, L_2, \dots, L_n$ .

As an example, the notion “if there exists a mall which is close to a house, then the house is a neighbour to the mall” is written

$$R2: \text{neighbour}(H,M) \leftarrow \text{mall}(M), \text{close}(M,H), \text{house}(H)$$

Since a *formula* can contain implication ( $F \leftarrow G$ ) and conjunction ( $F \wedge G$ ) thus a *full clause* is simply another representation of a specific type of *formula*. Similarly, a *Horn clause* is a type of *formula* where the implied *head* contains only a single positive *literal*.

**Definition 16 – Classification Rule** – A classification rule is of the form  $\text{head} \leftarrow \text{body}$ . The head is made up of at most a single positive literal  $L_0$  specifying one of the values of the term used for the classification task. The body is a conjunction of literals  $L_i$ , denoted as  $L_1 \wedge L_2 \wedge \dots \wedge L_n$ , or simply as  $L_1, L_2, \dots, L_n$ .



The goal of multi-relational classification using ILP is to find rules, using the format of *Horn clauses*, which form a well defined and accurate model. As an example, assuming the task is to classify malls as profitable or not profitable, the following rule can be constructed:

$$R3: \text{profit}(M, \text{'Yes'}) \leftarrow \text{mall}(M), \text{neighbour}(M, H), \text{house}(H)$$

R3 states that “if there exists a house neighbouring a mall then the mall is profitable”. The literals  $\text{mall}(M)$ ,  $\text{neighbour}(M, H)$  and  $\text{house}(H)$  make up the body of the rule and  $\text{profit}(M, \text{'Yes'})$  makes up the head. When a predicate  $\text{mall}$  is applied to the term  $M$  it creates the atom  $\text{mall}(M)$ .  $\text{house}(H)$  is implicitly existentially quantified, which is defined as follows:

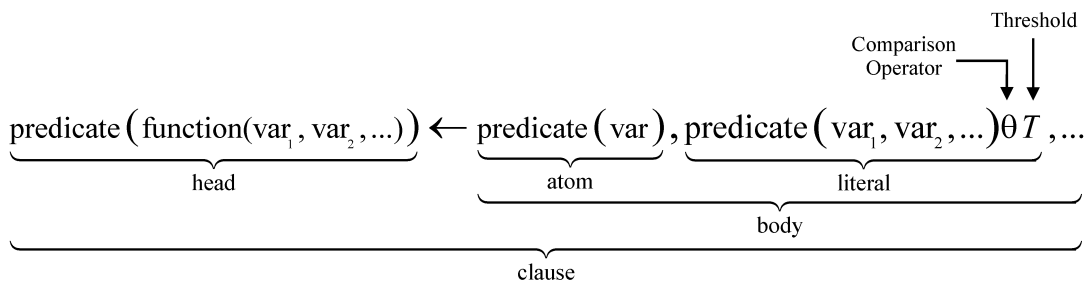
**Definition 17 – Existential Quantification** – The existential quantifier (EQ), symbolized with the symbol  $\exists$ , expresses that the literal holds for some (at least one) value of the variable in the literal. EQ literals can take the form  $\text{predicate}(\text{term})$  in the case of unary predicates or  $\text{predicate}(\text{term}_1, \text{term}_2, \dots, \text{term}_a)$  in the case of predicates with arity  $a$ .

In *clauses*, the symbol  $\exists$  is typically left out of the notation. One of the forms of a *literal* performs a comparison, using a *comparison operator*, of a *term* to a constant value, called a *threshold value*. They are defined as follows:

**Definition 18 – Comparison Operator** – A comparison operator  $\theta$  is one of the following arithmetic operators:  $=, <, <=, >, >=$ .

**Definition 19 – Threshold** – The threshold value  $T$  is a constant that the term in a literal is compared against. It is used to divide the dataset into two partitions, one not meeting the criterion expressed by the comparison operator and threshold (these are said to be not covered by the literal), the other one matching (covered by) the comparison operator and threshold. It has the format  $\text{literal } \theta T$ .

A summary of the above is illustrated in Figure 24.



**Figure 24 – Horn Clause format**

### 4.2.2 ILP Rule Language Involving Aggregation

Typically the number of spatial entities in a spatial dataset (a city for example) is very large, possibly on the order of millions. Clearly, the importance of any single entity, when in the presence of thousands of other entities of the same type, is questionable. From above, take as a running example the sample rule:

$$R3: \text{profit}(M, 'Yes') \leftarrow \text{mall}(M), \text{neighbour}(M, H), \text{house}(H)$$

It is relatively weak since all malls would most likely have some neighbouring houses. By adding an additional literal, for example, that the income at that house is greater than \$50,000, it is possible to create the rule:

$$R4: \text{profit}(M, 'Yes') \leftarrow \text{mall}(M), \text{neighbour}(M, H), \text{house}(H), \text{income}(H, I), I > 50,000$$

Even R4 is relatively weak since it is highly unlikely that a mall is profitable because of some distinct feature of a single neighbouring house unique only to profitable malls. Since spatial data is, in general, (spatially) dense with many entities neighbouring entities of another type hence the effect of a single entity is negligible. A mall, for example, is going to be successful not because there exists a single house in its neighbourhood with high income, but rather because the average purchasing power of the neighbourhood that it is in is above a certain value. The crime-rate in a certain area is typically not considered high because a certain building got robbed many times, but because the aggregate number of robberies over all buildings in the area is above a threshold. These notions cannot be expressed by the existential quantifier. To solve this problem, the ILP formalism was extended to allow for the aggregation over a single argument of a  $P/a$  predicate, referred to as a feature, to be used in ILP via *single-feature aggregation functions* as follows [98]:

**Definition 20 – Single-Feature Aggregation Function** – A single-feature aggregation function maps a bag of elements from the domain of the feature being aggregated to a single value from another (possibly different) domain.

This function is integrated into ILP via *single-feature aggregation literals*:

**Definition 21 – Single-Feature Aggregation Literal** – A single-feature aggregation (SFA) literal has the form  $\text{Agg}(\text{input}, \{\text{conditions}\}, \text{result})$  where *input* is a variable specifying the bag of feature values to be aggregated by *Agg*, constrained by *conditions*, and *result* is an output variable referencing the result of the aggregation.

With SFA literals another literal in the rule can reference *result* in order to add further constraints. For example, R2 could be extended by the SFA literal *avg* (average) as follows:

$$R5: \text{profit}(M, 'Yes') \leftarrow \text{mall}(M), \text{avg}(\text{income}(H), \{\text{neighbour}(M, H), \text{house}(H)\}, A), A > 50,000$$

Rule R5 denotes that “a mall is profitable if the average income of neighbouring houses is above \$50,000”. Figure 25 shows a compilation of SFA literals from [78, 42, 98] and the SQL language [52]. These literals describe the major properties of the input dataset and are non-redundant (with the exception of  $avg = \frac{sum}{count}$ , but  $avg$  does not increase computational complexity, and is a critical function). The advantage of including SFA literals in classification rules is an increase in accuracy, coverage and expressiveness [52]. The disadvantage is an increase in computational complexity.

### 4.3 Extensions of ILP for Spatial Classification

The ILP formalism cannot express feature-dependencies or spatial aggregation. It is constrained by the existential quantifier and aggregation of a single feature [98]. In the context of spatial entities, it is known that dependencies between features of an entity exist. This is illustrated by Tobler’s First Law Of Geography which states that ‘everything is related to everything else; but that near things are more related than those far apart’ [95]. Tobler’s law clearly states that there is a relationship between the distance of two entities and their feature values. In order to mine for useful spatial classifiers, this dependency must be exploited along with the information contained in the implicit spatial dataset, such as spatial features and relationships.

Literal	Result	Applicable to			
		Numeric	Date	Ordinal	Categorical
<b>sum</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	the total of all <i>input</i> values	√			
<b>min</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	the smallest of all <i>input</i> values	√	√	√	
<b>max</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	the largest of all <i>input</i> values	√	√	√	
<b>avg</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	the average of all <i>input</i> values	√	√		
<b>count</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	the number of <i>input</i> values	√	√	√	√
<b>range</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	difference between max and min	√	√		
<b>standard deviation</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	the dispersion or variation in a distribution of the given <i>input</i> values	√	√		
<b>least_frequent</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	the value which is repeated the fewest number of times			√	√
<b>most_frequent</b> ( <i>input</i> , { <i>conditions</i> }, <i>result</i> )	the value which is repeated the most number of times			√	√

**Figure 25 – Single-feature aggregations and the feature-types they are applicable to. The *conditions* need to be applicable to the entities in *input*.**

To do these tasks, this Chapter introduces a way of analyzing dependencies of features via multi-feature aggregation (Chapter 4.3.1). Also introduced into the ILP framework is a set of spatial features (Chapter 4.3.2) and literals (Chapter 4.3.3) which are extracted from the dataset for classification. Spatial techniques for aggregation are introduced into the ILP framework to allow for analysis that provides reliable results on spatial data (Chapter 4.3.4).

### 4.3.1 Multi-Feature Aggregation

Due to Tobler’s First Law Of Geography, dependencies between features of spatial entities must be considered. Analyzing and aggregating multiple features of the same entity-type simultaneously can yield valuable information in accordance with this law. For example, if house sizes increase as distance from a mall decrease, it could indicate that a mall is in a wealthy neighbourhood. There are multivariate functions, for example *correlation* [94], in other domains, such as statistics, which are able to express these dependencies, but these have not been explored in an ILP or classification context. Single-feature aggregation literals are adapted to multi-feature aggregation literals by allowing multiple features as arguments resulting in the following definitions.

**Definition 22 – Multi-Feature Aggregation Function** – A multi-feature aggregation function maps multiple lists of elements from the domains of the features to a single value possibly of another domain.

Literal	Returns	Applicable to cases where features are:		
		Numeric	Numeric or date	All ordinal or categorical
<b>linear_regression_coefficient</b> ( <i>input<sub>1</sub></i> , <i>input<sub>2</sub></i> { <i>conditions</i> }, <i>result</i> )	the slope of the line-of-best-fit, indicates relationship between two <i>inputs</i>	√	√	
<b>correlation</b> ( <i>input<sub>1</sub></i> , <i>input<sub>2</sub></i> { <i>conditions</i> }, <i>result</i> )	the degree to which the values are positively/negatively associated to each other	√	√ <sup>▽</sup>	
<b>covariance</b> ( <i>input<sub>1</sub></i> , <i>input<sub>2</sub></i> { <i>conditions</i> }, <i>result</i> )	the measure which indicates how two values vary together	√	√ <sup>▽</sup>	
<b>most_frequent_combination</b> ( <i>input<sub>1</sub></i> , <i>input<sub>2</sub></i> , ... <i>input<sub>n</sub></i> , { <i>conditions</i> }, <i>result</i> )	the value which is repeated the fewest number of times			√
<b>least_frequent_combination</b> ( <i>input<sub>1</sub></i> , <i>input<sub>2</sub></i> , ... <i>input<sub>n</sub></i> , { <i>conditions</i> }, <i>result</i> )	the value which is repeated the most number of times			√
<b>chi-square</b> ( <i>input<sub>1</sub></i> , <i>input<sub>2</sub></i> , ... <i>input<sub>n</sub></i> { <i>conditions</i> }, <i>result</i> )	indicates whether there is a dependency between two sets of categorical values			√

<sup>▽</sup> Treated as numbers by calculating the number of days from a certain date.

**Figure 26 – Proposed multi-feature literals and the *conditions* need to be applicable to the entities in *input*.**

**Definition 23 – Multi-Feature Aggregation Literal** – A multi-feature aggregation (MFA) literal has the form  $\text{Agg}(\{\text{input}_1, \text{input}_2, \dots, \text{input}_i\}, \{\text{conditions}\}, \text{result})$  where  $\{\text{input}_1, \text{input}_2, \dots, \text{input}_i\}$  specifies lists of corresponding feature values aggregated and constrained by conditions.  $\text{result}$  is the output variable of the MFA function corresponding to  $\text{Agg}$ .

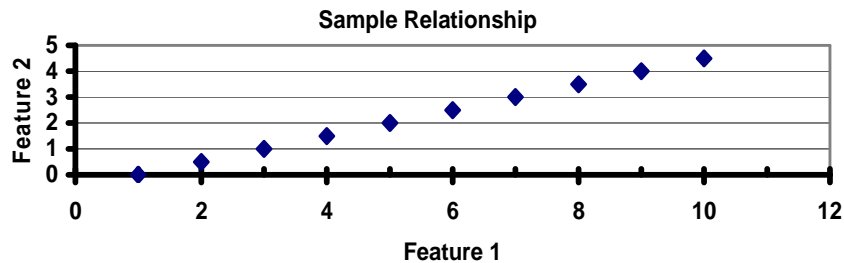
For example, instead of the single-feature aggregation (SFA) literal *average* (as in R5), R3 can be extended with the more powerful multi-feature aggregation (MFA) literal *correlation* to denote “a mall is profitable, if, for neighbouring houses, the correlation between household income and house size decreases” (i.e.: profitability of mall occurs if people have money but live in small houses). This is written as follows:

$$\text{R6: profit}(\text{M}, \text{'Yes'}) \leftarrow \text{mall}(\text{M}), \text{covariance}(\{\text{income}(\text{H}), \text{size}(\text{H})\}, \{\text{neighbour}(\text{M}, \text{H}), \text{house}(\text{H})\}, \text{C}), \text{C} < 0$$

A few common functions are used to measure relationships between features [94]. These same functions are also irreducible and complement each other in the properties of the relationship they measure. To capture these, this thesis introduces them to the ILP framework as Multi-Feature Aggregation literals (Figure 26). For example, using the data in Figure 27, these MFAs return the following values:

- **Linear regression coefficient** measures the relationship between the features and yields a value of 2.
- **Correlation coefficient** measures the departure of those features from independence and has value 1 (i.e.: they are dependent).
- **Covariance** measures how the features vary together and has value 4.125.

As opposed to regression, correlation, or covariance, which works on numeric values, the chi-square takes as input categorical values and measures their dependence. The ILP framework presented here generalizes to any type of multi-feature aggregation, non-linear correlations or



**Figure 27 – Sample relationship between two variables**

bivariate dependencies, for example. If the appropriate aggregation functions are added to the framework, the rule-learner will evaluate them and use them to construct rules. The literals presented in this thesis were specifically selected for their expressive power and it was found not to be necessary to add more complex aggregations.

With these additional aggregations the expressiveness of rules increases, but due to pair-wise features being evaluated for each MFA literal, the search-space increases significantly. As part of this thesis, the paper “A Method for Multi-Relational Classification Using Single and Multi-Feature Aggregation Functions” was published at the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD ‘2007) [38]. Experiments presented in that publication confirm that indeed the inclusion of MFA literals indeed improve classification precision and/or coverage. Runtime analysis is done in Chapter 5.4.

### 4.3.2 Spatial Features

Spatial data is unique from non-spatial data in that the entities can have associated to them a polygonal shape (e.g. the outline of a building or park). This polygonal shape implicitly contains important information, such as location or size, which must be extracted from the polygonal

Spatial Features	Description
<b>length</b> ( <i>input</i> , <i>L</i> )	the length <i>L</i> of the polygon representing entity <i>input</i>
<b>width</b> ( <i>input</i> , <i>W</i> )	the width <i>W</i> of the polygon representing entity <i>input</i>
<b>perimeter</b> ( <i>input</i> , <i>P</i> )	the perimeter <i>P</i> of the spatial polygon representing entity <i>input</i>
<b>area</b> ( <i>input</i> , <i>A</i> )	the area <i>A</i> of the spatial polygon representing entity <i>input</i>
<b>x_coord</b> ( <i>input</i> , <i>X</i> )	the X-coordinate <i>X</i> of the point entity
<b>y_coord</b> ( <i>input</i> , <i>Y</i> )	the Y-coordinate <i>Y</i> of the point entity

**Figure 28 – Existing spatial features expressing spatial properties of an entity**

Spatial Features	Description
<b>start_y</b> ( <i>input</i> , <i>Y</i> )	the starting Y-coordinate <i>Y</i> of the <i>input</i> entity
<b>end_x</b> ( <i>input</i> , <i>X</i> )	the ending X-coordinate <i>X</i> of the <i>input</i> entity
<b>end_y</b> ( <i>input</i> , <i>Y</i> )	the ending Y-coordinate <i>Y</i> of the <i>input</i> entity
<b>start_x</b> ( <i>input</i> , <i>X</i> )	the starting X-coordinate <i>X</i> of the <i>input</i> entity
<b>catchment_area</b> ( <i>input</i> , <i>C</i> )	the area <i>C</i> enclosed by the Voronoi cell of <i>input</i> , represents the region ‘attracted’ by the entity
<b>centroid_x</b> ( <i>input</i> , <i>X</i> )	the X-coordinate <i>X</i> of the centroid
<b>centroid_y</b> ( <i>input</i> , <i>Y</i> )	the Y-coordinate <i>Y</i> of the centroid

– represents the ends of road-segments, for example

**Figure 29 – Novel spatial features expressing spatial properties of an entity**

shape before it can be used for classification. For example, the area of a parking-lot might be relevant when searching for causes of theft from automobiles. Spatial features are used to describe these implicit features of entities:

**Definition 24 – Spatial Function** – *A spatial function is a function having an arity greater than 0 and returns the value of an implicit property derived from the input’s polygonal shape.*

**Definition 25 – Spatial Feature** – *A spatial feature is the result of a spatial function when applied to a variable. It describes some property of the input entity and has the form `SpatFeat(input, result)`. A constraint and a threshold could be applied to create literals.*

Some simple spatial features have been used in the literature [5, 87] (Figure 28). These however are not appropriate in describing entities such as road-segments or parks (the *perimeter* of ‘Highway 1’ is not useful). In order to increase the expressiveness of the language and measure all interesting properties of 2D entities, new features appropriate to 2D data are introduced (Figure 29). Since the proposed approach makes use of Voronoi calculations, an additional spatial feature, called *catchment\_area()*, is added to the proposed set. For a mall or store entity, the catchment area is defined as the area from which it attracts customers. The catchment area is given by the features of the Voronoi cell that the entity is in. The addition of this feature is motivated by research in marketing which use the catchment area in order to infer profitability [73]. To capture information like that in ILP rules, such a feature needs to be integrated into the rule language. Other features from other domains may be incorporated into this framework by following the format of spatial features.

### 4.3.3 Spatial Literals

Spatial data, by default, does not contain explicit relationships between entities. These relationships however need to be expressed by using a generic ILP construct. This thesis explicitly expresses spatial relationships with the use of *spatial literals*, as defined below:

**Definition 26 – Spatial Literal** – *A spatial literal represents a spatial relationship between two entities. It has the form `SpatLit(input1, input2)` or `SpatLit(input1, input2, S)`,  $S \in T$ , depending on whether an output variable is applicable.*

**Definition 27 – Neighbour Literal** – *A neighbour literal is a special spatial literal representing the existence of a relationship between two spatial entities, as defined by a neighbourhood definition. It has the form `neighbour(input1, input2)`.*

For example,  $house(H)$  and  $mall(M)$  can be connected with spatial literal  $neighbour(H, M)$ . The literal  $neighbour(H, M)$  was created because, according to the neighbourhood definition, such as the buffer-zone or Voronoi neighbourhood, there is a relationship between  $H$  and  $M$ .

There are three types of spatial literals: topological (8-intersection model), distance and direction [5, 28, 31, 32, 51, 75] (Figure 30). As these types of spatial literals are orthogonal, it is not possible to use any two to express all types of relationships. For example, without distance, it is possible to express that two entities are distinct and which relative direction, but not how close/far, they are relative to each other. Although distance and direction are needed only if the disjoint literal is true, in most cases all three types of literals are needed. In order to be complete, the set of topological relationships must be complemented by distance and direction. The approach presented in this thesis however uses Voronoi diagrams and the road network; hence very critical novel spatial literals need to be added to take advantage of these concepts (Figure 31). The literature sometimes makes use of literals such as  $north\_of()$ , but since the definition of *literal* used in this thesis encompasses a comparison operator and a threshold, the different

Literals	Description	
$disjoint(input_1, input_2)$	true if $input_1$ is completely separate from $input_2$	
$meet(input_1, input_2)$	if $input_1$ is strictly immediately adjacent to $input_2$	
$overlap(input_1, input_2)$	if $input_1$ extends over, or partially covers, $input_2$	
$covered\_by(input_1, input_2)$	$input_1$ completely envelops $input_2$	
$inside(input_1, input_2)$	if $input_1$ is within $input_2$ (a building in a city or park)	
$equal(input_1, input_2)$	if $input_1$ is identical and in the same location as $input_2$	
$covers(input_1, input_2)$	if $input_2$ completely envelops $input_1$	
$contains(input_1, input_2)$	if $input_2$ is within $input_1$	
$direction(input_1, input_2, D), D \theta T$	specifies angular position of $input_2$ relative to $input_1$ as a number or value such as 'East'	
$distance(input_1, input_2, D), D \theta T$	the distance between $input_1$ and $input_2$ when travelling in a straight line	

**Figure 30 – Existing spatial literals.  $\theta$  denotes a comparison operator and  $T$  a threshold.**

Literals	Description
$voronoi\_neighbour(input_1, input_2)$	if $input_1$ shares a relationship, according to the Delaunay triangulation with $input_2$
$road\_distance(input_1, input_2, D), D \theta T$	the distance between $input_1$ and $input_2$ when travelling along the road-network of a map
$travel\_time(input_1, input_2, D) D \theta T$	time required in order to cover the distance from $input_1$ to $input_2$ along the road network

**Figure 31 – Novel spatial literals.  $\theta$  denotes comparison operator,  $D$  the output, and  $T$  a threshold.**



directions can be captured using the literal *direction()*. For example instead of *north\_of(H, M)* the literals *direction(H, M, D)*, *D = 'North'* can be used.

#### 4.3.4 Spatial Aggregation Literals

Entities in spatial data are not independent and these dependencies must be captured through functions tailored to take into account this dependence. Non-spatial aggregations that are performed on spatial data might yield unreliable results, for example, regression analysis does not adjust for spatial dependency and thus can have unreliable parameter estimates and significance tests [67]. Spatial regression models allow for capturing these relationships and do not suffer from this weakness. Aside from *spatial\_trend*, which is only one of multiple spatial statistics methods, currently no work has been done in expressing these dependencies in ILP rules. Following the multi-feature aggregation format, this thesis expresses these spatial dependencies via *spatial aggregation literals* incorporated into the ILP language as follows:

**Definition 28 – Spatial Aggregation Literal** – A spatial aggregation (SA) literal *has the form*  $\text{Agg}(\{\text{input}_1, \text{input}_2, \dots, \text{input}_i\}, \{\text{conditions}\}, \text{result})$  *where*  $i > 0$  *and* *input* *specifies the lists of corresponding feature values aggregated and constrained by conditions. The variable result references the result of the aggregation function corresponding to Agg.*

As an example of the spatial aggregation literal, R7, *spatial\_trend* can be used to denote the rule “a mall is profitable if there exists a spatial trend between the mall and neighbouring houses such that the further the house from the mall, the larger the value of the house”:

R7: Profitable(M, 'Yes') ←

$\text{mall}(M), \text{spatial\_trend}(\{\text{distance}(M, H), \text{value}(H)\}, \{\text{house}(H), \text{neighbour}(H, M)\}, S), S > 0.$

The *result* is given in variable *S* onto which a restriction is placed such that it is greater than 0, to

Literals	Description
$\text{spatial\_trend}(\{\text{input}_1, \text{input}_2\} \{\text{conditions}\}, \text{result})$	indicates a tendency for a feature-value of $\text{input}_2$ to change with respect to distance from $\text{input}_1$
$\text{spatial\_autocorrelation}(\{\text{input}_1, \text{input}_2\} \{\text{conditions}\}, \text{result})$	analyze the degree of dependency among observations of $\text{input}_1$ and $\text{input}_2$ with respect to their location
$\text{area\_adjusted\_mean}(\{\text{input}_1, \text{input}_2\}, \{\text{conditions}\}, \text{result})$	the mean value of all $\text{input}_2$ entities, weighted by their Voronoi area, that are neighbours to $\text{input}_1$
$\text{v\_count}(\{\text{input}_1, \text{input}_2\}, \{\text{conditions}\}, \text{result})$	the count of all entities of $\text{input}_2$ in the Voronoi cell of $\text{input}_1$

**Figure 32 – Spatial Aggregation Literals, the *conditions* need to be applicable to the entities in *input*.**

indicate a positive relationship between the value of the house  $H$  and the distance between house  $H$  and mall  $M$ .

According to research in spatial statistics, common spatial occurrences are described by spatial *trends* and *autocorrelations* (also called ‘associations’) [63]. With the exception of *spatial\_trend* [31], spatial statistics have not been integrated into the ILP framework and have never been explored. This type of analysis however is critical in order to capture occurrences predicted by Tobler’s First Law. Hence this thesis incorporates the most common spatial statistics functions (Figure 32). Spatial trends describe the tendency of a feature-value to change consistently with an increase in distance away from a central entity whereas autocorrelation measures this change with respect to location in general. These functions are independent and describe different statistics of a set of spatial entities [63]. There are other statistics, such as the variogram, but these are very similar to the spatial aggregation literals in Figure 32 and hence are not included. The addition of these SA literals is expected to have similar impact, both in computational complexity and accuracy, as MFA literals. It is an open question how many and which underlying spatial influences can be captured by these literals, it might be possible that once the spatial processes are captured, other aggregations are not going to be confounded and thus become more important.

#### 4.4 Relationship between ILP and DB

Although classification results are expressed as ILP rules, the data the rules are derived from are stored in database  $D$  with schema  $S$  consisting of entities, tuples, and relationships. Hence literals are created from the following three database constructs: *entity literal*, *feature literal* and *relationship literal*:

- **Entity Literal.** Literals can be derived from *entity-types* in  $S$ . An ‘entity literal’ is a  $P/1$  literal  $t_i(v)$  where  $t_i$  denotes entity-type  $i \in S$  and  $v$  is a variable representing the feature which is the primary key of  $t_i$ . For example,  $house(H)$ .
- **Feature Literal.** Each entity-type  $t_i \in S$  contains *features* – properties of  $t_i$  that are stored in the database. A ‘feature literal’ is a  $P/2$  literal  $f(v, r)$  where  $f \in t_i$  denotes a feature such that  $f$  is neither a primary nor foreign key, and  $v$  is a variable representing the primary key of  $E$ , and  $r$  the resulting value. For example,  $size(H, r), r = 956$ .
- **Relationship Literal.** Relationships in  $S$  contain information about linked entities. A ‘relationship literal’ is associated with a pair of entity-types  $t_i \in S$  and  $t_j \in S$ , such that there is a foreign key relationship between  $t_i$  and  $t_j$ . This relationship can be represented

by a  $P/2$  literal  $r(i,j)$  where  $r$  denotes the relationship with  $i$  and  $j$  being the primary keys of  $t_i$  and  $t_j$  respectively.  $i$  could equal  $j$ .

Due to extensions for spatial and aggregation literals, more links are established:

- **Spatial Feature Literal.** Similar to a Feature Literal, except the feature is derived from the spatial representation of the entity.
- **Spatial Relationship Literal.** Similar to a Relationship Literal, except the feature is derived from the spatial properties of the relationship.
- **Aggregation Literal.** An *aggregation literal* represents the aggregation of a single or multiple feature values of entities from entity-type  $t_i \in S$ , all related to the same entity of a different type  $t_j \in S$ . They can be aggregated via single- (or multi-) feature aggregations. An aggregation literal is represented by a literal  $Agg(\{input_1, input_2, \dots, input_i\}, \{conditions\}, result)$  where all *input* parameters are features in  $t_i$  and *conditions* contains a feature literal referencing  $t_j$  along with a relationship literal referencing both  $t_i$  and  $t_j$ .

For example, consider:

R3: profit(M,'Yes')  $\leftarrow$  mall(M), neighbour(M,H), house(H)

the predicate *mall()* corresponds to a *mall* entity in the database. Since the literal *neighbour(M,H)* references the terms  $M$  and  $H$  which are applied to a predicate *neighbour()*, this implies that in the database there exists a *neighbour* table which references both the *mall* and *house* entity-types. All substitutions for  $M$  and  $H$  which are allowed for predicate *neighbour()* consist of the set of tuples in the database table *neighbour*. The above database and ILP terminology equivalencies are illustrated in Figure 33.

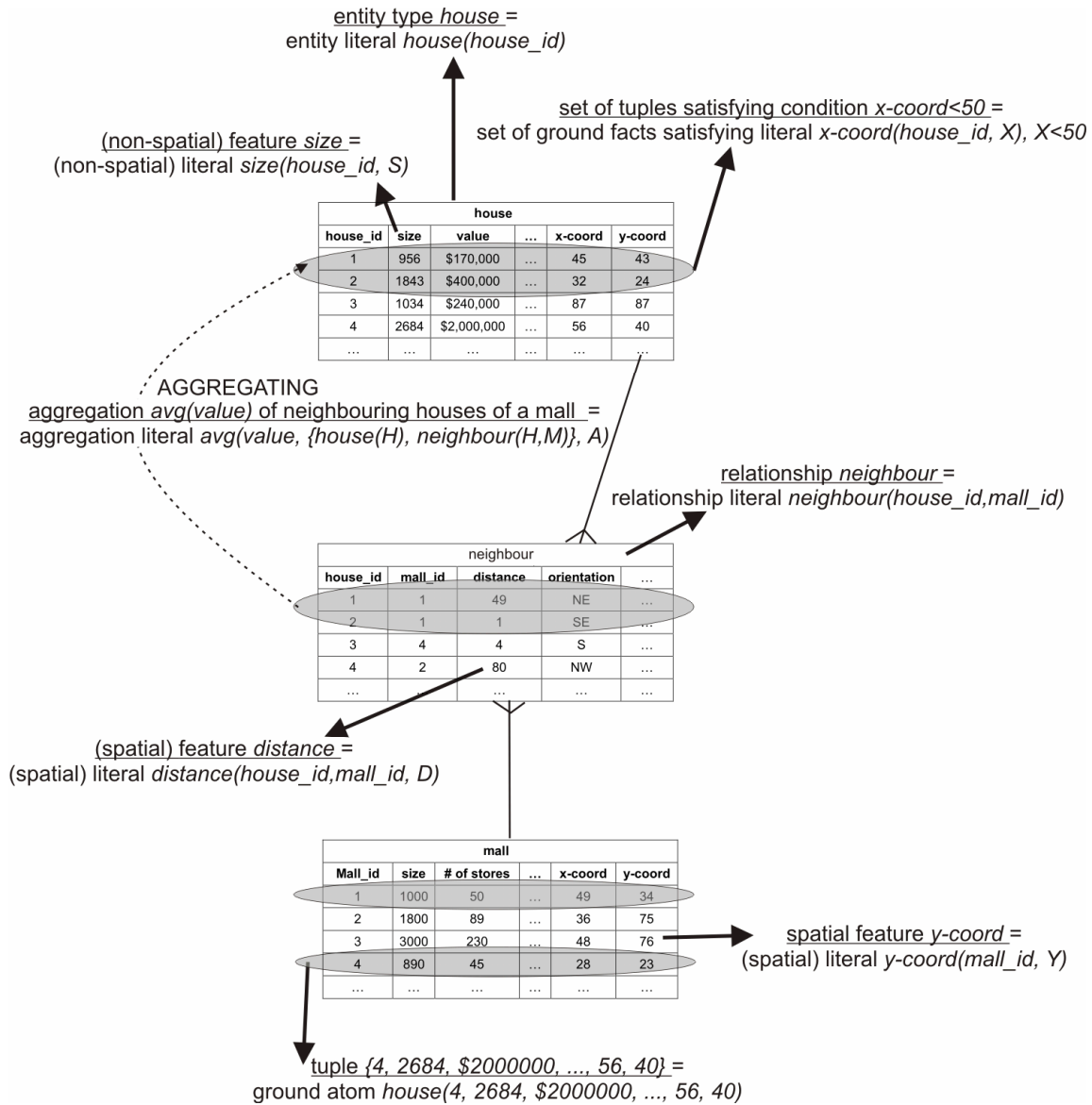


Figure 33 – Comparison of DB and ILP terminologies (underlined text is DB, non-underlined text is ILP terminology)

## CHAPTER FIVE: SPATIAL RULE LEARNING

This chapter explores the details of the method used to create spatial rules using a multi-relational foundation. Spatial data can be represented as multi-relational data, but it presents novel challenges not present in multi-relational problems which current multi-relational classifiers cannot solve. A novel, comprehensive spatial classification rule-learner, called *Unified Multi-relational Aggregation-based Spatial Classifier* (UnMASC), is introduced which addresses these challenges.

In order to learn classification rules on a multi-relational dataset, one of the entity-types from the dataset has to be selected by the user. This special entity-type is called the *target entity-type*, and is denoted as  $t_r$ . Each entity in the set of entities  $e^r$  of type  $t_r$  has exactly one class label assigned. The meaning of the class labels depends on the classification task. The task could be to either classify the entities into a discrete set of labels, or into a Boolean value. For example, the task could use a non-Boolean classifier to learn *how* profitable a *mall* is on a scale of 1 to 10, or use a Boolean classifier to simply classify whether it is *Profitable* or not

In general, the model can be built to describe all of the class labels simultaneously [68]. This thesis focuses on the two-class classification problem whereby the model is built to describe the class label selected by the user, referred to as the positive class, with the assumption that whichever entity is not described by the model does not belong to the positive class, but belongs to the negative class. The method presented in this thesis can however be generalized to work in a multi-class classification scenario. The goal of the classification task is to predict the class label for new entities not used to learn the model by looking at the feature values of related spatial and non-spatial entities, along with the relationships between them. Positively labelled entities refer to entities with the same label as the one the user selected, while negative entities refer to the non-positive target entities.

To learn the model, a rule-learner, based on the sequential covering algorithm [68], is applied. First, the spatial features are extracted and neighbourhoods established. Then the algorithm learns the classification model by generating one rule at a time and refining them incrementally by adding literals to the rule until some termination condition applies. The refinement is guided by

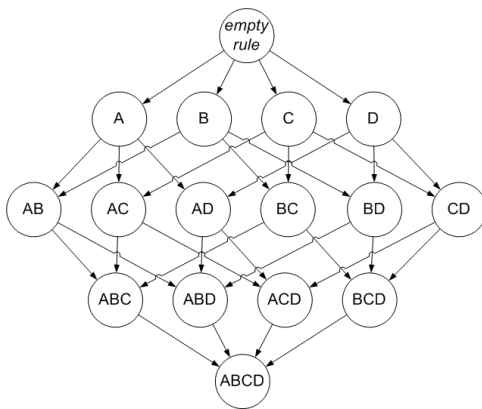
statistical measure so the rule quality improves during each refinement. The result is a set of rules which cover (ideally all) positively labelled entities in the dataset while covering a minimum number of negative ones.

There are multiple ways to construct the classification rules (Chapter 5.1), but in general they are guided by a series of refinements using a few standard measures (Chapter 5.1.3). However, this is a complex task (Chapter 5.4). Additional opportunities to prune the search space are still present (Chapter 5.5), which UnMASC considers. When the model is complete, statistics can be collected to measure the quality of the model (Chapter 5.6).

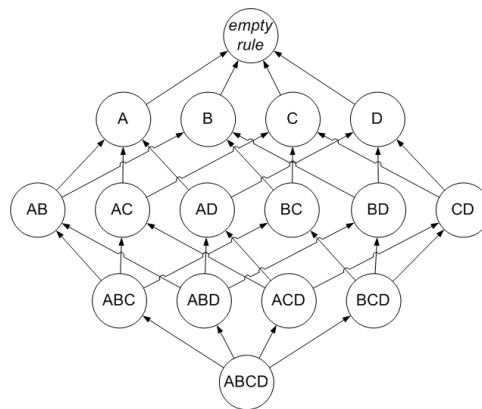
## 5.1 Rule Learning Strategies

Rule learners create models consisting of a set of rules. Each rule contains multiple conditions and describes a subset of entities which ideally contain the same class label. In order to find this rule, the set of all possible conditions that can be placed on the rule, called the rule-space, needs to be searched [88, 76]. This rule-space can be illustrated as a lattice, where each node is a condition (literal) that can be added to the rule. A small lattice, denoting the rule-space for conditions (literals)  $A$ ,  $B$ ,  $C$  and  $D$ , is shown in Figure 34.

There is a combination of conditions, corresponding to a specific node in the lattice, which is optimal according to some quality measure (discussed in Chapter 5.1.3); the task is to find this node. Rule learners need to search the rule-space in order to find the optimal combination of



**Figure 34** – Lattice showing a small *rule-space* being explored *top-down*



**Figure 35** – Lattice showing a small *rule-space* being explored *bottom-up*

conditions. There are two basic strategies to how rules are constructed:

- a strategy called *top-down* search (Chapter 5.1.1) starts at the *{empty rule}* and traverses the lattice down by adding conditions (Figure 34), and
- a strategy called *bottom-up* search (Chapter 5.1.2) starts at the most specific set of conditions and generalizes the rule by removing conditions (Figure 35).

In order to not evaluate each specific combination of conditions, a greedy strategy is typically used where only the best combination of conditions is evaluated further. For example, in the *top-down* strategy, if *{AD}* is found to have the best quality, only supersets of *{AD}*, that is *{ABD}* and *{ACD}*, will be evaluated further. Similarly, in the *bottom-up* strategy only *{A}* and *{D}* would be evaluated. Since a decision is made at each level, this strategy of evaluation will only evaluate a small subset of the possible set of conditions. UnMASC uses a *top-down* strategy, as is typical for ILP rule-learners.

### 5.1.1 Top-Down Rule Construction

A top-down classification method, such as CrossMine [100], starts with an empty rule that covers all entities not covered by any other previous rule. Then new conditions, in the form of literals, are added to narrow the subset of entities satisfying the conditions of the rule, but only if the additional condition meets two criteria. First, the rule quality must improve according to some quality measurement (see Chapters 5.1.3). Second, the number of entities covered by the rule must stay above a minimum threshold, said to be *supported*. This is repeated until no further conditions can be added, at which point the current rule is considered complete and added to the set of other rules. When a new rule is added to the existing set of rules, the set of corresponding entities satisfying that rule are removed from the dataset, and a new blank rule is started.

Measures which decide whether a rule should be refined further depend on the support value. The support of a rule is the number of possible substitutions for the variables in the rule. As an example, using the rule from a previous chapter:

R3: profit(M,'Yes') ← mall(M), neighbour(M,H), house(H).

One possible substitution is the replacement of the entity *M* in *mall(M)* by the value '*Lougheed Mall*' and entity *H* in *house(H)* is replaced by '*45 Main St.*'. If the rule is supported, it is changed by either applying a substitution to one of the literals or adding a new literal to the rule. R3, by using substitution, could be refined into:

R8: profit(M,'Yes')  $\leftarrow$  mall('Lougheed Mall'), neighbour(M,H), house(H)

or alternatively the algorithm could add a literal to the body of the clause generating the rule:

R9: profit(M,'Yes')  $\leftarrow$  mall(M), neighbour(M,H), house(H), size(H, S),  $S > 1,000$ .

Once the substitution or addition is done, the new rule is tested for consistency and support. If the new clause is found to be inconsistent with, or unsupported in, the dataset, then it is discarded. This is equivalent to the apriority pruning strategy where supersets of the current unsupported/inconsistent rule are pruned from the search-space and any further rules including those literals are not refined [26]. User-imposed constraints, representing the preferences of the user, on the head or body of the pattern can also influence the rule learning process [9]. One such constraint could be that the head/body must (not) include a specific literal.

### 5.1.2 Bottom-Up Rule Construction

A bottom-up approach covers all the positive examples by starting with very specific rules, possibly one per positive example, and generalizes them so that the set of rules is compacted but the coverage and accuracy are maintained. The idea is that the algorithm constructs a rule that is going to make other rules redundant, the more rules it makes redundant, the more compact the rule-set is, the more likely that that rule is going to be used in the rule-set. If the new generalized rule covers negative entities (i.e.: the rule is too general) then an attempt is made to specialize it so that no negative examples are covered. The end result is a compact set of rules which covers a minimum number of negative entities. Compactness can be measured by a metric which simply measures the size of the rule-set. [105]

### 5.1.3 Rule Quality Metrics

For the top-down rule construction method, the benefit of adding a specific literal to the rule must be measured. Two metrics are commonly used, FOIL-Gain and Information-Gain, but the two are very similar in their application. [100, 36]

#### FOIL-Gain

FOIL-Gain is a metric that is used to capture the total number of bits saved by the addition of an extra literal  $p$  to the rule  $r$  under consideration. The FOIL-gain for a literal  $p$ , given rule  $r$ , is defined as [100]:

$$\text{FOIL-Gain}_p = P_{r+p} \times (I_r - I_{r+p}) \quad (5.1)$$



and

$$I_r = -\log \frac{P_r}{P_r + N_r} \quad (5.2)$$

where the variables  $P_r$  and  $N_r$  denote the number of positive and negative target-entities satisfying rule  $r$ . When considering appending literal  $p$  to rule  $r$ , in order to create a new rule  $r+p$ , the number of positive and negative target-entities still satisfying rule  $r+p$  are denoted as  $P_{r+p}$  and  $N_{r+p}$  respectively.

### Information Gain

For decision trees, where there are  $k$  splits at each node in the tree, the main criterion is usually the information gained by adding literal  $A$  to the rule [36]. Information gain is defined as follows:

$$info\_gain(A) = entropy(P_r, N_r) - \sum_{i=1}^k \frac{P_{ri} + N_{ri}}{P_r + N_r} \times entropy(P_{ri}, N_{ri}) \quad (5.3)$$

where

$$entropy(P_r, N_r) = - \left( \frac{P_r}{P_r + N_r} \log \frac{P_r}{P_r + N_r} + \frac{N_r}{P_r + N_r} \log \frac{N_r}{P_r + N_r} \right) \quad (5.4)$$

and where  $N_r$  and  $P_r$  are the number of entities having a negative and positive label, respectively, on rule  $r$ . The literal with the highest possible information gain is added to the rule.

### Information Gain vs. FOIL-Gain

The information gain measure is applied to a decision tree where each node can be split  $k$  ways (classes), and each split has to be as pure as possible. This is not the case with the two-class classification problem dealt with in this thesis, where, instead of  $k$  classes there are only 2, and instead of considering the purity of each class only the purity of the positive class is measured. Given these two restrictions, the information gain simplifies into the FOIL gain measure as follows:

$$info\_gain(A) = entropy(P_r, N_r) - \sum_{i=1}^k \frac{P_{ri} + N_{ri}}{P_r + N_r} \times entropy(P_{ri}, N_{ri}) \quad (5.5)$$

Since the concern is only on the positive class when adding literal  $p$  to rule  $r$ , hence  $k = 1$  and equation (5.5) simplifies into

$$info\_gain(A) = entropy(P_r, N_r) - \frac{P_{rp} + N_{rp}}{P_r + N_r} \times entropy(P_{rp}, N_{rp}) \quad (5.6)$$

$$= - \left( \frac{P_r}{P_r + N_r} \log \frac{P_r}{P_r + N_r} + \frac{N_r}{P_r + N_r} \log \frac{N_r}{P_r + N_r} \right) + \frac{P_{rp} + N_{rp}}{P_r + N_r} \times \left( \frac{P_{rp}}{P_{rp} + N_{rp}} \log \frac{P_{rp}}{P_{rp} + N_{rp}} + \frac{N_{rp}}{P_{rp} + N_{rp}} \log \frac{N_{rp}}{P_{rp} + N_{rp}} \right) \quad (5.7)$$

$$= - \left( \frac{P_r}{P_r + N_r} \log \frac{P_r}{P_r + N_r} + \frac{N_r}{P_r + N_r} \log \frac{N_r}{P_r + N_r} \right) + \left( \frac{P_{rp}}{P_r + N_r} \log \frac{P_{rp}}{P_{rp} + N_{rp}} + \frac{N_{rp}}{P_r + N_r} \log \frac{N_{rp}}{P_{rp} + N_{rp}} \right) \quad (5.8)$$

Given that the type of classification UnMASC performs does not consider the accuracy of the negative class, those terms can be removed from the equation.

$$= - \left( \frac{P_r}{P_r + N_r} \log \frac{P_r}{P_r + N_r} \right) + \left( \frac{P_{rp}}{P_r + N_r} \log \frac{P_{rp}}{P_{rp} + N_{rp}} \right) \quad (5.9)$$

Which is equivalent to (5.1). Hence the FOIL-Gain is used for UnMASC.

#### 5.1.4 Dealing with Local Optima

As is standard in rule learning, the algorithm is greedy since at each step it makes the best decision at that step. It does so without a global view of the rule-learning. As such, it is capable of making locally optimal, yet globally sub-optimal, decisions in which literal to select. Since in a multi-relational setting the search-space is much larger, this weakness could be even worse. Multiple methods for dealing with this global sub-optimality exist, such as beam search [35], simulated annealing [86], or randomized restarts [104]. While these methods have been shown to improve the classification process, the focus in this thesis is on the introduction of multi-feature and spatial literals into the ILP framework, and hence these methods will not be used.

## 5.2 Rule Learning

The classification process starts with an initial rule which references the target entity-type  $t_r$ , *malls* for example. When evaluating refinements to the rule, all entity-types which are related to entities in the rule via a foreign key relationship need to be evaluated (at this point spatial relationships are already explicitly stated and hence have foreign-key relationships established). Different entity-type combinations (called *neigEntity*) need to be considered, depending on what

the rule already contains. The set of all *neigEntity*'s is called *neigEntitySet*. For example, *malls* have as neighbours other *malls*, *houses* and *roads*, hence *neigEntitySet* would contain the *neigEntity*'s  $\{malls\}$ ,  $\{malls, malls\}$ ,  $\{malls, houses\}$  and  $\{malls, roads\}$ . Had the rule already referenced *houses* as well, neighbours of *houses* would also require exploration, thus the *neigEntity*'s  $\{malls, houses, malls\}$ ,  $\{malls, houses, houses\}$ ,  $\{malls, houses, roads\}$  would also need to be added to *neigEntitySet*. The more entity-types the rule references, the more *neigEntity*'s *neigEntitySet* contains.

For each *neigEntity*, the goal is to search that specific set of neighbouring entities for the candidate literal with the highest FOIL-Gain, then return the feature, aggregation, constant value and comparison operator yielding this high FOIL-Gain. The decision of which literal to add to the rule is made only after the FOIL-Gain has been evaluated for all candidate literals for all *neigEntity*'s. If the entity-type in the literal has not been referenced in the rule, then up to three literals are added: an *Entity Literal* referencing  $t_i$ , a *Neighbour Literal* denoting the relationship of  $t_i$  to an entity-type already in the rule, and possibly a *Feature Literal* or *Aggregation Literal* with a condition on a (some) feature(s) from  $t_i$ . For example, if the classification process starts with rule:

$$R10: \text{profit}(M, 'Yes') \leftarrow \text{mall}(M)$$

This could be refined to

$$R11: \text{profit}(M, 'Yes') \leftarrow \text{mall}(M), \text{neighbour}(M, H), \text{house}(H)$$

by adding *house(H)* as an *Entity Literal*, *neighbour(M,H)* as a *Neighbour Literal*. Alternatively, a third literal (a *Feature Literal*) can also be added, for example *size(H, S), S > 1,000*, to create the rule

$$R12: \text{profit}(M, 'Yes') \leftarrow \text{mall}(M), \text{neighbour}(M, H), \text{house}(H), \text{size}(H, S), S > 1,000.$$

For non-empty rules, the search is more complicated since any entity-type with a relationship to another entity-type referenced by the rule must be evaluated. The more entity-types referenced, the more relationships there are, and the larger the *neigEntitySet* is. Note that depending on the neighbourhood definition and dataset, entities of type  $t_i$  could be neighbours to other entities of type  $t_i$ .

For each *neigEntity*, the relevant neighbourhood relationships are exploited, and the features for the required entities are retrieved from the database. These entities, their neighbourhood relationships and features, are then passed to the spatial rule learner which calculates the

aggregations and spatial features. The rule-learner then evaluates all features, keeps track of a set of candidate literals, and when the search is complete, adds the best candidate literal to the rule. This is repeated until the rule is finalized, at which point the entities which match the conditions in the rule are removed from the dataset, and another rule is started. If no further rules can be started, then the classification model is deemed to be complete.

### 5.3 Candidate Literal Search in Parallel

In order to determine the best literal to refine the rule with, numerous candidate literals (both spatial and non-spatial) need to be considered. In most state-of-the-art algorithms, such as CrossMine [100], the consideration of the candidate literals takes place one after another. Once all have been considered, the best is selected. This is highly inefficient given that current computers usually have multiple CPUs and that in most cases the candidate literals have to be evaluated independently from each other.

One technique of decreasing the time required to evaluate all literals is with the use of a query-pack, a set of ordered and related queries. If each database-query has similar internal logic, this approach relies on the DB or customer application to cache the search results, such that when a similar query is executed, part (or all) of the computation does not have to be redone [12]. This could have a benefit when exploring extensions to ILP rules, since the main body of the rule is generally the same, but with the last literal either modified or changed. Referring to Figure 34 above, in order to evaluate rule ‘ABC’ and ‘ABD’, the database has to evaluate ‘AB’, which it has to do twice in a normal scenario, but can do a single time with query-packs. This implementation speed-up technique however has been evaluated by modifying existing rule-learners such as TILDE [13] and WARMR [25], with a net speed-up factor between 1.5 and 5.7.

Another approach to decreasing the time required for candidate literal evaluation is to evaluate all literals in a parallel and independent fashion. Since the decision of which literal to add to the rule is made only after the FOIL-Gain has been evaluated for all possible rule extensions within all *neigEntity*'s, the evaluation of multiple rule extensions can be performed independently and in parallel. This would considerably speed up the search for the best literal to refine the rule with. The benefits of this would be the speed up of the set of candidates, at the expense of needing to implement a parallel architecture for evaluating the rules. Due to preliminary experimental testing, in the setup used for the thesis it was found that the database retrieval was a relatively small portion of the runtime, and the majority of the runtime was spent on performing the applicable aggregations, which would significantly change depending on what candidate

extension was being evaluated, hence a totally independent and parallel architecture would make a more significant difference in run-time than query-packs. Chapter 6.2 illustrates one way of implementing parallelization, by arranging the system into a server-client architecture. This allows for simultaneous evaluations of multiple rule extensions.

## 5.4 Complexity Analysis

In this section, the worst case runtime of the UnMASC algorithm is analyzed. Let  $Q_\tau$  be the number of target entities  $e^\tau$  around which rules are built and  $\beta$  the average number of Voronoi neighbours for each entity  $i$  in  $e^\tau$ , denoted  $e^{\tau,i}$ . Assume that  $f_c, f_d, f_n$  and  $f_s$  respectively represent the number of *categorical*, *date*, *numeric* and *spatial* features of an entity. Let  $|f| = f_c + f_d + f_n + f_s$ . As described in Chapter 4.4, there are 7 types of literals, of which one already exists in the database, three are extracted beforehand as part of pre-processing (Chapter 5.4.1), and the three aggregation literals are done on-the-fly (Chapter 5.4.2).

### 5.4.1 Features from Pre-Processing

**Spatial features** require extraction from the entity before they can be used in the rule-learner. From Figure 28 and Figure 29, out of the 13 spatial features, 12 can be extracted via composite spatial SQL functions of the DB2 Spatial Extender [49]. For example, the literal *start\_y(input, result)* is equivalent to the composite DB2 spatial function *ST\_Y(ST\_StartPoint(input))*. For these 12 features, the runtime for each spatial feature extracted is  $O(k Q_\tau \beta)$  where  $k$  represents the time required for the extraction by the database. The last spatial feature, *catchment\_area()*, cannot be expressed as a composite SQL function with DB2 Spatial Extender, since it requires Voronoi diagram calculations, which have complexity  $O(Q_\tau \log Q_\tau)$  [47]. Thus, to calculate all 13 spatial features, a total complexity of  $O(Q_\tau \log Q_\tau + 12k Q_\tau \beta)$  is required.

**Spatial Literals** and **Neighbour Literals** analyze the link between a target entity and its neighbours. There are 13 spatial literals (Figure 30 and Figure 31) and 1 neighbour literal, and  $Q_\tau$  target entities each with  $\beta$  neighbours. Hence total complexity is  $O(14 Q_\tau \beta)$ .

Thus the total pre-processing complexity is  $O(Q_\tau \log Q_\tau + 12k Q_\tau \beta + 14 Q_\tau \beta)$ .

### 5.4.2 Features from Run-Time

For each entity in  $e^r$ , **Single-Feature Aggregation Literals** (Figure 25) aggregate the values of all  $\beta$  neighbouring entities, each having  $|f|$  features. Hence the computation of SFA literals requires a runtime of  $O(\beta Q_\tau |f|)$ .

For **Multi-Feature Aggregation Literals** (Figure 26), 3 literals are evaluated by choosing 2 out of  $f_c$  categorical features and another 3 by choosing 2 out of  $(f_d + f_n + f_s)$  features yielding  $O\left(\beta Q_\tau \times (3C_2^{f_c} + 3C_2^{f_d+f_n+f_s})\right) = O(\beta Q_\tau |f|^2)$ .

Complexity for **Spatial Aggregation Literals** (Figure 32) are as follows: *spatial\_trend()* expresses relationships between distance and a feature value, hence is linear in  $|f|$  leading to  $O(\beta Q_\tau |f|)$ . *area\_adjusted\_mean()* is also  $O(\beta Q_\tau |f|)$  since it requires the values of a single feature. *spatial\_autocorrelation()* works with pair-wise features, thus is of  $O(\beta Q_\tau |f|^2)$ . *v\_count()* does not analyze features, but entities, hence is  $O(\beta Q_\tau)$ .

Thus, the total run-time complexity required is  $O(\beta Q_\tau |f| + \beta Q_\tau |f|^2 + \beta Q_\tau) = O(\beta Q_\tau |f|^2)$ .

### 5.4.3 Single Literal Search Complexity

For each literal, the runtime complexity, calculated above, is given by:

$$\begin{aligned}
 &= O(\text{features extracted during pre-processing} + \text{features extracted during run-time}) \\
 &= O(Q_\tau \log Q_\tau + 12k Q_\tau \beta + 14 Q_\tau \beta) + O(\beta Q_\tau |f|^2) \\
 &= O(Q_\tau \log Q_\tau + 12k Q_\tau \beta + 14 Q_\tau \beta + \beta Q_\tau |f|^2) \\
 &= O(12k Q_\tau \beta + \beta Q_\tau |f|^2) \\
 &= O(\beta Q_\tau |f|^2) \quad \text{assuming } 12k \ll |f|^2.
 \end{aligned}$$

Thus, per each literal, the runtime depends on the number of features and neighbors  $O(\beta Q_\tau |f|^2)$ .

### 5.4.4 Rule-Learning Complexity

Let  $l$  represent the number of literals per rule learnt during the rule-building process. Let  $s$ , where the domain of  $s$  is  $(0,1)$ , represent the user-specified minimum support. The overall worst-case complexity of the rule-learning is thus:

$$= \text{Number of rules} \times \text{number of literals per rule} \times \text{complexity of single literal search}$$

$$= 1/s \times l \times O(\beta Q_r |f|^2)$$

$$= O(\beta Q_r |f|^2)$$

assuming  $O(s) = O(l)$ , i.e. the number of rules is approximately the same as the number of literals per rule

Thus, the limiting factor in the rule-learning process is the evaluation of the multi-feature and spatial aggregations which introduce the  $|f|^2$  term into the equation and dominate the run-time.

## 5.5 Pruning

In order to improve the efficiency of the spatial rule-learning, UnMASC makes use of a few optimizations.

### 5.5.1 Pruning Threshold Values

The goal of adding literals to the rule is to make the entities that the rule covers purer, that is, to have all the entities be of the same class with as few exceptions as possible. In order to do this, a threshold value needs to be found which best splits the dataset into two partitions, with one of the partitions containing a set of entities having a purer set of class-labels than the original set. For example, the best threshold for the dataset in Figure 36 would be *423000* because the set of entities which have an average value less than *423000* have purer class-labels (3 out of 4 positive labels whereas before 3 out of 6 entities had positive labels).

When searching for the best literal and threshold value combination, each possible threshold value can be evaluated in order to find the best one. If there are  $Q$  entities, then there are up to  $Q$  distinct threshold values which must be considered. For example, when evaluating the *average value* feature in Figure 36 (an extended aggregated version of the data shown in Figure 5 – page 4), there are six feature values which need to be considered as threshold values:  $\{70000, 150000, 232000, 340000, 423000, 500000\}$ . The naïve rule-learner would evaluate all possible thresholds individually to find the one with the highest FOIL-Gain.

Although FOIL-Gain can be evaluated quickly for each individual threshold, it is not efficient to evaluate **all** thresholds in a large dataset. Assume that a candidate literal is being evaluated, and a feature is being searched for the best threshold, i.e. highest FOIL-Gain value. The FOIL-Gain formula requires that the number of positive and negative entities satisfying the literal be known. This is done most efficiently if the entities are sorted by increasing (or decreasing) feature values, at which point the list of thresholds can be scanned and a running count of positive and negative labels can incrementally be updated for each entity (Figure 37). With these running counts, the

FOIL-Gain can quickly be calculated for any threshold. It is however enough to evaluate the FOIL-Gain only for the entities where there is a change in class-labels with respect to the previous entity. This is because any entities that follow an entity of the same class-label will further improve (or reduce) the purity of the class, but will not alter the direction of the change. It does not make sense to consider a threshold value, when it is known that the next  $n$  entities have the same class-label and hence each will further refine/reduce the class-label purity. It is sufficient to evaluate only the entities where the class-label changes (with respect to the previous entity) since those are the entities that represent the local maxima/minima in class-label purity. Given this, instead of the six evaluations in Figure 37, only three need to be evaluated ( $\{70000, 150000, 423000\}$ ). UnMASC performs this pruning to considerably reduce the number of threshold evaluations required.

### 5.5.2 Pruning within the Existential Operator

Literals can include a comparison operator and a threshold value  $v$ . Assume the neighbourhood of entity  $e^{t_i}$  is being evaluated. When applying the existential operator to the feature values of the neighbourhood, the possible threshold values that need to be evaluated is the set of all unique feature values of the neighbours of  $e^{t_i}$ . The computation of the existential operator can be expensive for numerical data due to the possibility that all the values can be distinct, and each numerical value would have to be considered as a possible threshold. This is expensive because then entity counts (of the number of entities having higher and lower feature values) would need to be prepared, the FOIL-Gain calculated, etc.

There are three possible comparison operators:  $=$ ,  $>$  or  $<$ . For numerical data, if the literal contains the  $=$  comparison operator, then the resulting literal is too specific and not interesting

<i>People-House aggregated</i>						
ID	First Name	Last Name	Age	Average Value	Total Value	Label
1	Mark	Doe	34	70000	70000	-
2	John	Smith	45	150000	300000	+
3	Betty	Smith	39	232000	232000	+
4	Fred	Flint	54	340000	340000	+
5	Brian	Lam	70	423000	600000	-
6	Jason	Roth	25	500000	780000	-

Figure 36 – Aggregated table (Sorted by Average Value)

<i>People-House aggregated</i>			
Average Value	Label	+ve	-ve
70000	-	0	1
150000	+	1	1
232000	+	2	1
340000	+	3	1
423000	-	3	2
500000	-	3	3

Figure 37 – Class label count



(ex: the rule "the mall is profitable if a bank is exactly 4.5298km away"). However, if, for numerical data, the = comparison operator is not evaluated, then significant efficiency gains can be realized.

Ignoring the = comparison operator, a literal can be created with two comparison operators, > and <. Assume a literal  $L$  is selected with comparison operator > and threshold value  $v$ . An entity is covered by  $L$  if any of its neighbors have a feature-value greater than  $v$ . Let's say  $e^{i,q}$  has  $\beta$  neighbors with feature values  $\{f_1, f_2, \dots, f_j, \dots, f_\beta\}$  and is covered because  $f_j > v$ . The critical observation is that: if  $f_j > v$ , then  $\text{MAX}_{j=1}^{\beta} f_j > v$ , that is, if there is a neighbour of  $e^{i,q}$  with feature-value larger than  $v$ , then the largest feature-value of any neighbour of  $e^{i,q}$  must also be larger than  $v$ . This implies that, instead of evaluating  $\beta$  thresholds, it is only necessary to evaluate **one**. Thus, it holds that for the > operator, only the maximum neighbouring feature-value needs be considered against  $v$  in order to determine coverage. The converse is true for <, where only the minimum neighbouring feature-value needs to be considered against  $v$ .

The naïve computation requires the analysis of all  $Q_i$  entities, each with their  $\beta$  neighbours' feature-values, leading to a runtime of  $O(Q_i \beta)$ . Since the Existential operator can be optimized, and by ordering the evaluation of the aggregation functions such that *min* and *max* are evaluated before the existential operator, the runtime can be reduced to  $O(Q_i)$ . UnMASC makes use of this pruning strategy.

### 5.5.3 Pruning Aggregation Functions

All the aggregation functions used in this thesis were selected because they are representative of the common statistics measures that are used, while being relatively independent from each other. The exception to this is the interrelationship between the *sum*, *count* and *average* aggregation functions, but all have important meanings and hence all three are included. While these three aggregation functions have a dependency between them, the rest are independent and knowledge of one does not help in evaluating another, hence the set of aggregation functions cannot be decreased further without losing expressiveness. If there are a small set of basic aggregation functions, such as those used in this thesis, then pruning this set of aggregation functions does not seem profitable.

Single-feature aggregations allow a few pruning strategies. An efficient strategy for calculating some of the SFAs can start by evaluating *sum* and *count* leading immediately to *avg* values. If a literal such as  $\text{avg}(>v)$  can be found then it will immediately prune literals such as  $\text{min}(>v)$  or

$max() < v$  since they possibly cannot be true. Hence two of the aggregations do not have to be evaluated.

The multi-feature and spatial aggregations presented in this thesis analyze pair-wise subsets of features, hence the complexity is exponential in the number of features, i.e.:  $O(f^2)$ . Both sets of aggregation operators were selected to not include any redundant aggregation operators; hence the sets cannot be decreased further.

### 5.5.4 Pruning Spatial Features and Literals

Spatial features also do not present pruning possibilities. *centroid\_x*, for example, has no bearing on *area* or *start\_x*. However, there are some pruning possibilities for spatial literals, but these must be done on-the-fly. If the literal *disjoint(A,B)* is evaluated to *true*, meaning that entities *A* and *B* do not share a common area and their edges do not touch, then the rest of the topological relationships from the 8-intersection model [75] will automatically evaluate to *false*. The other spatial relationships (*distance*, *direction*, *road\_distance* and *travel\_time*) will however require evaluation. Conversely, if *disjoint(A,B)* is *false*, then *distance*, *direction*, *road\_distance* and *travel\_time* do not require evaluation, since the edges of *A* and *B* intersect in at least one location,

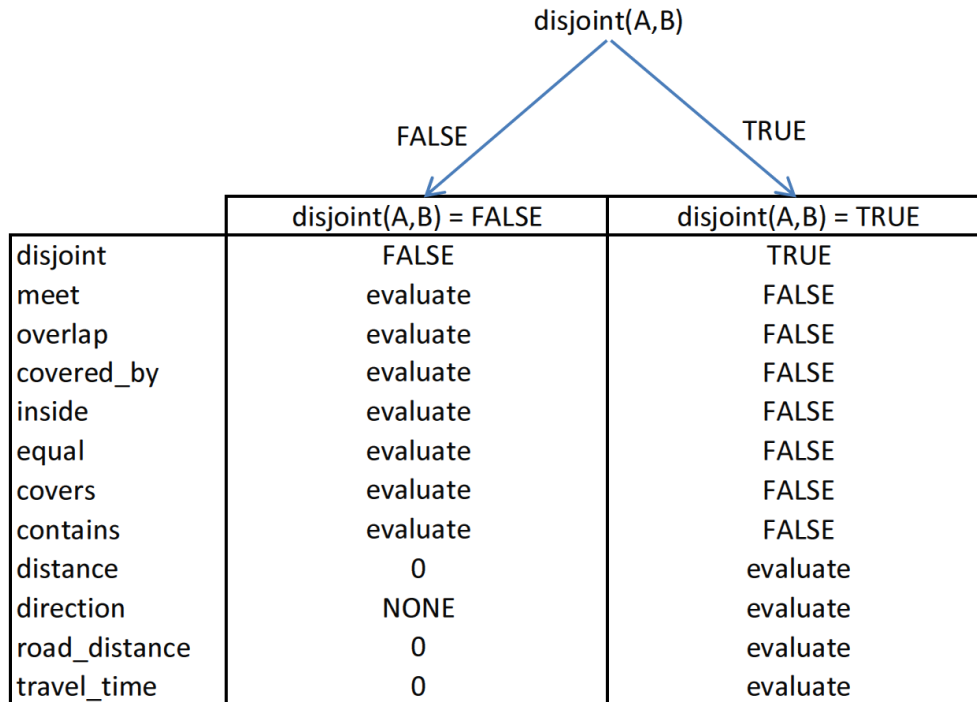


Figure 38 – Pruning of spatial literals

meaning the distance between A and B is 0 (note: neighbourhood for 2D entities can be defined in other ways). The other topological relationships can be evaluated one-by-one until one is found to be *true*, after which the rest of the topological relationships do not need to be evaluated since they are mutually exclusive and only one can be *true*. A summary of the pruning techniques available to spatial literals is shown in Figure 38. Instead of all 13 spatial literals being evaluated, with pruning, the expected number of evaluations is decreased to 5.

### 5.5.5 Monotonicity Properties of Aggregation

The results of the classification process are rules which are only applicable to a subset of the entities from the dataset. Each condition in a rule further reduces the subset until a new set of conditions is started. Conditions can also be placed *within* other conditions, but this can introduce inconsistencies into the results. Take a rule, stating that “a mall is profitable if the count of houses in its neighbourhood is 2”:

R13: profit(M,'Yes') ← mall(M), count(H, {neighbour(M,H), house(H)}, C), C > 2

A further refinement can be placed onto R13 to continue the classification process. For example, to refine the count of houses to only those with “values over \$200,000”:

R14: profit(M,'Yes') ← mall(M),count(H, {neighbour(M,H), house(H),  
value(H)>200,000},C),C>2

The problem with this type of refinement is that, had the original example been

R15: profit(M,'Yes') ← mall(M), count(H, {neighbour(M,H), house(H)}, C), C < 2

and a similar restriction added onto it to yield:

R16: profit(M,'Yes')←mall(M),count(H, {neighbour(M,H), house(H), value(H)>200,000},C),C<2

The rule now would have been an invalid refinement. Notice that it now states that “a mall has *less* than 2 houses with values more than \$200,000 neighbouring it”, using the ‘<’ comparison operator and not ‘>’. The two rules use the same literals; R14 however represents a valid refinement whereas R16 does not. This is better illustrated with an example using a dataset of malls and the values of their neighbouring houses:

Mall	Value of Neighbouring Houses
1	\$500,000 \$140,000 \$150,000
2	\$150,000
3	\$650,000 \$430,000 \$390,000
4	\$620,000

The following sets satisfy the rules from above:

Rule	Malls Satisfying Rule
R13	{1, 3}
R14	{3}
R15	{2, 4}
R16	{1, 4}

When R13 is refined into R14 the result is a proper subset of the set satisfying R13. This is because the combination of *count* operator with the *greater-than* comparison operator has a *monotone property*. Monotonicity, in the context of ILP rules, is defined as:

**Definition 29 – Monotone** – Given a Horn Clause  $H$  of the form  $L_0 \leftarrow L_1, L_2, L_n$ , and a set  $E$  of entities which satisfy  $H$ , an aggregation operator and comparison operator are said to be monotone if adding any other literal  $L_i$  to  $H$  yields a Clause  $H'$ :  $L_0 \leftarrow L_1, L_2, L_n, L_i$  which is satisfied by entities  $E' \subseteq E$ .

For example, adding a restriction into  $count() > t$  is always going to yield a proper subset, while the  $count() < t$  might yield a proper subset, but is not guaranteed to. The *count* aggregation and the *less-than* comparison operators do not yield a monotone refinement and hence any rule using those operators cannot be refined by adding another condition *into* an already existing condition. There is no generic rule that is able to predict which combination of aggregation and comparison operators is going to satisfy this monotonicity requirement. Some, such as *average*, are not monotonic with either  $>$  or  $<$  comparison operators, whereas *count* is monotonic with the  $>$  comparison operator. Hence the only solution would seem to be to individually categorize whether an aggregation and comparison operator combination is monotonic or not. Since this restriction exists, care has to be taken when introducing conditions into aggregation literals.

## 5.6 Measuring the Quality of Rules

During rule construction, the quality of each candidate literal is measured using FOIL-Gain (Chapter 5.1.3). While the quality of each addition to the rule must be measured, the quality of the entire rule must also be measured. There are several ways in order to measure the quality of the final rules. These measures are: precision, recall and accuracy [41]. The following subchapters will refer to Figure 39.

### 5.6.1 Precision

Precision is a measure which represents the exactness of the rules. It can be thought of as the number of *true* positive labelled entities identified by the classifier out of the set of *all* entities identified as positive. The larger the variability in the results, the smaller the precision will be. It is given by this formula:

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (5.10)$$

High precision does not imply high accuracy, as results could always be the same (high precision), but inaccurate. A precision score of 1 means that all the results that were retrieved were truly positive, even if the number of missed positive entities is unknown.

### 5.6.2 Accuracy

In classification, this measure denotes how well a classifier identifies the true positives and true negatives out of the set of all entities. It is given by this formula:

$$\text{accuracy} = \frac{\text{true positive} + \text{true negative}}{\text{true positive} + \text{false positive} + \text{false negative} + \text{true negative}} \quad (5.11)$$

	Known Positives	Known Negatives
Positives according to test	True Positive	False Positive
Negatives according to test	False Negative	True Negative

Figure 39 – Binary classification possible outcomes

### 5.6.3 Recall

Recall is a measure of completeness. It measures the number of positive entities *identified* out of the set of *all* true positive entities.

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \quad (5.12)$$

A recall score of 1, perfect, means that all true positives were identified, but the number of false positives identified along with the true positives is not measured.

## **CHAPTER SIX: THE CRIME DATA-WAREHOUSE AND THE IMPLEMENTATION OF UNMASC**

This thesis introduces several novel techniques towards mining spatial data: a Voronoi-based neighbourhood definition, additions to the rule-language capable of capturing dependency between features and spatial analysis, and lastly, the parallelization of the rule-learning. This chapter discusses how these techniques are implemented in what is called the UnMASC Framework (Figure 57). The framework is composed of a spatial database system, called the Crime Data-Warehouse, and a data-mining algorithm called *Unified Multi-relational Aggregation-based Spatial Classifier* (UnMASC).

As a result of collaboration between the Institute of Canadian Urban Research Studies (ICURS) research center at the School of Criminology at SFU and the Royal Canadian Mounted Police (RCMP), five years of real-world crime data was made available for research purposes at ICURS. This data was retrieved from the RCMP's Police Information Retrieval System (PIRS). PIRS, integrated with publically available datasets such as the road-network, formed the core of the Crime Data-Warehouse (CDW).

UnMASC and the Crime Data-Warehouse were both developed inside the secure computing facilities at the ICURS research center. ICURS is based at the School of Criminology at SFU and directed by Dr. Patricia Brantingham (Professor of Computational Criminology) and Dr. Paul Brantingham (Professor of Crime Analysis). Their leading research into Environmental Criminology made them, and the ICURS research center, perfect collaboration partners for developing a spatial classifier which mines entities in an urban setting for explanations on how crime patterns change due to their surrounding environment.

The set of experiments carried out in this thesis required a different dataset to be used, each requiring three parameters to be specified: the city, type of crime and entity-type, denoted hereafter as  $\psi_{CITY}$ ,  $\psi_{CRIME}$  and  $\psi_{LABEL}$ , respectively. For example, if  $\psi_{CITY}=\text{Burnaby}$ ,  $\psi_{CRIME}=\text{burglary}$  and  $\psi_{LABEL}=\text{Commercial}$ , then this results in a dataset capable of classifying burglary crimes for commercial entities within Burnaby.

The selection of a specific city led to the imposing of artificial boundaries within the dataset since those cities do not live in isolation, but are affected by other surrounding cities and towns. Crime does not stop just because the other side of the street is another police jurisdiction. Due to this cut-off, neighbourhood relationships are also affected. Any neighbourhood relationship is now built on ‘empty space’ where there should be data from another city. This would cause, for example, the Voronoi diagrams to extend forever out of the dataset, whereas otherwise they would be bounded. In order to bound the neighbourhood relationships, the relationships could be built on entities from the areas surrounding the target area, but then the surrounding area ignored for the classification process.

First the Crime Data-Warehouse had to be created, involving extensive data-cleaning and data consolidation into a cohesive database schema (Chapter 6.1). Only then could the data-mining application be constructed (Chapter 6.2), with the goals significantly improving upon existing methods (Chapter 6.3).

## 6.1 Creating the Crime Data-Warehouse

The following discusses the data sources that were the input into the CDW and the modifications required to transform them into the CDW. There were several requirements which had to be met (Chapter 6.1.1) by the datasets (Chapter 6.1.2). The specific datasets used to build the CDW however first need to be cleaned (Chapter 6.1.3), then the addresses contained in them verified (Chapter 6.1.4) before they could be transformed via multiple processes (Chapter 6.1.5) into the final Crime Data-Warehouse (Chapter 6.1.6).

### 6.1.1 Input Requirements

The UnMASC framework is a spatial classification framework capable of using any spatial or non-spatial dataset given, with a few restrictions. First, the spatial datasets must contain some spatial information for each entity. This must be one of:

- **Point data** – An X-Y coordinate must be present indicating the location of the entity. It must contain exactly one vertex of the form  $e_p^{i,q} = (x, y)$  where  $x, y \in \mathbb{R}$ .
- **Line data** – An ordered set of vertices ( $e_p^{i,q} = (x, y)$  where  $x, y \in \mathbb{R}$ ) must be associated to the entity. *Spatial features* can then be used to make explicit the different properties of the line, such as *length*.



- **Polygon data** – A closed ordered set of vertices ( $e_p^{i,q} = (x, y)$  where  $x, y \in \mathbb{R}$ ), forming a 2D polygon, must be associated to the entity. *Spatial features* can then be used to make explicit different properties of the polygon, such as *area*.

Second, all entities of all types, both spatial and non-spatial, must each contain a unique identifier, which is required when explicitly establishing relationships between entities. Finally, relationships involving any non-spatial entities must be explicitly stated as these cannot be extracted. Spatial entities contain implicit relationships with other spatial entities, no explicit relationships between spatial entities are necessary as UnMASC is able to extract these. Although UnMASC is able to extract the relationships on-the-fly, the datasets used for experiments had the spatial relationships pre-processed.

### 6.1.2 Data Sources

The following datasets were used as input to the pre-processing algorithm which transformed them into the crime data-warehouse:

**Police Information Retrieval System (PIRS).** Data made available by the RCMP containing the location, time, and type of calls for service for the entire division of RCMP in British Columbia (BC) between August 1, 2001 and August 1, 2006. It records all calls, and contains data about the subjects (people), as well as vehicles and business, involved in the event and what their involvement is.

**British Columbia Assessment Authority (BCAA).** Data containing information for all plots of land within British Columbia, Canada. The price, location (address) and size of the plot of land and building are some of the features contained in this dataset. Each plot of land has a designated use assigned to it, such as *Bank*, *Duplex*, *High-Density Housing* which denotes the type of property it is supposed to be. This dataset is dated 2005.

**Road-Network.** The road-network data is from the GIS Innovations 2007 dataset and contains the set of roads in BC. This dataset contains features such as number-of-lanes, length, number of stop-signs, speed limit and location.

**Parks.** The set of parks, containing features such as the type of park, area and perimeter. The data is from the GIS Innovations 2007 dataset.

**Lines.** This dataset includes electricity transmission lines, railroad tracks and hiking trails. Spatial features, such as length, can be derived from the polygon associated with each entity. The data is from the GIS Innovations 2007 dataset.

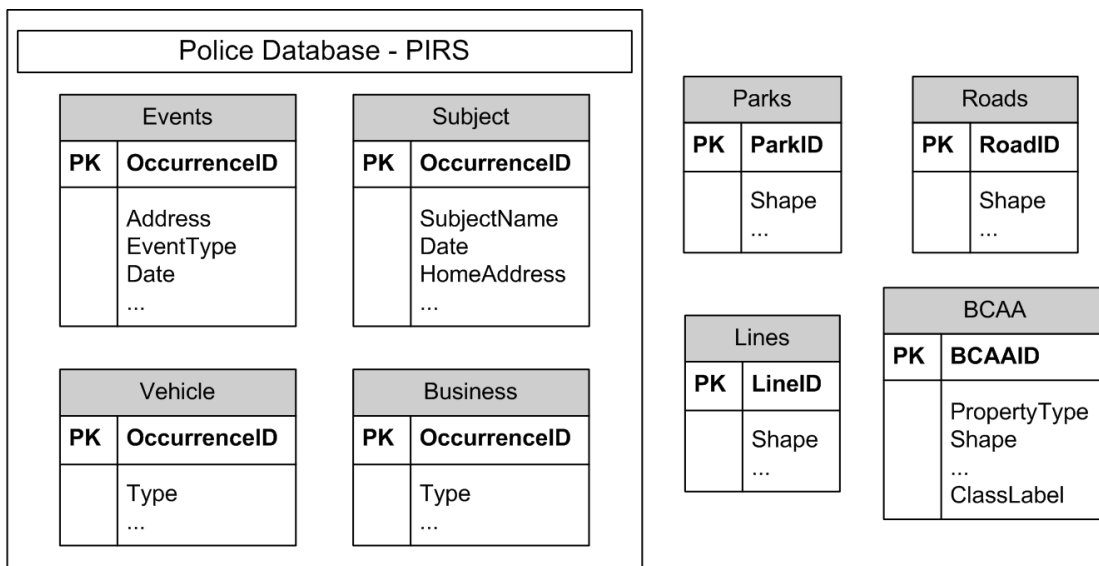
The schema of the input datasets is shown in Figure 40.

### 6.1.3 Data Cleaning

The original data came from multiple sources. The *Roads*, *Lines* and *Parks* came from a commercial source (GIS Innovations). The BCAA dataset likewise was from a commercial source. Both were in good shape and neither required cleaning.

The PIRS dataset, which contains the activities of the RCMP officers, came directly from the RCMP. It was mainly created by the activities and reports of the RCMP officers themselves, usually immediately after they finished some activity (search, arrest, ticket, etc) as part of their regular duties. This leads to minimal information entry, and hence missing fields are very common. The type of crime for each event was a required field. This dataset posed a very serious challenge. The *Address* field was very difficult to decipher and convert to a properly recognizable address, because:

- On-the-road police officers might stop someone in the middle of a rural area, with no apparent address to denote their exact location. This lead to entries in the *Address* field

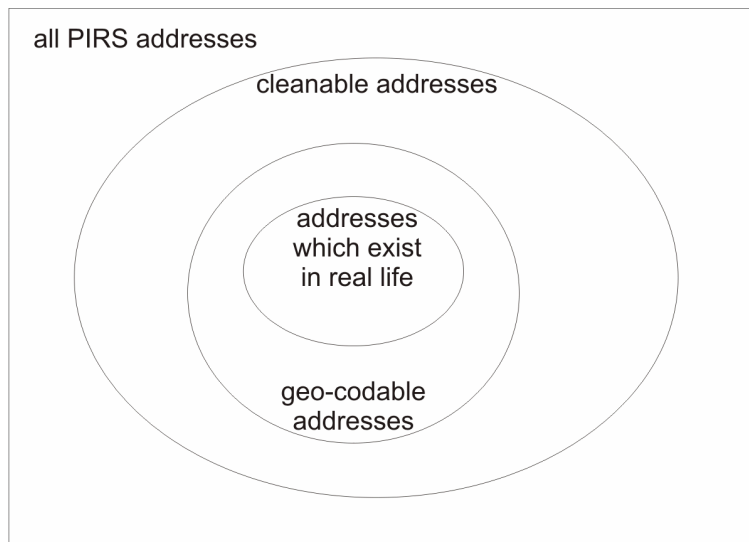


**Figure 40 – Original database setup**

which are not composed of the standard address components. For example, the *Address* field might contain an address between two consecutive houses or an intersection.

- Officers might enter in an address that is so imprecise that nobody out of that locality would know where the location is (for example, “Joe’s bar”).
- Different people use different syntax for entering addresses. Some enter the house number and street-name followed by city (for ex: “88 Main St., Burnaby”), while others enter it in reverse order (for ex: “Burnaby, Main St. 88”). The separator between the address components could be a comma, a period, or a space. Apartment numbers confuse the issue further, with some entering it immediately behind the house-number, possibly separated by a dash or space, while others enter it after the street-name. Some denote it by a #, while others add “apt.”.

Before the addresses can be used for any sort of analysis, they had to be converted into a standard format. This was done using a custom written complex parsing algorithm which attempted to determine what each segment of the address meant, based on its location within the address-string. The result of this process was an address in a standardized format, which a geography program would automatically understand. For example, “88 Main Str., Burnaby Apt 9” was converted into “88 Main Street, Apt 9, Burnaby”. Any addresses which could not be parsed cleanly were discarded. This does not however mean that the address actually exists (for example, there is no “Main Street” in Burnaby), only that the string is now in a standard format.



**Figure 41 – Address cleaning process**

### 6.1.4 Address Geo-coding and Validation

The result of the data cleaning process was a set of addresses which were in a standard format. The next step was to assign an X/Y coordinate denoting their real world geographic location, in the form of a longitude/latitude coordinate. This process is called geo-coding. ArcGIS 9.2 was used to geo-code the addresses from PIRS. The above mentioned GIS Innovations 2007 Road-Network dataset was used to geo-code against, which pruned out addresses that cannot exist because the specified street does not exist in the specified city, or the street does not extend to the house number specified. For example, “88 Main Street, Burnaby” would not geo-code because there is no “Main Street” in Burnaby. As another example, “4000 Carrigan Court” would not geo-code because Carrigan Court does not extend to the 4000-block.

The addresses that pass the geo-coding phase do not automatically fall into the set of valid addresses. The geo-coding process maps each address to a specific location on a specific street, but it does not mean that there is actually a building at that location. For example, “3906 Carrigan Court, Burnaby”, although a valid address, does not exist, but “3901 Carrigan Court” and “3911 Carrigan Court” do. The BCAA dataset (Chapter 6.1.2), containing the known list of valid

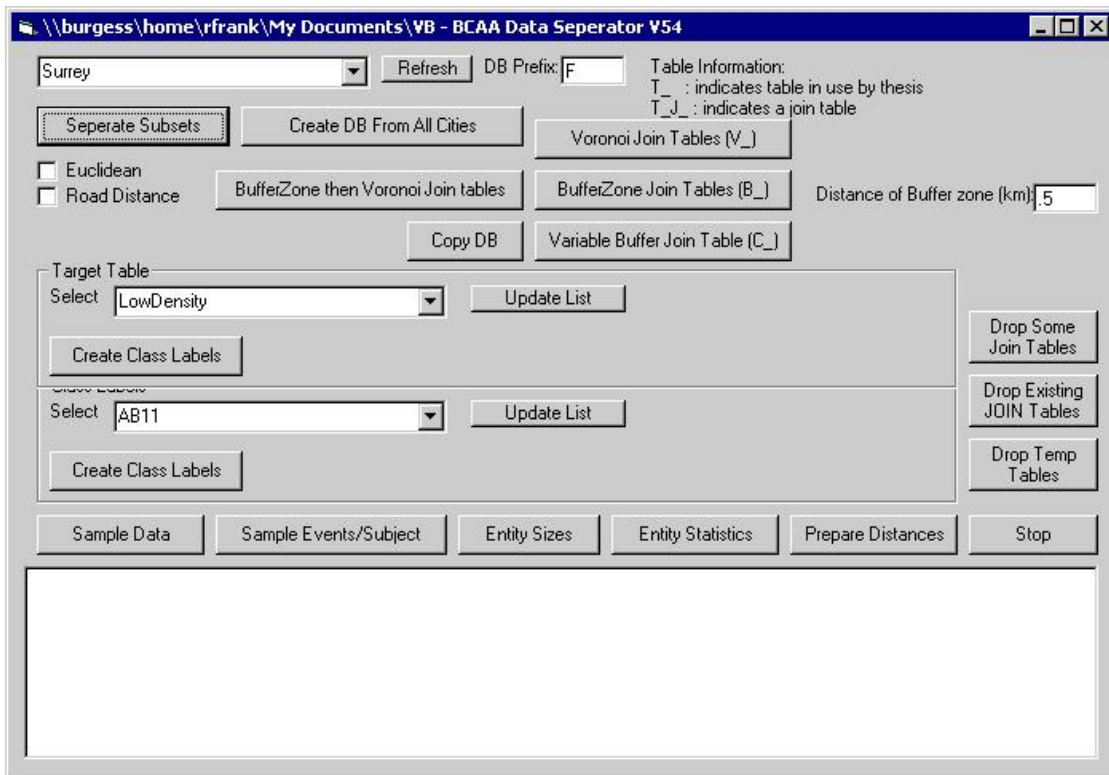


Figure 42 – Data pre-processor program

addresses, was used to validate the addresses. Thus, all crime addresses which could be parsed and geo-coded were checked against addresses in BCAA. If they existed then they were used for calculating the class label (see Chapter 6.1.5), otherwise they were discarded since they cannot be valid.

The process described here can be visualized as the Venn diagram in Figure 41. This process starts out with the set of all PIRS addresses, the addresses are cleaned, geocoded, and validated against the true set of addresses. The result is the subset of addresses which actually exist in real life. These addresses were then assigned class-labels derived from the PIRS dataset.

### 6.1.5 Data Transformation

The initial data was spread across multiple data sources. Even the class label of the target entities were not known explicitly. To get from the source datasets to the final Crime Warehouse, the data was transformed. In Stage 1, the class label was derived from the police data. In Stage 2, to store the neighbourhood relationships in an obvious and self-explanatory manner, the BCAA database, which consisted only of a single table, was split into multiple entity tables, one table per each entity-type. In Stage 3, all spatial features of all spatial entities were extracted since they never change. In Stage 4, the relationship between all entities was extracted, since the location and extent of the entities does not change, hence the relationship will never change either. Aggregate features, such as single and multi-feature aggregation literals, could not be calculated a-priori since they will constantly change depending on the conditions put on the rule-learner by the rule

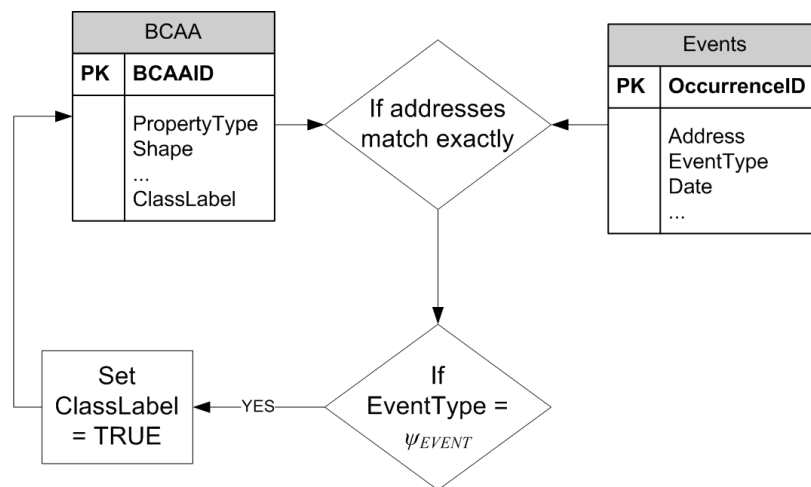


Figure 43 – Stage 1) Creating class labels

that it is learning. Had this been necessary, this would have been the final stage, Stage 5.

These different stages are described in detail below. The pre-processing program interface is shown in Figure 42.

### Stage 1) Creating the class label

To start the pre-processing stages, each entity in BCAA was assigned a Boolean class label, which was initially set to *false*. Using the PIRS dataset from the RCMP, each crime of type  $\psi_{CRIME}$  from PIRS that was within the city selected ( $\psi_{CITY}$ ) was matched by address (if an exact match was possible) to an entity in BCAA, and the Boolean class label for that entity in BCAA was changed to *true*. This process is shown in Figure 43. For example, if a burglary occurred at “8888 University Dr., Burnaby, BC”, the class label for the entity representing Simon Fraser University (which is located at “8888 University Dr.”) was changed to *true*. For example, information from PIRS was used to infer the class label for all residential properties in Burnaby: if a residence has been burglarized according to PIRS then the entity is labelled *burglarized*, otherwise it is labelled as *not-burglarized*. The respective task is to learn rules to determine why those residential properties were burglarized.

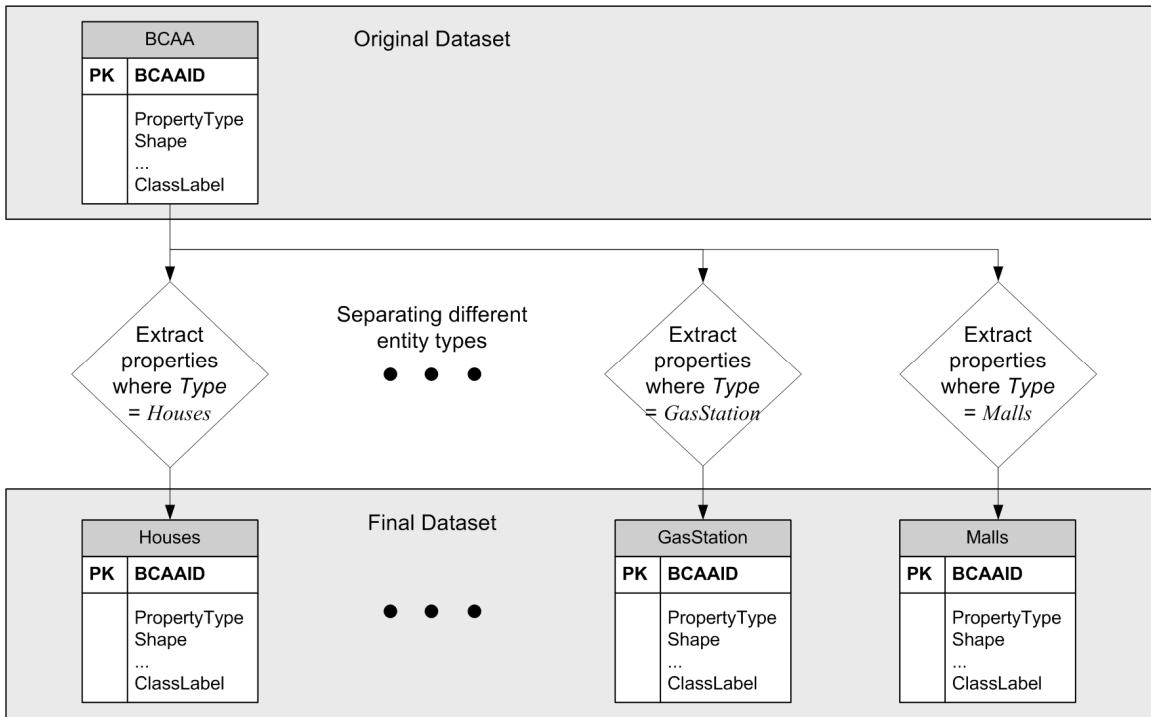


Figure 44 – Stage 2) Separating BCAA entity-types

Once an entity's class label has been changed to *true*, it cannot be changed anymore, it is going to stay *true* regardless of how many further crimes of type  $\psi_{CRIME}$  occur at that location. After an attempt was made to assign each crime to a BCAA entity, all the entities in the BCAA dataset are considered labelled and Stage 2 in the pre-processing is started.

### Stage 2) Separating the different entity-types

Initially, the BCAA database consisted of a single table, listing all of the properties within the Greater Vancouver Regional District. One of the fields of the BCAA table was *Type* which contained one of 26 different codes denoting different entity-types (low-density housing, commercial, etc). All entities of the same type were separated from BCAA into a table of their own, resulting in 26 tables, each containing entities of only a single type. In addition to these 26 entity-types, the entity-type *roads*, *lines* and *parks* were also integrated into the CDW, resulting in 29 entity-types in total. This process is shown in Figure 44, while the full list of entities after this stage is shown in Appendix Appendix B.

### Stage 3) Extracting spatial features

Another critical component in spatial databases is the full set of features for each spatial entity. For example, the area of entities might not be explicitly stated, but is important when working with house-plots, forests, lakes, etc. These spatial features are implied by the polygon that is associated with the entity itself. This step makes them explicit. From the polygons the spatial features are extracted by the use of spatial functions (see Chapter 4.3.2 for more details) and the resulting value stored as part of the CDW. For each entity-type now in the database, the spatial features listed in Figure 28 (page 48) and Figure 29 (page 48) are extracted and stored in the database along with the other features of that entity-type. This is shown in Figure 45.

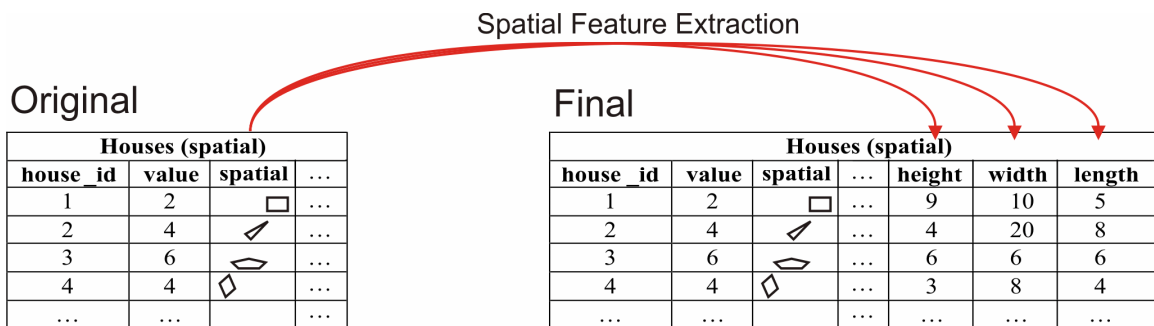


Figure 45 – Stage 3) Extracting spatial features during the preparation of the CDW

### Stage 4) Extracting the Relationships

The rule-learner makes use of all available information contained in the database, including the relationships between entities. These relationships are usually not explicitly stated, but only implicitly defined by their spatial location. This step makes them explicit by calculating them and storing them in the CDW. Following Figure 46, in the first step, all pairs of entities are retrieved and analyzed to determine whether they are neighbours or not. If there is no neighbourhood relationship between a certain pair (Stage 4a), the algorithm moves onto the next pair (Stage 4b). However, if there is a relationship, then the features of that relationship are calculated (Stage 4c) and stored in the database in a table which corresponds to the two entities being analyzed (Stage 4d). The algorithm for this process is shown in Figure 47.

Information about relationships between each type of entity must be stored in the database. Therefore, a relationship-table was created for each entity-type combination, hereafter denoted as  $J\{t_i, t_j\}$ , where  $t_i$  and  $t_j$  denote 2 types of entities (Figure 47, lines 2 and 7). If  $t_i$  equals  $t_j$ , then the relationship-table stores the relationships between entities of the same type. To establish the relationships between entities, UnMASC uses Voronoi Diagrams and its dual, the Delaunay Triangulation, as defined by Definition 7 (page 30). The Voronoi Cells and Delaunay Triangulation are calculated for entity-type  $t_i$  by the QHull algorithm [80] which is treated as a

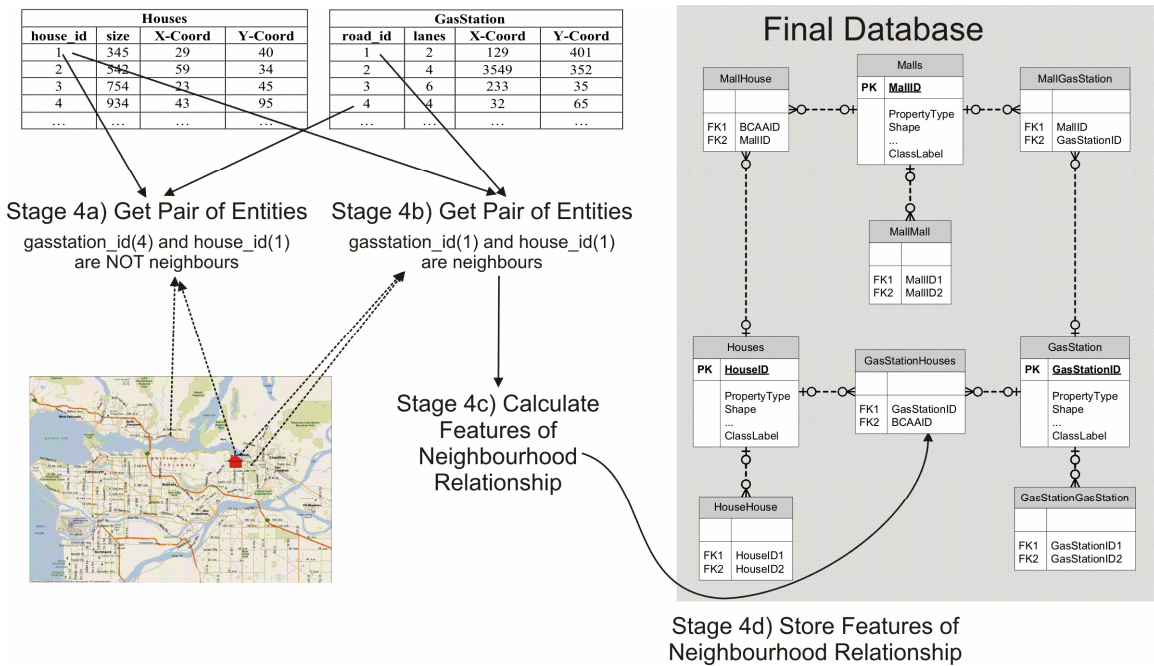


Figure 46 – Stage 4) Extracting Neighbourhood Relationships



black-box (Figure 47 – Line 3). The output of this black-box is the Voronoi structure, and a list of entity-pairs  $\{e^{i,1}, e^{i,2}\}$  denoting the existence of a relationship between entities  $e^{i,1}$  and  $e^{i,2}$ , both of type  $t_i$ . All such pairs are added to the database (lines 4-5). For all entities of type other than  $t_i$ , they are assigned to the entity whose Voronoi Cell they fall into (lines 8-10). This establishes the neighbourhood relationships which exist between all spatial entities in the database. The rule-learning makes use of these pre-processed relationships.

When this stage is reached, for  $|T|$  spatial entity-types there are  $|T|$  entity tables in the database. For the Crime Data-Warehouse that was built for this thesis  $|T|=29$ . Since in a spatial database, any entity-type can neighbour any other entity-type, a relationship table exists between each pair of entity-types. Since there is a relationship between all spatial entity-types, the database schema for the spatial entities forms a clique (see Figure 48 for a depiction of a subset of the CDW). In addition, entities are also related to other entities of the same type. For  $|T|$  entity-types, there are  $\binom{|T|}{2}$  relationship tables between entities of different types, and  $|T|$  tables between entities of the same type, resulting in  $\binom{|T|}{2} + |T| = \frac{|T|!}{2!(|T|-2)!} + |T|$  relationship tables. In the Crime Data-Warehouse, where  $|T|=29$ , the number of relationship tables that are generated is 435.

Whether two entities are connected via a relationship depends on the neighbourhood definition being used (see Chapter 3). If two entities are not neighbours, there is no corresponding entry for

---

**Algorithm 1** CalculateNeighbourhood (database  $DB$ )

---

```

1: For each spatial entity-type  $e^i$  in  $DB$ 
2:    $DB \leftarrow DB + J\{e^i, e^i\}$  // create table in  $DB$  referencing ID of  $e^i$  twice to store relationships
   // between  $e^i$ 
3:   // calculating relationship between entities of the same type  $e^i$ 
    $\{D, V\} \leftarrow QHull(e^i)$  // QHull is an external program and treated as a black-box
   //  $D$ : a set of neighbouring entities according to the Delaunay Triangulation based on entities in  $e^i$ 
   //  $V$ : a set of Voronoi cells according to the entities in  $e^i$ 

4:   For each pair  $\{e^{i,1}, e^{i,2}\} \in D$ 
5:     Add pair  $\{e^{i,1}, e^{i,2}\}$  into  $J\{e^i, e^i\}$ 
   // calculating relationship between entities of different types
6:   For each spatial table  $e^j \in DB - e^i$ 
7:      $DB \leftarrow DB + J\{e^i, e^j\}$  // create table in  $DB$  referencing ID of  $e^i$  and  $e^j$  to store relationships
   // between  $e^i$  and  $e^j$ 
8:     For each entity  $e^{i,k} \in e^j$ 
9:        $e^i \leftarrow$  entity corresponding to the cell  $V$  that  $e^j$  falls into
10:      Add pair  $\{e^i, e^j\}$  into  $J\{e^i, e^j\}$ 
11: Return ( $DB$ )

```

---

**Figure 47 – UnMASC Algorithm to calculate neighbourhood relationships in a database**

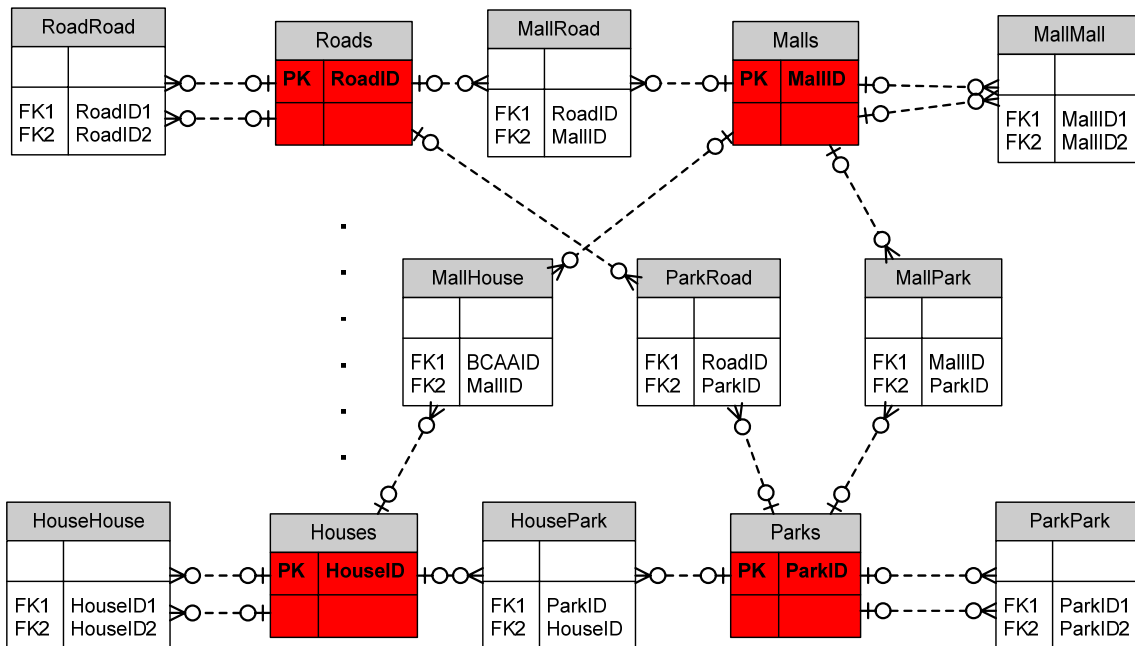
those entities in their relationship table. However, if two entities are neighbours, the relationship table contains the unique identifiers of the entities involved in the relationship, along with details of the relationship, such as direction, distance, etc (for a full list, see Figure 30 and Figure 31).

**Stage 5) Creating neighbourhood features via aggregation**

As each classification rule is being built, conditions (literals in the rule) are applied to the rule during run-time. These conditions change the set of entities under analysis; hence the results of the aggregation function are going to change. This dictates that the features of neighbourhoods (aggregation literals) must be extracted during the classification task and cannot be done as a pre-processing stage.

**6.1.6 Dataset Descriptions after Pre-processing**

As discussed above, for  $|T|$  spatial entity-types there are going to be  $|T|$  **entity** tables in the database. However, a relationship table exists between each pair of entity-types  $t_i$  and  $t_j$  (where  $t_i$  could equal  $t_j$ ) resulting in  $\binom{|T|}{2} + |T|$  **relationship** tables in the database. Aside from the relationship tables between entities of the same type, the schema is a clique. A full list of the



**Figure 48 – Partial schema of the crime data-warehouse. Entities are shown in red. The full schema contains 29 entity tables, and 435 relationship tables**

entity-types used for the experiments can be found in Appendix Appendix B, but since the database schema used for the experiments requires 435 relationship tables, visualising the full schema is impractical. A subset of the schema is shown in Figure 48. The Crime Data Warehouse did not contain any non-spatial entity sets, but those would have integrated seamlessly.

## 6.2 UnMASC Implementation Details

The following section describes how rules are built with UnMASC, a spatial classifier capable of performing rule-learning in parallel using spatial and non-spatial aggregations. Based on the database schema illustrated in Figure 1, page 1, the following chapter will use the rule R17:  $\text{profit}(M, \text{'Yes'}) \leftarrow \text{mall}(M)$  as the basis for the examples.

UnMASC uses a top-down method of learning rules. It is based on the popular sequential covering algorithm [45], similar to CrossMine [100], which works as follows. The classification model is generated one rule at a time and refined incrementally by adding literals until a termination condition applies (for example Minimum Support). Once a rule is finalised, the covered entities are removed from the training set, and a search for another rule starts. The rule-learning process ends when it is impossible to generate a new rule.

For any pair of two entity-types  $t_i$  and  $t_j$ , where possibly  $t_i = t_j$ , that need to be evaluated for any candidate literals, the rule-learner finds the next best literal by retrieving from the spatial join index the necessary neighbourhood relationships between  $t_i$  and  $t_j$  (Figure 49 steps 1-2). For the entities of type  $t_j$  that are deemed to be neighbours to entities of type  $t_i$ , all features of  $t_j$  are retrieved for additional analysis (steps 3-4). The relevant entities, their neighbourhood relationships and features, are then passed to the spatial rule learner (step 5) which calculates the aggregations and spatial features (step 6). The rule-learner then evaluates all features, keeps track of a set of candidate literals, and when the search is complete, adds the best candidate literal to the rule. Once no further candidate literals can be found, the rule is aborted, the entities satisfied by the rule removed from the dataset, and a new one started. The classification model is deemed complete, when no further rules can be created.

### 6.2.1 UnMASC Details

In most state-of-the-art algorithms, such as CrossMine [100], the evaluation of all candidate literals takes place one after another. This is highly inefficient given that current computers usually have multiple CPUs. When evaluating refinements to the rule, different neighbouring entity-type combinations (called *neigEntity*) need to be considered, depending on what the rule

already contains. The set of all *neigEntity*'s is called *neigEntitySet*. Since the decision of which literal to add to the rule is made only after the FOIL-Gain has been evaluated for all possible rule extensions within all *neigEntity*'s, the evaluation of multiple rule extensions can be performed independently and in parallel. UnMASC takes advantage of this by searching for the best candidate literal (i.e., possible rule extension) within multiple *neigEntity*'s simultaneously.

UnMASC itself is composed of two methods, *RuleLearner* (Chapter 6.2.2) and *LiteralEvaluator* (Chapter 6.2.3), which are structured in a server-client relationship. This is shown in Figure 50.

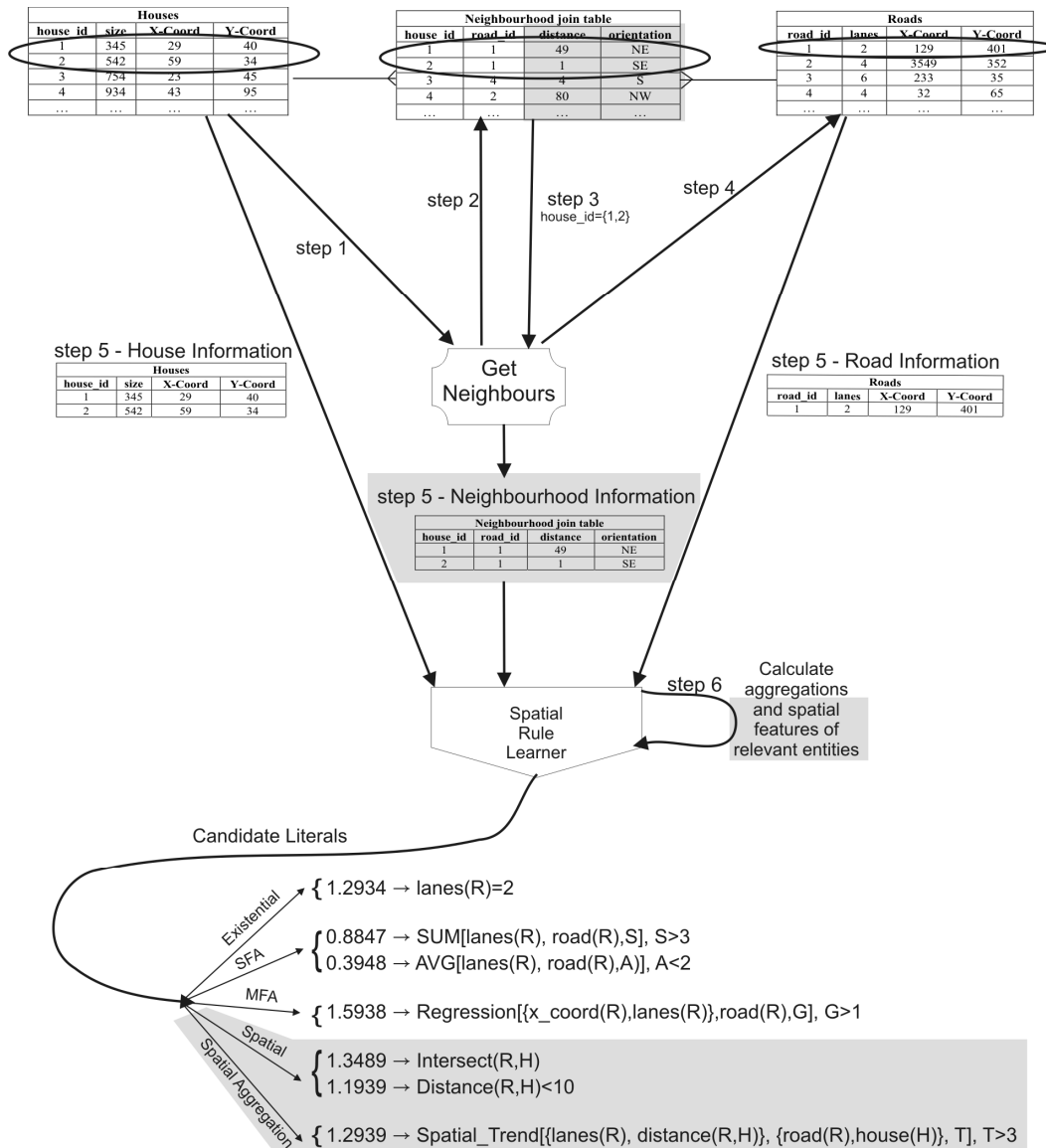
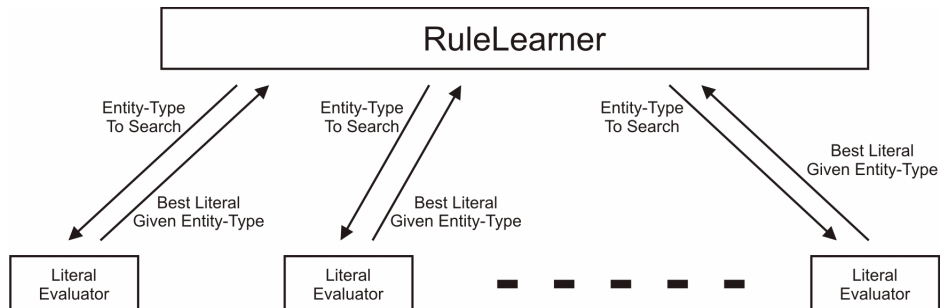


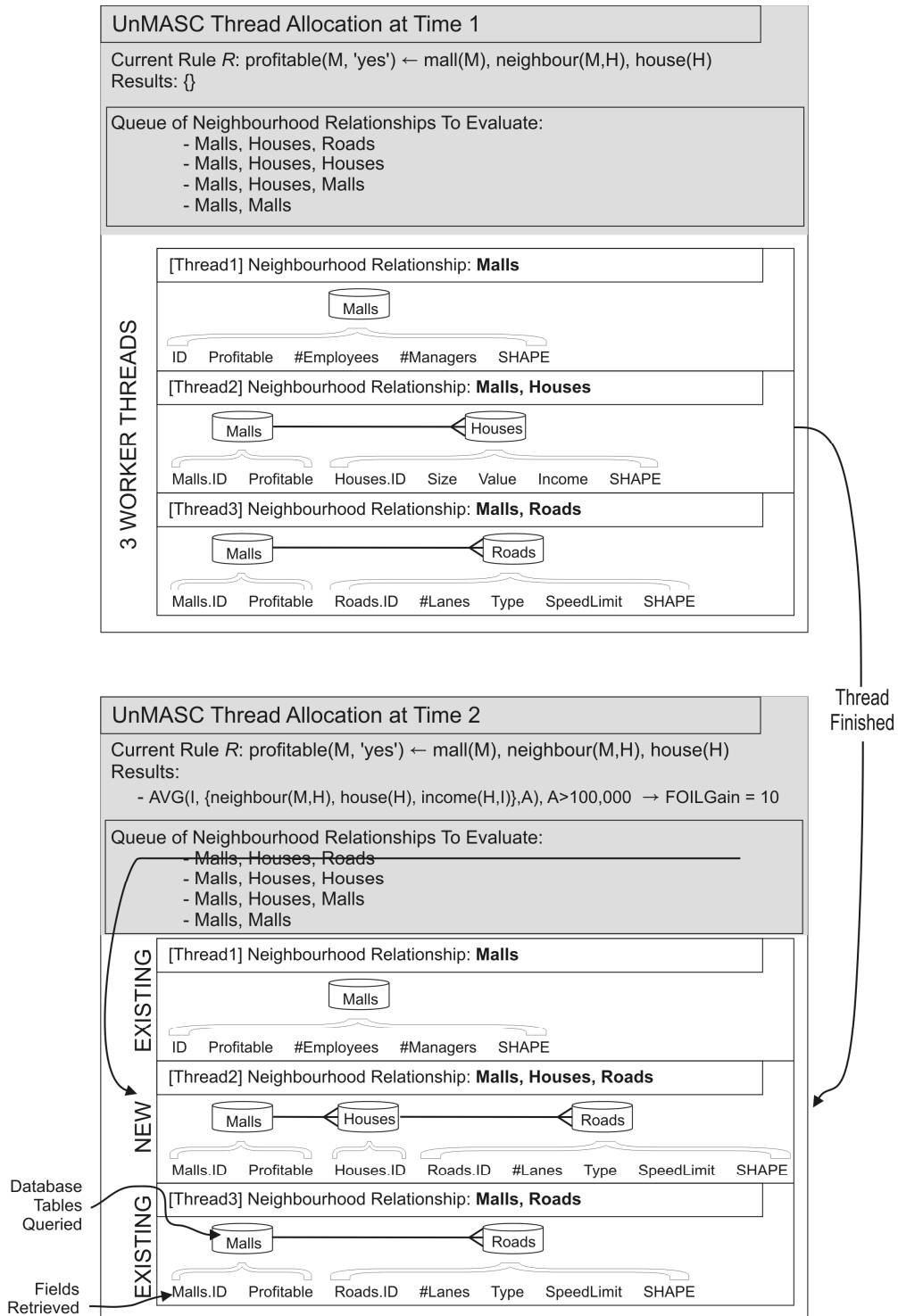
Figure 49 – Steps in candidate spatial literal generation. Components specific to spatial classification are highlighted.

Upon inception, *RuleLearner* launches multiple *LiteralEvaluator* threads simultaneously, each of which wait for *RuleLearner* to assign it a *neigEntity* to search. The number of simultaneous threads is limited by the number of CPUs in order to have each run on a separate CPU. If the size of *neigEntitySet* (i.e., the number of neighbourhood relationships that need to be searched in order to find the best candidate literal) exceeds the available CPUs, then they are placed into a queue and are evaluated when a CPU becomes available. This setup is illustrated in Figure 51. The different *LiteralEvaluator* threads all share the same main memory, although each has its own independent portion since each works with a different subset of the data. For each thread, the relevant entity-types, entities and features are retrieved from the database and loaded into main memory. *LiteralEvaluator* performs all aggregations and spatial analysis in order to find the best rule extension for that specific *neigEntity*. When the best candidate literal (for a specific *neigEntity*) has been found, *LiteralEvaluator* returns the best literal and FOIL-Gain to *RuleLearner*. At this time, *LiteralEvaluator* releases the memory, *RuleLearner* stores the result of the thread with the other results, and the next queued task is assigned to the available *LiteralEvaluator* (Figure 51 - Time 2). *RuleLearner* waits for all *NeigEntity*'s to be searched, then makes a global decision as to what the best candidate literal is.

In this fashion, UnMASC is able to perform the rule learning process in parallel. *RuleLearner* manages the rules and literals that have been built so far, while also managing the entity-types that need to be searched when evaluating the next best candidate literal. For each entity-type that needs to be searched, *RuleLearner* retrieves the appropriate data from the database and calls on *LiteralEvaluator* to perform the search. All single-, multi- and spatial-, aggregations are evaluated within each *LiteralEvaluator* in order to find the best literal for that specific entity-type.



**Figure 50 – UnMASC Client-Server Model**



**Figure 51 – Search process Example**

## 6.2.2 Method: RuleLearner

Initially, the rule-learning starts with an empty rule  $R$ , with the target-entity-type  $t_\tau$  given (Figure 52, lines 1-5). *RuleLearner* assembles the set *neigEntitySet* of *neigEntity* that require evaluation. If  $R$  is empty, then all entity-types that have a neighbourhood relationship with entities of type  $t_\tau$  are added into the set. Otherwise any entity-type with a neighbourhood relationship with any referenced entity-type in  $R$  is added to the set (lines 8-10). Once *neigEntitySet* is complete, *RuleLearner* starts to evaluate each. It does so by calling *LiteralEvaluator*, which, given the current rule and target entity IDs that satisfy the rule thus far, finds and returns the best literal (lines 11, 15). UnMASC introduces a queuing system into the algorithm, which allows for the parallelization of the literal evaluations (lines 12-14). The number of simultaneous threads of *LiteralEvaluator* is limited by the number of CPUs present. As a *LiteralEvaluator* completes the search for the best rule extension for a given *neigEntity*, a new *neigEntity* is assigned to it. This process is repeated until *NeigEntitySet* is empty, at which point the *neigEntity* that returned the highest FOILGain is selected and, if the resulting rule satisfies the minimum support criteria *minSupp*, is added to the rule (lines 16-18). If none of the refinements of a rule achieves the minimum support, then the current rule is considered complete, a new rule is started and the entities in the training dataset which satisfy the rule are removed (lines 19-20). The algorithm

---

**Algorithm 1** RuleLearner (*minSupp*, *DB*, *TargetEntityType*, *TargetLabel*)

---

```

1: RuleSet  $\leftarrow \emptyset$ 
2: EntityIDs  $\leftarrow$  IDs of entities of type TargetEntityType
3: UncoveredEntityIDs  $\leftarrow$  EntityIDs
4: while |UncoveredEntityIDs| > |EntityIDs|  $\times$  minSupp           //start a new rule
5:   Rule  $\leftarrow$  empty rule
6:   RuleEntities  $\leftarrow$  EntityIDs covered by Rule
7:   while |RuleEntities| > |EntityIDs|  $\times$  minSupp           //search for literal
8:     neigEntitySet  $\leftarrow \emptyset$ 
9:     for each neigEntity with relationship to entity-type referenced in Rule
10:      neigEntitySet  $\leftarrow$  neigEntitySet + neigEntity
11:      while |neigEntitySet| > 0
12:        while all threads busy
13:          WAIT
14:          Start Thread [ LiteralEvaluator(Rule, neigEntity, RuleEntities) ]
15:          neigEntitySet  $\leftarrow$  neigEntitySet - neigEntity
16:          BestLiteral  $\leftarrow$  result with highest FG value from all results
17:          Update Rule by adding BestLiteral
18:          RuleEntities  $\leftarrow$  remove from RuleEntities entities covered by BestLiteral
19:          UncoveredEntityIDs  $\leftarrow$  remove from UncoveredEntityIDs entities covered by Rule
20:          RuleSet  $\leftarrow$  RuleSet  $\cup$  {Rule}
21: return RuleSet

```

---

**Figure 52 – UnMASC RuleLearner (FG = FOILGain value)**

---

aborts when the set of entities not covered by any rule fall below the threshold  $minSupp$ , since at that point no literal can be constructed to satisfy that condition.

### 6.2.3 Method: LiteralEvaluator

*LiteralEvaluator* (Figure 53) returns the best literal and FOIL-Gain, given a *neigEntity* and training entities not covered by any previous rule. First, *LiteralEvaluator* retrieves the required dataset for analysis (lines 2-3), and, for each feature of *neigEntity*, applies all appropriate aggregation functions (lines 4-7). Since UnMASC is capable of performing multi-feature aggregation, for each feature of the entity, a second feature is selected and aggregated (lines 8-11). UnMASC also performs spatial aggregation, hence if *neigEntity* is a spatial entity, then the spatial features are extracted and spatial aggregation is done (lines 12-17). The aggregation, feature and threshold with the highest FOIL-Gain for that *neigEntity* are then returned to *RuleLearner* (line 19).

Since this method is part of a server-client architecture, when the candidate literal is returned to *RuleLearner*, the *LiteralEvaluator* method is complete and goes into a stand-by mode until a new *neigEntity* is assigned to it.

---

**Algorithm 2** *LiteralEvaluator(rule, neigEntity, entities)*

---

```

1:  $FG' = 0$ 
2: Determine relationships between entities covered by rule and entities in neigEntity
3: Retrieve relationships from database
4: for all currFeat1 of neigEntity
5:   for each singleAgg from all single-feature aggregation functions (incl. existential)
6:      $FG \leftarrow$  Calculate FOIL-Gain for currFeat1 using singleAgg
7:     if  $FG' < FG$  then [currFeat1', currFeat2', Aggr',  $FG'$ ] = [currFeat1, , singleAgg,  $FG$ ]
8:   for all currFeat2 of neigEntity
9:     for each multiAgg from all multi-feature aggregation functions
10:       $FG \leftarrow$  Calculate FOIL-Gain for <currFeat1, currFeat2> using multiAgg
11:      if  $FG' < FG$  then [currFeat1', currFeat2', Aggr',  $FG'$ ] = [currFeat1, currFeat2, multiAgg,  $FG$ ]
12:   if neigEntity is a spatial entity
13:     extract spatial features
14:     for each spatial feature spatFeat of neigEntity
15:       for each spatAgg from all spatial aggregation functions
16:          $FG \leftarrow$  Calculate FOIL-Gain for <currFeat1, spatFeat> using spatAgg
17:         if  $FG' < FG$  then [currFeat1', currFeat2', Aggr',  $FG'$ ] = [currFeat1, spatFeat, spatAgg,  $FG$ ]
18:    $candLit \leftarrow$  [neigEntity, currFeat1', currFeat2', Aggr',  $FG'$ ]
19: return candLit

```

---

**Figure 53 – UnMASC LiteralEvaluator (to denote best solution, an ' is used,  $FG$  = FOILGain value)**



## 6.2.4 Evaluation Optimization

There are considerable differences between the complexities of each *neigEntity* issued to the different *LiteralEvaluator* threads. For example, using the *neigEntity*s from Figure 51 Time 1, the cost for evaluating the *neigEntity*  $\{Malls\}$  is expected to be relatively small since there are only a few malls in a city and no aggregations are involved (since there's only a single entity-type).  $\{Malls, Houses\}$  however is expected to incur much higher cost since there are many houses in a city, and the evaluation requires that houses be aggregated over their neighbouring malls.  $\{Malls, Roads\}$  incurs a cost that is between the other two threads since the number of roads is likely to be smaller than the number of houses in a typical city.

Since the cost for each *neigEntity* varies, sometimes greatly, it is possible that a very costly *neigEntity* is evaluated last, in which case all but one of the threads are idle and the benefits of parallelization are invalidated. The risk of this depends only on the variability of the number of entities of each entity-type: the larger the variability, the larger the differences in task-sizes. If the cost can be estimated well enough, the queue containing all the *neigEntity*s can be reprioritized to avoid this scenario.

Since the size of each evaluation is unknown a priori, an approach to approximating the evaluation cost of each literal is introduced in this thesis. This is done as follows. Each *neigEntity* is made up of a permutation of  $n$  entity-types which can be denoted as  $\{t_1, \dots, t_{k-1}, t_k, \dots, t_n\}$ , where  $t_1 = t_r$ . Each entity-type  $t_{k-1}$  has, on average,  $|t_{k-1} \triangleright \triangleleft t_k| / |t_{k-1}|$  neighbours of type  $t_k$ , where  $|t_{k-1} \triangleright \triangleleft t_k|$  denotes the number of relationships between all entities of type  $t_{k-1}$  and  $t_k$ , and  $|t_{k-1}|$  denotes the number of entities of type  $t_{k-1}$ . For example, assume there are 10 malls ( $|t_{k-1}|$ ) and in total 100 neighbourhood relationships between malls and houses ( $|t_{k-1} \triangleright \triangleleft t_k|$ ), then this implies that, on average, there are  $100/10 = 10$  neighbouring houses per mall. Thus the total number of relationships between the target entity-type ( $t_1$ ) and the entity-type being searched ( $t_n$ ) can be estimated by

$$\text{cost}(t_1, t_n) = \prod_{k=2}^n \frac{|t_{k-1} \triangleright \triangleleft t_k|}{|t_{k-1}|}.$$

Multi-feature and spatial aggregation is then performed on the features ( $f$ ) of the entity-type that is evaluated ( $t_n$ ) by choosing two features to aggregate, which can be done in  ${}_f C_2 = \frac{f!}{2!(f-2)!}$  ways. Thus the total cost of a specific *neigEntity* is estimated by

$$\text{total\_cost}(t_1, t_n) = |t_k| \frac{f!}{2(f-2)!} \prod_{k=2}^n \frac{|t_{k-1} \triangleright \triangleleft t_k|}{|t_{k-1}|}$$

### 6.3 Comparison to CrossMine

To highlight the differences between the proposed approach and a popular multi-relational (MR) rule-learner, CrossMine [100], the search processes of both algorithms are compared. The entity-types that are referenced by a rule are referred to as *activated* whereas the entities that satisfy a particular rule are said to be *covered*.

CrossMine introduced the concept of TupleID propagation to efficiently mine MR classification rules using the ILP framework with the existential operator. The propagation appends to each entity of a non-target entity-type the IDs and class labels of entities which are related to each entity of the target type. The following example illustrates how TupleID, in the context of aggregation, cannot do what spatial classification needs. Using the running example of Figure 1 (page 1) CrossMine works as follows. Starting with the target entity-type *Malls* (Figure 54), the

Malls			
MallID	Profitable	# Employees	...
1	Y	50	...
2	Y	20	...
3	N	24	...

Figure 54 – *Malls* entities from the original spatial database

Roads				
RoadID	Type	...	MallID	Profitable
1	Highway	...	1, 2, 3	Y, Y, N
2	Main	...	1, 2	Y, Y
3	Court	...	1, 3	Y, N

Figure 55 – *Malls* propagated to *Roads* entities

Houses				
HouseID	Size	...	MallID	Profitable
1	900	...	1, 3	Y, N
2	1300	...	2	Y
3	4000	...	1, 2	Y, Y

Figure 56 – *Malls* propagated to *Roads*, then to *Houses*

ID of each entity from *Malls* is propagated and appended to all neighbouring *Roads* (Figure 55).

During the next iteration of propagations, *Roads* becomes the source table for the propagation and the related entity-type *House* becomes the destination. For each entity of type *House*, the neighbouring *Roads* are determined and the *IDs* and class labels of *Malls* are appended to *House*. The resulting set of features for *House* now contains all the original features of *House*, the feature *ID* and class label from *Malls* (Figure 56). Aggregation of houses can now be performed to find the ‘*total number of unique houses neighbouring each mall*’. Aggregating houses over roads to determine the ‘*number of houses neighbouring each road*’ however is not possible since there is no ID information from *Roads* in *House*. Due to this, TupleID propagation does not allow for aggregation over previously referenced tables but only the target table. Furthermore, no spatial aggregation can occur since there is no distance information within the set of features after propagation. CrossMine itself, as implemented in [100], does not perform **any** type of aggregation.

In order for CrossMine, when applied to spatial data, to perform literal-searches to the extent of UnMASC, the spatial dataset needs to be fully converted to a multi-relational dataset by pre-materializing all spatial features and all the spatial features of the spatial join index. Even with all of these features pre-materialized, the algorithm itself also must be extended to consider multi- or spatial-aggregation, since no algorithm does any of this.

Furthermore, spatial trends are impossible with the technique CrossMine employs. This is due to the way spatial trend works. In order to calculate it, information from two tables must be analyzed simultaneously: *distance* from the relationship-table and a feature value from the entity table. With CrossMine this information simply isn’t available since it only concentrates on a single table at a time. This is by design and is how CrossMine is able to gain runtime benefits.

Performing TupleID propagation is much more complex programmatically than table-joins in the database, and the run-time benefits of TupleID propagation would most likely be limited by the single-thread performance of the implementation (when compared to performance of the database engine on a multi-processor machine). Hence, in conclusion, the technique of TupleID propagation has too many shortcomings and is not be used by UnMASC.

## CHAPTER SEVEN: UNMASC EXPERIMENTS ON A CRIME DATA- WAREHOUSE

This Chapter presents the spatial rule-learning experimental results of UnMASC on the Crime Data Warehouse. First the experimental setup and numerical results are presented (Chapter 7.1) followed by anecdotal analysis (Chapter 7.2) as a result of analysis performed with the help of experts from the School of Criminology at SFU.

### 7.1 Experiments

Extensive experimentation was performed on the UnMASC framework. The datasets used for the experiments are discussed in Chapter 7.1.1, with the results being discussed in Chapter 7.1.2 and onwards. A screenshot of the UnMASC program is shown in Figure 58.

#### 7.1.1 Datasets

Based on input from experts in the Criminology Department at SFU, several classification scenarios were designed on which UnMASC was evaluated. Each classification scenario consisted of a city (denoted by  $\psi_{CITY}$ ), a crime type ( $\psi_{CRIME}$ ) and a target class ( $\psi_{LABEL}$ ). Independent datasets were created within the Crime Data-Warehouse in order to support these

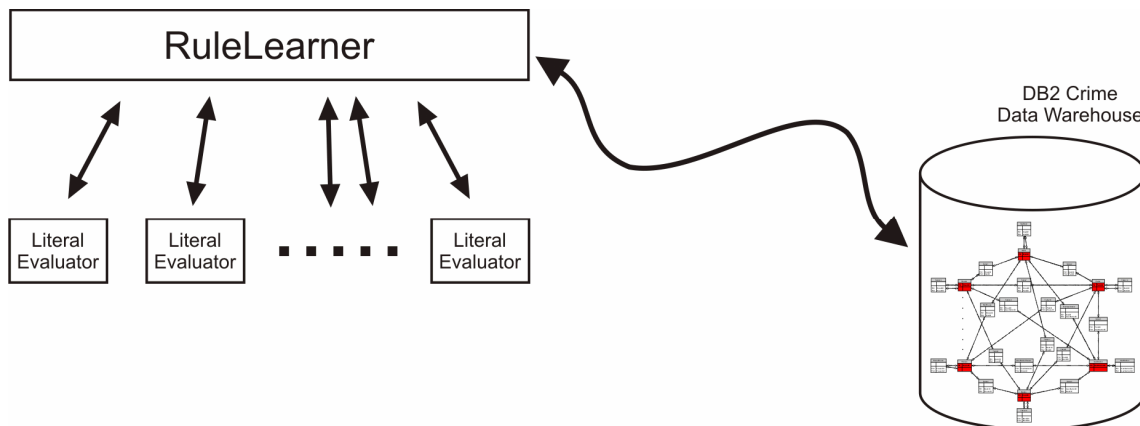


Figure 57 – UnMASC classifier and the Crime Data-Warehouse

classification scenarios (details in Chapter 6.1). All scenarios were tested and compared for their impact on precision (Chapter 5.6.1), accuracy (Chapter 5.6.2) and recall (Chapter 0) along with runtime requirements. The following scenarios, summarized in Figure 59, were selected:

- **Burglary from Commercial properties in Burnaby (BCB)**

$\psi_{CRIME}=burglary, \psi_{LABEL}=Commercial, \psi_{CITY}=Burnaby$

The set of 2812 commercial properties were selected as the target entities with 33% of the properties having been burgled within the 5-year period, hence these were considered to have a positive class label.

- **Burglary from High-Density properties in Burnaby (BHB)**

$\psi_{CRIME}=burglary, \psi_{LABEL}=HighDensity, \psi_{CITY}=Burnaby$

The 1036 high-density (high-rise) properties within Burnaby were selected for learning rules with the 44.3% of properties which were burgled selected as the target.

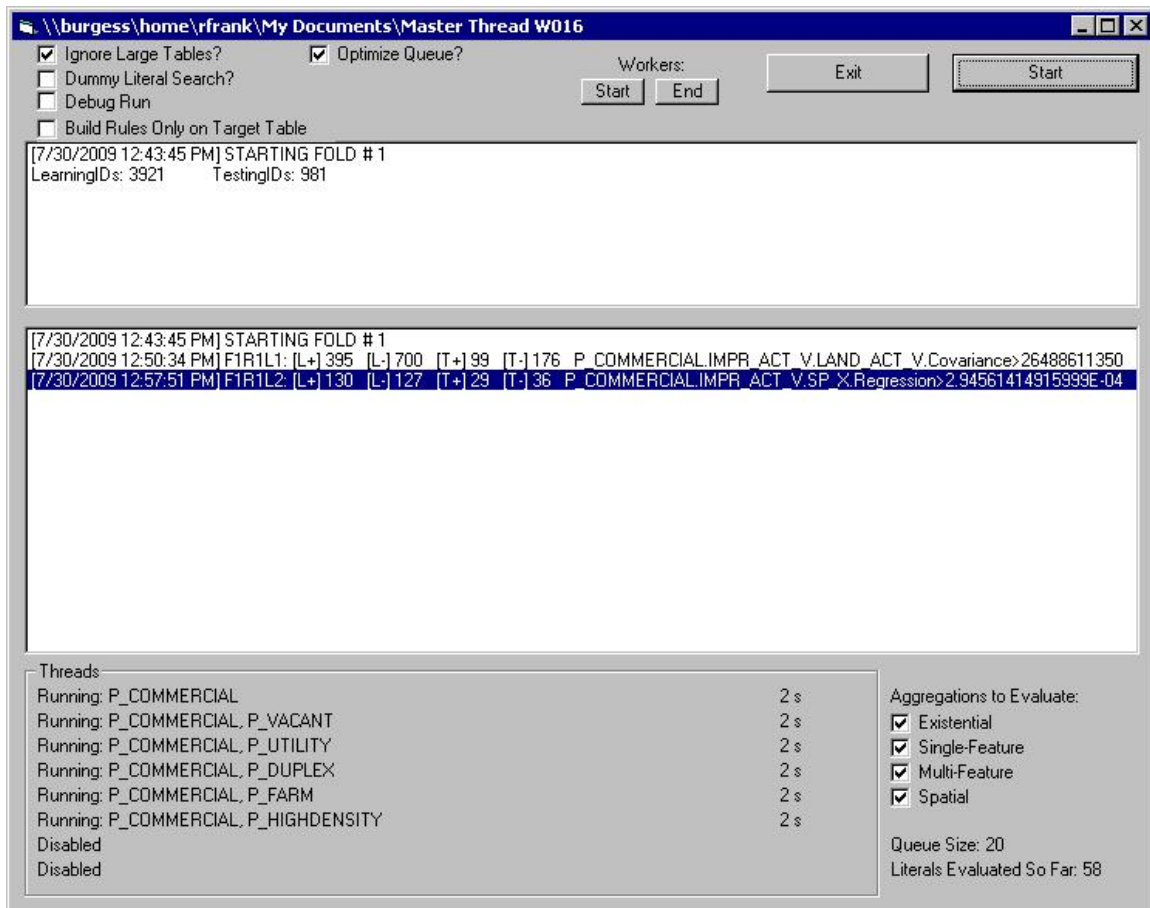


Figure 58 – UnMASC program running with 6 worker threads, evaluating all possible literal types.

- **Auto-theft from Commercial properties in Surrey (ACS)**

$$\psi_{CRIME}=autotheft, \psi_{LABEL}=Commercial, \psi_{CITY}=Surrey$$

The 4902 commercial properties in Surrey created a fairly skewed dataset, only 15.7% of the properties were involved in an auto-theft.

Hereafter the datasets will be referred to as *BCB*, *BHB* and *ACS* respectively. Experiments were run on an 8-CPU Windows server tied to a backend database with DB/2 v9.1 Spatial Extender. In experiments, where multiple instances of the *LiteralEvaluator* were run in parallel, the operating system was allowed to set the CPU-affinity of each instance, which meant no two instances were running on the same CPU simultaneously.

The three datasets, *BCB*, *BHB* and *ACS*, were used to evaluate the benefits of the contributions presented in this thesis. For each dataset, rules were learnt, and the results compared, using Voronoi-based neighbourhoods, buffer-zone neighbourhoods, and no neighbourhoods. The **Voronoi-based neighbourhood** was constructed according to the details presented in Chapter 3.2. For the **Buffer-zone neighbourhood**, instead of a fixed or manually selected buffer-zone size, the area of the buffer-zone for each entity-type was variable and given by  $area(\psi_{CITY})/Q;\pi$ . That is, the area of the buffer-zone was set to the average area available for each entity of type  $t_i$  assuming all entities of type  $t_i$  are uniformly distributed in city  $\psi_{CITY}$ . To highlight the importance of selecting the proper neighbourhood, rules were also learnt using **no neighbourhoods**, i.e. only the target entity-type was used for classification. This alternative uses only information on the target entity-type and since no neighbouring entity-types are evaluated, the search-space is much smaller.

Each neighbourhood definition created a very different number of relationships between the entities. For example, using the Voronoi-based neighbourhood definition, 4.6 million relationships are established in Surrey, whereas with the buffer-zone, there are 7.6 million relationships (Figure 59). Determining all entities which share a neighbourhood relationship is a

City	Neighbourhood Type	Entities				Relationships			
		Total	Average	Min	Max	Total	Average	Min	Max
Burnaby	None	66,260	2,209	1	66,260				
Burnaby	Voronoi-based	66,260	2,209	1	66,260	4,133,050	8,888	0	1,342,065
Burnaby	Buffer-zone	66,260	2,209	1	66,260	2,906,493	6,251	0	1,396,667
Surrey	None	130,993	4,517	1	88,995				
Surrey	Voronoi-based	130,993	4,517	1	88,995	4,645,816	10,680	0	653,839
Surrey	Buffer-zone	130,993	4,517	1	88,995	7,566,783	17,395	0	1,870,312

**Figure 59 – Statistics for database used for experiments**

lengthy process, and hence these relationships were pre-processed.

Interestingly, there is a significant difference between buffer-zones and Voronoi neighbourhoods in the number of neighbours each commercial property has. On average, for each commercial property, the number of neighbouring commercial properties using buffer-zones was 20 times that of Voronoi neighbourhoods, while the number of neighbouring non-commercial properties was only 0.4 times. This clearly indicates that buffer-zone rules were built more on neighbouring commercial properties and less on non-commercial properties, than the rules built on Voronoi neighbourhoods. Due to the use of zoning for city planning in Canada, commercial properties (as well as other property types) tend to be clustered, which the buffer-zone is unable to bypass and hence precision of the resulting rules suffers. This was also the reason for the large difference in run-times: when performing the aggregations, the buffer-zone had to aggregate much fewer values than the Voronoi approach, resulting in a reduction in runtime (and precision).

### **7.1.2 Experimental Results**

The results, shown in Figure 60 to Figure 62, and discussed in this chapter, focus on the following aspects of UnMASC:

- The effect of the neighbourhood definition (Chapter 7.1.3).
- The effect of adding multi-feature and spatial aggregation to the classification (Chapter 7.1.4).
- The effect of parallelization for the classification task (Chapter 7.1.5).

5-fold cross-validation was performed. Experiments using different combinations of aggregation operators were also performed: Existential (E), Single-Feature Aggregation (SFA), Multi-Feature Aggregation (MFA) and Spatial Aggregation (SA).

### **7.1.3 Evaluating the Effectiveness and Efficiency of the Voronoi Neighbourhood Definition**

As can be seen in the results of the *BCB* dataset, Figure 60, the precision and accuracy of using the buffer-zone neighbourhood relationship was very similar to the rules based on the target entity-type only. However, the recall was strangely lower with the buffer-zone neighbours than with using only the target entity-type. Using the Voronoi neighbourhood resulted in consistently higher precision and accuracy than both competing methods.

City Burnaby Entity Type Commercial Class Label Burglary  
 Class Label Distribution 931+ 1881-

Neighbourhood					Naïve		Optimized		
Definition	Aggregations	Precision	Recall	Accuracy	Serial	Parallel	Speed-Up	Parallel	Speed-Up
None	E	50.7%	70.1%	67.5%	1m				
	E SFA	50.5%	<b>72.0%</b>	67.4%	1m				
	E SFA MFA SA	50.8%	71.3%	67.6%	1h 14m				
Buffer	E	51.5%	66.3%	67.9%	11h 31m	4h 25m	2.60	3h 44m	<b>3.08</b>
	E SFA	50.6%	66.0%	67.4%	17h 09m	6h 48m	2.52	6h 08m	<b>2.79</b>
	E SFA MFA SA	51.8%	66.3%	68.3%	34h 36m	10h 37m	<b>3.26</b>	11h 35m	2.98
Voronoi	E	61.0%	67.5%	74.8%	27h 14m	6h 59m	3.89	6h 51m	<b>3.97</b>
	E SFA	61.3%	66.1%	74.9%	59h 30m	16h 31m	3.60	15h 14m	<b>3.90</b>
	E SFA MFA SA	<b>64.2%</b>	64.1%	<b>76.0%</b>	119h 34m	25h 12m	4.74	21h 32m	<b>5.55</b>

Figure 60 – Results for Burglary from Commercial properties in Burnaby (BCB)

City Burnaby Entity Type High Density Class Label Burglary  
 Class Label Distribution 459+ 577-

Neighbourhood					Naïve		Optimized		
Definition	Aggregations	Precision	Recall	Accuracy	Serial	Parallel	Speed-Up	Parallel	Speed-Up
None	E	65.0%	71.7%	70.4%	1m				
	E SFA	65.0%	71.7%	70.4%	1m				
	E SFA MFA SA	65.0%	71.7%	70.4%	1m				
Buffer	E	88.8%	78.0%	85.9%	4h 24m	4h 17m	1.02	1h 31m	<b>2.88</b>
	E SFA	88.9%	83.0%	87.8%	10h 45m	4h 37m	2.33	4h 16m	<b>2.51</b>
	E SFA MFA SA	89.1%	84.0%	88.4%	19h 17m	6h 37m	2.91	5h 37m	<b>3.43</b>
Voronoi	E	88.8%	80.7%	86.9%	14h 02m	3h 18m	4.24	3h 02m	<b>4.62</b>
	E SFA	89.1%	81.7%	87.4%	29h 10m	9h 17m	3.14	8h 07m	<b>3.59</b>
	E SFA MFA SA	<b>89.4%</b>	<b>86.2%</b>	<b>89.4%</b>	59h 24m	14h 25m	4.12	12h 25m	<b>4.78</b>

Figure 61 – Results for Burglary from High-density properties in Burnaby (BHB)

City Surrey Entity Type Commercial Class Label Auto Theft  
 Class Label Distribution 768+ 4134-

Neighbourhood					Naïve		Optimized		
Definition	Aggregations	Precision	Recall	Accuracy	Serial	Parallel	Speed-Up	Parallel	Speed-Up
None	E	36.1%	56.9%	77.5%	2h 13m				
	E SFA	34.7%	<b>62.6%</b>	75.7%	3h 12m				
	E SFA MFA SA	35.0%	61.6%	76.1%	4h 48m				
Buffer	E	38.8%	52.4%	79.4%	24h 58m	8h 34m	2.91	7h 51m	<b>3.18</b>
	E SFA	36.4%	57.3%	77.5%	34h 02m	10h 40m	3.19	9h 49m	<b>3.47</b>
	E SFA MFA SA	36.3%	58.8%	77.4%	57h 22m	14h 18m	4.01	13h 01m	<b>4.41</b>
Voronoi	E	42.4%	51.9%	81.3%	23h 55m	7h 17m	3.28	5h 53m	<b>4.05</b>
	E SFA	42.2%	58.5%	80.8%	58h 52m	11h 11m	<b>5.26</b>	11h 40m	5.04
	E SFA MFA SA	<b>43.6%</b>	58.0%	<b>81.5%</b>	164h 30m	31h 50m	5.17	29h 46m	<b>5.52</b>

Figure 62 – Results for Auto-theft from Commercial properties in Surrey (ACS)



For the *BHB* dataset, Figure 61, the results were different than was seen in *BCB*. With *BHB*, the rules from buffer-zones and Voronoi neighbourhood were consistently better than the rules based on the target entity-type only. The experiments using Voronoi neighbourhood did not yield the best recall (81.7% vs. 83.0%) or accuracy (87.4% vs. 87.8%) results in only 1 out of the 3 scenarios tested, clearly showing that that method is superior to the other two neighbourhood definitions.

For the last dataset, *ACS*, Figure 62, the results show that the precision and accuracy are clearly best with the Voronoi neighbourhood. The recall value however was a bit worse than the recall for the target entity-type only classifier. Again, the Voronoi neighbourhood improves the classification results.

In the case of no neighbourhood, UnMASC was very fast since rules were only built on the limited number of features of the target entity-type. The runtime requirement for the Voronoi method, as compared to the buffer-zone, was, on average, a factor of three higher, when the classifier rules were evaluated in serial fashion. They were a factor of 2.5 higher when the rules were evaluated in parallel. However, classifiers built on just the target entity-type usually had only a single rule with a few literals. For example, using Burglary in Burnaby as an example, the entire classifier based on only the target-entity-type contained the single rule:

R18: burglarized(H,'yes')  $\leftarrow$  highrise(H), building\_value(H,V),  $V > 3,707,000$

This rule states that expensive high-rises are more likely to be burglarized. Even though the recall for this rule is  $\sim 60\%$ , no further rules were ever generated given these parameters because no feature of high-rises provided a good feature for isolating the burglarized ones. This same rule was discovered even when aggregation features were allowed for rule-building.

Overall, the results indicate that using the Voronoi neighbourhood consistently yields the best precision and accuracy results, with recall suffering in a few cases.

#### 7.1.4 Evaluating Effectiveness of Aggregations

For each dataset and each neighbourhood relationship, three experiments were run in order to evaluate the difference between learning a classifier using:

- **E**: Only the Existential (E) Function
- **E-SFA**: Existential and Single-Feature Aggregation (SFA) Functions

- **E-SFA-MFA-SA:** Existential, Single-Feature Aggregation, Multi-Feature (MFA) and Spatial Aggregation (SA) Functions

It is expected that the larger the choice of features or aggregations, the better the classification rules. This turned out to be the case since, in general, all three datasets and all three neighbourhood definitions performed best when all the aggregations were analyzed. As seen in Figure 60-Figure 62, the combination of including all aggregations and using the Voronoi-based neighbourhood definition gave the best Precision, Recall and Accuracy in 7 of the 9 cases. This clearly indicates the superiority of including the additional aggregations.

The runtime however suffered due to the pair-wise feature analysis required both by multi-feature and spatial aggregations; in general, including all aggregations took about 1.5 times as long as the runtime of just the Existential and SFA functions.

As further consideration, since this is a spatial dataset, and there are spatial relationships between the entities, the meaning of the rules is much more appropriate as they are able to express spatial trends and other dependencies between the features. The rules produced by the *E* and *SFA* were not able to create rules with that type of meaning. For example, consider the rules shown in Figure 63. These three rules were found from the same dataset, same parameters ( $\psi_{CRIME}=burglary$ ,  $\psi_{LABEL}=Commercial$ ,  $\psi_{CITY}=Burnaby$ ), represented the same rule, and make an excellent basis for comparison. R19 was learnt using only the Existential aggregation operator, R20 was learnt using the Existential and Single-Feature Aggregation operators, while R21 was learnt using all available spatial and non-spatial aggregation operators.

R19 depends heavily on the distance between certain entity-types, but all of the comparison operators are *greater-than*, which weakens the rule because the fact that there exists a park further than 7.8km (L3, for example) does not mean that there is no park closer than 7.8kms. Had the comparison operator been the opposite (*less-than* 7.8km), it could have been concluded that somehow the presence of a park near-by influenced the burglary of commercial properties. In the present form, not much can be concluded.

R20 is a bit more meaningful. L1-L3 are the same as for R19, presenting no greater meaning than R19 did. L4 and L5 however include single-feature aggregation operators and represent more meaningful literals. According to L4, the average distance between the neighbouring parks and duplexes neighbouring those parks is more than 165m. Again, the *greater-than* comparison operator weakens the literal. L5 is better, it measures the building-value of neighbouring vacant properties to be less than \$46,100.

R21 was built with all available aggregation operators. As the first literal, a spatial trend is measured. The commercial properties which are south and relatively inexpensive, and get progressively expensive the further north they are located, have a higher chance of being burglarized. L2 expresses a relationship between neighbouring duplexes, about how the median distance to a duplex is less than 380m, implying that the commercial property is in a residential neighbourhood. L3 simply states that the neighbouring entertainment properties are to the east of 475,072 (*UTM* coordinates - the exact projection is *NAD 1983 UTM Zone 10 North*), while L4 detected some neighbouring malls about 13° east from north. Lastly, there were some industrial properties with an average building value of less than \$1,414,000.

Overall, Rules R19 and R20 are relatively weak because of their dependence mainly on the existential operator and the distance feature. R21 is able to express a much stronger concept, a spatial trend between south-north location and building-values. This allowed the rule to be more accurate with respect to the other rules: R21 covered 18 true positives and 9 false positives, while R19 and R20 covered 14 true positives and 10 false positives.

Although applying all of the aggregations yielded the best result in most cases, the difference

Rule	Literals in Rule
<b>Existential Aggregation</b>	
<i>Entity class labels covered by rule: (14+ 10-)</i>	
R19: burglarized(C, 'yes') ←	commercial(C), land_value(C,L), L>7,965,000 L1
	neighbour(C,H), hospital(H), distance(O,H,D1), D1 > 7,784.64 L2
	neighbour(C,P), park(P), distance(C, P, D2), D2 >7153 L3
	neighbour(P,O), other_type(O), distance(P,O,D3), D3 > 238.4 L4
	neighbour(C,M), motel(M), distance(C,M, D4), D4 > 170 L5
<b>Existential and Single-Feature Aggregation</b>	
<i>Entity class labels covered by rule: (14+ 10-)</i>	
R20: burglarized(C, 'yes') ←	commercial(C), land_value(C,L), L>7,965,000 L1
	neighbour(C,H), hospital(H), distance(O,H,D1), D1 > 7,784.64 L2
	neighbour(C,P), park(P), distance(C, P, D2), D2 >7153 L3
	AVG(D3, {distance(P,L,D3), neighbour(P,L), duplex(L)}, A), A>165.74 L4
	MAX(B, {neighbour(C,O), vacant(O), building_value(O,B)}, M), M<46100 L5
<b>Existential, Single-Feature, Multi-Feature and Spatial Aggregation</b>	
<i>Entity class labels covered by rule: (18+ 9-)</i>	
R21: burglarized(C, 'yes') ←	commercial(C), SP_TREND({B1, Y1}, {building_value(C,B1), SP_Y (C,Y1)}, T), T>66364303 L1
	MEDIAN(D1, {neighbour(C,D), duplex(L), distance(C,D,D1)}, M), M<380 L2
	neighbour(C,E), entertainment(E), SP_Y(E,Y2), Y2<475072 L3
	neighbour(C,M), mall(M), direction(C,M, R), R>3°, R<23° (° from north) L4
	AVG(B2, {neighbour(C, I), industrial(I), building_value(I, B2)}, A), A<1414000 L5

**Figure 63 – Comparison of rules found by different aggregation operators (Burglary for Commercial properties in Burnaby - Rule #2)**

between the “*E*”, “*E-SFA*” and “*E-SFA-MFA-SA*” scenarios were not as significant as expected. It is possible that during rule-building a local optima was encountered and the rule-building process got stuck in such an optima. Although there are techniques, such as randomized restart of searches [104], which are designed to get rule-builders out of local minima, these techniques were not explored for the purposes of this thesis.

### 7.1.5 Evaluating the Effectiveness of Parallelization

In order to evaluate the effectiveness of the parallel approach, three sets of experiments were carried out for each city and aggregation combination:

- UnMASC using only a **Serial** evaluation, where all candidate literals are evaluated sequentially.
- UnMASC using a **Naïve Parallel** technique, where the evaluation of the candidate literals is evaluated first-in-first-out (FIFO), with six worker threads evaluating the work simultaneously.
- UnMASC using an **Optimized Parallel** technique with six worker threads, where the optimization to the queue was carried out as described in Chapter 6.2.4.

In the three datasets, the serial evaluation quickly became prohibitively expensive, with it pushing 160h (~7days) for one of the runs. When selecting a new literal, there are a large number of candidates, evaluating the candidates in parallel is always faster than the serial evaluation. This is supported by the results: on average, the parallel method was 3.46x faster than the serial method.

If the number of candidates is greater than the number of CPUs, something that was always the case since there were 29 entity-types in the datasets but only six CPUs available, all candidates which cannot be evaluated initially were placed into a queue. It is possible that lengthy candidate evaluations are placed into the end of the queue, evaluated last and holding up the entire process. Thus, it is more efficient if those are moved to the front of the queue so they are evaluated first (see Chapter 6.2.4). This type of optimized parallelization was also evaluated, and improved the efficiency even further. Whereas the naïve parallelization was, on average, a factor of 3.46 faster than the serial evaluation, the optimized parallelization was faster by a factor of 3.88. In the experiments which had the longest runtime, the optimized parallelized method was far superior to the serial (a factor of 1.00) and the naïve parallel method (a factor of 4.68), with an average improvement of a factor of 5.29.

In certain cases, the speed-up provided by the parallelization was below a factor of 3. In these cases, there was a very large number of candidates which had to be evaluated, but each evaluation independently was relatively small. In these cases the benefit of the parallelization was dampened by the small overhead required by *RuleLearner*, the portion of UnMASC which managed the rules and the worker threads. In the most complex cases, ones involving all the aggregations, this overhead was relatively trivial, and the parallelization was closer to optimal.

The experimentation was carried out with six CPUs dedicated to evaluating literals. Results indicated that the speedup due to the use of using multiple CPUs varied between 42% and 92% of the theoretical optimal, with an average speedup of 65%. This finding was consistent with other small tests conducted with different numbers of CPUs.

Note that the rules are not guaranteed to be identical between the serial and two parallel methods. This is due to the reordering of candidate literal evaluations. It is possible that two candidate literals have the same FOIL-Gain, but are evaluated in a different order (due to the parallelization optimization). In this case whichever candidate literal was evaluated first will be added to the rule. Since in these cases, a different literal is added to the rule, further extensions of the rule can also be different, and so can subsequent rules. This situation was rare. The results in accuracy, precision and recall were very similar in these cases, and hence only the results with the optimized parallelized method were reported in Figure 60 - Figure 62.

## 7.2 Analysis of Classification Rules by Experts in Criminology

Rule	Literals in Rule	
R22: burglarized(H, 'yes') ←	highdensity(H),	L1
	neighbour(H,C), commercial(C), building_value(C, B), B<129,000	L2
	neighbour(C, S), school(S), direction(C, S, D1), D1 = 'NorthEast'	L3
	MAX(D2, {neighbour(S, I), industrial(I), distance(S, I, D2)}, M), M<735	L4
R23: burglarized(C, 'yes') ←	commercial(C),	L1
	MEDIAN(B <sub>1</sub> , {industrial(I), neighbour(C,I), building_value(I,B <sub>1</sub> )}, M), M<445,000	L2
	industrial(I), neighbour(C,I), TREND({B <sub>2</sub> , X}, {duplex(D), neighbour(I,D), building_value(D, B <sub>2</sub> ), x_coord(D,X)}, S), S>0.798	L3
	MAX(A, {park(P), neighbour(C,P), area(P,A)}, N), N<14,400	L4
R24: burglarized(H, 'yes') ←	highrise(H), building_value(H,V), V>2,160,000	L1
	AVG(D1, {commercial(C), neighbour(H,C), distance(H,C,D1)}, A), A<302	L2
	commercial(C), neighbour(H,C)	L3
	MEDIAN(W, {park(P), neighbour(C,P), width(P, W)}, M <sub>1</sub> ), M <sub>1</sub> <122	L4
	MEDIAN(X, {duplex(D), neighbour(C,D), land_value(D, X)}, M <sub>2</sub> ), M <sub>2</sub> >351,500	L5

Figure 64 – Rules discovered by UnMASC

Criminology is the scientific study of criminal behaviour. It is an interdisciplinary field, dealing with sociology, psychology and also law. There is a discipline within the school of Criminology called 'Environmental Criminology' [4]. It is not concerned with crimes against trees or other components of the environment, but is concerned about how the spatial neighbourhood (environment) shapes criminological activity. Two key people responsible for the development of Environmental Criminology are Drs. Paul and Patricia Brantingham [15]. This theory identifies time, law, offender, target/victim and space as the five components upon which a crime *can* occur. Without any of these components, no crime event can happen, because, for example, if something is not covered by law then it cannot be illegal and no crime has occurred. Environmental Criminology examines the space and time dimensions of a crime event, in order to determine what it is within the environment that was conducive to crime. A practical application of this subfield is 'crime prevention through environmental design' which attempts to shape the (urban) environment such that crime is prevented. For example, in order to decrease offenses neighbourhoods can be designed such that they form a community and the residents can observe the activities (and security) of the community.

Three classification rules, found by UnMASC, are shown in Figure 64. The below analysis is based on discussions carried out with experts in the field of Criminology: Dr. Patricia Brantingham and Dr. Bryan Kinney, both from SFU's School of Criminology.

Rule R22 was the first rule discovered using all aggregation operators for burglarized high-density properties within the city of Burnaby using the Voronoi diagram as the neighbourhood definition. The rule says that high-density buildings are burglarized if they are a neighbour to a commercial building with a building-value less than \$129,000 (L1), that there is a school north-east to the commercial building (L2), and the schools are at most 735m's away from an industrial property (L3). This rule is consistent with two theories from the Criminology domain. According to Crime Pattern Theory [15], each person (including offenders) will have an awareness and activity space, which is made up of the nodes, and paths between the nodes, that the person frequents due to everyday activities such as work or shopping. According to the theory, crime is most likely to occur where the activity space of both offenders and the victims/targets overlaps. The region described by the rule has at least four different land uses: high-density homes (the target entity), commercial, school and industrial. This indicates that this region probably sees heavy traffic from many different types of individuals, for example, industrial workers, shopkeepers, students and parents ferrying students to and from school. Although none of the types of people in this region imply a highly motivated criminological population, just the fact

that there is movement of a high volume of disparate people will increase the chances of a motivated offender entering the area. Hence, the activity spaces of many victims/targets will overlap with the activity space of a motivated offender, thus Crime Pattern Theory suggests that this area will see higher incidences of crime. Furthermore, the implications of this rule can also be described by another fundamental criminological theory, Routine Activities Theory [23]. This theory states that “direct-contact predatory violations”, intentionally damaging or taking a person or their property (for example burglary), have three properties:

- 1) motivated offenders,
- 2) suitable targets, and
- 3) absence of capable guardians.

High-density homes will contain many suitable targets (residences) which are unguarded (since the residents will be at work/school). Due to the many different types of individuals in the area, the chances of a motivated offender entering the region is higher than in areas where there is very little flow of people. Thus, this theory will also predict a higher incidence of crime. This rule was surprisingly accurate, described 133 (20%) of the burglarized high-rises and only 1 false positive.

R23 illustrates a rule that was found in the *BCB* dataset. It denotes that a commercial property (L1) is likely to be burglarized if the median building-value of neighbouring industrial properties is worth less than \$445,000 (L2) and has neighbouring parks with areas less than 14,400m<sup>2</sup> (L4). The commercial property also has neighbouring industrial properties with a spatial trend where duplexes are cheaper the further East they are, and more expensive the further West they are. This could indicate that the commercial property is in an inexpensive industrial neighbourhood, acting as an attractor to crime, while also a neighbour to parks which could be a source of criminals. This rule is interesting as it also matches up with awareness and activity spaces. Since awareness spaces of individuals (not just offenders) is most concentrated close to home, and offenders frequently live in lower socio-economic areas, the activities of offenders will likely concentrate on lower socio-economic areas. This trend was also found in criminological literature [83].

Another interesting rule involving high-rises, R24, was discovered. It states that a high-rise is likely to be burglarized if it is worth more than \$2.2 million (L1), the average distance to a commercial property is less than 302 m (L2) and, for neighbouring commercial properties (L3), neighbouring parks tend to be small (L4), and neighbouring duplexes tend to be worth at least \$351,500. This rule seems to indicate that expensive high-rises near high-traffic areas are going to see more burglaries as opposed to ones in areas without traffic.

## **CHAPTER EIGHT: CONCLUSIONS**

In this thesis a multi-relational Inductive Logic Programming-based approach to spatial classification was presented that posed novel challenges to which solutions were identified. A massive Crime Data-Warehouse, including 29 entity-types and their corresponding 435 types of relationships were assembled based on real-world city data with class labels built from real-world crime data provided by the RCMP.

The first challenge, due to the nature of spatial data, was that the neighbourhood relations are only implied through the location of the entities and not explicitly stated as in multi-relational data. In order to deal with this challenge, a Voronoi-diagram based neighbourhood definition was introduced to make explicit the spatial relationships. Both conceptually and experimentally it was demonstrated that this method of identifying neighbours and neighbourhoods is superior to the common distance-based buffer-zone approach.

The second challenge was that each of the resulting neighbourhood relationships have features, such as distance, direction and topological relationships, which are not explicitly stated in the database by default. These need to be calculated in order to allow the classifier to take advantage of them. Care has to be taken to select a subset of features to extract as not to overwhelm the classifier with redundant features. This thesis used a set of concise features which are independent. Also, in general, an entity in spatial data is related to many other entities. The importance of any single relationship, when in the presence of a large number of other such relationships, becomes insignificant. In order to overcome this challenge, this thesis introduces extensions into the Inductive Logic Programming framework to be able to capture not just individual spatial and non-spatial features, but aggregate properties of neighbourhood relationships and related entities. These additions clearly improved the classification precision, recall and accuracy, as demonstrated by the experiments presented in this thesis.

To perform the classification more efficiently, a scalable implementation was introduced, capable of evaluating multiple literal candidates simultaneously. This significantly decreased the runtime required for rule-learning. The strategy improved the classification performance by a factor of 5.5 on the most complex classification tasks, which was very close to the maximum given that there



were six CPUs dedicated to the rule-learning. The increase in performance was only limited by the number of CPUs available in the computer, had more computing power been available, the increase would have been higher.

Experiments on several real-world spatial datasets showed substantial gains in precision, recall and accuracy compared to existing approaches for neighbourhood definitions and aggregations. Anecdotal evidence was also provided by experts in the fields of Criminology to illustrate the meaningfulness of the rules discovered and the expressiveness of the language.

There are a few interesting directions this thesis can be extended in. First, it would be interesting to incorporate the temporal aspects into rule-learning to detect, for example, temporal trends and how changes over time in neighbourhood composition can lead to better predictions. This is doable with the current Crime Data-Warehouse, as the time each event happened is available. The ILP framework would however have to be extended in order to support temporal literals. UnMASC would also need to be extended since it can currently only compare dates against each other, but does not have the logic to temporally analyze them in the detail. As another direction, the focus of this thesis was not to contribute to the field of task-scheduling, hence the order each client was assigned work was rather naïve (i.e., largest work first), and optimizations could be introduced to make the ordering more efficient. To deal with the potential of encountering local minima, it would be interesting to evaluate the effectiveness of a randomized restart of the rule-builder in a parallel environment. Finally, the parallelization itself could be improved by looking for ways to break each unit of work into smaller pieces, perhaps where each unit of work consists of searching for the best candidate literal given an entity-type and a feature, not just an given an entity-type.

# APPENDICES

## Appendix A: Details of Literals

This chapter details the set of literals used in this thesis, their mathematical formula, when/if they are extracted from the database and the type of entity they apply to.

### A.1 Spatial Features

Spatial features are used to describe implicit features of entities (Chapter 4.3.2):

Spatial Features	Equation for entity q of type i	Types of Spatial Entity			DB2 Function	Pre-processing Cost		
		Point	Line	Poly		Cost for One Entity	Number of Entities	Total Cost
<b>length(input)</b> <b>N→S</b> length of the polygon representing entity <i>input</i>	$\text{MAX}_{p=1}^P (e_{p,y}^{i,q}) - \text{MIN}_{p=1}^P (e_{p,y}^{i,q})$		√	√	ST_MaxY()-ST_MinY()	O(p)	eβ	O(peβ)
<b>width(input)</b> <b>W→E</b> width of the polygon representing entity <i>input</i>	$\text{MAX}_{p=1}^P (e_{p,x}^{i,q}) - \text{MIN}_{p=1}^P (e_{p,x}^{i,q})$			√	ST_MaxX()-ST_MinX()	O(p)	eβ	O(peβ)
<b>perimeter(input)</b> perimeter of the spatial polygon representing entity <i>input</i>	$\sum_{p=1}^P \left( \sqrt{(e_{p+1,x}^{i,q} - e_{p,x}^{i,q})^2 + (e_{p+1,y}^{i,q} - e_{p,y}^{i,q})^2} \right)$			√	ST_Perimeter()	O(p)	eβ	O(peβ)

<b>area(input)</b> area of the spatial polygon representing entity <i>input</i>	$\frac{1}{2} \sum_{p=1}^P (e_{p,x}^{i,q} \times e_{p+1,y}^{i,q} - e_{p+1,x}^{i,q} \times e_{p,y}^{i,q})$			√	ST_Area()	O(p)	eβ	O(peβ)
<b>x_coord(input)</b> X-coordinate of the point entity	$e_{1,x}^{i,q}$	√			ST_X()	O(p)	eβ	O(peβ)
<b>y_coord(input)</b> Y-coordinate of the point entity	$e_{1,y}^{i,q}$	√			ST_Y()	O(p)	eβ	O(peβ)
<b>start_x(input)</b> starting X-coordinate of the <i>input</i> entity	$e_{1,x}^{i,q}$		√		ST_StartPoint()..ST_X()	O(p)	eβ	O(peβ)
<b>start_y(input)</b> starting Y-coordinate of the <i>input</i> entity	$e_{1,y}^{i,q}$		√		ST_StartPoint()..ST_Y()	O(p)	eβ	O(peβ)
<b>end_x(input)</b> ending X-coordinate of the <i>input</i> entity	$e_{P,y}^{i,q}$		√		ST_EndPoint()..ST_X()	O(p)	eβ	O(peβ)
<b>end_y(input)</b> ending Y-coordinate of the <i>input</i> entity	$e_{P,y}^{i,q}$		√		ST_EndPoint()..ST_Y()	O(p)	eβ	O(peβ)
<b>catchment_area(input)</b> area enclosed by the Voronoi cell of <i>input</i> , represents the size of region 'attracted' by the entity	<i>Voronoi Algorithm [see 8, 47, 65]</i>	√	√	√	<i>Voronoi Algorithm [see 8, 47, 65]</i>			$O(e \log e)$
<b>centroid_x(input)</b> X-coordinate of the centroid	$\sum_{p=1}^P \left( \frac{e_{p,x}^{i,q}}{P} \right)$		√	√	ST_Centroid()..ST_X()	O(p)	eβ	O(peβ)
<b>centroid_y(input)</b> Y-coordinate of the centroid	$\sum_{p=1}^P \left( \frac{e_{p,y}^{i,q}}{P} \right)$		√	√	ST_Centroid()..ST_Y()	O(p)	eβ	O(peβ)
<b>max_x(input)</b> maximum X-coordinate of all the vertices	$MAX_{p=1}^P (e_{p,x}^{i,q})$			√	ST_MaxX()	O(p)	eβ	O(peβ)

<b>min_x(input)</b> minimum X-coordinate of all the vertices	$\underset{p=1}{MIN} \left( e_{p,x}^{i,q} \right)$		√	ST_MinX()	O(p)	eβ	O(peβ)
<b>max_y(input)</b> maximum Y-coordinate of all the vertices	$\underset{p=1}{MAX} \left( e_{p,y}^{i,q} \right)$		√	ST_MaxY()	O(p)	eβ	O(peβ)
<b>min_y(input)</b> minimum Y-coordinate of all the vertices	$\underset{p=1}{MIN} \left( e_{p,y}^{i,q} \right)$		√	ST_MinY()	O(p)	eβ	O(peβ)

## A.2 Single-Feature Aggregation Literals

A single-feature aggregation function aggregates the values of a feature of multiple entities related to an entity (Chapter 4.2.2).

Single-feature Aggregation Literals	Equation for entity q, feature g, of type i	Run-time Cost			Applicable Feature Types			
		Cost for One Feature	Number of Features	Total Cost	Cat	Date	Num	Spat
<b>sum(input, {conditions}, result)</b> total of all <i>input</i> values	$\sum_{q=1}^Q g_q$	O(eβ)	f <sub>ns</sub>	O(f <sub>ns</sub> eβ)			√	√
<b>min(input, {conditions}, result)</b> smallest of all <i>input</i> values	$\underset{q=1}{MIN} g_q$	O(eβ)	f <sub>dns</sub>	O(f <sub>dns</sub> eβ)		√	√	√
<b>max(input, {conditions}, result)</b> largest of all <i>input</i> values	$\underset{q=1}{MAX} g_q$	O(eβ)	f <sub>dns</sub>	O(f <sub>dns</sub> eβ)		√	√	√
<b>avg(input, {conditions}, result)</b> average of all input values	$\bar{g} = \sum_{q=1}^Q \frac{g_q}{Q}$	O(eβ)	f <sub>ns</sub>	O(f <sub>ns</sub> eβ)		√	√	√
<b>count(input, {conditions}, result)</b> number of input values	v	O(eβ)	f <sub>cdns</sub>	O(f <sub>cdns</sub> eβ)	√	√	√	√
<b>range(input, {conditions}, result)</b> difference between max and min	$\underset{q=1}{MAX} g_q - \underset{q=1}{MIN} g_q$	O(eβ)	f <sub>dns</sub>	O(f <sub>dns</sub> eβ)		√	√	√

<b>standard deviation</b> <i>(input, {conditions}, result)</i> dispersion or variation in a distribution of the given input values	$\sum_{q=1}^Q (g_q - \bar{g})^2$	O(2eβ)	f <sub>dns</sub>	O(2f <sub>dns</sub> eβ)		√	√	√
<b>least_frequent</b> <i>(input, {conditions}, result)</i> value which is repeated the fewest number of times	$\arg \min_{q=1}^Q (freq(g_q))$	O(eβ)	f <sub>c</sub>	O(f <sub>c</sub> eβ)	√			
<b>most_frequent</b> <i>(input, {conditions}, result)</i> value which is repeated the most number of times	$\arg \max_{q=1}^Q (freq(g_q))$	O(eβ)	f <sub>c</sub>	O(f <sub>c</sub> eβ)	√			

**Cat** = Categorical feature type

**Num** = Numerical feature type

**Spat** = Spatial feature type

### A.3 Multi-Feature Aggregation Literals

A multi-feature aggregation function aggregates the values of multiple feature of multiple entities related to an entity (Chapter 4.3.1).

Multi-Feature Aggregation Literals	Equation for entity q, features f and g, of type i	Run-time Cost			Applicable Feature Types			
		Cost for One Feature	Number of Permutations of Features	Total Cost	Cat	Date	Num	Spat
<b>linear_regression_coefficient</b> <i>({input<sub>1</sub>, input<sub>2</sub>} {conditions}, result)</i> slope of the line-of-best-fit, indicates relationship between two <i>inputs</i>	$\sum_{q=1}^Q (g_q - \bar{g})(f_q - \bar{f}) / \sum_{q=1}^Q (g_q - \bar{g})^2$	O(2eβ)	f <sub>dns</sub> (f <sub>dns</sub> -1)	O(f <sub>dns</sub> <sup>2</sup> eβ)		√	√	√
<b>correlation</b> <i>({input<sub>1</sub>, input<sub>2</sub>} {conditions}, result)</i> degree to which the values are positively/negatively associated to each other	$\frac{\overline{g \times f} - \bar{g} \times \bar{f}}{\sqrt{g^2 - (\bar{g})^2} \sqrt{f^2 - (\bar{f})^2}}$	O(eβ)	f <sub>dns</sub> (f <sub>dns</sub> -1)	O(f <sub>dns</sub> <sup>2</sup> eβ)		√	√	√

<b>covariance</b> <i>({input<sub>1</sub>, input<sub>2</sub>}, {conditions}, result)</i> measure which indicates how two values vary together	$\overline{g \times f} - \overline{g} \times \overline{f}$	O(eβ)	f <sub>dns</sub> (f <sub>dns</sub> -1)	O(f <sub>dns</sub> <sup>2</sup> eβ)		√	√	√
<b>most_frequent_combination</b> <i>({input<sub>1</sub>, input<sub>2</sub>, ..., input<sub>i</sub>}, {conditions}, result)</i> value which is repeated the fewest number of times	$\arg \max_{q=1}^Q (freq[f_q, g_q])$	O(eβ)	f <sub>dnsc</sub> (f <sub>dnsc</sub> -1)	O(f <sub>dnsc</sub> <sup>2</sup> eβ)	√	√	√	√
<b>least_frequent_combination</b> <i>({input<sub>1</sub>, input<sub>2</sub>, ..., input<sub>i</sub>}, {conditions}, result)</i> value which is repeated the most number of times	$\arg \min_{q=1}^Q (freq[f_q, g_q])$	O(eβ)	f <sub>dnsc</sub> (f <sub>dnsc</sub> -1)	O(f <sub>dnsc</sub> <sup>2</sup> eβ)	√	√	√	√
<b>chi-square</b> <i>({input<sub>1</sub>, input<sub>2</sub>, ..., input<sub>i</sub>}, {conditions}, result)</i> indicates whether there is a dependency between two sets of categorical values	$\sum_{q=1}^Q \frac{(g_q f_q - \overline{g f})^2}{\overline{g f}}$	O(2eβ)	f <sub>c</sub> (f <sub>c</sub> -1)	O(f <sub>c</sub> <sup>2</sup> 2eβ)	√			


**Cat** = Categorical feature type








**Num** = Numerical feature type

**Spat** = Spatial feature type

## A.4 Spatial Literals

Spatial literals are used to describe implicit relationships between multiple entities (Chapter 4.3.3):

Spatial Literals (neighbourhood relationships)	Depiction	DB2 Function	Pre-processing Costs		
			Cost of One Operation	Number of Operations	Total Cost
<b>disjoint</b> ( <i>input<sub>1</sub>, input<sub>2</sub></i> ) true if <i>input<sub>1</sub></i> is completely separate from <i>input<sub>2</sub></i>		ST_Disjoint(e <sup>i,k</sup> , e <sup>i,j</sup> )	O(s) *	eβ	O(eβs)

<b>meet</b> ( $input_1, input_2$ )		$ST\_Touches(e^{i,k}, e^{i,j})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
if $input_1$ is strictly immediately adjacent to $input_2$					
<b>overlap</b> ( $input_1, input_2$ )		$ST\_Crosses(e^{i,k}, e^{i,j})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
if $input_1$ extends over, or partially covers, $input_2$					
<b>covered_by</b> ( $input_1, input_2$ )		$ST\_Contains(e^{i,k}, e^{i,j})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
$input_1$ completely envelops $input_2$					
<b>inside</b> ( $input_1, input_2$ )		$ST\_Within(e^{i,j}, e^{i,k})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
if $input_1$ is within $input_2$ (a building in a city or park)					
<b>equal</b> ( $input_1, input_2$ )		$ST\_Equals(e^{i,k}, e^{i,j})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
if $input_1$ is identical and in the same location as $input_2$					
<b>covers</b> ( $input_1, input_2$ )		$ST\_Contains(e^{i,j}, e^{i,k})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
if $input_2$ completely envelops $input_1$					
<b>contains</b> ( $input_1, input_2$ )		$ST\_Within(e^{i,k}, e^{i,j})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
if $input_2$ is within $input_1$					
<b>direction</b> ( $input_1, input_2$ )	$\text{mod}[\text{int}\{450 - \text{atan2}(\Delta x, \Delta y) \times \frac{180}{\pi}\}, 360]$	$RF\_Direction(e^{i,k}, e^{i,j})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
specifies the angular position of $input_2$ relative to $input_1$ as a number or value such as 'East'					
<b>distance</b> ( $input_1, input_2$ )	$\sqrt{(e_{1,x}^{t-1,q'} - e_{1,x}^{t,q})^2 + (e_{1,y}^{t-1,q'} - e_{1,y}^{t,q})^2}$	$ST\_Distance(e^{i,k}, e^{i,j})$	$O(s)^*$	$e\beta$	$O(e\beta s)$
distance between $input_1$ and $input_2$ when travelling in a straight line					
<b>voronoi_neighbour</b> ( $input_1, input_2$ )			$O(s)^*$	$e \log e$	$O(s \times e \log e)$
if $input_1$ shares a relationship, according to the Delaunay triangulation with $input_2$					
<b>road_distance</b> ( $input_1, input_2$ )			$O(1)^\wedge$	$e\beta$	$O(e\beta)$
distance between $input_1$ and $input_2$ when					

travelling along the road-network of a map					
<b>travel_time</b> ( <i>input</i> <sub>1</sub> , <i>input</i> <sub>2</sub> )					
time required in order to cover the distance from <i>input</i> <sub>1</sub> to <i>input</i> <sub>2</sub> along the road network					

O(1) ^

eβ

O(eβ)

\* Feature extracted via composite spatial SQL functions performed inside DB2 Spatial Extender, runtime is assumed to be *s*

^ The distances between each pair of intersections (of roads) are calculated within ArcGIS and then saved into a DB2 table.

To calculate distances between two entities, a custom-written DB2 function RoadDistance(A,B) is used [see Chapter C.1 for pseudo-code].

## A.5 Spatial Aggregation Literals

Spatial aggregation literals are used to describe spatial properties of the neighbourhood of an entity (Chapter 4.3.4):

Spatial Aggregation Literal	Evaluating feature <i>g</i>	Run-Time Cost		
		Cost for one Feature	Number of Permutations of Features	Total Cost
<b>spatial_trend</b> ( <i>{input</i> <sub>1</sub> , <i>input</i> <sub>2</sub> <i>{conditions}</i> }, <i>result</i> )  indicates a tendency for a feature-value of <i>input</i> <sub>2</sub> to change with respect to distance from <i>input</i> <sub>1</sub> equivalent to correlation between feature-value and distance - formula requires 2 passes of data, has to be done for each feature	$\frac{\overline{gd} - \bar{g} \times \bar{d}}{\sqrt{g^2 - \bar{g}^2} \sqrt{d^2 - \bar{d}^2}}$	O(eβ)	f <sub>dns</sub>	O(f <sub>dns</sub> eβ)
<b>spatial_autocorrelation</b> ( <i>{input</i> <sub>1</sub> , <i>input</i> <sub>2</sub> <i>{conditions}</i> }, <i>result</i> )  analyze the degree of dependency among observations of <i>input</i> <sub>1</sub> and <i>input</i> <sub>2</sub> with respect to their location - to calculate average requires the first pass. to calculate the double-sigmas requires n2.	$\frac{ Q  \sum_{j=1}^Q \sum_{k=1}^Q [d_{j,k} (g_j - \bar{g})(g_k - \bar{g})]}{\sum_{j=1}^Q \sum_{k=1}^Q d_{j,k} \times \sum_{j=1}^Q (g_j - \bar{g})^2}$	O(2eβ <sup>2</sup> )	f <sub>dns</sub>	O(f <sub>dns</sub> 2eβ <sup>2</sup> )
<b>area_adjusted_mean</b> ( <i>{input</i> <sub>1</sub> , <i>input</i> <sub>2</sub> <i>{conditions}</i> }, <i>result</i> )  mean value of all <i>input</i> <sub>2</sub> entities, weighted by their Voronoi area, that are neighbours to <i>input</i> <sub>1</sub>	$\frac{\bar{g}}{\text{voronoi\_area}(e^{i,q})}$	O(β × e log e)	f <sub>dns</sub>	O(f <sub>dns</sub> β × e log e)



## Appendix B: Summary of Datasets in the Crime Data-Warehouse

Entity	Description	Source	# of Entities in Surrey, BC	# of Entities In Burnaby, BC	Type of Data
Airports	The set of airports	A subset of the BCAA dataset	0	2	Point
Ambulance	The set of locations from where ambulances leave	A subset of the BCAA dataset	4	1	Point
Bank	The set of banks	A subset of the BCAA dataset	15	18	Point
Civic	The set of government buildings	A subset of the BCAA dataset	192	30	Point
Commercial	The set of commercial properties, such as stores, does not include grocery stores, nor collection of stores such as malls.	A subset of the BCAA dataset	4902	2812	Point
Condominium	The set of condominiums. Each individual condo has a separate entry. Average values, for example, can be calculated by averaging each condo contained in a complex.	A subset of the BCAA dataset	11393	15653	Point
Correctional	The set of correctional institutions.	A subset of the BCAA dataset	1	6	Point
Duplex	The set of duplex houses.	A subset of the BCAA dataset	1334	2654	Point
Entertainment	The set of movie theatres, night clubs, etc.	A subset of the BCAA dataset	82	58	Point
Farm	The set of farms.	A subset of the BCAA dataset	1149	71	Point
Fire	The set of fire-fighter stations.	A subset of the BCAA dataset	17	6	Point
Gas station	The set of gas-stations.	A subset of the BCAA dataset	80	43	Point
Grocery	The set of grocery-stores. Does not include other types of commercial properties.	A subset of the BCAA dataset	11	3	Point
High-Density Housing	The set of high-rise buildings. These are different from, and hence do not contain, condominiums.	A subset of the BCAA dataset	1073	1036	Point
Hospital	The set of hospitals.	A subset of the BCAA dataset	2	10	Point
Industrial	The set of industrial locations, such as factories.	A subset of the BCAA dataset	789	453	Point
Lines	The set of rail-roads, transmission lines, etc.	GIS Innovations – 2007	279	85	Line

Low-Density Housing	The set of low-density (simplex) housing.	A subset of the BCAA dataset	88995	34284	Point
Mall	The set of malls.	A subset of the BCAA dataset	53	40	Point
Motel	The set of motels/hotels within the province.	A subset of the BCAA dataset	61	20	Point
Parks	The set of parks within BC.	GIS Innovations – 2007	206	157	Polygon
Police	The set of police-stations.	A subset of the BCAA dataset	7	2	Point
Post-Secondary Schools	The set of schools classified as post-secondary schools. Does not contain schools not classified as post-secondary.	A subset of the BCAA dataset	5	4	Point
Recreation	The set of recreational facilities, public swimming-pools for example.	A subset of the BCAA dataset	15	3	Point
Road	The set of roads for BC. Each tuple represents a road-segment, and can be joined up with other road-segments via a spatial join.	GIS Innovations – 2007	12951	5253	Line
School	The set of schools classified as elementary- or high-schools. Does not contain post-secondary schools.	A subset of the BCAA dataset	157	65	Point
Transit	The set of sky-train stops.	A subset of the BCAA dataset	11	12	Point
Utility	The set of public utility (hydro-stations, dams, etc).	A subset of the BCAA dataset	2894	2721	Point
Vacant	The set of vacant plots of land.	A subset of the BCAA dataset	3610	442	Point

**Figure 65 – Description of datasets used for experimentation. Dataset is restricted to Burnaby and Surrey, BC.**

## Appendix C: Pseudo-code

### C.1 Road-Distance Function

Function RoadDistance(point A, point B)

RoadSegmentA  $\leftarrow$  Road segment point A is on

RoadSegmentAS  $\leftarrow$  Starting point of road segment A

RoadSegmentAE  $\leftarrow$  Ending point of road segment A

// Distance from point A to the starting-point of road-segment A

RoadSegmentASDistance  $\leftarrow$  Distance(A, RoadSegmentAS)

// Distance from point A to the ending-point of road-segment A

RoadSegmentAEDistance  $\leftarrow$  Distance(A, RoadSegmentAE)

RoadSegmentB  $\leftarrow$  Road segment point B is on

RoadSegmentBS  $\leftarrow$  Starting point of road segment B

RoadSegmentBE  $\leftarrow$  Ending point of road segment B

// Distance from point B to the starting-point of road-segment B

RoadSegmentBSDistance  $\leftarrow$  Distance(B, RoadSegmentBS)

// Distance from point B to the ending-point of road-segment B

RoadSegmentBEDistance  $\leftarrow$  Distance(B, RoadSegmentBE)

// calculate the 4 possible distanced between two pairs of intersections:

// {AS-BS, AE-BS, AS-BE, AE-BE}, as precomputed by ArcGIS

Distance\_AS\_BS  $\leftarrow$  Distance(RoadSegmentAS, RoadSegmentBS)

Distance\_AE\_BS  $\leftarrow$  Distance(RoadSegmentAE, RoadSegmentBS)

Distance\_AS\_BE  $\leftarrow$  Distance(RoadSegmentAS, RoadSegmentBE)

Distance\_AE\_BE  $\leftarrow$  Distance(RoadSegmentAE, RoadSegmentBE)

RoadDistance  $\leftarrow$  MIN(

Distance\_AS\_BS + RoadSegmentASDistance + RoadSegmentBSDistance,

Distance\_AE\_BS + RoadSegmentAEDistance + RoadSegmentBSDistance,

Distance\_AS\_BE + RoadSegmentASDistance + RoadSegmentBEDistance,

Distance\_AE\_BE + RoadSegmentAEDistance + RoadSegmentBEDistance

)

End Function

### C.2 Angle Function

The below function calculates the direction two entities are with relation to each other. The code was used in DB2, as a user-defined function.

```
CREATE FUNCTION RFRANK.RF_DIRECTION(X DOUBLE, Y DOUBLE)
  RETURNS DECIMAL(20,10)
```

```

LANGUAGE SQL
DETERMINISTIC
READS SQL DATA

BEGIN ATOMIC
  DECLARE ANGLE DECIMAL(20,10);
  DECLARE X1 DECIMAL(20,10);
  DECLARE Y1 DECIMAL(20,10);

  SET X1 = CAST(X as DECIMAL(20,10));
  SET Y1 = CAST(Y as DECIMAL(20,10));

  IF X1 = Y1 THEN
    SET ANGLE = NULL;
  ELSE
    SET ANGLE = MOD(INT(450-ATAN2(CAST(X as DECIMAL(20,10)),
      CAST(Y as DECIMAL(20,10))))*180/3.1415926535, 360);
  END IF;

RETURN ANGLE;
END

```

## REFERENCES

- 1) Adam, N. R.; Janeja, V. P.; Atluri, V.: “*Neighborhood based detection of anomalies in high dimensional spatio-temporal sensor datasets*”, In Proceedings of the ACM symposium on Applied computing (2004)
- 2) Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: “*Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications*”, In Proceedings of SIGMOD (1998)
- 3) Agrawal, R., Imielinski, T., and Swami, A.: “*Mining association rules between sets of items in large databases.*” In Proceedings of SIGMOD, pp 207-216 (1993)
- 4) Andresen, M.A.: “*The place of environmental criminology within criminological thought.*” In M.A. Andresen, P.J. Brantingham, and J.B. Kinney (eds.) *Classics in Environmental Criminology*. Co-published: Burnaby, BC, SFU Publications and Boca Raton, FL, Taylor & Francis (2010 – forthcoming)
- 5) Appice, A.; Ceci, M.; Lanza, A.: “*Discovery of spatial association rules in geo-referenced census data: a relational mining approach*”. In Proceedings of Intelligent Data Analysis (2003)
- 6) Apt., K.R., Bol, R.: “*Logic Programming and Negation: A Survey*”, Journal of Logic Programming, No. 19-20, pp 9-71 (1994)
- 7) Atramentov, A., Leiva, H., Honavar, V.: “*A multi-relational decision tree learning algorithm -implementation and experiments.*” In ILP Volume 2835 of LNAI, Springer-Verlag, pp 38-56 (2003)
- 8) Aurenhammer, F.: “*Voronoi diagrams - a survey of a fundamental geometric data structure*”. In ACM Computing Surveys, Volume 23 Issue 1, pp 345-405 (1991)
- 9) Barnett, V. and Lewis, T.: “*Outliers in Statistical Data*”, John Wiley & Sons, New York. (1984)
- 10) Bembenik, R. and Rybinski, H.: “*Mining Spatial Association Rules with no Distance Parameter*”, Advances in Soft Computing, Vol. 35 (2006)
- 11) Bereg, S.: “*Recent Developments and Open Problems in Voronoi Diagrams.*” In Proceedings of 3rd International Symposium on Voronoi Diagrams in Science and Engineering (2006)
- 12) Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G. and Vandecasteele, H.: “*Improving the efficiency of inductive logic programming through the use of query packs*”, Journal of Artificial Intelligence Research, vol16, pp 135-166. (2002)

- 13) Blockeel, H., De Raedt, L., and Ramon, J.: “*Top-down induction of clustering trees*” in Proceedings of the 15<sup>th</sup> International Conference on Machine Learning, pp. 55-63.
- 14) Bohm, C., Kailing, K., Kroger, P., and Zimek, A.: “*Computing Clusters of Correlation Connected Entities*”. In Proceedings of SIGMOD, pp 455-466 (2004)
- 15) Brantingham, P.J, Brantingham, P.L.: “*Environmental Criminology*”, Prospect Heights, IL: Waveland Press. (1990)
- 16) Brecheisen, S., Kriegel, H-P., Kroger, P., Pfeife, M.: “*Visually Mining Through Cluster Hierarchies*”, In Proceedings of SIAM, pp 400-412 (2004)
- 17) Ceci, M.; Appice, A.; Malerba, D.: “*Spatial Associative Classification at Different Levels of Granularity: A Probabilistic Approach*”. Lecture Notes in Computer Science, Volume 3202, pp 99 (2004)
- 18) Ceri, S; Gottlob, G; Tanca, L.: “*Logic programming and databases*”. Springer-Verlag, c1990, ISBN: 3-54051-728-6
- 19) Challis, R. E., and Kitney, R. I.: “*Biomedical signal processing (in four parts). Part I Time-domain methods.*” Medical & Biological Engineering & Computing, Volume 28, pp 509-524 (1991)
- 20) Chaudhuri, S., Datar, M., Motwani, R., and Narasayya, V.: “*Overcoming Limitations of Sampling for Aggregation Queries*”, In Proceedings of ICDE (2001)
- 21) Chelghoum, N., Zeitouni, K.: “*Spatial Data Mining Implementation - Alternatives And Performances*”, In Proceedings of GEOINFO (2004)
- 22) Chelghoum, N.; Zeitouni, K.; Boulmakoul, A.: “*A decision tree for multi-layered spatial data*”. In Proceedings of Symposium on Geospatial Theory – Processing Applications (2002)
- 23) Cohen, L.E., Felson, M.: “*Social Change and Crime Rate Trends: A Routine Activity Approach*”, American Sociological Review, Vol. 44, No. 4. pp 588-608. (1979)
- 24) Cressie, N.A.C.: “*Statistics for spatial data*”, Published: New York; J. Wiley, ISBN 0471843369 (1991)
- 25) Dehaspe, L., and Toivonen, H.: “*Discovery of frequent datalog patterns.*” Data Mining and Knowledge Discovery, Vol 3-1, pp 7-36.
- 26) Dzeroski, S.: “*Multi-relational data mining: an introduction*”. In SIGKDD Explorations, Volume 5, pp 1–16 (2003)
- 27) Dzeroski, S; Lavrac, N.: “*Relational Data Mining*”. Springer, c2001, ISBN: 3-54042-289-7

- 28) Egenhofer, M.J.: “*Reasoning about Binary Topological Relations*” In Proceedings of the Second Symposium on Large Spatial Databases, pp143–160 (1991)
- 29) Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: “*Clustering for Mining in Large Spatial Databases*”. KI-Journal, ScienTec Publishing, Vol. 1 (1998)
- 30) Ester M., Kriegel H.-P., Sander J., Xu X.: “*A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*”, In Proceedings of KDD (1996)
- 31) Ester, M.; Kriegel, H.P.; Sander, J.: “*Spatial Data Mining: A Database Approach*”. In Proceedings of SSD (1997)
- 32) Ester, M.; Kriegel, H-P.; Sander, J.: “*Knowledge discovery in spatial databases*”. 23<sup>rd</sup> Annual German Conference on Artificial Intelligence, pp 61-74 (1999)
- 33) Ester, M., Frommelt, A., Kriegel, H.P., and Sander, J.: “*Algorithms for Characterization and Trend Detection in Spatial Databases*” In Proceedings of KDD, pp 44-50 (1998)
- 34) Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: “*Knowledge discovery and data mining toward a unifying framework.*” In Proceeding KDD, pp 82-88 (1996)
- 35) Fertig, C., Freitas, A. A., Arruda, L. V. R., Kaestner C.: “*A Fuzzy Beam-Search Rule Induction Algorithm*” In Proceedings of PKDD Lecture Notes in Artificial Intelligence 1704 (1999)
- 36) Frank, E., and Witten, I.H.: “*Selecting multiway splits in decision trees.*” Technical Report 96/31, University of Waikato, Department of Computer Science, Hamilton, New Zealand (1996)
- 37) Frank, R., Ester, M., and Knobbe, A.: “*A Multi-Relational Approach to Spatial Classification*”, In Proceedings of KDD (2009)
- 38) Frank, R., Moser, F., Ester, M.: “*A Method for Multi-Relational Classification Using Single and Multi-Feature Aggregation Functions*”, In Proceedings of PKDD (2007)
- 39) Frank, R., Jin, W., Ester, M.: “*Efficiently Mining Regional Outliers in Spatial Data*”, Proc. 10th International Symposium on Spatial and Temporal Databases (2007)
- 40) Frawley, W., Piatetsky-Shapiro, G., Matheus, C.: “*Knowledge Discovery in Databases: An Overview*”. AI Magazine, pp 213-228 (Fall 1992)
- 41) Freitas, A.A.: “*On rule interestingness measures.*” Knowledge-Based Systems, Volume 12, pp 309-315 (1999)

- 42) Giannotti, F.; Manco, G.; Turini, F.: “*Specifying Mining Algorithms with Iterative User-Defined Aggregates*” In IEEE Transactions on Knowledge and Data Engineering, Volume 16, no. 10, pp. 1232-1246, (2004)
- 43) Graf, M., Winter, S.: “*Netzwerk-Voronoi-Diagramme*”. Österreichische Zeitschrift für Vermessung und Geoinformation, Volume 91, No. 3, pp 166-174 (2003)<sup>2</sup>
- 44) Han, J., Kamber, M., Tung, A. K. H.: “*Spatial clustering methods in data mining: A survey*”. In H. Miller and J. Han, editors, *Geographic Data Mining and Knowledge Discovery*. ISBN 0-20346-802-3 (2001)
- 45) Han, J., Kamber, M.: “*Data Mining: Concepts and Techniques, 2nd ed.*”, The Morgan Kaufmann Series in Data Management Systems. ISBN 1-55860-901-6 (2006)
- 46) Heywood, D. I., Cornelius, S., Carver, S.: “*An Introduction to Geographical Information Systems US Edition (Co-Pub)*”, Prentice Hall Series in Geographic Information Science. ISBN 0-13016-238-8 (1998)
- 47) Hoff, K.; Culver, T.; Keyser, J.; Lin, M.; Manocha, D.: “*Fast computation of generalized voronoi diagrams using graphics hardware*”. In Proceedings of SIGGRAPH (1999)
- 48) Huang, Y., Shekhar, S., Xiong, H.: “*Discovering Co-location Patterns from Spatial Datasets: A General Approach*”, In Proceedings of TKDE, Volume 16, No. 12, pp 1472-1485 (2004)
- 49) “IBM DB2 Spatial Extender – User’s Guide and Reference”, IBM (2002).
- 50) Keim, D., Wawryniuk, M.: “*Identifying Most Predictive Items*”, Workshop on Pattern Representation and Management (2004)
- 51) Klosgen, W., May, W.: “*Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database*”, Lecture Notes in Computer Science, vol2431 (2002)
- 52) Knobbe, A.J.; Siebes, A.; Marseille, B.: “*Involving Aggregate Functions in Multi-Relational Search*”. In Proceedings of Principles of Data Mining and Knowledge Discovery (2002)
- 53) Koperski, K.; Han, J.; Stefanovic, N.: “*An Efficient Two-Step Method for Classification of Spatial Data*”. In Proceedings of SDH (1998)
- 54) Koperski, K., Han, J.: “*Discovery of Spatial Association Rules in Geographic Information Databases*”. In Proceedings of SSD, pp 47-66 (1995)

---

<sup>2</sup> “Network Voronoi Diagrams”, english translation available at [www.sli.unimelb.edu.au/winter/pub.htm](http://www.sli.unimelb.edu.au/winter/pub.htm)



- 55) Koperski, K.: “*A Progressive Refinement Approach to Spatial Data Mining*”, PhD Thesis, Simon Fraser University (April 1999)
- 56) Kovalerchuk, B.; Vityaev, E.; Yusufov, H.: “*Symbolic Methodology in Numeric Data Mining: Relational Techniques for Financial Applications*”. In CoRR: Computational Engineering, Finance, and Science (2002)
- 57) Krogel, M. A., Rawles, S., Železný, F., Flach, P. A., Lavrač, N., Wrobel, S.: “*Comparative evaluation of approaches to propositionalization*”. In: Proc. 13th Internat. Conference on Inductive Logic Programming. Springer–Verlag, Berlin 2003
- 58) Krogel, M. A. and Wrobel, S.: “*Transformation-Based Learning Using Multirelational Aggregation*”. In Proceedings of the 11th international Conference on inductive Logic Programming. C. Rouveirol and M. Sebag, Eds. Lecture Notes In Computer Science, vol. 2157. Springer-Verlag, London, 142-155. (2001)
- 59) Laffan, S.W.; Nielsen, O.M.; Silcock, H.; Hegland, M.: “*Sparse Grids: a new predictive modelling method for the analysis of geographic data.*” International Journal of Geographical Information Science, Vol. 19, No. 3, pp 267-292, (2005)
- 60) Lavrac, N., Dzeroski, S.: “*Inductive Logic Programming: Techniques and Applications.*” Ellis Horwood, New York (1994)
- 61) Lavrac, N., Flach, P., Zupan, B.: “*Rule Evaluation Measures: A Unifying View*”, In Proceedings of ILP (1999)
- 62) Lavrac, N., Zelezny, F., and Flach, P. A.: “*RSD: Relational subgroup discovery through first-order feature construction.*” In S. Matwin and C. Sammut, editors, Proceedings of the Twelfth International Conference on Inductive Logic Programming (ILP), pages 149-165. Springer, 2002
- 63) Longley, P.A.; Goodchild, M.F.; Maguire, D.J; Rhind, D.W.: “*Geographical Information Systems: Principles, Techniques, Applications and Management*”, Wiley, 2nd Edition (1999)
- 64) Maclaren, H.: “*A Divide and Conquer Approach to Using Inductive Logic Programming for Learning User Models*”, PhD Thesis. The University of York, UK (2003)
- 65) Mayya, N.; Rajan, V. T.: “*Voronoi diagrams of polygons: A framework for shape representation*”. In the Journal of Mathematical Imaging and Vision, Volume 6, Issue 4, pp 355–378 (1996)
- 66) Meka, A., Singh, A. K.: “*Distributed Spatial Clustering in Sensor Networks*” in Proceedings of the 10th International Conference on Extending Database Technology (EDBT), p. 980-1000 (2006)
- 67) Miller, H. J.: “*Tobler's First Law and spatial analysis*”. Annals of the Association of American Geographers, 94, 284-289. (2004)

- 68) Mitchell, T. M.: *“Machine Learning”*. McGraw-Hill ISBN 0-07042-807-7 (1997)
- 69) Nedellec, C., Rouveirol, C., Ade, H., Bergadano, F., and Tausend, B.: *“Declarative bias in ILP”* In Advances in Inductive Logic Programming, Volume 32 of Frontiers in Artificial Intelligence and Applications, IOS Press, pp 82-103 (1996)
- 70) Ng, R.T., Han, J.: *“CLARANS: A Method for Clustering Entities for Spatial Data Mining”*. In Proceedings of TKDE, Volume 14, No. 5, pp 1003-1016 (2002)
- 71) Nienhuys-Cheng, S-H; Wolf, R.: *“Foundations of Inductive Logic Programming, Series”*. In Lecture Notes in Computer Science, Volume 1228 pp 404 (1997)
- 72) Nilsson, U; Małuszyński, J.: *“Logic, programming and Prolog, 2<sup>nd</sup> edition”*. Chichester, c1995. ISBN: 0-47195-996-0
- 73) Ohyama, T.: *“Some Voronoi diagrams that consider consumer behavior analysis”*. In Industrial Mathematics of the Japan Journal of Industrial and Applied Mathematics (2005)
- 74) Ohyama, T.: *“Application of the Additively Weighted Voronoi Diagram to Flow Analysis”*, The 2<sup>nd</sup> International Symposium on Voronoi Diagrams in Science and Engineering (2005)
- 75) Papadias, D., Theodoridis, Y.: *“Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures”*. International Journal of Geographic Information Science Volume 11, Issue 2, pp 111-138 (1997)
- 76) Patel, R., Swami, D. K., Pardasani, K. R.: *“Lattice Based Algorithm for Incremental Mining of Association Rules”*, International Journal of Theoretical and Applied Computer Sciences, Vol 1, No. 1, pp 119-128. (2006)
- 77) Pei, J., Zhang, X., Cho, M., Wang, H., Yu, P.S.: *“MaPle: A Fast Algorithm for Maximal Pattern-based Clustering”*. In Proceedings of ICDM (2003)
- 78) Perlich, C.; Provost, F.: *“Aggregation-Based Feature Invention and Relational Concept Classes”*. In Proceedings of KDD (2003)
- 79) Perlich, C. and Provost, F.: *“Distribution-Based Aggregation for Relational Learning from Identifier Attributes”* Machine Learning, Vol. 62, No. 1-2, pp. 65-105 (2006)
- 80) QHull - <http://www.qhull.org/>
- 81) Rakovic, S., Grieder, P., Jones, C.: *“Computation of Voronoi Diagrams and Delaunay Triangulation via Parametric Linear Programming”*, ETH Technical Report AUT04-03
- 82) Reddy, C., Tadepalli, P.: *“Inductive logic programming for speedup learning”*, In Proceedings of IJCAI workshop on Frontiers of ILP (1997)

- 83) Rengert, G.F. and Wasilchick, J. “*Suburban Burglary: A Time and Place for Everything*”. Springfield, IL: Charles C. Thomas (1985)
- 84) Reynolds, H. D.: “*The modifiable areal unit problem: empirical analysis by statistical simulation*”. PhD. Thesis, University of Toronto (1998)
- 85) Roddick, J. F., Lees, B. G.: “*Paradigms for Spatial and Spatio-Temporal Data Mining*”. Geographic Data Mining and Knowledge Discovery. Taylor and Francis. Research Monographs in Geographic Information Systems. (2001)
- 86) Santo, E.R., Lopes, D.R., Rangayyan, R.M.: “*Radial basis functions - simulated annealing classification of mammographic calcifications*”, Engineering in Medicine and Biology Society, 2004. IEMBS apos;04. 26th Annual International Conference of the IEEE, Volume 1, Issue , 1-5 Page(s):1644 – 1647 (2004)
- 87) Santos, M. Y.; Amaral L. A.: “*Geo-spatial data mining in the analysis of a demographic database*”. In Soft Computing - A Fusion of Foundations, Methodologies and Applications, Volume 9, Issue 5, pp 374-384 (2005)
- 88) Sahami, M.: “*Learning Classification Rules Using Lattices (Extended Abstract)*”, In Proceedings of the 8th European Conference on Machine Learning). N. Lavrac and S. Wrobel, Eds. Lecture Notes In Computer Science, vol. 912. Springer-Verlag, London, 343-346. (1995)
- 89) Scheffer, T.: “*Finding Association Rules That Trade Support Optimally against Confidence*”, Lecture Notes in Computer Science, Volume 2168, pp 424 (2001)
- 90) Schlossberg, M.: “*GIS, The US Census And Neighbourhood Scale Analysis*”, Planning, Practice & Research, Vol. 18, No. 2–3, pp 213-217, (2003)
- 91) Shekhar, S., Huang, Y.: “*Discovering Spatial Co-location Patterns: A Summary of Results*”, In Proceedings of SSTD (2001)
- 92) Shekhar, S., Lu, C.T., Zhang, P.: “*A unified approach to detection spatial outliers*”, GeoInformatica 7, pp 139-166 (2003)
- 93) Shekhar, S., Zhang, P., Huang, Y., Vatsavai, R.: “*Trends in Spatial Data Mining*”, as a Chapter to appear in *Data Mining: Next Generation Challenges and Future Directions*, H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha(eds.) (2003)
- 94) Stamatis, D. H.: “*Six Sigma and Beyond: Statistics and Probability, Volume III*”, St. Lucie Press (2004)
- 95) Tobler, W.R.: “*A computer movie simulating urban growth in the Detroit region*”. In Economic Geography, Volume 46, pp 234-240, (1970)
- 96) Valduriez, P.: “*Join indices*”, ACM Transactions on Database Systems, pp 218-246 (1990)

- 97) Valls, A., Torra, V., Domingo-Ferrer, J.: “*Aggregation methods to evaluate multiple protected versions of the same confidential data set*”, in *Soft Methods in Probability, Statistics and Data Analysis*, eds. P. Grzegorzewski, O. Hryniewicz and M. A. Gil, Series “*Advances in Soft Computing*”, Heidelberg: Physica-Verlag, pp 355-362 (2002)
- 98) Vens, C.; Van Assche, A.; Blockeel, H.; Dzeroski, S.: “*First order random forests with complex aggregates*”. In *Proceedings of ILP* (2004)
- 99) Xiong, H.; Shekhar, S.; Huang, Y.; Kumar, V.: “*A Framework for Discovering Co-location Patterns in Data Sets with Extended Spatial Entities*”, In *Proceedings of SDM* (2004)
- 100) Yin, X.; Han J.; Yang J.; Yu, P.S.: “*CrossMine: Efficient Classification Across Multiple Database Relations*”. In *Proceedings of ICDE* (2004)
- 101) Yin, X.; Han, J.; Yang, J.: “*Efficient Multi-relational Classification by Tuple ID Propagation*”. In the *Workshop on Multi-relational Data Mining in with KDD* (2003)
- 102) Zeitouni, K., Yeh, L., Aufaure, M-A.: “*Join indices as a tool for spatial data mining*”, *International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining, Lecture Notes in Artificial Intelligence n° 2007*, Springer, pp 102-114 (2000)
- 103) Železný, F., Lavrac, Nada,: “*Propositionalization-based relational subgroup discovery with RSD*” *Machine Learning, 2006. VI 62* (2006)
- 104) Železný, F., Srinivasan, A. and David, C. P. Jr.: “*Randomised restarted search in ILP*”, *Machine Learning*, v.64 n.1-3, p.183-208, September 2006
- 105) Zelle, J.M., Mooney, R.J., Konvisser, J.B.: “*Combining Top-down and Bottom-up Techniques in Inductive Logic Programming*” in *Proceedings of Machine Learning*, pp 343-351 (1994)
- 106) Zhang, J., Samal, A. and Soh L.: “*Polygon-based Spatial Clustering*”. In *Proceedings of GeoComputation* (2005)
- 107) Zhang, X., Mamoulis, N., Cheung, D. W., Shou, Y.: “*Fast mining of spatial collocations.*” In *Proceedings of SIGKDD* (2004)s of SIGKDD (2004)